



Cisco Secure Workload User Guide SaaS, Release 3.9

First Published: 2023-12-21

Americas Headquarters

Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA 95134-1706
USA
<http://www.cisco.com>
Tel: 408 526-4000
800 553-NETS (6387)
Fax: 408 527-0883

THE SPECIFICATIONS AND INFORMATION REGARDING THE PRODUCTS IN THIS MANUAL ARE SUBJECT TO CHANGE WITHOUT NOTICE. ALL STATEMENTS, INFORMATION, AND RECOMMENDATIONS IN THIS MANUAL ARE BELIEVED TO BE ACCURATE BUT ARE PRESENTED WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED. USERS MUST TAKE FULL RESPONSIBILITY FOR THEIR APPLICATION OF ANY PRODUCTS.

THE SOFTWARE LICENSE AND LIMITED WARRANTY FOR THE ACCOMPANYING PRODUCT ARE SET FORTH IN THE INFORMATION PACKET THAT SHIPPED WITH THE PRODUCT AND ARE INCORPORATED HEREIN BY THIS REFERENCE. IF YOU ARE UNABLE TO LOCATE THE SOFTWARE LICENSE OR LIMITED WARRANTY, CONTACT YOUR CISCO REPRESENTATIVE FOR A COPY.

The Cisco implementation of TCP header compression is an adaptation of a program developed by the University of California, Berkeley (UCB) as part of UCB's public domain version of the UNIX operating system. All rights reserved. Copyright © 1981, Regents of the University of California.

NOTWITHSTANDING ANY OTHER WARRANTY HEREIN, ALL DOCUMENT FILES AND SOFTWARE OF THESE SUPPLIERS ARE PROVIDED "AS IS" WITH ALL FAULTS. CISCO AND THE ABOVE-NAMED SUPPLIERS DISCLAIM ALL WARRANTIES, EXPRESSED OR IMPLIED, INCLUDING, WITHOUT LIMITATION, THOSE OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NON-INFRINGEMENT OR ARISING FROM A COURSE OF DEALING, USAGE, OR TRADE PRACTICE.

IN NO EVENT SHALL CISCO OR ITS SUPPLIERS BE LIABLE FOR ANY INDIRECT, SPECIAL, CONSEQUENTIAL, OR INCIDENTAL DAMAGES, INCLUDING, WITHOUT LIMITATION, LOST PROFITS OR LOSS OR DAMAGE TO DATA ARISING OUT OF THE USE OR INABILITY TO USE THIS MANUAL, EVEN IF CISCO OR ITS SUPPLIERS HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

Any Internet Protocol (IP) addresses and phone numbers used in this document are not intended to be actual addresses and phone numbers. Any examples, command display output, network topology diagrams, and other figures included in the document are shown for illustrative purposes only. Any use of actual IP addresses or phone numbers in illustrative content is unintentional and coincidental.

All printed copies and duplicate soft copies of this document are considered uncontrolled. See the current online version for the latest version.

Cisco has more than 200 offices worldwide. Addresses and phone numbers are listed on the Cisco website at www.cisco.com/go/offices.

Cisco and the Cisco logo are trademarks or registered trademarks of Cisco and/or its affiliates in the U.S. and other countries. To view a list of Cisco trademarks, go to this URL: <https://www.cisco.com/c/en/us/about/legal/trademarks.html>. Third-party trademarks mentioned are the property of their respective owners. The use of the word partner does not imply a partnership relationship between Cisco and any other company. (1721R)

© 2023–2024 Cisco Systems, Inc. All rights reserved.



CONTENTS

CHAPTER 1

Get Started with Cisco Secure Workload 1

Supported Web Browsers 1

Quick Start Wizard 2

Get Started with Segmentation and Microsegmentation 2

General Process for Implementing Microsegmentation 2

Set Up Microsegmentation for Workloads Running on Bare Metal or Virtual Machines 5

Set Up Microsegmentation for Cloud-Based Workloads 6

Set Up Microsegmentation for Kubernetes-Based Workloads 7

CHAPTER 2

Deploy Software Agents on Workloads 9

Deploy Software Agents 9

Supported Platforms and Requirements 10

Installing Linux Agents for Deep Visibility and Enforcement 10

Requirements and Prerequisites to Install Linux Agents 10

Supported Methods to Install Linux Agents 10

Verify Linux Agent Installation 16

Installing Windows Agents for Deep Visibility and Enforcement 16

Requirements and Prerequisites for Installing Windows Agent 16

Supported Methods to Install Windows Agents 17

Verify Windows Agent Installation 21

Verify Windows Agent in the Configured Service User Context 21

Modify Service Account 22

Deploying Agents on a VDI Instance or VM Template (Windows) 23

Windows Agent Installer and Npcap—For Windows 2008 R2 24

Windows Agent Flow Captures: For All Windows OS Excluding Windows Server 2008 R2 25

Installing AIX Agents for Deep Visibility and Enforcement 25

- Requirements and Prerequisites for Installing AIX Agents 25
- Install AIX Agent using the Agent Script Installer Method 26
- Verify AIX Agent Installation 29
- Installing Kubernetes or OpenShift Agents for Deep Visibility and Enforcement 29
 - Requirements and Prerequisites 29
 - Install Kubernetes or OpenShift Agent using the Agent Script Installer Method 31
- Installing Solaris Agents for Deep Visibility 32
 - Requirements and Prerequisites for Installing Solaris Agents 32
 - Install Solaris Agent using the Agent Script Installer Method 32
 - Verify Solaris Agent Installation 34
- (Manual Installations Only) Update the User Configuration File 34
- Other Agent-Like Tools 35
- Connectivity Information 35
- Security Exclusions 37
- Service Management of Agents 39
 - Service Management for RHEL, CentOS, OracleLinux-6.x, and Ubuntu-14 39
 - Service Management for RHEL, CentOS, OracleLinux-7.x and Later 39
 - Service Management for Windows Server or Windows VDI 39
 - Service Management for AIX 40
 - Service Management for Kubernetes Agent Installations 40
 - Service Management for Solaris 40
- Policy Enforcement with Agents 41
 - Agent Enforcement on the Linux Platform 41
 - Linux iptables or ip6tables 41
 - Caveats 42
 - Agent Enforcement on the Windows Platform in WAF mode 43
 - Windows Firewall with Advanced Security 43
 - Secure Workload Rules and the Windows Firewall 43
 - Security Profiles 43
 - Effective Setting and Mixed-List Policies 44
 - Stateful Enforcement 44
 - Caveats 45
 - Agent Enforcement on the Windows Platform in WFP Mode 45
 - Windows Filtering Platform 45

| | |
|---|----|
| Advantages of WFP over WAF | 45 |
| Agent Support for WFP | 46 |
| Agent WFP support and Windows Firewall | 46 |
| Effective Setting and Mixed-List Policies | 47 |
| Stateful Enforcement | 47 |
| Visibility of Configured WFP Filters | 47 |
| Disable Stealth Mode Filters in WFP Mode | 47 |
| Delete Configured WFP Filters | 48 |
| Known Limitations in WFP Mode | 48 |
| Configure Policies for Windows Attributes | 48 |
| Recommended Windows OS-Based Policy Configuration | 51 |
| Known limitations | 51 |
| Caveats | 52 |
| Verify and Troubleshoot Policies with Windows OS-Based Filtering Attributes | 52 |
| Enforcement of Kubernetes Pods on Windows Nodes | 60 |
| Agent Enforcement on AIX Platform | 62 |
| IPFilter | 62 |
| Caveats | 63 |
| Known Limitations | 63 |
| Check Agent Status and Statistics | 63 |
| View Agent Details | 64 |
| Software Agent Config | 65 |
| Requirements and Prerequisites for Configuring Software Agents | 65 |
| User Roles and Access to Agent Configuration | 65 |
| Configure Software Agents | 66 |
| Create an Agent Configuration Profile | 67 |
| Creating an Agent Config Intent | 72 |
| Creating a Remote VRF configuration for agents | 73 |
| Create an Interface Configuration Intent | 73 |
| View Detailed Agent Status in the Workload Profile | 74 |
| Rehoming of Agents | 76 |
| Enable Rehoming | 76 |
| Select Agents to Rehome | 78 |
| Disable Rehoming | 78 |

| | |
|--|----|
| Generate Agent Token | 79 |
| Host IP Address Change when Enforcement is Enabled | 80 |
| Upgrading Software Agents | 81 |
| Upgrade Agents from UI | 81 |
| Manual Agent Upgrade | 84 |
| Upgrade Behaviour of Kubernetes/Openshift Agent | 84 |
| Removing Software Agents | 85 |
| Remove a Deep Visibility or Enforcement Linux Agent | 85 |
| Removing a Deep Visibility/Enforcement Windows Agent | 86 |
| Remove a Deep Visibility or Enforcement AIX Agent | 87 |
| Remove Universal Linux Agent | 87 |
| Remove Universal Windows Agent | 87 |
| Remove an Enforcement Kubernetes or OpenShift Agent | 88 |
| Remove a Deep Visibility Solaris Agent | 88 |
| Data collected and exported by workload agents | 88 |
| Registration | 88 |
| Agent upgrade | 89 |
| Config server | 89 |
| Network Flow Information | 89 |
| Machine information | 90 |
| Agent statistics | 91 |
| Enforcement Alerts | 91 |
| Enforcement UI Alerts Details | 93 |
| Enforcement Alert Details | 93 |
| Example of alert_details for an enforcement alert | 94 |
| Sensor Alerts | 94 |
| Sensor UI Alerts Details | 96 |
| Sensor Alert Details | 96 |
| Example of alert_details for a sensor alert | 97 |
| Frequently Asked Questions | 97 |
| General | 97 |
| Agent deployment | 97 |
| Linux | 97 |
| Windows | 98 |

| | |
|--|-----|
| Kubernetes | 99 |
| Anomaly Types | 101 |
| Agent Inactivity | 101 |
| Upgrade Failure | 101 |
| Convert Failed | 102 |
| Convert Capability | 102 |
| Policy Out of Sync | 102 |
| Flow Export: Pcap Open | 103 |
| Flow Export: HTTPS Connectivity | 103 |
| Certificate Issues | 103 |
| Windows | 103 |
| Certificate Issues for NPCAP installer | 105 |
| Windows Host Rename | 107 |
| Check If Platform Is Currently Supported | 108 |
| Windows | 108 |
| Linux | 108 |
| AIX | 108 |
| Windows Installer Issues | 108 |
| Required Windows Services | 109 |
| Npcap Issues | 109 |
| Npcap will not upgrade (manually or via agent) | 109 |
| Npcap will not install | 109 |
| Verify if Npcap is fully installed | 110 |
| Network Connectivity issues during NPCAP installation or upgrade | 111 |
| NIC teaming compatibility issues with NPCAP | 112 |
| VDI instance VM does not report network flows | 112 |
| Network Performance with NPCAP | 112 |
| OS Performance and/or stability Issues | 113 |
| GPO Configurations | 114 |
| Agent To Cluster Communications | 114 |
| Types of connections | 115 |
| Checking the connection state | 115 |
| SSL Troubleshooting | 116 |
| Agent Communications Overview | 116 |

| | |
|---|-----|
| Configuring IP traffic for Agent Communications | 116 |
| Troubleshooting SSL/TLS Connections | 117 |
| Agent operations | 119 |

CHAPTER 3**External Orchestrators in Secure Workload 123**

| | |
|---|-----|
| Navigate to the External Orchestrators Page | 123 |
| List of External Orchestrators | 124 |
| Create External Orchestrator | 126 |
| Edit External Orchestrator | 129 |
| Delete External Orchestrator | 130 |
| Orchestrator generated labels | 130 |
| Secure Connector | 130 |
| Technical Details | 131 |
| Requirements for Secure Connector Client | 132 |
| Secure Connector Client Deployment | 132 |
| Proxy Support | 132 |
| Deployment Overview | 132 |
| Deploy the Secure Connector Client | 132 |
| [Optional] Deploy Specific Version of Secure Connector Client | 133 |
| Verify Secure Connector Client State | 135 |
| Secure Connector Client Status | 135 |
| Secure Connector Alerts | 136 |
| Upgrade Secure Connector Client | 138 |
| Uninstall Secure Connector Client | 138 |
| Amazon Web Services | 138 |
| Prerequisites | 138 |
| Configuration fields | 138 |
| Workflow | 139 |
| Orchestrator generated labels | 139 |
| Instance-specific labels | 140 |
| Troubleshooting | 140 |
| Kubernetes/OpenShift | 140 |
| Requirements and Prerequisites | 141 |
| Configuration Fields | 141 |

| | |
|---|-----|
| Orchestrator Golden Rules | 143 |
| Workflow | 144 |
| Kubernetes Role-Based Access Control (RBAC) Resource Considerations | 144 |
| Orchestrator-generated labels | 147 |
| Troubleshooting | 147 |
| VMware vCenter | 148 |
| Prerequisites | 148 |
| Configuration fields | 148 |
| Workflow | 149 |
| Orchestrator generated labels | 149 |
| Instance-specific labels | 149 |
| Caveats | 149 |
| Troubleshooting | 150 |
| DNS | 150 |
| Prerequisites | 150 |
| Configuration fields | 150 |
| Workflow | 150 |
| Generated labels | 151 |
| Caveats | 151 |
| Troubleshooting | 151 |
| Behavior of Full/Delta polling for DNS Orchestrators | 152 |
| Unsupported Features | 152 |
| Infoblox | 152 |
| Prerequisites | 153 |
| Configuration fields | 153 |
| Workflow | 153 |
| Orchestrator generated labels | 154 |
| Generated labels | 154 |
| Caveats | 154 |
| Troubleshooting | 154 |
| F5 BIG-IP | 155 |
| Prerequisites | 155 |
| Configuration fields | 155 |
| Workflow | 156 |

| | |
|--|--|
| Orchestrator generated labels | 156 |
| Generated labels | 157 |
| Policy enforcement for F5 BIG-IP | 157 |
| Policy Enforcement for F5 Ingress Controller | 158 |
| Caveats | 161 |
| Troubleshooting | 161 |
| Citrix Netscaler | 161 |
| Prerequisites | 161 |
| Configuration fields | 161 |
| Workflow | 162 |
| Orchestrator generated labels | 162 |
| Generated labels | 163 |
| Policy enforcement for Citrix Netscaler | 163 |
| Caveats | 164 |
| Troubleshooting | 164 |
| TAXII | 165 |
| Prerequisites | 165 |
| Configuration fields | 165 |
| Workflow | 166 |
| Generated labels | 166 |
| Caveats | 167 |
| Troubleshooting | 167 |
| Behavior of Full polling for TAXII Orchestrators | 167 |
| <hr/> | |
| CHAPTER 4 | Configure and Manage Connectors for Secure Workload |
| | 169 |
| What are Connectors | 169 |
| Connectors for Flow Ingestion | 170 |
| NetFlow Connector | 170 |
| F5 Connector | 177 |
| NetScaler Connector | 181 |
| Cisco Secure Firewall Connector | 185 |
| Meraki Connector | 190 |
| ERSPAN Connector | 194 |
| Connectors for Endpoints | 198 |

| | |
|---|-----|
| AnyConnect Connector | 198 |
| ISE Connector | 203 |
| Connectors for Inventory Enrichment | 211 |
| ServiceNow Connector | 212 |
| How to Configure the ServiceNow Connector | 212 |
| ServiceNow Instance Configuration | 213 |
| Processing ServiceNow records | 216 |
| Sync Interval Configuration | 216 |
| Explore Command to Delete the Labels | 217 |
| Finding VRF ID for a Tenant | 217 |
| Getting to Explore Command UI | 218 |
| Running the Commands | 218 |
| Frequently Asked Questions | 218 |
| Limits | 219 |
| Connectors for Alert Notifications | 219 |
| Syslog Connector | 219 |
| Email Connector | 222 |
| Slack Connector | 225 |
| PagerDuty Connector | 227 |
| Kinesis Connector | 228 |
| Cloud Connectors | 230 |
| AWS Connector | 231 |
| Azure Connector | 244 |
| GCP Connector | 252 |
| Identity Connectors | 261 |
| Configure an OpenLDAP Connector | 261 |
| Configuration | 261 |
| Inventory | 263 |
| Event Log | 264 |
| Advanced Settings | 265 |
| Connector Alerts | 265 |
| Alert Configuration | 265 |
| Alert Type | 266 |
| Appliance/Connector down | 266 |

| | |
|---|-----|
| Appliance/Connector system usage | 267 |
| Connector Configuration Error | 268 |
| Connector UI Alert Details | 269 |
| Alert Details | 269 |
| Example of Alert Details | 269 |
| Connector UI Alert Details | 270 |
| Life Cycle Management of Connectors | 270 |
| Enabling a Connector | 270 |
| Viewing Connector-Related Information | 273 |
| Deleting a Connector | 274 |
| Monitoring a Connector | 275 |
| Virtual Appliances for Connectors | 275 |
| Types of Virtual Appliances | 275 |
| Secure Workload Ingest | 275 |
| Secure Workload Edge | 277 |
| Deploying a Virtual Appliance | 279 |
| Decommissioning a Virtual Appliance | 285 |
| Monitoring a Virtual Appliance | 285 |
| Security Considerations | 286 |
| Configuration Management on Connectors and Virtual Appliances | 286 |
| Test and Apply | 286 |
| NTP Configuration | 286 |
| Log Configuration | 289 |
| Endpoint Configuration | 290 |
| Slack Notifier Configuration | 291 |
| PagerDuty Notifier Configuration | 291 |
| Kinesis Notifier Configuration | 292 |
| Email Notifier Configuration | 292 |
| Syslog Notifier Configuration | 293 |
| Syslog Severity Mapping Configuration | 293 |
| ISE Instance Configuration | 294 |
| Discovery | 294 |
| LDAP Configuration | 295 |
| Remove | 301 |

| | |
|---|---|
| Troubleshooting | 301 |
| Allowed set of commands | 301 |
| Show Logs | 302 |
| Show Service Logs | 302 |
| Show Running Configuration | 303 |
| Show Service Running Configuration | 304 |
| Show System Commands | 305 |
| Show Docker Commands | 307 |
| Show Docker Instance Commands | 308 |
| Show Supervisor Commands | 310 |
| Show Supervisor Service Commands | 311 |
| Network Connectivity Commands | 312 |
| List Files | 313 |
| List Service Files | 314 |
| Packet Capture | 315 |
| Update Listening Ports of Connectors | 316 |
| Update Alert Notifier Connector Log Configuration | 318 |
| Collect Snapshot From Appliance | 319 |
| Collect Snapshot From Connector | 320 |
| Collect Controller Profile | 321 |
| Collect Connector Profile | 322 |
| Override connector alert interval for Appliance | 323 |
| Override connector alert interval for Connector | 324 |
| Hawkeye Dashboards | 325 |
| Appliance Controller Dashboard | 325 |
| Service Dashboard | 326 |
| AnyConnect Service Dashboard | 326 |
| Appliance and Service DIO Dashboard | 327 |
| General Troubleshooting Guidelines | 328 |
| Log Files | 329 |
| Cisco Secure Firewall Management Center | 330 |
| | |
| CHAPTER 5 | Manage Inventory for Secure Workload 333 |
| | Workload Labels 334 |

- Importance of Labels **335**
- Subnet-based Label Inheritance **335**
- Label Prefixes **335**
 - Labels Generated by Cloud Connectors **336**
 - Labels Related to Kubernetes Clusters **337**
- Importing Custom Labels **340**
 - Guidelines for Uploading Label Files **340**
 - Label Key Schema **340**
 - Upload Custom Labels **341**
 - Search Labels **342**
 - Manually Assign or Edit Custom Labels **342**
 - Download Labels **343**
 - Change Labels **343**
- Disable Labels **343**
- Review Label Change Impact **344**
- Delete Labels **345**
- View Labels Usage **346**
- Create a Process for Maintaining Labels **346**
- Scopes and Inventory **347**
 - Scopes **348**
 - Scope Filter **350**
 - Full Scope Queries **352**
 - Providing Access to Scopes **352**
 - Viewing Scope **353**
 - Searching for flows referencing a scope **353**
 - Creating a New Scope **354**
 - Scope Overlap **356**
 - Editing Scopes **357**
 - Delete a Scope **360**
 - Reset the Scope Tree **362**
 - Commit Changes **363**
 - Change Log **364**
 - Creating a New Tenant **365**
 - Inventory **366**

| | |
|--------------------------------------|-----|
| Searching Inventory | 367 |
| Suggest Child Scopes | 370 |
| Steps to perform scope suggestion | 372 |
| Filters | 377 |
| Create an Inventory Filter | 378 |
| Review Filter Change Impact | 379 |
| Create a Domain Filter | 380 |
| Restrict to Ownership Scope | 381 |
| Review Scope/Filter Change Impact | 382 |
| Scope Query Change Impact Modal | 383 |
| Membership Changes | 383 |
| Dependencies | 384 |
| Filter Query Change Impact Modal | 385 |
| Membership Changes | 386 |
| Dependencies | 386 |
| Inventory Profile | 387 |
| Workload Profile | 388 |
| Labels and Scopes Tab | 388 |
| Agent Health Tab | 389 |
| Process List Tab | 391 |
| Process Snapshot Tab | 392 |
| Interfaces Tab | 393 |
| Software Packages Tab | 393 |
| Vulnerabilities Tab | 394 |
| Agent Configuration Tab | 395 |
| Agent Statistics Tab | 396 |
| Concrete Policies Tab | 397 |
| Container Policies Tab | 398 |
| Network Anomalies Tab | 399 |
| File Hashes Tab | 399 |
| Software Packages | 400 |
| Packages Tab | 400 |
| Common Vulnerabilities and Exposures | 401 |
| Windows Packages and CVEs | 401 |

| | |
|---|-----|
| Inventory Filters | 402 |
| Vulnerability Data Visibility | 403 |
| Workload Profile Page | 403 |
| Packages Tab | 403 |
| Process List Tab | 404 |
| Process Snapshot Tab | 405 |
| Vulnerabilities Tab | 405 |
| Inventory Filters | 406 |
| CVE ID Based Filter | 406 |
| Common Vulnerability Scoring System Impact Score Based Filter | 407 |
| CVSS V2 Attributes Based Filters | 408 |
| CVSS V3 Attributes Based Filters | 409 |
| Cisco Security Risk Score-Based Filter | 410 |
| Cisco Security Risk Score Attributes-Based Filters | 411 |
| Malicious Inventory-Based Filter | 412 |
| Service Profile | 412 |
| Pod Profile | 413 |
| Container Vulnerability Scanning | 414 |

CHAPTER 6

| | |
|--|------------|
| Manage Policies Lifecycle in Secure Workload | 417 |
| Segmentation Policy Basics | 417 |
| Use Workspaces to Manage Policies | 418 |
| Working with Policies: Navigating to the Workspaces Page | 418 |
| Create a Workspace | 419 |
| Primary and Secondary Workspaces | 419 |
| Rename a Workspace | 420 |
| View Workloads in a Scope | 420 |
| Search Within a Workspace | 420 |
| Deleting Workspaces | 423 |
| About Policies | 425 |
| Policy Attributes | 425 |
| Policy Rank: Absolute, Default, and Catch-All | 425 |
| Policy Inheritance and the Scope Tree | 426 |
| About Consumer and Provider in Policies | 426 |

| | |
|--|-----|
| Policy Example | 427 |
| Create and Discover Policies | 427 |
| Best Practices for Creating Policies | 427 |
| Manually Create Policies | 428 |
| If the Add Policy Button Is Not Available | 429 |
| Policies for Specific Purposes | 430 |
| Create InfoSec Policies to Block Traffic from Outside Your Network | 430 |
| Create Policies to Address Immediate Threats | 431 |
| Create a Policy to Quarantine Vulnerable Workloads | 431 |
| Policy Templates | 432 |
| System-Defined Policy Templates | 432 |
| Create Custom Policy Templates | 432 |
| Applying a Template | 435 |
| Discover Policies Automatically | 436 |
| Policy Discovery Details | 437 |
| How to Automatically Discover Policies | 437 |
| Discover Policies for One Scope or for a Branch of the Scope Tree | 439 |
| Verify the Workloads That Policy Discovery Will Apply To | 441 |
| Automatically Discover Policies | 443 |
| Stop Automatic Policy Discovery in Progress | 444 |
| Advanced Features of Automatic Policy Discovery | 445 |
| Approve Policies | 463 |
| Iteratively Revise Policies | 464 |
| View, Compare, and Manage Discovered Policy Versions | 466 |
| Policy Discovery Kubernetes Support | 469 |
| Import/Export | 471 |
| Export a Workspace | 471 |
| Import | 471 |
| Platform-Specific Policies | 472 |
| Windows | 472 |
| Kubernetes and OpenShift | 484 |
| Grouping Workloads: Clusters and Inventory Filters | 487 |
| Clusters | 488 |
| Cluster Confidence | 489 |

| | |
|--|-----|
| View Clusters | 490 |
| Making Changes to Clusters | 491 |
| Convert a Cluster to an Inventory Filter | 493 |
| Creating or Deleting Clusters | 493 |
| Comparing Versions of Generated Clusters: Diff Views | 494 |
| Preventing Cluster Modification During Automatic Policy Discovery Reruns | 496 |
| Approving Clusters | 496 |
| Address Policy Complexities | 498 |
| Policy Priorities | 498 |
| Policy Global Ordering and Conflict Resolution | 499 |
| Validate the Order and Priority of Policies | 502 |
| (Advanced) Change Policy Priorities | 502 |
| When Consumer and Provider Are in Different Scopes: Policy Options | 504 |
| (Advanced) Create Cross-Scope Policies | 505 |
| Troubleshoot Cross-Scope Policies | 516 |
| Effective Consumer or Effective Provider | 517 |
| About Deleting Policies | 520 |
| Review and Analyze Policies | 520 |
| Review Automatically Discovered Policies | 520 |
| Address Low-Confidence Policies | 522 |
| Troubleshoot Automatic Policy Discovery Results | 522 |
| Policy Visual Representation | 523 |
| Quick Analysis | 525 |
| Live Policy Analysis | 527 |
| Start Live Policy Analysis | 528 |
| Stop Live Policy Analysis | 529 |
| Policy Analysis Results: Understand the Basics | 529 |
| Example: Impact of Policies Analyzed in Other Scopes | 530 |
| Policy Analysis Details | 530 |
| Suggested Steps for Investigating Flows | 532 |
| Run Policy Experiments to Test Current Policies Against Past Traffic | 534 |
| After Changing Policies, Analyze Latest Policies | 535 |
| Policy Label Flags | 536 |
| View, Compare, and Manage Analyzed Policy Versions | 536 |

| | |
|---|-----|
| Activity Logs of Policy Analysis | 537 |
| Enforce Policies | 538 |
| Check Agent Health and Readiness to Enforce | 539 |
| Enable Policy Enforcement | 540 |
| Policy Enforcement Wizard | 544 |
| Enforcement on Containers | 546 |
| Verify That Enforcement Is Working as Expected | 547 |
| View Enforced Policies for a Specific Workload (Concrete Policies) | 548 |
| Verify That Enforcement Is Enabled for Agents | 549 |
| Verify That Enforced Policies Are Being Pushed to Agents | 549 |
| If There Are Too Many Policies for the Agent | 550 |
| Modify Enforced Policies | 551 |
| Enforce New and Revised Policies | 551 |
| View, Compare, and Manage Enforced Policy Versions | 551 |
| Revert Enforced Policies to an Earlier Version | 552 |
| Disable Policy Enforcement | 552 |
| Enforcement History | 553 |
| About Policy Versions (v* and p*) | 553 |
| Comparison of Policy Versions: Policy Diff | 556 |
| Activity Logs and Version History | 558 |
| Automatic Deletion of Old Policy Versions | 559 |
| Conversations | 559 |
| Conversations Table View | 559 |
| Choosing Consumer or Provider | 560 |
| Conversation Filters | 560 |
| Explore Observations | 562 |
| Conversation Observation Hovered | 562 |
| Filtering | 563 |
| Conversations Chart View | 563 |
| Top Consumers/Providers of Conversations | 564 |
| Automated Load Balancer Config for Automatic Policy Discovery (F5 Only) | 566 |
| Terminology | 567 |
| Deployment | 567 |
| Clusters | 568 |

| | |
|---|-----|
| Policies | 569 |
| Caveats | 571 |
| Policies Publisher | 571 |
| Prerequisites | 571 |
| Getting Kafka Client Certificates | 571 |
| Protobuf Definition File | 572 |
| Data Model of Secure Workload Network Policy | 573 |
| Reference Implementation of Secure Workload Network Policies Client | 573 |

CHAPTER 7
Configure and Monitor Forensic Events 575

| | |
|---------------------------------|-----|
| Compatibility | 575 |
| Forensics Signals | 576 |
| Privilege Escalation | 577 |
| User Log on | 577 |
| User Log on Failed | 578 |
| Shellcode | 578 |
| File Access | 578 |
| User Account | 578 |
| Unseen Command | 578 |
| Unseen Library | 579 |
| Raw Socket Creation | 579 |
| Binary Changed | 579 |
| Library Changed | 579 |
| Side Channel | 580 |
| Follow User Logon | 580 |
| Follow Process | 580 |
| Forensic Configuration | 581 |
| Forensic Rules | 581 |
| Adding a Forensic Rule | 581 |
| Basic Forensic Rule Composition | 582 |
| Default Secure Workload Rules | 582 |
| Default MITRE ATT&CK Rules | 584 |
| Forensic profiles | 590 |
| Add a Profile | 590 |

| | |
|---|-----|
| Edit a Profile | 591 |
| Clone a Profile | 591 |
| Default Profile - Secure Workload Profile | 592 |
| Default Profile - MITRE ATT&CK Profile | 592 |
| Forensic visualization | 593 |
| Accessing Forensic Page | 593 |
| Browsing Forensic Events | 596 |
| Inspecting a Forensic Event | 596 |
| Fields Displayed in Forensic Events | 597 |
| Common Fields | 597 |
| Process Info | 598 |
| Privilege Escalation | 598 |
| User Logon | 598 |
| User Logon Failed | 599 |
| Shellcode | 600 |
| File Access | 600 |
| User Account | 600 |
| Unseen Command | 601 |
| Unseen Library | 601 |
| Raw Socket Creation | 601 |
| Library Changed | 602 |
| Side Channel | 602 |
| Follow User Logon | 602 |
| Follow Process | 602 |
| Network Anomaly | 602 |
| Forensic Analysis - Searchable Fields | 602 |
| Miscellaneous Fields | 603 |
| Search Terms in Forensic Analysis | 603 |
| Common Fields | 603 |
| Binary Changed | 603 |
| File Access | 603 |
| Follow Process | 604 |
| Follow User Logon | 604 |
| Ldap | 604 |

| | |
|---|-----|
| Library Changed | 605 |
| Privilege Escalation | 605 |
| Process Info | 605 |
| Raw Socket | 605 |
| Shellcode | 606 |
| Side Channel | 606 |
| Unseen Command | 606 |
| Unseen Library | 607 |
| User Account | 607 |
| User Logon | 607 |
| User Logon Failed | 608 |
| Forensics alerts | 609 |
| Accessing Forensic Alerts | 609 |
| Checking Alert Details | 609 |
| External Integration | 610 |
| Forensics Score | 611 |
| Where to See Forensic Score | 611 |
| How the Forensic Score is Calculated | 612 |
| How to Improve Forensic Score | 613 |
| Caveats | 613 |
| PCR-Based Network Anomaly Detection | 613 |
| Forensic Rules for Network Anomaly Events | 614 |
| Rule Attributes | 614 |
| Rule Actions | 616 |
| Where to See Network Anomaly Events | 617 |
| Rule Severities and Network Anomaly Scores | 619 |
| PCR Data and Network Anomaly Events Retention | 619 |
| Network Anomaly Latency | 619 |
| Caveats | 620 |
| Process Hash Anomaly Detection | 620 |
| How to Enable Process Hash Feature | 621 |
| Where to See Process Hash Score | 621 |
| How the Process Hash Score is Calculated | 621 |
| How to Improve Process Hash Score | 623 |

Threat Info Details 623

Caveats 623

CHAPTER 8

Network Flows - Traffic Visibility 625

Corpus Selector 627

Columns and Filters 627

Filtered Time series 633

Top N Charts 634

Observations List 635

Flow Details 636

Explore Observations 637

Client-Server Classification 639

Sensor Type Recommendation 641

Identifying Producers (aka Servers) and Consumers (aka Clients) for a flow 642

Conversation Mode 642

Visibility in Proxied Flows 643

Visibility of Well-Known Malicious IPv4 Addresses 645

CHAPTER 9

Configure Alerts 647

Alert Types and Publishers 649

Create Alerts 650

Alert Configuration Modal 652

Summary Alerts 653

Summarization Versus Snoozing 654

Secure Workload Alerts Notifier (TAN) 654

Configure Notifiers 655

Choose Alert Publishers 655

External Syslog Tunneling Moves to TAN 656

Connection Chart 656

View Alerts Trigger Rules 657

Alerts Trigger Rules Details 657

Generate Test Alerts 658

Current Alerts 660

Snooze Alerts 661

| | |
|----------------------------------|-----|
| Alert Details | 662 |
| Common Alert Structure | 662 |
| General Alert Format by Notifier | 664 |
| Kafka (DataTaps) | 664 |
| Email | 664 |
| PagerDuty | 665 |
| Syslog | 666 |
| Slack | 666 |
| Kinesis | 666 |

| | | |
|-------------------|--|------------|
| CHAPTER 10 | Monitor Configurations in Secure Workload | 669 |
| | Agent Monitoring | 669 |
| | Agent Monitoring Type | 669 |
| | Agent Status and Statistics | 671 |
| | Enforcement Status | 673 |
| | Enforcement Status for Cloud Connectors | 674 |
| | Pause Policy Updates | 675 |

| | | |
|-------------------|--------------------------------|------------|
| CHAPTER 11 | View Security Dashboard | 677 |
| | View the Security Dashboard | 677 |
| | Security Score | 678 |
| | Security Score Categories | 678 |
| | High-Level View | 678 |
| | Scope Level Score Details | 678 |
| | Overall Score | 679 |
| | Daily Time Series | 680 |
| | Score Breakdown | 681 |
| | Score Details | 681 |
| | Vulnerability Security Score | 683 |
| | Process Hash Score | 684 |
| | Attack Surface Score | 686 |
| | Forensics Score | 690 |
| | Network Anomaly Score | 692 |
| | Segmentation Compliance Score | 694 |

| | | |
|-------------------|-------------------------------------|------------|
| CHAPTER 12 | View Vulnerability Dashboard | 697 |
| | Vulnerability Dashboard | 699 |
| | CVEs Tab | 701 |
| | Packages Tab | 702 |
| | Workloads Tab | 703 |
| | Pods Tab | 705 |

| | | |
|-------------------|---------------------------------|------------|
| CHAPTER 13 | View Reporting Dashboard | 707 |
| | Reporting Dashboard | 707 |
| | Schedule Email Reports | 707 |
| | Overview | 708 |
| | Operation | 713 |
| | Compliance | 716 |

| | | |
|-------------------|--|------------|
| CHAPTER 14 | Set up System Configurations in Secure Workload | 719 |
| | Roles | 719 |
| | Abilities and Capabilities | 720 |
| | Menu Access by Role | 721 |
| | Create a Role | 723 |
| | Edit a Role | 725 |
| | Change Log | 726 |
| | Collection Rules | 727 |
| | Rules | 727 |
| | Priority | 727 |
| | Session Configuration | 728 |
| | Idle Session | 728 |
| | Preferences | 729 |
| | Change Your Landing Page Preference | 729 |
| | Changing a Password | 729 |
| | Recovering Passwords | 729 |
| | Enabling Two-Factor Authentication | 730 |
| | Disabling Two-Factor Authentication | 731 |
| | Scopes | 732 |

| | |
|---|-------------------------------------|
| Users | 732 |
| Add a User | 732 |
| Edit User Details or Roles | 734 |
| Deactivating a User Account | 736 |
| Reactivating a User Account | 736 |
| Change Log – Users | 737 |
| <hr/> | |
| CHAPTER 15 | Secure Workload OpenAPIs 739 |
| OpenAPI Authentication | 740 |
| Generate API Key and Secret | 740 |
| Workspaces and Security Policies | 742 |
| Workspaces | 742 |
| Workspace Object | 742 |
| List Applications | 743 |
| Retrieve a Single Workspace | 743 |
| Create a Workspace | 743 |
| Import a New Version | 746 |
| Validate a Set of Policies | 746 |
| Delete a Workspace | 747 |
| Update a Workspace | 747 |
| Retrieve Workspace Details | 748 |
| List Workspace Versions | 748 |
| Delete Workspace Version | 749 |
| Compare Workspace versions | 750 |
| Analyze latest policies | 750 |
| Disable policy analysis on a single workspace | 751 |
| Enforce a single workspace | 752 |
| Disable enforcement for a single workspace | 752 |
| Initiate Automatic Policy Discovery | 753 |
| Get Status of a Policy Discovery Run | 756 |
| Policies | 756 |
| Policy object | 756 |
| Get Policies | 757 |
| Get Specific Policy | 760 |

| | |
|---|-----|
| Search for a Specific Policy With Policy Identifier | 761 |
| Create a Policy | 762 |
| Update a Policy | 763 |
| Adding Service Ports to a Policy | 764 |
| Updating Service Ports of a Policy | 764 |
| Deleting Service Ports of a Policy | 764 |
| Deleting a Policy | 765 |
| Deleting a Policy with Identifier | 765 |
| Policy Quick Analysis | 766 |
| Policy Statistics | 769 |
| Unused Policies | 772 |
| Policy Templates | 774 |
| Get Policy Templates | 774 |
| Get Specific Policy Template | 774 |
| Create a Policy Template | 775 |
| Update a Policy Template | 775 |
| Deleting a Policy Template | 776 |
| Download a Policy Template | 776 |
| Clusters | 777 |
| Cluster object | 777 |
| Get Clusters | 778 |
| Get Specific Cluster | 778 |
| Create a Cluster | 779 |
| Update a Cluster | 780 |
| Deleting a Cluster | 781 |
| Conversations | 781 |
| Search Conversations in a Policy Discovery Run | 781 |
| Top N Conversations in a Policy Discovery Run | 783 |
| Supported Dimensions | 785 |
| Supported metrics | 786 |
| Exclusion Filters | 786 |
| Exclusion Filter object | 786 |
| Get Exclusion Filters | 787 |
| Get Specific Exclusion Filter | 787 |

| | |
|--|-----|
| Create an Exclusion Filter | 788 |
| Update an Exclusion Filter | 789 |
| Deleting an Exclusion Filter | 789 |
| Default Exclusion Filters | 790 |
| Default Exclusion Filter object | 790 |
| Get Default Exclusion Filters | 791 |
| Get Specific Default Exclusion Filter | 791 |
| Create a Default Exclusion Filter | 791 |
| Update a Default Exclusion Filter | 792 |
| Deleting a Default Exclusion Filter | 793 |
| Live Analysis | 793 |
| Flow dimensions available in Live Analysis | 793 |
| Flow metrics available in Live Analysis | 794 |
| Download flows available via Live Analysis | 794 |
| Scopes | 796 |
| Scope object | 796 |
| Get scopes | 797 |
| Create a scope | 797 |
| Get specific scope | 798 |
| Update a scope | 798 |
| Delete a specific scope | 798 |
| Get scopes in policy priority order | 799 |
| Update the policy order | 799 |
| Commit scope query changes | 799 |
| Submit a group suggestion request | 800 |
| Get group suggestion status | 800 |
| Configure Alerts | 801 |
| Alert Object | 801 |
| Get Alerts | 802 |
| Create an Alert | 802 |
| Get Specific Alert | 803 |
| Update an Alert | 803 |
| Delete Specific Alert | 804 |
| Roles | 804 |

| | |
|--------------------------------------|-----|
| Role object | 804 |
| Get roles | 805 |
| Create a role | 805 |
| Get specific role | 806 |
| Update a role | 806 |
| Give a role access to scope | 806 |
| Delete specific role | 808 |
| Users | 808 |
| User object | 808 |
| Get users | 809 |
| Create a new user account | 809 |
| Get specific user | 810 |
| Update a user | 810 |
| Enable/reactivate a deactivated user | 811 |
| Add role to the user account | 811 |
| Remove role from the user account | 812 |
| Delete specific user | 812 |
| Inventory filters | 813 |
| Inventory Filter Object | 813 |
| Get inventory filters | 813 |
| Create an inventory filter | 814 |
| Validate an inventory filter query | 815 |
| Get specific inventory filter | 815 |
| Update specific inventory filter | 815 |
| Delete a specific inventory filter | 816 |
| Flow Search | 816 |
| Query for Flow Dimensions | 816 |
| Query for Flow Metrics | 816 |
| Query for Flows | 817 |
| Filters | 818 |
| Primitive Filter Types | 819 |
| Logical Filter Types | 820 |
| TopN Query for Flows | 820 |
| Flow Count | 822 |

| | |
|--|-----|
| Inventory | 823 |
| Query for inventory dimensions | 823 |
| Inventory search | 823 |
| Inventory Statistics | 825 |
| Inventory count | 826 |
| Inventory vulnerability | 827 |
| Workload | 828 |
| Workload details | 828 |
| Workload Statistics | 829 |
| Installed Software Packages | 830 |
| Workload Vulnerabilities | 831 |
| Workload Long Running Processes | 833 |
| Workload Process Snapshot Summary | 834 |
| Workload Process Snapshot | 834 |
| JSON Object Definitions | 836 |
| Default Policy Generation Config | 837 |
| Policy Generation Config object | 837 |
| Get the Default Policy Generation Config | 838 |
| Set the Default Policy Generation Config | 838 |
| Forensics Intent | 839 |
| Forensic intent object | 839 |
| Listing a forensic intents | 839 |
| Retrieving a Single Forensic Intent | 840 |
| Creating a Forensic Intent | 840 |
| Update a Forensic Intent | 840 |
| Delete a Forensic Intent | 840 |
| Forensics Intent Orders | 841 |
| Forensic Intent Order Object | 841 |
| Retrieve the Current Forensic Intent Order | 841 |
| Creating a Forensic Intent Order | 841 |
| Forensics Profiles | 842 |
| Forensic Profile Object | 842 |
| Listing Forensic Profiles | 843 |
| Retrieving a Single Forensic Profile | 843 |

| | |
|---|-----|
| Creating a Forensic Profile | 843 |
| Update a Forensic Profile | 843 |
| Delete a Forensic Profile | 844 |
| Forensics Rules | 844 |
| Forensic Rule Object | 844 |
| Listing a Forensic Rules | 845 |
| Retrieving a Single Forensic Rule | 845 |
| Creating a Forensic Rule | 845 |
| Update a Forensic Rule | 846 |
| Delete a Forensic Rule | 846 |
| Enforcement | 847 |
| Agent Network Policy Config | 847 |
| Concrete Policy Statistics | 847 |
| JSON Object Definitions | 848 |
| Client Server configuration | 854 |
| Host Config | 854 |
| Port Config | 856 |
| Software Agents | 858 |
| Agent APIs | 858 |
| Software agent configuration using Intents | 859 |
| Interface Config Intents | 862 |
| VRF configuration for agents behind NAT | 864 |
| Secure Workload software download | 865 |
| API to get supported platforms | 865 |
| API to get supported software version | 865 |
| API to create installer ID | 866 |
| API to download Secure Workload software | 866 |
| Secure Workload Agents Upgrade | 867 |
| API to upgrade an agent to specific version | 867 |
| User Uploaded Filehashes | 868 |
| User Filehash Upload | 868 |
| User Filehash Delete | 869 |
| User Filehash Download | 869 |
| User-Defined Labels | 870 |

| | |
|---|-----|
| Scope-Dependent APIs | 870 |
| Scope-Independent APIs | 878 |
| Scope-Independent Labels | 879 |
| Virtual Routing and Forwarding | 882 |
| VRF Object | 882 |
| Get VRFs | 882 |
| Create a VRF | 883 |
| Get Specific VRF | 883 |
| Update a VRF | 884 |
| Delete Specific VRF | 884 |
| Orchestrators | 885 |
| Orchestrator Object | 885 |
| Ingress Controller | 887 |
| Pod Selector | 887 |
| Controller Config | 887 |
| Infoblox Config | 888 |
| Get Orchestrators | 888 |
| Create Orchestrators | 889 |
| Get Specific Orchestrator | 890 |
| Update an Orchestrator | 890 |
| Delete Specific Orchestrator | 891 |
| Orchestrator Golden Rules | 891 |
| Orchestrator Golden Rules Object | 891 |
| Get Orchestrator Golden Rules | 891 |
| Create or Update Golden Rules | 891 |
| FMC Orchestrator Domains | 892 |
| Orchestrator FMC Domains Object | 892 |
| Get FMC Domains | 893 |
| Update FMC Domain Configuration for FMC External Orchestrator | 893 |
| RBAC (Role-Based Access Control) Considerations | 894 |
| High Availability and Failover Considerations | 894 |
| Kubernetes RBAC Resource Considerations | 894 |
| Service Health | 896 |
| Get Service Health | 896 |

| | |
|---|-----|
| Secure Connector | 896 |
| Get Status | 896 |
| Get Token | 897 |
| Rotate Certificates | 897 |
| Kubernetes Vulnerability Scanning | 897 |
| Get Kubernetes Registries used for Pod Vulnerability Scanning | 897 |
| Add Credentials to Kubernetes Registry | 898 |
| Get Kubernetes Pod Scanners | 900 |
| Edit Scanner Filter Query and Action | 901 |
| Policy Enforcement Status for External Orchestrators | 902 |
| Get Policy Enforcement Status for All External Orchestrators | 902 |
| Get Policy Enforcement Status for an External Orchestrator | 902 |
| Download Certificates for Managed Data Taps and Datasinks | 903 |
| Get List of Managed Data Taps for a Given VRF ID | 903 |
| Download Managed Data Tap Certificates for a Given MDT ID | 903 |
| Get List of DataSinks for a Given VRF ID | 903 |
| Download DataSink Certificates for a Given DataSink ID | 904 |
| Change Logs | 904 |
| Change Log Object | 904 |
| Search | 905 |
| Non-Routable Endpoints | 906 |
| Non-Routable Endpoint Object | 906 |
| GET Non-Routable Endpoints | 907 |
| Create a Non-Routable Endpoint | 907 |
| GET Specific Non-Routable Endpoints with Name | 908 |
| GET Specific Non-Routable Endpoints with ID | 908 |
| Update Specific Non-Routable Endpoint Name | 908 |
| Delete Specific Non-Routable Endpoint with Name | 909 |
| Delete Specific Non-Routable Endpoint with ID | 909 |
| Config and Command Schemas for External Appliances and Connectors | 909 |
| Config Groups APIs | 909 |
| API to Get the Schema of Config | 909 |
| API to Get the Schema of Troubleshooting Commands | 911 |
| External Appliances | 912 |

- External Appliances APIs 912
- Connectors 927
 - Connectors APIs 927

CHAPTER 16

Configuration Limits in Secure Workload 951

- Cloud Connectors 951
- Connectors 952
 - Secure Workload Virtual Appliances for Connectors 953
- Label Limits 953
- Limits Related to Policies 954
- Additional Features 955
- Data-In or Data-Out 956



CHAPTER 1

Get Started with Cisco Secure Workload

Today's networks include applications running in a hybrid multicloud environment that uses bare metal, virtualization, and cloud-based and container-based workloads. The key challenge in such an environment is improving application and data security without compromising on agility. Cisco Secure Workload provides comprehensive workload protection by bringing security closer to applications and tailoring the security posture that is based on the application behavior. Secure Workload achieves this tailoring by using advanced machine learning and behavior analysis techniques. It provides a ready-to-use solution to support the following security use cases:

- Implement a zero-trust model with microsegmentation policies that allow only traffic that is required for business purposes.
- Identify anomalies on workloads using behavioral baselining and analysis.
- Detect Common Vulnerabilities and Exposures in the software packages that are installed on the servers.
- Recommend quarantining of servers if vulnerabilities persist after enforcing policies and blocking communication.

Workloads and IP Addresses in Secure Workload

In Cisco Secure Workload, a workload is an IP address; hosts that have software agents that are installed are called workloads and hosts that do not have an agent that is installed on them are called IP addresses.



Note To view the End User License Agreement and Supplemental End User License Agreement for your product, see [End User License Agreement](#) and [Supplemental End User License Agreements](#).

- [Supported Web Browsers, on page 1](#)
- [Quick Start Wizard, on page 2](#)
- [Get Started with Segmentation and Microsegmentation, on page 2](#)

Supported Web Browsers

Secure Workload supports the following web browsers:

- Google Chrome
- Microsoft Edge

Quick Start Wizard

An optional wizard can guide you through creating the first branch of your scope tree, which is a first step toward generating and enforcing policies for an application you choose. The wizard introduces the concepts and benefits of labels and scopes.

The following user roles can access the wizard:

- Site administrator
- Technical support
- Root scope owner

To access the wizard, do any one of the following:

- Sign in to Cisco Secure Workload.
- Click the link in the blue banner. The blue banner appears at the top of all pages.
- Choose **Overview** from the main menu.



Note You cannot access the wizard if scopes are already defined in **Organize > Scopes and Inventory**. Delete the existing scopes to access the wizard.

Get Started with Segmentation and Microsegmentation

Use the high-level procedures given here to set up segmentation and microsegmentation policies using Secure Workload.

General Process for Implementing Microsegmentation

The intent of segmentation and microsegmentation is to allow only the traffic that is required for business purposes and to block all other traffic.

Procedure

- Step 1** Ensure that Secure Workload supports the platforms and versions that your workloads are running on, and the systems that provide essential information to your policies. See [Secure Workload Compatibility Matrix](#).
- Step 2** Install agents on workloads.
- Agents gather flow data and other information that is required that is for Secure Workload to group workloads and determine appropriate policies. The agents also enforce approved policies. For more information, including links to lists of supported platforms and requirements, see [Deploy Software Agents](#).
- Step 3** Gather or upload labels that describe your workloads.

Labels let you easily understand the purpose of each workload and provide other key information about each workload.

You need this information to group workloads, apply appropriate policies, and understand the policies that Secure Workload suggests. Labels are the foundation of maintaining groups that simplify policy management. For more information, see [Workload Labels](#) and [Importing Custom Labels, on page 340](#).

Step 4 Create a scope tree based on your workload labels.

The logical groups of workloads that labels help you create are called scopes, and a well-chosen set of labels helps you create a hierarchical map of your network called a scope tree. This hierarchical view of the workloads on your network is key to efficiently creating and maintaining policies. The hierarchical view enables you to create a policy once and apply it automatically to every workload on that branch of the tree. The view also lets you delegate responsibility for certain applications (or parts of your network) to people who have the expertise needed to determine the correct policies for those workloads.

You can query workloads and group them into scopes based on their labels. For example, you can create a scope called Email-app that includes all of the workloads that have the labels Application = Email-app and Environment = Production. You can create a parent scope for the Application = Email-app scope by using the query Environment = Production. The Production scope includes the production Email-app and all other workloads labeled with Environment = Production.

For more information, see [Scopes and Inventory](#).

If you have not yet created any scopes, you can use the Quick Start wizard to create a scope tree. For more information, see [Quick Start Wizard, on page 2](#).

Step 5 Create a workspace for each scope for which you want to create policies.

The workspace is where you manage policies for the workloads in that scope. For more information, see [Use Workspaces to Manage Policies](#).

Step 6 Manually create policies that apply across your network.

For example, you might want to allow access from all internal workloads to your NTP server, and deny all external traffic, or deny access from all non-internal hosts unless explicitly permitted. Policies can be absolute, meaning that they cannot be overridden by more specific policies, or default, where they can be overridden by more specific policies.

For more information, see [Manually Create Policies, on page 428](#).

Secure Workload has policy templates that make policy creation easier. For more information, see [Policy Templates, on page 432](#).

You can enforce manually created policies without waiting for the policies to be discovered. For more information, see [Enforce Policies, on page 538](#).

Step 7 Automatically discover policies based on existing traffic patterns.

Secure Workload analyzes traffic between workloads, groups workloads based on their behavior, and suggests a set of policies that are intended to allow the traffic that your organization needs, so you can block all other traffic.

Analysis of more data flow over a longer time period leads to more accurate policy suggestions.

You can discover policies iteratively. (There is more information about this later in this procedure.)

a. Discover policies for a branch of your scope tree.

If you are just getting started, you can have temporary set of policies in place and provide protection against future threats.

b. Discover policies for single scopes.

Typically, you will do this for scopes at or near the bottom of your scope tree. These scopes usually include workloads for a single application.

For more information, see [Discover Policies Automatically](#) and [Discover Policies for One Scope or for a Branch of the Scope Tree](#), on page 439.

Step 8

Review and analyze your policies.

Examine your policies carefully to ensure that they have the effects you expect and that there are no unintended side effects.

Work with subject-matter experts and application owners in your organization to understand the needs of the organization and the appropriateness of suggested policies.

a) Review the policies and clusters that Secure Workload has suggested.

(Clusters are groups of workloads within a scope that are closely related and may warrant policies that are more tailored than policies targeted at the entire scope. For more information, see [Grouping Workloads: Clusters and Inventory Filters](#), on page 487.)

For more information, see [Review Automatically Discovered Policies](#), on page 520.

b) Analyze your policies to see how they affect actual traffic on your network.

Use policy analysis and other tools in Secure Workload to confirm that your policies allow the traffic your organization needs in order to conduct business. For more information, see [Live Policy Analysis and Policy Visual Representation](#), on page 523.

As you analyze the results of your policies, keep the following points in mind:

- Policies in workspaces for higher scopes of a branch might affect the workloads of lower scopes of the branch. For more information, see [Policy Inheritance and the Scope Tree](#), on page 426.
- Microsegmentation creates a miniature firewall around each workload. In order for a connection to be successful, both consumer and provider of the transaction must have policies allowing the traffic. If both workloads are not in the same scope, creating these policies may require extra steps. For more information, see [When Consumer and Provider Are in Different Scopes: Policy Options](#), on page 504.

Step 9

Iteratively discover policies as needed.

More traffic flow produces more accurate policy suggestions. For example, for a monthly report even three weeks worth of data may not capture all essential traffic. Continue to discover policies and review and analyze new policy suggestions. Each discovery run suggests policies based on the current traffic flows.

You can also iteratively discover policies to capture changes in policy discovery settings and approved clusters. For more information, see [Iteratively Revise Policies](#), on page 464.

Before you re-run automatic policy discovery, ensure that you approve policies and clusters that you want to retain.

Each time you re-discover policies, you must review and analyze them.

Step 10

When you are ready, enforce policies.

After you have determined that the policies associated with a workspace (and hence, the associated scope) are appropriate and will block unwanted traffic while not interrupting essential services, you can enforce those policies.

You can iteratively enforce policies; for example, you might initially enforce just the manually created policies in scopes near the top of your tree, then over time, enforce discovered policies in scopes lower in the tree.

For more information, see [Enforce Policies, on page 538](#).

Set Up Microsegmentation for Workloads Running on Bare Metal or Virtual Machines

Procedure

- Step 1** Gather the IP addresses of workloads on your network.
- For each workload, you will also want the application name, application owner, environment (production or non-production), and other information such as geographical region that will determine the policies to be applied.
- If you do not have a Configuration Management Database (CMDB), you can collect this information in a spreadsheet.
- To get started, choose a single application that you can focus on.
- Step 2** Install agents on supported bare-metal-based or virtual workloads.
- For more information, see [Deploy Software Agents](#).
- Step 3** Upload labels that describe these workloads.
- For more information, see [Workload Labels](#) and [Importing Custom Labels, on page 340](#).
- Optionally, you can run the quick start wizard to create labels and the first branch of your scope tree. For more information about the wizard, see [Quick Start Wizard](#).
- Step 4** If needed, create or update your scope tree based on your labels.
- For more information, see [Scopes and Inventory](#).
- Step 5** Create a workspace for each scope for which you want to apply policies.
- For more information, see [Use Workspaces to Manage Policies](#).
- Step 6** Create manual policies that apply across your network.
- For more information, see [Manually Create Policies, on page 428](#).
- Step 7** For more information on platform-specific policies, see [Platform-Specific Policies, on page 472](#).
- Step 8** Automatically discover policies in workspaces associated with lower-level scopes.
- For more information, see [Discover Policies Automatically](#) and subtopics.
- Step 9** Review and analyze the suggested policies.

For more information, see [Review and Analyze Policies, on page 520](#) and subtopics.

Step 10 Iteratively discover policies as needed.

For more information, see [Iteratively Revise Policies, on page 464](#) and subtopics.

Step 11 When you are ready, enforce the policies.

You can enforce policies when you are satisfied with the behavior of the policies in each workspace.

You must enforce policies both in the workspace and in the agent configuration.

For more information, see [Enforce Policies, on page 538](#).

Set Up Microsegmentation for Cloud-Based Workloads

Procedure

Step 1 Install agents on your cloud-based workloads, if required.

Cloud connectors provide VPC/VNet level granularity in policy discovery and enforcement. Install agents on supported platforms if you require policy discovery and enforcement at a more granular level.

Install agents based on the operating system on which your cloud service is running. For more information, see [Deploy Software Agents](#).

Step 2 Set up cloud connectors to gather labels and flow data.

For more information, see:

- [AWS Connector](#).
- [Azure Connector](#).
- [GCP Connector](#)

Step 3 Create workspaces for the scopes created by the connector.

For more information, see [Use Workspaces to Manage Policies](#).

Step 4 Automatically discover policies.

Discover policies for each VPC/VNet-defined scope, and if applicable, for more granular scopes.

For more information, see [Discover Policies Automatically](#).

Step 5 Review and analyze the suggested policies.

See [Review and Analyze Policies, on page 520](#) and subtopics.

Step 6 Iteratively discover policies as needed.

See [Iteratively Revise Policies, on page 464](#) and subtopics.

Step 7 Approve and enforce policies for each scope.

You must enable enforcement in the applicable workspace and in the connector for each VPC or VNet, and for any agents installed on individual workloads.

- For more information, see [Enforce Policies, on page 538](#) and subtopics.
- For more information on:
 - AWS-based workloads, see [Best Practices When Enforcing Segmentation Policy for AWS Inventory](#).
 - Azure-based workloads, see [Best Practices When Enforcing Segmentation Policy for Azure Inventory](#).
 - GCP-based workloads, see [Best Practices When Enforcing Segmentation Policy for GCP Inventory](#).

Set Up Microsegmentation for Kubernetes-Based Workloads

Procedure

- Step 1** Install agents on Kubernetes-based workloads. Ensure that you check the requirements and prerequisites. For more information, see [Installing Kubernetes or OpenShift Agents for Deep Visibility and Enforcement](#). Agents are automatically installed on all future workloads managed by the applicable Kubernetes service.
- Step 2** Gather labels for your Kubernetes-based workloads. For more information on:
- Plain-vanilla Kubernetes and OpenSource workloads, see [External Orchestrators in Secure Workload, on page 123](#) and [Kubernetes/OpenShift, on page 140](#).
 - Elastic Kubernetes Services (EKS) Running on Amazon Web Services (AWS), see [AWS Connector, on page 231](#) and [Managed Kubernetes Services Running on AWS \(EKS\), on page 242](#).
 - Azure Kubernetes Services (AKS), see [Azure Connector, on page 244](#) and [Managed Kubernetes Services Running on Azure \(AKS\), on page 251](#)
 - Google Kubernetes Engine (GKE) running on Google Cloud Platform (GCP), see [Managed Kubernetes Services Running on GCP \(GKE\), on page 260](#).
- Step 3** Create or update your scope tree based on your labels. For more information, see [Scopes and Inventory](#).
- Step 4** Create a workspace for each scope for which you want to apply policies. For more information, see [Use Workspaces to Manage Policies](#).
- Step 5** Automatically discover policies for each low-level scope. For more information, see [Discover Policies Automatically](#).
- Step 6** For more information on applicable additional options, see [Platform-Specific Policies, on page 472](#).
- Step 7** Review and analyze the suggested policies.

For more information, see [Review and Analyze Policies, on page 520](#).

Step 8 Iteratively discover, review, and analyze policies as needed.

For more information, see [Iteratively Revise Policies, on page 464](#).

Step 9 When you are ready, approve and enforce policies for each scope.

You must enable policy enforcement in the workspace and for the agents.

For more information, see [Enforce Policies, on page 538](#) and [Enforcement on Containers, on page 546](#).

What to do next

Related Information:

- [Workload Labels](#)
- [Scopes and Inventory](#)
- [Deploy Software Agents](#)
- [Manage Policies Lifecycle in Secure Workload](#)
- [Secure Workload Quick Start Guide](#)



CHAPTER 2

Deploy Software Agents on Workloads

A Secure Workload software agent is a lightweight piece of software that you install on your workloads. The purpose of the agent is to:

- Collect host information such as network interfaces and active processes running in the system.
- Monitor and collect network flow information.
- Enforce security policies by setting firewall rules for hosts on which the software agent is installed and enabled.

Agents automatically update the Secure Workload inventory when interface addresses change. You do not need to install agents on end-user (employee) computers.

- [Deploy Software Agents, on page 9](#)
- [Security Exclusions, on page 37](#)
- [Service Management of Agents, on page 39](#)
- [Policy Enforcement with Agents, on page 41](#)
- [Software Agent Config, on page 65](#)
- [View Detailed Agent Status in the Workload Profile, on page 74](#)
- [Rehoming of Agents, on page 76](#)
- [Generate Agent Token, on page 79](#)
- [Host IP Address Change when Enforcement is Enabled, on page 80](#)
- [Upgrading Software Agents, on page 81](#)
- [Removing Software Agents, on page 85](#)
- [Data collected and exported by workload agents, on page 88](#)
- [Enforcement Alerts, on page 91](#)
- [Sensor Alerts, on page 94](#)
- [Frequently Asked Questions, on page 97](#)

Deploy Software Agents



Note Installer scripts downloaded from LDAP or AD accounts with automatic role mapping fail once you are logged out. To give the installer scripts uninterrupted access to the cluster, enable Use Local Authentication.

On deployment, the agent is assigned a unique identity by the Secure Workload cluster based on a set of parameters specific to the host where the agent is running. If the host name and the BIOS UUID are a part of the set of parameters, you may encounter the following issues:

1. Registration failure when cloning a virtual machine and retaining the BIOS UUID and host name, and when instant cloning a VDI. The registration failure happens because Secure Workload already has a registered software agent using the same parameters set. You can delete the registered agent using OpenAPI. In some cases, a duplicate BIOS UUID configured during startup is changed by VMware after a certain period of time. Agent registration recovers once the Cisco Secure Workload services are restarted.
2. A new identity is generated for the agent if the host name is changed and the host rebooted. The redundant or the old agent is marked as inactive after a certain period of time. For more information, see Frequently Asked Questions section.

Supported Platforms and Requirements

For information on supported platforms and additional requirements for software agents, see:

- The release notes for your release, see [Release Notes](#).
- The agent install wizard in the Secure Workload web portal: In the navigation menu, click **Manage > Workloads > Agents**, then click the **Installer** tab. Choose an installation method, a platform, and if applicable, an agent type to see supported platform versions.
- [Support Matrix](#) for additional dependencies.
- The following sections for details on additional requirements for each platform and agent type.

Installing Linux Agents for Deep Visibility and Enforcement

Requirements and Prerequisites to Install Linux Agents

- See [Supported Platforms and Requirements](#).
- Root privileges to install and execute the services.
- 1-GB storage space for agent and log file.
- Security exclusions are configured on the security applications that are monitoring the host to prevent these applications from blocking agent installation or agent activity. For more information, see [Security Exclusions](#).
- A special user, **tet-sensor**, is created in the host where the agent is installed. If PAM or SELinux is configured on the host, tet-sensor user must be granted appropriate privileges for executing the tet-sensor process and making connections to collectors. If an alternative install directory is provided and SELinux is configured, ensure that execution is allowed for that location.
- You must be able to use the unzip command, if the agent is installed using the AutoInstall (installer script) method.

Supported Methods to Install Linux Agents

Methods to install a Linux agent for deep visibility and enforcement:

- [Install Linux Agent Using the Agent Script Installer Method, on page 11](#)
 - [Agent Support for NVIDIA Bluefield Networking Platform](#)
- [Install Linux Agent using the Agent Image Installer Method, on page 11](#)

Install Linux Agent using the Agent Image Installer Method

We recommend the automated installer script method for installing Linux agents. Use the image installer method if you have a specific reason for using this manual method..

Prerequisite:

Configure the `ACTIVATION_KEY` and `HTTPS_PROXY` in the `user.cfg` file for SaaS clusters and when you are installing the agent on a non-default tenant of on-premises clusters with multiple tenants. For more information, see [\(Manual Installations Only\) Update the User Configuration File](#).

To install a Linux agent using the agent image method:

Procedure

-
- Step 1** Navigate to Agent Installation Methods:
- If you are a first-time user, launch the Quick Start wizard and click **Install Agents**.
 - In the navigation pane, choose **Manage > Agents**, and select the **Installer** tab.
- Step 2** Click **Agent Image Installer**.
- Step 3** In the **Platform** field, enter Linux.
- Step 4** Enter the required agent type and the version of the agent, and then from the results, download the required version of the agent.
- Step 5** Copy the RPM package to all the Linux hosts for deployment.
- Note** If the agent is already installed on the host, do not reinstall the agent. To upgrade the agent, see [Upgrading Software Agents](#) section.
- Step 6** Based on your platform, run the RPM commands with root privileges.
- For RHEL/CentOS/Oracle platforms, run the command: `rpm -ivh <rpm_filename>`
 - For Ubuntu platform:
 - To retrieve the dependency list and ensure all dependencies are met, run the command: `rpm -qpR <rpm_filename>`
 - Install the agent with “--nodeps” option by running the command: `rpm -ivh \\\--nodeps <rpm_filename>`
-

Install Linux Agent Using the Agent Script Installer Method

We recommend the installer script method to deploy Linux agents for deep visibility and enforcement.



- Note**
- The installed Linux agent supports both deep visibility and enforcement.
 - By default, enforcement is disabled. To enable enforcement, see [Create an Agent Configuration Profile](#).

To install a Linux agent using the script installer method:

Procedure

- Step 1** Navigate to Agent Installation methods:
- If you are a first-time user, launch the **Quick Start Wizard** and click **Install Agents**.
 - From the navigation pane, choose **Manage > Agents**, and select the **Installer** tab.
- Step 2** Click **Agent Script Installer**.
- Step 3** From the **Select Platform** drop-down list, choose **Linux**.
- To view the supported Linux platforms, click **Show Supported Platforms**.
- Step 4** Choose the tenant to install the agents.
- Note** Secure Workload SaaS clusters do not require selecting a tenant.
- Step 5** If you want to assign labels to the workload, choose the label keys and enter label values.
- When the installed agent reports IP addresses on the host, the installer CMDB labels selected here, along with other uploaded CMDB labels that have been assigned to IPs reported by this host, would be automatically assigned to the new IP address. If there are conflicts between uploaded CMDB labels and installer CMDB labels:
- Labels assigned to an exact IP address take precedence over labels assigned to the subnet.
 - Existing labels assigned to an exact IP address take precedence over installer CMDB labels.
- Step 6** If an HTTP proxy is required to communicate with Secure Workload, choose **Yes**, and then enter a valid proxy URL.
- Step 7** In the **Installer expiration** section, select an option:
- No expiration: The installer script can be used multiple times.
 - One time: The installer script can be used only once.
 - Time bound: You can set the number of days for which the installer script can be used.
 - Number of deployments: You can set the number of times the installer script can be used.
- Step 8** Click **Download** and save the file to the local disk.
- Step 9** Copy the installer shell script on Linux hosts and run the following command to grant execute permission to the script: `chmod u+x tetrations_installer_default_sensor_linux.sh`
- Note** The script name may differ depending on the selected agent type and scope.

Step 10

To install the agent, run the following command with root privileges:

```
./tetration_installer_default_sensor_linux.sh
```

Note If an agent is already installed on the tenant, you cannot proceed with the installation.

We recommend running the precheck, as specified in the script usage details.

Linux installer script usage details:

```
bash tetration_linux_installer.sh [--pre-check] [--skip-pre-check=<option>] [--no-install]
  [--logfile=<filename>] [--proxy=<proxy_string>] [--no-proxy] [--help] [--version]
  [--sensor-version=<version_info>] [--ls] [--file=<filename>] [--save=<filename>] [--new]
  [--reinstall] [--unpriv-user] [--force-upgrade] [--upgrade-local]
  [--upgrade-by-uuid=<filename>] [--basedir=<basedir>] [--logbasedir=<logbdir>]
  [--tmpdir=<tmp_dir>] [--visibility] [--golden-image]
  --pre-check: run pre-check only
  --skip-pre-check=<option>: skip pre-installation check by given option; Valid options
  include 'all', 'ipv6' and 'enforcement'; e.g.: '--skip-pre-check=all' will skip all
  pre-installation checks; All pre-checks will be performed by default
  --no-install: will not download and install sensor package onto the system
  --logfile=<filename>: write the log to the file specified by <filename>
  --proxy=<proxy_string>: set the value of CL_HTTPS_PROXY, the string should be formatted
  as http://<proxy>:<port>
  --no-proxy: bypass system wide proxy; this flag will be ignored if --proxy flag was
  provided
  --help: print this usage
  --version: print current script's version
  --sensor-version=<version_info>: select sensor's version; e.g.: '--sensor-version=3.4.1.0';
  will download the latest version by default if this flag was not provided
  --ls: list all available sensor versions for your system (will not list pre-3.1 packages);
  will not download any package
  --file=<filename>: provide local zip file to install sensor instead of downloading it
  from cluster
  --save=<filename>: download and save zip file as <filename>
  --new: remove any previous installed sensor; previous sensor identity has to be removed
  from cluster in order for the new registration to succeed
  --reinstall: reinstall sensor and retain the same identity with cluster; this flag has
  higher priority than --new
  --unpriv-user=<username>: use <username> for unpriv processes instead of tet-sensor
  --force-upgrade: force sensor upgrade to version given by --sensor-version flag; e.g.:
  '--sensor-version=3.4.1.0 --force-upgrade'; apply the latest version by default if
  --sensor-version flag was not provided
  --upgrade-local: trigger local sensor upgrade to version given by --sensor-version flag;
  e.g.: '--sensor-version=3.4.1.0 --upgrade-local'; apply the latest version by default if
  --sensor-version flag was not provided
  --upgrade-by-uuid=<filename>: trigger sensor whose uuid is listed in <filename> upgrade
  to version given by --sensor-version flag; e.g.: '--sensor-version=3.4.1.0
  --upgrade-by-uuid=/usr/local/tet/sensor_id'; apply the latest version by default if
  --sensor-version flag was not provided
  --basedir=<base_dir>: instead of using /usr/local use <base_dir> to install agent. The
  full path will be <base_dir>/tetration
  --logbasedir=<log_base_dir>: instead of logging to /usr/local/tet/log use <log_base_dir>.
  The full path will be <log_base_dir>/tetration
  --tmpdir=<tmp_dir>: instead of using /tmp use <tmp_dir> as temp directory
  --visibility: install deep visibility agent only; --reinstall would overwrite this flag
  if previous installed agent type was enforcer
  --golden-image: install Cisco Secure Workload Agent but do not start the Cisco Secure
  Workload Services; use to install Cisco Secure Workload Agent on Golden Images in VDI
  environment or Template VM. On VDI/VM instance created from golden image with different
  host name, Cisco Secure Workload Services will work normally
```

**Note**

- Ubuntu uses the native .deb package, and new installations and reinstallations switch to this package type. Upgrades from previous versions continue with the .rpm package.
- Ubuntu .deb package is installed under `/opt/cisco/tetration`.
- There is no relocation support for the .deb package and so the `-basedir` option is not supported for Ubuntu.

Agent Support for NVIDIA Bluefield Networking Platform

A data processing unit (DPU) is a programmable processor that is designed to manage data-centric tasks, including but not limited to data transfer, power optimization, security, compression, analytics, and encryption.

The NVIDIA DPU is a smart network interface card (SmartNic) with excellent network performance. It delivers a high-speed Ethernet NIC capability and it enables the execution of software directly on the NIC itself, allowing for interception, monitoring, and manipulation of network traffic passing through the NIC.

NVIDIA facilitates the functionality through the provision of the DOCA SDK. Leveraging virtualization technology based on PCIe Single Root I/O Virtualization (SR-IOV), the DPU establishes a mechanism for virtual machines (VMs) to communicate directly without hypervisor involvement. The DPU incorporates an OpenVSwitch-based hardware-accelerated eSwitch for network control, enhancing overall efficiency.

Requirements and Prerequisites

- Ensure that Ubuntu 22.04-based DOCA is installed on the BlueField networking platform.
- Set up the DPU card network to enable an agent's connection to the cluster through one of the out-of-band interfaces. Options include `oob_net0`, `tmfif0_net0`, or the in-band connection through `enp3s0f0s0`.

Agent Installation

The installation follows a Linux-like process.

1. Navigate to Agent Installation Methods:
 - If you are a first-time user, launch the **Quick Start** wizard and click **Install Agents**.
 - From the navigation pane, choose **Manage > Workloads > Agents**.
2. Under the **Installer** tab, click **Agent Script Installer**.
3. From the **Select Platform** drop-down list, choose **Linux**.

To view the supported Linux platforms, click **Show Supported Platforms**.

**Note**

Secure Workload Agent is only supported on the Ubuntu 22-based DOCA SDK.

4. Choose the tenant to install the agents.



Note Selecting a tenant is not required for Secure Workload SaaS clusters.

5. (Optional) If you want to assign labels to the workload, choose the label keys and enter label values.
6. If an HTTP proxy is required to communicate with Secure Workload, click **Yes**, and then enter a valid proxy.
7. In the **Installer expiration** section, select one of the available options:
 - **No expiration:** The installer script can be used multiple times.
 - **One time:** The installer script can be used only once.
 - **Time-bound:** You can set the number of days for which the installer script can be used.
 - **Number of deployments:** You can set the number of times the installer script can be used.
8. Click **Download** to download the Linux installer script on to DPU using one of the network devices.
9. Run the installer script. For more information, see [Install Linux Agent using the Agent Script Installer Method](#).

Figure 1: Install Script

The screenshot shows the 'Install Scripts' interface with the following sections:

- Agent Script Installer:** Includes a 'Download' button, a 'Select Platform' dropdown (set to 'Linux'), and a 'Show Supported Platforms' button.
- Which tenant is your agent going to be installed under? *** A dropdown menu showing 'DPUTENANT'.
- Which labels would you like us to apply to this workload? (optional)** A section with 'Label Key' and 'Label Value' input fields, and an 'Add another' link.
- Does your network require HTTP Proxy to reach Secure Workload? *** Radio buttons for 'Yes' and 'No'.
- Installer expiration *** Radio buttons for 'No expiration', 'One Time', 'Time Bounded', and 'Number of Deployments'.
- Installation Instructions:**
 - Download Installer:** Download the script file and copy it to all the Linux hosts for deployment.
 - Installation Precheck (Optional):** Run the installer script with `--pre-check` as a root user. A terminal snippet shows: `$ bash tetration_installer_agent_enforcer_linux_k8s.sh --pre-check`.
 - Dependency Packages:** A list of packages to be checked: `curl`, `rpm`, `netcat`, `iptables`, `iputils`, `dnsmasq`, `dnsmasq-libs`, `dnsmasq-libs`, `dnsmasq-libs`, `dnsmasq-libs`, `dnsmasq-libs`, `dnsmasq-libs`.
 - Installation:** Run the installer script as a root user. A terminal snippet shows: `$ bash tetration_installer_agent_enforcer_linux_k8s.sh --new`.

Choose **Software Agents > Agent List** and click a **Hostname**. Under **Interfaces**, you can view the current mapping of interfaces with the associated IP addresses.

Figure 2: Interface Mapping

| Name | Mac Address | VRF | Family Type | IP Address | Network |
|------|-------------------|-----------|-------------|-------------------|-------------------|
| vif1 | 52:54:00:ca:1c:1c | DPUTENANT | IPv4 | 172.16.100.100 | 255.255.255.255 |
| vif1 | 52:54:00:ca:af:7a | DPUTENANT | IPv4 | 172.16.100.100 | 255.255.255.255 |
| vif1 | 52:54:00:ca:af:7a | DPUTENANT | IPv6 | fe80::ca:af:7a::1 | :::ffff:ffff:ffff |
| vif1 | 52:54:00:ca:af:7a | DPUTENANT | IPv6 | fe80::ca:af:7a::1 | :::ffff:ffff:ffff |
| vif1 | 52:54:00:8a:92:3a | DPUTENANT | IPv4 | 172.16.100.100 | 255.255.255.255 |
| vif1 | 52:54:00:8a:92:3a | DPUTENANT | IPv6 | fe80::8a:92:3a::1 | :::ffff:ffff:ffff |
| vif1 | 52:54:00:8a:92:3a | DPUTENANT | IPv6 | fe80::8a:92:3a::1 | :::ffff:ffff:ffff |

Choose **Investigate** > **Traffic** to monitor the network traffic between virtual machines (VMs) when those are utilizing the SR_IOV virtual network interfaces provided by the DPU. The agent on the DPU enables the segmentation of network traffic between these virtual network interfaces.

Verify Linux Agent Installation

Procedure

Run the command `sudo rpm -q tet-sensor` `sudo rpm -q tet-sensor`.

```
sudo rpm -q tet-sensor
```

A single entry as output confirms that a Linux agent is installed on the host.

Sample output: `tet-sensor-3.1.1.50-1.el6.x86_64`

The specific output may differ depending on the platform and architecture.

Installing Windows Agents for Deep Visibility and Enforcement

Requirements and Prerequisites for Installing Windows Agent

- See the Supported Platforms and Requirements section.
- Administrator privileges are required for installation and service execution.
- Npcap must be installed on workloads running Windows 2008 R2 or when the installed agent version is earlier than version 3.8. If the Npcap driver is not already installed, the recommended Npcap version is installed in the background by the agent after the service starts. For more information, see the Npcap version information.
- One GB storage space for agent and log files.
- Enable the Windows services required for agent installation. Some of the Windows services could have been disabled if your Windows hosts have been security hardened, or have deviated from the default configurations. For more information, see the Required Windows Services section.

- Security exclusions configured on security applications that are monitoring the host and that could block agent installation or agent activity. For more information, see [Security Exclusions](#).

Supported Methods to Install Windows Agents

There are two methods to install Windows agents for deep visibility and enforcement.

- [Install Windows Agent using the Agent Script Installer Method, on page 17](#)
- [Install Windows Agent using the Agent Image Installer Method, on page 19](#)

You can also install using a golden image. For more information, see [Deploying Agents on a VDI Instance or VM Template \(Windows\)](#).

Install Windows Agent using the Agent Script Installer Method

We recommend the script installer method to deploy Windows agents for deep visibility and enforcement.



Note

- The installed Windows agent supports both deep visibility and enforcement.
- By default, enforcement is disabled. To enable enforcement, see [Create an Agent Configuration Profile, on page 67](#).

To install a Windows agent using the script installer method:

Procedure

- Step 1** Navigate to Agent Installation Methods:
- If you are a first-time user, launch the Quick Start wizard and click **Install Agents**.
 - From the navigation pane, choose **Manage > Agents**, and select the **Installer** tab.

- Step 2** Click **Agent Script Installer**.

- Step 3** From the **Select Platform** drop-down menu, choose **Windows**.

To view the supported Windows platforms, click **Show Supported Platforms**.

- Step 4** Choose the tenant to install the agents.

Note Selecting a tenant is not required for Secure Workload SaaS clusters.

- Step 5** If you want to assign labels to the workload, choose the label keys and enter label values.

When the installed agent reports IP addresses on the host, the installer CMDB labels selected here, along with other uploaded CMDB labels that have been assigned to IPs reported by this host, would be assigned to the new IP address. If there are conflicts between uploaded CMDB labels and installer CMDB labels:

- Labels assigned to an exact IP address take precedence over labels assigned to the subnet.
- Existing labels assigned to an exact IP address take precedence over installer CMDB labels.

Step 6 If HTTP proxy is required to communicate with Secure Workload, choose **Yes**, and then enter a valid proxy URL.

Step 7 Under the **Installer expiration** section, select one from the available options:

- No expiration: The installer script can be used multiple times.
- One time: The installer script can be used only once.
- Time bound: You can set the number of days for which the installer script can be used.
- Number of deployments: You can set the number of times the installer script can be used.

Step 8 Click **Download** and save the file to the local disk.

Step 9 Copy the installer PowerShell script to all the Windows hosts for deployment and run the script with administrative privileges.

- Note**
- Depending on the system settings, the command `Unblock-File` may need to be run before other commands.
 - The script does not run if the agent is already installed on the tenant.

We recommend running the pre-check, as specified in the script usage details.

Windows installer script usage details:

```
# powershell -ExecutionPolicy Bypass -File tetration_windows_installer.ps1 [-preCheck]
[-skipPreCheck <Option>] [-noInstall] [-logFile <FileName>] [-proxy <ProxyString>] [-noProxy]
[-help] [-version] [-sensorVersion <VersionInfo>] [-ls] [-file <FileName>] [-save <FileName>]
[-new] [-reinstall] [
-npcap] [-forceUpgrade] [-upgradeLocal] [-upgradeByUUID <FileName>] [-visibility]
[-goldenImage] [-installFolder <Installation Path>]
  -preCheck: run pre-check only
  -skipPreCheck <Option>: skip pre-installation check by given option; Valid options include
'all', 'ipv6' and 'enforcement'; e.g.: '-skipPreCheck all' will skip all pre-installation
checks; All pre-checks will be performed by default
  -noInstall: will not download and install sensor package onto the system
  -logFile <FileName>: write the log to the file specified by <FileName>
  -proxy <ProxyString>: set the value of HTTPS_PROXY, the string should be formatted as
http://<proxy>:<port>
  -noProxy: bypass system wide proxy; this flag will be ignored if -proxy flag was provided

  -help: print this usage
  -version: print current script's version
  -sensorVersion <VersionInfo>: select sensor's version; e.g.: '-sensorVersion 3.4.1.0.win64';
will download the latest version by default if this flag was not provided
  -ls: list all available sensor versions for your system (will not list pre-3.1 packages);
will not download any package
  -file <FileName>: provide local zip file to install sensor instead of downloading it from
cluster
  -save <FileName>: downloaded and save zip file as <FileName>
  -new: remove any previous installed sensor; previous sensor identity has to be removed
from cluster in order for the new registration to succeed
  -reinstall: reinstall sensor and retain the same identity with cluster; this flag has
higher priority than -new
  -npcap: overwrite existing npcap
  -forceUpgrade: force sensor upgrade to version given by -sensorVersion flag; e.g.:
'-sensorVersion 3.4.1.0.win64 -forceUpgrade'; apply the latest version by default if
-sensorVersion flag was not provided
```

```

-upgradeLocal: trigger local sensor upgrade to version given by -sensorVersion flag; e.g.:
'-sensorVersion 3.4.1.0.win64 -upgradeLocal'; apply the latest version by default if
-sensorVersion flag was not provided
-upgradeByUUID <FileName>: trigger sensor whose uuid is listed in <FileName> upgrade to
version given by -sensorVersion flag; e.g.: '-sensorVersion 3.4.1.0.win64 -upgradeByUUID
"C:\Program Files\Cisco Tetration\sensor_id"'; apply the latest version by default if
-sensorVersion flag was not provided
-visibility: install deep visibility agent only; -reinstall would overwrite this flag if
previous installed agent type was enforcer
-goldenImage: install Cisco Secure Workload Agent but do not start the Cisco Secure
Workload Services; use to install Cisco Secure Workload Agent on Golden Images in VDI
environment or Template VM. On VDI/VM instance created from golden image with different
host name, Cisco Secure Workload Services will work normally
-installFolder: install Cisco Secure Workload Agent in a custom folder specified by
-installFolder e.g.: '-installFolder "c:\custom sensor path"'; default path is "C:\Program
Files\Cisco Tetration"

```

Install Windows Agent using the Agent Image Installer Method

We recommend the automated installer script method for installing Windows agents. Use the image installer method if you have a specific reason for using this manual method.



Note Do not manually deploy an older MSI agent version when an existing agent is already running on the host.

Site-related files that are in the package:

- **ca.cert**—Mandatory—CA certificate for sensor communications.
- **enforcer.cfg**—Mandatory only when installing enforcement sensor—Contains configuration of enforcement endpoints.
- **sensor_config**—Mandatory—Configuration for deep visibility sensor.
- **sensor_type**—Type of the sensor (enforcement or deep visibility).
- **site.cfg**—Mandatory—Global site endpoint configuration.
- **user.cfg**—Mandatory for SaaS—Sensor activation key and proxy configuration.

Prerequisite:

Configure the `ACTIVATION_KEY` and `HTTPS_PROXY` in the `user.cfg` file for SaaS clusters and when you are installing the agent on a non-default tenant of on-premises clusters with multiple tenants. For more information, see [\(Manual Installations Only\) Update the User Configuration File](#).

To install a Windows agent using the agent image method:

Procedure

- Step 1** Navigate to Agent Installation Methods:
- If you are a first-time user, launch the Quick Start wizard and click **Install Agents**.
 - From the navigation pane, choose **Manage > Agents**, and select the **Installer** tab.
- Step 2** Click **Agent Image Installer**.

- Step 3** In the **Platform** field, enter Windows.
- Step 4** Enter the required agent type and the version of the agent, and then from the results, download the required version of the agent.
- Step 5** Copy the `tet-win-sensor<version>.win64-<clustername>.zip` file to all the Windows hosts for deployment.
- Step 6** Ensure that you have administrative privileges and extract the ZIP file.
- Step 7** In the extracted folder, run the following command to install the agent: `msiexec.exe /i`

```
TetrationAgentInstaller.msi
```

Additionally, the following options are available for MSI installer.

Table 1: Available Options for MSI Installer

| Options | Description |
|---|--|
| <code>agenttype=<AgentType></code> | <i>AgentType</i> should be either <i>sensor</i> or <i>enforcer</i> , depending on whether enforcement is required. By default, the installer checks the content of the <code>sensor_type</code> file in the same folder and uses the content to overwrite the passed parameter. However, if agent is installed in <i>/quiet</i> mode, the option is required. |
| <code>overwritenpcap=yes</code> | For Windows 2008 R2, by default, the agent does not attempt to upgrade Npcap if Npcap already exists. Pass this parameter to upgrade the existing Npcap. If this option is used, subsequent agent auto-upgrades also upgrade Npcap to newer supported versions. |
| <code>nostart=yes</code> | Pass this parameter, when installing the agent using a golden image in a VDI environment or VM template, to prevent agent service— <code>CswAgent</code> from starting automatically. On VDI/VM instances created using the golden image and with a different host name, these services, as expected, start automatically. |
| <code>installfolder=<FullPathCustomFolder></code> | Use this parameter, at the end of the install command, to install the agent in a custom folder. |
| <code>serviceuser=<Service UserName></code> | Use this parameter, at the end of the install command, to configure the service user. The default service user is “LocalSystem”. For local user, <code>serviceuser=.\<Service UserName></code> For domain user, <code>serviceuser=<domain_name>\<samaccount name></code> Service user must have local administrative privileges. |
| <code>servicepassword=<Service UserPassword></code> | Use this parameter, at the end of the install command, to configure the password for the service user. The password must be in plain-text format. |

| Options | Description |
|--------------------------------|--|
| proxy="<proxy_address>" | Use this parameter to set the HTTPS proxy for accessing the Secure Workload cluster. |
| activationkey=<activation Key> | Use this parameter to specify the tenant if agent is not being installed under the default tenant. |



- Note**
- If activation key and proxy options are used during manual installation, you do not need to manually configure *user.cfg*.
 - For Windows OS other than Windows 2008 R2, when you upgrade to version 3.8, the installed Npcap is automatically uninstalled by the Windows agent.
 - If the agent is already installed on the host, do not reinstall the agent. To upgrade the agent, see Upgrading Software Agents section.

Verify Windows Agent Installation

Procedure

- Step 1** Ensure that the folder `C:\Program Files\Cisco Tetration` (or the custom folder) exists.
- Step 2** Ensure that the service— *CswAgent*, for deep visibility and enforcement, exists and is in the running state. Run command `cmd.exe` with administrative privileges.
- Run the command `sc query cswagent`
- Check if the status is **Running**
- Run the command `sc qc cswagent`
- Check if the DISPLAY-NAME is **Cisco Secure Workload Deep Visibility**
- OR
- Run the command `services.msc`
- Find the name **Cisco Secure Workload Deep Visibility**
- Check if the status is **Running**

Verify Windows Agent in the Configured Service User Context

1. Ensure that the service *CswAgent* running in the configured service user context. *CswAgent* runs in the same service user context.
- Run the command `cmd.exe` with **Admin** privileges

Run the command `sc qc cswagent`

Check `SERVICE_START_NAME` <configured service user>

OR

Run the command `services.msc`

Find the name **Cisco Secure Workload Deep Visibility**

Check **Log On As** for the <configured service user>

Find the name **Cisco Secure Workload Enforcement**

Check **Log On As** for the <configured service user>

OR

Run the command `tasklist /v | find /i "cswengine"`

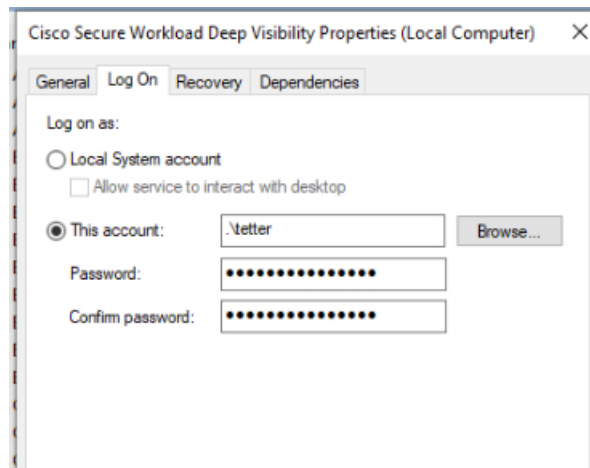
Check the user context for the running processes (5th column)

Modify Service Account

After installing Windows Agents, use one of the following methods to modify the existing Deep Visibility and Enforcement services.

- Use `services.msc`.

Figure 3: Modify Service Account based on services.msc Account



- Use any third party application to configure the services.
- Use the following commands:
 1. Run cmd as an administrator.
 2. Modify the services using the service account name by running the following commands:
 - `sc config cswagent obj= <service user name> password= <password>`
 3. Verify service configurations by running the following commands:
 - `sc qc cswagent`

4. Restart the CswAgent service by running the following commands:
 - a. `sc stop cswagent`
 - b. `sc start cswagent`

Deploying Agents on a VDI Instance or VM Template (Windows)

By default, agent services start automatically after agents are installed. When installing on a golden image, you must use installer flags to prevent these services from starting. When instances are cloned from the golden image, agent services, as expected, start automatically.

Agent will not install Npcap on golden VMs, but will be automatically installed if needed on VM instances cloned from a golden image. For more information, see [Windows Agent Installer and Npcap—For Windows 2008 R2](#).

Install the agent on a golden image in a VDI environment or VM template

Procedure

- Step 1** Install the agent on a golden image in a VDI environment or VM template using an MSI installer or PowerShell installer script:
- Use MSI installer with **nostart=yes**
- For more information, see [Install Windows Agent using the Agent Image Installer Method, on page 19](#).
 - `msiexec.exe /<MSI installer> nostart="yes" /quiet /norestart /!*v <installer_log_file>` OR
- OR
- Use PowerShell installer with the **-goldenImage** flag.
- For more information, see [Install Windows Agent using the Agent Script Installer Method, on page 17](#).
- Step 2** Ensure that the folder `C:\Program Files\Cisco Tetration` (or the custom folder) exists.
- Step 3** Ensure that the service CswAgent exists and is stopped:
- Run the command `cmd.exe` with **Admin** privileges.
- Run the command `sc query cswagent`
- Check if the STATE is **Stopped**.
- Step 4** The VM template is now configured.
- Step 5** Shut down the VM template.
-

Create a new VDI instance VM

Procedure

-
- Step 1** Create a new VDI instance VM by cloning the VM template.
- Step 2** Reboot the VDI instance VM.
- Step 3** After rebooting the VDI instance VM, ensure that the service CswAgent is running in the configured service context. See [Verify Windows Agent Installation](#).
- Step 4** On the VDI instance VM, ensure that the NPCAP driver is installed and running:
- Run the command `cmd.exe` with Admin privileges
- Run the command `sc query npcap`
- Check if STATE is **Running**
- Step 5** On the VDI instance VM, ensure that the agent is registered using a valid `sensor_id`:
- Check the `sensor_id` file in the installation folder.
 - If the `sensor_id` starts with “uuid”, it is not a valid `sensor_id`.
 - If the agent fails to register but the Secure Workload web interface shows that the agent is registered:
 - Delete the agent using OpenAPI. For more information, see [Deploy Software Agents](#).
- Note**
- Do not change the host name of the golden image or VM template.
 - If the golden image or VM template is rebooted after installing the agent, Secure Workload services start running after the reboot.
 - If the VDI instance VM fails to report network flows, see the *VDI Instance VM in Network Flows* section.
-

Windows Agent Installer and Npcap—For Windows 2008 R2

1. For supported Npcap versions, see the Support Matrix at <https://www.cisco.com/go/secure-workload/requirements/agents>.
2. Installation:

If Npcap is not installed, the agent installs the supported version ten seconds after the service starts. If User has Npcap installed but the version is older than the supported version, Npcap is not be upgraded. Manually upgrade or uninstall Npcap, run the agent installer with the option **overwrittenpcap=yes**, or run installer script with **-npcap** to get the supported Npcap version. If Npcap driver is in use by any application, the agent upgrades Npcap at a later time.
3. Upgrade:

If Npcap is installed by Windows Agent and the version is older than the supported version, Npcap is upgraded to the supported version ten seconds after the service starts. If Npcap driver is in use by any

application, the agent upgrades Npcap at a later time. If Npcap is not installed by Windows Agent, Npcap is not upgraded.

4. Uninstall:

If Npcap is installed by the Windows Agent, the agent uninstalls Npcap. If Npcap is installed by the user, but upgraded by the agent installer with **overwrittenpcap=yes**, Npcap is not uninstalled. If Npcap driver is in use by any application, the agent does not uninstall Npcap.

Windows Agent Flow Captures: For All Windows OS Excluding Windows Server 2008 R2

From the latest version of Windows, the agent uses ndiscap.sys (Microsoft in-built) driver and Events Tracing using Windows (ETW) framework to capture the network flows.

During the upgrade to the latest version:

- The agent switches to ndiscap.sys from npcap.sys.
- The agent installer uninstalls Npcap if:
 - Npcap is installed by the agent.
 - Npcap is not in use.
 - OS version is not Windows Server 2008 R2.

After the agent services are started, the agent creates ETW sessions, CSW_MonNet, and CSW_MonDns (for DNS data), and initiates the capture of network flows.



Note

- On Windows Server 2012, network packets are parsed for DNS data.
- The Windows agent on hosts with Windows Server 2012 and later capture consumer and provider usernames and the usernames are available in the flow observations. This feature is not supported on Windows Server 2008 R2 because of limitations in the OS. In the agent configuration profile, configure the following to capture the usernames:
 - Enable PID/ User Lookup.
 - Set Flow Analysis Fidelity to Detailed.

Installing AIX Agents for Deep Visibility and Enforcement



Note

Process tree, Package (CVE), and Forensic Event reporting features are not available on AIX. Additionally, some aspects of those features may not be available on specific minor releases of otherwise supported platforms due to OS limitations.

Requirements and Prerequisites for Installing AIX Agents

- See [Supported Platforms and Requirements](#).

- Additional requirements for deep visibility:
 - Root privileges to install and execute the services.
 - Storage requirement for agent and log files: 500 MB.
 - Security exclusions configured on any security applications that are monitoring the host. These exclusions are to prevent other security applications from blocking agent installation or agent activity. For more information, see [Security Exclusions](#).
 - AIX supports flow capture of only 20 network devices (6 network devices if version is AIX 7.1 TL3 SP4 or earlier). The deep visibility agent captures from a maximum of 16 network devices, leaving the other 4 capture sessions available for exclusive generic system usage (For example, tcpdump).
 - The deep visibility agent does the following to ensure flow capture of 20 network devices:
 - The agent creates 16 bpf device nodes under the agents directory (/opt/cisco/tetration/chroot/dev/bpf0 - /opt/cisco/tetration/chroot/dev/bpf15)
 - tcpdump and other system tools using bpf will scan through the system device nodes (/dev/bpf0-/dev/bpf19) until they find an unused node (!EBUSY)
 - The bpf nodes created by the agent and the system bpf nodes share the same major/minor, with each major or minor being opened only by one instance (either tcpdump or agent).
 - The agent does not access the system device nodes nor does it create them as the tcpdump does (tcpdump-D creates /dev/bpf0. . . /dev/bpf19 if they do not exist).
 - Running iptrace on the system prevents, in certain scenarios, flow capture from tcpdump and the deep visibility agent. This is a known design issue and needs to be checked with IBM.
 - To check if this scenario exists, before installing the agent, run tcpdump. If error message is **tcpdump: BIOCSETIF: en0: File exists** the iptrace is blocking flow capture. Stop iptrace to resolve the issue.
 - All deep visibility functions are not supported in AIX. Package and process accounting are among the ones not supported.
- Additional requirements for policy enforcement:
 - If IP Security Filter is enabled (that is, smitty IPsec4), agent installation fails in pre-check. We recommend you to disable IP Security Filter before installing the agent.
 - If IP Security is enabled when the Secure Workload enforcer agent is running, an error is reported and the enforcer agent stops enforcing. Contact support to safely disable the IP Security filter when the enforcer agent is running.

Install AIX Agent using the Agent Script Installer Method

Deep visibility and enforcement AIX agents can only be installed using the Agent Script Installation method.



-
- Note**
- The installed AIX agent supports both deep visibility and enforcement.
 - By default, enforcement is disabled. To enable enforcement, see [Create an Agent Configuration Profile, on page 67](#).
-

To install an AIX agent:

Procedure

- Step 1** Navigate to Agent Installation Methods:
- If you are a first-time user, launch the Quick Start wizard and click **Install Agents**.
 - From the navigation pane, choose **Manage > Agents**, and select the **Installer** tab.
- Step 2** Click **Agent Script Installer**.
- Step 3** From the **Select Platform** drop-down menu, choose **AIX**.
- To view the supported AIX platforms, click **Show Supported Platforms**.
- Step 4** Choose the tenant to install the agents.
- Note** Selecting a tenant is not required for Secure Workload SaaS clusters.
- Step 5** If you want to assign labels to the workload, choose the label keys and enter label values.
- When the installed agent reports IP addresses on the host, the installer CMDB labels selected here, along with other uploaded CMDB labels that have been assigned to IPs reported by this host, would be automatically assigned to the new IP address. If there are conflicts between uploaded CMDB labels and installer CMDB labels:
- Labels assigned to an exact IP address take precedence over labels assigned to the subnet.
 - Existing labels assigned to an exact IP address take precedence over installer CMDB labels.
- Step 6** If HTTP proxy is required to communicate with Secure Workload, choose **Yes**, and then enter a valid proxy URL.
- Step 7** Under the **Installer expiration** section, select one from the available options:
- No expiration: The installer script can be used multiple times.
 - One time: The installer script can be used only once.
 - Time bound: You can set the number of days for which the installer script can be used.
 - Number of deployments: You can set the number of times the installer script can be used.
- Step 8** Click **Download** and save the file to the local disk.
- Step 9** Copy the installer shell script to all the AIX hosts for deployment.
- Step 10** To grant execute permission to the script, run the command: `chmod u+x tetration_installer_default_sensor_aix.sh`

Note The script name may differ depending on the agent type and scope.

Step 11

To install the agent, run the following command with root privileges:

```
./tetration_installer_default_sensor_aix.sh
```

Note If an agent is already installed on the host, you cannot proceed with the installation.

We recommend running the pre-check, as specified in the script usage details.

AIX installer script usage details:

```
ksh tetration_installer_default_enforcer_aix.sh [--pre-check] [--pre-check-user]
[--skip-pre-check=<option>] [--no-install] [--logfile=<filename>] [--proxy=<proxy_string>]
[--no-proxy] [--help] [--version] [--sensor-version=<version_info>] [--ls]
[--file=<filename>] [--osversion=<osversion>] [--save=<filename>] [--new] [--reinstall]
[--unpriv-user] [--libs=<libs.zip|tar.Z>] [--force-upgrade] [--upgrade-local]
[--upgrade-by-uuid=<filename>] [--logbasedir=<logbdir>] [--tmpdir=<tmp_dir>] [--visibility]
[--golden-image]
--pre-check: run pre-check only
--pre-check-user: provide alternative to nobody user for pre-check su support
--skip-pre-check=<option>: skip pre-installation check by given option; Valid options
include 'all', 'ipv6' and 'enforcement'; e.g.: '--skip-pre-check=all' will skip all
pre-installation checks; All pre-checks will be performed by default
--no-install: will not download and install sensor package onto the system
--logfile=<filename>: write the log to the file specified by <filename>
--proxy=<proxy_string>: set the value of HTTPS_PROXY, the string should be formatted as
http://<proxy>:<port>
--no-proxy: bypass system wide proxy; this flag will be ignored if --proxy flag was
provided
--help: print this usage
--version: print current script's version
--sensor-version=<version_info>: select sensor's version; e.g.: '--sensor-version=3.4.1.0';
will download the latest version by default if this flag was not provided
--ls: list all available sensor versions for your system (will not list pre-3.3 packages);
will not download any package
--file=<filename>: provide local zip file to install sensor instead of downloading it
from cluster
--osversion=<osversion>: specify osversion for --save flag;
--save=<filename>: download and save zip file as <filename>; will download package for
osversion given by --osversion flag; e.g.: '--save=myimage.aix72.tar.Z --osversion=7.2'
--new: remove any previous installed sensor; previous sensor identity has to be removed
from cluster in order for the new registration to succeed
--reinstall: reinstall sensor and retain the same identity with cluster; this flag has
higher priority than --new
--unpriv-user=<username>: use <username> for unpriv processes instead of tet-snsr
--libs=<libs.zip|tar.Z>: install provided libs to be used by agents
--force-upgrade: force sensor upgrade to version given by --sensor-version flag; e.g.:
'--sensor-version=3.4.1.0 --force-upgrade'; apply the latest version by default if
--sensor-version flag was not provided
--upgrade-local: trigger local sensor upgrade to version given by --sensor-version flag;
e.g.: '--sensor-version=3.4.1.0 --upgrade-local'; apply the latest version by default if
--sensor-version flag was not provided
--upgrade-by-uuid=<filename>: trigger sensor whose uuid is listed in <filename> upgrade
to version given by --sensor-version flag; e.g.: '--sensor-version=3.4.1.0
--upgrade-by-uuid=/usr/local/tet/sensor_id'; apply the latest version by default if
--sensor-version flag was not provided
--logbasedir=<log_base_dir>: instead of logging to /opt/cisco/tetration/log use
<log_base_dir>. The full path will be <log_base_dir>/tetration
--tmpdir=<tmp_dir>: instead of using /tmp use <tmp_dir> as temp directory
--visibility: install deep visibility agent only; --reinstall would overwrite this flag
if previous installed agent type was enforcer
```

```
--golden-image: install Cisco Secure Workload Agent but do not start the Cisco Secure
Workload Services; use to install Cisco Secure Workload Agent on Golden Images in VDI
environment or Template VM. On VDI/VM instance created from golden image with different
host name, Cisco Secure Workload Services will work normally
```

Verify AIX Agent Installation

Procedure

Run command `lsllpp -c -l tet-sensor.rte`, confirm that there is one entry as follows.

Note The specific output may differ depending on the version

```
$ sudo lsllpp -c -l tet-sensor.rte /usr/lib/objrepos:tet-sensor.rte:3.4.1.19::COMMITTED:I:TET tet
sensor package:
```

```
$ sudo lssrc -s tet-sensor
```

```
Subsystem Group PID Status tet-sensor 1234567 active
```

```
$ sudo lssrc -s tet-enforcer
```

```
Subsystem Group PID Status tet-enforcer 7654321 active
```

Installing Kubernetes or OpenShift Agents for Deep Visibility and Enforcement

Requirements and Prerequisites

Operating system support information is available at [Agent OS support matrix](#).

Requirements

- The install script requires Kubernetes or OpenShift administrator credentials to start privileged agent pods on the cluster nodes.
- Secure Workload entities are created in the **tetration** namespace.
- The node or pod security policies must permit privileged mode pods.
- busybox:1.33 images must either be preinstalled or be downloadable from Docker Hub.
- For containerd run time, if the `config_path` is not set, modify your `config.toml` (default location: `/etc/containerd/config.toml`) as follows:

```
...
    [plugins."io.containerd.grpc.v1.cri".registry]
      config_path = "/etc/containerd/certs.d"
  ...
```

Restart the containerd daemon.

- To run on Kubernetes or OpenShift control plane nodes, the `-toleration` flag can be used to pass in a toleration for the Secure Workload pods. The toleration that is usually passed is the NoSchedule toleration that normally prevents pods from running on control plane nodes.
- For Windows worker nodes:
 - Supported Windows worker node container runtime: ContainerD.
 - ContainerD config: Configure the following containerd change.

```
...
    [plugins."io.containerd.grpc.v1.cri".registry]
    config_path = "/etc/containerd/certs.d"
  ...
```

Remove configurations under **registry.mirrors**. The default configuration file location is `C:\Program Files\containerd\config.toml`.

Restart the containerd daemon after the configuration changes.

- The image **mcr.microsoft.com/oss/kubernetes/windows-host-process-containers-base-image:v1.0.0** must either be preinstalled or downloadable on the Windows worker node.
- The existing Kubernetes agent which is upgrading to the newer version includes the Windows DaemonSet agent automatically. However, the previous script does not uninstall the Windows DaemonSet agent. Download the latest installer script to uninstall the Windows DaemonSet agent.
- Supported on:
 - Microsoft Windows Server 2022
 - Windows Server 2019
 - Kubernetes 1.27 and later

Requirements for Policy Enforcement

IPVS-based kube-proxy mode is not supported for OpenShift.

These agents should be configured with the Preserve Rules option that is enabled. For more information, see [Create an Agent Configuration Profile](#).

For enforcement to function properly, any installed CNI plug-in must:

- Provide flat address space (IP network) between all nodes and pods. Network plug-ins that masquerade the source pod IP for intracluster communication are not supported.
- Not interfere with Linux iptables rules or marks that are used by the Secure Workload Enforcement Agent (mark bits 21 and 20 are used to allow and deny traffic for NodePort services)

The following CNI plug-ins are tested for the above requirements:

- Calico (3.13) with the following Felix configurations: (*ChainInsertMode: Append, IptablesRefreshInterval: 0*) or (*ChainInsertMode: Insert, IptablesFilterAllowAction: Return, IptablesMangleAllowAction: Return, IptablesRefreshInterval: 0*). All other options use their default values.

For more information on setting these options, see the Felix configuration reference.

Install Kubernetes or OpenShift Agent using the Agent Script Installer Method



Note The agent script installer method automatically installs agents on nodes included later.

Procedure

- Step 1** Navigate to the Agent Installation Methods:
- If you are a first-time user, launch the Quick Start wizard and click **Install Agents**.
 - From the navigation pane, choose **Manage > Agents**, and select the **Installer** tab.
- Step 2** Click **Agent Script Installer**.
- Step 3** From the **Select Platform** drop-down menu, choose **Kubernetes**.
- To view the supported Kubernetes or OpenShift platforms, click **Show Supported Platforms**.
- Step 4** Choose the tenant to install the agents.
- Note** Selecting a tenant is not required for Secure Workload SaaS clusters.
- Step 5** If HTTP proxy is required to communicate with Secure Workload, choose **Yes**, and then enter a valid proxy URL.
- Step 6** Click **Download** and save the file to the local disk.
- Step 7** Run the installer script on a Linux machine which has access to the Kubernetes API server and a kubectl configuration file with administrative privileges as the default context/cluster/user.
- The installer attempts to read the file from its default location (`~/.kube/config`). However, you can explicitly specify the location of the config file using the `--kubeconfig` command.

The installation script provides instructions for verifying the Secure Workload Agent Daemonset and the Pods that were installed.



Note The HTTP Proxy configured on the agent installer page prior to download only controls how Secure Workload agents connect to the Secure Workload cluster. This setting does not affect how Docker images are fetched by Kubernetes or OpenShift nodes, because the container runtime on those nodes uses its own proxy configuration. If the Docker images are not fetched from the Secure Workload cluster, debug the image pulling process of the container and add a suitable HTTP proxy.

Installing Solaris Agents for Deep Visibility

Requirements and Prerequisites for Installing Solaris Agents

- See [Supported Platforms and Requirements](#).
- Root privileges to install and execute the services.
- One GB storage space for agent and log files.
- Configuration of security exclusions on security applications that are monitoring the host, to prevent other security applications from blocking of agent installation or agent activity. For more information, see [Security Exclusions](#).

Install Solaris Agent using the Agent Script Installer Method

The installed Solaris agent supports both deep visibility and process or package visibility.

Procedure

Step 1 Navigate to Agent Installation Methods:

- If you are a first-time user, launch the Quick Start wizard and click **Install Agents**.
- From the navigation pane, choose **Manage > Agents**, and select the **Installer** tab.

Step 2 Click **Agent Script Installer**.

Step 3 From the **Select Platform** drop-down menu, choose **Solaris**.

To view the supported Solaris platforms, click **Show Supported Platforms**.

Step 4 Choose the tenant to install the agents.

Note Tenant selection is not required for Secure Workload SaaS clusters.

Step 5 If you want to assign labels to the workload, choose the label keys and enter label values.

When the installed agent reports IP addresses on the host, the installer CMDB labels selected here, along with other uploaded CMDB labels that have been assigned to IPs reported by this host, would be assigned automatically to the new IP address. If there are conflicts between uploaded CMDB labels and installer CMDB labels:

- Labels assigned to an exact IP address take precedence over labels assigned to the subnet.
- Existing labels assigned to an exact IP address take precedence over installer CMDB labels.

Step 6 If HTTP proxy is required to communicate with Secure Workload, choose **Yes**, and then enter a valid proxy URL.

Step 7 Under the **Installer expiration** section, select one from the available options:

- No expiration: The installer script can be used multiple times.
- One time: The installer script can be used only once.

- Time bound: You can set the number of days for which the installer script can be used.
- Number of deployments: You can set the number of times the installer script can be used.

Step 8 Click **Download** and save the file to the local disk.

Step 9 Copy the installer shell script on Solaris hosts and run the following command to grant execute permission to the script: `chmod u+x tetration_installer_default_sensor_solaris.sh`

Note The script name may differ depending on the selected agent type and scope.

Step 10 To install the agent, run the following command with root privileges:

```
./tetration_installer_default_sensor_solaris.sh
```

Note If an agent is already installed on the tenant, you cannot proceed with the installation.

We recommend running the precheck, as specified in the script usage details.

Solaris installer script usage details:

```
tetration_installer_default_sensor_solaris.sh [--pre-check] [--skip-pre-check=<option>]
[--no-install] [--logfile=<filename>] [--proxy=<proxy_string>] [--no-proxy] [--help]
[--version] [--sensor-version=<version_info>] [--ls] [--file=<filename>] [--save=<filename>]
[--new] [--reinstall] [--unpriv-user] [--force-upgrade] [--upgrade-local]
[--upgrade-by-uuid=<filename>] [--basedir=<basedir>] [--logbasedir=<logbdir>]
[--tmpdir=<tmp_dir>] [--visibility] [--golden-image]
--pre-check: run pre-check only
--skip-pre-check=<option>: skip pre-installation check by given option; Valid options
include 'all', 'ipv6' and 'enforcement'; e.g.: '--skip-pre-check=all' will skip all
pre-installation checks; All pre-checks will be performed by default
--no-install: will not download and install sensor package onto the system
--logfile=<filename>: write the log to the file specified by <filename>
--proxy=<proxy_string>: set the value of CL_HTTPS_PROXY, the string should be formatted
as http://<proxy>:<port>
--no-proxy: bypass system wide proxy; this flag will be ignored if --proxy flag was
provided
--help: print this usage
--version: print current script's version
--sensor-version=<version_info>: select sensor's version; e.g.: '--sensor-version=3.4.1.0';
will download the latest version by default if this flag was not provided
--ls: list all available sensor versions for your system (will not list pre-3.1 packages);
will not download any package
--file=<filename>: provide local zip file to install sensor instead of downloading it
from cluster
--save=<filename>: download and save zip file as <filename>
--new: remove any previous installed sensor; previous sensor identity has to be removed
from cluster in order for the new registration to succeed
--reinstall: reinstall sensor and retain the same identity with cluster; this flag has
higher priority than --new
--unpriv-user=<username>: use <username> for unpriv processes instead of nobody
--force-upgrade: force sensor upgrade to version given by --sensor-version flag; e.g.:
'--sensor-version=3.4.1.0 --force-upgrade'; apply the latest version by default if
--sensor-version flag was not provided
--upgrade-local: trigger local sensor upgrade to version given by --sensor-version flag;
e.g.: '--sensor-version=3.4.1.0 --upgrade-local'; apply the latest version by default if
--sensor-version flag was not provided
--upgrade-by-uuid=<filename>: trigger sensor whose uuid is listed in <filename> upgrade
to version given by --sensor-version flag; e.g.: '--sensor-version=3.4.1.0
--upgrade-by-uuid=/usr/local/tet/sensor_id'; apply the latest version by default if
--sensor-version flag was not provided
```

```

--logbasedir=<log_base_dir>: instead of logging to /opt/cisco/secure-workload/log use
<log_base_dir>. The full path will be <log_base_dir>/secure-workload
--tmpdir=<tmp_dir>: instead of using /tmp use <tmp_dir> as temp directory
--visibility: install deep visibility agent only; --reinstall would overwrite this flag
if previous installed agent type was enforcer
--golden-image: install Cisco Secure Workload Agent but do not start the Cisco Secure
Workload Services; use to install Cisco Secure Workload Agent on Golden Images in VDI
environment or Template VM. On VDI/VM instance created from golden image with different
host name, Cisco Secure Workload Services will work normally

```

Verify Solaris Agent Installation

Procedure

- Step 1** Run the command: `sudo pkg list tet-sensor`
- Step 2** A single entry as output confirms that a Solaris agent is installed on the host.
Sample output:

| NAME (PUBLISHER) | VERSION | IFO |
|--------------------|---------|-----|
| tet-sensor (cisco) | 3.8.1.1 | i-- |

Note The specific output may differ depending on the platform and architecture.

(Manual Installations Only) Update the User Configuration File

The following procedure is required only for installations involving *all* of the following:

- Secure Workload SaaS, or on-premises clusters with multiple tenants (on-premises clusters that use only the default tenant do NOT need this procedure)
- Manual installation
- Linux or Windows platform

Agents require an activation key to register to the Secure Workload cluster. they require a cluster activation key. Additionally, they might need an HTTPS proxy to reach the cluster.



Note In Windows Environment, you do not need to manually configure user.cfg, if activationkey and proxy options are used during manual installation.

Before installation, configure the required variables in the user configuration file:

Procedure

- Step 1** To retrieve your activation key, navigate to **Manage > Agents**, click the **Installer** tab, click **Manual Install using classic packaged installers**, then click **Agent Activation Key**.

- Step 2** Open the `user.cfg` file in the Secure Workload Agent installation folder. (Example: `/usr/local/tet` on Linux or `C:\Program Files\Cisco Tetration` on Windows). The file contains a list of variables in the form of “key=value”, one on each line.
- Step 3** Add the activation key to the `ACTIVATION_KEY` variable. Example:
`ACTIVATION_KEY=7752163c635ef62e6568e9e852d07bd21bfd60d0`
- Step 4** If the agent requires an HTTPS proxy, add the `http` protocol proxy server and port using the `HTTPS_PROXY` variable. Example: `HTTPS_PROXY=http://proxy.my-company.com:80`
-

Other Agent-Like Tools

AnyConnect Agents

No Secure Workload agent is required for platforms supported by Cisco AnyConnect Secure Mobility agent with Network Visibility Module (NVM). AnyConnect connector registers these agents and exports flow observations, inventories, and labels to Secure Workload. For more information, see [AnyConnect Connector](#).

For Windows, Mac, or Linux platforms, see [Cisco AnyConnect Secure Mobility Client Data Sheet](#).

ISE Agents

A Secure Workload agent on the endpoint is not required for endpoints registered with Cisco Identity Service Engine (ISE). ISE connector collects metadata about endpoints from ISE through pxGrid service on ISE appliance. It registers the endpoints as ISE agents on Secure Workload and pushes labels for the inventories on these endpoints. For more information, see [ISE Connector](#).

SPAN Agents

SPAN agents work with the ERSPAN connector. For more information, see [ERSPAN Connector](#).

Third-Party and Additional Cisco Products

- For integrations using external orchestrators configured in Secure Workload, see [External Orchestrators in Secure Workload, on page 123](#).
- For integrations using connectors configured in Secure Workload, see [What are Connectors](#).

Connectivity Information

In general, when the agent is installed on the workload, it makes several network connections to the back-end services hosted on the Secure Workload cluster. The number of connections will vary depending on the agent type and its functions.

The following table captures various permanent connections that are made by various agent types.

Table 2: Agent Connectivity

| Agent type | Config server | Collectors | Enforcement backend |
|---------------------------|-------------------|-------------------|---------------------|
| visibility (on-premises) | CFG-SERVER-IP:443 | COLLECTOR-IP:5640 | N/A |
| visibility (SaaS) | CFG-SERVER-IP:443 | COLLECTOR-IP:443 | N/A |
| enforcement (on-premises) | CFG-SERVER-IP:443 | COLLECTOR-IP:5640 | ENFORCER-IP:5660 |
| enforcement (SaaS) | CFG-SERVER-IP:443 | COLLECTOR-IP:443 | ENFORCER-IP:443 |
| docker images | CFG-SERVER-IP:443 | N/A | N/A |

Legends:

- CFG-SERVER-IP is the IP address of the config server.
- COLLECTOR-IP is the IP address of the collector. Deep visibility and enforcement agents connect to all available collectors.
- ENFORCER-IP is the IP address of the enforcement endpoint. The enforcement agent connects to only one of the available endpoints.
- For Kubernetes/Openshift agent deployments, the installation script does not contain the agent software - Docker images containing the agent software are pulled from the Secure Workload cluster by every Kubernetes/Openshift node. These connections are established by the container run time image fetch component and directed at CFG-SERVER-IP:443.

Navigate to **Platform > Cluster Configuration** to know the config server IP and collector IP.

- **Sensor VIP** is for the config server IP: The IP address that has been set up for the config server in this cluster.
- **External IPs** are for collectors IPs and enforcer: If this is populated, when assigning external cluster IP addresses, the selection process is restricted to only IP addresses defined in this list, that are part of the external network.



Note

- The Secure Workload agent always acts as a client to initiate the connections to the services hosted within the cluster, and never opens a connection as a server.
- Agents, for which upgrade is supported, periodically perform HTTPS requests (port 443) to the cluster sensor VIP to query for available packages.
- An agent can be located behind a NAT server.

Connections to the cluster might be denied if the workload is behind a firewall, or if the host firewall service is enabled. In such cases, administrators must create appropriate firewall policies to allow the connections.

Security Exclusions

Software agents continuously interact with the host operating system during their normal operations. This operation may cause other security applications installed on the host such as antivirus, security agents, and others, to raise alarms or block the actions of Secure Workload agents. Therefore, to ensure that agents are installed successfully and are functioning, you must configure the necessary security exclusions on the security applications that are monitoring the host.

Table 3: Security Exclusions for Agent Directories

| Host OS | Directories |
|---------|--|
| AIX | /opt/cisco/tetration |
| Linux | /usr/local/tet or /opt/cisco/tetration or <user chosen inst dir> |
| | /var/opt/cisco/secure-workload |
| Windows | C:\Program Files\Cisco Tetration |
| | C:\ProgramData\Cisco Tetration |
| Solaris | /opt/cisco/secure-workload |

Table 4: Security exclusions for Agent Processes

| Host OS | Processes |
|---------|---------------|
| AIX | csw-agent |
| | tet-sensor |
| | tet-enforcer |
| | tet-main |
| Linux | csw-agent |
| | tet-sensor |
| | tet-enforcer |
| | tet-main |
| | enforcer |
| Windows | CswEngine.exe |
| | TetEnfC.exe |

| Host OS | Processes |
|---------|--------------|
| Solaris | csw-agent |
| | tet-sensor |
| | tet-enforcer |
| | tet-main |

Table 5: Security Exclusions for Agent Actions

| Host OS | Actions |
|---------|--|
| AIX | Access /dev/bpf*, /dev/ipl, /dev/kmem |
| | Invokes cfg_ipf, curl, ipf, ippool, ipfstat lslpp, lsfilt, prtconf |
| | Scan /proc |
| Linux | Invokes curl, ip[6]tables-save, ip[6]tables-restore, rpm/dpkg |
| | Scan /proc, open netlink sockets |
| Windows | Access registry |
| | Register to firewall events |
| | Invokes c:\windows\system32\netsh.exe |
| Solaris | Invokes curl, lssp, pkg, smbios |
| | Scan /proc |

Table 6: Security Exclusions for Agents Scripts or Binaries Executions

| Host OS | Invoked scripts/binaries |
|---------|--------------------------|
| AIX | - |
| Linux | - |
| Windows | dmidecode.exe |
| | npcap-installer.exe |
| | sensortools.exe |
| | signtool.exe |
| Solaris | - |

Service Management of Agents

Software agents are deployed as a service in all supported platforms. This section describes methods to manage the services for various functions and platforms.



Note Unless specified otherwise, all the commands in this section require root privileges on Linux or Unix, or administrative privileges on Windows to run.

Service Management for RHEL, CentOS, OracleLinux-6.x, and Ubuntu-14

Run the following commands for:

- **Starting a service:** `start csw-agent`
- **Stopping a service:** `stop csw-agent`
- **Restarting a service:** `restart csw-agent`
- **Checking service status:** `status csw-agent`

Service Management for RHEL, CentOS, OracleLinux-7.x and Later

The commands are also applicable to:

- AlmaLinux, Rocky Linux- 8.x and later
- Amazon Linux 2 and later
- Debian 8 and later
- SLES-12SPx and later
- Ubuntu-16.04 and later

Run the following commands for:

- **Starting a service:** `systemctl start csw-agent`
- **Stopping a service:** `systemctl stop csw-agent`
- **Restarting a service:** `systemctl restart csw-agent`
- **Checking service status:** `systemctl status csw-agent`

Service Management for Windows Server or Windows VDI

Run the following commands for:

- **Starting a service:** `net start <service-name>`

Example: **net start cswagent** for deep visibility and enforcement service

- **Stopping a service:** `net stop <service-name>`
Example: **net stop cswagent** for deep visibility and enforcement service
- **Restarting a service:**
 1. `net stop <service-name>`
 2. `net start <service-name>`
- **Checking service status:** `sc query <service-name>`
Example: **sc query cswagent** for deep visibility and enforcement service

Service Management for AIX

Run the following commands for:

- **Starting a service:** `startsrc -s csw-agent`
- **Stopping a service:** `stopsrc -s csw-agent`
- **Restarting a service:**
 1. `stopsrc -s csw-agent`
 2. `startsrc -s csw-agent`
- **Checking service status:** `lssrc -s csw-agent`

Service Management for Kubernetes Agent Installations

- **Starting or stopping a service:** It is not possible to start or stop the agents on a specific node because they are not installed as individual services, but as a cluster-wide daemon set.
- **Restarting an agent on a node:** Locate the Secure Workload agent pod on the node and run the appropriate Kubernetes command to kill it. The pod is automatically restarted.
- **Checking the status of pods:** `kubectl get pod -n tetration` OR `oc get pod -n tetration` (for OpenShift) lists the status of all Secure Workload agent pods in the Kubernetes cluster.

Service Management for Solaris

Run the following commands for:

- **Starting a service:** `svcadm enable csw-agent`
- **Stopping a service:** `svcadm disable csw-agent`
- **Restarting a service:** `svcadm restart csw-agent`
- **Checking service status:** `svcs -l csw-agent`

Policy Enforcement with Agents

By default, agents that are installed on your workloads have the capability to enforce policy, but enforcement is disabled. When you are ready, you can enable these agents to enforce policy on selected hosts that are based on the configured intent.

When an agent enforces a policy, it applies an ordered set of rules that specify whether the firewall should ALLOW or DROP specific network traffic that is based on parameters such as the source, destination, port, protocol, and direction. For more information on policies, see [Manage Policies Lifecycle in Secure Workload, on page 417](#).

Enforcement using agents

- Agents receive policies over a secured TCP or SSL channel.
- Agents run in a privileged domain. On Linux machines, the agent runs as root; on Windows machines, the agent runs as SYSTEM.
- Depending on the platform, when policy enforcement is enabled, agents can completely control the firewall or work with existing configured rules.
- For details about enforcement options and to enable and configure agents to enforce policies, see [Create an Agent Configuration Profile, on page 67](#).

Advanced details

When you enable enforcement, golden rules are formulated to allow the agent to connect to the controller. Agents communicate with the Enforcement Front End (EFE) of the controller through a bidirectional and secure channel using the TLS or SSL protocol. Messages from the controller are signed by the policy generator and verified by the agent.

The agent receives policies in a platform-independent schema from the controller. The agent converts these platform-independent policies into platform-specific policies and programs the firewall on the endpoint.

The agent actively monitors the firewall state. If the agent detects any deviation in the enforced policies, it enforces the cached policies into the firewall again. The agent also monitors its own consumption of system resources such as CPU and memory.

The agent periodically sends a status and stats report to the controller using EFE. The status report includes the status of the latest programmed policies such as success, failure, or error, if any. The stats report includes the policy stats such as allowed and dropped packets, and byte count depending on the platform.

Agent Enforcement on the Linux Platform

On the Linux platform, the agent uses the iptables, ip6tables, or ipset to enforce network policies. After the agent is enabled on the host, by default, it controls, and programs iptables. If the IPv6 network stack is enabled, then the agent controls the IPv6 firewall using ip6tables.

Linux iptables or ip6tables

The Linux kernel has iptables and ip6tables that are used to set up, maintain, and inspect the tables of IPv4 and IPv6 packet filter rules. The iptables and ip6tables consist of many predefined tables. Each table contains predefined chains and can also contain user-defined chains. These chains contain sets of rules and each of

these rules specifies the match criteria for a packet. Predefined tables include raw, mangle, filter, and NAT. Predefined chains include INPUT, OUTPUT, FORWARD, PREROUTING, and POSTROUTING.

The Secure Workload agent programs a filter table that contains rules to allow or drop packets. The filter table consists of the predefined chains INPUT, OUTPUT, and FORWARD. Along with these, the agent adds custom TA chains to categorize and manage the policies from the controller. These TA chains contain Secure Workload rules that are derived from the policies along with rules that are generated by the agent. When the agent receives platform-independent rules, it parses and converts them into iptable, ip6table, or ipset rules and inserts these rules into TA defined chains in the filter table. After programming the firewall, the agent monitors the firewall for any rule or policy deviation and if so, reprograms the firewall. It keeps track of the policies that are programmed in the firewall and reports their stats periodically to the controller.

Here is an example to depict this behavior:

A typical policy in a platform-independent network policy message consists of:

```
source set id: "test-set-1"
destination set id: "test-set-2"
source ports: 20-30
destination ports: 40-50
ip protocol: TCP
action: ALLOW
. . .
set_id: "test-set-1"
  ip_addr: 1.2.0.0
  prefix_length: 16
  address_family: IPv4
set_id: "test-set-2"
  ip_addr: 3.4.0.0
  prefix_length: 16
  address_family: IPv4
```

Along with other information, the agent processes this policy and converts it into platform-specific ipset and iptables rule:

```
ipset rule:
Name: ta_f7b05c30ffa338fc063081060bf3
Type: hash:net
Header: family inet hashsize 1024 maxelem 65536
Size in memory: 16784
References: 1
Members:
1.2.0.0/16
Name: ta_1b97bc50b3374829e11a3e020859
Type: hash:net
Header: family inet hashsize 1024 maxelem 65536
Size in memory: 16784
References: 1
Members:
3.4.0.0/16
iptables rule:
TA_INPUT -p tcp -m set --match-set ta_f7b05c30ffa338fc063081060bf3 src -m set --match-
.→set ta_1b97bc50b3374829e11a3e020859 dst -m multiport --sports 20:30 -m multiport --
.→dports 40:50 -j ACCEPT
```

Caveats

ipset Kernel Module

When enforcement is enabled and preserve rules is disabled in the Agent Config profile, the agents running on Linux hosts ensures that the ipset kernel module has a sufficiently large *max_sets* configuration. In case a

change is needed, the agent reloads the ipset kernel module with a new *max_sets* value. If Preserve Rules is enabled, the agents check the current ipset module *max_sets* value, but does not make any change. The current configured *max_sets* value can be found in `cat /sys/module/ip_set/parameters/max_sets`.

Host Firewall Backup

The first time that enforcement is enabled in the Agent Config profile, the agents running on Linux hosts, store the current content of ipset and ip[6]tables in `/opt/cisco/tetration/backup` before taking control of the host firewall.

Successive disable or enable transitions of enforcement configuration do not generate backups. The directory is not removed after agent uninstallation.

Agent Enforcement on the Windows Platform in WAF mode

On the Windows platform, the Secure Workload agent uses the Windows Firewall to enforce network policies.

Windows Firewall with Advanced Security

A native component on Windows, the Windows Firewall with Advanced Security, regulates network traffic that is based on the following types of settings:

- Rules that regulate inbound network traffic.
- Rules that regulate outbound network traffic.
- Override rules that is based on the authentication status of the source and destination of the network traffic.
- Rules that apply to IPsec traffic and to Windows services.

The Secure Workload Network Policy is programmed using inbound and outbound firewall rules.

Secure Workload Rules and the Windows Firewall

On the Windows platform, the Secure Workload Network Policy is enforced as follows:

1. The platform-independent firewall rules from the Secure Workload Network policy are translated into Windows Firewall rules.
2. The rules are programmed in Windows Firewall.
3. The Windows Firewall enforces the rules.
4. The Windows Firewall and its ruleset are monitored. If a change is detected, the deviation is reported and the Secure Workload Network policy is reset in the Windows Firewall.

Security Profiles

Windows Firewall groups the rules based on the network that the host is connected to. These rule groups are called Profiles and there are three such profiles:

- Domain Profile
- Private Profile

- Public Profile

The Secure Workload rules are programmed into all the profiles, but only rules in active profiles are continuously monitored.

Effective Setting and Mixed-List Policies

The set of rules in the Windows Firewall is not ordered based on precedence. When multiple rules match a packet, the most restrictive of those rules take effect meaning that DENY rules take precedence over ALLOW rules. For more information, see the article on [Microsoft TechNet](#).

Consider the mixed-list, both allow and deny, policy example from the Enforcement Agent section:

```
1. ALLOW 1.2.3.30 tcp port 80
2. ALLOW 1.2.3.40 udp port 53
3. BLOCK 1.2.3.0/24 ip
4. ALLOW 1.2.0.0/16 ip
5. Catch-all: DROP ingress, ALLOW egress
```

When a packet headed for the host 1.2.3.30 TCP port 80 reaches the firewall, it matches all the rules, but the most restrictive of them all, Rule number 3, is the one that will be enforced and the packet will be dropped. This behavior is contrary to the expectation that the rules will be evaluated in order, Rule 1 is the rule that is enforced, and that the packet will be allowed.

This difference in behavior is expected in the Windows platform owing to the design of the Windows Firewall described above. This behavior can be observed in mixed-list policies with overlapping rules that have different rule actions.

For example,

```
1. ALLOW 1.2.3.30 tcp
2. BLOCK 1.2.3.0/24 tcp
```

Interference from Other Firewalls or Policies

We recommend that you grant the agent full and exclusive control of the Windows Firewall to enforce the Secure Workload Network Policy as intended. Agents cannot reliably enforce the policy if:

- A third-party firewall is present. (The Windows Firewall is required to be the active firewall product on the host.)
- The Firewall is disabled for the current profiles.
- Conflicting firewall settings are deployed using Group Policy. Some of the conflicting settings are:
 - Firewall rules.
 - Default inbound or outbound actions in the current profiles that differ from the catch-all rules of the policy.
 - Firewall disabled for the current profiles.

Stateful Enforcement

Windows Advanced Firewall is considered as a **stateful** firewall, that is for certain protocols such as TCP, the firewall maintains internal state tracking to detect if a new packet hitting the firewall belongs to a known connection. Packets belonging to a known connection are allowed without the firewall rules having to be

examined. A stateful firewall enables bidirectional communication without rules having to be established in the INBOUND and OUTBOUND tables.

For example, consider the following rule for a web server: **Accept all TCP connections to port 443**

The intention is to accept all TCP connections on port 443 to the server, and allow the server to communicate back to the clients. In this case, only one rule is inserted in the INBOUND table, allowing TCP connections on port 443. No rule is required to be inserted in the OUTBOUND table. Inserting a rule in the OUTBOUND table is implicitly done by the Windows Advanced Firewall.



Note Stateful tracking applies only to protocols that establish and maintain explicit connections. For other protocols, both INBOUND and OUTBOUND rules must be programmed to enable bidirectional communication.

When enforcement is enabled, a given concrete rule is programmed as **stateful** when the protocol is TCP (the agent decides, based on the context, whether the rule is to be inserted in the INBOUND table or the OUTBOUND table). For other protocols (including **ANY**), both INBOUND and OUTBOUND rules are programmed.

Caveats

Host firewall backup

When enforcement is enabled for the first time in the Agent Config profile, the agents running on Windows hosts, before taking control of the host firewall, export the current Windows Advanced Firewall content to `ProgramData\Cisco\Tetration\backup`. Successive disable or enable transitions of Enforcement configuration do not generate backups. The directory is not removed upon agent uninstallation.

Agent Enforcement on the Windows Platform in WFP Mode

On the Windows platform, the agent enforces the network policies by programming Windows Filtering Platform (WFP) filters. Windows Advanced Firewall is not used to configure the network policy.

Windows Filtering Platform

Windows Filtering Platform (WFP) is a set of APIs provided by Microsoft to configure filters for processing network traffic. Network traffic processing filters are configured using kernel-level APIs and user level APIs. WFP filters can be configured at various layers, Network Layer, Transport Layer, Application Layer Enforcement (ALE). Secure Workload WFP filters are configured at the ALE layer, similar to Windows firewall rules. Each layer has several sublayers, ordered by weight, from highest to lowest. Within each sublayer, filters are ordered by weight, from highest to lowest. A network packet traverses through all the sublayers. At each sublayer, the network packet traverses through the matching filters that are based on weight, from highest to lowest and returns the action: Permit or Block. After passing through all the sublayers, the packet is processed based on the rule that Block action overrides Permit.

Advantages of WFP over WAF

- Avoids Windows Firewall configuration dependencies.
- Overcomes GPO restrictions.
- Ensures ease of migration and policy reversion.

- Allows you to control policy ordering.
- Avoids strict block-first policy order of Windows Firewall.
- Reduces CPU overhead on policy update.
- Creates an efficient 1:1 policy rule filter.
- Ensures a faster single-step update.

Agent Support for WFP

When enforcement is configured to use WFP, Secure Workload filters override Windows Firewall rules.

In WFP mode, the agent configures the following WFP objects:

- Provider has a GUID and name, is used for filter management, and does not affect packet filtering
- Sublayer has a GUID, name, and weight. The Secure Workload sublayer is configured with higher weight than the Windows Advanced Firewall sublayer.
- Filter has name, GUID, ID, weight, layer ID, sublayer key, action (PERMIT/ BLOCK), and conditions. WFP filters are configured for Golder rules, Self Rules, and Policy Rules. The agent also configures the port scanning prevention filters. Secure Workload filters are configured with the `FWPM_FILTER_FLAG_CLEAR_ACTION_RIGHT` flag. This flag ensures that Secure Workload filters are not overridden by Microsoft Firewall rules. For each Secure Workload Network policy rule, one or more WFP filters are configured based on the direction (inbound or outbound) and protocol.

For TCP inbound policy,

```
id: 14 , TCP Allow 10.195.210.184 Dir=In localport=3389
```

The WFP filters configured are:

```
Filter Name:                Secure Workload Rule 14
-----
EffectiveWeight:            18446744073709551589
LayerKey:                   FWPM_LAYER_ALE_AUTH_LISTEN_V4
Action:                     Permit
Local Port:                 3389
Filter Name:                Secure Workload Rule 14
-----
EffectiveWeight:            18446744073709551589
LayerKey:                   FWPM_LAYER_ALE_AUTH_RECV_ACCEPT_V4
Action:                     Permit
RemoteIP:                   10.195.210.184-10.195.210.184
```

Secure Workload agent configures **Secure Workload Default Inbound** and **Secure Workload Default Outbound** filters for inbound and outbound CATCH-ALL policy respectively.

Agent WFP support and Windows Firewall

- The agent **does not monitor** WAF rules or WAF profiles.
- The agent **does not monitor** firewall states.
- The agent **does not require** firewall state to be enabled.
- The agent **does not conflict** with GPO policies.

Effective Setting and Mixed-List Policies

Agent enforcement in WFP mode supports mixed-list or grey list policies.

Consider the mixed-list (both allow and deny) policy example from the Enforcement Agent section:

```
1. ALLOW 1.2.3.30 tcp port 80-          wt1000
2. BLOCK 1.2.3.0/24 ip-                wt998
3. ALLOW 1.2.0.0/16 ip-                wt997
4. Catch-all: DROP ingress, ALLOW egress - wt996
```

When a packet headed for the host 1.2.3.30 tcp port 80 reaches the firewall, it matches filter 1 and is allowed. However, a packet that is headed for the host 1.2.3.10 is blocked because of filter 2. A packet that is headed for host 1.2.2.10 is allowed by filter 3.

Stateful Enforcement

Secure Workload WFP filters are configured at the ALE layer. Network traffic is filtered for socket connect(), listen(), and accept() operations. Network packets related to a L4 connection are not filtered after the connection is established.

Visibility of Configured WFP Filters

You can view the configured Secure Workload WFP filters using `c:\program files\tetration\tetenf.exe`. Supported options:

- With administrative privileges, run `cmd.exe`.
- Run `c:\program files\tetration\tetenf.exe -l -f <-verbose> <-output=outfile.txt>`.

OR

- With administrative privileges, run `cmd.exe`.
- Run `netsh wfp show filters`.
- View the **filters.xml** file for configured Secure Workload filters.

Disable Stealth Mode Filters in WFP Mode

To disable stealth mode filters (port scanning filters):

Procedure

-
- Step 1** Edit `\conf\enforcer.cfg`.
 - Step 2** Add `disable_wfp_stealth_mode: 1`
 - Step 3** Save the file.
 - Step 4** With administrative privileges, restart the CswAgent service by:
 - a) Run the command: `sc stop cswagent` to stop the CswAgent Service.
 - b) Run the command: `sc start cswagent` to start the CswAgentService.
 - Step 5** To verify:
 - a) With administrative privileges, run `cmd.exe`.

- b) Run the command: `c:\program files\tetration\tetenf.exe -l -f <-verbose> <-output=outfile.txt>`.

"Tetration Internal Rule block portscan" filters are not configured.

Delete Configured WFP Filters

You can delete the configured Secure Workload WFP filters using `c:\program files\tetration\tetenf.exe`. To avoid accidental deletions of filters, when you run the delete command, specify the token in `<yyyymm>` format, where `yyyy` is the current year and `mm` is the current month in the numerical form. For example, if today's date is 01/21/2021, then the token is **-token=202101**

Supported options are:

- With administrative privileges, run `cmd.exe`.
- To delete all configured Secure Workload filters, run `c:\program files\tetration\tetenf.exe -d -f -all - token=<yyyymm>`
- To delete all configured Secure Workload WFP objects, run `c:\program files\tetration\tetenf.exe -d -all -token=<yyyymm>`
- To delete a Secure Workload WFP filter by name, run `c:\program files\tetration\tetenf.exe -d -name=<WFP filter name> -token=<yyyymm>`

Known Limitations in WFP Mode

- The **Preserve Rules** setting in Agent Config Profile has no effect when you set Enforcement Mode to WFP.

Configure Policies for Windows Attributes

For more granularity when enforcing a policy on Windows-based workloads, you can filter network traffic by:

- Application Name
- Service Name
- User Names with or without User Groups

This option is supported in both WAF and WFP modes. Windows OS-based filters are categorized as *consumer filters* and *provider filters* in the generated network policy. The Consumer filters filter the network traffic that is initiated on the consumer workload and Provider filters filter the network traffic that is destined for the provider workload.

Before you begin

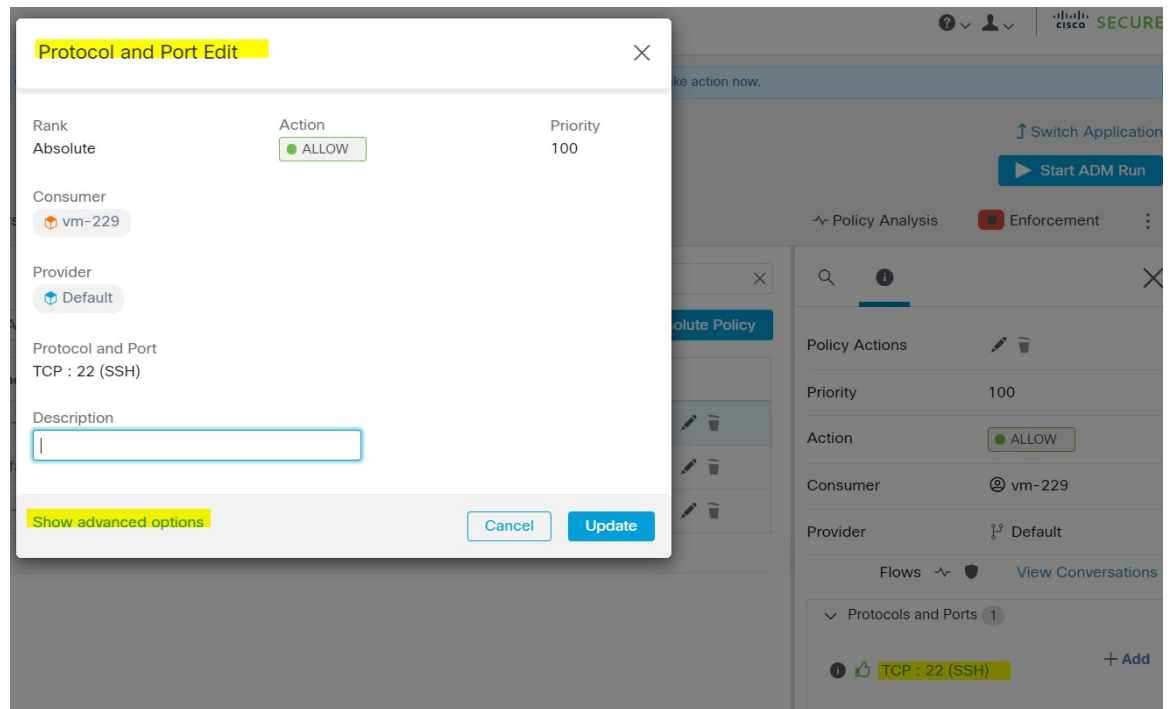
This procedure assumes you are modifying an existing policy. If you have not yet created the policy to which you want to add a Windows OS-based filter, create that policy first.



Important See [Caveats, on page 52](#) and [Known limitations, on page 51](#) for policies involving Windows attributes.

Procedure

- Step 1** In the navigation pane, click **Defend > Segmentation**.
- Step 2** Click the scope that contains the policy for which you want to configure Windows OS-based filters.
- Step 3** Click the workspace in which you want to edit the policy.
- Step 4** Click **Manage Policies**.
- Step 5** Choose the policy to edit.
- Important** Consumer and Provider must include only Windows workloads.
- Step 6** In the table row for the policy to edit, click the existing value in the **Protocols and Ports** column.
- Step 7** In the pane on the right, click the existing value under **Protocols and Ports**.
- In the example, click **TCP : 22 (SSH)**.



- Step 8** Click **Show advanced options**.

While using process level controls a consumer/provider scope or filter should only contain Windows agents. Otherwise, non-Windows OSs (Linux, AIX) will skip the policy and report a sync error in Enforcement Status. See the [user guide](#) for more information.

Consumer Service

Consumer Binary Path

Consumer Users or User Groups ⓘ

Provider Service

Provider Binary Path

Provider Users or User Groups ⓘ

Hide advanced options

Step 9 Configure consumer filters based on Application name, Service name, or User name.

- The application name must be a full pathname.
- Service name must be a short service name.
- User name can be a local user name (For example, `tetter`) or domain user name (For example, `sensor-dev@sensor-dev.com` or `sensor-dev\sensor-dev`)
- User group can be local user group (For example, `Administrators`) or domain user group (For example, `domain users\sensor-dev`)
- Multiple user names and/ or user group names can be specified, separated by ",".(For example, `sensor-dev@sensor-dev.com,domain users\sensor-dev`)
- Service name and User name cannot be configured together.

Step 10 Configure provider filters based on Application name, Service name, or User name.

Follow the same guidelines as given for consumer filters in the previous step.

Step 11 Enter the paths to the binary, as applicable.

For example, enter `c:\test\putty.exe`

Step 12 Click **Update**.

Recommended Windows OS-Based Policy Configuration

Always specify ports and protocols in policies when possible; we recommend not to allow ANY port, ANY protocol.

For example, a generated policy with port and protocol restrictions might look like this:

```
dst_ports {
  start_port: 22
  end_port: 22
  consumer_filters {
    application_name: "c:\\test\\putty.exe"
  }
}
ip_protocol: TCP
```

In contrast, if you allow network connections that are initiated by iperf.exe with ANY protocol and ANY port, the generated policy looks like this:

```
match_set {
  dst_ports {
    end_port: 65535
    consumer_filters {
      application_name: "c:\\test\\iperf.exe"
    }
  }
  address_family: IPv4
  inspection_point: EGRESS
  match_comment: "PolicyId=61008290755f027a92291b9d:61005f90497d4f47cedacb86:"
}
```

For the above filter, Secure Workload creates a policy rule to allow the network traffic on the provider as follows:

```
match_set {
  dst_ports {
    end_port: 65535
  }
  address_family: IPv4
  inspection_point: INGRESS
  match_comment: "PolicyId=61008290755f027a92291b9d:61005f90497d4f47cedacb86:"
}
```

This network rule opens all the ports on the Provider. We strongly recommend not to create OS-based filters with *Any* protocol.

Known limitations

- Windows 2008 R2 does not support Windows OS based filtering policies.
- Network policy can be configured with a single user name whereas MS Firewall UI supports multiple users.

Caveats

- While using the Windows OS-based policies, a consumer/ provider scope or filter should only contain Windows agents. Otherwise, non-Windows OSs (Linux, AIX) skip the policy and report a sync error in Enforcement Status.
- Avoid creating Windows OS filters with *loose* filtering criteria. Such criteria may open unwanted network ports.
- If OS filters are configured for consumer, then the policies are applicable only to consumer, similarly if it is configured for provider then it is applicable only to provider.
- Due to limited or no knowledge of the process context, user context or service context of the network flows, there will be discrepancy in the policy analysis if the policies have Windows OS-based filters.

Verify and Troubleshoot Policies with Windows OS-Based Filtering Attributes

If you use Windows OS-based filtering attributes, the following topics provide you with verification and troubleshooting information.

Cisco TAC can use this information as needed to troubleshoot such policies.

Policies Based on Application Name

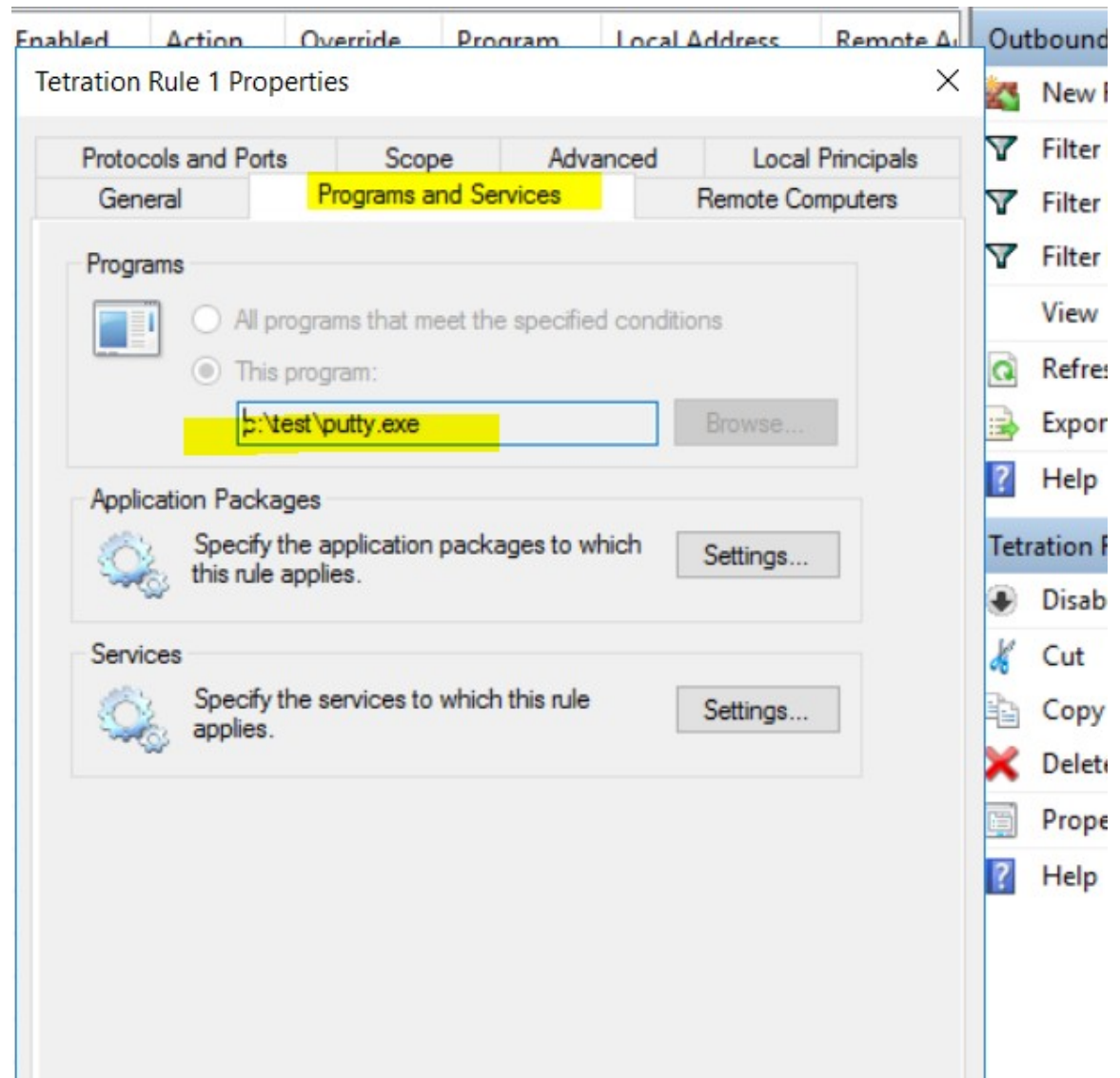
Use the following information to verify and troubleshoot policies based on application name on Windows OS workloads.

The following sections describe the way policies should appear on the workload for an application binary entered as `c:\test\putty.exe`.

Sample Policy Based on Application Name

```
dst_ports {
  start_port: 22
  end_port: 22
  consumer_filters {
    application_name: "c:\test\putty.exe"
  }
}
ip_protocol: TCP
address_family: IPv4
inspection_point: EGRESS
```

Generated Firewall Rule



Generated Filter Using netsh

To verify, using native Windows tools, that a filter has been added to an advanced policy:

- With administrative privileges, run `cmd.exe`.
- Run `netsh wfp show filters`.
- The output file, **filters.xml**, is generated in the current directory.
- Check `FWPM_CONDITION_ALE_APP_ID` for the application name in the output file: `filters.xml`.

```
<fieldKey>FWPM_CONDITION_ALE_APP_ID</fieldKey>
  <matchType>FWP_MATCH_EQUAL</matchType>
  <conditionValue>
    <type>FWP_BYTE_BLOB_TYPE</type>
```

```

                                <byteBlob>
                                  <data>
.→5c006400650076006900630065005c0068006100720064006400690073006b0076006f006
.→</data>
                                <asString>\device\harddiskvolume2\temp\putty.exe</
.→asString>
                                </byteBlob>
                                </conditionValue>

```

Generated WFP Filter Using `tetenf.exe -l -f`

```

Filter Name:                Secure Workload Rule 1
-----
EffectiveWeight:           18446744073709551592
LayerKey:                  FWPM_LAYER_ALE_AUTH_CONNECT_V4
Action:                    Permit
RemoteIP:                  10.195.210.15-10.195.210.15
Remote Port:               22
Protocol:                  6
AppID:                     \device\harddiskvolume2\test\putty.exe

```

Invalid Application Name

- In WAF mode, Firewall rule is created for an invalid application name.
- In WFP mode, the WFP filter is not created for an invalid application name but the NPC is not rejected. The agent logs a warning message and configures the rest of the policy rules.

Policies Based on Service Name

Use the following information to verify and troubleshoot policies based on Service name on Windows OS workloads.

The following sections describe the way that the policies should appear on the workload.

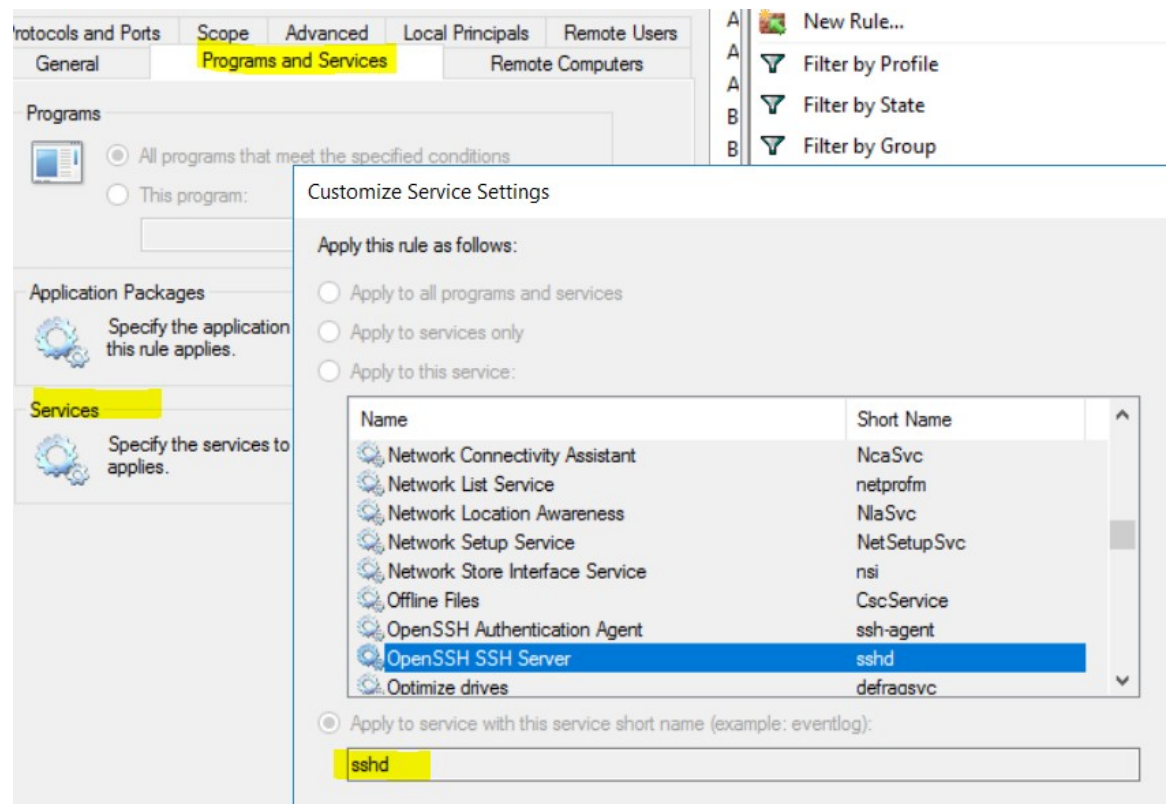
Sample Policy Based on Service Name

```

dst_ports {
  start_port: 22
  end_port: 22
  provider_filters {
    service_name: "sshd"
  }
}
ip_protocol: TCP
address_family: IPv4
inspection_point: INGRESS

```


Generated Firewall Rule



Generated Filter Using netsh

To verify using native Windows tools, that a filter has been added for an advanced policy:

- With administrative privileges, run `cmd.exe`.
- Run `netsh wfp show filters`.
- The output file, **filters.xml**, is generated in the current directory.
- Check `FWPM_CONDITION_ALE_USER_ID` for user name in the output file: filters.xml.

```
<item>
    <fieldKey>FWPM_CONDITION_ALE_USER_ID</fieldKey>
    <matchType>FWP_MATCH_EQUAL</matchType>
    <conditionValue>
        <type>FWP_SECURITY_DESCRIPTOR_TYPE</type>
        <sd>O:SYG:SYD: (A;;CCRC;;;S-1-5-80-3847866527-469524349-687026318-
        →516638107)</sd>
    </conditionValue>
</item>
```

Generated WFP Filter Using tetenf.exe -l -f

```
Filter Name:          Secure Workload Rule 3
-----
EffectiveWeight:     18446744073709551590
```

```
LayerKey:          FWPM_LAYER_ALE_AUTH_RECV_ACCEPT_V4
Action:           Permit
Local Port:       22
Protocol:         6
User or Service:  NT SERVICE\sshd
```

Invalid Service Name

- In WAF mode, the Firewall rule is created for a nonexistent service name.
- In WFP mode, the WFP filter is not created for a nonexistent service name.
- Service SID type must be *Unrestricted* or *Restricted*. If the service type is *None*, the Firewall Rule and WFP filter can be added but they have no effect.

To verify the SID type, run the following command:

```
sc qsidtype <service name>
```

Policies Based on User Group or User Name

Use the following information to verify and troubleshoot policies based on user name (with and without user group name) on Windows OS workloads.

Sections in this topic describe the way that the policies should appear on the workload.

Examples in this topic are based on policies that are configured with the following information:

Figure 4: Policies Based on User Group or User Name

Description

While using process level controls, a consumer/provider scope or filter should only contain Windows agents. Otherwise, non-Windows OSs (Linux, AIX) will skip the policy and report a sync error in Enforcement Status. See the [user guide](#) for more information.

Consumer Service

Consumer Binary Path

Consumer Users or User Groups ⓘ
sensor-dev\domain users,sensor-dev@se

Provider Service

Provider Binary Path

Provider Users or User Groups ⓘ

Sample Policy Based on User Name

```
dst_ports {
  start_port: 30000
  end_port: 30000
  provider_filters {
    user_name: "sensor-dev\sensor-dev"
  }
}
ip_protocol: TCP
address_family: IPv4
inspection_point: EGRESS
```

Sample Policy Based on User Group and User Name

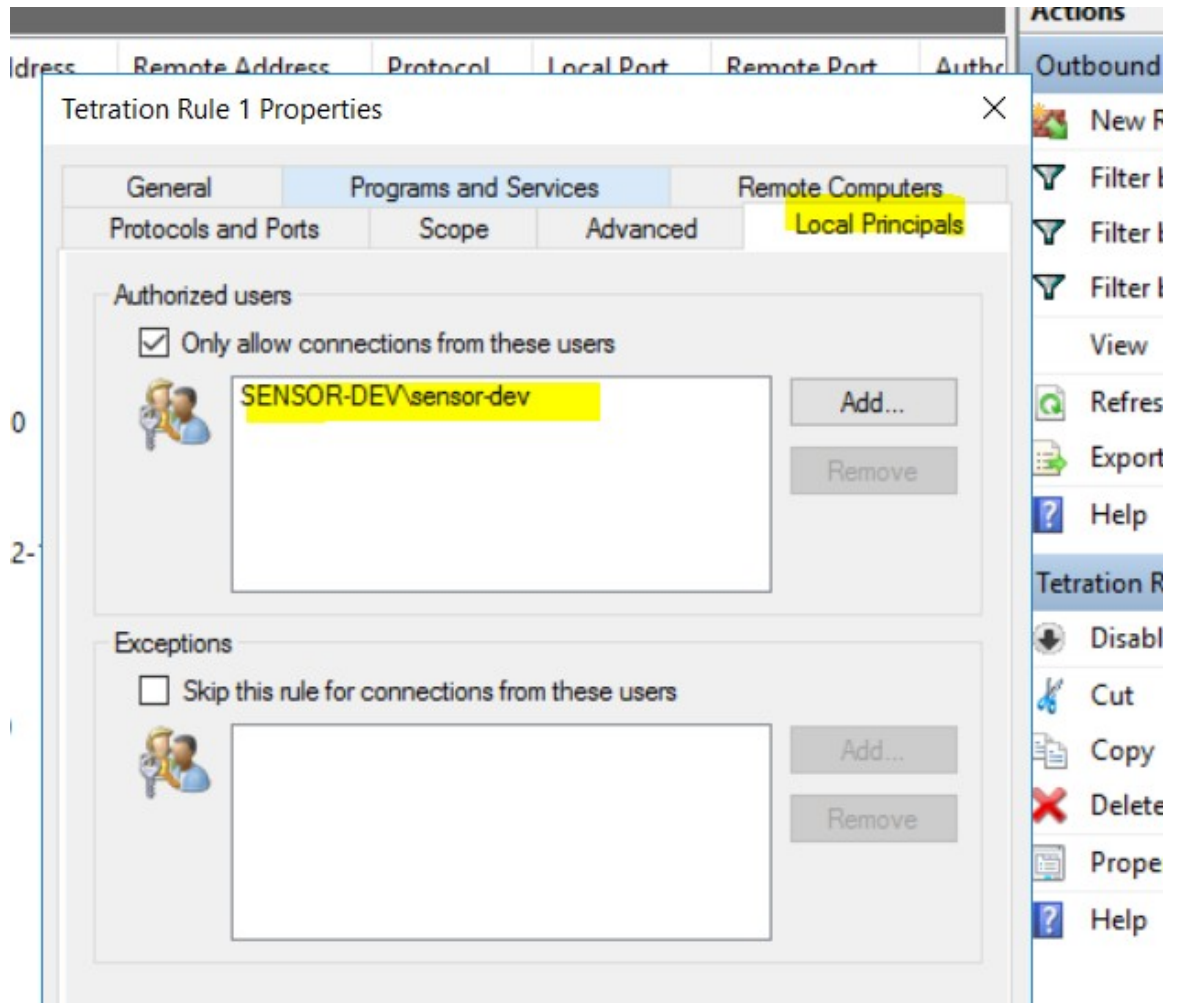
```
dst_ports {
  start_port: 30000
  end_port: 30000
  provider_filters {
    user_name: "sensor-dev\domain users,sensor-dev\sensor-dev"
  }
}
ip_protocol: TCP
```

```
address_family: IPv4
inspection_point: EGRESS
```

Generated Firewall Rule

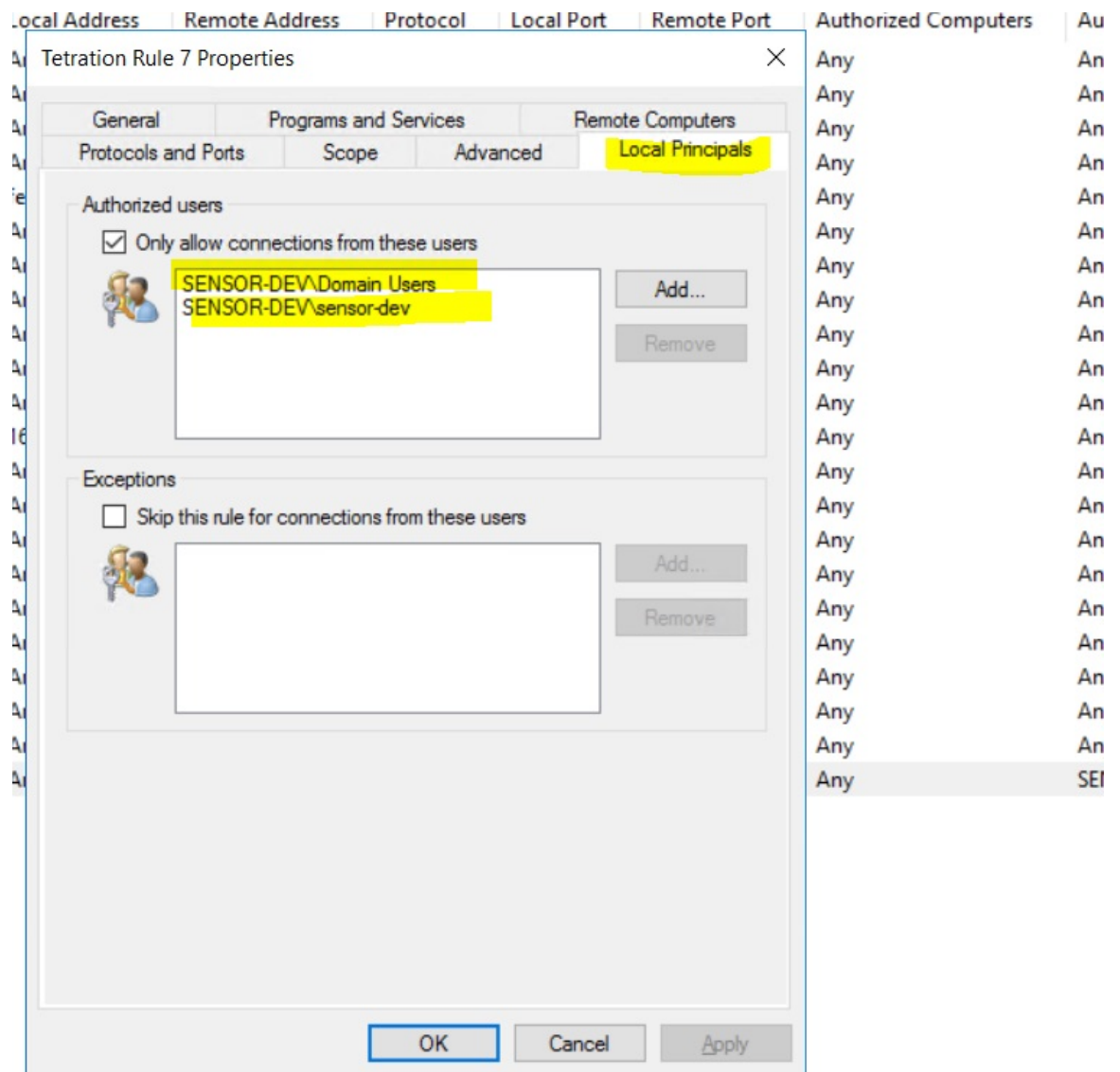
Firewall Rule Based on User Name

Example: Firewall rule based on User Name, sensor-dev\\sensor-dev



Firewall Rule Based on User Group and User Name

Example: Firewall rule based on User Name, sensor-dev\\sensor-dev and user group, domain users\\sensor-dev



Generated Filter Using netsh

To verify using native Windows tools that a filter has been added for an advanced policy:

- With administrative privileges, run `cmd.exe`.
- Run `netsh wfp show filters`.
- The output file, **filters.xml**, is generated in the current directory.
- Check `FWPM_CONDITION_ALE_USER_ID` for user name in the output file: `filters.xml`.

```
<item>
  <fieldKey>FWPM_CONDITION_ALE_USER_ID</fieldKey>
  <matchType>FWP_MATCH_EQUAL</matchType>
  <conditionValue>
    <type>FWP_SECURITY_DESCRIPTOR_TYPE</type>
```

```

        <sd>O:LSD: (A;;CC;;;S-1-5-21-4172447896-825920244-2358685150)</sd>
    </conditionValue>
</item>

```

Generated WFP Filters Using `tetenf.exe -l -f`

Filter based on User Name

Example: WFP Rule based on User Name, SENSOR-DEV\sensor-dev

```

Filter Name:                Secure Workload Rule 1
-----
EffectiveWeight:           18446744073709551590
LayerKey:                  FWPM_LAYER_ALE_AUTH_CONNECT_V4
Action:                    Permit
RemoteIP:                  10.195.210.15-10.195.210.15
Remote Port:               30000
Protocol:                  6
User or Service:           SENSOR-DEV\sensor-dev

```

Filter based on User Group and User Name

Example: WFP Rule based on User Name, SENSOR-DEV\sensor-dev and User Group name, SENSOR-DEV\Domain Users

```

Filter Name:                Secure Workload Rule 1
-----
EffectiveWeight:           18446744073709551590
LayerKey:                  FWPM_LAYER_ALE_AUTH_CONNECT_V4
Action:                    Permit
RemoteIP:                  10.195.210.15-10.195.210.15
Remote Port:               30000
Protocol:                  6
User or Service:           SENSOR-DEV\Domain Users, SENSOR-DEV\sensor-dev

```

Service name and user name cannot be configured for a Network policy rule.



Note The network policy is rejected by the Windows agent if the user name or the user group is invalid.

Enforcement of Kubernetes Pods on Windows Nodes

After you install the Kubernetes DaemonSet agent on the Windows worker nodes, it captures the network flows from the Windows worker nodes and the Kubernetes pods in an AKS environment.

Requirements

- Enforcement of Kubernetes pods is supported in an AKS environment with Windows nodes.
- Enforcement mode MUST be WFP with **Preserve Rules** set to Off.
- Supported on Microsoft Windows Server 2019 and Windows Server 2022.

The policies are enforced on vSwitch for ports that are connected to pods using VFP. The Virtual Filtering Platform (VFP) is a component of vSwitch used to configure filters for processing network traffic. While enforcing the policies, the Preserve Mode is Off.

Each filter has the following attributes:

- Id: Filter Name
- Direction: In or Out
- RuleType: Switch or Host.
 - Configure the filter on vSwitch when the type is Switch.
 - Create a WFP filter when the type is Host.
- Action: Allow or Block
- LocalPorts: This can be a port or range. For example, 80 or 100-200.
- RemotePorts: Same as LocalPorts.
- LocalAddresses: It is an address or range. For example, 10.224.0.5, 10.224.1.0/24 (10.224.1.1-10.224.1.10 is not allowed).
- RemoteAddress: Same as LocalAddresses
- Protocol: ICMP/TCP/UDP/IGMP Protocol 255 is IPPROTO_RAW and 256 – PROTO_MAX

The ports can only be specified for UDP and TCP, and ports are not allowed in the policy unless a protocol is specified.

Configuring a policy on a virtual port is a transaction-based operation. If one of the filters is invalid, enforcing the entire policy is rendered unsuccessful.

This is the stateful enforcement. Application, user, or service-based policies are currently not supported.

Compatibility with Calico

Pods enforcement works in "preserve rules" off mode. When the Windows agent enforces the rules on pods, it deletes the already configured policies. If the Calico plug-in enforces the network policies after the agent, the agent identifies it as **deviation** and network policies that are configured by Calico are deleted and agent policies are re-enforced.



Note The enforced policies are deleted when the Windows agent is uninstalled on the Windows nodes.

Visibility of Configured VFP Filters

An option to list the pod filters using Secure Workload is not available. In an AKS environment, you can use the built-in PowerShell script. Run the following PowerShell script: `c:\k\debug\collectlogs.ps1`. View the output files **vfoutput.txt** and **hnsdiag.txt** for the configured filters.

Delete VFP Filters Configured by Windows Agent

1. Run **cmd.exe** with administrative privileges.
2. Run the command: `<installation folder>\tetenf.exe -d -f -pods -token=<yyyymm>`.



Note The command deletes VFP filters for all the pods.

Troubleshoot Enforced Policies and Network Flows

1. Run the command: `netsh wfp start capture keywords=19.`
2. Run network traffic.
3. Stop capturing the flows: `netsh wfp stop capture.`
4. Extract **wfpdiag.xml** from the **wfpdiag.cab** file. View the dropped flows.

To map the allowed or dropped network flows to Pod policies:

1. Start ETW session: `logman start <session name> -p Microsoft-Windows-Hyper-V-VfpExt -o <output file.etl> -ets`
2. Run network traffic.
3. Stop capturing flows: `logman stop <session name>.`
4. In the command prompt, run: `tracert <output file.etl>.` The command creates the **dumpfile.xml** file. View the network flows.

Agent Enforcement on AIX Platform

On the AIX platform, the Secure Workload agent uses IPFilter utilities to enforce network policies. By default, after the agent is enabled on the host, the agent controls and programs the IPv4 filter table. IPv6 enforcement is not supported.

IPFilter

The IPFilter package on AIX is used to provide firewall services and is available on AIX as a kernel expansion pack. It loads as a kernel extension module, `/usr/lib/drivers/ipf`. It includes `ipf`, `ippool`, `ipfstat`, `ipmon`, `ipfs`, and `ipnat` utilities that are used to program ipfilter rules and each of these rules specifies the match criteria for a packet. For more information, see the IPFilter pages in the AIX manual.

When enforcement is enabled, the agent uses IPFilter to program the IPv4 filter table that contains rules for allowing or dropping of IPv4 packets. The agent groups these rules to categorize and manage the policies using the controller. These rules include Secure Workload rules that are derived from the policies and rules that are generated by the agent.

When an agent receives platform-independent rules, it parses and converts them into ipfilter or ippool rules and inserts these rules into the filter table. After programming the firewall, the enforcement agent monitors the firewall for any rule or policy deviation and if so, reprograms the firewall. The agent keeps track of the policies that are programmed in the firewall and reports their status periodically to the controller.

A typical policy in a platform-independent network policy message consists of:

```
source set id: "test-set-1"
destination set id: "test-set-2"
source ports: 20-30
```



```

destination ports: 40-50
ip protocol: TCP
action: ALLOW
...
set_id: "test-set-1"
  ip_addr: 1.2.0.0
  prefix_length: 16
  address_family: IPv4
set_id: "test-set-2"
  ip_addr: 5.6.0.0
  prefix_length: 16
  address_family: IPv4

```

Along with other information, the agent processes the policy and converts it into platform-specific ippool and ipfilter rule:

```

table role = ipf type = tree number = 51400
{ 1.2.0.0/16; };

table role = ipf type = tree number = 75966
{ 5.6.0.0/16; };

pass in quick proto tcp from pool/51400 port 20:30 to pool/75966 port 40:50 flags S/SA group
TA_INPUT
pass out quick proto tcp from pool/75966 port 40:50 to pool/51400 port 20:30 flags A/A group
TA_OUTPUT

```

Caveats

Host Firewall Backup

When enforcement is enabled for the first time in an Agent Config Profile, the agents running on AIX hosts, before taking control of the host firewall, store the current content of ippool and ipfilter into */opt/cisco/tetration/backup*. Successive disable or enable transitions of enforcement configuration do not generate backups. The directory is not removed upon agent uninstallation.

Known Limitations

IPv6 enforcement is not supported.

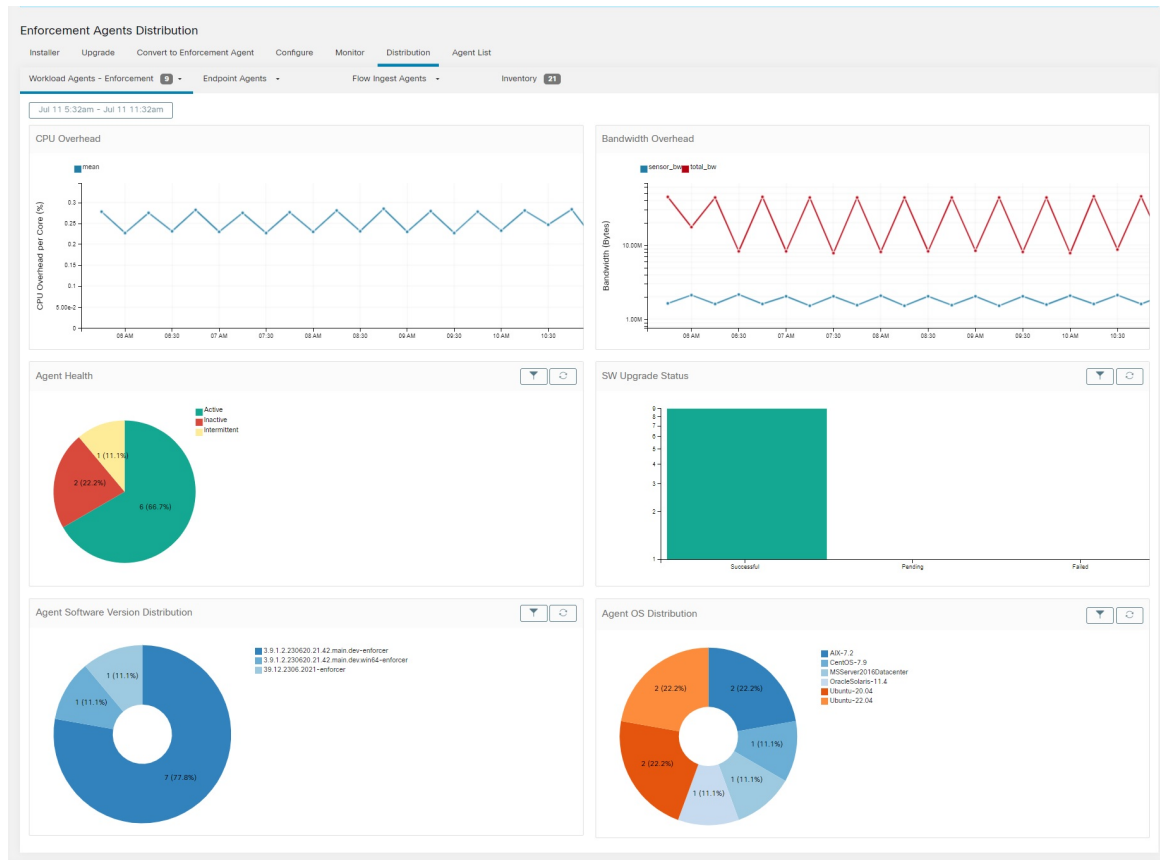
Check Agent Status and Statistics

Procedure

-
- Step 1** In the navigation pane, click **Manage > Workloads > Agents**.
 - Step 2** Click the **Distribution** tab.
 - Step 3** Click an agent type from the top of the page.
 - Step 4** On this page, you can check CPU Overhead, Bandwidth Overhead, Agent Health, Software Update Status, Agent Software Version Distribution, and Agent OS Distribution.

For more information about this page, see the Agent Status and Statistics section.

Figure 5: Agent Distribution Page



Note **Agent Health:** The agent periodically checks in every 10–30 minutes. If there is no check-in for more than 1 hour 30 minutes, then the agent is inactive. To reduce false alarms, the agent health status is set to intermittent instead of inactive if the check-in gap is between 1 hour and 1 hour 30 minutes.

For more information on the enforcement status, see the Enforcement Status section.

View Agent Details

The following steps provide one of the available options to navigate to the Workload Profile page, which displays details about the workload and its installed agent.

Procedure

- Step 1** In the navigation pane, click **Organize** > > **Scopes and Inventory**.
- Step 2** Search for a workload for which you want to view details.

- Step 3** Click the IP address to view the details such as agent health, IP address, Scopes, Inventory Type, Enforcement Groups, Experimental Groups, User Labels, and Traffic Volume (Total Bytes/Total Packets).
-

For more information, see [Workload Profile, on page 388](#).

Software Agent Config

Requirements and Prerequisites for Configuring Software Agents

- Ensure that you have the required Secure Workload user role credentials:

- Site Administrator
- Customer Support

For more information, see [User Roles and Access to Agent Configuration, on page 65](#).

- Ensure that you have privileges on the host to run the agent service on each workload. For more information, see [Service Management of Agents](#).
- Verify the supported platforms, requirements, and installation instructions for agents. For more information, see [Deploy Software Agents](#).

User Roles and Access to Agent Configuration

1. Root scope owners have access only to create a Configuration Profile and Configuration Intent specification.
2. As a Root Scope owner, you can create configuration profiles that are associated with only owned scopes and impose these configuration profiles on agents.



Note Under the Agent Configuration Profile, you can now view the number of intents using the configuration profile before you edit the profile.

Figure 6: Software Agent Configuration for Scope Owners

The screenshot shows the 'Agent Config Profiles' section on the left and the 'Agent Config Intents' section on the right.

Agent Config Profiles: A table with columns 'Name', 'Config', and 'Actions'. The 'Default' profile is selected. The 'Config' column lists various settings with toggle icons (green for on, grey for off):

- Enforcement
 - Windows Enforcement Mode - WFP (on)
 - Preserve Rules (off)
 - Allow Broadcast (on)
 - Allow Multicast (on)
 - Allow Link Local Addresses (on)
 - CPU Quota Mode - Adjusted (3%) (on)
 - Memory Quota Limit - 512MB (on)
- Flow Visibility
 - Flow Analysis Fidelity - Conversations (on)
 - Data Plane (on)
 - Auto-Upgrade (on)
 - PID Lookup (off)
 - Service Protection (off)
 - CPU Quota Mode - Adjusted (3%) (on)
 - Memory Quota Limit - 512MB (on)
 - Cleanup Period (off)
 - Flows Disk Quota - 512MB (on)
- Process Visibility and Forensics
 - Forensics (off)
 - Process Visibility (on)
 - Package Visibility (on)
 - Meltdown Exploit Detection (off)
 - CPU Quota Mode - Adjusted (3%) (on)
 - Memory Quota Limit - 256MB (on)

The 'Actions' column for the 'Default' profile shows an 'Edit' button and the text 'Used by 1 Intent'.

Agent Config Intents: A section with a 'Create Intent' button. It contains a search bar with the text 'Apply profile Default to filter Everything'. Below it is a 'View Deleted Agent Config Intents' link and another 'Create Intent' button. The 'Interface Config Intents' section shows 'No intents found' and a 'Create Intent' button. The 'Agent Remote VRF Configurations' section shows 'No configs found' and a 'Create Config' button.

- Site administrators have access to all the components in the Agent Configuration page that includes specifying interface configuration intents, remote virtual routing and forwarding configurations.

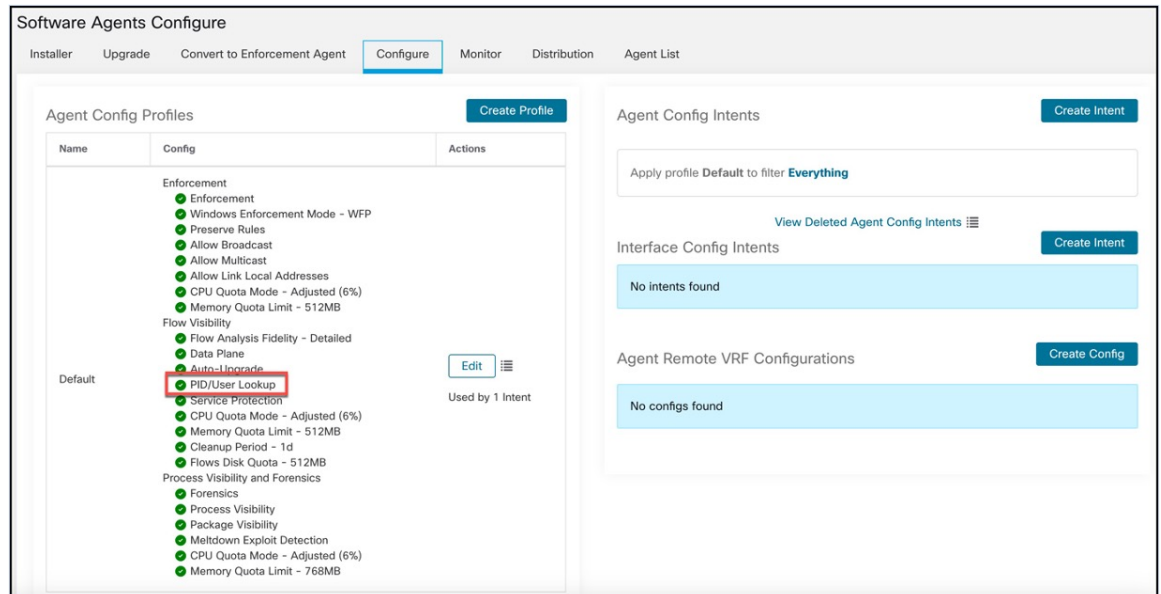
Configure Software Agents

On the Software Agent Configuration page, configure the software agents to create intents that are associated with either an **Inventory Filter** or a **Scope**. For each agent, apply the first matching intent. For more information, see [Manage Inventory for Secure Workload, on page 333](#).



Note For any Secure Workload deployment, use the default agent configuration on all agents that are not associated with any specific configuration profile.

Figure 7: Software Agent Configuration



Create an Agent Configuration Profile

Before you begin

See [Requirements and Prerequisites for Configuring Software Agents](#), on page 65.

Procedure

- Step 1** In the navigation pane, choose **Manage > Workloads > Agents**.
- Step 2** Click the **Configure** tab.
- Step 3** Click the **Create Profile** button.
- Step 4** Enter a name for the profile and choose the scope where the profile is available.
- Step 5** Enter the appropriate values in the fields listed in the following table.

Table 7: Creating Software Agent Configuration Profile Field Descriptions

| Field | Description |
|--------------------|--|
| Enforcement | |
| Enforcement | <p>Enable - Enable policy enforcement on the agent. After you enable enforcement, the agent enforces the most recently received policy set. Disable (Default) - The agent does not enforce a policy.</p> <p>Note If you enable, disable and re-enable policy enforcement on the agent, it clears the firewall state and sets the catch-all default action to ALLOW.</p> |

| Field | Description |
|-----------------------------------|---|
| Windows Enforcement Mode | <p>On Windows workloads, agents can enforce network policies using:</p> <ul style="list-style-type: none"> • WFP - Windows Filtering Platform (by directly programming WFP filters in the Windows Filter Engine). See Agent Enforcement on the Windows Platform in WFP Mode, on page 45. • WAF (Default) - Windows Advanced Firewall. See Agent Enforcement on the Windows Platform in WAF mode, on page 43. |
| Preserve Rules | <p>Enable - Preserves existing firewall rules on the agent.</p> <p>Disable (Default) - Clears existing firewall rules before applying enforcement policy rules from Secure Workload.</p> <p>Behaviour of the Preserve Rules attribute is platform-specific. You can view the details of the attributes in the Preserve Rules section in each platform.</p> |
| Allow Broadcast | <p>Enable (Default) - Adds rules to the firewall to allow ingress and egress broadcast traffic on workload.</p> <p>Disable - Does not add any rules. The broadcast traffic drops if the default policy on the agent is DENY.</p> |
| Allow Multicast | <p>Enable (Default) - Adds rules to the firewall to allow ingress and egress multicast traffic on workload.</p> <p>Disable - Does not add any rules. The Multicast traffic drops if the default policy on the agent is DENY.</p> |
| Allow Link Local Addresses | <p>Enable (Default) - Adds rules to the firewall to allow link local addresses traffic on workload.</p> <p>Disable - Does not add any rules. The Multicast traffic drops if the default policy on the agent is DENY.</p> |
| CPU Quota Mode | <p>Adjusted (Default) - The CPU limit adjusts according to the number of CPUs on the system. For example, if there are 10 CPUs, set the CPU limit to 3%, the agents use only a total of 30% (measured by top).</p> <p>Top - The CPU limit value matches the top view on average. For example, if you set the CPU limit to 3% and there are 10 CPUs in the system, the CPU usage is 3%. It is a fairly restrictive mode, use it only when necessary.</p> <p>Disable - Disable the CPU limit feature. The agent uses CPU resources that used in the operating system.</p> <p>For more information, see Secure Workload Data Sheet.</p> |
| CPU Quota Limit (%) | Specify the actual limit in percentage of the system processing power. |
| Memory Quota Limit (MB) | Specify the memory limit (in MB) for processes. If the process hits this limit, it restarts. |

| Field | Description |
|-------------------------------|--|
| Flow Visibility | |
| Flow Analysis Fidelity | <p>Conversations (Default) - Enable conversation mode on all agents.</p> <p>Detailed - Enable detailed mode on all agents.</p> |
| Data Plane | <p>Enable (Default) - Enable the agent to send reports to the cluster.</p> <p>Disable - Disable the agent's reports.</p> |
| Auto-Upgrade | <p>Enable (Default) - Automatically upgrade the agent when a new package is available.</p> <p>Disable - Do not automatically upgrade the agent.</p> |
| PID/User Lookup | <p>Enable - Process ID (PID) and User Lookup in agents.</p> <p>Set the Flow Analysis Fidelity option to detailed mode for PID and User Lookup. When you enable this feature, the agent associates network flows with running processes and users in the workload. During the process, note that some flows that might not be associated with any process even after you enable the configuration.</p> <p>Disable (Default) - Do not enable process ID and User Lookup in agents.</p> <p>Note User Lookup is not supported on Windows Server 2008 R2.</p> |
| Service Protection | <p>Enable - Enable service protection on the agent. When enabled, the agent ensures it prevents users from disabling the service, from uninstalling the agent, and from restarting the service. However, after disabling the service protection, you can continue to stop or can uninstall the agent.</p> <p>Note</p> <ul style="list-style-type: none"> • Do not disable service protection for normal auto upgrade of an agent. • Do not enable service protection for manual upgrade of an agent. • Service protection blocks any forced upgrades, such as using the installer script - forceUpgrade option. • Any system-initiated upgrade works when you enable the service protection. <p>Disable(*)-By default, disable the service protection on the agent.</p> <p>Detailed (Default) - Enable detailed mode on all agents.</p> <p>Note This feature is available only for Windows agent.</p> |

| Field | Description |
|--------------------------------|---|
| CPU Quota Mode | <p>Adjusted (Default) - Adjust the CPU limit according to the number of CPUs on the system. For example, if there are 10 CPUs in the system, set the CPU limit to 3%.</p> <p>Choose this mode to allow the agent to use a total of 30% (measured by top).</p> <p>Top- The CPU limit value matches the top view on average. For example, set the CPU limit to 3% for the 10 CPUs in the system, the CPU usage is only 3%. It is a fairly restrictive mode and uses it only when necessary.</p> <p>Disable - Disable the CPU limit feature. The agent uses CPU resources that are used in the operating system.</p> |
| CPU Quota Limit (%) | Specify the actual limit in percentage of the system processing power that the agent can use. |
| Memory Quota Limit (MB) | Specify the memory limit in MB that the process allows to use. If the process hits this limit, the process restarts. |
| Cleanup period (days) | <p>Enable - Enable automated cleanup on the agent. Enter the number of days after which remove the inactive agent.</p> <p>Disable (Default) - Do not enable automated cleanup on the agent.</p> |
| Flows Disk Quota (MB) | <p>Enter the maximum size limit (in MB) for storing the flow data.</p> <p>If the Flows Disk Quota field is:</p> <ul style="list-style-type: none"> • 0: The agents do not store offline flows locally. • Blank: Enable the Flows Time Window field. After you enter the duration in the Flows Time Window, the Flows Disk Quota field automatically sets the value to 16 GB. <p>You can either choose the Flows Disk Quota or the Flows Time Window option for flow log buffering in case of connectivity break between the agent and the cluster.</p> <p>For example, if you have set the Flows Time Window as one hour and the agent is unable to communicate with the cluster, the agent stores flow data for the last hour. Any flow data locally stored on the workload beyond the last hour is overridden by newer logs.</p> <p>Specify in MB the total size limit of stored flow data.</p> |

| Field | Description |
|---|--|
| Flows Time Window (Hours) | <p>Specify in hours how long the agent must capture and store flows locally.</p> <p>Choose either Flows Disk Quota or Flows Time Window; it's either size-based or time-based rotation. On choosing Flows Time Window, set the Flows Disk Quota to 16 GB. Setting Flows Disk Quota to 0 disables this feature.</p> <p>The flow data is rotated when it reaches either size limit or time limit.</p> <p>This field is displayed only when there is no value that is entered in the Flows Disk Quota field.</p> <p>Enter the duration, in hours, for the agents to capture the flows and store them locally.</p> <ul style="list-style-type: none"> • After the connectivity to the agents is restored, the agents send the live flow data. • While sending the live flow data, the agents also initiates to upload the buffered telemetry data. The telemetry data is sent in small packets at regular intervals. • Depending on the size of the buffered telemetry data and transmission transfer speed, it takes multiple intervals to send all the buffered data. • The agents progressively deletes the locally stored flow data. <p>Remove the outdated flow data that is stored locally after it reaches the configured size or time limit.</p> |
| Process Visibility and Forensics | |
| Forensics | <p>Enable - Enable forensics on the agent. This feature consumes extra CPU cycles that are specified in the CPU limit below. For example, if the CPU limit is 3% and you enable this feature, the agent uses up to 6% in total.</p> <p>Disable (Default) - Disable forensics on the agent.</p> |
| Meltdown Exploit Detection | <p>Enable - Enable Forensics and Meltdown exploit detection on the agent. For more information, see Side Channel in the Compatibility, on page 575.</p> <p>Disable (Default) - Disable Meltdown exploit detection on the agent.</p> |
| CPU Quota Mode | <p>Adjusted (Default) - Adjust the CPU limit according to the number of CPUs on the system. For example, set the CPU limit to 3% with 10 CPUs in the system. Choose this mode to use a total of 30% (measured by top).</p> <p>Top - The CPU limit value matches the top view on average. For example, set the CPU limit to 3% with 10 CPUs in the system, the CPU usage remains at 3%. Use this restrictive mode only if necessary.</p> <p>Disable - Disable the CPU limit feature, the agent uses CPU resources permissible by the operating system.</p> |
| CPU Quota Limit (%) | Specify the actual limit, in percentage, of the system processing power the agent can use. |
| Memory Quota Limit (MB) | Specify the memory limit (in MB). If the storage limit goes beyond the specified limit, the process restarts. |

Step 6 Click **Save**

What to do next

Associate the created profile with an agent configuration intent. For more information, see [Creating an Agent Config Intent, on page 72](#).

Creating an Agent Config Intent

Before you begin

- See [Requirements and Prerequisites for Configuring Software Agents, on page 65](#).
- Create an agent config profile. See [Create an Agent Configuration Profile, on page 67](#).

Procedure

- Step 1** In the navigation bar on the left, click **Manage > Agents**.
- Step 2** Click the **Configure** tab.
- Step 3** Click the **Create Intent** button next to the **Agent Config Intent** heading.
- Step 4** Enter the appropriate values in the fields listed in the table below:

| Field | Description |
|---------------------------|---|
| Profile (required) | Enter the name of an existing profile and select it from the dropdown menu. |
| Filter (required) | Enter the name of an existing filter or scope or select <i>Create new filter</i> from the dropdown menu. See Filters for more information on creating filters. |

Step 5 Click **Save**.

Figure 8: Agent Config Intents

Agent Config Intents

Apply profile to filter

Apply profile **Default** to filter **Everything**

Creating a Remote VRF configuration for agents

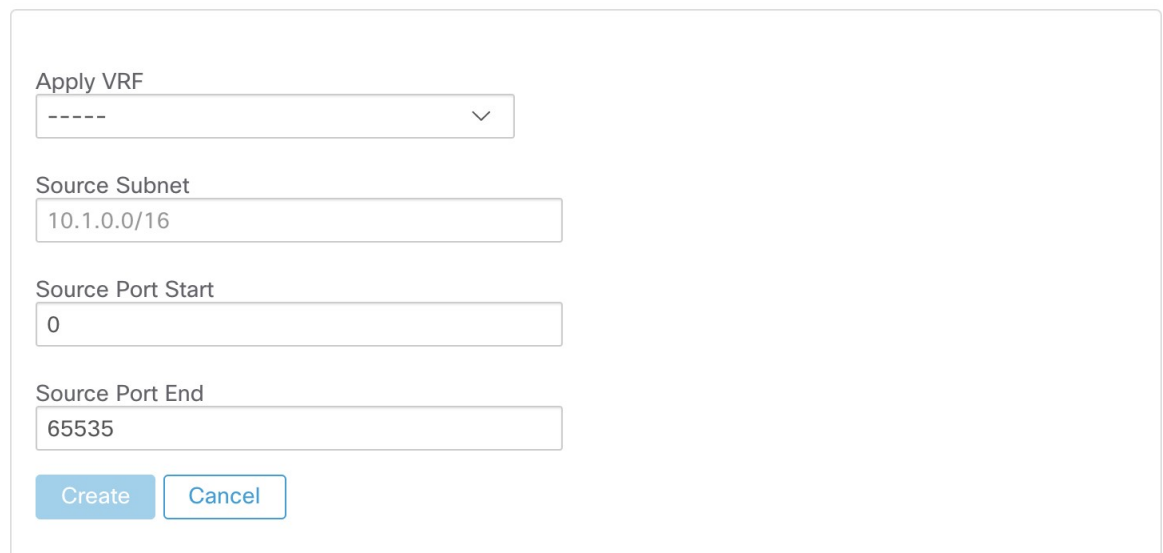
This is the recommended way to assign VRFs for Secure Workload software agents. Using this configuration, Secure Workload appliance assigns VRFs to software sensors based on the source IP address and source port seen for those agent on connections to Secure Workload appliance.

Procedure

- Step 1** In the navigation bar on the left, click **Manage > Agents**.
- Step 2** Click the **Configure** tab.
- Step 3** Click the **Create Config** button next to the **Agent Remote VRF Configurations** heading.
- Step 4** Enter the appropriate values in the fields and click **Save**.

Figure 9: Remote VRF configuration

Agent Remote VRF Configurations



The screenshot shows a configuration form titled "Agent Remote VRF Configurations". It contains the following fields and controls:

- Apply VRF:** A dropdown menu with a downward arrow and a dashed line indicating a selection.
- Source Subnet:** A text input field containing "10.1.0.0/16".
- Source Port Start:** A text input field containing "0".
- Source Port End:** A text input field containing "65535".
- Buttons:** Two buttons at the bottom: "Create" (highlighted in blue) and "Cancel".

Create an Interface Configuration Intent

We recommend assigning virtual routing and forwarding (VRFs) to agents in using Remote VRF configuration settings. In rare cases, when agent hosts have multiple interfaces that must be assigned to different VRFs, you can choose to assign them VRFs using Interface Configuration Intents.

Procedure

- Step 1** Navigate to **Manage > Agents**.
- Step 2** Click the **Configure** tab.
- Step 3** Click the **Create Intent** button next to the **Interface Config Intent** heading.

Step 4 Enter the appropriate values in the fields listed in the table:

| Field | Description |
|---------------|---|
| VRF | Choose a VRF from the drop-down list (required). |
| Filter | Enter the name of an existing filter or scope or choose <i>Create a new filter</i> from the drop-down list (required). For more information, see Filters . |

Step 5 Click **Save**.

Figure 10: Interface Configuration Intents

Interface Config Intents

Apply VRF Default to filter

Save Cancel

No intents found

Agent Remote VRF Configuration

No configs found

Create Config

Everything

Filter

Test

Default

Unknown

Tetration

Tetration:Campus

Tetration:Internet

Create new filter

5 of 42 matching scopes shown

Note When you delete an interface with a higher priority config intent, the agents do not fall back to the default catch all intent.

View Detailed Agent Status in the Workload Profile

Procedure

- Step 1** Follow the steps above to check Agent status.
- Step 2** On the Enforcement Agents page, click **Agent OS Distribution**. Select an operating system and click filter image on the top-right corner of the box.
- Step 3** On the Software Agent List page, agents with selected operating system Distribution is listed.

Step 4 Click on **Agent** for the agent details, and click IP address. On the Workload Profile page, you can view details of the Host Profile, Agent Profile and agent specific details, such as Bandwidth, Long-lived Processes, Packages, Process Snapshot, Configuration, Interfaces, Stats, Policies, Container Policies and so on.

Step 5 Click **Config** tab to see the configuration on the end-host.

Step 6 Click **Policies** tab to see the enforced policies on the end-host.

Figure 11: Workload Profile - Config

Figure 12: Workload Profile - Policies

| Priority | Packets | Bytes | Actions | Direction | Family | Proto | Src Inventory | Src Ports | Dest Inventory | Dest Ports |
|----------|---------|-------|---------|-----------|--------|-------|------------------|-----------|------------------|------------|
| 1 | N/A | N/A | ALLOW | INGRESS | IPv4 | TCP | any | any | 172.21.95.163/32 | 22 |
| 2 | N/A | N/A | ALLOW | EGRESS | IPv4 | TCP | 172.21.95.163/32 | 22 | any | any |
| 3 | N/A | N/A | ALLOW | INGRESS | IPv4 | TCP | any | 22 | 172.21.95.163/32 | any |
| 4 | N/A | N/A | ALLOW | EGRESS | IPv4 | TCP | 172.21.95.163/32 | any | any | 22 |
| 5 | N/A | N/A | ALLOW | INGRESS | IPv4 | ST | ubuntuhosts | any | 172.21.95.163/32 | any |
| 6 | N/A | N/A | ALLOW | EGRESS | IPv4 | ST | 172.21.95.163/32 | any | ubuntuhosts | any |
| 7 | N/A | N/A | ALLOW | INGRESS | IPv4 | ST | ubuntuhosts | any | 172.21.95.163/32 | any |
| 8 | N/A | N/A | ALLOW | EGRESS | IPv4 | ST | 172.21.95.163/32 | any | ubuntuhosts | any |
| 9 | N/A | N/A | ALLOW | INGRESS | IPv4 | STP | ubuntuhosts | any | 172.21.95.163/32 | any |
| 10 | N/A | N/A | ALLOW | EGRESS | IPv4 | STP | 172.21.95.163/32 | any | ubuntuhosts | any |
| 11 | N/A | N/A | ALLOW | INGRESS | IPv4 | STP | ubuntuhosts | any | 172.21.95.163/32 | any |
| 12 | N/A | N/A | ALLOW | EGRESS | IPv4 | STP | 172.21.95.163/32 | any | ubuntuhosts | any |
| 13 | N/A | N/A | ALLOW | INGRESS | IPv4 | SUNND | ubuntuhosts | any | 172.21.95.163/32 | any |

Note **Fetch All Stats** is not supported on Windows agent hosts, which is used to provide statistics for individual policies.

Rehoming of Agents

Rehoming of agents is the method to move users from On-premises to SaaS or SaaS to On-premises.

User Roles

- Site Administrator
- Customer Support Representative

You can migrate to or from a SaaS environment, especially, when you move from SaaS to On-premises, you must work with an internal support team.

Workflow

- Enter the Activation Key, Sensor virtual IP, and Sensor certificate authority (CA) and [Enable Rehoming, on page 76](#).
- [Select Agents to Rehome, on page 78](#).
- [Disable Rehoming, on page 78](#).



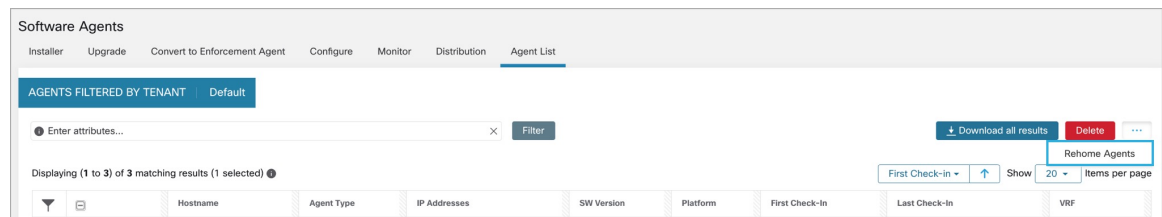
Note At any given time, you can move an agent to only one destination. We recommend that you Disable Agent Rehoming after you move the agent.

Enable Rehoming

Procedure

- Step 1** In the left navigation menu, click **Manage > Workloads > Agents**.
- Step 2** Click the **Agent List** tab.
- Step 3** Click the menu icon and select **Rehome Agents**.

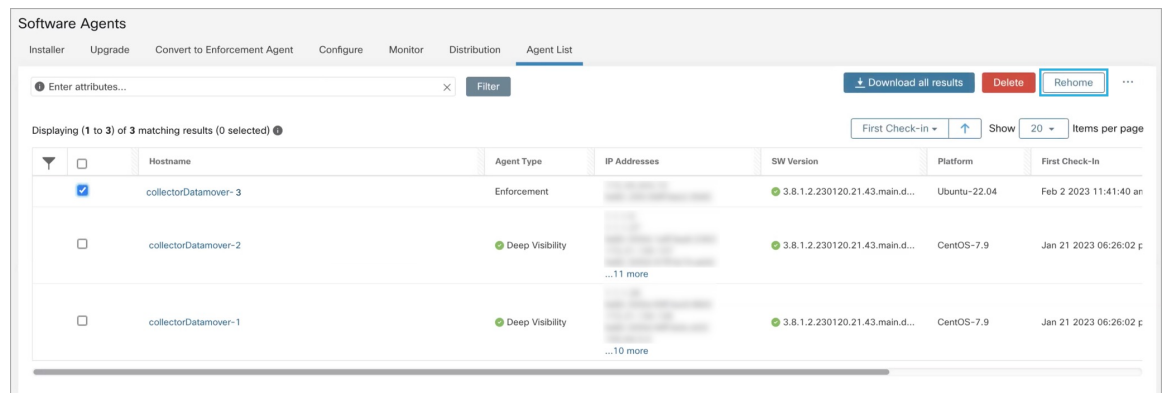
Figure 13: Rehome Agents



Step 4 On the **Agent Rehoming** window, fill in the following details:

| Field | Description |
|---|---|
| Destination Scope Activation Key | <ol style="list-style-type: none"> Navigate to Manage > Workloads > Agents. Click the Installer tab. Select Manual install using classic packaged installers. Click Next. Click Agent Activation Key. Copy the Key value and paste it into the Destination Scope Activation Key field. |
| Destination Sensor VIP | <ol style="list-style-type: none"> Navigate to Platforms > Cluster Configuration. Copy the Sensor VIP and paste it into the Destination Sensor VIP field. |
| HTTPS proxy | Enter a proxy domain or address if the agent needs to use a proxy for outbound communication. |
| Destination Sensor CA Cert | <ol style="list-style-type: none"> Navigate to Platforms > Cluster Configuration. Click Download Sensor CA Cert. |

Figure 14: Enable Agent Rehoming



Step 5 Click **Enable Agent Rehoming**.

The configuration is saved. The Rehome button appears at the top right.

Select Agents to Rehome

Procedure

Step 1 Select an agent.

Step 2 Click **Rehome**.

Figure 15: Select Agents to Rehome

The screenshot shows the 'Software Agents' interface. At the top, there are navigation tabs: Installer, Upgrade, Convert to Enforcement Agent, Configure, Monitor, Distribution, and Agent List (which is selected). Below the tabs is a search bar with the placeholder text 'Enter attributes...' and a 'Filter' button. To the right of the search bar are buttons for 'Download all results', 'Delete', and 'Rehome' (which is highlighted with a red box). Below the search bar, it says 'Displaying (1 to 3) of 3 matching results (0 selected)'. There is also a 'First Check-in' dropdown, an up arrow, and a 'Show 20 Items per page' dropdown. The main content is a table with the following columns: Hostname, Agent Type, IP Addresses, SW Version, Platform, and First Check-In. The table contains three rows of agent information. The first row is selected with a blue checkbox. The 'Rehome' button is highlighted in the top right corner of the interface.

| Hostnames | Agent Type | IP Addresses | SW Version | Platform | First Check-In |
|----------------------|-----------------|--------------|--------------------------------|--------------|------------------------|
| collectorDatamover-3 | Enforcement | ... | 3.8.1.2.230120.21.43.main.d... | Ubuntu-22.04 | Feb 2 2023 11:41:40 an |
| collectorDatamover-2 | Deep Visibility | ...11 more | 3.8.1.2.230120.21.43.main.d... | CentOS-7.9 | Jan 21 2023 06:26:02 p |
| collectorDatamover-1 | Deep Visibility | ...10 more | 3.8.1.2.230120.21.43.main.d... | CentOS-7.9 | Jan 21 2023 06:26:02 p |

Step 3 Click **Yes** to confirm.

Disable Rehoming



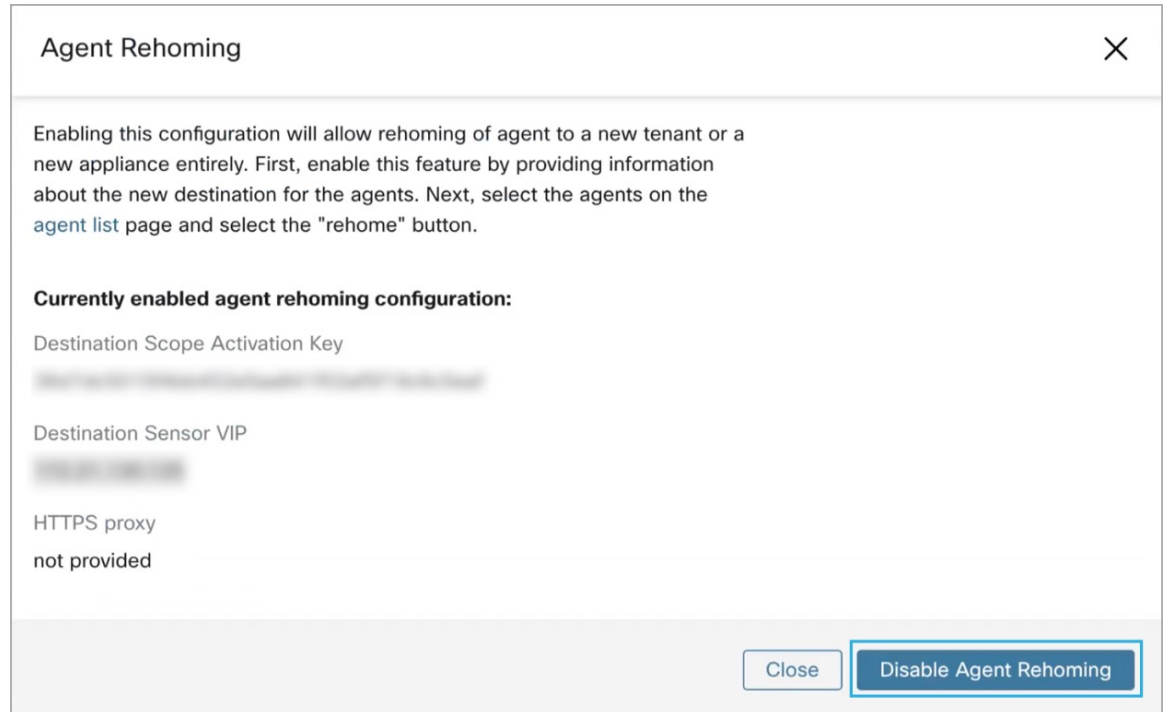
Note If there are multiple users rehoming to or from SaaS, the site administrator has to move each tenant or an appliance separately. To do this, disable Rehoming to clear the settings, and then enable Rehoming for the new user.

Procedure

Step 1 Click the menu icon and choose **Rehome Agents**.

Step 2 On the **Agent Rehoming window**, click **Disable Agent Rehoming**.

Figure 16: Disable Agent Rehoming



Generate Agent Token

In the agent configuration profile, you can enable service protection to prevent uninstallation, disabling, and stopping Windows agent services. To perform any changes to the agents, you can disable this protection on the agent configuration profile. However, if you are unable to disable the protection because of connectivity issues, you can generate an agent token to disable the service protection on workload. The token is valid for 15 minutes.

Supported roles to generate and retrieve agent tokens:

- **Site administrators:** For clusters or tenants.
- **Customer support:** For tenants.
- **Agent installer:** For agent-specific tokens.



Note You can generate time-based agent tokens only for Windows OS-based software agents.

To generate and download agent tokens, perform these steps:

Procedure

- Step 1** In the navigation pane, click **Manage > Workloads > Agents > Agent List**.
- Based on your requirement, you can choose one of the agent token types—Cluster, tenant, or agent-specific. For the agent-specific token, go to Step 5.
- Step 2** Click the menu icon and choose **Agent Token**.
- Note** The **Agent Token** option is only visible for site administrators or customer support user roles.
- Step 3** Select a token type:
- Token For Cluster—This option is visible only to site administrators and the token is applicable for all the agents.
 - Token For Tenant—Applicable for the agents under a selected tenant.
- Step 4** To download the token key, click **Download Token**.
- Step 5** To view and download token key details of a specific agent:
- a) Go to the **Agent List** tab and click the required agent. Under **Agent Details > Agent Token**, you can view the token key and expiry details of the token.
 - b) To download the agent-specific token, click **Download Token**.
-

What to do next

After downloading the agent token file, run the following command on the agent to disable service protection: `"C:\Program Files\Cisco Tetration\TetSen.exe" -unprotect <token>`, where `token` is the downloaded agent token.

After the service protection is disabled using a token, it may be automatically re-enabled when the service restarts and connects to the Secure Workload cluster.

Host IP Address Change when Enforcement is Enabled

Changing the IP address on hosts when enforcement is enabled may have an impact if the host IP is seen in the host firewall rules and catch all is set to deny. In this scenario, the following steps are recommended to change the host IP address:

Procedure

- Step 1** On the Secure Workload UI, create a new Agent Config Profile with enforcement disabled.
- Step 2** Create Intent with list of hosts that need IP address change with their old and new IP address.
- Step 3** Apply the newly created Agent Config Profile to the Intent and save the Intent.
- Step 4** These selected hosts should have enforcement disabled.
- Step 5** Change the IP address on these hosts.
- Step 6** On the Secure Workload UI, update the filters in the scope with the new IP address of these hosts.

- Step 7** Verify the IP address change from Agent Workload Profile page “Interfaces” tab. In the “Policies” tab, make sure policies are generated with new IP address.
- Step 8** Remove the Intent/Profile created above.
- Step 9** If the original Agent Config Profile for the scope had enforcement disabled, then enable enforcement.
-

Upgrading Software Agents

Upgrade Agents from UI

Agents can be upgraded using Agent Config Intent workflow as described here - [Software Agent Config](#). While configuring an agent config profile, there is an **Auto Upgrade** option which can be enabled or disabled. If the option is enabled, the agents matching inventory filter criteria are automatically upgraded to the latest available version.

On the **Software Agents > Agent List** page, software agents with outdated versions are highlighted with a warning sign under the **SW Version** column. It is important to upgrade these agents to the latest available version on the cluster.

To use software agent config intent workflow to configure software agent upgrade:

Procedure


- Step 1** Create an inventory filter on the **Inventory Filters** page. For more information, see [Filters](#).

Figure 17: Inventory Filter

+ Create an Inventory Filter

1 Define ————— 2 Summary


Name


Development Linux VMs 

Create a query based on Inventory Attributes:

Inventory is matched dynamically based on the query. The labels can include Hostname, Address/Subnet, OS, and more. The [full list](#) is in the user guide.

A preview of matching inventory items will be shown in the next step.

Query 

Hostname contains linux 

[Show advanced options](#)

Step 2 Create an Agent Config profile for the agents selected by the inventory filter. Optionally, you can enable the **Auto Upgrade** option to automatically upgrade the selected agents.

Figure 18: Agent Config

[Create Profile](#)

| Name ↑ | Config | Actions |
|---------|---|---|
| Default | <p>Enforcement</p> <ul style="list-style-type: none"> <input checked="" type="checkbox"/> Enforcement <input checked="" type="checkbox"/> Windows Enforcement Mode - WAF <input type="checkbox"/> Preserve Rules <input checked="" type="checkbox"/> Allow Broadcast <input checked="" type="checkbox"/> Allow Multicast <input checked="" type="checkbox"/> Allow Link Local Addresses <input checked="" type="checkbox"/> CPU Quota Mode - Adjusted (3%) <input checked="" type="checkbox"/> Memory Quota Limit - 512MB <p>Flow Visibility</p> <ul style="list-style-type: none"> <input checked="" type="checkbox"/> Flow Analysis Fidelity - Detailed <input checked="" type="checkbox"/> Data Plane <input checked="" type="checkbox"/> Auto-Upgrade <input type="checkbox"/> PID Lookup <input checked="" type="checkbox"/> CPU Quota Mode - Adjusted (3%) <input checked="" type="checkbox"/> Memory Quota Limit - 512MB <p>Process Visibility and Forensics</p> <ul style="list-style-type: none"> <input type="checkbox"/> Forensics <input type="checkbox"/> Meltdown Exploit Detection <input checked="" type="checkbox"/> CPU Quota Mode - Adjusted (3%) <input checked="" type="checkbox"/> Memory Quota Limit - 256MB | Edit |
| VM | <p>Enforcement</p> <ul style="list-style-type: none"> <input type="checkbox"/> Enforcement <input checked="" type="checkbox"/> Windows Enforcement Mode - WAF <input type="checkbox"/> Preserve Rules <input checked="" type="checkbox"/> Allow Broadcast <input checked="" type="checkbox"/> Allow Multicast <input checked="" type="checkbox"/> Allow Link Local Addresses <input checked="" type="checkbox"/> CPU Quota Mode - Adjusted (3%) <input checked="" type="checkbox"/> Memory Quota Limit - 512MB <p>Flow Visibility</p> <ul style="list-style-type: none"> <input checked="" type="checkbox"/> Flow Analysis Fidelity - Detailed <input checked="" type="checkbox"/> Data Plane <input checked="" type="checkbox"/> Auto-Upgrade <input type="checkbox"/> PID Lookup <input checked="" type="checkbox"/> CPU Quota Mode - Adjusted (3%) <input checked="" type="checkbox"/> Memory Quota Limit - 512MB <p>Process Visibility and Forensics</p> <ul style="list-style-type: none"> <input type="checkbox"/> Forensics <input type="checkbox"/> Meltdown Exploit Detection <input checked="" type="checkbox"/> CPU Quota Mode - Adjusted (3%) <input checked="" type="checkbox"/> Memory Quota Limit - 256MB | Edit Delete |

[View Deleted Agent Config Profiles](#)

Step 3 Create an agent config intent to apply the config profile to the agents selected using inventory filter. If the auto upgrade option is enabled, the selected agents are automatically upgraded.

It normally takes up to 30 minutes to upgrade an agent after an agent profile is applied to them.

Figure 19: Agent Config Intent

Agent Config Intents

Apply profile to filter

Apply profile **Default** to filter **Everything**

Note Auto Upgrade setting in the default agent profile applies to ERSPAN.

Manual Agent Upgrade

The following section explains how to manually upgrade agents without using the Sensor Config intent workflow.

Procedure

- Step 1** In the left navigation pane, click **Manage > Workloads > Agents**.
- Step 2** Click the **Upgrade** tab.
Deep visibility and enforcement agents are displayed and for each agent only newer versions to which it is upgradable are listed. By default, the latest version is selected.
- Step 3** To filter specific agents, enter your search query in the filter box. For example, enter Platform = CentOS-7.6.
- Step 4** Select the agents to be upgraded to the selected version and click **Upgrade**.

Note Under normal circumstances, allowing the agent to automatically upgrade is strongly recommended and is the only supported upgrade method. If you want to control the upgrade by manually downloading the latest version and directly deploying it to the agents which are running on workloads, ensure that you follow the safety precautions.

Upgrade Behaviour of Kubernetes/Openshift Agent

Agents installed on Kubernetes/Openshift nodes using the daemonset installer script are capable of self-upgrade. The upgrade process is controlled by either the auto-upgrade option or by manually triggering an upgrade for any node in the Kubernetes/Openshift cluster. The mechanism of the upgrade in this environment is to upgrade the Docker image in the daemonset specification, which means that an upgrade of one agent affects all agents covered by the daemonset, as explained in the next paragraph.

When a Daemonset Pod specification changes, Kubernetes/Openshift will trigger a graceful shutdown, fetch the new docker image(s) and start the Secure Workload agent pods on ALL nodes in the Kubernetes/Openshift

cluster. This will cause agents to be upgraded on other nodes, even if the policy to allow upgrades is applicable only to a subset of the nodes in the cluster.

If auto-upgrade is disabled for all nodes, manual upgrade is possible by downloading a new installer script and re-running the install. The installation script auto-detects the case of new installation vs upgrading an existing installation and will work to manually upgrade the daemonset pods when it detects an installation is already in place.

Removing Software Agents

Remove a Deep Visibility or Enforcement Linux Agent

RPM based installation:

1. Run command: `rpm -e tet-sensor`

Agent uninstallation event is communicated to the cluster and the agent will be marked as uninstalled on **Software Agent** page.

Manually delete the agent from UI on the **Software Agent** page or the user can enable automated cleanup or removal of the agent by turning on the **cleanup period** from agent config profiles.



Note By default, the **cleanup period** is turned off.

Ubuntu .deb based installation:

Fresh installation of Ubuntu agents now uses the native .deb format.

1. Run command: `dpkg --purge tet-sensor`

Agent uninstallation event is communicated to the cluster and the agent will be marked as uninstalled on **Software Agent** page.

Manually delete the agent from UI on **Software Agent** page or the user can enable automated cleanup or removal of the agent by turning on the **cleanup period** from agent config profiles.



Note

- By default, the **cleanup period** is turned off.
- During the agent operations, it is possible that some kernel modules will be loaded automatically by the kernel. For example, if enforcement is enabled in Linux, Netfilter modules might be loaded. Agents do not have a list of modules loaded by kernel. Therefore, during agent uninstallation, it cannot possibly unload the kernel modules.
- If enforcement agent applied a policy to the system firewall, uninstalling agent clears the applied policy and opens the system firewall.

Figure 20: Agent Uninstallation Alert

The screenshot shows the Cisco Secure Workload interface with the 'Software Agents' page. The 'Agent List' tab is active, displaying a table of agents. A tooltip points to a red status icon for the agent 'bd4-ai-hy-centos76', indicating it was uninstalled on Feb 9 9:43pm.

| Hostname | Agent Type | IP Addresses | SW Version | Platform | First Check-In | Last Check-In | VRP |
|-----------------------------|-----------------|--|------------------------------------|--------------------|-------------------------------|------------------------------|----------------------|
| Uninstalled on Feb 9 9:43pm | Enforcement | 172.26.231.175 fe80:250:56ff:fe91:3d8b | 3.8.1.2.2301.3021-enforcer | OracleSolaris-11.4 | Feb 9 2023 02:59:20 pm (PST) | Feb 9 2023 08:59:44 pm (PST) | Default |
| bd4-ai-hy-centos76 | Enforcement | 172.20.207.106 fe80:a65c:ca3a:b5e5:e097 192.168.122.1 | 3.8.1.2.230130.21.43.main.dev-e... | CentOS-7.6 | Feb 9 2023 04:38:44 pm (PST) | Feb 9 2023 09:33:26 pm (PST) | Default |
| sensor-dev-rocky90 | Enforcement | 10.195.210.122 fe80:250:56ff:fe91:ca35 | 3.8.1.2.230130.21.43.main.dev-e... | RockyLinux-9.0 | Feb 3 2023 12:02:31 am (PST) | Feb 9 2023 09:01:48 pm (PST) | Default |
| sensor-dev-oracle9 | Enforcement | 10.195.210.121 fe80:250:56ff:fe91:1c2d | 3.8.1.2.230130.21.43.main.dev-e... | OracleServer-9.0 | Feb 3 2023 12:01:09 am (PST) | Feb 9 2023 09:23:27 pm (PST) | Default |
| sensor-dev-alm9 | Enforcement | 10.195.210.120 fe80:250:56ff:fe91:5389 | 3.8.1.2.230130.21.43.main.dev-e... | AlmaLinux-9.0 | Feb 3 2023 12:00:00 am (PST) | Feb 9 2023 09:22:52 pm (PST) | Default |
| hartmut-u16 | Enforcement | 172.26.231.235 fe80:250:56ff:fe91:34c4 | 3.7.1.5.dev-enforcer | Ubuntu-16.04 | Feb 2 2023 11:27:44 am (PST) | Feb 9 2023 09:19:46 pm (PST) | Default |
| p91-isa06 | Enforcement | 172.29.157.105 fe80:288a:97fe:fe3e:5902 | 3.8.1.2.230130.21.43.main.dev-e... | AlmaLinux-9.0 | Feb 2 2023 08:46:08 am (PST) | Feb 9 2023 09:20:33 pm (PST) | Default |
| sensor-dev-deb11 | Enforcement | 172.20.207.225 fe80:250:56ff:fe91:d737 | 3.8.1.2.230130.21.43.main.dev-e... | Debian-11 | Feb 2 2023 08:44:43 am (PST) | Feb 9 2023 09:21:10 pm (PST) | Default |
| agent-rsg-deb10 | Enforcement | 10.195.210.132 fe80:250:56ff:fe91:13d9 | 3.8.1.2.230130.21.43.main.dev-e... | Debian-10 | Feb 2 2023 08:43:16 am (PST) | Feb 9 2023 09:18:56 pm (PST) | Default |
| sensor-dev-deb9 | Enforcement | 10.195.210.199 fe80:250:56ff:fe91:c542 | 3.8.1.2.230130.21.43.main.dev-e... | Debian-9 | Feb 2 2023 08:41:18 am (PST) | Feb 9 2023 09:19:10 pm (PST) | Default |
| sensor-dev-deb8 | Enforcement | 10.195.210.145 fe80:250:56ff:fe91:d8a4 | 3.8.1.2.230130.21.43.main.dev-e... | Debian-8 | Feb 2 2023 08:39:05 am (PST) | Feb 9 2023 09:21:08 pm (PST) | Default |
| p91-isa09 | Enforcement | 172.29.157.24 fe80:288a:97fe:fe3e:9202 :ac1d:9d18 | 3.8.1.2.230130.21.43.main.dev-e... | AIX-7.2 | Feb 1 2023 08:44:31 am (PST) | Feb 9 2023 09:08:44 pm (PST) | Default |
| collectorDatacenter-1 | Deep Visibility | 1.1.1.26 fe80:5554:aaff:fa20:bd3c 10.195.248.22 fe80:5554:56ff:fe9b:b306 100.64.1.2 10 more | 3.8.1.2.230130.21.43.main.dev-s... | CentOS-7.9 | Jan 31 2023 08:08:47 pm (PST) | Feb 9 2023 09:09:39 pm (PST) | Tetration Default |

Removing a Deep Visibility/Enforcement Windows Agent

There are two options to uninstall Secure Workload agents:

Procedure

- Step 1** Go to Control Panel / Programs / Programs And Features, and uninstall **Cisco Secure Workload Agent (Cisco Tetration Agent)**.
- Step 2** Alternatively, run the shortcut **Uninstall.Ink** within **'C:\Program Files\Cisco Tetration'**
- Step 3** If enforcement agent applied a policy to the system firewall, uninstalling agent clears the applied policy, and opens the system firewall.

The Agent uninstallation event will be communicated to the cluster and the agent will be marked as uninstalled on **Software Agent** page.

Manually delete the agent from UI on the **Software Agent** page or the user can enable automated cleanup or removal of the agent by turning on the **cleanup period** from agent config profiles.

Note By default, the **cleanup period** is turned off.

- Note**
- If Npcap has been installed during agent installation, it will also get uninstalled.
 - By default log files, config files and certs will not get removed during uninstall. If you'd like to remove them, run the shortcut **UninstallAll.Ink** in same folder.

Remove a Deep Visibility or Enforcement AIX Agent

Procedure

Run command: `'installp -u tet-sensor'`.

The Agent uninstallation event will be communicated to the cluster and the agent will be marked as uninstalled on the **Software Agent** page.

Manually delete the agent from UI on the **Software Agent** page or the user can enable automated cleanup or removal of the agent by turning on the **cleanup period** from agent config profiles.

Note

- By default, the **cleanup period** is turned off.
 - The Deep Visibility Agent is controlled by System Resource Controller as tet-sensor. It is possible to start, stop, restart, and remove it. The service is made persistent with inittab as tet-sen-engine.
 - The Enforcement Agent is controlled by System Resource Controller as tet-enforcer. It is possible to start, stop, restart, and remove it. The service is made persistent with inittab as tet-enf-engine.
 - During the agent operations, it is possible that some kernel modules will be loaded automatically by the kernel. For example, if enforcement is enabled in AIX, ipfilter modules are loaded. Agents do not have a list of modules loaded by kernel. Therefore, during agent uninstallation, it cannot possibly unloaded the kernel modules.
 - If enforcement agent applied a policy to the system firewall, uninstalling agent clears the applied policy and opens the system firewall.
-

Remove Universal Linux Agent

Procedure

- Step 1** Run the uninstall script: `'usr/local/tet-light/uninstall.sh'`
- Step 2** Delete the agent from UI on the **Software Agent** page
-

Remove Universal Windows Agent

Procedure

- Step 1** Run the uninstall script: `'C:\Program Files\Cisco Tetration\Lightweight Sensor\uninstall.cmd'`

Step 2 Delete the agent from UI on the **Software Agent** page

Remove an Enforcement Kubernetes or OpenShift Agent

Procedure

Step 1 Locate the original installer script or download a new script from the Secure Workload UI.

Step 2 Run the uninstall option: **install.sh --uninstall**. The same considerations apply as during the install.

- Only supported on Linux x86_64 architectures.
- Either ~/.kube/config contains an admin credentials user or use the --kubeconfig option to point to the kubectl admin credentials file.

Step 3 Delete the agents for all the Kubernetes nodes from UI on the **Software Agent** page

Remove a Deep Visibility Solaris Agent

Procedure

Step 1 Run command: `pkg uninstall tet-sensor`

Step 2 Delete the agent on the **Software Agent** page.

Data collected and exported by workload agents

This section describes the main components of a software agent, how it is registered with backend services, what data are collected and exported to the cluster for analytical purposes.

Registration

After the agent has been successfully installed onto the system, it needs to register with the backend services to obtain a valid unique identifier. The following information is sent in the registration request:

- Hostname
- BIOS-UUID
- Platform information (such as CentOS-6.5)
- Self-generated client certificate (generated with openssl command)
- Agent type (visibility or enforcement.)

If the agent fails to obtain a valid id from the server, it will keep retrying until it gets one. It is very important that the agent is registered, otherwise all the subsequent communication with other services (such as collectors) will be rejected.

Agent upgrade

Periodically (around 30 minutes), the agent sends a message to backend service to report its current version. The backend service uses the agent's id and its current version to decide whether a new software package is available for the agent. The following information is sent:

- Agent's id (obtained after successful registration)
- Current agent's version

Config server

Agents export the following information to the configured config server:

- Hostname
- Agent's id (obtained after successful registration)
- List of interfaces, each includes:
 1. Interface's name
 2. IP family (IPv4 or IPv6)
 3. IP addresses
 4. Netmask
 5. Mac addresses
 6. Interface's index

As soon as any interface property changes (such as an IP address of an existing interface changes, or a new interface comes up), this list is refreshed and reported to the config server.

Network Flow Information

Network flow information is the summarization of all packets flowing through the system. There are two modes of capturing flow information: Detailed and Conversation. By default, the **Conversation** mode is used to capture the flow information. The captured flows are exported to a collector and the exported information includes:

- Flow identifier: Uniquely identify the network flow. It includes the general information such as: IP protocol, source and destination IP, and layer 4 ports.
- IP Information: Contains information that is seen in the IP header, such as: TTL, IP flags, Packet ID, IP options, and Fragmentation flags.
- TCP Information: Contains information that is seen in the TCP header, such as: sequence number, Ack number, TCP options, Rcvd windows size.

- Flow Information: Statistics of the flow (such as total packets, total bytes, TCP flags statistics, packet length statistics, and socket statistics), interface index from which the flow was observed, start time and end time of flow.
- In a K8s environment, the agent captures network flows from pods and hosts, and then correlates the flows and reports as related flows. This is qualified with the following CNIs:
 - Calico
 - Flannel
 - Weave
 - AKS/GKE/AWS VPC CNI
 - Openshift CNI
 - Cilium CNI



Note Network flows are captured from pods and hosts, however, the correlation of flows is not possible when Cilium CNI is used.

In Conversation mode, the agent exports only TCP flows that are bidirectional in nature along with other connectionless flows. Conversation mode is supported for Windows, AIX, and Linux platforms. For more information on Conversation mode, see [Conversation Mode](#).



-
- Note**
- In K8s environment, correlation of Pod or Host flows are not done in Conversation mode.
 - In either of the modes, agents do not export the following flows:
 - ARP/RARP conversations
 - Agent's flows to collectors
-

Machine information

Machine info describes all the processes running on the host. In addition, it contains network information that is associated with the processes and the command used to launch the processes. Machine info is exported every minute and includes the following information:

- Process ID
- User ID: owner of the process
- Parent Process ID
- Command string used to launch the process
- Socket information: protocol (such as UDP or TCP), address type: IPv4 or IPv6, source and destination IP, source and destination port, TCP state, process's start and end time, path to process binary
- Forensic information: for more information, see the section [Compatibility, on page 575](#).

Agent statistics

Agent keeps track of various statistics, including system's statistics and its own, such as:

- Agent's start time and uptime
- Agent's run time in user mode and kernel mode
- Number of packets received and dropped
- Number of successful and failed SSL connections
- Total flow packets and bytes
- Total exported flows and packets to collectors
- Agent's memory and CPU usage

Enforcement Alerts

There are three types of enforcement alerts:

- Agent Reachability
This alert detects when the agent is not reachable. This alert triggers if the agent has not communicated with the Secure Workload cluster for more than the configured number of seconds.
- Workload Firewall
This alert triggers if enforcement is configured on a workload but the workload Firewall is detected to be off, since this condition will prevent Secure Workload Agent from enforcing traffic policies.
- Workload Policy
This alert triggers if the workload firewall rules are different from the Secure Workload policies applicable to this workload (the workload's "concrete policies".)

Figure 21: Enforcement Alerts Types

Configure Enforcement Alerts
See All Configured Enforcement Alerts ✕

Alert Name ⓘ

Alert Types ⓘ

For Scope: **TenantTesting**

Alert Condition ⓘ

Severity

Hide Advanced Settings ^

Individual Alerts

Summary Alerts

You can set the Severity of the alert as well as other per-type configuration parameters. To configure enforcement alerts, see [Configure Alerts, on page 647](#).

Figure 22: View Configured Enforcement Alerts

| Alerts Trigger Rules | | | |
|----------------------|----------------------------|--|--|
| Alert Type | | | |
| All | | Enter attributes... | <input type="button" value="Filter Alerts"/> |
| Alert Type ↑↓ | Alert Name ↑↓ | Configuration ↑↓ | Actions ↑↓ |
| ENFORCEMENT | Agent_Not_Reachable | Scope : Default when Agent not Reachable (seconds) > 300 | 🗑️ ✎ |
| ENFORCEMENT | Workload_Firewall | Scope : Default when Firewall = Off | 🗑️ ✎ |
| ENFORCEMENT | Workload_Policy_Deviations | Scope : Default when Policy = Deviated | 🗑️ ✎ |

Enforcement UI Alerts Details

Figure 23: Enforcement alert details

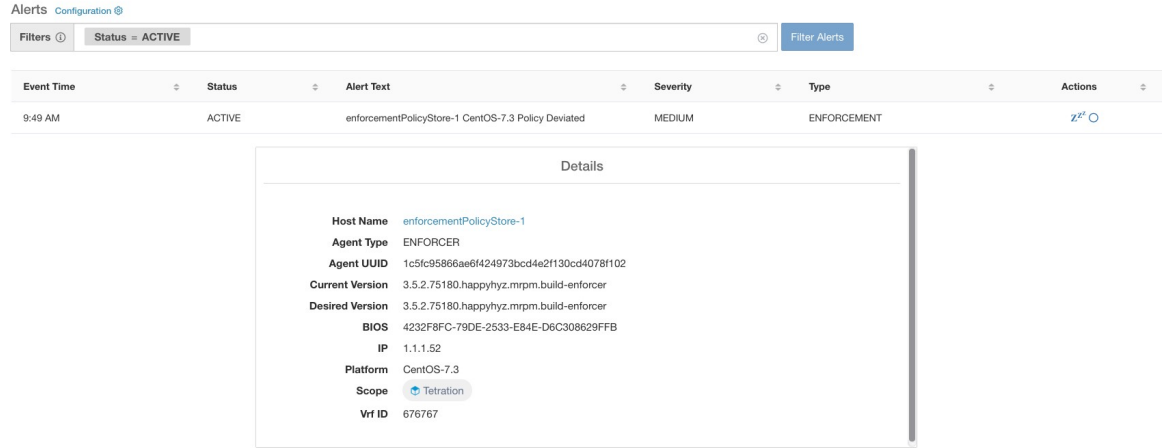
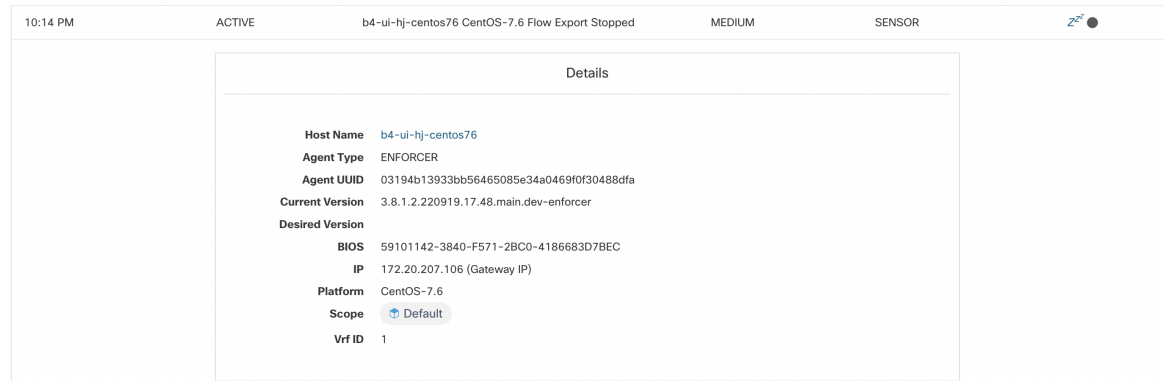


Figure 24: Enforcement alert details when proxy is enabled on the host



Enforcement Alert Details

See [Common Alert Structure](#) for general alert structure and information about fields. The `alert_details` field is structured and contains the following subfields for enforcement alerts

| Field | Alert Type | Format | Explanation |
|-----------|------------|--------|--|
| AgentType | <i>all</i> | string | “ENFORCER” or “SENSOR” depending on the installed type |
| HostName | <i>all</i> | string | Host name on which the agent is deployed |
| IP | <i>all</i> | string | IP address of the node/gateway |

| Field | Alert Type | Format | Explanation |
|-------------------|------------|---------|--|
| Bios | <i>all</i> | string | BIOS UUID of the node |
| Platform | <i>all</i> | string | Platform/OS information of the node |
| CurrentVersion | <i>all</i> | string | Software version of the agent on the node |
| DesiredVersion | <i>all</i> | string | Software version desired for the agent |
| LastConfigFetchAt | <i>all</i> | integer | Unix timestamp of when the agent last sent https request |

Example of alert_details for an enforcement alert

```
{
  "AgentType": "ENFORCER",
  "Bios": "72EF1142-03A2-03BC-C2F8-F600567BA320",
  "CurrentVersion": "3.5.1.1.mrpm.build.win64-enforcer",
  "DesiredVersion": "",
  "HostName": "win2k12-production-db",
  "IP": "172.26.231.193",
  "Platform": "MSServer2012R2Standard"
}
```

Sensor Alerts

Sensor Alert Configuration provides the ability to configure different types of alerts, you can set the severity of the alert and types of configuration parameters.

For more information, see [Alert Configuration Modal](#).



Note Starting Secure Workload 3.5, you can configure Sensor Alerts, using the *Alert Configuration Model*.

Configure Sensor alerts to report when an agent fails to upgrade. This alert triggers if the agent failed to upgrade to the needed version.

Configure Sensor alerts to detect when agent flow export must stop. This alert triggers if connectivity is blocked between the agent and the cluster, therefore preventing flows and other system information from sent or delivered.

Configure sensor alerts to detect when agent check_in times out. This alert triggers if the cluster does not received a check-in request from an agent after more than 90 minutes.

Figure 25: Configure Sensor Alerts

Configure Sensors Alerts
See All Configured Sensors Alerts ✕

Alert Name ?

Alert Types ?

Agent Upgrade

Agent Flow Export

Agent Check In

Agent Memory Usage

Agent CPU Quota

Amount Of Flow Observations

New Agent Registered

Pcap Status

Agent Uninstalled

Not Recommended Cipher

Deprecated TLS Version

Agent Auto Removal

For Scope: **Default**

Alert Condition ?

Severity

Low

Medium

High

Critical

Immediate Action

Hide Advanced Settings ^

Individual Alerts

Enable

Disable

Summary Alerts

None

Hourly

Daily

Cancel

Create

Figure 26: View Sensor Alerts

Alerts Trigger Rules

Alert Type ?

Sensors

Enter attributes...

✕

Filter Alerts

| Alert Type ? | Alert Name ? | Configuration ? | Actions ? |
|---|---|--|--|
| SENSORS | Upgrade_Status | Scope : Tetration when Agent Upgrade Status = Failed | 🗑️ ✎ |
| SENSORS | Iface_Flow_Export_Status | Scope : Tetration when Agent Flow Export Status = Stopped | 🗑️ ✎ |
| SENSORS | Upgrade_Srv_CheckIn | Scope : Tetration when Agent Check-In Service = Inactive | 🗑️ ✎ |
| SENSORS | Agent_Mem_Usage | Scope : Tetration when Deep Visibility Memory Usage (MB) > 512 and Enforcement Memory Usage (MB) > 512 and Forensic Memory Usage (MB) > 256 | 🗑️ ✎ |
| SENSORS | Agent_CPU_Quota | Scope : Tetration when Deep Visibility CPU Quota (%) > 3 and Enforcement CPU Quota (%) > 3 and Forensic CPU Quota (%) > 3 | 🗑️ ✎ |
| SENSORS | Amt_Of_Flow_Obs | Scope : Tetration when Amount of Flow Observations > 500000 | 🗑️ ✎ |
| SENSORS | Agent_Uninstalled | Scope : Tetration when Agent Uninstalled = On | 🗑️ ✎ |
| SENSORS | Agent_Auto_Removal | Scope : Tetration when Alert before Removal (minutes) = 5 | 🗑️ ✎ |

Sensor UI Alerts Details

Figure 27: Sensor Alerts

The screenshot shows the 'Alerts' section of the Sensor UI. At the top, there are tabs for 'Alerts' and 'Configuration'. Below the tabs, there is a filter bar with 'Filters' and 'Status = ACTIVE'. A 'Filter Alerts' button is also present. The main area displays a table of alerts with columns: Event Time, Status, Alert Text, Severity, Type, and Actions. One alert is visible: 11:13 AM, ACTIVE, b4-ui-centos76 CentOS-7.6 Agent Inactive, MEDIUM, SENSOR. Below the table, a 'Details' panel is open, showing the following information:

- Host Name: b4-ui-centos76
- Agent Type: ENFORCER
- Agent UUID: c6c2fbed5e510f5f4eb43b98d30add8ab3fd907
- Current Version: 3.6.1.2.201213.21.41.main.dev-enforcer
- Desired Version:
 - BIOS: 59101142-3840-F571-2BC0-4186683D7BEC
 - IP: 172.20.207.106
- Platform: CentOS-7.6
- Scope: Default
- Vrf ID: 1

Sensor Alert Details

For the general structure of alerts and for information about fields, see Common Alert Structure. The `alert_details` field is structured and contains the following subfields for sensor alerts

| Field | Alert Type | Format | Explanation |
|-------------------|------------|---------|--|
| AgentType | <i>all</i> | string | ENFORCER or SENSOR depending on the installed type |
| HostName | <i>all</i> | string | Host name on which the agent is deployed |
| IP | <i>all</i> | string | IP address of the node/gateway |
| Bios | <i>all</i> | string | BIOS UUID of the node |
| Platform | <i>all</i> | string | Platform/OS information of the node |
| CurrentVersion | <i>all</i> | string | Software version of the agent on the node |
| DesiredVersion | <i>all</i> | string | Software version desired for the agent |
| LastConfigFetchAt | <i>all</i> | integer | Unix timestamp of when the agent last sent HTTPS request |

Example of alert_details for a sensor alert

```
{
  "AgentType": "SENSOR",
  "Bios": "72EF1142-03A2-03BC-C2F8-F600567BA320",
  "CurrentVersion": "3.5.1.1.mrpm.build.win64-sensor",
  "DesiredVersion": "",
  "HostName": "win2k12-production-db",
  "IP": "172.26.231.193",
  "Platform": "MSServer2012R2Standard"
}
```

Frequently Asked Questions

This section lists some potential issues that you could possibly face during deployment and operating the software agents.

General

Log files: Log files get stored inside the <install-location>/logs or <install-location>/log folder. The log files get monitored and rotated through the Secure Workload services.

Agent deployment

Linux

Q: What do I do when the command

```
rpm -Uvh tet-sensor-1.101.2-1.el6-dev.x86_64.rpm
```

fails to install agents and displays the following error:

```
error: cannot create transaction lock on /var/lib/rpm/.rpm.lock (Permission denied).
```

A: If you do not have the right privileges to install the agents, either switch to root or use sudo to install the agents.

Q: What happens when you run “sudo rpm -Uvh tet-sensor-1.0.0-121.1b1bb546.el6-dev.x86_64.rpm” and encounter the following error:

```
Preparing... ##### [100%]
which: no lsb_release in (/sbin:/bin:/usr/sbin:/usr/bin:/usr/X11R6/bin)
error: %pre(tet-sensor-site-1.0.0-121.1b1bb546.x86_64) scriptlet failed, exit status 1
error: install: %pre scriptlet failed (2), skipping tet-sensor-site-1.0.0-121.1b1bb546
```

A: The system does not satisfy the requirements to install the agents. In this particular case, lsb_release tool is not installed.

For more information, see the Software Agents Deployment Label section and install the required dependencies.

Q: What happens when you run “sudo rpm -Uvh tet-sensor-1.0.0-121.1b1bb546.el6-dev.x86_64.rpm” and encounter the following error:

```

Unsupported OS openSUSE project
error: %pre(tet-sensor-1.101.1-1.x86_64) scriptlet failed, exit status 1
error: tet-sensor-1.101.1-1.x86_64: install failed
warning: %post(tet-sensor-site-1.101.1-1.x86_64) scriptlet failed, exit status 1

```

A: Your OS is not supported to run software agents (in this particular case, “openSUSE project” is a non-supported platform).

For more information, see the Software Agents Deployment Label section.

Q: After I have installed all the dependencies and run installation with proper privileges with no errors. How do I know the agents installation was successful?

A: After you have installed the agents, to verify if the installation, run the following command:

```

$ ps -ef | grep -e csw-agent -e tet-
root      14158      1  0 Apr03 ?        00:00:00 csw-agent
root      14160 14158  0 Apr03 ?        00:00:00 csw-agent watch_files
root      14161 14158  0 Apr03 ?        00:00:03 csw-agent check_conf
root      14162 14158  0 Apr03 ?        00:01:03 tet-sensor -f conf/.sensor_config
root      14163 14158  0 Apr03 ?        00:02:38 tet-main --sensoridfile=./sensor_id
root      14164 14158  0 Apr03 ?        00:00:22 tet-enforcer --logtostderr
tet-sen+  14173 14164  0 Apr03 ?        00:00:21 tet-enforcer --logtostderr
tet-sen+  14192 14162  0 Apr03 ?        00:07:23 tet-sensor -f conf/.sensor_config

```

You must see three entries of *csw-agent* and at least two entries of *tet-sensor*. If the services are not running, ensure that the following directories are available, else the installation has failed.

- /usr/local/tet for most Linux distributions
- /opt/cisco/tetration for AIX, Ubuntu
- /opt/cisco/secure-workload for Solaris, Debian

Windows

Q: When I run the PowerShell agent installer script, I get one of the following errors:

1. The underlying connection was closed: An unexpected error occurred on a receive.
2. The client and server cannot communicate, because they do not possess a common algorithm

A: It is most likely because host and the server has mismatched SSL/TLS protocols configured. One can check the SSL/TLS version using the following command:

```
[Net.ServicePointManager]::SecurityProtocol
```

To set the SSL/TLS to be matching with server one can use the following command (note, this is not a permanent change, only temporary with the current PowerShell session):

```
[Net.ServicePointManager]::SecurityProtocol =
[System.Net.SecurityProtocolType]'Ssl3,Tls,Tls11,Tls12'
```

Q: When I run the MSI installer from the downloaded bundle, I get the following error:

```

This installation package could not be opened. Verify that the package exists and that you
can access it, or contact the application vendor to verify that this is a valid Windows
Installer package.

```

A: Make sure `C:\Windows\Installer` path exists. If running the MSI installer from the command line, make sure to not include the relative path when pointing to the msi file. Example of correct syntax:

```
msiexec /i "TetrationAgentInstaller.msi" /! *v "msi_install.log" /norestart
```

Q: I have observed that Windows Sensor software fails to upgrade if underlying NIC is Nutanix VirtIO Network Driver.

A: There is an incompatibility issue between Npcap 0.9990 and Nutanix VirtIO Network Driver version earlier than 1.1.3 and Receive Segment Coalescing is enabled.

The resolution for this is to upgrade Nutanix VirtIO Network Driver to version 1.1.3 or later.

Q: I have installed windows sensor. The sensor doesnt seem to register and the sensor_id file contains the following: uuid-invalid-platform

A: You may not have system32 in PATH variable for Windows. Check if system32 is in PATH, if not run the following:

```
set PATH=%PATH%;C:\Windows\System32\
```

Q: I am not receiving the network flows from Kubernetes Pods on Windows Nodes.

A: To verify if the required sessions are running to capture the flows from Kubernetes pods on Windows nodes, perform the following:

1. Run **cmd.exe** with administrative privileges.
2. Run the following command: `logman query -ets`

Ensure that the following sessions are running:

- CSW_MonNet: Captures network flows
- CSW_MonHCS: Monitors creation of pods
- CSW_MonNat: Monitors NATed flows

Kubernetes

If the installer script fails during Kubernetes Daemonset Installation, there are a large number of possible reasons.

Q: Is the Docker Registry serving images reachable from nodes ?

A: Debug Direct or HTTPS Proxy issues with the cluster pulling images from Cisco Secure Workload cluster

Q: Is the container runtime complaining about SSL/TLS insecure errors ?

A: Verify that the Secure Workload HTTPS CA certificates are installed on all Kubernetes nodes in the appropriate location for the container runtime.

Q: Docker Registry authentication and authorization of image downloads failures ?

A: From each node, attempt to manually docker pull the images from the registry urls in the Daemonset spec using the Docker pull secrets from the secret created by the Helm Chart. If the manually image pull also fails, need to pull logs from the Secure Workload Cluster registryauth service to debug the issue further.

Q: Is the Kubernetes cluster hosted inside the Secure Workload appliance healthy ?

A: Check the service status page for the cluster to ensure all related services are healthy. Run the dstool snapshot from the explore page and retrieve the logs generated.

Q: Are the Docker Image Builder daemons running ?

A: Verify from the dstool logs that the build daemons are running.

Q: Are the jobs that build Docker images failing ?

A: Verify from the dstool logs that the images have not been built. Docker build pod logs can be used to debug errors during the buildkit builds. Enforcement Coordinator logs can also be used to debug the build failures further.

Q: Are the jobs creating Helm Charts failing ?

A: Verify from the dstool logs that the Helm Charts have not been built. Enforcement Coordinator logs will contain the output of the helm build jobs and can be used to debug the exact reason for the Helm Chart build job failures.

Q: Installation bash script was corrupt ?

A: Attempt to download the installation bash script again. The bash script contains binary data appended to it. If the bash script is edited in any way with a text editor or saved as a text file, special characters in the binary data may be mangled/modified by the text editor.

Q: Kubernetes cluster configuration – too many variants and flavors, we support classic K8s.

A: If the customer is running a variant of Kubernetes, there can be many failure modes at different stages of the deployment. Classify the failure stage - kubectrl command run failure, helm command run failures, pod image download failures, pod privileged mode options rejected, pod image trust content signature failures, pod image security scan failures, pod binaries fail to run (architecture mismatch), pods run but the Secure Workload services fail to start, Secure Workload services start but have runtime errors due to unusual operating environment.

Q: Are the Kubernetes RBAC credentials failing ?

A: In order to run privileged daemonsets, we need admin privileges to the K8s cluster. Verify the the kubectrl config file has its default context pointing towards the target cluster and admin-equivalent user for that cluster.

Q: Busybox image available or downloadable from all cluster nodes ?

A: Fix the connectivity issues and manually test that the busybox image can be downloaded. The exact version of busybox that is used in the pod spec must be available (pre-seeded) or downloadable on all cluster nodes.

Q: API Server and etcd errors or a general timeout during the install ?

A: Due to the instantiation of daemonset pods on all nodes in the Kubernetes cluster, the CPU/Disk/Network load on the cluster can spike suddenly. This is highly dependent on the customer specific installation details. Due to the overload, the installation process (images pulled on all nodes and written to disks) might take too long or overload the Kubernetes API server or the Secure Workload Docker Registry endpoint or, if configured, the proxy server temporarily. After a brief wait for image pulls on all nodes to complete and a reduction in CPU/Disk/Network load on the Kubernetes cluster nodes, retry the installation script again. API Server and etcd errors from the Kubernetes control plane indicate that the Kubernetes control plane nodes may be underprovisioned or affected by the sudden spike in activity.

Q: Secure Workload Agent experiencing runtime issues with its operations ?

A: Refer to the Linux Agent troubleshooting section if the pods are correctly deployed and the agent has started running but is experiencing runtime issues. The troubleshooting steps are the same once the Kubernetes deployment has successfully installed and started the pods.

Anomaly Types

These are the most common issues encountered on the workflow when using and managing Secure Workload Agents.

Agent Inactivity

Agent has stopped checking to the cluster services. This can happen due to several reasons:

- The host might have been down
- The network connectivity has been broken or blocked by firewall rules
- The agent service has been stopped

All platforms

- Verify the host is active and healthy
- Verify the agent service is up and running
- Verify the network connectivity to the cluster is working

Upgrade Failure

Agent upgrade has failed. This can be triggered by few cases such as:

- Not finding the package when the check in script attempts to download it - the upgrade package cannot be unpacked or the installer from the package cannot be verified.
- Installation process failing from an OS issue or dependency.

Windows

- Missing CA root certificate: [Certificate Issues](#)
- If agent was originally installed manually with a MSI install package, check if the Windows edition matches list of supported platforms in user guide: [Check If Platform Is Currently Supported](#)
- Check to make sure OS is configured correctly for Windows Installer operation: [Windows Installer Issues](#)
- Make sure there is enough free disk space on host

Linux

- If the host OS has been upgraded since the last agent installation, verify the current release matches list of supported platforms in user guide: [Check If Platform Is Currently Supported](#)
- Make sure there have been no changes to the required dependencies since the last installation. You can run the agent installer script with `-no-install` option to re-verify these dependencies.
- Make sure there is enough free disk space on host

AIX

- Make sure there have been no changes to the required dependencies since the last installation. You can run the agent installer script with `-no-install` option to re-verify these dependencies.
- Make sure there is enough free disk space on host

Convert Failed

The current agent type mismatches desired agent type and the convert attempt has timed out. This issue can be caused by a communication issue when an agent does `check_in` to download the package, or wss service failed to push `convert_command` to the agent.

All Platforms

- Verify the current release and agent type matches list of supported platforms in user guide: [Check If Platform Is Currently Supported](#)

Convert Capability

The ability to convert the agent from one type (such as deep visibility) to another type (such as enforcement) is not available by all agents. If an agent that is not capable to do the conversion is required to convert, the anomaly will be reported.

Policy Out of Sync

The current policy (NPC) version last reported by the agent does not match the current version generated on the cluster. This can be caused by a communications error between the agent and the cluster, the agent failing to enforce the policy with the local firewall, or the agent enforcement service not running.

Windows

- If enforcement mode is WAF, verify there are no GPOs present on the host that would prevent the Firewall from being enabled, adding rules (with Preserve Rules Off) or setting default actions: [GPO Configurations](#)
- Verify there is connectivity between the host and the cluster: [SSL Troubleshooting](#)
- Verify the generated rule count is less than **2000**
- Verify the WindowsAgentEngine service is running: `sc query windowsagentengine`
- Verify there are available system resources

Linux

- Verify iptables and ipset is present with the `iptables` and `ipset` command
- Verify there is connectivity between the host and the cluster: [SSL Troubleshooting](#)
- Verify the tet-enforcer process is running: `ps -ef | grep tet-enforcer`

AIX

- Verify ipfilter is installed and running with the `ipf -V` command

- Verify there is connectivity between the host and the cluster: [SSL Troubleshooting](#)
- Verify the tet-enforcer process is running: `ps -ef | grep tet-enforcer`

Flow Export: Pcap Open

If the Secure Workload Agent cannot open the pcap device to capture flows, you see errors in the Agent logs. A successfully opened Pcap device will report as follows:

Windows Log: `C:\Program Files\Cisco Tetration\Logs\TetSen.exe.log`

```
I0609 15:25:52.354 24248 Started capture thread for device <device_name>
I0609 15:25:52.354 71912 Opening device {<device_id>}
```

Linux Log: `/usr/local/tet/logs/tet-sensor.log`

```
I0610 03:24:22.354 16614 Opening device <device_name>
[2020/06/10 03:24:23:3524] NOTICE: lws_client_connect_2: <device_id>: address 172.29.
.→136.139
```

Flow Export: HTTPS Connectivity

Connectivity between the agent and the cluster is externally blocked therefore preventing flows and other system information from being delivered. This is caused by one or more configuration issues with network firewalls, SSL decryption services, or third party security agents on the host.

- If there are known firewalls or SSL decryption security devices between the agent and the cluster, make sure that communications to all Secure Workload collector and VIPs IP addresses are being permitted. For on-prem clusters, the list of collectors will be listed under **Troubleshoot > Virtual Machines** in the navigation bar at the left side of the Secure Workload web interface. Look for collectorDatamover-*. For Secure Workload cloud, all the IP addresses that need to be permitted will be listed in your Portal.
- To help identify if there is SSL decryption, `openssl s_client` can be used to make a connection and display the returned certificate. Any additional certificate added to the chain will be rejected by the Agent's local CA. [SSL Troubleshooting](#)

Certificate Issues

Windows

Certificate Issues for MSI installer

MSI installer is signed using code signing certificate:

For MSI Installer, version 3.6.x onwards and 3.5.1.31 onwards

- Leaf Certificate: Cisco Systems, Inc
- Intermediate Certificate: DigiCert Trusted G4 Code Signing RSA4096 SHA384 2021 CA1
- Root Certificate: DigiCert Trusted Root G4

For MSI Installer, earlier versions

- Leaf Certificate: Cisco Systems, Inc

- Intermediate Certificate: Symantec Class 3 SHA256 Code Signing CA
- Root Certificate: VeriSign Class 3 Public Primary Certification Authority - G5

It uses timestamp certificate:

For MSI Installer, version 3.6.x onwards and 3.5.1.31 onwards

- Leaf Certificate: Symantec SHA256 TimeStamping Signer - G3
- Intermediate Certificate: Symantec SHA256 TimeStamping CA
- Root Certificate: VeriSign Universal Root Certification Authority

For MSI Installer, earlier versions

- Leaf Certificate: Symantec SHA256 TimeStamping Signer - G2
- Intermediate Certificate: Symantec SHA256 TimeStamping CA
- Root Certificate: VeriSign Universal Root Certification Authority

Windows Sensor Installation or upgrade will fail if digital signature of MSI installer is invalid. Digital signature is invalid if

- MSI Installer Signing Root Certificate or MSI Installer timestamp Root Certificate is not in a “Trusted Root Certification Authority” store
- MSI Installer Signing Root Certificate or MSI Installer timestamp Root Certificate is expired or revoked.

Issue 1

Installation of agent might fail with below error in the TetUpdate.exe.log “Msi signature is not trusted. 0x800b0109”

Resolution

- Run the command *certmgr* from command prompt
- Check if MSI Installer Signing Root Certificate or MSI Installer timestamp Root Certificate is in *Untrusted Certificates* store.
- Move it to *Trusted Root Certification Authority* store.

Issue 2

Windows Sensor upgrade fails with the following error in TetUpdate.exe.log “Msi signature is not trusted. 0x800B010C”

A certificate was explicitly revoked by its issuer.

Resolution

- Run the command *certmgr* from command prompt
- Check if MSI Installer Signing Root Certificate or MSI Installer timestamp Root Certificate is in *Untrusted Certificates* store.

- Copy it to *Trusted Root Certification Authority* store.

Issue 3

Windows Sensor upgrade fails with the following in TetUpdate.exe.log “Msi signature is not trusted. 0x80096005”

Resolution

- Run the command *certmgr* from command prompt
- Check if MSI Installer Signing Root Certificate and MSI Installer timestamp Root Certificate is in “Trusted Root Certification Authority” store

If it the certificate is missing, import it from other machine.

To import the certificate, follow below steps:

First export the certificate VeriSign Universal Root Certification Authority from one of Working server. Follow below steps:

- Run the command *certmgr* from command prompt
- Right click on the certificate “VeriSign Universal Root Certification Authority” under “Trusted Root Certification Authorities” and go to All tasksExport.
- Copy the exported certificate to the Non-working server and then import the certificate.

To import the certificate, follow below steps:

First export the certificate VeriSign Universal Root Certification Authority from one of Working server. Follow below steps:

- Run the command *certmgr* from command prompt
- Right click on the certificates tab under Trusted Root Certification Authorities and go to All tasksImport.
- Select the Root certificate that you copied and add it in the store.

Certificate Issues for NPCAP installer

Applicable to Windows 2012 , Windows 2012 R2, Windows 8, Windows 8.1

NPCAP version: 1.55

NPCAP Signing Certificate:

- Leaf Certificate: Insecure.Com LLC
- Intermediate Certificate: DigiCert EV Code Signing CA (SHA2)
- Root Certificate: DigiCert High Assurance EV Root CA

NPCAP Timestamp certificate:

- Leaf Certificate: DigiCert Timestamp 2021

- Intermediate Certificate: DigiCert SHA2 Assured ID Timestamping CA
- Root Certificate: DigiCert Assured ID Root CA

Issue 1

Windows Agent Installation might fail with below error in `msi_installer.log`

```
CheckServiceStatus : Exception System.InvalidOperationException: Service npcap was not found
  on computer
  \'. -> System.ComponentModel.Win32Exception: The specified service does not exist as an
  installed service
```

Resolution

- Run the command `certmgr` from command prompt
- Check “DigiCert High Assurance EV Root CA” in “Trusted Root Certification Authority” store.
- If it the certificate is missing, import it from other machine.

To import the certificate, follow below steps:

First export the certificate “DigiCert High Assurance EV Root CA” from one of Working server. Follow below steps:

- Run the command `certmgr` from command prompt
- Right click on the certificate “DigiCert High Assurance EV Root CA” under “Trusted Root Certification Authorities” and go to All tasksExport.
- Copy the exported certificate to the Non-working server and then import the certificate.

To import the certificate, follow below steps:

- Run the command `certmgr` from command prompt
- Right click on the certificates tab under Trusted Root Certification Authorities and go to All tasksImport.
- Select the Root certificate that you copied and add it in the store.

Applicable to Windows 2008 R2

NPCAP version: 0.991

NPCAP Signing Certificate:

- Leaf Certificate: Insecure.Com LLC
- Intermediate Certificate: DigiCert EV Code Signing CA
- Root Certificate: DigiCert High Assurance EV Root CA

NPCAP Timestamp certificate:

- Leaf Certificate: DigiCert Timestamp Responder
- Intermediate Certificate: DigiCert Assured ID CA-1

- Root Certificate: VeriSign DigiCert Assured ID Root CA

Issue 1

Windows Agent Installation might fail with below error in msi_installer.log

```
CheckServiceStatus : Exception System.InvalidOperationException: Service npcap was not found
on
computer \.'. -> System.ComponentModel.Win32Exception: The specified service does not exist
as an
installed service
```

Resolution

- Run the command *certmgr* from command prompt
- Check *DigiCert High Assurance EV Root CA* in *Trusted Root Certification Authority* store.
- If it the certificate is missing, import it from other machine.

To import the certificate, follow below steps:

First export the certificate “DigiCert High Assurance EV Root CA” from one of Working server. Follow below steps:

- Run the command *certmgr* from command prompt
- Right click on the certificate “DigiCert High Assurance EV Root CA” under “Trusted Root Certification Authorities” and go to All tasksExport.
- Copy the exported certificate to the Non-working server and then import the certificate.

To import the certificate, follow below steps:

- Run the command *certmgr* from command prompt
- Right click on the certificates tab under Trusted Root Certification Authorities and go to All tasksImport.
- Select the Root certificate that you copied and add it in the store.

Windows Host Rename

Scenario 1: Not able to see IP Addresses and VRF info after renaming the Windows Host Steps to fix the issue:

- Remove the entry(with new Hostname that is missing IP Addresses and VRF info) from the TaaS UI.
- Uninstall ‘Cisco Secure Workload Agent’ from the Windows Host and delete the ‘Cisco Tetration’ directory (typically the path for this will be : ‘C:Program FilesCisco Tetration’).
- Install ‘Cisco Secure Workload Agent’ on the Windows Host.

Following the above steps should register the Agent on the TaaS UI successfully with the IP Addresses and VRF info.

Scenario 2: Planned Windows Host rename (in advance) Steps to follow:

- Uninstall ‘Cisco Secure Workload Agent’ from the Windows Host and delete the ‘Cisco Tetration’ directory (typically the path for this will be : ‘C:\Program Files\Cisco Tetration’).
- Rename the Windows Host and Reboot.
- Install ‘Cisco Secure Workload Agent’ on the Windows Host(with new Hostname).

Following the above steps for planned Host rename should register the Agent on the TaaS UI with new Hostname.

Check If Platform Is Currently Supported

Windows

- Run the command *winver.exe*
- Compare this release to what is listed here: [Supported Platforms and Requirements](#)

Linux

- Run *cat /etc/os-release*
- Compare this release to what is listed here: [Supported Platforms and Requirements](#)

AIX

- Run the command *uname -a*
 - Note: The major and minor versions are reversed
- ```
p7-ops2> # uname -a
AIX p7-ops2 1 7 00F8AF944C00
```
- In this example, the first number after the host name is the minor and the second number is the major version, so AIX version 7.1. Compare this release to what is listed here: [Supported Platforms and Requirements](#)

## Windows Installer Issues

- Make sure there is a *C:\Windows\Installer* directory. This is not visible in File Explorer, easiest way to verify is in a CMD session and running: *dir C:\Windows\Installer*
- Check if the *Windows Installer* service is not disabled. It must be set to *Manual*
- Check to see if there are no other errors being reported by Windows Installer. Check Windows System Event logs under **Windows Logs > Application > Source > MsiInstaller**

## Required Windows Services

Below is a list of services, that when disabled, have been linked to installation issues of the agent. It is recommended these services are running during the initial installation and any upgrade of the Deep Visibility and Enforcement agents.

**Table 8: Required Windows Services**

| Service                | Purpose for installation                                                  |
|------------------------|---------------------------------------------------------------------------|
| Device Setup Manager   | Device driver management for the installation of the Npcap filter driver. |
| Device Install Service | Also used for the installation of the Npcap filter driver.                |
| Windows Installer      | Required for the installation of agent MSI package.                       |
| Windows Firewall       | Required for WAF enforcement mode.                                        |
| Application Experience | Used to determine compatibility executables on the system.                |



**Note** Application Experience service only applies to Windows Server 2008, 2008R2, 2012, 2012R2 and Windows 7. If disabled, a file lock may occur during Npcap installation causing it to fail.

## Npcap Issues

Npcap is a pcap tool used for Windows Agent only. Ten seconds after the agent service starts, it will attempt to install or upgrade Npcap to the supported version. If Npcap service fails to install or upgrade, the agent will retry the installation within the next 30 minutes. After 3 failed attempts, the agent will attempt to rollback Npcap to a previous supported version if available. After, the agent will no longer try to install Npcap. You can check *C:\Program Files\Cisco Tetration\Logs\TetUpdate.exe.log* and *C:\Program Files\Cisco Tetration\Logs\npcap\_install.log* to identify the error.

### Npcap will not upgrade (manually or via agent)

- Npcap will sometimes not uninstall correctly if a process is currently using the Npcap libraries. To check for this run the following command:

```
PS C:\Program Files\Npcap> .\NPFInstall.exe -check_dll
WindowsSensor.exe, Wireshark.exe, dumpcap.exe
```

If you see processes listed, they must be stopped before the Npcap upgrade can continue. If no processes are using Npcap the above command will simply show *<NULL>*

### Npcap will not install

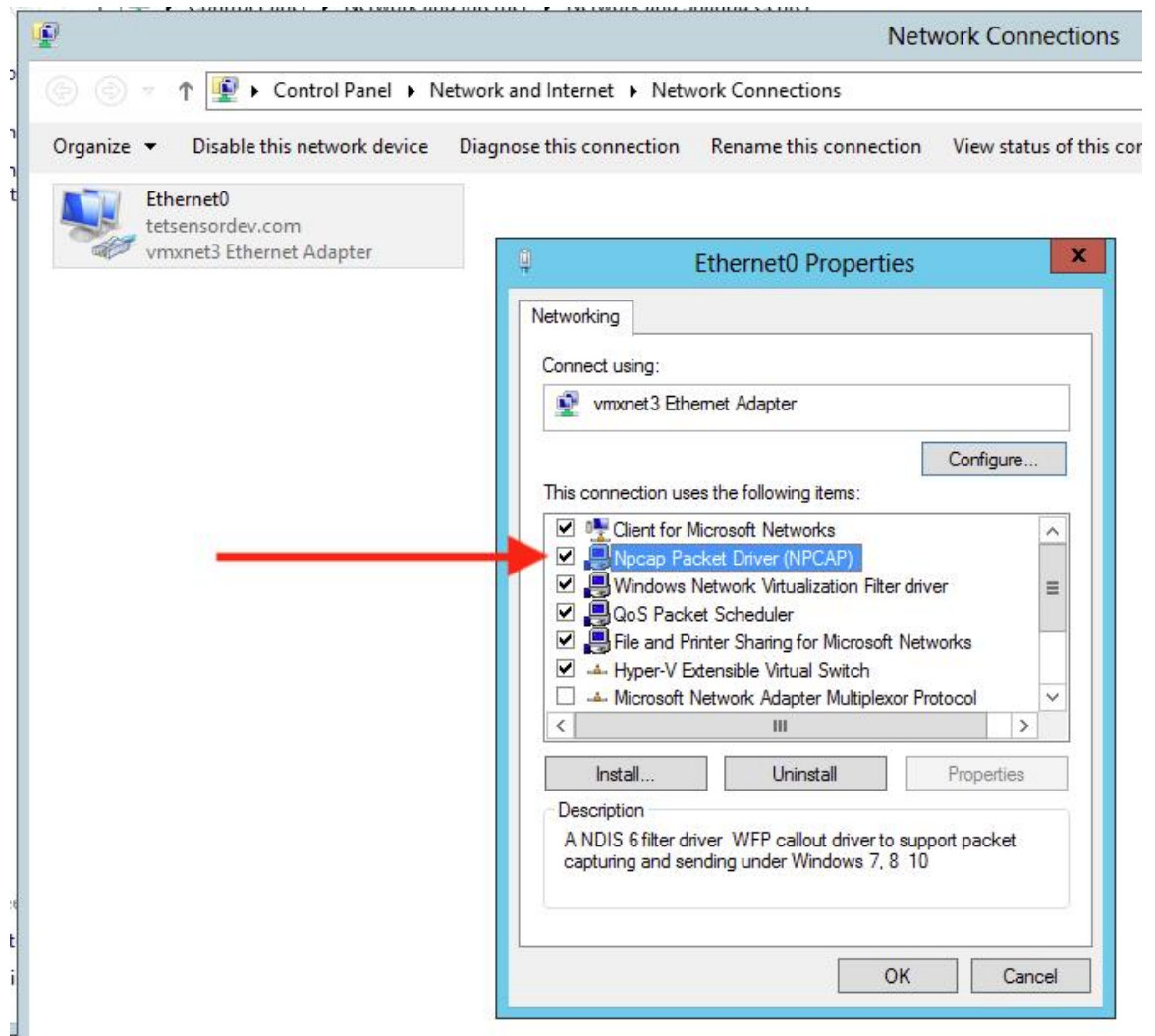
- Check CA certificates installed on the system: [Certificate Issues for NPCAP installer](#)
- Check Windows Installer issues: [Windows Installer Issues](#)

- Verify no other user on the system is making changes to the network interfaces. This can cause a COM lock preventing NDIS driver binding.

## Verify if Npcap is fully installed

### Procedure

- Step 1** Check **Control Panel > Programs and Features** to see if Npcap is listed as an installed application
- Step 2** Make sure the Npcap Packet Driver has a binding to the NIC in question (checkmark is present)



- Step 3** Check if the network driver is installed
- ```
C:\Windows\system32>pnputil -e | findstr Nmap
Driver package provider : Nmap Project
```
- Step 4** Check if the driver service is installed and RUNNING


```
C:\Windows\system32>sc query npcap
SERVICE_NAME: npcap
        TYPE : 1 KERNEL_DRIVER
        STATE : 4 RUNNING
```

Step 5 Check if the registry entry is there (used by the agent to verify Npcap exists already)

```
C:\Windows\system32>reg query HKLM\software\wow6432node\npcap
HKEY_LOCAL_MACHINE\software\wow6432node\npcap
        AdminOnly REG_DWORD 0x1
        WinPcapCompatible REG_DWORD 0x0
        (Default) REG_SZ C:\Program Files\Npcap
```

Step 6 Check if the installed Npcap program files are all there

```
C:\Windows\system32>dir "c:\program files\npcap"
Directory of c:\program files\npcap
04/29/2020 02:42 PM <DIR> .
04/29/2020 02:42 PM <DIR> ..
01/22/2019 08:16 AM 868 CheckStatus.bat
11/29/2016 03:43 PM 1,034 DiagReport.bat
12/04/2018 11:12 PM 8,908 DiagReport.ps1
01/09/2019 09:22 PM 2,959 FixInstall.bat
04/29/2020 02:42 PM 134,240 install.log
01/11/2019 08:52 AM 9,920 LICENSE
03/14/2019 08:59 PM 10,434 npcap.cat
03/14/2019 08:57 PM 8,657 npcap.inf
03/14/2019 09:00 PM 74,040 npcap.sys
03/14/2019 08:57 PM 2,404 npcap_wfp.inf
03/14/2019 09:00 PM 270,648 NPFInstall.exe
04/29/2020 02:42 PM 107,783 NPFInstall.log
03/14/2019 09:01 PM 175,024 Uninstall.exe
13 File(s) 806,919 bytes
2 Dir(s) 264,417,628,160 bytes free
```

Step 7 Check to see if the .sys driver file is in the Windows driver folder

```
C:\Windows\system32>dir "C:\Windows\System32\Drivers\npcap.sys"
Directory of C:\Windows\System32\Drivers
03/14/2019 09:00 PM 74,040 npcap.sys
1 File(s) 74,040 bytes
```

Network Connectivity issues during NPCAP installation or upgrade

Applicable to Windows 2016 Only

If you have a 3rd party LWF (Light Weight Filter) driver (e.g. netmon) or a teaming adapter is configured in your setup, and NPCAP is installed during agent deployment, you might experience

RDP is reconnected

NetBios service is restarted

Similar network connectivity issues

This is due to a BUG in Windows 2016 OS

NIC teaming compatibility issues with NPCAP

Teaming NIC functionality is based on underneath Physical NICs (Intel, Broadcom, Realtek, MS virtual adapter etc) and Teaming driver configuration (switch based, loadbalancing or failover, algorithm to distribute the packets across multiple NICs).

Some NPCAP versions have compatibility issues with Teaming NICs, especially during binding to the underneath Teaming NICs.

The current Secure Workload Sensor software is tested using Microsoft supported NIC teaming.

```
NIC type : Intel(R) 82574L Gigabit Network Connection
Teaming Mode : Switch Independent
Load Balancing Mode: Address Hash
OS : Windows 2012 , Windows 2012 R2, Windows 2016, Windows 2019
NPCAP version: 1.55
```



Note Windows 2008R2 does not support Microsoft supported NIC teaming.

VDI instance VM does not report network flows

The TetSensor service occasionally does not capture the network flows on cloned VMs when NPCAP service is running. This can happen when the agent is installed without the **nostart** flag using MSI installer or without **goldenImage** flag using PowerShell Installer on a VM template or golden image.

In this case, Secure Workload agent services start running on the VM template. NPCAP is installed and bound to the Network stack on the VM template. When a new VM is cloned from the VM template, NPCAP is not properly bound to the Network stack on the new cloned VM. As a result, NPCAP fails to capture the network flows.

Network Performance with NPCAP

It is observed that Network performance will be affected when Windows TetSensor service is running. Windows Tet- Sensor service (tetsen.exe) captures the network flows using NPCAP. NPCAP implementation to capture the network flows and the network flows to the tetsen.exe affects the network performance.

Compare the Network Performance after installing tetsensor, Client : Windows 2016

NPCAP 1.55

TetSensor Config : Conversation Mode with Enforcement mode WFP

Server : Windows 2016

NPCAP 1.55

TetSensor Config : Conversation Mode with Enforcement mode WFP

Run cmd : iperf3.exe -c <server_ip> -t 40

Table 9: 121071: Network Performance with NPCAP 155

| Setup | Network Performance |
|--|---|
| No TetSensor Installed NO NPCAP | [ID] Interval Transfer Bandwidth [4] 0.00-40.00 sec 18.2 GBytes 3.90 Gbits/sec sender [4] 0.00-40.00 sec 18.2 GBytes 3.90 Gbits/sec receiver |
| TetSensor Installed NPCAP Installed | [ID] Interval Transfer Bandwidth [4] 0.00-40.00 sec 17.3 GBytes 3.72 Gbits/sec sender [4] 0.00-40.00 sec 17.3 GBytes 3.72 Gbits/sec receiver |

Network Performance with NPCAP 0.9990

Compare the Network Performance after installing tetsensor, Client : Windows 2016

NPCAP 0.9990

TetSensor Config : Conversation Mode with Enforcement mode WFP

Server : Windows 2016

NPCAP 0.9990

TetSensor Config : Conversation Mode with Enforcement mode WFP

Run cmd : iperf3.exe -c <server_ip> -t 40 .. table:: Network Performance with NPCAP 0.9990

class longtable

| Setup | Network Performance |
|--|---|
| TetSensor Installed NPCAP Installed | [ID] Interval Transfer Bandwidth [4] 0.00-40.00 sec 16.3 GBytes 3.50 Gbits/sec sender [4] 0.00-40.00 sec 16.3 GBytes 3.50 Gbits/sec receiver |



Note Performance may vary based on Windows NPCAP version installed, Windows OS, and network Configuration.

OS Performance and/or stability Issues

OS may experience unknown performance or stability issues if the installed NPCAP version or NPCAP configuration is not supported by the Secure Workload Software.

Supported NPCAP Version: : 0.991 and 1.55

GPO Configurations

Agents that enforce policy require only the Firewall to be enabled with either a local setting or GPO. All other GPO settings should not be set and left as “Not Configured.”

- To check if a GPO setting is blocking enforcement you can check the *C:\Program Files\Cisco Tetration\Logs\TetEnf.exe.log* log and search for the following error examples:
- **Rules conflicting with “Preserve Rules=No” setting:** “There are firewall rules set in the Group Policy. Secure Workload agent does not have permission to remove these”
- **Firewall set to off:** “GPO has disabled firewall for DomainProfile”
- **Default Action is set:** “Group Policy has conflicting default inbound action for DomainProfile”
- To check what GPO policies are being applied to the host, run *gpresult.exe /H gpresult.html* and open the generated HTML report. In the example below *Secure Workload Agent Firewall* is applying a Inbound rule which will conflict with Enforcement if “Preserve Rules” is set to “No.”

| Policy | Setting | Winning GPO |
|---------------------------------------|----------------|--------------------------|
| Firewall state | On | Tetration Agent Firewall |
| Inbound connections | Not Configured | |
| Outbound connections | Not Configured | |
| Apply local firewall rules | Not Configured | |
| Apply local connection security rules | Not Configured | |
| Display notifications | Not Configured | |
| Allow unicast responses | Not Configured | |
| Log dropped packets | Not Configured | |
| Log successful connections | Not Configured | |
| Log file path | Not Configured | |
| Log file maximum size (KB) | Not Configured | |

| Inbound Rules | | |
|--------------------|-------------|--------------------------|
| Name | Description | Winning GPO |
| HTTPS Inbound Rule | | Tetration Agent Firewall |

Inbound/Outbound Rules Not Recommended

This rule might contain some elements that cannot be interpreted by the current version of GPMC reporting module. Enabled True

Agent To Cluster Communications

The Secure Workload Agent maintains connections to the cluster over multiple channels. Depending on the type of Agent, the number of connections varies.

Types of connections

- **WSS:** Persistent socket connection over port 443 to the cluster
- **Check in:** A HTTPS call to the cluster every 15-20 minutes to check for current configurations, check for updates and to update the active state of the agent to the cluster. This also reports upgrade failures.
- **Flow export:** Persistent SSL connection over port 443 (TaaS) or 5640 (On-premise) to send flow metadata to the cluster
- **Enforcement:** Persistent SSL connection over port 443 (TaaS) or 5660 (On-premise) to pull in enforcement policies and report enforcement state

Checking the connection state

The Teration UI will report either an inactive agent (no longer checking-in), no exported flows (on Agent Workload Profile page under Stats), or failed enforcement. Depending on the error, you can check different logs on the workload to help determine the source of the issue.

Inactive Agent

Windows Log: *C:\Program Files\Cisco Tetration\Logs\TetUpdate.exe.log*

Linux Log: */usr/local/tet/logs/check_conf_update.log*

An HTTP response code of 304 is expected and means there is no configuration change. Error code = 2 is expected as well. Any other HTTP response code will indicate a issue talking to the WSS service on the Secure Workload cluster.

```
Tue 06/09/2020 17:25:25.08 check_conf_update: "curl did not return 200 code, it's 304,
.-> exiting"
Tue 06/09/2020 17:25:25.08 check_conf_update: "error code after running check_conf_
.->update = 2"
```

- **304** Expected, no config change. Successful check-in
- **401** Registration is not successful, missing Activation Key (TaaS)
- **403** Agent already registered to the cluster with same UUID
- **000** Indicates connection issue with SSL. Either curl could not reach the WSS server or there is a issue with the certificate. See SSL troubleshooting: [SSL Troubleshooting](#)

No exported flows

Windows Log: *C:\Program Files\Cisco Tetration\Logs\TetSen.exe.log*

Linux Log: */usr/local/tet/logs/tet-sensor.log*

The following indicates a successful connection to WSS

```
cfgserver.go:261] config server: StateConnected, wss://<config_server_ip>:443/wss/
.-><sensor_id>/forensic, proxy:
```

The following indicates a successful connection to the Collectors

```
collector.go:258] next collector: StateConnected, ssl://<collector_ip>:5640
```

If there are errors connecting to either WSS or the Collectors, check your firewall configuration or verify if any SSL decryption is occurring between the agent and Secure Workload. See: [SSL Troubleshooting](#)

Failed to enforce policy

Windows Log: *C:\Program Files\Cisco Tetration\Logs\TetEnf.exe.log*

Linux Log: */usr/local/tet/logs/tet-enforcer.log*

```
ssl_client.cpp:341] Successfully connected to EFE server
```

If there are errors connecting to the EFE server, check your firewall configuration or verify if any SSL decryption is occurring between the agent and Secure Workload. See: [SSL Troubleshooting](#)

SSL Troubleshooting

Agent Communications Overview

Secure Workload agents use TLS to secure the TCP connections to the Secure Workload Cloud SaaS servers. These connections are broken down into three distinctive channels.

- Agent -> Cisco Secure Workload SaaS control channel over port TCP/443 (TLS) (sensorVIP)

This is a low volume control channel that allows the agent to register with Secure Workload and also handles configuration pushes and software upgrade notifications.

- Agent -> Cisco Secure Workload SaaS flow data over TCP/443 (TLS) (collector)

Flow data is the extracted flow metadata information; the data will be sent to 1 set of 16 IP addresses at a time. The second set of IP addresses is for standby. This is around 1 – 5% of actual server traffic.

- Agent -> Cisco Secure Workload SaaS enforcement data over TCP/443 (TLS) (efe)

The enforcement data channel is a low volume control channel that is used to push the policies to the sensors and also gather enforcement statistics.

The sensor validates the TLS certificate from the Secure Workload Cloud control, data and enforcement servers against a local CA that is installed with the agent. No other CAs are used, so any other certificate sent to the agent will result in a verification failure and the agent will not connect. This will result in the agent not registering, checking-in, sending flows or receiving enforcement policies.

Configuring IP traffic for Agent Communications

A typical configuration for most will be to have a perimeter firewall and possibly a proxy between the agents (workflows) and Secure Workload TaaS.



Note Secure Workload gathers your gateway/NAT IP information during the on-boarding and automatically adds the information at the time of tenant creation. If you add new IP addresses or change IP addresses in the portal, the changes require review and approval by Secure Workload staff.

In addition to adding your gateway/NAT IP addresses in the TaaS portal, there might be more changes required to your network to allow the traffic outbound and unmodified:

Allow outbound port 443 over TLS/HTTPS on the perimeter firewall

Configure proxy bypass and SSL/TLS bypass on the web proxy, if a decrypting web proxy is being used.



Note If you are using a transparent web proxy at the data center, you must route the specific SaaS IP address and configure the bypass rules. Sensors are connections that cannot do automatic HTTPS redirection.

The list of IPs the agents communicates with is available on the TaaS portal. The IPs to add to your firewall outbound configuration and proxy bypass are labeled collector-n, efe-n (only if enforcement is being deployed), and sensorVIP. There are typically 17 to 33 IPs to add for agent communication, but there could more or less depending on your TaaS configuration.



Note Ensure that the outbound firewall rules allow traffic to flow to the destination metadata collector IPs.



Note You can now edit the outbound gateway IP addresses on the user interface. This functionality allows the allowed metadata traffic originating from your workloads to their destination.

Troubleshooting SSL/TLS Connections

As discussed in the previous section, it is important to configure your explicit or transparent web proxy to bypass SSL/TLS decryption for agent communications. If the bypass is not configured, these proxies might attempt to decrypt

SSL/TLS traffic by sending its own certificate to the agent. Because the agent only uses its local CA to validate the certificate, these proxy certificates will cause connection failures.

Symptoms include agent failing to register to the cluster, agent not checking-in, agent not sending flows, and/or agent not receiving enforcement configuration (if enforcement is enabled).



Note Troubleshooting steps below are assuming default installation paths were used. Windows: C:\Program Files\Cisco Tetration Linux: /usr/local/tet. If you installed your agents in a different location, substitute that location in the instructions.

SSL/TLS Connection issues are reported in the agent logs. To verify if there are SSL errors in the logs, run the following commands for the associated issue being observed.

Registration, check-in

Linux

```
grep "NSS error" /usr/local/tet/log/check_conf_update.log
```

Windows (PowerShell)

```
get-content "C:\Program Files\Cisco Tetration\logs\TetUpdate.exe.log" | select-  
->string -pattern "curl failed SSL peer certificate"
```

Flows

Most of the SSL/TLS connection issues seen are during the initial connection and registration of the agent. Sending flows relies on the registration to be complete before attempting to connect. SSL/TLS errors seen here would be the result of the sensorVIP IPs being allowed but not the collector IPs.

Linux

```
grep "SSL connect error" /usr/local/tet/log/tet-sensor.log
```

Windows (PowerShell)

```
get-content "C:\Program Files\Cisco Tetration\logs\WindowsSensor*.log" | select-
->string -pattern "Certificate verification error"
```

Enforcement

Linux

```
grep "Unable to validate the signing cert" /usr/local/tet/log/tet-enforcer.log
```

Windows (PowerShell)

```
get-content "C:\Program Files\Cisco Tetration\logs\WindowsSensor*.log" | select-
->string -pattern "Handshake failed"
```

If an SSL error is seen in the log checks above you can verify what certificate is being sent to the Agents with the following commands.

Explicit Proxy - where a proxy is configured in user.cfg

Linux

```
curl -v -x http://<proxy_address>:<port> https://<sensorVIP>:443
```

Windows (PowerShell)

```
cd "C:\Program Files\Cisco Tetration"
.\curl.exe -kv -x http://<proxy_address>:<port> https://<sensorVIP>:443
```

Transparent Proxy - No user.cfg proxy configuration required. It's a proxy configured between all HTTP(S) traffic from agent to the internet.

Linux

```
openssl s_client -connect <sensorVIP from TaaS Portal>:443 -CAfile /usr/local/tet/
->cert/ca.cert
```

Windows (PowerShell)

```
cd C:\Program Files\Cisco Tetration
.\openssl.exe s_client -connect <sensorVIP from TaaS Portal>:443 -CAfile cert\ca.cert
```

You are looking for the following in the openssl s_client response

```
Verify return code: 0 (ok)
```

If you see an error, examine the certificate. An example certificate (chain) should include only the following cert (CN IP is an example):

Certificate chain

```
0 s:/C=US/ST=CA/L=San Jose/O=Cisco Systems, Inc./OU=Tetration, Insieme BU/CN=129.146.
->155.109
i:/C=US/ST=CA/L=San Jose/O=Cisco Systems, Inc./OU=Tetration Analytics/CN=Customer CA
```


If you see additional certificates, then there is possibly a Web decrypting proxy between the agent and Secure Workload. Contact your security or network group and verify if the proxy bypass is configured using the listed IPs from the Configuring IP traffic for Agent Communications section.

Windows sensor installation script fails on Windows 2016 servers: Error message that might appear “The underlying connection was closed: An unexpected error occurred on a receive.” Possible reason might be the SSL/TLS versions set in PowerShell.

To check the SSL/TLS versions running, run the following command:

```
[Net.ServicePointManager]::SecurityProtocol
```

If the output from the above command is:

```
Ssl3, Tls
```

Then use the below command to change the allowed protocols and retry the installation:

```
[Net.ServicePointManager]::SecurityProtocol = [System.Net.SecurityProtocolType]'Ssl3,  
,Tls,Tls11,Tls12'
```

Agent operations

Q: I have installed the agents successfully, but I didn't see it on UI Sensor Monitoring page.

A: An agent is required to register with backend server running within cluster before it could start operating. When an agent is not shown on UI page, most likely it's because the registration has failed. There are a few things we could check to see why a registration failed:

- Check if the connection between the agent and the backend server is working properly
- Check if the curl request could be sent to backend server properly
- Check HAProxy access and backend server logs to see if the registration request made it to the server
- Check the error return from curl request in the log file

Q: The agent is installed and I could find in on UI page. However, the “SW Ver” column shows “initializing” instead of a version string.

A: After the initial agent is installed and registered with the backend server, it would take another 30 minutes for the agent to report its version.

Q: The agent is upgraded properly, but the “SW Ver” fields still show the old version after a long time (like several hours).

A: After the agent is upgraded successfully, it will try to send a curl request to report its current running version and check for new version in the same request. It is possible that the request couldn't make it to the backend, due to several reason:

- The request is timed out, couldn't get the response in time
- The network is facing problem, agent couldn't connect to backend servers

Q: I have an agent running on RHEL/CentOS-6.x and it is working properly. I am planning to upgrade the OS to RHEL/CentOS-7.x. Would the agent still work after the upgrade?

A: currently we do not support the scenario in which the OS has been upgraded, especially upgrading the major releases. In order to have the agent work after OS upgrade, do the following steps:

- Uninstall the existing agent software
- Clean up all files, including certs
- Go to UI, delete the agent entry
- Upgrade the OS to the desired version
- Install the agent software on the new OS

Q: I have an agent running on RHEL/CentOS-6.x and it is working properly. I am planning to rename the host. Would the agent still work after rename/reboot?

A: An agent identity is calculated based on the host's uniqueness, including hostname and bios-uuid. Changing hostname changes the host's identify. It is recommended to do the following:

- Uninstall the existing agent software
- Clean up all files, including certs
- Go to UI, delete the old agent entry
- Rename the host and reboot
- Install the agent software again

Q: On Windows host, firewall deviation was caused by adding/deleting/modifying a rule. How do I find the rule?

A: On deviation detection, agent logs the last 15 seconds of firewall events to "C:\Windows\System32\config\systemprofile\AppData\Roaming\tet\firewall_events". Rule that caused deviation will be found in the latest file created as policy_dev_<policy id>_<timestamp>.txt

Q: I have installed the agent on a Windows host successfully. Why do I not see any reported flows from the sensor?

A: Npcap is required to collect flows on a Windows host. Ten seconds after the agent is installed successfully, it will install Npcap. If the sensor does not report flows after several minutes, check if the agent and the backend server is connected and if Npcap is installed properly on the [Npcap Issues](#).

Q: I have installed the agent on Windows host, 2008 R2, successfully. Why does the system clock drift when tetsensor service is running?

A: This is a known problem with Go and Windows 2008 R2. For more information, see [Golang and Win2008 R2](#).

The process, tet-main.exe, running as a part of tetsensor service, is built using Go Version 1.15. That is why the system clock drifts when the tetsensor service is running.

This issue occurs when Windows 2008 R2 workload is configured to use the external NTP server or Domain Controller as NTP server.

The possible work around :

1. Periodically force NTP to sync the clock: w32tm /resync /force
2. Disable tet-main.exe manually.
 - Run cmd.exe with "administrator" privilege.
 - Run regedit.exe

- Go to “HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Services\TetSensor”
- Double click on “ImagePath”
- Edit value, remove tet-main.exe
before “C:\Program Files\Cisco Tetration\TetSenEngine.exe” TetSensor TetSen.exe “-f sensor_config”
tet-main.exe ” ” TetUpdate.exe
after “C:\Program Files\Cisco Tetration\TetSenEngine.exe” TetSensor TetSen.exe “-f sensor_config”
TetUpdate.exe
- Restart tetsensor service



Note Disable tet-main.exe after every time agent is upgraded.

3. Remove external NTP server configuration:

- Run command : w32tm /config /update /manualpeerlist: /syncfromflags:manual /reliable:yes
- Restart Windows Time Service, W32Time



CHAPTER 3

External Orchestrators in Secure Workload

External orchestrators are used to gather existing metadata describing your workloads from systems on your network. Some external orchestrators can also enforce segmentation policy.

For deployments where an authorized system of record exists with labels for workloads, we provide a way for automatically importing the labels through external orchestrator integrations. Any modifications in the system of record will be learnt automatically by Secure Workload and used for updating labels in your inventory.

For detailed information about the power and uses of labels, see [Workload Labels](#).

Due to recent GUI updates, some of the images or screenshots used in the user guide may not fully reflect the current design of the product. We recommend using this guide in conjunction with the latest version of the software for the most accurate visual reference.

- [Navigate to the External Orchestrators Page, on page 123](#)
- [List of External Orchestrators, on page 124](#)
- [Create External Orchestrator, on page 126](#)
- [Edit External Orchestrator, on page 129](#)
- [Delete External Orchestrator, on page 130](#)
- [Orchestrator generated labels, on page 130](#)
- [Secure Connector, on page 130](#)
- [Amazon Web Services, on page 138](#)
- [Kubernetes/OpenShift, on page 140](#)
- [VMware vCenter, on page 148](#)
- [DNS, on page 150](#)
- [Infoblox, on page 152](#)
- [F5 BIG-IP, on page 155](#)
- [Citrix Netscaler, on page 161](#)
- [TAXII, on page 165](#)

Navigate to the External Orchestrators Page

The main page for external orchestrators can be reached by selecting **Manage > Workloads > External Orchestrators** from the menu bar on the left.

List of External Orchestrators

The External Orchestrators page shows the existing external orchestrators and provides functions to modify and delete them as well as to create new external orchestrators:

Table 10: External Orchestrators

| Type | Description/ When to use |
|----------------------|---|
| VMware vCenter | To import virtual machine data, such as host name, IP address, and labels, from a vCenter server to Secure Workload. The generated labels can be used to create Secure Workload scopes and enforcement policies. |
| Amazon Web Services | (You cannot create new AWS orchestrators; instead, create AWS connectors. See AWS Connector . Any existing AWS orchestrators are read-only). To import data of EC2 server instances, such as host name, IP address, and labels, from the given AWS account to Secure Workload. The generated labels are useful to create Secure Workload scopes and policies. |
| Kubernetes/OpenShift | To import Kubernetes' entities, such as nodes, pods, services, and labels. These labels can be used within Secure Workload to define scopes and policies. |
| DNS | To import A/AAAA and CNAME records from a DNS server via zone transfer. This produces DNS names as labels, which are useful in defining Secure Workload scopes and policies. |
| Infoblox | To import networks, hosts, and A/AAAA records with extensible attributes from an Infoblox appliance with IPAM/DNS enabled. The imported extensible attributes can be used as labels in Secure Workload scopes and policies. |
| F5 BIG-IP | To read virtual server configurations from the given F5 load balancer and generate labels for the provided services, which can be used to define enforcement policies in Secure Workload. The policy enforcement feature will translate them into F5 policy rules via F5 REST API. |
| Citrix Netscaler | To read virtual server configurations from the given Netscaler load balancer and generate labels for the provided services, which can be used to define enforcement policies in Secure Workload. The policy enforcement feature will translate them into Netscaler ACLs via its REST API. |

| Type | Description/ When to use |
|-----------------------------------|--|
| Secure Firewall Management Center | To deploy policies to all Secure Firewall Threat Defense (formerly known as Firepower Threat Defense or FTD) devices registered to the given Secure Firewall Management Center using REST API. |

Figure 28: External Orchestrator

| Name | Type | Description | Enforcement | Created At | Connection Status | Secure Connector Status | Actions |
|------------|------------------|-------------------------|-------------|--------------------------|-------------------|-------------------------|---------|
| fmc-test-1 | FMC | arbhata NPI | Enabled | Jul 19 10:16:55 pm (IST) | Success | | |
| FS | FS BIG-IP | FS orchestrator | Disabled | Jul 19 11:34:44 pm (IST) | Success | Success | |
| Citrix NS | Citrix Netscaler | Citrix NS | Enabled | Jul 19 11:36:24 pm (IST) | Failure | | |
| K8S | Kubernetes | Kubernetes orchestrator | N/A | Jul 19 11:39:38 pm (IST) | Failure | Success | |

Each row shows a short version of the external orchestrator with its *Name*, *Type*, *Description*, *Enforcement*, *Created at*, *Connection Status*, and *Secure Connector Status*. The *Connection Status* tells if a connection to the given external data source could be made successfully. The *Secure Connector Status* displays the status of the Secure Connector tunnel- Success or Failure. If the tunnel is not enabled then N/A is displayed.

Enable Secure Connector tunnel while creating an external orchestrator configuration. If Secure Connector tunnel is enabled, the external orchestrator 'Connection Status' is dependent on both the authentication status and secure connector status. If Secure Connector tunnel is not enabled, the external orchestrator 'Connection Status' is only dependent on the authentication status. Irrespective of the status- Success or Failure, you can click the respective row to get more details. For more details on the Secure Connector client metrics, click the **Status** row or in the left pane, navigate to **Manage > Workloads > Secure Connector**.

Figure 29: External Orchestrator Authentication Failure

Configuration Details

| | |
|------------------------------|--|
| Id | 6206f36275f825848a8e9f8 |
| Type | Kubernetes |
| Name | K8S |
| Description | Kubernetes orchestrator |
| Delta Interval (s) | 60 |
| Full Snapshot Interval (s) | 3600 |
| Certificate | -----BEGIN CERTIFICATE----- MIIDCCFCgphbG81Llud8B1xLpQDYJkoZJhVnMAELB0wFTETMBGA1UE Aww31Z3JzZDR1czAafuyljA9T1u9zA5T8aFylytA9T1u9zE2MTBwDQo F3VnB0wFTETMBGA1UEAww31Z3JzZDR1czAafuyljA9T1u9zA5T8aFylytA9T1u9zE2MTBwDQo bWUwFTETMBGA1UEAww31Z3JzZDR1czAafuyljA9T1u9zA5T8aFylytA9T1u9zE2MTBwDQo e-cub9k1f4z4fj9E25uim9P9M1100L2z9F1L6f1L2750T5L6A N0u9g8R5kZgvt9Mf1k/200V1g9pSAU668brr06LPHb+cc7N480TYvG4c q9m6zuzerj35c7h9vrc0M18Bf6wJ8PheZv0m9P7z1F1z4K7T1A6uLk9M 17F3k46d6k9k1k112610u1m6b6w+KwP1+HE8BpT8E2838P8Y809 p0zV0hg0F7z11te019v+Z77kgz1Zu1F80YkC4V9Z8CC7X70bb0vAbh q9P9z0268AB8pWLAB98V9084FEB8C8k6w0V081Bhw07XK9T808 A0Lk9V9P8B8BpWuA128LPS170q081EgM1H9T1H9Z0p0YJkoZJhVnMAEL B0wFTETMBGA1UEAww31Z3JzZDR1czAafuyljA9T1u9zA5T8aFylytA9T1u9zE2MTBwDQo vE/3h8C9y5sLuKk088Y1ccp208U0EYg81L8L5CK8BpPh3584R8V9K3 Ecy13g1z1k961F7k1E91d0M26u85F8kV9v+109yHf110m02L80m808 r07P8F8CCTF9g1KCY1k4C18C1v8z86uL8V8w8P8C8p8g8p8t8d8E q8V8m1z3g1K8V8p8B251C15c12889h2kQF7w8181cy8m1k181f/cg8J3U q8K7k8c9p8808T8k1T8j8p8L1008-----END CERTIFICATE----- |
| Accept Self-signed Cert | true |
| Secure Connector Tunnel | true |
| Golden Rules | () |
| Hosts List | C"host_name":"192.168.18.2","port_number":6443} |
| Authentication Failure | true |
| Authentication Failure Error | 6206f36275f825848a8e9f8 connection failure: 6206f36275f825848a8e9f8 ServerVersion read failure: the server has asked for the client to provide credentials |
| Peers | 172.28.171.191:68768 |
| Status | Secure Connector Status + Connection Status > Status Success Failure Failure |

Create External Orchestrator

A new external orchestrator can be created by clicking the **Create New Configuration** button in the external orchestrators main page. This leads to a modal dialog, where you can enter a name and choose an external orchestrator type. The picture below shows the basic configuration page:

Figure 30: Create External Orchestrator Configuration

The following table describes the common fields for external orchestrators. Depending on the selected type the *Basic Config* page requires additional parameters to be given. These will be covered by the respective section of the individual external orchestrators below.

| Common Field | Required | Description |
|--------------|----------|--|
| Type | Yes | Select an external orchestrator from the list. |
| Name | Yes | Name of the external orchestrator, which must be unique for the active tenant. |

| Common Field | Required | Description |
|---------------------------|----------|--|
| Description | No | Description of the external orchestrator. |
| Full Snapshot Interval(s) | Yes | Interval in seconds the external orchestrator will try to import the full snapshot of configuration from the selected <i>Type</i> . |
| Accept Self-signed Cert | No | Check this option to accept self-signed server certificates for the HTTPS connection used by Secure Workload to retrieve configuration data from the selected <i>Type</i> . Default is not to allow self-signed server certificates. |
| Secure Connector Tunnel | No | Check this option to set connections to the Secure Workload cluster to be tunneled through a Secure Connector tunnel. |



Note The fields *Delta interval* and *Verbose TSDB Metrics* as shown in the picture above are optional and applicable only for certain external orchestrators, which are explained in the respective description below.

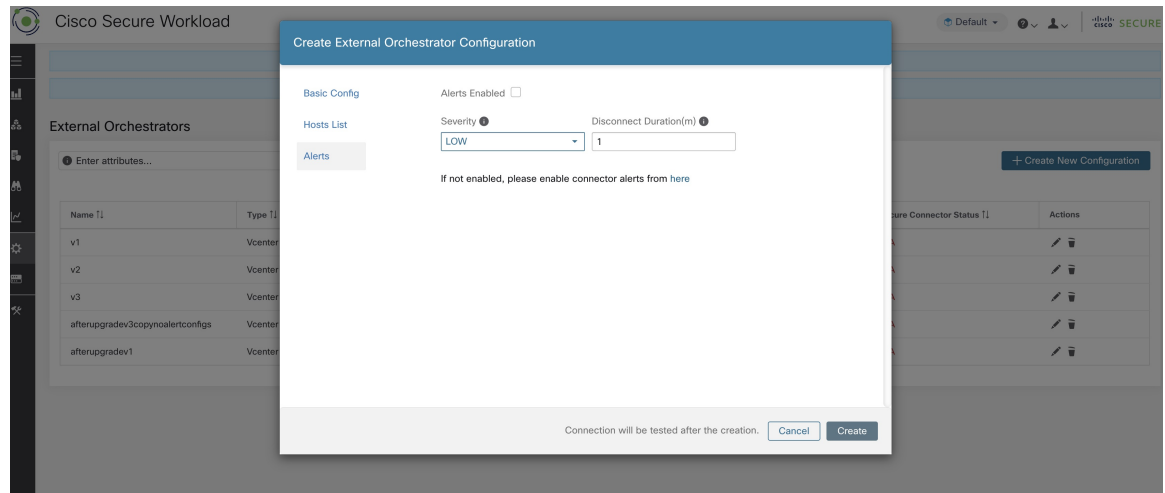
Except for the external orchestrator type *AWS*, the *Hosts List* must be given. It specifies the network address(es) of the external data source from which the external orchestrator will fetch data and generate labels. This can be done by clicking on the tab *Hosts List* on the left hand side, which is shown in the following picture:

Figure 31: External Orchestrator's Hosts List

In order to add new host list entry click the plus sign. Each row must contain a valid DNS host name, IPv4 or IPv6 address and a port number. Depending on the chosen external orchestrator type, you can enter multiple hosts for high availability or redundancy purposes. For more information, see the description for the chosen external orchestrator.

To set the alert for the external orchestrator, this can be done by clicking on the *Alert* tab on the left hand side, which is shown in the following picture:

Figure 32: External Orchestrator's Alerts



For each external orchestrator, configuring *alerts* requires additional parameters to be given. These will be covered by the respective section of the individual external orchestrators below.

To enable alerts for this external orchestrator, check the *Alert enabled* check box.



Note Make sure that Connector Alerts are also enabled from **Manage > Workloads > Alert Configs** page.

Select the *Alert Severity* level and *Disconnect Duration* in minutes for configuring external orchestrator alert.

| Field | Description |
|------------------------|---|
| Severity | Select severity level of this rule: LOW, MEDIUM, HIGH, CRITICAL or IMMEDIATE ACTION |
| Disconnect Duration(m) | The amount of time that a connection is disconnected. |

Click the **Create** button to create the new external orchestrator, whose configuration details can be viewed by clicking on the respective row in the list view:

Figure 33: External Orchestrator's Configuration Details

| Configuration Details | |
|------------------------------|--|
| Id | 59e15d2f755f02424c0ff38a |
| Type | Vcenter |
| Name | mock_config |
| Description | mockdata |
| Delta Interval (s) | 60 |
| Full Snapshot Interval (s) | 3600 |
| Username | mock |
| Password | changeme |
| Certificate | asd |
| Key | 123 |
| Secure Connector Tunnel | true |
| Authentication Failure Error | e1 |
| Peers | 172.31.182.228:45906 |
| Status | Secure Connector Status + Connection Status > Status + Success ✔ Success ✔ Success ✔ |





Note Since the first full snapshot pull from an external orchestrator is an asynchronous operation, expect about one minute for the connection status field to be updated.

Edit External Orchestrator

Click the pencil button on the right hand side of an external orchestrator row as shown below to open a modal dialog similar to the one for creating an external orchestrator, where the configuration can be modified.

Figure 34: Edit External Orchestrator

| Name ↑ | Type ↓ | Description ↑ | Enforcement ↑ | Created At ↑ | Connection Status ↑ | Edit ↑ |
|-------------|---------|---------------|---------------|---------------------------|---------------------|---|
| mock_config | Vcenter | mockdata | N/A | Oct 14 03:41:19 am (EEST) | Success |   |



- Note**
- The **Type** field is not editable.
 - If a configuration uses keys/certificates for authentication, the keys and certificates have to be provided every time the configuration is updated.
 - Since the configuration changes of an external orchestrator is an asynchronous operation, expect about one minute for the connection status field to be updated and to confirm the correctness of entered changes.

Click the **Update** button to save the changes made to the configuration.

Delete External Orchestrator



Caution Deleting an external orchestrator also deletes labels provided by that orchestrator, which will impact policies. In order to delete an external orchestrator click the trash bin button as shown below:

Figure 35: Delete External Orchestrator

| Name | Type | Description | Enforcement | Created At | Connection Status | |
|-------------|---------|-------------|-------------|---------------------------|-------------------|--------|
| mock_config | Vcenter | mockdata | N/A | Oct 14 03:41:19 am (EEST) | Success | Delete |

Orchestrator generated labels

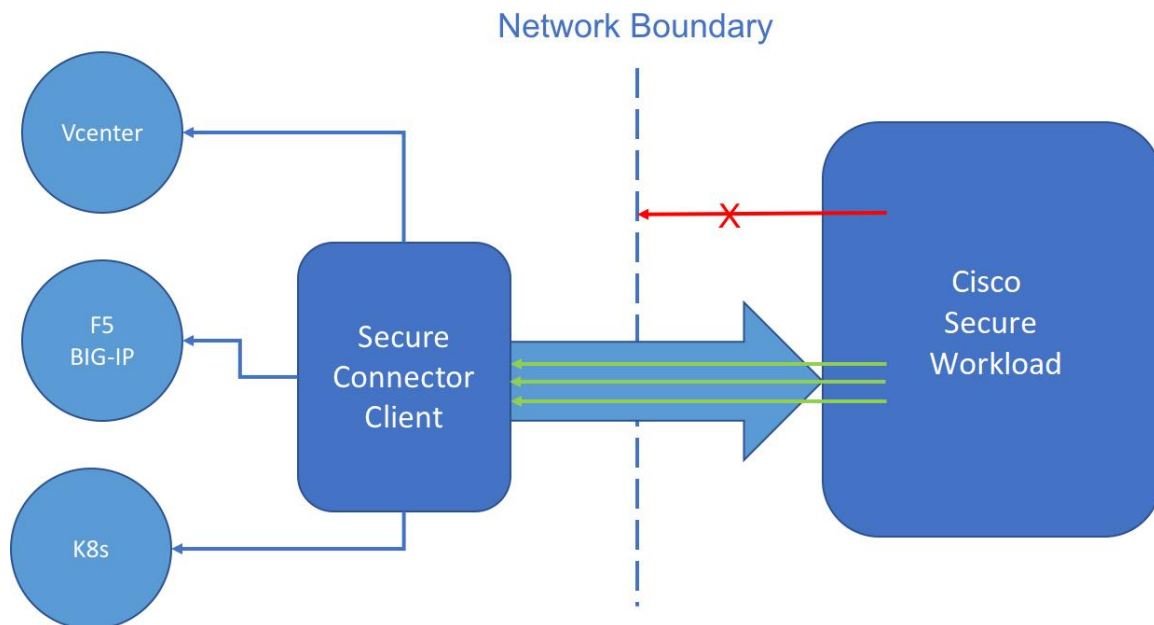
Secure Workload adds the following labels to all the AWS instances.

| Key | Value |
|----------------------------------|---|
| orchestrator_system/orch_type | aws |
| orchestrator_system/cluster_name | <Name of kubernetes cluster> |
| orchestrator_system/name | <Name of the connector> |
| orchestrator_system/cluster_id | <UUID of the orchestrator's configuration in /product/> |

Secure Connector

In order for Secure Workload to import user tags or enforce policies on external orchestrators (see [External Orchestrators in Secure Workload](#)), Secure Workload needs to establish outgoing connections to the orchestrator API servers (vCenter, Kubernetes, F5 BIG-IP, etc.). Sometimes it is not possible to allow direct incoming connections to the orchestrators from the Secure Workload cluster. Secure Connector solves this issue by establishing an outgoing connection from the same network as the orchestrator to the Secure Workload cluster. This connection is used as a reverse tunnel to pass requests from the cluster back to the orchestrator API server.

Figure 36: Secure Connector



For each root scope, only one tunnel may be active at any time. Attempts to start additional tunnels will be rejected with an error message indicating that one is already active. The active tunnel can be used to connect to multiple orchestrators that are reachable from the network in which the client is running. A per-orchestrator configuration is used to indicate whether connections to that orchestrator should go through the Secure Connector tunnel.

All communication between the Secure Connector client and the Secure Workload cluster is mutually authenticated and encrypted using TLS.

For improved security, customers are advised to install the Secure Connector client on an isolated appropriately secured machine. The machine should have firewall rules to allow outgoing connections only to the Secure Workload cluster and any external orchestrator API servers Secure Workload should be allowed to access.

To configure orchestrators to use the Secure Connector tunnel, see instructions for configuring the external orchestrator for your product.

For more details on OpenAPI endpoints for the Secure connector, see [Secure Connector API endpoints](#)

Technical Details

To bootstrap the tunnel, the Secure Connector client creates a public or private key pair and signs its public key certificate remotely by the server. A cryptographic single-use time-limited token is used to secure this remote signing process and identify the root scope to which the client belongs. On the server side, each root scope has a unique certificate that the client uses to authenticate the server. These certificates are periodically rotated to ensure the continued secrecy of communication.

The Secure Connector client is internally constructed of a tunnel client and a SOCKS5 server. After the tunnel is started, the client waits for incoming tunnelling connections from the Secure Workload Cluster. Incoming connections are handled by the SOCKS5 server and forwarded to the destination host.

Requirements for Secure Connector Client

The following are the requirements for the Secure Connector client:

- RHEL or CentOS 7 (x86_64)
- 2 CPU cores
- 4 GB RAM
- Sufficient network bandwidth for handling data from the on-premises orchestrators that use the Secure Connector.
- Outgoing connectivity to the Secure Workload cluster on port 443 (direct or through HTTP(S) proxy).
- Outgoing connectivity to internal Orchestrator API servers (direct).

Secure Connector Client Deployment

Proxy Support

The Secure Connector client supports connecting to the Secure Workload cluster through an HTTP(S) proxy. If needed, the proxy server must be configured by setting the `HTTPS_PROXY` environment variable for the client. To set the variable, add the following line in the `[Service]` section of the systemd service file located at `/etc/systemd/system/tetration-secure-connector.service`. This setting will not persist across re-installations. For a sticky configuration, the line can be added in a new file at `/etc/systemd/system/tetration-secure-connector.service.d/10-https-proxy.conf`. For either configurations to take effect, reload the systemd config by running `systemctl daemon-reload`.

```
[Service]
Environment="HTTPS_PROXY=<Proxy Server Address>"
```

Deployment Overview

The Secure Connector creates a reverse tunnel from the Secure Workload cluster to your internal network in order to reach your orchestrator API servers.

Starting the Secure Connector client requires downloading Secure Connector RPM and generating a one-time registration token.

1. [Download Latest Secure Connector Client RPM](#) on a supported platform.
2. [Generate Registration Token](#).
3. [Copy the Token and Start the Client](#) on the host to start the client.

Deploy the Secure Connector Client

Download Latest Secure Connector Client RPM

Procedure

-
- Step 1** In the navigation pane, click **Manage > Workloads > Secure Connector**.

- Step 2** Click **Download Latest RPM**.
- Step 3** Copy the RPM package to the Linux host for deployment, and then execute the following command with root privileges: `rpm -ivh <rpm_filename>`

Generate Registration Token

Procedure

- Step 1** Click **Manage > Workloads > Secure Connector**.
- Step 2** Click **Generate Registration Token**.

Copy the Token and Start the Client

After generating a registration token on the **Secure Connector** page, you will have a *registration.token* file that contains the single-use limited-time token for bootstrapping the client. Stop the Secure Connector client on the host and copy the token file where you have installed the Secure Connector client package.

1. To stop the client, run the following command: `systemctl stop tetratation-secure-connector`
2. Copy the *registration.token* file to the `/etc/tetration/cert/` folder.
3. To restart the client, run the following command: `systemctl start tetratation-secure-connector`

[Optional] Deploy Specific Version of Secure Connector Client

Procedure

- Step 1** Download a specific version of Secure Connector Client RPM.
- a) In the navigation pane, click **Manage > Workloads > Agents**.
 - b) Click the **Installer** tab.
 - c) Click **Manual Install using classic packaged installers**, then click **Next**.
The Secure Connector Client packages have the agent type as *Secure Connector*.
 - d) Find the appropriate version (if multiple are available on the cluster) and click **Download**.
 - e) Copy the RPM package to the Linux host for deployment, and then execute the following command with root privileges: `rpm -ivh <rpm_filename>`.

- Step 2** Retrieve a new token using the API.

Secure Connector tokens can also be retrieved through OpenAPI ([Get Token](#)). The following Python and Bash snippets can be used to retrieve a new token. Note that the API key used must have the *external_integration* capability and must have write access to the specified root scope. See [OpenAPI Authentication](#) for information on installing the Secure Workload OpenAPI client for python and creating a new API key.

- **Python snippet for token retrieval**

```

from tetpyclient import RestClient
from urllib import quote

API_ENDPOINT = "https://<UI_VIP_OR_DNS_FOR_TETRATION_DASHBOARD>"
ROOT_SCOPE_NAME = r""""<ROOT_SCOPE_NAME>""""
API_CREDENTIALS_FILE = "<API_CREDENTIALS_JSON_FILE>"
OUTPUT_TOKEN_FILE = "registration.token"

if __name__ == "__main__":
    client = RestClient(API_ENDPOINT,
                       credentials_file=API_CREDENTIALS_FILE) # Add (verify=False) to
skip certificate verification
    escaped_root_scope_name = quote(ROOT_SCOPE_NAME, safe='')
    resp = client.get('/secureconnector/name/{}/token'.format(escaped_root_scope_name))
    if resp.status_code != 200:
        print 'Error ({}): {}'.format(resp.status_code, resp.content)
        exit(1)
    else:
        with open(OUTPUT_TOKEN_FILE, 'w') as f:
            f.write(resp.content)

```

• BASH snippet for token retrieval

```

#!/bin/bash
HOST="https://<UI_VIP_OR_DNS_FOR_TETRATION_DASHBOARD>"
API_KEY="<API_KEY>"
API_SECRET="<API_SECRET>"
ROOTSCOPE_NAME="<ROOT_SCOPE_NAME>" # if the name contains spaces or special characters,
it should be url-encoded
TOKEN_FILE="registration.token"
INSECURE=1 # Set to 0 if you want curl to verify the identity of the cluster

METHOD="GET"
URI="/openapi/v1/secureconnector/name/${ROOTSCOPE_NAME}/token"
CHK_SUM=""
CONTENT_TYPE=""
TS=$(date -u "+%Y-%m-%dT%H:%M:%S+0000")
CURL_ARGS="-v"
if [ $INSECURE -eq 1 ]; then
    CURL_ARGS=$CURL_ARGS "-k"
fi

MSG=$(echo -n -e "$METHOD\n$URI\n$CHK_SUM\n$CONTENT_TYPE\n$TS\n")
SIG=$(echo "$MSG" | openssl dgst -sha256 -hmac $API_SECRET -binary | openssl enc -base64)

REQ=$(echo -n "curl $CURL_ARGS $HOST$URI -w '%{http_code}' -H 'Timestamp: $TS' -H 'Id:
$API_KEY' -H 'Authorization: $SIG' -o $TOKEN_FILE")
status_code=$(sh -c "$REQ")
if [ $status_code -ne 200 ]; then
    echo "Failed to get token. Status: " $status_code
else
    echo "Token retrieved successfully"
fi

```

Step 3 Copy the token and start the client. For detailed instructions, see [Copy the Token and Start the Client, on page 133](#).

Verify Secure Connector Client State

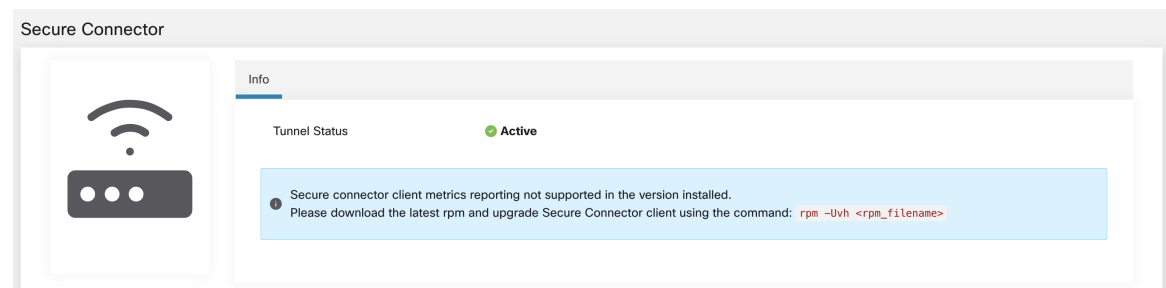
- To check if the Secure Connector client is installed, query the RPM database for the *tet-secureconnector-client-site* package by running the following command: `rpm -q tet-secureconnector-client-site`
- To check the state of the installed client, you can check the status of the *tetration-secure-connector* systemd service by running the following command: `systemctl status tetration-secure-connector`

Secure Connector Client Status

On the **External Orchestrators** page, the status of the configured external orchestrators and Secure Connector tunnel is displayed. If Secure Connector is enabled while configuring the external orchestrators, you can view the Secure Connector client metrics on the **Secure Connector** page.

However, if the Secure Connector tunnel status is **Active** but the client metrics are not visible, it implies an older version of Secure Connector is installed. A message to upgrade the Secure Connector Client version is displayed as shown:

Figure 37: Secure Connector Client upgrade message



Note For instructions on installing the latest Secure Connector RPM, see [Download Latest Secure Connector Client RPM](#)

To view the client metrics:

Procedure

- Step 1** Under **Configure Details**, click the **Status** row. The **Secure Connector** page is displayed.
- Note** To access the status of the Secure Connector tunnel, select **Manage > Workloads > Secure Connector** in the left pane.
- Step 2** Select the tabs - **General**, **Interface**, or **Routes** to access more details on the connectivity status between the client and Secure Workload cluster.

| Tabs | Description |
|------------------|---|
| General | Lists the following information: <ul style="list-style-type: none"> • Tunnel Status • Hostname • IP Address • HTTP/HTTPS Proxy • Version- Lists the build version. • No. vCPU's • Total Memory (GB) • Uptime- Lists the uptime of the VM where the Secure Connector client is running. • Last Heartbeat Received- Lists the day and timestamp of heartbeat last received from client. • No. of Heartbeat Failures (Last 1 day)- Lists the number of times the connectivity to the Secure Connector client failed in a day. If the client remains to stay inactive, the count is not incremented. The count is reset at the end of day. • Round Trip Latency (ms) |
| Interface | Lists the interface details of the VM where the Secure Connector client is running. |
| Routes | The route table lists the destination IP addresses, gateway, genmask, and interface. |

Secure Connector Alerts

The alert is generated when Secure Connector stops functioning or if there is no heartbeat in the past one minute.

Step 1: To enable alert, click **Manage > Workloads > Secure Connector**.

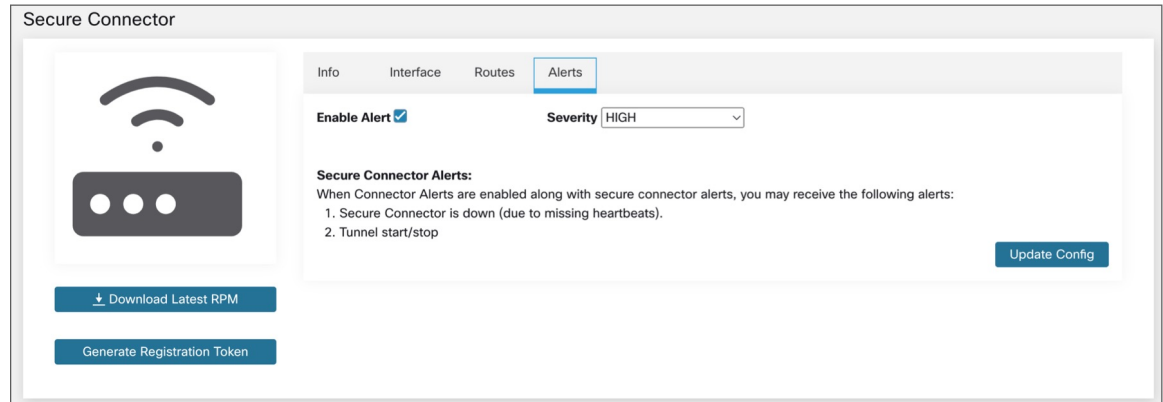
Step 2: Click the **Alerts** tab.

Step 3: Check the **Enable Alert** checkbox.

Step 4: Choose a **Severity** value from the drop-down.

Step 5: Click **Update Config**.

Figure 38: Enable Secure Connector Alerts



Note Ensure that the Connectors alerts is enabled in the **Manage > Alerts - Configuration** page.

Navigate to **Investigate > Alerts** and click on an alert to view more details.

Alert text: Secure Connector: <reason for connection failure>

Figure 39: Secure Connector Alert

| event time ↑↓ | Status ↑↓ | alert text ↑↓ | severity ↑↓ | type ↑↓ | actions ↑↓ |
|---|-----------|---|-------------|-----------|------------|
| 6:26 AM | ACTIVE | Secure Connector: No heartbeat in last 1 minute | HIGH | CONNECTOR | |
| Details | | | | | |
| <p>Name Secure Connector</p> <p>Type Secure Connector</p> <p>Last Checkin At Jun 26 2023 00:55:11 UTC</p> <p>Hostname <code>hamesha-ctrl005</code></p> <p>Total Memory (GB) 31.26</p> <p>No. vCPU's 8</p> <p>VM IPs 127.0.0.1, 172.29.203.37, 172.17.0.1</p> | | | | | |

Table 11: Alert Details

| Field | Type | Description |
|--------------------------|--------|---|
| Name | String | Secure Connector name |
| Type | String | Secure Connector type |
| Last Checkin At | String | Last known time when there was a heartbeat |
| Hostname | String | Name of the machine hosting this Secure Connector |
| Total Memory (GB) | String | RAM in GB |

| Field | Type | Description |
|------------|--------|--|
| No. vCPU's | String | Number of CPUs |
| VM IPs | String | List of network interfaces on the Secure Connector client host |

Upgrade Secure Connector Client

The Secure Connector client does not support automatic updates. To deploy a new version:

1. Run the following command to uninstall the current version: `rpm -e tet-secureconnector-client-site`
2. Deploy the new version. For detailed instructions, see [Deploy the Secure Connector Client, on page 132](#).

Uninstall Secure Connector Client

The Secure Connector Client can be uninstalled using the following command: `rpm -e tet-secureconnector-client-site`

Amazon Web Services



Note AWS external orchestrator functionality is now part of the new AWS cloud connector feature. If you upgraded to this release, your existing AWS external orchestrators are now read-only; if you need to make changes, create a new AWS connector. For complete information, see [AWS Connector](#).

Secure Workload supports automated ingestion of inventory live from an AWS region. When an external orchestrator configuration is added for type “aws”, the Secure Workload appliance will connect to the AWS endpoint and fetch the metadata for all the instances in running/stopped state.

Prerequisites

- Security tokens (access key and secret key) used should have the right kind of IAM privileges to allow fetching of orchestrator information.

Configuration fields

| Attribute | Description |
|-----------|---|
| ID | Unique identifier for the orchestrator. |
| Name | User-specified name of the orchestrator. |
| Type | Type of orchestrator - (<i>aws</i> in this case) |

| Attribute | Description |
|-------------------------|--|
| Description | A brief description of the orchestrator. |
| AWS Access Key ID | ACCESS KEY associated with the account for which orchestrator config is being created. |
| AWS Secret Access key | SECRET KEY associated with the account you create for the orchestrator configuration. Note Re-enter the SECRET KEY if you modify the orchestrator configuration. |
| AWS Region | The Region in which workload has been deployed. If a workload is spread across multiple regions, a separate config is required for every region. See the link below for correct <i>region</i> values. :ref: https://docs.aws.amazon.com/general/latest/gr/rande.html . |
| Accept Self-signed Cert | Is automatically marked true for AWS. User cannot edit it. |
| Full Snapshot Interval | Full snapshot interval in seconds. Orchestrator Inventory manager will perform a full refresh poll from the orchestrator. |
| Delta Snapshot Interval | Delta snapshot interval in seconds. Orchestrator Inventory manager will only fetch incremental updates from the orchestrator. |
| Hosts List | AWS orchestrator type doesn't require hosts list. The endpoint for AWS will be derived from <i>AWS Region</i> field above. This field should be left empty. |
| Verbose TSDB metrics | If enabled, tsdb metrics for each individual orchestrator will be reported. Else an aggregation of all orchestrator metrics will be reported. |
| Secure Connector Tunnel | Tunnel connections to this orchestrator's hosts through the Secure Connector tunnel. |

Workflow

- Configure an AWS orchestrator filled with the configuration fields above.

Orchestrator generated labels

Secure Workload adds the following labels to all the AWS instances.

| Key | Value |
|----------------------------------|---|
| orchestrator_system/orch_type | aws |
| orchestrator_system/cluster_name | <Name of kubernetes cluster> |
| orchestrator_system/name | <Name of the connector> |
| orchestrator_system/cluster_id | <UUID of the orchestrator's configuration in product > |

Instance-specific labels

The following labels are instance specific.

| Key | Value |
|-----------------------------------|---|
| orchestrator_system/workload_type | vm |
| orchestrator_system/machine_id | <InstanceID assigned by AWS> |
| orchestrator_system/machine_name | <PublicDNS(FQDN) given to this node by AWS> |
| orchestrator_<AWS Tag Key> | <AWS Tag Value> |

Troubleshooting

- Confusion between AWS Region and Availability Zone.

Both these values are interrelated and should not be confused. For example us-west-1 might be the region and availability zone can be either of us-west-1a or us-west-1b etc. While configuring orchestrator, *Region* should be used. Refer to <https://docs.aws.amazon.com/general/latest/gr/rande.html> for all regions.

- Connectivity/Credentials issue after updating the orchestrator config.

Customers must re-submit the *AWS Secret Key* every time the config gets updated.

Kubernetes/OpenShift



Note EKS and AKS external orchestrator functionalities are now part of the new AWS and Azure cloud connector features, respectively. If you upgraded to this release, your existing EKS and AKS external orchestrators are now read-only; if you need to make changes, create a new AWS or Azure connector. For complete information, see the relevant topics under [Cloud Connectors](#).

The external orchestrator for plain-vanilla Kubernetes and OpenShift has not changed.

Secure Workload supports automated ingestion of inventory live from a Kubernetes cluster. When an external orchestrator configuration is added for a Kubernetes/OpenShift cluster, Secure Workload connects to the cluster's API server and tracks the status of nodes, pods and services in that cluster. For each object type,

Secure Workload imports all Kubernetes labels and labels associated with the object. All values are imported as-is.

In addition to importing the labels defined for Kubernetes/OpenShift objects, Secure Workload also generates labels that facilitate the use of these objects in inventory filters. These additional labels are especially useful in defining scopes and policies.

For more information about all of these labels, see [Labels Related to Kubernetes Clusters](#).

If enforcement is enabled on the Kubernetes nodes (enforcement agents are installed and the configuration profile enables enforcement on these agents), enforcement policies will be installed in both the nodes as well as inside the pod namespaces using the information ingested about the Kubernetes entities via this integration.

About Kubernetes on Cloud Platforms

For the following managed kubernetes services running on supported cloud platforms, this orchestrator's functionality is provided using cloud connectors:

- Elastic Kubernetes Service (EKS) running on Amazon Web Services (AWS)
- Azure Kubernetes Service (AKS) running on Microsoft Azure
- Google Kubernetes Engine (GKE) running on Google Cloud Platform (GCP)

For details about obtaining data from kubernetes clusters on cloud platforms, see the topics under [Cloud Connectors](#).

Requirements and Prerequisites

- For supported Kubernetes and OpenShift versions, see <https://www.cisco.com/go/secure-workload/requirements/integrations>
- Secure Connector tunnel, if needed for connectivity.

Configuration Fields

The following configuration fields pertain to Kubernetes Orchestrator configuration in the Orchestrator Object.

| Field | Description |
|------------------------|---|
| Name | User specified name of the orchestrator. |
| Description | User specified description of the orchestrator. |
| Delta Interval | Interval (in seconds) to check the Kubernetes endpoint for changes |
| Full Snapshot Interval | Interval (in seconds) to perform a full snapshot of Kubernetes data |
| Username | Username for the orchestration endpoint. |
| Password | Password for the orchestration endpoint. |
| Certificate | Client certificate used for authentication. |

| Field | Description |
|-------------------------|--|
| Key | Key corresponding to client certificate. |
| Auth Token | Opaque authentication token (bearer token). |
| CA Certificate | CA Certificate to validate orchestration endpoint. |
| Accept Self-Signed Cert | Checkbox to disable strictSSL checking of the Kubernetes API server certificate |
| Verbose TSDB Metrics | Maintain per Kubernetesorchestrator metrics - if set to False, only Secure Workload clusterwide metrics are maintained. |
| Secure connector Tunnel | Tunnel connections to this orchestrator's hosts through the Secure Connector tunnel |
| Hosts List | Array of { "host_name", port_number} pairs that specify how Secure Workload must connect to the orchestrator |
| K8s manager type | Manager type for the kubernetes cluster(None for Vanilla/Openshift kubernetes deployments) |
| AWS cluster name | Name of the orchestrator as specified at time of creation of cluster (Pre-existing EKS) |
| AWS Access ID | ACCESS KEY associated with the account for which orchestrator config is being created (Pre-existing EKS) |
| AWS Secret Access Key | The SECRET KEY associated with the account the orchestrator configuration is created. Re-enter the SECRET KEY every time the config is edited. (Pre-existing EKS) |
| AWS Region | The Region in which workload has been deployed. If a workload is spread across multiple regions, a separate config is required for every region. See the link below for correct <i>region</i> values. .ref: https://docs.aws.amazon.com/general/latest/gr/rande.html . (Pre-existing EKS) |
| AWS Assume Role ARN | Amazon resource number of the roles to assume while connecting to the orchestrator ref: https://docs.aws.amazon.com/STS/latest/APIReference/API_AssumeRole.html (Pre-existing EKS) |
| Azure Tenant ID | Tenant ID associated with Azure subscription. (Pre-existing AKS only) |

| Field | Description |
|---------------------|---|
| Azure Client ID | Globally unique ID associated with the application that needs to authenticate with Azure AD. (Pre-existing AKS only) |
| Azure Client Secret | Password associated with the service principal for the application that needs to authenticate with Azure AD. (Pre-existing AKS only) |

Orchestrator Golden Rules

The golden rules object attributes are described below. These golden rules allow a concise specification of rules necessary for the Kubernetes cluster to stay functional once enforcement is enabled on the Kubernetes cluster nodes.

| Attribute | Description |
|--------------|--------------------------------------|
| Kubelet Port | Kubelet node-local API port |
| Services | Array of Kubernetes Services objects |

The kubelet port is necessary to create policies to allow traffic from the Kubernetes management daemons to kubelets such as for live logs, execs of pods in interactive mode etc. Vital connectivity between the various kubernetes services and daemons is specified as a series of services - each entry in the services array has the following structure

- Description: A string that describes the service
- Addresses: A list of service endpoint addresses of the format <IP>:<port>/<protocol>.
- Consumed By: A list of consumers of the endpoints (allowed values are Pods or Nodes)



Note If **kubernetes** is chosen as the type, Golden Rules configuration will be allowed.

Figure 40: Create Golden Rules Configuration for Kubernetes Type

Create External Orchestrator Configuration

Save changes to configure Golden Rules?

Basic Config

Type
Kubernetes

Hosts List

Golden Rules

K8s Manager Type
(None)

Name
Name

Description
Description of the orchestrator

Delta Interval (s)
60

Full Snapshot Interval (s)
3600

Connection will be tested after the creation.

Workflow

- Configure Secure Connector tunnel, if needed, for connectivity from the Secure Workload cluster to a Kubernetes API server (or servers).
- Configure a Kubernetes orchestrator filled with the configuration fields above.
- Configure the Golden Rules for the Kubernetes orchestrator.

Kubernetes Role-Based Access Control (RBAC) Resource Considerations

The Kubernetes client attempts to GET/LIST/WATCH the following resources. It is highly recommended NOT to configure the admin key/cert or an admin service account.

The provided Kubernetes authentication credentials should have a minimum set of privileges to the following resources:

| Resources | Kubernetes Verbs |
|------------------------|------------------|
| endpoints | [get list watch] |
| namespaces | [get list watch] |
| nodes | [get list watch] |
| Pods | [get list watch] |
| services | [get list watch] |
| ingresses | [get list watch] |
| replicationcontrollers | [get list watch] |
| replicasets | [get list watch] |
| deployments | [get list watch] |
| daemonsets | [get list watch] |
| statefulsets | [get list watch] |
| jobs | [get list watch] |
| cronjobs | [get list watch] |

Essentially, you can create a special service account on your Kubernetes server with these minimal privileges. An example sequence of kubectl commands is below that will facilitate the creation of this serviceaccount. Note the use of the clusterrole (not role) and clusterrolebindings (not rolebindings) - these are cluster-wide roles and not per namespace. Using a role/rolebinding will not work as Secure Workload attempts to retrieve data from all namespaces.

```
$ kubectl create serviceaccount csw.read.only
```

Create the clusterrole.

A sample clusterrole.yaml with minimal privileges is provided below

```
kind: ClusterRole
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: csw.read.only
rules:
  - apiGroups:
    - ""
    resources:
      - nodes
      - services
      - endpoints
      - namespaces
      - pods
      - replicationcontrollers
      - ingresses
    verbs:
      - get
      - list
```

```

- watch
- apiGroups:
- extensions
- networking.k8s.io
resources:
- ingresses
verbs:
- get
- list
- watch
- apiGroups:
- apps
resources:
- replicasets
- deployments
- statefulsets
- daemonsets
verbs:
- get
- list
- watch
- apiGroups:
- batch
resources:
- jobs
- cronjobs
verbs:
- get
- list
- watch

$ kubectl create -f clusterrole.yaml

```



Note API groups for these different resources are susceptible to change across Kubernetes versions. The sample above should work for Kubernetes versions 1.20-1.24 and might require some tweaks for other versions.

Create the cluster role binding

```
$ kubectl create clusterrolebinding csw.read.only --clusterrole=csw.read.
.>only --serviceaccount=default:csw.read.only
```

To retrieve the authtoken secret from the serviceaccount (used in the Auth Token field in the GUI) and decode from base64, you can retrieve the name of the secret by listing the serviceaccount with yaml output.

```
$ kubectl get serviceaccount -o yaml csw.read.only
apiVersion: v1
kind: ServiceAccount
metadata:
  creationTimestamp: 2020-xx-xxT19:59:57Z
  name: csw.read.only
  namespace: default
  resourceVersion: "991"
  selfLink: /api/v1/namespaces/default/serviceaccounts/e2e.minimal
  uid: ce23da52-a11d-11ea-a990-525400d58002
secrets:
- name: csw.read.only-token-vmvmz

```

Listing the secret in yaml output mode will yield the token but in Base64 format (which is standard Kubernetes procedure for secret data). Secure Workload does not accept the token in this format, you must decode it from Base64.

```
$ kubectl get secret -o yaml csw.read.only-token-vmvmz
apiVersion: v1
data:
  ca.crt: ...
  namespace: ZGVmYXVsdA==
  token: ZXlKaGJHY2lPaUpTVX...HRfZ2JwMVZR
kind: Secret
metadata:
  annotations:
    kubernetes.io/service-account.name: csw.read.only
    kubernetes.io/service-account.uid: ce23da52-a11d-11ea-a990-525400d58002
  creationTimestamp: 2020-05-28T19:59:57Z
  name: csw.read.only-token-vmvmz
  namespace: default
  resourceVersion: "990"
  selfLink: /api/v1/namespaces/default/secrets/csw.read.only-token-vmvmz
  uid: ce24f40c-a11d-11ea-a990-525400d58002
type: kubernetes.io/service-account-token
```

To list the secret and output only the `.data.token` field and decode from base 64 encoding in one command, the following command that use the `--template` option is helpful.

```
$ kubectl get secret csw.read.only-token-vmvmz --template "{{ .data.token }}" | base64 -d
```

This authtoken can be used for configuring a Kubernetes orchestrator in the Secure Workload UI instead of username/password or key/cert.

See EKS specific RBAC considerations.

Orchestrator-generated labels

See [Labels Related to Kubernetes Clusters](#).

Troubleshooting

- Client key or certificate Credentials parsing or mismatch

These must be supplied in PEM format and be the correct entry from the `kubectl.conf` file. We have encountered customers pasting CA certs into client cert fields, as well as keys and certs not matching each other.

- Gcloud credentials instead of GKE credentials

Customers using GKE under the `gcloud` CLI mistakenly provide the `gcloud` credentials when the GKE cluster credentials are needed.

- Kubernetes cluster version unsupported

Using an incompatible version of Kubernetes may result in failures. Verify that the Kubernetes version is in the supported versions list.

- Credentials have insufficient privileges

verify that the authtoken or user or client key or cert used has all the privileges listed in the table above.

- Kubernetes inventory keeps flipping around

The `hosts_list` field specifies a pool of API servers for the same Kubernetes cluster - you cannot use this to configure multiple Kubernetes clusters. Secure Workload will probe for aliveness and randomly select one of these endpoints to connect to and retrieve the Kubernetes inventory information. No load balancing is performed here, nor is there a guarantee of evenly distributing load across these endpoints. If these are different clusters, the Kubernetes inventory will keep flipping between them, depending on which cluster's API server we connect to.

- Multiple authorization methods

Multiple authorization methods may be filled in during configuration (username or password, authtoken, client key or certificate) and will be used in the client connection established with the API server. The standard Kubernetes rules for valid simultaneous authorization methods apply here.

- SSL Certificate validation fails

If the Kubernetes API endpoint is behind a NAT or load balancer, then the DN in the SSL certificate generated on the kube control plane nodes may mismatch with the IP address configured in Secure Workload. This will cause an SSL validation failure even if the CA certificate is provided and is valid. The `Insecure` knob bypasses strict server SSL certificate validation and will help workaroud this issue but can lead to MITM issues. The correct fix for this is to change the CA certificate to provide SAN (Subject Alternative Name) entries for all DNS or IP entries that can be used to connect to the Kubernetes cluster.

VMware vCenter

vCenter integration allows user to fetch bare metal and VM attributes from configured vCenter.

When an external orchestrator configuration is added for type “vCenter”, Secure Workload fetches bare metal and VM attributes for all the bare metals and VM's controlled by that vCenter instance. Secure Workload will import the following attributes of a bare metal/VM:- a) Hostname b) IP addresses c) BIOS UUID d) Categories/Labels.

A new inventory will be created in Secure Workload with the above bare metal/VM attributes, if the inventory is not present in the appliance. If the inventory is already present in the appliance (created by Secure Workload visibility sensor running on the bare metal/VM), the existing inventory will be labelled with the fetched bare metal/VM Categories/Labels list.

Prerequisites

- Secure Connector Tunnel, if needed for connectivity.
- vCenter version supported is 6.5+

Configuration fields

Beside the common configuration fields as described in *Create External Orchestrator* the following fields can be configured:

- **Hosts List** is an array of hostname/ip and port pairs pointing to the vCenter server from which bare metal/VM attributes will be fetched.

Workflow

- First, the user must verify that the vCenter server is reachable on that IP/Port from the Secure Workload cluster.
- For TaaS or in cases where the vCenter server is not directly reachable, the user must configure a secure connector tunnel to provide connectivity.

Orchestrator generated labels

Secure Workload adds the following labels to all the VM's learnt from vCenter server.

| Key | Value |
|----------------------------------|--|
| orchestrator_system/orch_type | vCenter |
| orchestrator_system/cluster_name | <Name given to this cluster's configuration> |
| orchestrator_system/cluster_id | <UUID of the cluster's configuration in product > |

Instance-specific labels

The following labels are instance specific.

Table 12: The following labels are instance specific.

| Key | Value |
|-----------------------------------|--------------------------------------|
| orchestrator_system/workload_type | vm |
| orchestrator_system/machine_id | <i>BIOS UUID of bare metal/VM</i> |
| orchestrator_system/machine_name | <i>Hostname of the bare metal/VM</i> |
| orchestrator_ '<Category Name>' | <Tag Value> |

Caveats

- When an external orchestrator configuration is added for vCenter, Secure Workload software will connect to the vCenter server specified in the hosts list. After the connection to the server is successful, Secure Workload software will import hostnames, IP addresses and Category/Labels for all the bare metals and Virtual Machines present in the vCenter server. In order to import hostnames and IP addresses of the bare metals and VM's, VM tools must be installed on all the bare metals and VM's. If VM tools is not installed for a given bare metal/Virtual Machine, Secure Workload software will not display Category/Labels for that particular bare metal/VM.
- Secure Workload software doesn't import Custom attributes of the bare metal/VM.
- It is recommended to set **Delta** interval timer to more than 10 min so as to reduce the load on the vCenter server. Any change in the inventory/labels on the vCenter server will have a propagation delay of at least 10 min, once the above mentioned timer is modified.

Troubleshooting

- Connectivity Issues

In case, Secure Workload appliance is not able to connect/reach the vCenter server, **Connection Status** tab of the External orchestrator will display the failure status along with the appropriate error if any.

- Secure Workload software health check.

Check the **MAINTENANCE/Service Status** page to see if any service is down. Check if **OrchestratorInventoryManager** is up and running.

DNS

The DNS Integration allows Secure Workload to annotate known inventory with DNS information such as hostnames from CNAME and A/AAAA records.

When an external orchestrator configuration is added for type “dns”, the Secure Workload appliance will attempt to connect to the DNS server(s) and perform a zone transfer download of DNS records. These records (only A/AAAA and CNAME records) will be parsed and used to enrich inventory in the Secure Workload pipelines (as belonging to the Tenant under which the orchestrator is configured) with a single multi-value label called “orchestrator_system/dns_name”, whose value will be the DNS entries that point (directly or indirectly) to that IP address.

Prerequisites

- Secure Connector Tunnel, if needed for connectivity
- Supported DNS Servers: BIND9, servers supporting AXFR (RFC 5936), Microsoft Windows Server 2016

Configuration fields

- **DNS zones** is an array of strings, each of which represents a DNS zone to be transferred from the DNS server. All dns zones must have a trailing period (“.”) character.
- **Hosts List** is an array of hostname/ip and port pairs pointing to the DNS server(s) from which to fetch the DNS records. Multiple DNS servers may be configured here for HA purposes only. High Availability behavior across multiple DNS servers specified in the hosts_list is “first healthy server” and will favor the earlier entries in the hosts_list. Zones cannot be split across the DNS servers.

Workflow

- First, the user must verify that the DNS server is reachable on that IP/Port from the Secure Workload cluster.
- For TaaS or in cases where the DNS server is not directly reachable, the user must configure a secure connector tunnel to provide connectivity.

- Configure the correct DNS Zone Transfers ACLs/configuration on the DNS server. Refer to the documentation for the particular DNS server software for more information.

Generated labels

orchestrator_system/dns_name -> a multi-value field whose values are all the CNAME and A/AAAA hostnames pointing to that IP.

Caveats

- The DNS orchestrator feed is a *metadata feed* - IP addresses learnt from a DNS zone transfer will not create inventory items in Secure Workload, rather, labels for an existing IP address will be updated with the new DNS metadata. DNS data for unknown IPs is silently discarded. In order to annotate DNS metadata to IPs not learnt from any sensor or via any other orchestrator integrations, IPs must be uploaded via the CMDB bulk upload mechanism to create inventory entries for them. Subnets learnt from CMDB uploads do not create inventory entries.
- Only CNAME and A/AAAA records from the DNS server are processed. CNAME records will be processed to their ultimate IPv4/IPv6 records via the A/AAAA records they point to. Only a single level of deferencing is supported (i.e. chains of CNAME -> CNAME -> A/AAAA or longer are not deferenced) as long as the CNAME points to an A/AAAA record from that same orchestrator. CNAME deferencing across different DNS orchestrators is not supported.

Troubleshooting

- Connectivity Issues

Secure Workload will attempt to connect to the provided ip/hostname and port number using a TCP connection originating from one of the Secure Workload appliance servers or from the cloud in the case of TaaS or from the VM hosting the Secure Workload Secure Connector VPN tunnel service. In order to correctly establish this connection, firewalls must be configured to permit this traffic.

- DNS AXFR Privilege Issues

In addition, most DNS servers (BIND9 or Windows DNS or Infoblox) require additional configuration when client IPs attempt DNS zone transfers (AXFR requests as per the DNS protocol opcodes) as these are more resource intensive and privileged as compared to simple DNS requests to resolve individual DNS records. These errors typically show up as AXFR refused with reason code 5 (REFUSED).

Thus, any manual testing to establish that the DNS server is configured correctly must not depend on successful hostname lookups but rather they must test AXFR requests specifically (using a tool such as dig).

Any failure to perform an AXFR zone transfer from the DNS server will be reported in the "authentication_failure_error" field by Secure Workload appliance.

Also, note that Secure Workload will attempt zone transfers from all configured DNS zones and all must succeed in order for the DNS data to be injected into the Secure Workload label database.

- Inventory Hostname fields are not populated by DNS Field 'hostname' is always learnt from the Secure Workload sensor. If the inventory was uploaded via CMDB upload and not from the sensor, it may be

missing the hostname. All data from the DNS orchestrator workflow only shows up under the “orchestrator_system/dns_name” label and will never populate the hostname field.

Behavior of Full/Delta polling for DNS Orchestrators

Default Full Snapshot Interval is 24 hours

Default Delta Snapshot Interval is 60 minutes

These are also the minimum allowed values for these timers.

DNS Records may rarely change. So, for optimal fetching behaviour, at every delta snapshot interval, Secure Workload will check if the serial numbers of any of the DNS zones has changed from the previous interval. If no zones have changed, no action is needed.

If any zones have changed, we will perform a zone transfer from all configured DNS zones (not just the single zone that has changed).

Every full snapshot interval, Secure Workload will perform zone transfer downloads from all zones and inject into the label database regardless of whether the zone serial numbers have changed.

Unsupported Features



Warning

- DNAME aliasing and lookups are not supported.
- Incremental Zone Transfers (IXFR) are not supported.

Infoblox

The Infoblox integration allows Secure Workload to import Infoblox subnets, hosts (*record:host*) and A/AAAA records into Secure Workload inventory database. The extensible attribute names and values are imported as is and can be used as Secure Workload labels to define scopes and enforcement policies.



Note

Only Infoblox objects with extensible attributes are considered, ie. those without any extensible attributes attached will be excluded from the import.

Below picture shows an example of generated labels for a host object imported from Infoblox with the extensible attribute *Department*:

- While creating infoblox config, the user has an option to deselect any of the record types(subnet, host, A/AAAA records).

Orchestrator generated labels

Secure Workload adds the following system labels to all objects retrieved from Infoblox.

| Key | Value |
|----------------------------------|---|
| orchestrator_system/orch_type | infoblox |
| orchestrator_system/cluster_id | <UUID of the external orchestrator in Secure Workload |
| orchestrator_system/cluster_name | <Name given to this external orchestrator> |
| orchestrator_system/machine_id | <Infoblox object reference/identifier> |
| orchestrator_system/machine_name | <Infoblox host (DNS) name> |

Generated labels

All Infoblox extensible attributes will be imported as Secure Workload labels with the prefix *orchestrator_*. For instance, a host with an extensible attribute called *Department* can be addressed in Secure Workload inventory search as *orchestrator_Department*.

| Key | Value |
|-------------------------------------|---|
| orchestrator_<extensible attribute> | <value(s) of the extensible attribute as retrieved from Infoblox> |

Caveats

- The maximal number of subnets that can be imported from Infoblox is 50000.
- The maximal number of hosts and A/AAAA records that can be imported from Infoblox is 400000 in total.

Troubleshooting

- Connectivity issue Secure Workload will attempt to connect to the provided IP/hostname and port number using an HTTPS connection originating from one of the Secure Workload appliance servers or from the cloud in the case of TaaS or from the VM hosting the Secure Workload Secure Connector tunnel service. In order to correctly establish this connection, firewalls must be configured to permit this traffic. Also, make sure the given credentials are correct and have privileges to send REST API requests to the Infoblox appliance.

- Not all expected objects are imported Secure Workload imports only subnets, hosts and A/AAAA records with attached extensible attributes. Note there is a limit number objects that can be imported from Infoblox, see *Caveats*.
- Could not find subnets in inventory It is not possible to use inventory search to find Infoblox subnets as Secure Workload inventory by design includes only IP addresses, ie. hosts and A/AAAA records.
- Could not find a host or A/AAAA record Secure Workload imports all extensible attributes as retrieved from Infoblox. Remember to add the prefix *orchestrator_* to the extensible attribute name in eg. inventory search. Note subnets extensible attributes, if not marked as inherited in Infoblox, are not part of hosts and hence not searchable in Secure Workload.

F5 BIG-IP

The F5 BIG-IP integration allows Secure Workload to import the *Virtual Servers* from an F5 BIG-IP load balancer appliance and to derive service inventories. A service inventory corresponds to an F5 BIG-IP virtual server, whose service is characterized by the *VIP* (virtual IP address), protocol and port. Once imported into Secure Workload this service inventory will have labels such as *service_name*, which can be used in inventory search as well as to create Secure Workload scopes and policies.

A big benefit of this feature is the enforcement of policies in that the *external orchestrator for F5 BIG-IP* translates Secure Workload policies to security rules assigned to the virtual server and deploys them to the F5 BIG-IP load balancer via its REST API.

Prerequisites

- Secure Connector Tunnel, if needed for connectivity
- F5 BIG-IP REST API endpoint version 12.1.1

Configuration fields

Beside the common configuration fields as described in *Create External Orchestrator* the following fields can be configured:

| Field | Required | Description |
|------------|----------|--|
| Hosts List | Yes | This specifies the REST API endpoint for F5 BIG-IP load balancer. If High Availability is configured for F5 BIG-IP, enter the standby member node so that in case of a failover, the external orchestrator switches over to the current node. If you want to import labels from another F5 BIG-IP load balancer, you need to create a new external orchestrator. |

| Field | Required | Description |
|--------------------|----------|--|
| Enable Enforcement | No | Default value is false (unchecked). If checked, this allows Secure Workload <i>policy enforcement</i> to deploy security policy rules to the corresponding F5 BIG-IP load balancer. Note the given credentials must have write access for the F5 BIG-IP REST API. |
| Route Domain | No | Default value is 0 (zero). The route domain specifies which virtual server are to be considered by the external orchestrator. This is determined by the list of partitions assigned to the given route domain, and only the virtual servers defined in those partitions will be imported in Secure Workload. |

Workflow

- First, the user must verify that the F5 BIG-IP REST API endpoint is reachable from Secure Workload.
- For TaaS or in cases, where the F5 BIG-IP appliance is not directly reachable, the user must configure a Secure Connector tunnel to provide connectivity.
- Create an external orchestrator with type *F5 BIG-IP*.
- Depending on the *delta interval* value it might take up to 60 seconds (default delta interval) for the first full snapshot of F5 BIG-IP virtual servers to complete. Thereafter the generated labels can be used to create Secure Workload scopes and enforcement policies.

Orchestrator generated labels

Secure Workload adds the following system labels for an external orchestrator for *F5 BIG-IP*:

| Key | Value |
|-----------------------------------|--|
| orchestrator_system/orch_type | f5 |
| orchestrator_system/cluster_id | <UUID of the external orchestrator> |
| orchestrator_system/cluster_name | <Name given to this external orchestrator> |
| orchestrator_system/workload_type | service |
| orchestrator_system/namespace | <Partition the virtual server belongs to> |
| orchestrator_system/service_name | <Name of the F5 BIG-IP virtual server> |

Generated labels

For each virtual server the external orchestrator will generate the following labels:

| Key | Value |
|--------------------------------------|--------------------------------|
| orchestrator_annotation/snat_address | <Virtual servers SNAT address> |

Policy enforcement for F5 BIG-IP

This feature enables Secure Workload to translate logical policies with provider groups that match labelled *F5 BIG-IP* virtual servers into *F5 BIG-IP* security policy rules and deploys them to the load balancer appliance using its REST API. As mentioned above any assignment of existing security policy to the respective *F5 BIG-IP* virtual server will be replaced by a new assignment pointing to Secure Workload generated security policy. Existing security policies will not be changed or removed from the *F5 BIG-IP* policy list.

By default, enforcement is not enabled in the external orchestrator configuration:

Figure 42: Configuration Option "Enable Enforcement"

Create External Orchestrator Configuration

Route Domain

Basic Config

Hosts List

Username

Username for the orchestration workload

Password

Password for the orchestration workload

CA Certificate

CA Certificate to validate orchestration workload

Accept Self-signed Cert

Secure Connector Tunnel

Enable Enforcement

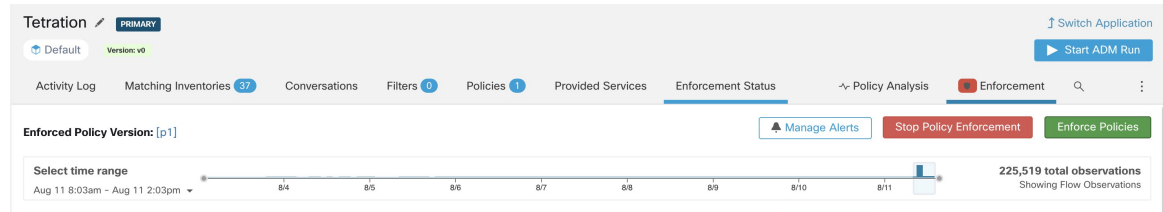
Connection will be tested after the creation.

This option can be modified any time as needed.

Enabling enforcement does not deploy policies to the load balancer appliance unless and until you enable enforcement in a workspace that includes at least one policy that applies to the load balancer, or due to any updates of inventories.

However, disabling enforcement for the orchestrator will cause all deployed security policy rules being removed from the *F5 BIG-IP* load balancer immediately.

Figure 43: Workspace Policy Enforcement



Note

- The orchestrator for *F5 BIG-IP* also detects any deviation of security policy rules and replaces it with Secure Workload policies, ie. any policy changes towards the virtual servers should be done with Secure Workload only.
- When policy enforcement is stopped or the external orchestrator is deleted, the security policy for virtual servers will become empty as all Secure Workload policies will be removed from *F5 BIG-IP* load balancer.

The OpenAPI Policy enforcement status for external orchestrator can be used to retrieve the status of Secure Workload policy enforcement to the load balancer appliance associated with the external orchestrator. This helps to verify if the deployment of security policy rules to the *F5 BIG-IP* appliance has succeeded or failed.

Policy Enforcement for F5 Ingress Controller

Secure Workload enforces policies both at the *F5 BIG-IP* load balancer and at the backend pods when the pods are exposed to the external clients using Kubernetes ingress object.

Following are the steps to enforce the policy using the F5 ingress controller.

Procedure

Step 1 Create an external orchestrator for *F5 BIG-IP* load balancer as described earlier.

Step 2 Create an external orchestrator for Kubernetes/OpenShift as described here.

```

→ ~ k8s get ingress
NAME          HOSTS      ADDRESS          PORTS    AGE
test-ingress  *         192.168.60.100  80      7s

```

Step 3 Create an ingress object in the Kubernetes cluster. A snapshot of the yaml file used to create the ingress object is provided in the following picture.


```

→ ~
→ ~ k8s get ingress test-ingress -o yaml
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  annotations:
    virtual-server.f5.com/ip: 192.168.60.100
    virtual-server.f5.com/partition: k8scluster
  creationTimestamp: "2019-07-26T18:34:39Z"
  generation: 1
  name: test-ingress
  namespace: default
  resourceVersion: "8310"
  selfLink: /apis/extensions/v1beta1/namespaces/default/ingresses/test-ingress
  uid: 06f8a705-afd4-11e9-97fb-525400d58002
spec:
  backend:
    serviceName: nginx
    servicePort: 80
status:
  loadBalancer:
    ingress:
      - ip: 192.168.60.100
→ ~

```

Step 4 Deploy an F5 ingress controller pod in the Kubernetes cluster.

```

→ ~ k8s get deploy -n kube-system
NAME                DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE
coredns             2         2         2             2           31m
k8s-bigip-ctlr-cluster 1         1         1             1           5m20s
→ ~

```

Step 5 Create a backend service, which is accessed by the consumers outside the cluster. In the example provided below we have created a *nginx* service.

```

→ ~
→ ~ k8s get deploy
NAME                DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE
nginx               1         1         1             0           5s
→ ~

```

Step 6 Create a policy between external consumer and backend service. Enforce the policy using *Policy Enforcement* tab.

The screenshot shows the Tetration console interface. The top navigation bar includes 'Activity Log', 'Matching Inventories (46)', 'Conversations', 'Filters (13)', 'Policies (155)', 'Provided Services', and 'Enforcement Status'. The main area displays a table of policies. The selected policy is:

| Priority | Action | Consumer | Provider | Protocols And Ports |
|----------|--------|--------------------------------|----------|---------------------|
| 100 | ALLOW | OTHER: RCDN9-DCIO3N-ACE-Client | Default | TCP: Any |

The right-hand pane shows the 'Policy Actions' details for this policy:

- Priority: 100
- Action: ALLOW
- Consumer: OTHER: RCDN9-DCIO3N-ACE-Client-vl200
- Provider: Default
- Flows: View Conversations
- Protocols and Ports: TCP: Any

Step 7

Check the policies on *F5 BIG-IP* load balancer and backend pods. In case of F5 load balancer Secure Workload will apply the appropriate allow/drop rule where the source will be the consumer specified in step 6 and the destination will be VIP [VIP for the ingress virtual service for F5]. In case of backend pods, Secure Workload will apply the appropriate allow/drop rule where the source will be the SNIP [in case SNAT pool is enabled] or F5 IP [auto map enabled] and destination will be backend pod IP.

The screenshot shows the F5 BIG-IP configuration page for 'Virtual Servers: Virtual Server List > Ingress_192-168-60-100_80'. The 'Policy Settings' section is expanded to show various security options:

- Destination: 192.168.60.100/80
- Service: HTTP
- Application Security Policy: Disabled
- Protocol Security: Disabled
- Network Firewall: Enforcement: Enabled; Policy: Tetration_policy_1_ingress_192-168-60-100_80; Staging: Disabled
- Network Address Translation: Use Device Policy: Disabled; Use Route Domain Policy: None
- Service Policy: None
- IP Intelligence: Disabled
- DoS Protection Profile: Disabled
- Log Profile: Disabled

Below the settings is a 'Rule List' table:

| Name | Rule List | Description | State | Schedule | Source Address/Region | Port | VLAN / Tunnel | Destination Address/Region | Port | Protocol | Rule | Action | Logging | Service Policy |
|---|---|-------------|---------|----------|--|------|---------------|----------------------------|------|----------|--------|--------|----------|----------------|
| Rule_1_fepak9fmgz_ingress_192-168-60-100_80 | Ttp_ruleList_1_fepak9fmgz_ingress_192-168-60-100_80 | | Enabled | | 172.0.21.132 192.168.10.21/32 192.168.60.21/32 | Any | Any | 192.168.60.100/32 | 80 | 6 (TCP) | Drop | Drop | Disabled | |
| (Default) | Rule_CatchAll | | Enabled | | Any | Any | Any | 192.168.60.100 | Any | Any | Accept | Accept | Disabled | |

Caveats

- During deployment phase of *F5 BIG-IP* HA mode, enable the *configuration sync* option. This ensures the external orchestrator can fetch the latest list of virtual servers from the currently connected host.
- In case of *F5 BIG-IP* HA deployment mode, if *Auto-Map* is configured instead of SNAT pool for Address translation, ensure that the *Primary BIG-IP* is configured with the floating *Self IP* address.
- Only VIP specified as a single address is supported, ie. VIP given as a subnet is not supported.

Troubleshooting

- Connectivity issue Secure Workload will attempt to connect to the provided IP/hostname and port number using an HTTPS connection originating from one of the Secure Workload appliance servers or from the cloud in the case of *TaaS* or from the VM hosting the Secure Workload Secure Connector tunnel service. In order to correctly establish this connection, firewalls must be configured to permit this traffic. Also, make sure the given credentials are correct and have privileges with read and write access to send REST API requests to the *F5 BIG-IP* appliance.
- Security rules not found In case no security rules for a defined virtual server are found, after policy enforcement was performed, make sure the corresponding virtual server is enabled, ie. its availability/status must be *available/enabled*.

Citrix Netscaler

The Citrix Netscaler integration allows Secure Workload to import the *Load Balancing Virtual Servers* from a Netscaler load balancer appliance and to derive service inventories. A service inventory corresponds to a Netscaler service provided by a virtual server and has labels such as *service_name*, which can be used in inventory search and to create Secure Workload scopes and policies.

A big benefit of this feature is the enforcement of policies in that the *external orchestrator for Citrix Netscaler* translates Secure Workload policies to Netscaler ACLs rules and deploys them to the Netscaler load balancer via its REST API.

Prerequisites

- Secure Connector Tunnel, if needed for connectivity
- Netscaler REST API endpoint version 12.0.57.19

Configuration fields

Beside the common configuration fields as described in *Create External Orchestrator* the following fields can be configured:

| Common Field | Required | Description |
|--------------------|----------|--|
| Hosts List | Yes | This specifies the REST API endpoint for Citrix Netscaler load balancer. If High Availability is configured on Citrix Netscaler, enter another member node so that in case of a failover, the external orchestrator switches over to the current node. If you want to import labels from another Citrix Netscaler load balancer, create a new external orchestrator. |
| Enable Enforcement | No | Default value is false (unchecked). If checked, this allows Secure Workload <i>policy enforcement</i> to deploy ACL rules to the corresponding Citrix Netscaler load balancer. Note the given credentials must have write access for the Citrix Netscaler REST API. |

Workflow

- First, the user must verify that the Netscaler REST API endpoint is reachable from the Secure Workload cluster.
- For TaaS or in cases, where the Netscaler appliance is not directly reachable, the user must configure a Secure Connector tunnel to provide connectivity.
- Create an external orchestrator with type *Citrix Netscaler*.
- Depending on the *delta interval* value it might take up to 60 seconds (default delta interval) for the first full snapshot of Netscaler virtual servers to complete. Thereafter the generated labels can be used to create Secure Workload scopes and enforcement policies.
- Enforce policies from Secure Workload to deploy Netscaler ACL rules.

Orchestrator generated labels

Secure Workload adds the following system labels for an external orchestrator for *Citrix Netscaler*:

| Key | Value |
|-----------------------------------|--|
| orchestrator_system/orch_type | nsbalancer |
| orchestrator_system/cluster_id | <UUID of the external orchestrator> |
| orchestrator_system/cluster_name | <Name given to this external orchestrator> |
| orchestrator_system/workload_type | service |

| Key | Value |
|----------------------------------|---|
| orchestrator_system/service_name | <Name of the load balancing virtual server> |

Generated labels

For each load balancing virtual server the external orchestrator will generate the following labels:

| Key | Value |
|--------------------------------------|--------------------------------|
| orchestrator_annotation/snat_address | <Virtual servers SNAT address> |

Policy enforcement for Citrix Netscaler

This feature enables Secure Workload to translate logical policies with provider groups that match labelled *Citrix Netscaler* virtual servers into *Citrix Netscaler* ACL rules and deploys them to the load balancer appliance using its REST API. As mentioned above all existing ACL rules will be replaced by Secure Workload generated policy rules.

By default, the field *Enable Enforcement* is not checked, ie. disabled, in the dialog *Create Orchestrator* as shown in the picture below:

Figure 44: Configuration Option "Enable Enforcement"

Create External Orchestrator Configuration

Route Domain

Basic Config

Hosts List

Username

Username for the orchestration workload

Password

Password for the orchestration workload

CA Certificate

CA Certificate to validate orchestration workload

Accept Self-signed Cert

Secure Connector Tunnel

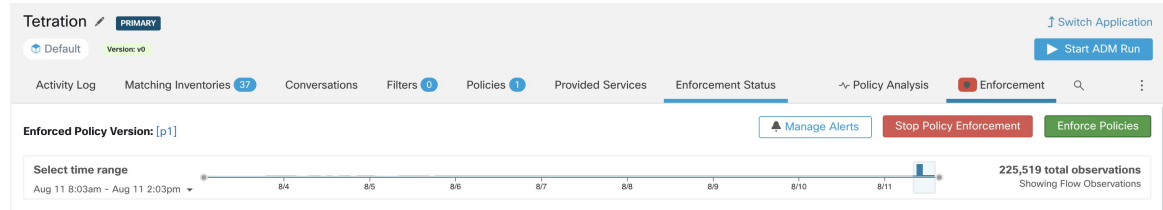
Enable Enforcement

Connection will be tested after the creation.

Just click on the designated check box to enable enforcement for the orchestrator. This option can be modified any time as needed.

Enable enforcement for the orchestrator, regardless whether it is done by creating or editing the orchestrators configuration, will not deploy the current logical policies to the load balancer appliance immediately. This task is performed as part of the workspace policy enforcement to be triggered by the user as shown in the following picture or due to any updates of inventories. However, disable enforcement for the orchestrator will cause all deployed ACL rules being removed from the *Citrix Netscaler* load balancer immediately.

Figure 45: Workspace Policy Enforcement



Note

- The orchestrator for *Citrix Netscaler* also detects any deviation of ACL rules and replaces it with Secure Workload policies, ie. any policy changes towards the load balancing virtual servers should be done with Secure Workload only.
- When policy enforcement is stopped or the external orchestrator is deleted, the ACLs will become empty as all Secure Workload policies will be removed from *Citrix Netscaler* load balancer.

The OpenAPI Policy enforcement status for external orchestrator can be used to retrieve the status of Secure Workload policy enforcement to the load balancer appliance associated with the external orchestrator. This helps to verify if the deployment of ACL rules to the *Citrix Netscaler* appliance has succeeded or failed.

Caveats

- If enforcement is enabled, the Secure Workload policies will always be deployed to the global list of ACLs, ie. partition *default*.
- Only VIP specified as a single address is supported, ie. VIP given as an address pattern is not supported.
- Visibility for the detected services (*Citrix Netscaler* virtual servers) is not supported.

Troubleshooting

- Connectivity issue Secure Workload will attempt to connect to the provided IP/hostname and port number using an HTTPS connection originating from one of the Secure Workload appliance servers or from the cloud in the case of *TaaS* or from the VM hosting the Secure Workload Secure Connector tunnel service. In order to correctly establish this connection, firewalls must be configured to permit this traffic. Also, make sure the given credentials are correct and have privileges with read and write access to send REST API requests to the *Citrix Netscaler* appliance.
- ACL rules not found In case no ACL rules are found, after policy enforcement was performed, make sure the corresponding virtual server is enabled, ie. its status must be *up*.

TAXII

The TAXII (Trusted Automated Exchange of Intelligence Information) Integration allows Secure Workload to ingest threat intelligence data feeds from security vendors to annotate network flows and process hashes with STIX (Structured Threat Information Expression) indicators such as malicious IPs, malicious hashes.

When an external orchestrator configuration is added for type “taxii”, the Secure Workload appliance will attempt to connect to the TAXII server(s) and poll STIX data feed collections. The STIX data feeds (only IPs and binary hashes indicators) will be parsed and used to annotate network flows and process hashes in the Secure Workload pipelines (as belonging to the Tenant under which the orchestrator is configured).

Network flows with either provider or consumer addresses matched imported malicious IPs will be tagged with multi-value label “orchestrator_malicious_ip_by_<vendor name>” where <vendor name> is the user orchestrator configuration input TAXII vendor, and the label value is “Yes”.

The ingested STIX binary hash indicators will be used to annotate workload process hashes, which will be displayed (if matched) in the Security Dashboard / Process Hash Score Details and in the Workload Profile / File Hashes.

Prerequisites

- Secure Connector Tunnel, if needed for connectivity
- Supported TAXII Servers: 1.0
- Supported TAXII feeds with STIX version: 1.x

Configuration fields

Beside the common configuration fields as described in *Create External Orchestrator* the following fields can be configured:

| Common Field | Required | Description |
|------------------------|----------|---|
| Name | Yes | User specified name of the orchestrator. |
| Description | Yes | User specified description of the orchestrator. |
| Vendor | Yes | The vendor provides intelligence data feeds. |
| Full Snapshot Interval | Yes | The interval (in seconds) to perform a full snapshot of the TAXII feed. (Default: 1 day) |
| Poll Url | Yes | The polling full URL path to poll data. |

| Common Field | Required | Description |
|-------------------------|----------|--|
| Collection | Yes | The TAXII feed collection name to be polled. |
| Poll Days | Yes | The number of earlier days threat data to poll from TAXII feed. |
| Username | | Username for authentication. |
| Password | | Password for authentication. |
| Certificate | | Client certificate used for authentication. |
| Key | | Key corresponding to client certificate. |
| CA Certificate | | CA Certificate to validate orchestration endpoint. |
| Accept Self-Signed Cert | | Checkbox to disable strictSSL checking of the TAXII API server certificate |
| Secureconnector Tunnel | | Tunnel connections to this orchestrator's hosts through the Secure Connector tunnel. |
| Hosts List | Yes | The hostname/ip and port pairs pointing to the TAXII server(s). |

Workflow

- First, the user must verify that the TAXII server is reachable on that IP/Port from the Secure Workload cluster.
- Configure the correct TAXII server with the poll path and TAXII feed name.

Generated labels

| Key | Value |
|----------------------------------|---|
| orchestrator_system/orch_type | <i>TAXII</i> |
| orchestrator_system/cluster_id | UUID of the cluster's configuration in Secure Workload. |
| orchestrator_system/cluster_name | Name given to this cluster's configuration>. |

| Key | Value |
|---|--|
| orchestrator_malicious_ip_by_ <vendor> | Yes if the flow provider/consumer address matches the imported TAXII malicious IPs data. |

Caveats

- The TAXII integration is supported only on on-premise Secure Workload.
- Only IPs and hashes indicators from TAXII feeds are ingested.
- Maximum number of ingested IPs is 100K (most recently updated) per TAXII feed.
- Maximum number of ingested hashes is 500K (most recently updated) for all TAXII feeds.
- Only TAXII feeds with STIX version 1.x are supported.

Troubleshooting

- Connectivity Issues

The Secure Workload will attempt to connect to the provided poll URL path from one of the Secure Workload appliance servers or from the VM hosting the Secure Workload Secure Connector VPN tunnel service. In order to correctly establish this connection, firewalls must be configured to permit this traffic.

Behavior of Full polling for TAXII Orchestrators

Default Full Snapshot Interval is 24 hours

Every full snapshot interval, Secure Workload will perform pulling TAXII feeds of IPs and hashes up to the above limits into the label database.



CHAPTER 4

Configure and Manage Connectors for Secure Workload

Connectors enable Secure Workload to integrate with external resources, such as network switches, routers, firewalls, and endpoint management systems, to collect telemetry data, ingest flow observations, and enrich inventory and endpoint context.

Table 13: Feature History

| Feature Name | Release | Feature Description | Where to Find |
|---------------------------------|---------|---|--|
| Identity Connector for OpenLDAP | 3.9 | The Identity Connector serves as a centralized hub for integrating with identity stores, allowing you to seamlessly pull users, user groups, and other attributes from the OpenLDAP server. | Identity Connectors, on page 261 |

- [What are Connectors, on page 169](#)
- [Connector Alerts, on page 265](#)
- [Life Cycle Management of Connectors, on page 270](#)
- [Virtual Appliances for Connectors, on page 275](#)
- [Configuration Management on Connectors and Virtual Appliances, on page 286](#)
- [Troubleshooting, on page 301](#)
- [Cisco Secure Firewall Management Center, on page 330](#)

What are Connectors

Connectors in Cisco Secure Workload are integrations that allow Secure Workload to interact with and gather data from various resources for different purposes. To configure and work with connectors, from the navigation pane, choose **Manage > Connectors**.



Note Connectors require a virtual appliance. For more information, see [Virtual Appliances for Connectors](#).

Connectors for Flow Ingestion

Connectors stream flow observations from different Network switches, routers, and other middle-boxes (such as load balancers and firewalls) to Secure Workload for flow ingestion.

Secure Workload supports flow ingestion through NetFlow v9, IPFIX, and custom protocols. In addition to flow observations, middle-box connectors actively stitch client-side and server-side flows to understand which client flows are related to which server flows.

| Connector | Description | Deployed on Virtual Appliance |
|--|--|-------------------------------|
| NetFlow | Collect NetFlow V9 and/or IP-FIX telemetry from network devices such as routers and switches. | Secure Workload Ingest |
| F5 BIG-IP | Collect telemetry from F5 BIG-IP, stitch client, and server side flows, enrich client inventory with user attributes. | Secure Workload Ingest |
| Citrix NetScaler | Collect telemetry from Citrix ADC, stitch client, and server side flows. | Secure Workload Ingest |
| Cisco Secure Connector Firewall | Collect telemetry data from Secure Firewall ASA, Secure Firewall Threat Defense, stitch client, and server side flows. | Secure Workload Ingest |
| Meraki | Collect telemetry data from Meraki firewalls. | Secure Workload Ingest |
| ERSPAN | Collect ERSPAN telemetry data from network devices which support ERSPAN | Secure Workload Ingest |
| See also | Cloud Connectors | – |

For more information about required virtual appliances, see [Virtual Appliances for Connectors](#).

NetFlow Connector

NetFlow connector allows Secure Workload to ingest flow observations from routers and switches in the network.

This solution enables the hosts to avoid running software agents since the Cisco switches relay NetFlow records to a NetFlow connector hosted in a Secure Workload Ingest appliance for processing.

Figure 46: NetFlow connector

NetFlow ✔

Info IP bindings Log Alert Troubleshoot

Listening for

| | | |
|-----------------|----|--|
| NETFLOW9 | on | 172.21.156.31 : 4729 / udp |
| NETFLOW9 | on | 2001:420:28d:2022::3:2f31 : 4729 / udp |
| IPFIX | on | 172.21.156.31 : 4739 / udp |
| IPFIX | on | 2001:420:28d:2022::3:2f31 : 4739 / udp |

Sources

| IP Address | Protocol | Last received at |
|---------------------------|----------|-------------------------|
| 2001:420:28d:2022::3:2f05 | NETFLOW9 | Mar 7 08:29:23 pm (PST) |

NetFlow

Enabled on February 28, 2023

Data Ingest Appliance

Enable Another

Delete Connector

Capabilities

Flow Visibility

What is NetFlow

NetFlow protocol allows routers and switches to aggregate traffic passing through them into flows and export these flows to a flow collector.

The flow collector receives these flow records and stores them in their flow storage for offline querying and analysis. Cisco routers and switches support NetFlow.

Typically, the setup involves the following steps:

1. Enable the NetFlow feature on one or more network devices and configure the flow templates that devices should export.
2. Configure the NetFlow collector endpoint information on the remote network devices. This NetFlow collector is listening on the configured endpoint to receive and process NetFlow flow records.

Flow Ingestion to Secure Workload

NetFlow connector is essentially a NetFlow collector. The connector receives the flow records from the network devices and forwards them to Secure Workload for flow analysis. You can enable a NetFlow connector on a Secure Workload Ingest appliance and run it as a Docker container.

NetFlow connector also registers with Secure Workload as a Secure Workload NetFlow agent. NetFlow connector decapsulates the NetFlow protocol packets (that is, flow records); then processes and reports the flows like a regular Secure Workload agent. Unlike a Deep Visibility Agent, it does not report any process or interface information.



Note NetFlow connector supports NetFlow v9 and IPFIX protocols.



Note Each NetFlow connector should report only flows for one VRF. The connector exports the flows and places them in the VRF based on the Agent VRF configuration in the Secure Workload cluster.

To configure the VRF for the connector, choose **Manage > Agents** and click the **Configuration** tab. In this page, under the *Agent Remote VRF Configurations* section, click *Create Config* and provide the details about the connector.

The form requests you to provide: the name of the VRF, the IP subnet of the connector, and the range of port numbers that can potentially send flow records to the cluster.

Rate Limiting

NetFlow connector accepts up to 15000 flows per second. Note that a given NetFlow v9 or IPFIX packet could contain one or more flow and template records. NetFlow connector parses the packets and identifies the flows. If the connector parses more than 15000 flows per second, it drops the additional flow records.

Also note the Secure Workload customer supports the NetFlow connector only if the flow rate is within this acceptable limit.

If the flow rate exceeds 15000 flows per second, we recommend first adjusting the flow rate to fall within the limits, and maintaining this level for at least three days (to rule out issues related to higher incoming flow rate).

If the original issue persists, customer support starts to investigate the issue and identify proper workaround and/or solution.

Supported Information Elements

NetFlow connector *only* supports the following information elements in NetFlow v9 and IPFIX protocols. For more information, see [IP Flow Information Export \(IPFIX\) Entities](#).

| Element ID | Name | Description | Mandatory |
|------------|--------------------|---|-----------|
| 1 | octetDeltaCount | Number of octets in incoming packets for this flow. | Yes |
| 2 | packetDeltaCount | Number of incoming packets for this flow. | Yes |
| 4 | protocolIdentifier | The value of the protocol number in the IP packet header. | Yes |
| 6 | tcpControlBits | TCP control bits observed for packets of this flow. The agent handles FIN, SYN, RST, PSH, ACK, and URG flags. | No |

| Element ID | Name | Description | Mandatory |
|------------|--------------------------|---|-----------------|
| 7 | sourceTransportPort | The source port identifier in the transport header. | Yes |
| 8 | sourceIPv4Address | The IPv4 source address in the IP packet header. | Either 8 or 27 |
| 11 | destinationTransportPort | The destination port identifier in the transport header. | Yes |
| 12 | destinationIPv4Address | The IPv4 destination address in the IP packet header. | Either 12 or 28 |
| 27 | sourceIPv6Address | The IPv6 source address in the IP packet header. | Either 8 or 27 |
| 28 | destinationIPv6Address | The IPv6 destination address in the IP packet header. | Either 12 or 28 |
| 150 | flowStartSeconds | The absolute timestamp of the first packet of the flow (in seconds). | No |
| 151 | flowEndSeconds | The absolute timestamp of the last packet of the flow (in seconds). | No |
| 152 | flowStartMilliseconds | The absolute timestamp of the first packet of the flow (in milliseconds). | No |
| 153 | flowEndMilliseconds | The absolute timestamp of the last packet of the flow (in milliseconds). | No |
| 154 | flowStartMicroseconds | The absolute timestamp of the first packet of the flow (in microseconds). | No |
| 155 | flowEndMicroseconds | The absolute timestamp of the last packet of the flow (in microseconds). | No |
| 156 | flowStartNanoseconds | The absolute timestamp of the first packet of the flow (in nanoseconds). | No |

| Element ID | Name | Description | Mandatory |
|------------|--------------------|---|-----------|
| 157 | flowEndNanoseconds | The absolute timestamp of the last packet of the flow (in nanoseconds). | No |

How to configure NetFlow on the Switch

The following steps are for a Nexus 9000 switch. The configurations may slightly differ for other Cisco platforms. In any case, refer to the official Cisco configuration guide for the Cisco platform you're configuring.

Procedure

Step 1 Enter global configuration mode.

```
switch# configure terminal
```

Step 2 Enable NetFlow feature.

```
switch(config)# feature netflow
```

Step 3 Configure a flow record.

The following example configuration shows how to generate five tuple information of a flow in a NetFlow record.

```
switch(config)# flow record ipv4-records
switch(config-flow-record)# description IPv4Flow
switch(config-flow-record)# match ipv4 source address
switch(config-flow-record)# match ipv4 destination address
switch(config-flow-record)# match ip protocol
switch(config-flow-record)# match transport source-port
switch(config-flow-record)# match transport destination-port
switch(config-flow-record)# collect transport tcp flags
switch(config-flow-record)# collect counter bytes
switch(config-flow-record)# collect counter packets
```

Step 4 Configure a flow exporter.

The following example configuration specifies the NetFlow protocol version, NetFlow template exchange interval, and NetFlow collector endpoint details. Specify the IP and port on which you enable the NetFlow connector on a Secure Workload Ingest appliance.

```
switch(config)# flow exporter flow-exporter-one
switch(config-flow-exporter)# description NetFlowv9ToNetFlowConnector
switch(config-flow-exporter)# destination 172.26.230.173 use-vrf management
switch(config-flow-exporter)# transport udp 4729
switch(config-flow-exporter)# source mgmt0
switch(config-flow-exporter)# version 9
switch(config-flow-exporter-version-9)# template data timeout 20
```

Step 5 Configure a flow monitor.

Create a flow monitor and associate it with a flow record and flow exporter.

```
switch(config)# flow monitor ipv4-monitor
switch(config-flow-monitor)# description IPv4FlowMonitor
```



```
switch(config-flow-monitor)# record ipv4-records
switch(config-flow-monitor)# exporter flow-exporter-one
```

Step 6 Apply the flow monitor to an interface.

```
switch(config)# interface Ethernet 1/1
switch(config-if)# ip flow monitor ipv4-monitor input
```

The above steps configure NetFlow on the Nexus 9000 to export NetFlow v9 protocol packets for ingress traffic going through interface 1/1. It sends the flow records to 172.26.230.173:4729 over a UDP protocol. Each flow record includes five tuple information of the traffic and the byte/packet count of the flow.

Figure 47: Running configuration of NetFlow on Cisco Nexus 9000 Switch

```
switch# show running-config netflow

!Command: show running-config netflow
!Time: Wed Mar 21 04:25:21 2018

version 7.0(3)I7(1)
feature netflow

flow timeout 60
flow exporter flow-exporter-173
  destination 172.26.230.173 use-vrf management
  transport udp 4729
  source mgmt0
  version 9
  template data timeout 20
flow record ipv4-records
  match ipv4 source address
  match ipv4 destination address
  match ip protocol
  match transport source-port
  match transport destination-port
  collect transport tcp flags
  collect counter bytes
  collect counter packets
  collect timestamp sys-uptime first
  collect timestamp sys-uptime last
flow monitor ipv4-monitor
  record ipv4-records
  exporter flow-exporter-173

interface Ethernet1/1
  ip flow monitor ipv4-monitor input

interface Ethernet1/2
  ip flow monitor ipv4-monitor input

switch#
```

How to Configure the Connector

For information about required virtual appliances, see [Virtual Appliances for Connectors](#). For NetFlow connectors, IPv4 and IPv6 (dual stack mode) addresses are supported. However, do note that dual stack support is a BETA feature.

The following configurations are allowed on the connector.

- *Log*: For more information, see [Log Configuration](#).

In addition, the listening ports of IPFIX protocol on the connector can be updated on the Docker container in Secure Workload Ingest appliance using an allowed command. This command can be issued on the appliance by providing the connector ID of the connector, type of the port to be update, and the new port information. The connector ID can be found on the connector page in Secure Workload UI. For more information, see [update-listening-ports](#).

Limits

| Metric | Limit |
|---|-------|
| Maximum number of NetFlow connectors on single Secure Workload Ingest appliance | 3 |
| Maximum number of NetFlow connectors on one Tenant (root scope) | 10 |
| Maximum number of NetFlow connectors on Secure Workload | 100 |

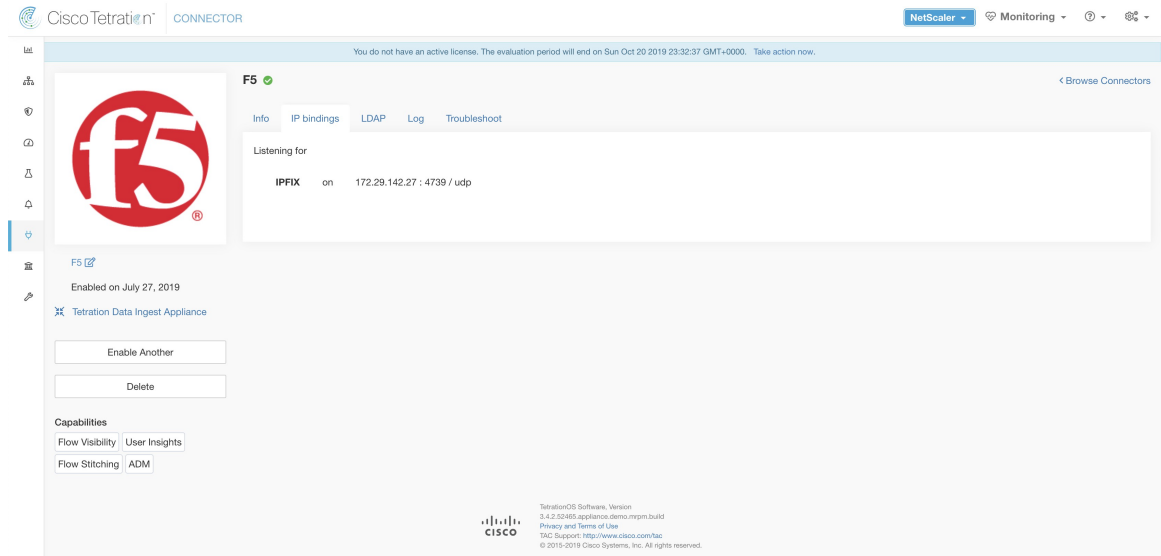
F5 Connector

The F5 connector allows Secure Workload to ingest flow observations from F5 BIG-IP ADCs.

It allows Secure Workload to remotely monitor of flow observations on F5 BIG-IP ADCs, stitching client-side and server-side flows, and annotating users on the client IPs (if user information is available).

Using this solution, the hosts don't need to run software agents because F5 BIG-IP ADCs configure the export of IPFIX records to the F5 connector for processing.

Figure 48: F5 connector



What is F5 BIG-IP IPFIX

F5 BIG-IP IPFIX logging collects flow data for traffic going through the F5 BIG-IP and exports IPFIX records to flow collectors.

Typically, the setup involves the following steps:

1. Create the IPFIX Log-Publisher on the F5 BIG-IP appliance.
2. Configure the IPFIX Log-Destination on the F5 BIG-IP appliance. This log-destination listens on the configured endpoint to receive and process flow records.
3. Create an F5 iRule that publishes IPFIX flow records to the log-publisher.
4. Add the F5 iRule to the virtual server of interest.



Note F5 connector supports F5 BIG-IP software version 12.1.2 and above.

Flow Ingestion to Secure Workload

F5 BIG-IP connector is essentially an IPFIX collector. The connector receives the flow records from F5 BIG-IP ADCs, stitch the NATED flows, and forwards them to Secure Workload for flow analysis. In addition, if LDAP configuration is provided to the F5 connector, it determines values for configured LDAP attributes of a user associated with the transaction (if F5 authenticates the user before processing the transaction). The attributes are associated to the client IP address where the flow happened.



Note F5 connector supports only the IPFIX protocol.



Note Each F5 connector reports only flows for one VRF. The connector puts the flows it exports into the VRF based on the Agent VRF configuration in the Cisco Secure Workload cluster.

To configure the VRF for the connector, choose **Manage > Agents** and click the **Configuration** tab. In this page, under the *Agent Remote VRF Configurations* section, click the *Create Config* and provide the details about the connector. The form requests you to provide: the name of the VRF, the IP subnet of the connector, and the range of port numbers that can potentially send flow records to the cluster.

How to configure IPFIX on F5 BIG-IP

The following steps are for F5 BIG-IP load balancer. (Ref: [Configuring F5 BIG-IP for IPFIX](#))

| Purpose | Description |
|---|---|
| 1. Create a pool of IPFIX collectors. | On a F5 BIG-IP appliance, create the pool of IPFIX collectors. These are the IP addresses associated with F5 connectors on a Secure Workload Ingest appliance. F5 connectors run in Docker containers on the VM listen on port 4739 for IPFIX packets. |
| 2. Create a log-destination. | The log destination configuration on a F5 BIG-IP appliance specifies the actual pool of IPFIX collectors that are used. |
| 3. Create a log-publisher. | A log publisher specifies where F5 BIG-IP sends the IPFIX messages. The publisher is bound with a log-destination. |
| 4. Add a F5 and Secure Workload approved iRule. | Secure Workload and F5 developed iRules that will export flow records to F5 connectors. These iRules will export complete information about a given transaction: including all the endpoints, byte and packet counts, flow start and end time (in milliseconds). F5 connectors will create 4 independent flows and match each flow with its related flow. |
| 5. Add the iRule to the virtual server. | In the iRule settings of a virtual server, add the Secure Workload, approved iRule to the virtual server. |

The above steps configures IPFIX on F5 BIG-IP load balancer to export IPFIX protocol packets for traffic going through the appliance. Here is a sample config of F5.

Figure 49: Running configuration of IPFIX on F5 BIG-IP load balancer

```

root@(localhost)(cfg-sync Standalone)(Active)(/Common)(tmsh)# show running-config ltm virtual vip-1 rules
ltm virtual vip-1 {
  rules {
    ipfix-rule-1
  }
}
root@(localhost)(cfg-sync Standalone)(Active)(/Common)(tmsh)# show running-config ltm pool ipfix-pool-1
ltm pool ipfix-pool-1 {
  members {
    10.28.118.6:ipfix {
      address 10.28.118.6
      session monitor-enabled
      state up
    }
  }
  monitor gateway_icmp
}
root@(localhost)(cfg-sync Standalone)(Active)(/Common)(tmsh)# show running-config ltm virtual vip-1 rules
ltm virtual vip-1 {
  rules {
    ipfix-rule-1
  }
}
root@(localhost)(cfg-sync Standalone)(Active)(/Common)(tmsh)# show running-config sys log-config
sys log-config destination ipfix ipfix-collector-1 {
  pool-name ipfix-pool-1
  transport-profile udp
}
sys log-config publisher ipfix-pub-1 {
  destinations {
    ipfix-collector-1 { }
  }
}
root@(localhost)(cfg-sync Standalone)(Active)(/Common)(tmsh)#

```

In the example above, flow records will be published to *ipfix-pub-1*. *ipfix-pub-1* is configured with log-destination *ipfix-collector-1* which sends the IPFIX messages to IPFIX pool *ipfix-pool-1*. *ipfix-pool-1* has 10.28.118.6 as one of the IPFIX collectors. The virtual server *vip-1* is configured with IPFIX iRule *ipfix-rule-1* which specifies the IPFIX template and how the template gets filled and sent.

- F5 and Secure Workload approved iRule for TCP virtual server. For more information, see [L4 iRule for TCP virtual server](#).
- F5 and Secure Workload approved iRule for UDP virtual server. For more information, see [L4 iRule for UDP virtual server](#).
- F5 and Secure Workload approved iRule for HTTPS virtual server. For more information, see [iRule for HTTPS virtual server](#).



Note Before using the iRule downloaded from this guide, update the **log-publisher** to point to the log-publisher configured in the F5 connector where you add the iRule.



Note F5 has published a GitHub repository, [f5-tetration](#) to help you to start with flow-stitching. The iRules for publishing IPFIX records to the F5 connector for various protocol types are available at: [f5-tetration/irules](#).

Visit the site for the latest iRule definitions. In addition, F5 also develops a script to:

1. Install the correct iRule for the virtual servers.
2. Add a pool of IPFIX collector endpoints (where F5 connectors listen for IPFIX records).
3. Configure the log-collector and log-publisher.
4. Bind the correct iRule to the virtual servers.

This tool minimizes manual configuration and user error while enabling flow-stitching use-case. The script is available at [f5-tetration/scripts](#).

How to Configure the Connector

For information about required virtual appliances, see [Virtual Appliances for Connectors](#).

The following configurations are allowed on the connector.

- **LDAP:** LDAP configuration supports discovery of LDAP attributes and provide a workflow to pick the attribute that corresponds to username and a list of up to 6 attributes to fetch for each user. For more information, see [Discovery](#).
- **Log:** For more information, see [Log Configuration](#).

In addition, the listening ports of IPFIX protocol on the connector can be updated on the Docker container in Secure Workload Ingest appliance using a command that is allowed to be run on the container. This command can be issued on the appliance by providing the connector ID of the connector, type of the port to be update, and the new port information. The connector ID can be found on the connector page in Secure Workload UI. For more information, see [update-listening-ports](#).

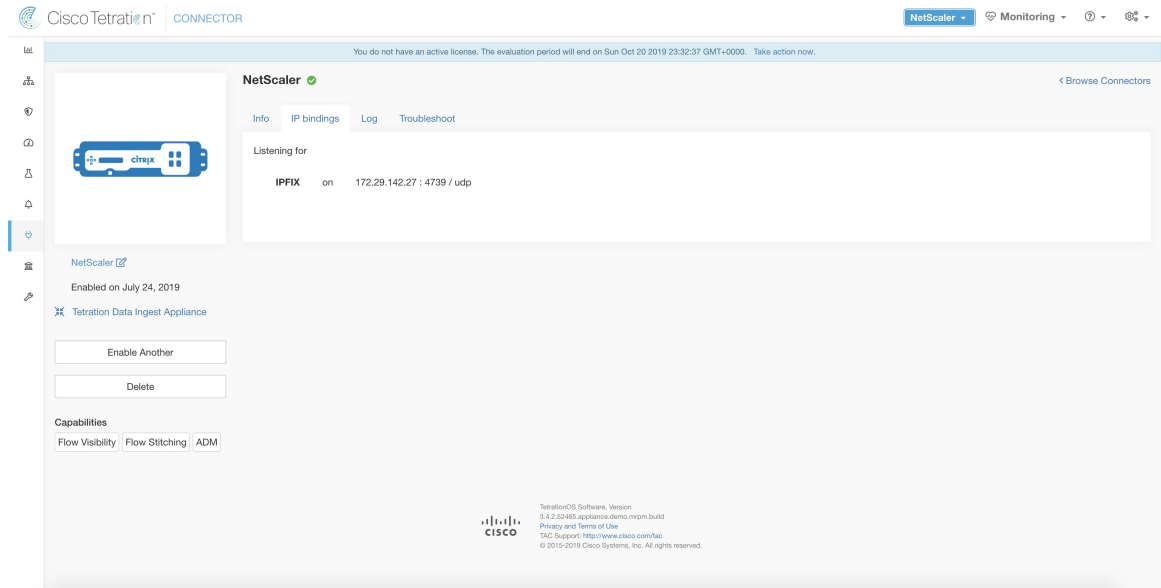
Limits

| Metric | Limit |
|---|-------|
| Maximum number of F5 connectors on one Secure Workload Ingest appliance | 3 |
| Maximum number of F5 connectors on one Tenant (rootscope) | 10 |
| Maximum number of F5 connectors on Secure Workload | 100 |

NetScaler Connector

NetScaler connector allows Secure Workload to ingest flow observations from Citrix ADCs (Citrix NetScalers). It allows Secure Workload to remotely monitor flow observations on Citrix ADCs and stitch client-side and server-side flows. Using this solution, the hosts do not need to run software agents, because Citrix ADCs will be configured to export IPFIX records to NetScaler connector for processing.

Figure 50: NetScaler connector



What is Citrix NetScaler AppFlow

Citrix NetScaler AppFlow collects flow data for traffic going through the NetScaler and exports IPFIX records to flow collectors. Citrix AppFlow protocol uses IPFIX to export the flows to flow collectors. Citrix AppFlow is supported in Citrix NetScaler load balancers.

Typically, the setup involves the following steps:

1. Enable AppFlow feature on one or more Citrix NetScaler instances.
2. Configure the AppFlow collector endpoint information on the remote network devices. This AppFlow collector will be listening on configured endpoint to receive and process flow records.
3. Configure AppFlow actions and policies to export flow records to AppFlow collectors.



Note NetScaler connector supports Citrix ADC software version 11.1.51.26 and above.

Flow Ingestion to Secure Workload

NetScaler connector is essentially a Citrix AppFlow (IPFIX) collector. The connector receives the flow records from Citrix ADCs, stitch the NATed flows and forwards them to Secure Workload for flow analysis. A NetScaler connector can be enabled on a Cisco Secure Workload Ingest appliance and runs as a Docker container. NetScaler connector also registers with Secure Workload as a Secure Workload NetScaler agent.



Note NetScaler connector supports only IPFIX protocol.



Note Each NetScaler connector should report only flows for one VRF. The flows exported by the connector is put in the VRF based on the Agent VRF configuration in the Secure Workload cluster. To configure the VRF for the connector, go to: **Manage > Agents** and click the Configuration tab. In this page, under *Agent Remote VRF Configurations* section, click *Create Config* and provide the details about the connector. The form requests the user to provide: the name of the VRF, IP subnet of the connector, and range of port numbers that can potentially send flow records to the cluster.

How to configure AppFlow on NetScaler

The following steps are for NetScaler load balancer. (Ref: [Configuring AppFlow](#))

Procedure

Step 1 Enable AppFlow on NetScaler.

```
enable ns feature appflow
```

Step 2 Add AppFlow collector endpoints.

The collector receives the AppFlow records from NetScaler. Specify the IP and port of NetScaler connector enabled on a Secure Workload Ingest appliance as an AppFlow collector.

```
add appflow collector c1 -IPAddress 172.26.230.173 -port 4739
```

Step 3 Configure an AppFlow action.

This lists the collectors that will get AppFlow records if the associated AppFlow policy matches.

```
add appflow action a1 -collectors c1
```

Step 4 Configure an AppFlow policy.

This is a rule that has to match for an AppFlow record to be generated.

```
add appflow policy p1 CLIENT.TCP.DSTPORT(22) a1
add appflow policy p2 HTTP.REQ.URL.SUFFIX.EQ("jpeg") a1
```

Step 5 Bind AppFlow policy to Virtual Server.

Traffic hitting the IP of the virtual server (VIP) will be evaluated for AppFlow policy matches. On a match, a flow record is generated and sent to all collectors listed in the associated AppFlow action.

```
bind lb vserver lb1 -policyname p1 -priority 10
```

Step 6 Optionally, bind AppFlow policy globally (for all virtual servers).

An AppFlow policy could also be bound globally to all virtual servers. This policy applies to all traffic that flows through Citrix ADC.

```
bind appflow global p2 1 NEXT -type REQ_DEFAULT
```

Step 7 Optionally, template refresh interval.

Default value for template refresh is 60 seconds.

```
set appflow param -templatereferesh 60
```

The above steps configures AppFlow on Citrix NetScaler load balancer to export IPFIX protocol packets for traffic going through NetScaler. The flow records will be sent to either 172.26.230.173:4739 (for traffic going through vsrver lb1) and to 172.26.230.184:4739 (for all traffic going through the NetScaler). Each flow record includes 5 tuple information of the traffic and the byte/packet count of the flow.

The following screenshot shows a running configuration of AppFlow on a Citrix NetScaler load balancer.

Figure 51: Running configuration of AppFlow on Citrix NetScaler load balancer

```
MAARUMUG-M-M1PB:~ maarumug$ ssh nsroot@172.26.231.131
#####
#                                                                    #
#      WARNING: Access to this system is for authorized users only    #
#      Disconnect IMMEDIATELY if you are not an authorized user!      #
#                                                                    #
#####
Password:
Last login: Fri Dec 15 12:32:45 2017 from 10.128.140.136
Done
> sh run | grep appflow
add appflow collector c1 -IPAddress 172.26.230.174
add appflow collector c2 -IPAddress 172.26.230.173
set appflow param -templateRefresh 60 -connectionChaining ENABLED
add appflow action act1 -collectors c1 c2
add appflow policy pol1 true act1
bind appflow global pol1 1 NEXT -type REQ_DEFAULT
>
```

How to Configure the Connector

For information about required virtual appliances, see [Virtual Appliances for Connectors](#). The following configurations are allowed on the connector.

- *Log*: . For more information, see [Log Configuration](#).

In addition, the listening ports of IPFIX protocol on the connector can be updated on the Docker container in Secure Workload Ingest appliance using a an allowed command. This command can be issued on the appliance by providing the connector ID of the connector, type of the port to be update, and the new port information. The connector ID can be found on the connector page in Secure Workload UI. . For more information, see [update-listening-ports](#).

Limits

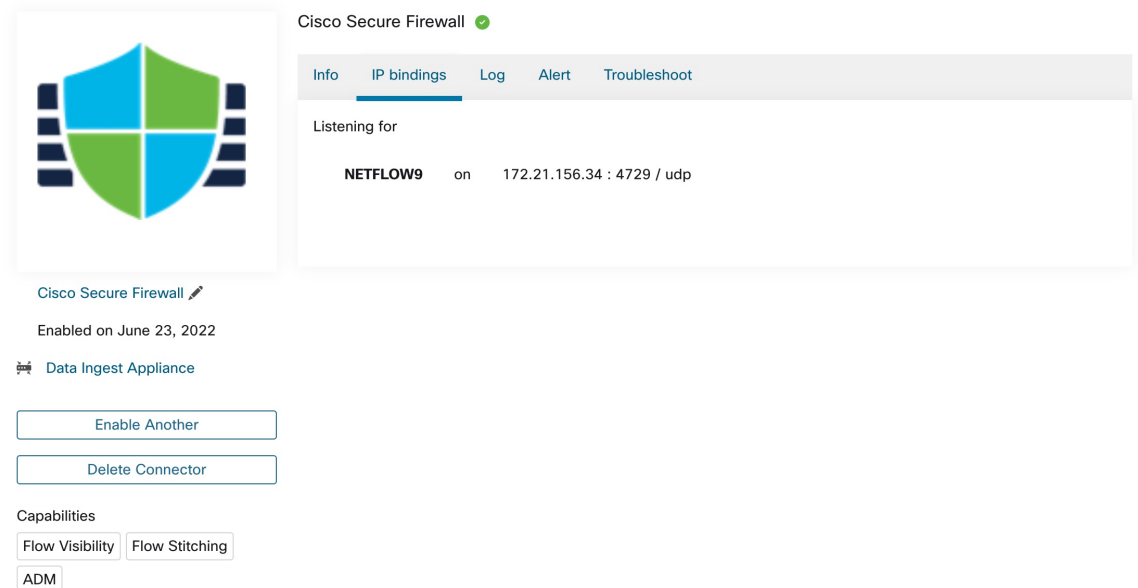
Table 14: Limits

| Metric | Limit |
|--|-------|
| Maximum number of NetScaler connectors on one Secure Workload Ingest appliance | 3 |
| Maximum number of NetScaler connectors on one Tenant (rootscope) | 10 |
| Maximum number of NetScaler connectors on Secure Workload | 100 |

Cisco Secure Firewall Connector

Secure Firewall Connector (formerly known as ASA Connector) allows Secure Workload to ingest flow observations from Secure Firewall ASA (formerly known as Cisco ASA) and Secure Firewall Threat Defense (formerly known as Firepower Threat Defense or FTD). Using this solution, the hosts do not need to run software agents, because the Cisco switches will relay NetFlow Secure Event Logging (NSEL) records to Secure Firewall Connector hosted in a Secure Workload Ingest appliance for processing.

Figure 52: Secure Firewall Connector



Cisco Secure Firewall ASA NetFlow Secure Event Logging (NSEL) provides a stateful, IP flow monitoring that exports significant events in a flow to a NetFlow collector. When an event causes a state change on a flow, an NSEL event is triggered that sends the flow observation along with the event that caused the state change to the NetFlow collector. The flow collector receives these flow records and stores them in their flow storage for offline querying and analysis.

Typically, the setup involves the following steps:

1. Enable NSEL feature on Secure Firewall ASA and/or Secure Firewall Threat Defense.
2. Configure the Secure Firewall connector endpoint information on Secure Firewall ASA and/or Secure Firewall Threat Defense. Secure Firewall connector will be listening on configured endpoint to receive and process NSEL records.

Flow Ingestion to Secure Workload

Secure Firewall connector is essentially a NetFlow collector. The connector receives the NSEL records from Secure Firewall ASA and Secure Firewall Threat Defense, and forwards them to Secure Workload for flow analysis. Secure Firewall connector can be enabled on a Secure Workload Ingest appliance and runs as a Docker container.

Secure Firewall connector also registers with Secure Workload as a Secure Workload agent. Secure Firewall connector decapsulates the NSEL protocol packets (i.e., flow records); then processes and reports the flows like a regular Secure Workload agent. Unlike a Deep Visibility Agent, it does not report any process or interface information.



Note Secure Firewall connector supports NetFlow v9 protocol.



Note Each Secure Firewall connector should report only flows for one VRF. The flows exported by the connector is put in the VRF based on the Agent VRF configuration in Secure Workload cluster. To configure the VRF for the connector, go to: **Manage > Agents** and click the **Configuration** tab. In this page, under *Agent Remote VRF Configurations* section, click *Create Config* and provide the details about the connector. The form requests the user to provide: the name of the VRF, IP subnet of the connector, and range of port numbers that can potentially send flow records to the cluster.

Handling NSEL Events

The following table shows how various NSEL events are handled by Secure Firewall connector. For more information about these elements, see [IP Flow Information Export \(IPFIX\) Entities](#) document.

| Flow Event Element ID: 233 Element Name: <i>NF_F_FW_EVENT</i> | Extended Flow Event Element ID: 33002 Element Name: <i>NF_F_FW_EXT_EVENT</i> | Action on Secure Firewall connector |
|---|--|-------------------------------------|
| 0 (default, ignore this value) | Don't care | No op |
| 1 (Flow created) | Don't care | Send flow to Secure Workload |
| 2 (Flow deleted) | > 2000 (indicates the termination reason) | Send flow to Secure Workload |

| Flow Event Element ID: 233 Element Name: <i>NF_F_FW_EVENT</i> | Extended Flow Event Element ID: 33002 Element Name: <i>NF_F_FW_EXT_EVENT</i> | Action on Secure Firewall connector |
|---|--|--|
| 3 (Flow denied) | 1001 (denied by ingress ACL) | Send flow with disposition marked as rejected to Secure Workload |
| | 1002 (denied by egress ACL) | |
| | 1003 (denied connection by ASA interface or denied ICMP(v6) to device) | |
| | 1004 (first packet on TCP is not SYN) | |
| 4 (Flow alert) | Don't care | No op |
| 5 (Flow updated) | Don't care | Send flow to Secure Workload |

Based on the NSEL record, Secure Firewall connector sends flow observation to Secure Workload. NSEL flow records are bidirectional. So, Secure Firewall connector sends 2 flows: forward flow and reverse flow to Secure Workload.

Here are the details about flow observation sent by Secure Firewall connector to Secure Workload.

Forward Flow observation

| Field | NSEL Element ID | NSEL Element Name |
|---------------------|-----------------|------------------------------------|
| Protocol | 4 | <i>NF_F_PROTOCOL</i> |
| Source Address | 8 | <i>NF_F_SRC_ADDR_IPV4</i> |
| | 27 | <i>NF_F_SRC_ADDR_IPV6</i> |
| Source Port | 7 | <i>NF_F_SRC_PORT</i> |
| Destination Address | 12 | <i>NF_F_DST_ADDR_IPV4</i> |
| | 28 | <i>NF_F_DST_ADDR_IPV6</i> |
| Destination Port | 11 | <i>NF_F_DST_PORT</i> |
| Flow Start Time | 152 | <i>NF_F_FLOW_CREATE_TIME_MSEC</i> |
| Byte Count | 231 | <i>NF_F_FWD_FLOW_DELTA_BYTES</i> |
| Packet Count | 298 | <i>NF_F_FWD_FLOW_DELTA_PACKETS</i> |

Reverse Flow Information

| Field | NSEL Element ID | NSEL Element Name |
|----------|-----------------|----------------------|
| Protocol | 4 | <i>NF_F_PROTOCOL</i> |

| Field | NSEL Element ID | NSEL Element Name |
|---------------------|-----------------|------------------------------------|
| Source Address | 12 | <i>NF_F_DST_ADDR_IPV4</i> |
| | 28 | <i>NF_F_DST_ADDR_IPV6</i> |
| Source Port | 11 | <i>NF_F_DST_PORT</i> |
| Destination Address | 8 | <i>NF_F_SRC_ADDR_IPV4</i> |
| | 27 | <i>NF_F_SRC_ADDR_IPV6</i> |
| Destination Port | 7 | <i>NF_F_SRC_PORT</i> |
| Flow Start Time | 152 | <i>NF_F_FLOW_CREATE_TIME_MSEC</i> |
| Byte Count | 232 | <i>NF_F_REV_FLOW_DELTA_BYTES</i> |
| Packet Count | 299 | <i>NF_F_REV_FLOW_DELTA_PACKETS</i> |

NAT

If the client to ASA flow is NATed, NSEL flow records indicate the NATed IP/port on the server side. Secure Firewall connector uses this information to stitch server to ASA and ASA to client flows.

Here is the NATed flow record in the forward direction.

| Field | NSEL Element ID | NSEL Element Name |
|---------------------|-----------------|------------------------------------|
| Protocol | 4 | <i>NF_F_PROTOCOL</i> |
| Source Address | 225 | <i>NF_F_XLATE_SRC_ADDR_IPV4</i> |
| | 281 | <i>NF_F_XLATE_SRC_ADDR_IPV6</i> |
| Source Port | 227 | <i>NF_F_XLATE_SRC_PORT</i> |
| Destination Address | 226 | <i>NF_F_XLATE_DST_ADDR_IPV4</i> |
| | 282 | <i>NF_F_XLATE_DST_ADDR_IPV6</i> |
| Destination Port | 228 | <i>NF_F_XLATE_DST_PORT</i> |
| Flow Start Time | 152 | <i>NF_F_FLOW_CREATE_TIME_MSEC</i> |
| Byte Count | 231 | <i>NF_F_FWD_FLOW_DELTA_BYTES</i> |
| Packet Count | 298 | <i>NF_F_FWD_FLOW_DELTA_PACKETS</i> |

The forward flow will be marked as related to the NATed flow record in the forward direction (and vice versa)

Here is the NATed flow record in the reverse direction

| Field | NSEL Element ID | NSEL Element Name |
|----------|-----------------|----------------------|
| Protocol | 4 | <i>NF_F_PROTOCOL</i> |

| Field | NSEL Element ID | NSEL Element Name |
|---------------------|-----------------|------------------------------------|
| Source Address | 226 | <i>NF_F_XLATE_DST_ADDR_IPV4</i> |
| | 282 | <i>NF_F_XLATE_DST_ADDR_IPV6</i> |
| Source Port | 228 | <i>NF_F_XLATE_DST_PORT</i> |
| Destination Address | 225 | <i>NF_F_XLATE_SRC_ADDR_IPV4</i> |
| | 281 | <i>NF_F_XLATE_SRC_ADDR_IPV6</i> |
| Destination Port | 227 | <i>NF_F_XLATE_SRC_PORT</i> |
| Flow Start Time | 152 | <i>NF_F_FLOW_CREATE_TIME_MSEC</i> |
| Byte Count | 232 | <i>NF_F_REV_FLOW_DELTA_BYTES</i> |
| Packet Count | 299 | <i>NF_F_REV_FLOW_DELTA_PACKETS</i> |

The reverse flow will be marked as related to the NATed flow record in the reverse direction (and vice versa).



Note Only NSEL element IDs listed in this section are supported by Secure Firewall connector.

TCP Flags Heuristics

The NSEL records do not have TCP flags information. The Secure Firewall connector uses the following heuristics to set the TCP flags so that the flows can be further analyzed by automatic policy discovery:

- If there are at least one forward packets, adds `SYN` to the forward flow TCP flags.
- If there are at least two forward packets and one reverse packet, adds `ACK` to the forward flow TCP flags and `SYN-ACK` to the reverse flow TCP flags.
- If the previous condition holds true and the flow event is Flow deleted, adds `FIN` to both forward and reverse TCP flags.

How to Configure NSEL on Secure Firewall ASA

The following steps are guidelines on how to configure NSEL and export NetFlow packets to a collector (i.e., Secure Firewall connector). For more information, see the official Cisco configuration guide at [Cisco Secure Firewall ASA NetFlow Implementation Guide](#) for more details.

Here is an example NSEL configuration.

```

flow-export destination outside 172.29.142.27 4729
flow-export template timeout-rate 1
!
policy-map flow_export_policy
 class class-default
  flow-export event-type flow-create destination 172.29.142.27
  flow-export event-type flow-teardown destination 172.29.142.27
  flow-export event-type flow-denied destination 172.29.142.27
  flow-export event-type flow-update destination 172.29.142.27

```

```

user-statistics accounting
service-policy flow_export_policy global

```

In this example, Secure Firewall ASA appliance is configured to send NetFlow packets to *172.29.142.27* on port *4729*. In addition, *flow-export* actions are enabled on *flow-create*, *flow-teardown*, *flow-denied*, and *flow-update* events. When these flow events occur on ASA, a NetFlow record is generated and sent to the destination specified in the configuration.

Assuming a Secure Firewall connector is enabled on Secure Workload and listening on *172.29.142.27:4729* in a Secure Workload Ingest appliance, the connector will receive NetFlow packets from Secure Firewall ASA appliance. The connector processes the NetFlow records as discussed in [Handling NSEL Events](#) and exports flow observations to Secure Workload. In addition, for NATed flows, the connector stitches the related flows (client-side and server-side) flows.

How to Configure the Connector

For information about required virtual appliances, see [Virtual Appliances for Connectors](#). The following configurations are allowed on the connector.

- *Log*: For more information, see [Log Configuration](#).

In addition, the listening ports of IPFIX protocol on the connector can be updated on the Docker container in Secure Workload Ingest appliance using an allowed command. This command can be issued on the appliance by providing the connector ID of the connector, type of the port to be update, and the new port information. The connector ID can be found on the connector page in Secure Workload UI. For more information, see [update-listening-ports](#).

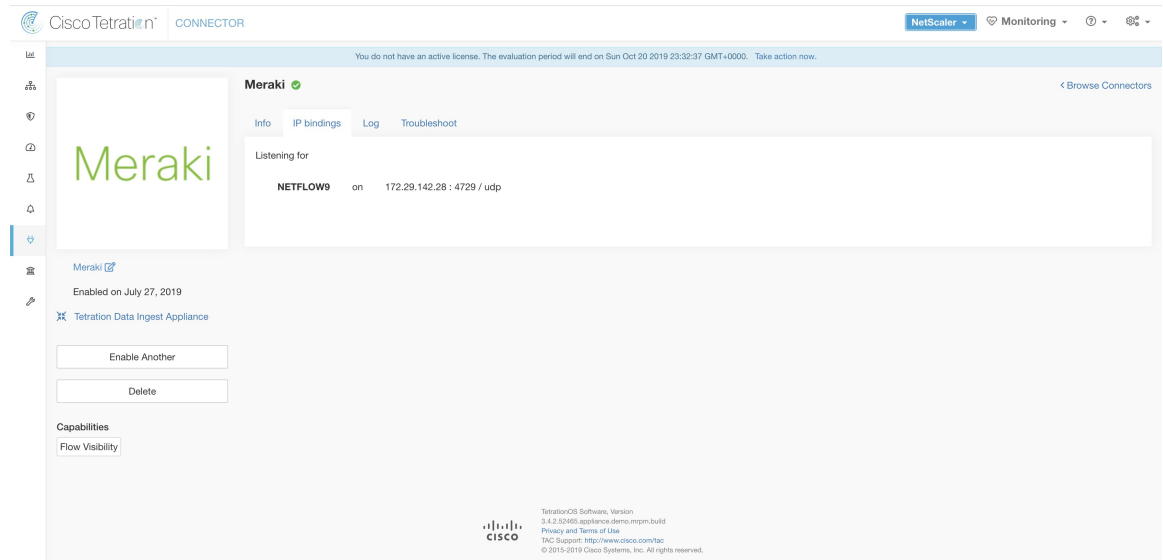
Limits

| Metric | Limit |
|--|-------|
| Maximum number of Secure Firewall connectors on one Secure Workload Ingest appliance | 1 |
| Maximum number of Secure Firewall connectors on one Tenant (rootscope) | 10 |
| Maximum number of Secure Firewall connectors on Secure Workload | 100 |

Meraki Connector

Meraki connector allows Secure Workload to ingest flow observations from Meraki firewalls (included in Meraki MX security appliances and wireless access points). Using this solution, the hosts do not need to run software agents, because the Cisco switches will relay NetFlow records to Meraki connector hosted in a Secure Workload Ingest appliance for processing.

Figure 53: Meraki connector



What is NetFlow

NetFlow protocol allows network devices such as [Meraki Firewall](#) to aggregate traffic that passes through them into flows and export these flows to a flow collector. The flow collector receives these flow records and stores them in their flow storage for offline querying and analysis.

Typically, the setup involves the following steps:

1. Enable NetFlow statistics reporting on Meraki Firewall.
2. Configure the NetFlow collector endpoint information on Meraki Firewall.

Flow Ingestion to Secure Workload

Meraki connector is essentially a NetFlow collector. The connector receives the flow records from the Meraki firewalls that are configured to export NetFlow traffic statistics. It processes the NetFlow records and sends the flow observations reported by Meraki firewalls to Secure Workload for flow analysis. A Meraki connector can be enabled on a Secure Workload Ingest appliance and runs as a Docker container.

Meraki connector also registers with Secure Workload as a Secure Workload Meraki agent. Meraki connector decapsulates the NetFlow protocol packets (i.e., flow records); then processes and reports the flows like a regular Secure Workload agent. Unlike a Deep Visibility Agent, it does not report any process or interface information.



Note Meraki connector supports NetFlow v9 protocol.



Note Each Meraki connector should report only flows for one VRF. The flows exported by the connector is put in the VRF based on the Agent VRF configuration in Secure Workload cluster. To configure the VRF for the connector, go to: **Manage > Agents** and click the **Configuration** tab. In this page, under *Agent Remote VRF Configurations* section, click *Create Config* and provide the details about the connector. The form requests the user to provide: the name of the VRF, IP subnet of the connector, and range of port numbers that can potentially send flow records to the cluster.

Handling NetFlow Records

Based on the NetFlow record, Meraki connector sends flow observation to Secure Workload. Meraki NetFlow flow records are bidirectional. So, Meraki connector sends 2 flows: forward flow and reverse flow to Secure Workload.

Here are the details about flow observation sent by Meraki connector to Secure Workload.

Forward Flow observation

| Field | Element ID | Element Name |
|---------------------|------------|---|
| Protocol | 4 | <i>protocolIdentifier</i> |
| Source Address | 8 | <i>sourceIPv4Address</i> |
| Source Port | 7 | <i>sourceTransportPort</i> |
| Destination Address | 12 | <i>destinationIPv4Address</i> |
| Destination Port | 11 | <i>destinationTransportPort</i> |
| Byte Count | 1 | <i>octetDeltaCount</i> |
| Packet Count | 2 | <i>packetDeltaCount</i> |
| Flow Start Time | | Set based on when the NetFlow record for this flow is received on the connector |

Reverse Flow Information

| Field | Element ID | |
|---------------------|------------|---------------------------------|
| Protocol | 4 | <i>protocolIdentifier</i> |
| Source Address | 8 | <i>sourceIPv4Address</i> |
| Source Port | 7 | <i>sourceTransportPort</i> |
| Destination Address | 12 | <i>destinationIPv4Address</i> |
| Destination Port | 11 | <i>destinationTransportPort</i> |
| Byte Count | 23 | <i>postOctetDeltaCount</i> |

| Field | Element ID | |
|-----------------|------------|---|
| Packet Count | 24 | <i>postPacketDeltaCount</i> |
| Flow Start Time | | Set based on when the NetFlow record for this flow is received on the connector |

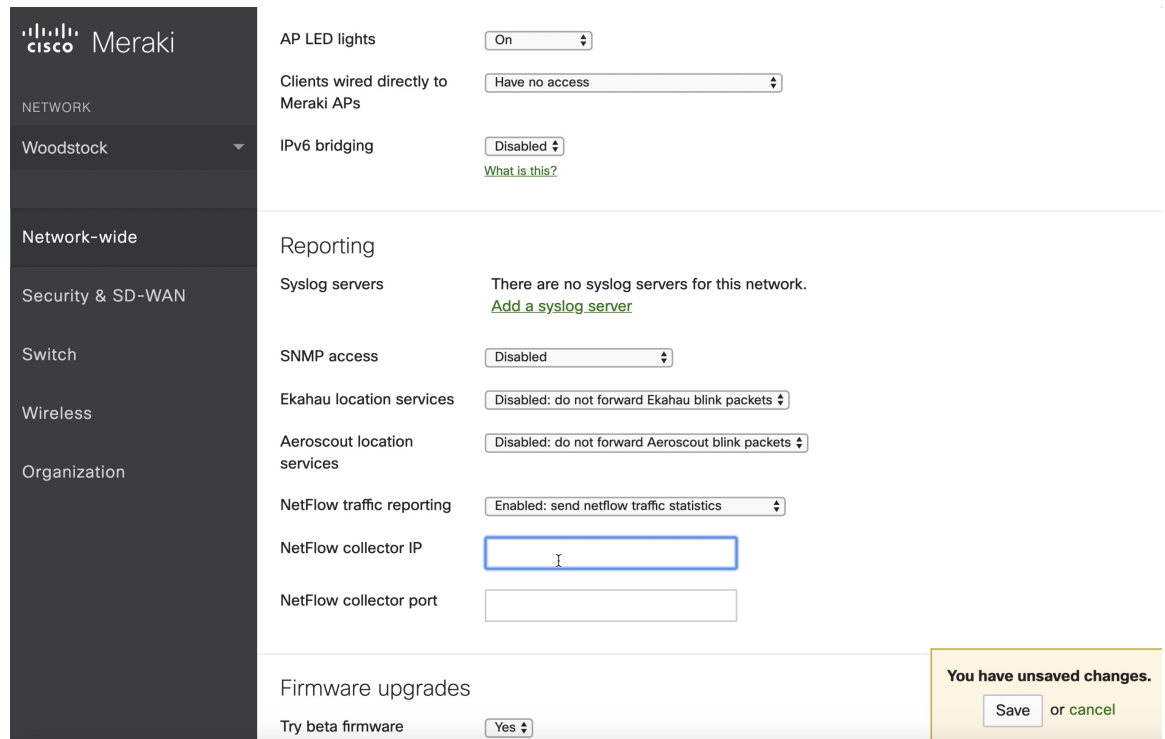
How to configure NetFlow on Meraki Firewall

The following steps show how to configure NetFlow reporting on Meraki Firewall.

Procedure

- Step 1** Login to Meraki UI console.
- Step 2** Navigate to **Network-wide > General**. In *Reporting* settings, enable **NetFlow traffic reporting** and make sure the value is set to *Enabled: send NetFlow traffic statistics*.
- Step 3** Set **NetFlow collector IP** and **NetFlow collector port** to the IP and port on which Meraki connector is listening in Secure Workload Ingest appliance. Default port on which Meraki connector listens for NetFlow records is 4729.
- Step 4** Save the changes.

Figure 54: Enabling NetFlow on a Meraki Firewall



How to Configure the Connector

For information about required virtual appliances, see [Virtual Appliances for Connectors](#). The following configurations are allowed on the connector.

- *Log*: For more information, see [Log Configuration](#).

In addition, the listening ports of NetFlow v9 protocol on the connector can be updated on the Docker container in Secure Workload Ingest appliance using an allowed command. This command can be issued on the appliance by providing the connector ID of the connector, type of the port to be update, and the new port information. The connector ID can be found on the connector page in Secure Workload UI. For more information, see [update-listening-ports](#).

Limits

| Metric | Limit |
|---|-------|
| Maximum number of Meraki connectors on one Secure Workload Ingest appliance | 1 |
| Maximum number of Meraki connectors on one Tenant (rootscope) | 10 |
| Maximum number of Meraki connectors on Secure Workload | 100 |

ERSPAN Connector

ERSPAN connector allows Secure Workload to ingest flow observations from routers and switches in the network. Using this solution, the hosts do not need to run software agents, because the Cisco switches will relay the hosts' traffic to the ERSPAN connector for processing.

What is ERSPAN

Encapsulated Remote Switch Port Analyzer (ERSPAN) is a feature present in most of Cisco switches. It mirrors frames seen by a network device, encapsulates them in a IP packet and sends them to a remote analyzer. Users can select a list of interfaces and/or VLANS on the switch to be monitored.

Commonly, the setup involves configuring source ERSPAN monitoring session(s) on one or more network devices and configuring the destination ERSPAN monitoring session(s) on the remote network device(s) directly connected to a traffic analyzer.

The Secure Workload ERSPAN connector provides both the destination ERSPAN session and traffic analyzer functionalities; therefore there is no need to configure any destination sessions on the switches with the Secure Workload solution.

What are the SPAN Agents

Each ERSPAN connector registers a SPAN agent with the cluster. The Secure Workload SPAN agents are regular Secure Workload agents configured to only process ERSPAN packets: Like Cisco destination ERSPAN sessions, they decapsulate the mirrored frames; then they process and report the flows like a regular Secure Workload agent. Unlike Deep Visibility Agents, they do not report any process or interface information.

What is the Ingest Appliance for ERSPAN

The Secure Workload Ingest appliance for ERSPAN is a VM that internally runs three ERSPAN Secure Workload connectors. It uses the same OVA or QCOW2 as the normal Ingest appliance.

Each connector runs inside a dedicated Docker container to which one vNIC and two vCPU cores with no limiting quota are exclusively assigned.

The ERSPAN connector register a SPAN agent with the cluster with the container hostname: <VM hostname>-<interface IP address>.

The connectors and agents are preserved/restored upon VM, Docker daemon or Docker container crash/reboot.



Note The ERSPAN connector's status will be reported back to the Connector page. See the Agent List page and check the corresponding SPAN agents state.

For more information about required virtual appliances, see [Virtual Appliances for Connectors](#). For ERSPAN connectors, IPv4 and IPv6 (dual stack mode) addresses are supported. However, do note that dual stack support is a BETA feature.

How to configure the source ERSPAN session

The following steps are for a Nexus 9000 switch. The configurations may slightly differ for other Cisco platforms. For configuring a Cisco platform, see the Cisco Secure Workload User Guide.

Figure 55: Configuring ERSPAN source on Cisco Nexus 9000

```
Enter the configuration mode
# config terminal

Configure the erspan source IP address
(config)# monitor erspan origin ip-address 172.28.126.1 global

Create and configure the source erspan session
(config)# monitor session 10 type erspan-source
(config-erspan-src)# source interface ethernet 1/23 both
(config-erspan-src)# source vlan 315, 512
(config-erspan-src)# destination ip 172.28.126.194

Turn on the monitor session
(config-erspan-src)# no shut

Persist the configuration
# copy runnin-config startup-confi
```

The above steps created a source ERSPAN session with id 10. The switch will mirror the frames ingressing and egressing (both) the interface eth1/23 and the ones on VLANS 315 and 512. The outer GRE packet carrying the mirrored frame will have source IP 172.28.126.1 (must be the address of a L3 interface on this switch) and destination IP 172.28.126.194. This is one of the IP addresses configured on the ERSPAN VM.

Supported ERSPAN formats

The Secure Workload SPAN Agents can process ERSPAN type I, II and III packets described in the proposed [ERSPAN RFC](#). Therefore they can process ERSPAN packets generated by Cisco devices. Among the non

RFC compliant formats, they can process the ERSPAN packets generated by VMware vSphere Distributed Switch (VDS).

Performance considerations when configuring ERSPAN source

Carefully choose the ERSPAN source's port/VLAN list. Although the SPAN agent has two dedicated vCPUs, the session may generate considerable amount of packets which could saturate the processing power of the agent. If an agent is receiving more packets than it can process, it will be shown in the Agent Packet Misses graph on the cluster's Deep Visibility Agent page.

More fine grained tuning on which frames the ERSPAN source will mirror can be achieved with ACL policies, usually via the filter configuration keyword.

If the switch supports it, the ERSPAN source session can be configured to modify the maximum transport unit (MTU) of the ERSPAN packet (commonly the default value 1500 bytes), usually via a mtu keyword. Decreasing it will limit the ERSPAN bandwidth usage in your network infrastructure, but it will have no effect on the SPAN Agent load, given the agent's workload is on a per-packet basis. When reducing this value, allow room for 160 bytes for the mirrored frame. For the ERSPAN header overhead details, see the proposed [ERSPAN RFC](#).

There are three versions of ERSPAN. The smaller the version, the lower the ERSPAN header overhead. Version II and III allow for applying QOS policies to the ERSPAN packets, and provide some VLAN info. Version III carries even more settings. Version II is usually the default one on Cisco switches. While Secure Workload SPAN Agents support all three versions, at the moment they do not make use of any extra information the ERSPAN version II and III packets carry.

Security considerations

The Ingest Virtual Machine for ERSPAN guest Operating System is CentOS 7.9, from which OpenSSL server/clients packages were removed.



Note CentOS 7.9 is the guest operating system for Ingest and Edge virtual appliances in Secure Workload 3.8.1.19 and earlier releases. Starting Secure Workload 3.8.1.36, the operating system is AlmaLinux 9.2.

Once the VM is booted and the SPAN agent containers are deployed (this takes a couple of minutes on first time boot only), no network interfaces, besides the loopback, will be present in the Virtual Machine. Therefore the only way to access the appliance is via its console.

The VM network interface are now moved inside the Docker containers. The containers run a centos:7.9.2009 based Docker image with no TCP/UDP port open.



Note Starting Secure Workload 3.8.1.36, the containers run almalinux/9-base:9.2.

Also, the containers are run with the base privileges (no `--privileged` option) plus the `NET_ADMIN` capability. In the unlikely case a container is compromised, the VM guest OS should not be compromisable from inside the container.

All the other security consideration valid for Secure Workload Agents running inside a host do also apply to the Secure Workload SPAN Agents running inside the Docker containers.

Troubleshooting

Once SPAN Agents show in active state in the cluster Monitoring/Agent Overview page, no action is needed on the ERSPAN Virtual Machine, user does not need to log into it. If that is not happening or if the flows are not reported to the cluster, following information will help pinpoint deployment problems.

In normal conditions, on the VM:

- `systemctl status tet_vm_setup` reports an *inactive* service with *SUCCESS* exit status;
- `systemctl status tet-nic-driver` reports an *active* service;
- `docker network ls` reports five networks: *host*, *none* and three `erspan-<iface name>`;
- `ip link` only reports the loopback interface;
- `docker ps` reports three running containers;
- `docker logs <cid>` for each container contains the message: `INFO success: tet-sensor entered RUNNING state, process has stayed up for > than 1 seconds (startsecs)`
- `docker exec <cid> ifconfig` reports only one interface, besides the loopback;
- `docker exec <cid> route -n` reports the default gateway;
- `docker exec <cid> iptables -t raw -S PREROUTING` reports the rule `-A PREROUTING -p gre -j DROP`;

If any of the above does not hold true, check the deployment script logs in `/local/tetration/logs/tet_vm_setup.log` for the reason why the SPAN agent containers deployment failed.

Any other agent registration/connectivity issue can be troubleshooted the same way it is done for agents running on a host via the `docker exec` command:

- `docker exec <cid> ps -ef` reports the two `tet-engine`, `tet-engine check_conf` instances and two `/usr/local/tet/tet-sensor -f /usr/local/tet/conf/.sensor_config` instances, one with root user and one with `tet-sensor` user, along with the process manager `/usr/bin/ python /usr/bin/supervisord -c /etc/supervisord.conf -n` instance.
- `docker exec <cid> cat /usr/local/tet/log/tet-sensor.log` shows the agent's logs;
- `docker exec <cid> cat /usr/local/tet/log/fetch_sensor_id.log` shows the agent's registration logs;
- `docker exec <cid> cat /usr/local/tet/log/check_conf_update.log` shows the configuration update polling logs;

If necessary, traffic to/from the container can be monitored with `tcpdump` after setting into the container's network namespace:

1. Retrieve the container's network namespace (SandboxKey) via `docker inspect <cid> | grep SandboxKey`;
2. Set into the container's network namespace `nsenter --net=/var/run/docker/netns/...`;
3. Monitor `eth0` traffic `tcpdump -i eth0 -n`.

Limits

| Metric | Limit |
|---|------------------|
| Maximum number of ERSPAN connectors on one Secure Workload Ingest appliance | 3 |
| Maximum number of ERSPAN connectors on one Tenant (rootscope) | 24 (12 for TaaS) |
| Maximum number of ERSPAN connectors on Secure Workload | 450 |

Connectors for Endpoints

Connectors for endpoints provide endpoint context for Secure Workload.

| Connector | Description | Deployed on Virtual Appliance |
|-------------------|---|-------------------------------|
| AnyConnect | Collect telemetry data from Cisco AnyConnect Network Visibility Module (NVM) and enrich endpoint inventories with user attributes | Secure Workload Ingest |
| ISE | Collect information about endpoints and inventories managed by Cisco ISE appliances and enrich endpoint inventories with user attributes and secure group labels (SGL). | Secure Workload Edge |

For more information about required virtual appliances, see [Virtual Appliances for Connectors](#).

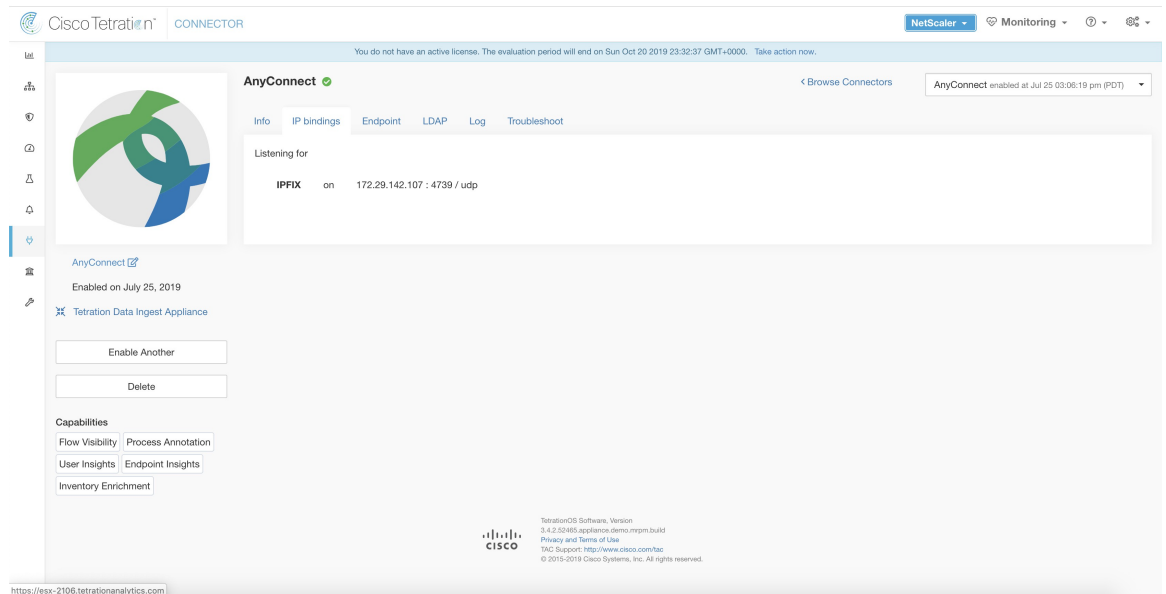
AnyConnect Connector

AnyConnect connector monitors endpoints that run [Cisco AnyConnect Secure Mobility Client](#) with [Network Visibility Module \(NVM\)](#). Using this solution, the hosts do not need to run any software agents on endpoints, because NVM sends host, interface, and flow records in IPFIX format to a collector (e.g., AnyConnect connector).

AnyConnect connector does the following high-level functions.

1. Register each endpoint (supported user devices such as a desktop, a laptop, or a smartphone) on Cisco Secure Workload as an AnyConnect agent.
2. Update interface snapshots from these endpoints with Secure Workload.
3. Send flow information exported by these endpoints to Secure Workload collectors.
4. Periodically send process snapshots for processes that generate flows on the endpoints tracked by the AnyConnect connector.
5. Label endpoint interface IP addresses with Lightweight Directory Access Protocol (LDAP) attributes corresponding to the logged-in-user at each endpoint.

Figure 56: AnyConnect connector



What is AnyConnect NVM

AnyConnect NVM provides visibility and monitoring of endpoint and user behavior both on and off premises. It collects information from endpoints that includes the following context.

1. **Device/Endpoint Context:** Device/endpoint specific information.
2. **User Context:** Users associated with the flow.
3. **Application Context:** Processes associated with the flow.
4. **Location Context:** Location specific attributes -if available.
5. **Destination Context:** FQDN of the destination. AnyConnect NVM generates 3 types of records.

| NVM Record | Description |
|-------------------------|---|
| Endpoint Record | Device/endpoint information including unique device identifier (UDID), hostname, OS name, OS version and manufacturer. |
| Interface Record | Information about each interface in the endpoint including the endpoint UDID, interface unique identifier (UID), interface index, interface type, interface name, and MAC address. |
| Flow Record | Information about flows seen on the endpoint including endpoint UDID, interface UID, 5-tuple (source/destination ip/port and protocol), in/out byte counts, process information, user information, and fqdn of the destination. |

Each record is generated and exported in IPFIX protocol format. When the device is in a trusted network (on-premise/VPN), AnyConnect NVM exports records to a configured collector. AnyConnect connector is an example IPFIX collector that can receive and process IPFIX stream from AnyConnect NVM.



Note AnyConnect connector supports AnyConnect NVM from 4.2+ versions of Cisco AnyConnect Secure Mobility Client.

How to configure AnyConnect NVM

See [How to Implement AnyConnect NVM](#) document for step by step instructions on how to implement AnyConnect NVM using either [Cisco Secure Firewall ASA](#) or [Cisco Identity Services engine \(ISE\)](#). Once NVM module is deployed, an NVM profile should be specified and pushed to and installed on the endpoints running Cisco AnyConnect Secure Mobility Client. When specifying NVM profile, the IPFIX collector should be configured to point to AnyConnect connector on port 4739.

AnyConnect connector also registers with Secure Workload as a Secure Workload AnyConnect Proxy agent.

Processing NVM records

AnyConnect connector processes AnyConnect NVM records as shown below.

Endpoint Record

Upon receiving an endpoint record, AnyConnect connector registers that endpoint as AnyConnect agent on Secure Workload. AnyConnect connector uses the endpoint specific information present in the NVM record along with AnyConnect connector's certificate to register the endpoint. Once an endpoint is registered, data-plane for the endpoint is enabled by creating a new connection to one of the collectors in Secure Workload. Based on the activity (flow records) from this endpoint, AnyConnect connector checks-in the AnyConnect agent corresponding to this endpoint with the cluster periodically (20-30 minutes).

AnyConnect NVM starts to send agent version from 4.9. By default, the AnyConnect endpoint would be registered as version 4.2.x on Secure Workload. This version indicates the minimum supported AnyConnect NVM version. For the AnyConnect endpoints with version 4.9 or newer, the corresponding AnyConnect agent on Secure Workload would show the actual version installed.



Note The AnyConnect agent installed version is not controlled by Secure Workload. Attempting to upgrade the AnyConnect endpoint agent on Secure Workload UI would not take effect.

Interface Record

Interface Record IP address for an interface is not part of the AnyConnect NVM interface record. IP address for an interface is determined when flow records start coming from the endpoint for that interface. Once IP address is determined for an interface, AnyConnect connector sends a complete snapshot of all interfaces of that endpoint whose IP address is determined to config server of Secure Workload. This associates the VRF with the interface data and flows coming in on these interfaces will now be marked with this VRF.

Flow Record

Upon receiving a flow record, AnyConnect connector translates the record to the format that Secure Workload understands and sends FlowInfo over the dataplane corresponding to that endpoint. Furthermore, it stores process information included in the flow record locally. In addition, if LDAP configuration is provided to AnyConnect connector, it determines values for configured LDAP attributes of the logged-in-user of the endpoint. The attributes are associated to the endpoint IP address where the flow happened. Periodically, process information and user labels are pushed to Secure Workload.



Note Each AnyConnect connector will report only endpoints/interfaces/ flows for one VRF. The endpoints and interfaces reported by AnyConnect connector are associated with the VRF based on the Agent VRF configuration in Secure Workload. The flows exported by the AnyConnect connector agent on behalf of the AnyConnect endpoint belong to the same VRF. To configure the VRF for the agent, go to: **Manage > Agents** and click the **Configuration** tab. In this page, under “Agent Remote VRF Configurations” section, click “Create Config” and provide the details about the AnyConnect connector. The form requests the user to provide: the name of the VRF, IP subnet of the host on which the agent is installed, and range of port numbers that can potentially send flow records to the cluster.

Duplicate UDIDs in Windows Endpoints

If endpoint machines are cloned from the same golden image, it is possible that the UDID of all cloned endpoints are identical. In such cases, AnyConnect connector receives endpoint records from these endpoints with identical UDID and registers them on Secure Workload with same UDID. When interface/flow records are received by the connector from these endpoints, it is impossible for the connector to determine the correct AnyConnect agent on Secure Workload to associate the data. The connector associates all the data to one endpoint (and it is not deterministic).

To deal with this problem, AnyConnect NVM 4.8 release ships a tool called *dartcli.exe* to find and regenerate UDID on the endpoint.

- *dartcli.exe -u* retrieves the UDID of the endpoint.
- *dartcli.exe -nu* regenerates the UDID of the endpoint. To run this tool, use the following steps.

```
C:\Program Files (x86)\Cisco\Cisco AnyConnect Secure Mobility Client\DART>dartcli.exe
-u
UDID : 8D0D1E8FA0AB09BE82599F10068593E41EF1BFFF

C:\Program Files (x86)\Cisco\Cisco AnyConnect Secure Mobility Client\DART>dartcli.exe
-nu
Are you sure you want to re-generate UDID [y/n]: y
Adding nonce success
UDID : 29F596758941E606BD0AFF49049216ED5BB9F7A5

C:\Program Files (x86)\Cisco\Cisco AnyConnect Secure Mobility Client\DART>dartcli.exe
-u
UDID : 29F596758941E606BD0AFF49049216ED5BB9F7A5
```

Periodic Tasks

Periodically, AnyConnect connector sends process snapshots and user labels on AnyConnect endpoint inventories.

1. **Process Snapshots:** every 5 minutes, AnyConnect connector walks through the processes it maintains locally for that interval and sends process snapshot for all the endpoints that had flows during that interval.
2. **User Labels:** every 2 minutes, AnyConnect connector walks through the LDAP user labels it maintains locally and updates User Labels on those IP addresses.

For user labels, AnyConnect connector creates a local snapshot of LDAP attributes of all users in the organization. When AnyConnect connector is enabled, configuration for LDAP (server/port information, attributes to fetch for a user, attribute that contains the username) may be provided. In addition, the LDAP user credentials to access LDAP server may be provided. LDAP user credentials are encrypted and never revealed in the AnyConnect connector. Optionally, an LDAP certificate may be provided for securely accessing LDAP server.



Note AnyConnect connector creates a new local LDAP snapshot every 24 hours. This interval is configurable in LDAP configuration of the connector.

How to Configure the Connector

For information about required virtual appliances, see [Virtual Appliances for Connectors](#). The following configurations are allowed on the connector.

- *LDAP:* LDAP configuration supports discovery of LDAP attributes and provide a workflow to pick the attribute that corresponds to username and a list of up to 6 attributes to fetch for each user. For more information, see [Discovery](#).
- *Endpoint:* For more information, see [Endpoint Configuration](#).
- *Log:* For more information, see [Log Configuration](#).

In addition, the listening ports of IPFIX protocol on the connector can be updated on the Docker container in Secure Workload Ingest appliance using an allowed command. This command can be issued on the appliance by providing the connector ID of the connector, type of the port to be update, and the new port information. The connector ID can be found on the connector page in Secure Workload UI. For more information, see [update-listening-ports](#).

Limits

| Metric | Limit |
|---|-------|
| Maximum number of AnyConnect connectors on one Secure Workload Ingest appliance | 1 |
| Maximum number of AnyConnect connectors on one Tenant (rootscope) | 50 |
| Maximum number of AnyConnect connectors on Secure Workload | 500 |

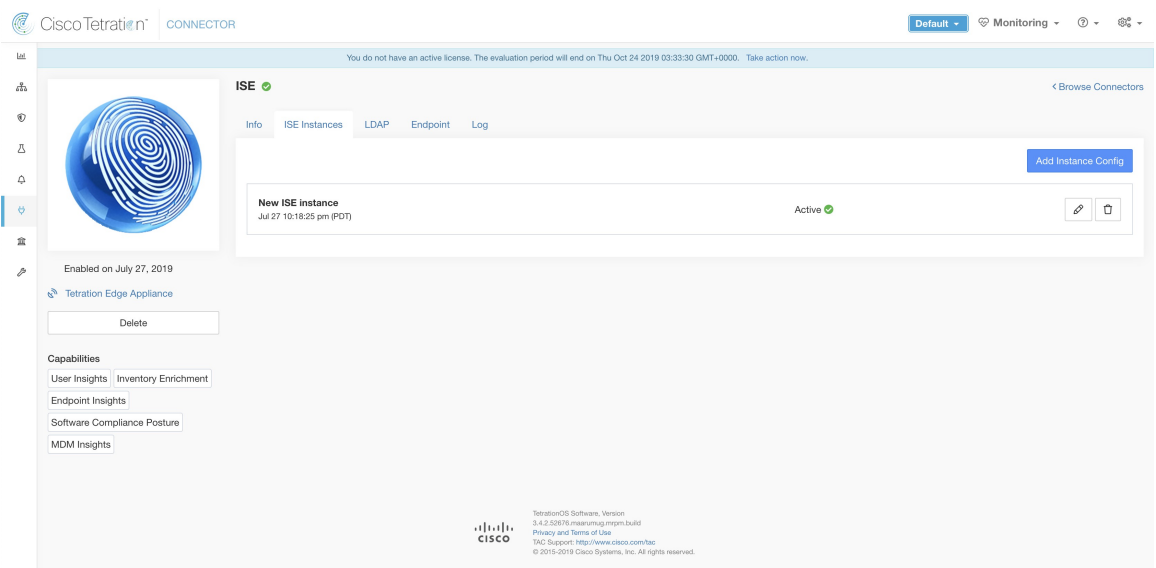
ISE Connector

The ISE connector in Secure Workload connects with [Cisco Identity Services Engine \(ISE\)](#) and ISE Passive Identity Connector (ISE-PIC) using the [Cisco Platform Exchange Grid \(pxGrid\)](#), to retrieve contextual information, such as metadata, for endpoints reported by ISE.

An ISE connector performs these functions:

1. Registers each endpoint that are identified as an ISE endpoint on Secure Workload.
2. Updates metadata information on Secure Workload regarding the endpoints, such as MDM details, authentication, Security Group labels, ISE group name, and ISE group type.
3. Periodically takes a snapshot and updates the cluster with active endpoints visible on the ISE.

Figure 57: ISE connector



Note Each ISE connector will register only endpoints and interfaces for one VRF. The endpoints and interfaces reported by ISE connector are associated with the VRF based on the Agent VRF configuration in Secure Workload. To configure the VRF for the agent, go to: **Manage > Workloads > Agents** and click the **Configuration** tab. In this page, under the **Agent Remote VRF Configurations** section, click **Create Config** and provide the details about the ISE connector. The form requests the user to provide: the name of the VRF, IP subnet of the host on which the agent is installed, and range of port numbers that can potentially register ISE endpoints and interfaces on Secure Workload.



Note The ISE endpoint agents are not listed on the Agents List page; instead ISE endpoints with the attributes can be viewed on the Inventory page.

How to Configure the Connector



Note ISE version 2.4+ and ISE PIC version 3.1+ are required for this integration.

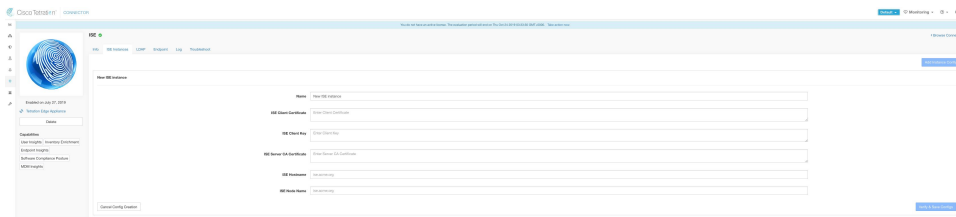
For information about required virtual appliances, see [Virtual Appliances for Connectors](#). For ISE connectors, IPv4 and IPv6 (dual stack mode) addresses are supported. However, do note that dual stack support is a BETA feature.

The following configurations are allowed on the connector.

- **ISE Instance:** ISE connector can connect to multiple instances of ISE using provided configurations. Each instance requires ISE certificate credentials along with hostname and nodename to connect to ISE. For more information, see [ISE Instance Configuration](#).
- **LDAP:** LDAP configuration supports discovery of LDAP attributes and provides a workflow to select the attribute that corresponds to username and a list of up to six attributes to fetch for each user. For more information, see [Discovery](#).
- **Endpoint:** For more information, see [Endpoint Configuration](#).
- **Log:** For more information, see [Endpoint Configuration](#).

ISE Instance Configuration

Figure 58: ISE instance config



Note Starting Cisco Secure Workload version 3.7, the SSL certificate for Cisco ISE pxGrid node requires Subject Alternative Names (SAN) for this integration. Ensure the certification configuration of the ISE nodes is done by your ISE administrator prior to performing the integration with Secure Workload.

To verify your pxGrid node's certificate and confirm if SAN is configured, you need to do the following to verify the certificate from ISE.

Procedure

- Step 1** Go to **Certificates** under **Administration > System**.
- Step 2** Under **Certificate Management**, select **System Certificates**, select your "Used by" pxGrid certificate and choose **View** to review the pxGrid node cert.
- Step 3** Scroll the certificate and ensure the Subject Alternative Names are configured for this certificate.


- Step 4** This certificate should be signed by a valid Certificate Authority (CA), which should also be used to sign the pxGrid client certificate used for the Secure Workload ISE connector.

Figure 59: Example of a valid ISE pxGrid node

Certificate Hierarchy □

ca. [REDACTED].com

ce-ise27.[REDACTED]



ce-ise27.[REDACTED].com

Issued By : ca.[REDACTED].com

Expires : Fri, 2 Aug 2024 19:19:37 UTC

Certificate status is good

Organization Unit (OU) **Tetration Engineering**

Organization (O) **SBG**

City (L) **San Jose**

State (ST) **California**

Country (C) **US**

Serial Number [REDACTED] C0:C2:03:1B:D5:80:57:00:00:00:00:00:0C

Subject Alternative Names IP:172.[REDACTED],IP:1[REDACTED] DNS:ce-ise27.[REDACTED] DNS:ce-ise27.[REDACTED]

Close

- Step 5** You can now generate the pxGrid client certificate signing request using the following template on any host installed with OpenSSL.

```
[req]
distinguished_name = req_distinguished_name
req_extensions = v3_req
x509_extensions = v3_req
prompt = no
[req_distinguished_name]
C = YOUR_COUNTRY
ST = YOUR_STATE
L = YOUR_CITY
O = YOUR_ORGANIZATION
OU = YOUR_ORGANIZATION_UNIT
```

```
CN = ise-connector.example.com
[v3_req]
subjectKeyIdentifier = hash
basicConstraints = critical,CA:false
subjectAltName = @alt_names
keyUsage = critical,digitalSignature,keyEncipherment
extendedKeyUsage = serverAuth,clientAuth
[alt_names]
IP.1 = 10.x.x.x
DNS.1 = ise-connector.example.com
```

Save the file as 'example-connector.cfg' and use the OpenSSL command from your host to generate a Certificate Signing Request (CSR) and the certificate private key with the following command.

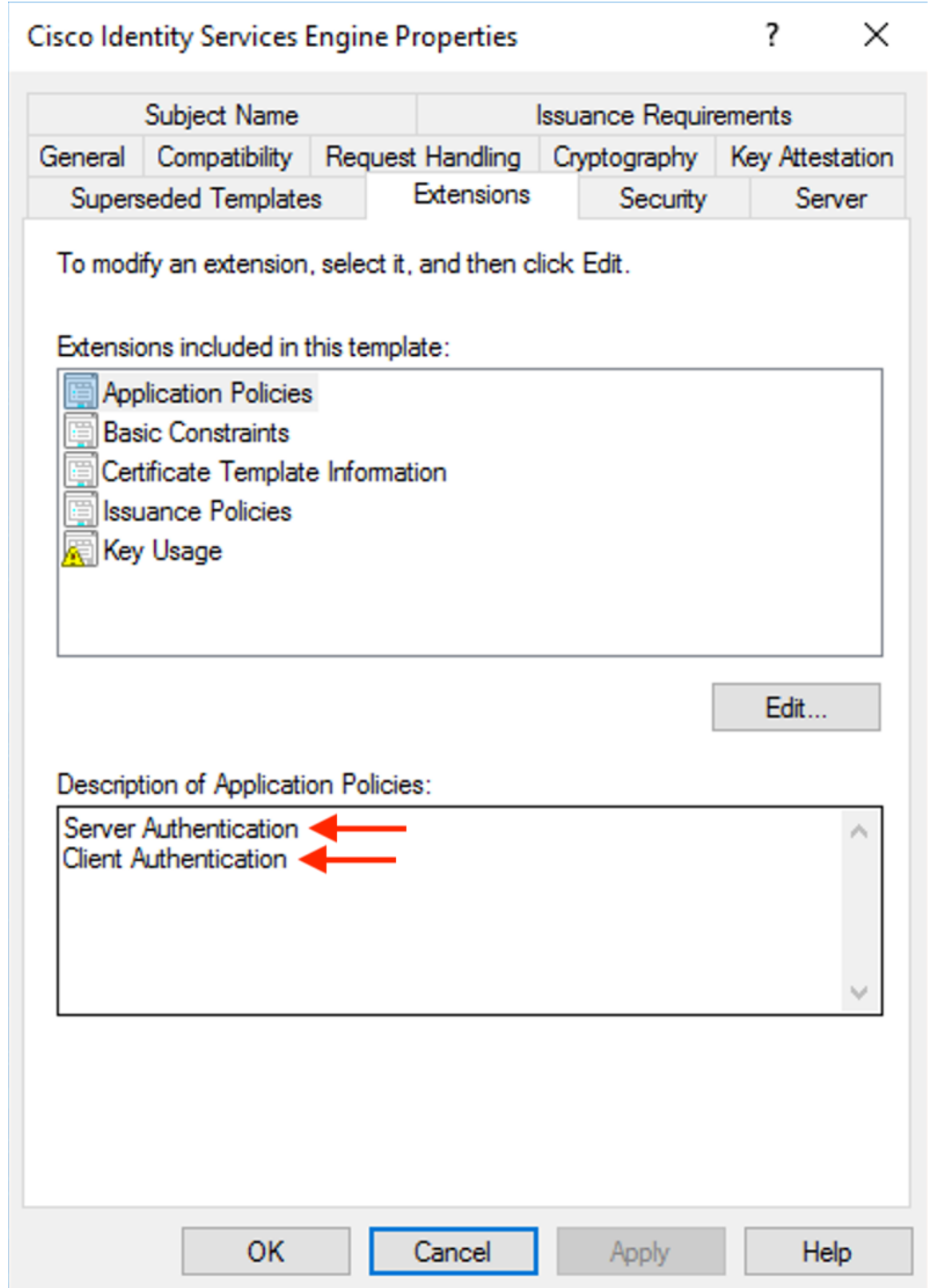
```
openssl req -newkey rsa:2048 -keyout example-connector.key -nodes -out example-connector.csr
-config example-connector.cfg
```

Step 6 Sign the Certificate Signing Request (CSR) by your CA using a Windows CA server. If you are also using a Windows CA server, run the following command to sign the pxGrid client's CSR.

```
certreq -submit -binary -attrib "CertificateTemplate:CiscoIdentityServicesEngine"
example-connector.csr example-connector.cer
```

Note Windows CA requires a Certificate Template. This template should contain the following extensions.

Figure 60: Extensions of Application Policies for a Certificate



Step 7 Copy the signed client certificate and the root CA in PEM format onto your host. This is the same host that generates the client CSR and the private key. Use OpenSSL to ensure the client certificate is in X.509 PEM format. Run the following command using OpenSSL to convert the signed client certificate to the X.509 PEM format.

```
openssl x509 -inform der -in example-connector.cer -out example-connector.pem
```

Step 8 You can also confirm the PEM that is signed by the CA, use the following command.

```
openssl verify -CAfile root-ca.example.com.pem example-connector.pem  
example-connector.pem: OK
```

Note For multi-node ISE deployment with pxGrid, all the pxGrid nodes must trust the Certs used for the Secure Workload ISE Connector.

Step 9 Using the above example's file names, copy the ISE client cert - example-connector.pem, client key - example-connector.key and CA – root-ca.example.com.pem into the respective fields on the ISE configuration page on Secure Workload as shown below.

Note Before upgrading to the latest version of Secure Workload, ensure that you delete the ISE connector to remove any existing configuration data. After the upgrade is complete, configure the ISE connector with the new filters you want to apply.

Figure 61: ISE Connector Configuration

Create new ISE Instance Config

Name

ISE Client Certificate

ISE Client Key

ISE Server CA Certificate

ISE Hostname

ISE Node Name

Ignore ISE Attributes (optional)

ISE IPv4 Subnet Filter (CIDR format) (optional)

ISE IPv6 Subnet Filter (CIDR format) (optional)

Table 15: ISE Connector Configuration

| Field | Description |
|------------------------|---|
| Name | Enter an ISE instance name. |
| ISE Client Certificate | Copy and paste ISE client certificate. |
| ISE Client Key | Copy and paste the ISE client key. The client key must be a clear key, which is not password protected. |

| Field | Description |
|---|--|
| ISE Server CA Certificate | Copy and paste Root CA certificate. |
| ISE Hostname | Enter ISE hostname (FQDN). |
| ISE Node Name | Enter ISE node name. |
| Ignore ISE Attributes (Optional) | Select one or more ISE attributes from the list. Use this option if you do not want to ingest all contextual information of endpoints reported through ISE. |
| ISE IPv4 Subnet Filter (CIDR Format) (Optional) | Enter multiple IPv4 subnets to filter ISE endpoints. |
| ISE IPv6 Subnet Filter (CIDR Format) (Optional) | Enter multiple IPv6 subnets to filter ISE endpoints. |

**Note**

- If an IP Address is used instead of FQDN for the ISE Hostname, then use the IP address in the ISE CA certificate SAN, else, there may be connection failures.
- Number of active endpoints on ISE is not a snapshot, it depends on the configurations on ISE and the aggregation duration for computing the metric. The agent count on Secure Workload is always a snapshot based on last pull from ISE and pxgrid updates, typically the active device count over last one day (default refresh frequency for full snapshots is a day). Due to the difference in the way these numbers are depicted, it is possible that these two numbers will not always match.

Processing ISE records

ISE connector processes records as described below.

Endpoint Record

ISE connector connects to ISE instance and subscribes for any updates for endpoints over pxGrid. Upon receiving an endpoint record, ISE connector registers that endpoint as ISE agent on Secure Workload. ISE connector uses the endpoint specific information present in endpoint record along with ISE connector's certificate to register the endpoint. Once an endpoint is registered, ISE connector uses the endpoint object for inventory enrichment by sending this as user labels on Secure Workload. When ISE connector gets a disconnected endpoint from ISE, it deletes the inventory enrichment from Secure Workload.

Security Group Record

ISE connect also subscribes for updates about Security Group Labels change via pxGrid. On receiving this record, ISE connectors maintains a local database. It uses this database to map SGT name with value on receiving an endpoint record.

Periodic Tasks

ISE connector periodically shares user labels on ISE endpoint inventories.

1. **Endpoint Snapshots:** Every 20 hours, ISE connector fetches a snapshot of endpoints and security group labels from ISE instance and updates the cluster if any change is detected. This call does not compute for endpoints that are disconnected in case we do not see endpoints on Secure Workload coming from ISE.
2. **User Labels:** Every 2 minutes, ISE connector scans through the LDAP user and ISE endpoint labels maintained locally and updates user labels on those IP addresses.

For user labels, ISE connector creates a local snapshot of LDAP attributes of all users in the organization. When ISE connector is enabled, configuration for LDAP (server/port information, attributes to fetch for a user, attribute that contains the username) may be provided. In addition, the LDAP user credentials to access LDAP server may be provided. LDAP user credentials are encrypted and never revealed in the ISE connector. Optionally, an LDAP certificate may be provided for securely accessing LDAP server.



Note ISE connector creates a new local LDAP snapshot every 24 hours. This interval is configurable in LDAP configuration of the connector.



Note On upgrading Cisco ISE device, ISE connector will need to be re-configured with new certificates generated by ISE after upgrade.

Limits

| Metric | Limit |
|---|-------|
| Maximum number of ISE instances that can be configured on one ISE connector | 20 |
| Maximum number of ISE connectors on one Secure Workload Edge appliance | 1 |
| Maximum number of ISE connectors on one Tenant (rootscope) | 1 |
| Maximum number of ISE connectors on Secure Workload | 150 |



Note Maximum number of ISE agents supported per connector is 400000.
 Maximum number of ISE agents supported per connector is 20000. If there is a use case that requires support for more ISE agents, please contact Secure Workload support.

Connectors for Inventory Enrichment

Connectors for inventory enrichment provides additional meta-data and context about the inventories (IP addresses) monitored by Secure Workload.

| Connector | Description | Deployed on Virtual Appliance |
|------------|--|-------------------------------|
| ServiceNow | Collect endpoint information from ServiceNow instance and enrich the inventory with ServiceNow attributes. | Secure Workload Edge |
| See also: | Cloud Connectors | — |

For more information about required virtual appliances, see [Virtual Appliances for Connectors](#).

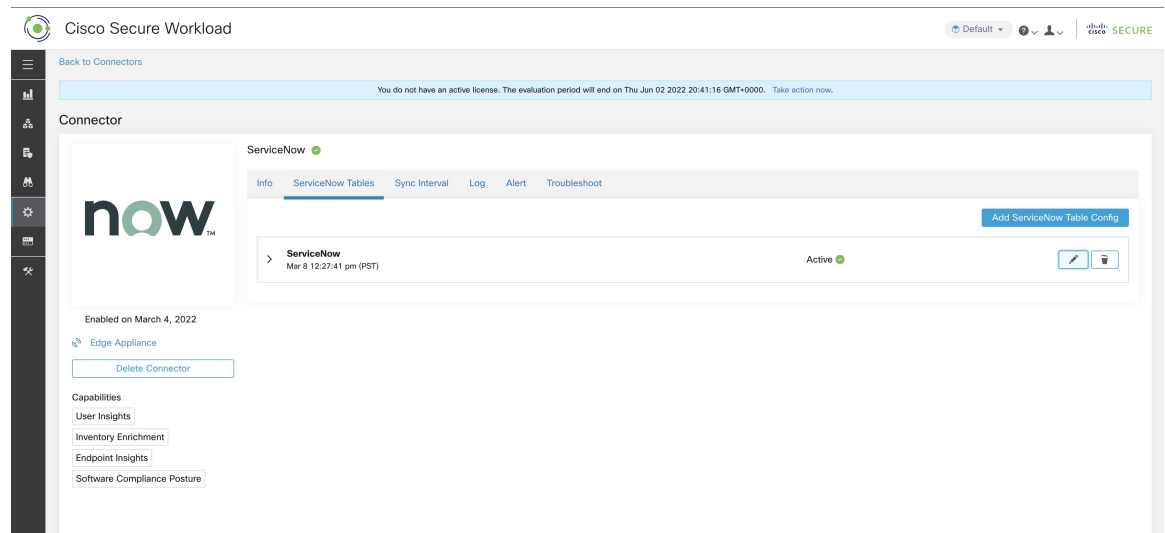
ServiceNow Connector

ServiceNow connector connects with [ServiceNow Instance](#) to get all the ServiceNow CMDB related labels for the endpoints in ServiceNow inventory. Using this solution, we can get enriched metadata for the endpoints in Cisco Secure Workload.

ServiceNow connector does the following high-level functions.

1. Update ServiceNow metadata in Secure Workload's inventory for these endpoints.
2. Periodically take snapshot and update the labels on these endpoints.

Figure 62: ServiceNow connector



How to Configure the ServiceNow Connector

For information about required virtual appliances, see [Virtual Appliances for Connectors](#). The following configurations are allowed on the connector.

- *ServiceNow Tables*: ServiceNow Tables configures the ServiceNow instance with its credentials, and the information about ServiceNow tables to fetch the data from.
- *Scripted REST api*: [ServiceNow scripted REST API](#) tables can be configured similar to ServiceNow tables.

- *Sync Interval*: Sync Interval configuration allows to make change the periodicity at which Secure Workload should query ServiceNow instance for updated data.
- *Log*: For more information, see [Log Configuration](#).

ServiceNow Instance Configuration

Figure 63: ServiceNow instance config

The screenshot shows the Cisco Secure Workload interface for configuring a ServiceNow connector. The main configuration area is titled 'ServiceNow' and is currently active. It displays the following configuration details:

| | |
|----------------------------|---|
| Username | ***** |
| Password | ***** |
| Instance URL | https://ven02560.service-now.com/ |
| Include Scripted APIs | <input checked="" type="checkbox"/> enabled |
| Table to query | /api/cacl/linux_servers_test/linux_servers_test |
| Table IP Address Attribute | ip_address |
| Table Attributes to Fetch | host_name |

Additional information visible in the interface includes: 'Enabled on March 4, 2022', 'Edge Appliance' status, and a list of capabilities such as 'User Insights', 'Inventory Enrichment', 'Endpoint Insights', and 'Software Compliance Posture'.

You will need the following items to successfully configure a ServiceNow instance.

1. ServiceNow username
2. ServiceNow password
3. ServiceNow Instance URL
4. Include Scripted APIs

Subsequently, Secure Workload performs a discovery of all the tables from the ServiceNow Instance and Scripted REST API's (only if Include Scripted APIs check box is enabled). It presents user with the list of tables to choose from, once a user selects table, Secure Workload fetches all the list of attributes from that table for the user to select. User has to chose the ip_address attribute from the table as the key. Subsequently, user can chose upto 10 unique attributes from the table. See the following figures for each step.



Note ServiceNow Connector can only support integrating with tables having **IP Address** field.



Note To integrate with ServiceNow Scripted REST API's you need to enable the Scripted APIs check box, which would give you a similar workflow to any other table.



Note For Scripted REST API's to integrate with ServiceNow Connector, they cannot have path parameters. Also, they need to support **sysparm_limit**, **sysparm_fields** and **sysparm_offset** as query parameters.



Note The ServiceNow user roles need to include **cmdb_read** for tables and **web_service_admin** for Scripted REST API's to integrate with Cisco Secure Workload.

Figure 64: ServiceNow instance config first step

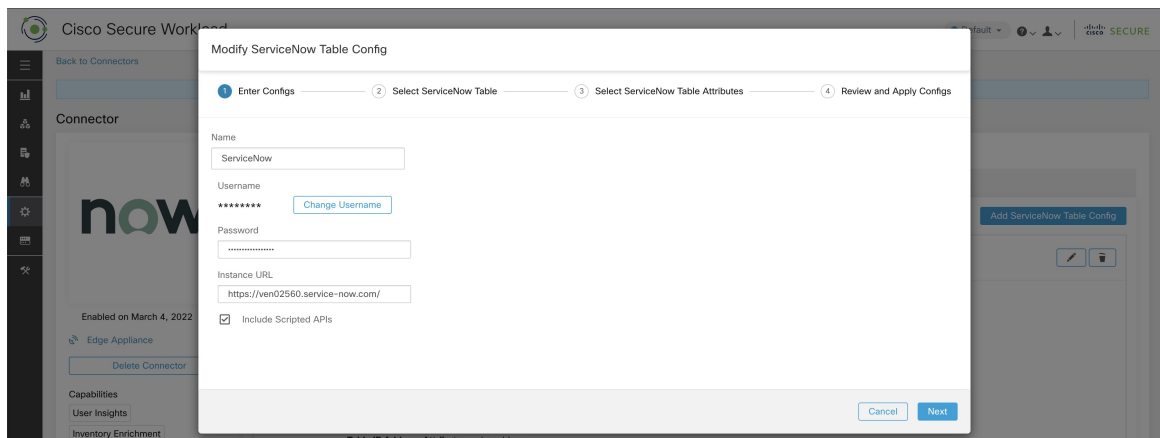


Figure 65: Secure Workload Fetches the Table Info from ServiceNow Instance

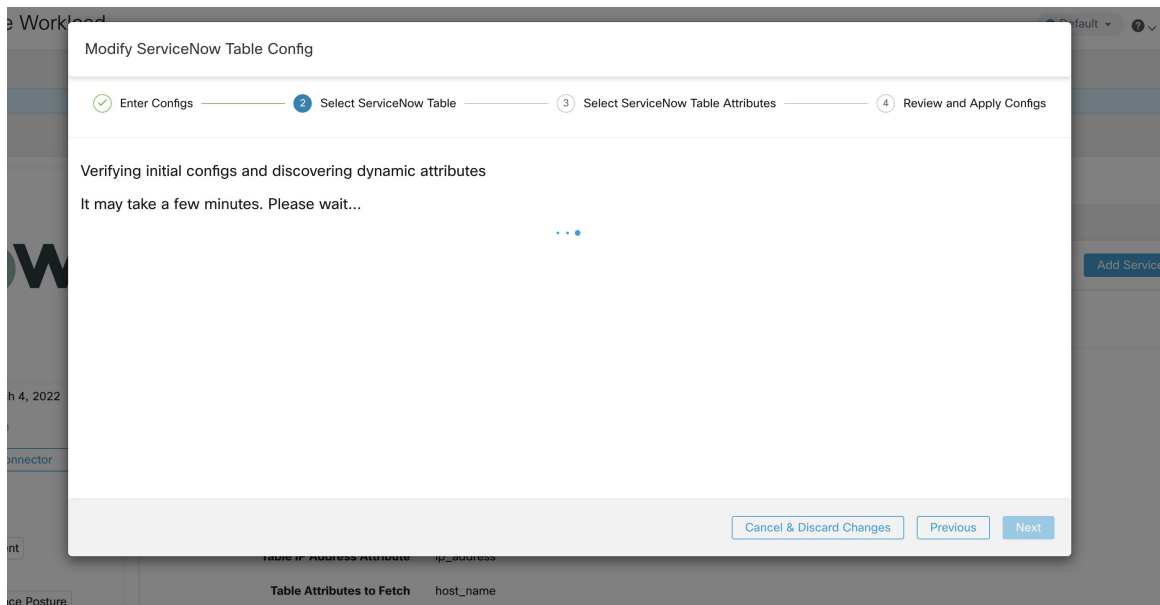


Figure 66: Secure Workload presents the list of tables

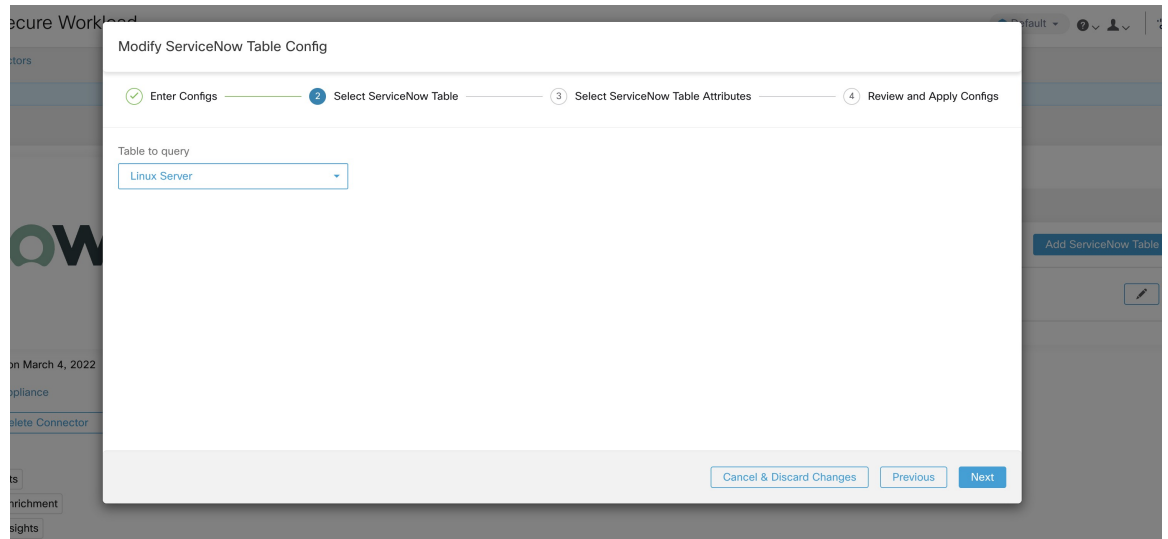


Figure 67: User selects the ip_address attribute, and other attribute in the table

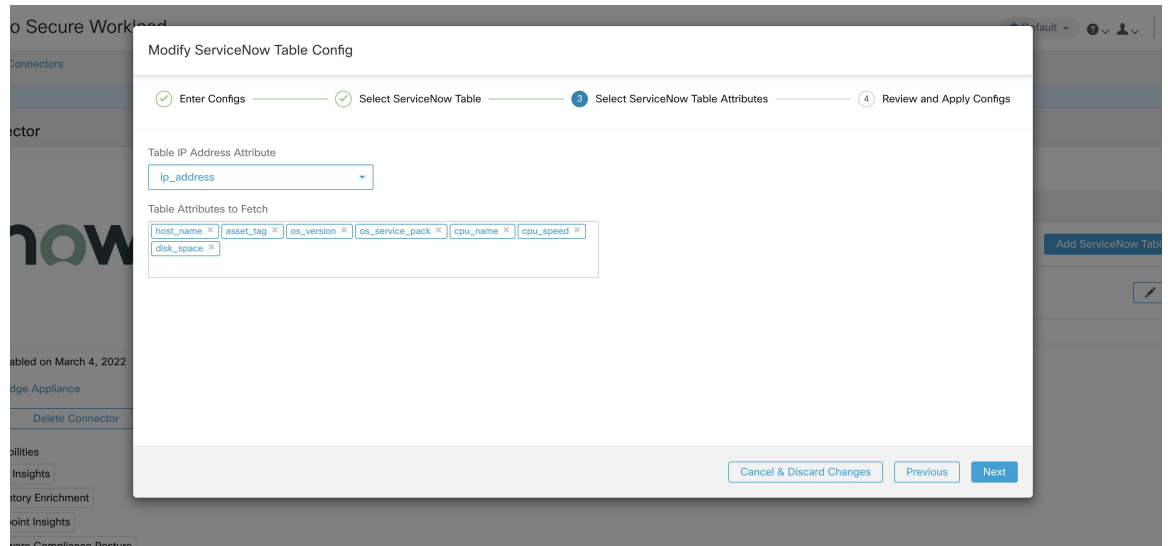


Figure 68: User finalizes the ServiceNow config

Processing ServiceNow records

Based on the instance url you gives in configuration, ServiceNow connector connects to ServiceNow Instance. ServiceNow Instance uses HTTP calls using `https://{Instance URL}/api/now/doc/table/schema`, to obtain the initial table schema from the ServiceNow Table API. Based on the configured Tables, it queries those tables to fetch the ServiceNow labels/metadata. Secure Workload annotates the ServiceNow labels to IP addresses in its inventory. ServiceNow connector periodically fetches new labels and updates Secure Workload inventory.



Note Secure Workload fetches records from ServiceNow tables periodically. This is configurable under SyncInterval tab in the ServiceNow connector. The default sync interval is 60 minutes. For cases where integrating with ServiceNow table with large number of entries, this sync interval should be set to a higher value.



Note Secure Workload will delete any entry not seen for 10 continuous sync intervals. In case the connection to ServiceNow instance is down for that long that could result in cleaning up of all labels for that instance.

Sync Interval Configuration

1. Secure Workload ServiceNow connector provides a way to configure the frequency of sync between Secure Workload and ServiceNow instance. By default the sync interval is set to 60 minutes, but it can be changed under the sync interval configuration as **Data fetch frequency**.
2. For detecting deletion of a record, Secure Workload ServiceNow connector relies on syncs from ServiceNow instances. If an entry is not seen in 48 consecutive sync intervals, we go ahead and delete the entry. This can be configured under sync interval config as **Delete entry interval**.

3. If any additional parameters are to be passed when calling REST api's for ServiceNow tables, you can configure them as part of *Additional Rest API url params*. This configuration is optional. For example, to get a reference lookup from ServiceNow the following url parameters can be used **sysparm_exclude_reference_link=true&sysparm_display_value=true**

Figure 69: Sync Interval Configuration

The screenshot shows the Cisco Secure Workload interface. At the top, there's a navigation bar with 'Cisco Secure Workload' and 'Tetration' dropdown. Below that, a message states: 'You do not have an active license. The evaluation period will end on Thu Nov 18 2021 11:50:41 GMT+0000. Take action now.' The main content area is titled 'Connector' and shows the configuration for 'ServiceNow'. The 'Sync Interval' tab is selected, showing the following settings:

| Parameter | Value |
|--|--|
| Data fetch frequency (in minutes) | 60 |
| Delete entry interval (in multiple of fetch frequency) | 48 |
| Additional Rest API url params | sysparm_exclude_reference_link=true&sysparm_display_value=true |

Additional details visible in the screenshot include: 'Enabled on August 24, 2021', 'Tetration Edge Appliance', and a 'Delete Connector' button. Below the configuration, there are 'Capabilities' listed: User Insights, Inventory Enrichment, Endpoint Insights, and Software Compliance Posture.

Explore Command to Delete the Labels

In case user wants to cleanup the labels for a particular IP for a given instance immediately, without waiting for delete interval, they can do so using an explore command. Here are the steps to run the command.

1. Finding vrf ID for a Tenant
2. Getting to Explore command UI
3. Running the commands

For TaaS cluster, contact TaaS Operation team to cleanup labels for ServiceNow labels.

Finding VRF ID for a Tenant

Site Admins and **Customer Support** users can access the **Tenant** page under the **Platform** menu in the navigation bar at the left side of the window. This page displays all of the currently configured Tenants and VRFs. For more information, see the Tenants section for more details.

On Tenants page, ID field of `Tenants` table is vrf ID for the Tenant.

Getting to Explore Command UI

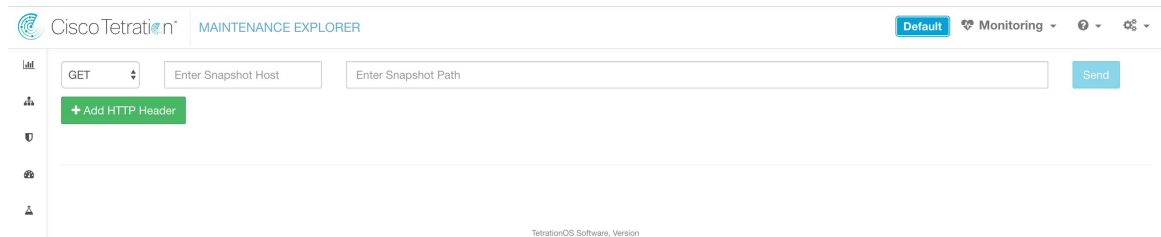
To reach the Maintenance Explorer command interface, choose **Troubleshoot > Maintenance Explorer** from the left navigation bar in the Secure Workload web interface.



Note Customer Support privileges are required to access explore menu. If explore tab does not show up, the account may not have needed permissions.

Click on explore tab in the drop down menu to get to the Maintenance Explorer page.

Figure 70: Maintenance Explorer tab



Running the Commands

- Choose the action as `POST`
- Enter snapshot host as `orchestrator.service.consul`
- Enter snapshot path

To delete the labels for a particular IP for a `servicenow` instance:

```
servicenow_cleanup_annotations?args=<vrf-id> <ip_address> <instance_url> <table_name>
```

- Click Send



Note If after deleting using explore command, we see the record show up in ServiceNow instance, it will be repopulated

Frequently Asked Questions

1. What if ServiceNow CMDB table does not have IP address.

In such case, the recommendation is to create a [View on ServiceNow](#) which will have desired fields from current table along with IP address (potentially coming from a JOIN operation with another table). Once such a view is created, it can be used in place of table name.

2. What if ServiceNow instance requires MFA.

Currently we do not support integrating with ServiceNow instance with MFA.

Limits

| Metric | Limit |
|---|-------|
| Maximum number of ServiceNow instances that can be configured on one ServiceNow connector | 20 |
| Maximum number of attributes that can be fetched from one ServiceNow instance | 10 |
| Maximum number of ServiceNow connectors on one Secure Workload Edge appliance | 1 |
| Maximum number of ServiceNow connectors on one Tenant (rootscope) | 1 |
| Maximum number of ServiceNow connectors on Secure Workload | 150 |

Connectors for Alert Notifications

Connectors for alert notifications enable Secure Workload to publish Secure Workload alerts on various messaging and logging platforms. These connectors run on TAN service on Secure Workload Edge Appliance.

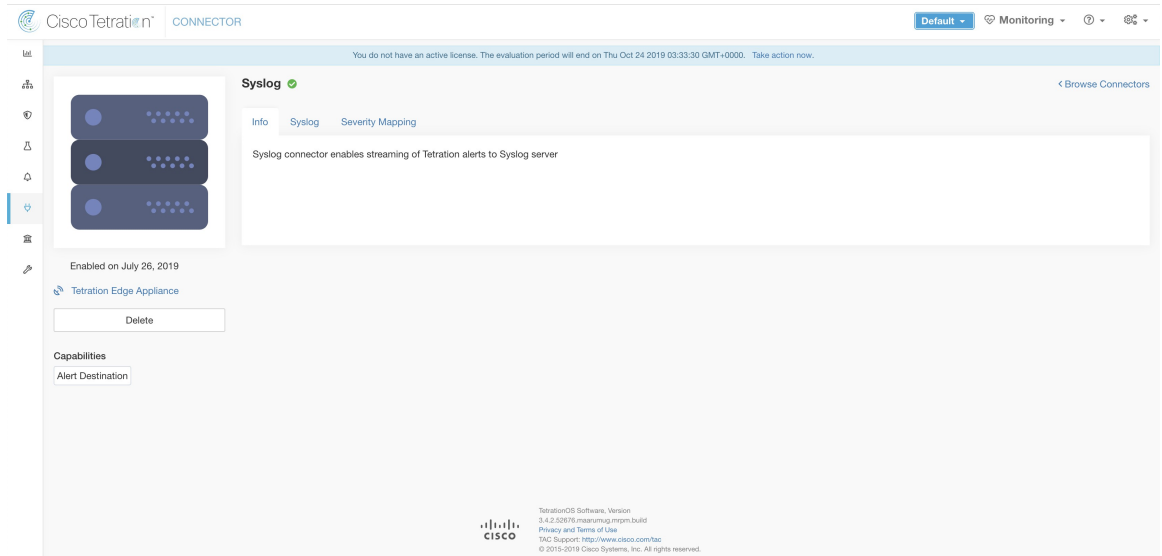
| Connector | Description | Deployed on Virtual Appliance |
|-------------------|--|-------------------------------|
| Syslog | Send Secure Workload alerts to Syslog server. | Secure Workload Edge |
| Email | Send Secure Workload alerts on Email. | Secure Workload Edge |
| Slack | Send Secure Workload alerts on Slack. | Secure Workload Edge |
| Pager Duty | Send Secure Workload alerts on Pager Duty. | Secure Workload Edge |
| Kinesis | Send Secure Workload alerts on Amazon Kinesis. | Secure Workload Edge |

For more information about required virtual appliances, see [Virtual Appliances for Connectors](#).

Syslog Connector

When enabled, TAN service on Cisco Secure Workload Edge appliance can send alerts to Syslog server using configuration.

Figure 71: Syslog connector



The following table explains the configuration details for publishing Secure Workload alerts on Syslog server. For more information, see [Syslog Notifier Configuration](#).

| Parameter Name | Type | Description |
|-----------------------|--------------|---|
| Protocol | drop-down | Protocol to use to connect to server |
| | • <i>UDP</i> | |
| | • <i>TCP</i> | |
| Server Address | string | IP address or hostname of the Syslog server |
| Port | number | Listening port of Syslog server. Default port value is 514. |

Figure 72: Sample configuration for Syslog Connector

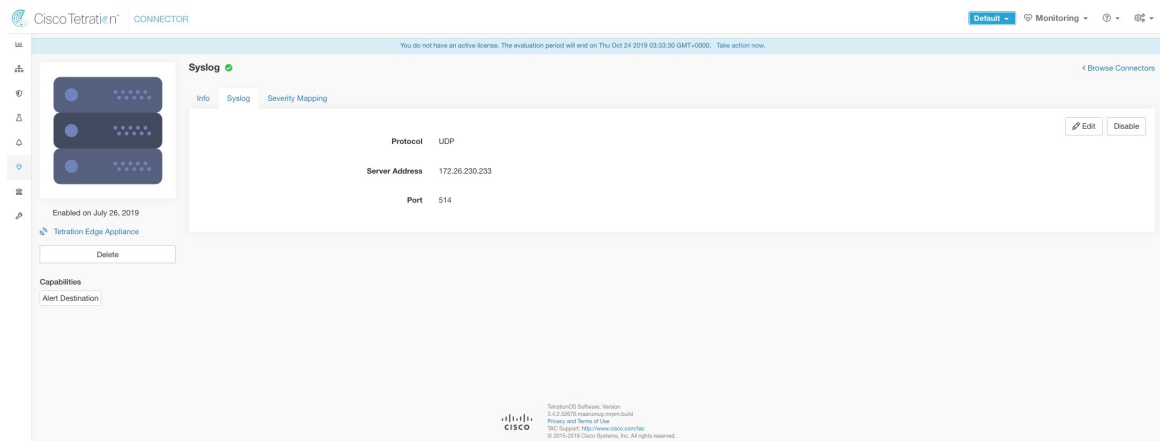


Figure 73: Sample alert

```

Jul 28 21:53:08 tan-5d3b5191f786e9752c77031 Tetration.Alert[4235]: [CRIT] [{"keyId": "cfec2077-508e-30dd-80d7-f2a29701a121", "eventTime": "1564350600000", "alertTime": "1564350820334", "alertText": "Enforcement Annotated Flows contains escaped for u003application_id:5d3b41c14974f01fac220a\u003e", "severity": "CRITICAL", "tenantId": "0", "type": "COMPLIANCE", "alertDetails": {"\provider_scope_ids": ["5d3b744e4974f446636ff13"], "consumer_scope_ids": ["5d3b744e4974f446636ff13"], "protocol": "UDP", "policy_type": "ENFORCED_POLICY", "internal_trigger": {"\datasource": "\compliance", "rules": {"\field": "\policy_violations", "type": "contains", "value": "\escaped"}, "label": "\Alert Trigger"}, "consumer_scope_names": ["Default"], "provider_scope_names": ["Default"], "time_range": ["1564350600000", "1564350799999"], "policy_category": "\ESCAPED"}, "\provider_port": "0", "application_id": "\5d3b41c14974f01fac220a", "\escaped_count": "2"}, "\rootScopeId": "\5d3b744e4974f446636ff13", "alertConfId": "\5d3b41fb9d1577e0895eb08"}
Jul 28 21:53:08 tan-5d3b5191f786e9752c77031 Tetration.Alert[4235]: [CRIT] [{"keyId": "4e497664-cab7-3e0b-9e68-626cc5549e", "eventTime": "1564350600000", "alertTime": "1564350820334", "alertText": "Enforcement Annotated Flows contains escaped for u003application_id:5d3b41c14974f01fac220a\u003e", "severity": "CRITICAL", "tenantId": "0", "type": "COMPLIANCE", "alertDetails": {"\provider_scope_ids": ["5d3b744e4974f446636ff13"], "consumer_scope_ids": ["5d3b744e4974f446636ff13"], "protocol": "UDP", "policy_type": "ENFORCED_POLICY", "internal_trigger": {"\datasource": "\compliance", "rules": {"\field": "\policy_violations", "type": "contains", "value": "\escaped"}, "label": "\Alert Trigger"}, "consumer_scope_names": ["Default"], "provider_scope_names": ["Default"], "time_range": ["1564350600000", "1564350799999"], "policy_category": "\ESCAPED"}, "\provider_port": "25", "application_id": "\5d3b41c14974f01fac220a", "\escaped_count": "2"}, "\rootScopeId": "\5d3b744e4974f446636ff13", "alertConfId": "\5d3b41fb9d1577e0895eb08"}
Jul 28 21:54:22 tan-5d3b5191f786e9752c77031 Tetration.Alert[4235]: [CRIT] [{"keyId": "3ba07063-8065-36ed-9792-256af68180c0", "eventTime": "1564350720000", "alertTime": "1564350900801", "alertText": "Enforcement Annotated Flows contains escaped for u003application_id:5d3b41c14974f01fac220a\u003e", "severity": "CRITICAL", "tenantId": "0", "type": "COMPLIANCE", "alertDetails": {"\provider_scope_ids": ["5d3b744e4974f446636ff13"], "consumer_scope_ids": ["5d3b744e4974f446636ff13"], "protocol": "TCP", "policy_type": "ENFORCED_POLICY", "internal_trigger": {"\datasource": "\compliance", "rules": {"\field": "\policy_violations", "type": "contains", "value": "\escaped"}, "label": "\Alert Trigger"}, "consumer_scope_names": ["Default"], "provider_scope_names": ["Default"], "time_range": ["1564350720000", "1564350799999"], "policy_category": "\ESCAPED"}, "\provider_port": "443", "application_id": "\5d3b41c14974f01fac220a", "\escaped_count": "8"}, "\rootScopeId": "\5d3b744e4974f446636ff13", "alertConfId": "\5d3b41fb9d1577e0895eb08"}
Jul 28 21:54:22 tan-5d3b5191f786e9752c77031 Tetration.Alert[4235]: [DEBUG] [{"keyId": "4c080007-263f-3253-afba-b9660c0d059b", "eventTime": "1564350720000", "alertTime": "1564350900801", "alertText": "Enforcement Rejected Flows u003e-1 for u003application_id:5d3b41c14974f01fac220a\u003e", "severity": "LOW", "tenantId": "0", "type": "COMPLIANCE", "alertDetails": {"\provider_scope_ids": ["5d3b744e4974f446636ff13"], "consumer_scope_ids": ["5d3b744e4974f446636ff13"], "protocol": "TCP", "policy_type": "ENFORCED_POLICY", "internal_trigger": {"\datasource": "\compliance", "rules": {"\field": "\rejected_count", "type": "\gt", "value": "-1"}, "label": "\Alert Trigger"}, "consumer_scope_names": ["Default"], "provider_scope_names": ["Default"], "time_range": ["1564350720000", "1564350799999"], "policy_category": "\ESCAPED"}, "\provider_port": "443", "application_id": "\5d3b41c14974f01fac220a", "\rejected_count": "0"}, "\rootScopeId": "\5d3b744e4974f446636ff13", "alertConfId": "\5d3b5234e9d157483207125e"}
Jul 28 21:54:22 tan-5d3b5191f786e9752c77031 Tetration.Alert[4235]: [DEBUG] [{"keyId": "d0446cc-9c2d-34e5-885b-79b738648478", "eventTime": "1564350720000", "alertTime": "1564350900801", "alertText": "Enforcement Rejected Flows u003e-1 for u003application_id:5d3b41c14974f01fac220a\u003e", "severity": "LOW", "tenantId": "0", "type": "COMPLIANCE", "alertDetails": {"\provider_scope_ids": ["5d3b744e4974f446636ff13"], "consumer_scope_ids": ["5d3b744e4974f446636ff13"], "protocol": "UDP", "policy_type": "ENFORCED_POLICY", "internal_trigger": {"\datasource": "\compliance", "rules": {"\field": "\rejected_count", "type": "\gt", "value": "-1"}, "label": "\Alert Trigger"}, "consumer_scope_names": ["Default"], "provider_scope_names": ["Default"], "time_range": ["1564350720000", "1564350799999"], "policy_category": "\ESCAPED"}, "\provider_port": "53", "application_id": "\5d3b41c14974f01fac220a", "\rejected_count": "0"}, "\rootScopeId": "\5d3b744e4974f446636ff13", "alertConfId": "\5d3b5234e9d157483207125e"}
Jul 28 21:54:22 tan-5d3b5191f786e9752c77031 Tetration.Alert[4235]: [DEBUG] [{"keyId": "d0a322b-99db-380b-8363-41066b75f911", "eventTime": "1564350720000", "alertTime": "1564350900801", "alertText": "Enforcement Rejected Flows u003e-1 for u003application_id:5d3b41c14974f01fac220a\u003e", "severity": "LOW", "tenantId": "0", "type": "COMPLIANCE", "alertDetails": {"\provider_scope_ids": ["5d3b744e4974f446636ff13"], "consumer_scope_ids": ["5d3b744e4974f446636ff13"], "protocol": "UDP", "policy_type": "ENFORCED_POLICY", "internal_trigger": {"\datasource": "\compliance", "rules": {"\field": "\rejected_count", "type": "\gt", "value": "-1"}, "label": "\Alert Trigger"}, "consumer_scope_names": ["Default"], "provider_scope_names": ["Default"], "time_range": ["1564350720000", "1564350799999"], "policy_category": "\ESCAPED"}, "\provider_port": "123", "application_id": "\5d3b41c14974f01fac220a", "\rejected_count": "0"}, "\rootScopeId": "\5d3b744e4974f446636ff13", "alertConfId": "\5d3b5234e9d157483207125e"}
Jul 28 21:54:22 tan-5d3b5191f786e9752c77031 Tetration.Alert[4235]: [CRIT] [{"keyId": "cfec2077-508e-30dd-80d7-f2a29701a121", "eventTime": "1564350720000", "alertTime": "1564350900801", "alertText": "Enforcement Annotated Flows contains escaped for u003application_id:5d3b41c14974f01fac220a\u003e", "severity": "CRITICAL", "tenantId": "0", "type": "COMPLIANCE", "alertDetails": {"\provider_scope_ids": ["5d3b744e4974f446636ff13"], "consumer_scope_ids": ["5d3b744e4974f446636ff13"], "protocol": "UDP", "policy_type": "ENFORCED_POLICY", "internal_trigger": {"\datasource": "\compliance", "rules": {"\field": "\policy_violations", "type": "contains", "value": "\escaped"}, "label": "\Alert Trigger"}, "consumer_scope_names": ["Default"], "provider_scope_names": ["Default"], "time_range": ["1564350720000", "1564350799999"], "policy_category": "\ESCAPED"}, "\provider_port": "0", "application_id": "\5d3b41c14974f01fac220a", "\escaped_count": "2"}, "\rootScopeId": "\5d3b744e4974f446636ff13", "alertConfId": "\5d3b41fb9d1577e0895eb08"}
Jul 28 21:54:22 tan-5d3b5191f786e9752c77031 Tetration.Alert[4235]: [CRIT] [{"keyId": "ad6e167-c2d9-336f-b465-b6d60b46f6d", "eventTime": "1564350720000", "alertTime": "1564350900801", "alertText": "Enforcement Annotated Flows contains escaped for u003application_id:5d3b41c14974f01fac220a\u003e", "severity": "CRITICAL", "tenantId": "0", "type": "COMPLIANCE", "alertDetails": {"\provider_scope_ids": ["5d3b744e4974f446636ff13"], "consumer_scope_ids": ["5d3b744e4974f446636ff13"], "protocol": "UDP", "policy_type": "ENFORCED_POLICY", "internal_trigger": {"\datasource": "\compliance", "rules": {"\field": "\policy_violations", "type": "contains", "value": "\escaped"}, "label": "\Alert Trigger"}, "consumer_scope_names": ["Default"], "provider_scope_names": ["Default"], "time_range": ["1564350720000", "1564350799999"], "policy_category": "\ESCAPED"}, "\provider_port": "53", "application_id": "\5d3b41c14974f01fac220a", "\escaped_count": "2"}, "\rootScopeId": "\5d3b744e4974f446636ff13", "alertConfId": "\5d3b41fb9d1577e0895eb08"}
    
```

Syslog Severity Mapping

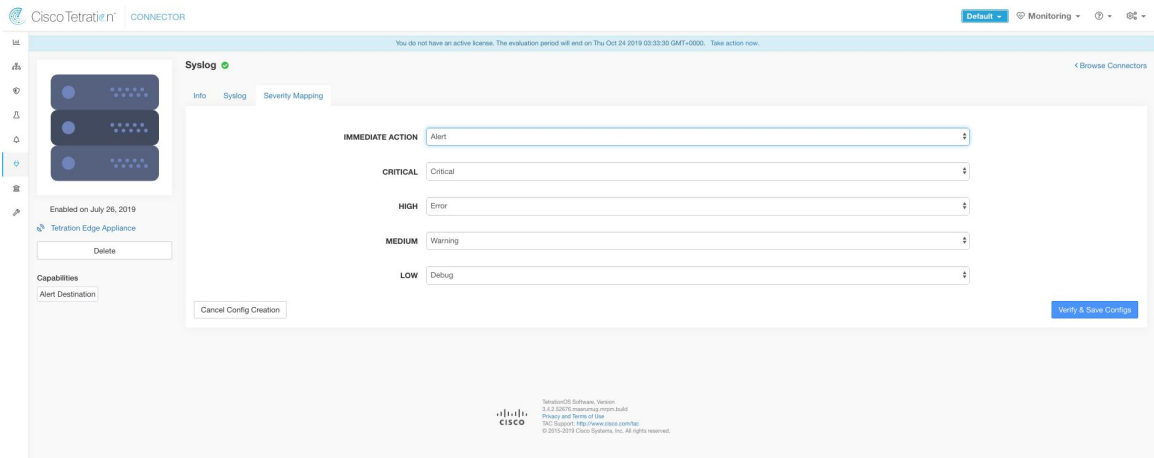
The following table shows the default severity mapping for Secure Workload alerts on Syslog.

| Secure Workload Alerts Severity | Syslog Severity |
|---------------------------------|-----------------|
| LOW | LOG_DEBUG |
| MEDIUM | LOG_WARNING |
| HIGH | LOG_ERR |
| CRITICAL | LOG_CRIT |
| IMMEDIATE ACTION | LOG_EMERG |

This setting can be modified using **Severity Mapping** configuration under Syslog Connector. You can choose any corresponding Syslog priority for each Secure Workload Alert Severity and change the Severity Mapping. For more information, see [Syslog Severity Mapping Configuration](#).

| Parameter Name | Dropdown of mappings |
|------------------|------------------------|
| IMMEDIATE_ACTION | • <i>Emergency</i> |
| CRITICAL | • <i>Alert</i> |
| HIGH | • <i>Critical</i> |
| MEDIUM | • <i>Error</i> |
| LOW | • <i>Warning</i> |
| | • <i>Notice</i> |
| | • <i>Informational</i> |
| | • <i>Debug</i> |

Figure 74: Sample config for Syslog Severity Mapping.



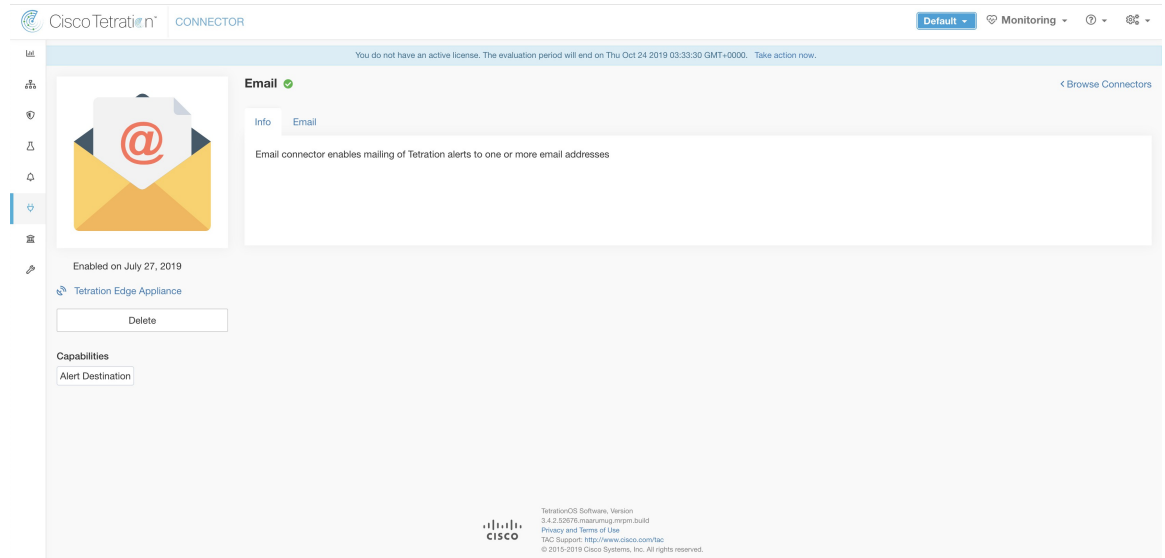
Limits

| Metric | Limit |
|---|-------|
| Maximum number of Syslog connectors on one Secure Workload Edge appliance | 1 |
| Maximum number of Syslog connectors on one Tenant (rootscope) | 1 |
| Maximum number of Syslog connectors on Secure Workload | 150 |

Email Connector

When enabled, TAN service on Secure Workload Edge Appliance can send alerts to given configuration.

Figure 75: Email connector



The following table explains the configuration details for publishing Secure Workload alerts on Email. For more information, see [Email Notifier Configuration](#).

Table 16: Email Notifier Configuration for more details

| Parameter Name | Type | Description |
|--------------------|----------|---|
| SMTP Username | string | SMTP server username. This parameter is optional. |
| SMTP Password | string | SMTP server password for the user (if given). This parameter is optional. |
| SMTP Server | string | IP address or hostname of the SMTP server |
| SMTP Port | number | Listening port of SMTP server. Default value is 587. |
| Secure Connection | checkbox | Should SSL be used for SMTP server connection? |
| From Email Address | string | Email address to use for sending alerts |
| Default Recipients | string | Comma separated list of recipient email addresses |

Figure 76: Sample configuration for Email Connector

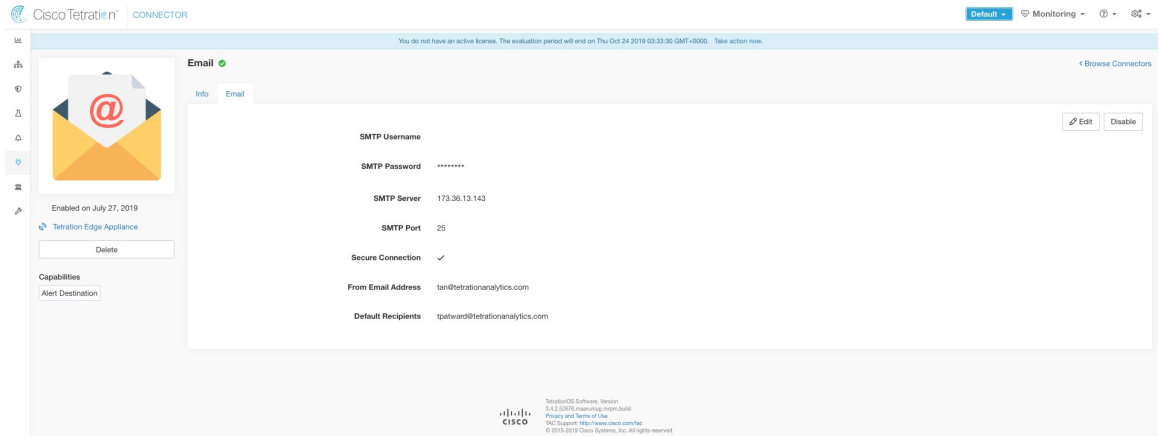
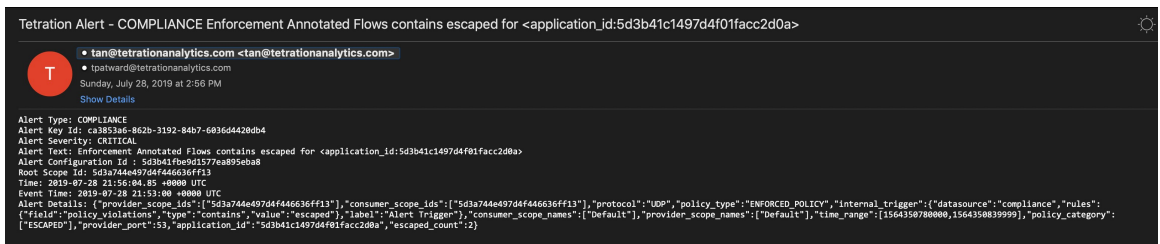


Figure 77: Sample alert



Note

- SMTP username/password is optional. If no username is provided, we try to connect to SMTP server without any auth.
- If secure connection box is not checked, we will send alerts notification over non-secure connection.
- Default Recipients list is used to send alert notifications. This can be overridden per alert if required in Alert configuration.

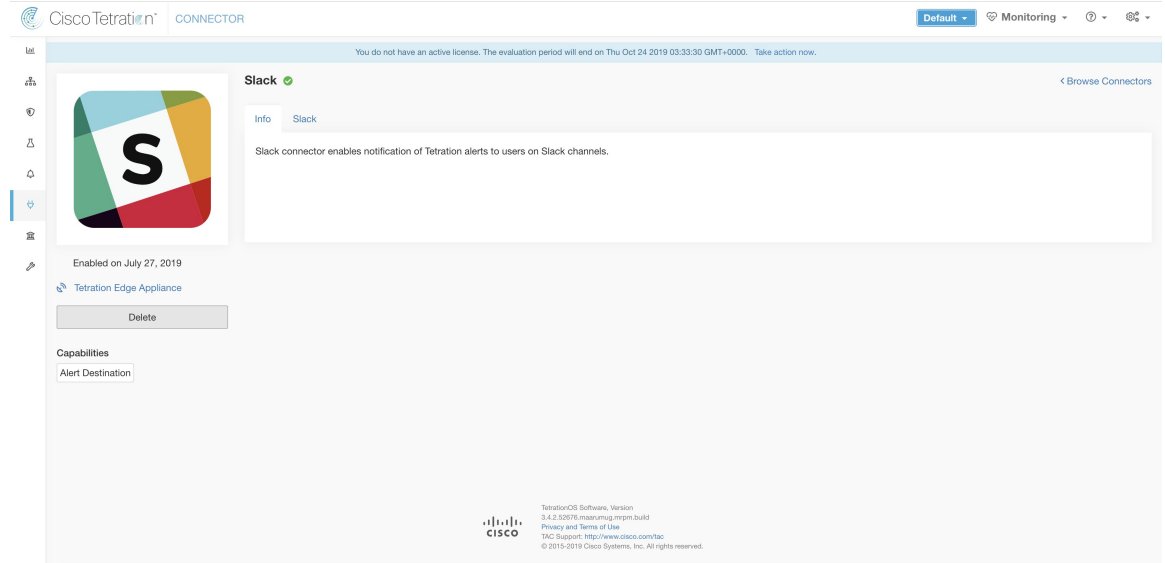
Limits

| Metric | Limit |
|--|-------|
| Maximum number of Email connectors on one Secure Workload Edge appliance | 1 |
| Maximum number of Email connectors on one Tenant (rootscope) | 1 |
| Maximum number of Email connectors on Secure Workload | 150 |

Slack Connector

When enabled, TAN service on Secure Workload Edge appliance can send alerts to Slack using configuration.

Figure 78: Slack connector



The following table explains the configuration details for publishing Secure Workload alerts on Slack. For more information, see [Slack Notifier Configuration](#).

| Parameter Name | Type | Description |
|-------------------|--------|---|
| Slack Webhook URL | string | Slack webhook on which Secure Workload alerts should be published |



Note • To generate slack webhook go [here](#).

Figure 79: Sample configuration for Slack Connector

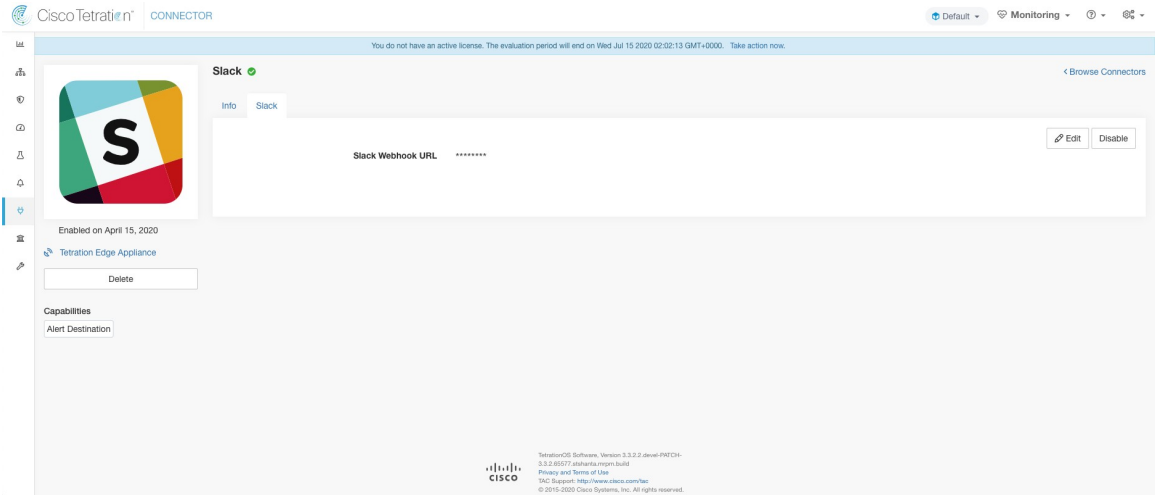
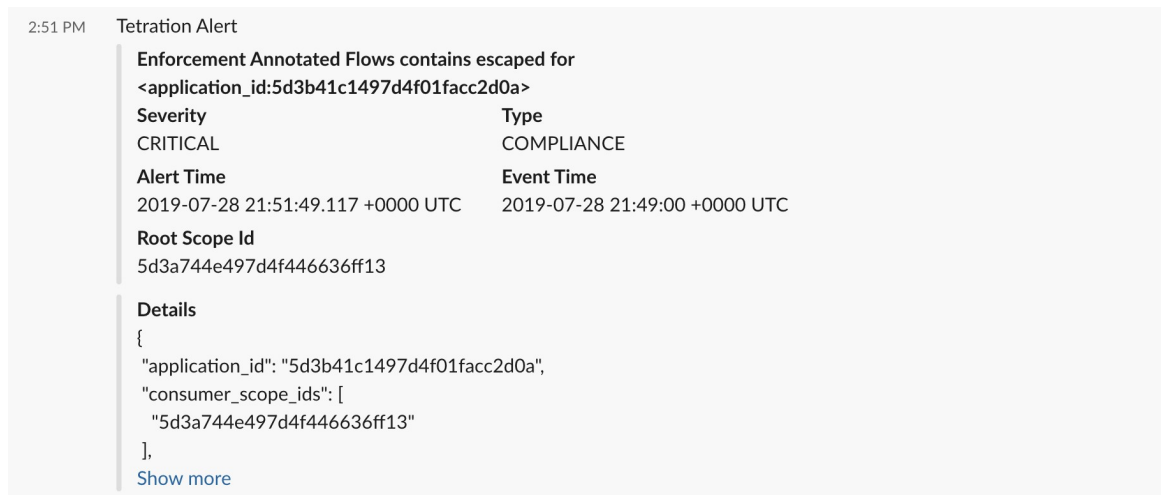


Figure 80: Sample alert



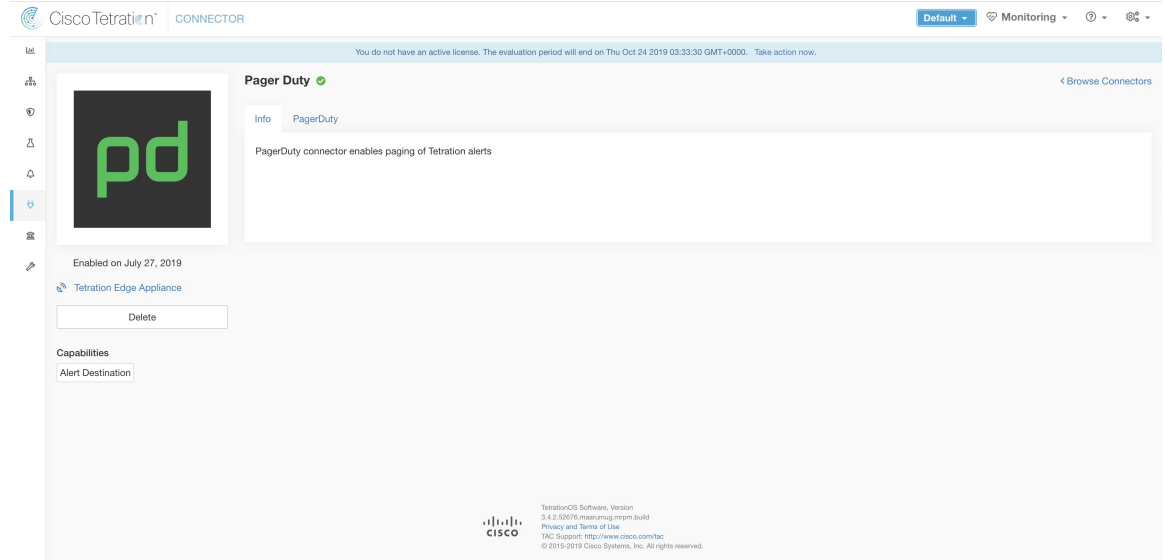
Limits

| Metric | Limit |
|--|-------|
| Maximum number of Slack connectors on one Secure Workload Edge appliance | 1 |
| Maximum number of Slack connectors on one Tenant (rootscope) | 1 |
| Maximum number of Slack connectors on Secure Workload | 150 |

PagerDuty Connector

When enabled, TAN service on Secure Workload Edge appliance can send alerts to PagerDuty using configuration.

Figure 81: PagerDuty connector



The following table explains the configuration details for publishing Secure Workload alerts on PagerDuty. For more information, see [PagerDuty Notifier Configuration](#).

| Parameter Name | Type | Description |
|-----------------------|--------|--|
| PagerDuty Service Key | string | PagerDuty service key for pushing Secure Workload alerts on PagerDuty. |

Figure 82: Sample configuration for PagerDuty Connector

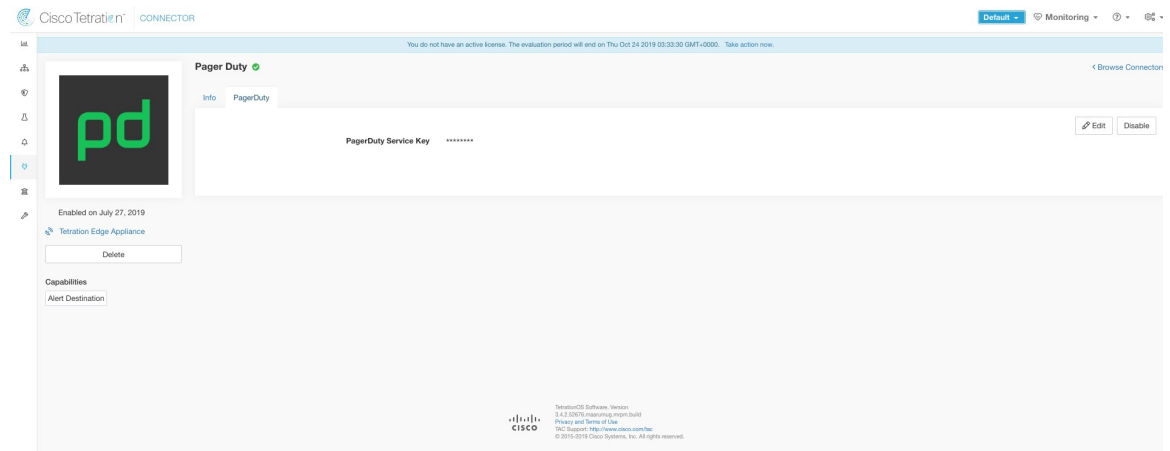
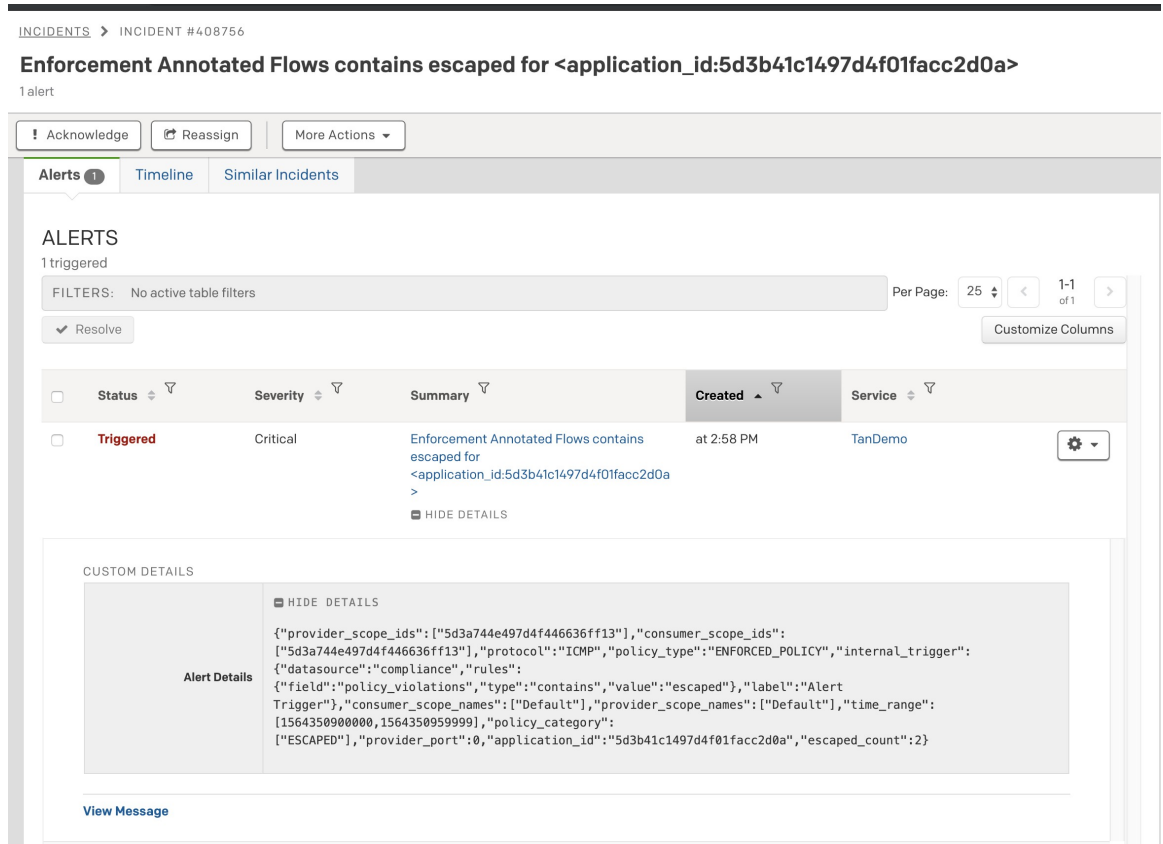


Figure 83: Sample alert



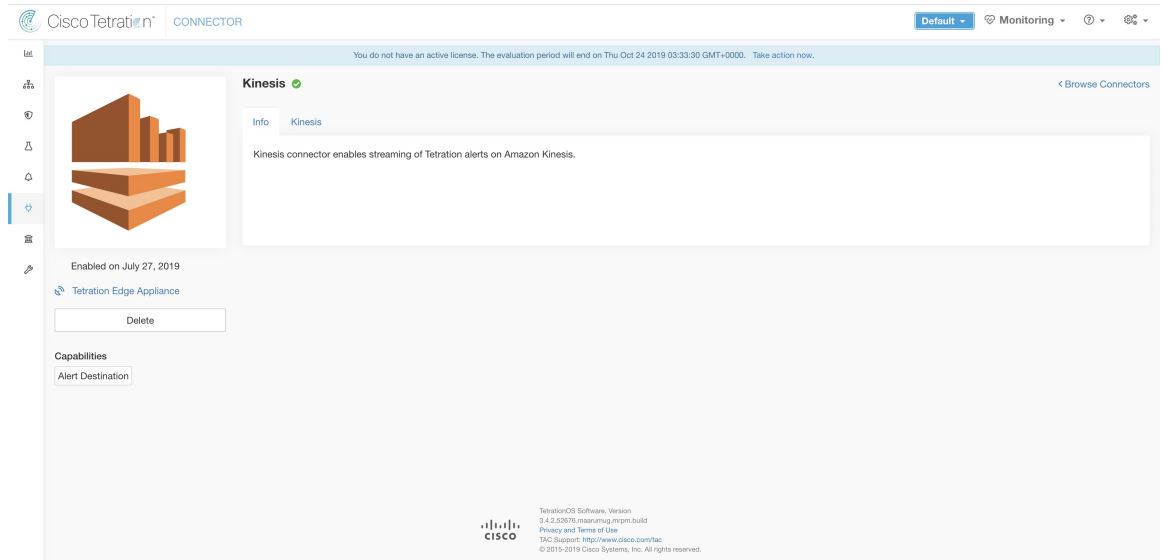
Limits

| Metric | Limit |
|--|-------|
| Maximum number of PagerDuty connectors on one Secure Workload Edge appliance | 1 |
| Maximum number of PagerDuty connectors on one Tenant (rootscope) | 1 |
| Maximum number of PagerDuty connectors on Secure Workload | 150 |

Kinesis Connector

When enabled, TAN service on Secure Workload Edge appliance can send alerts using configuration.

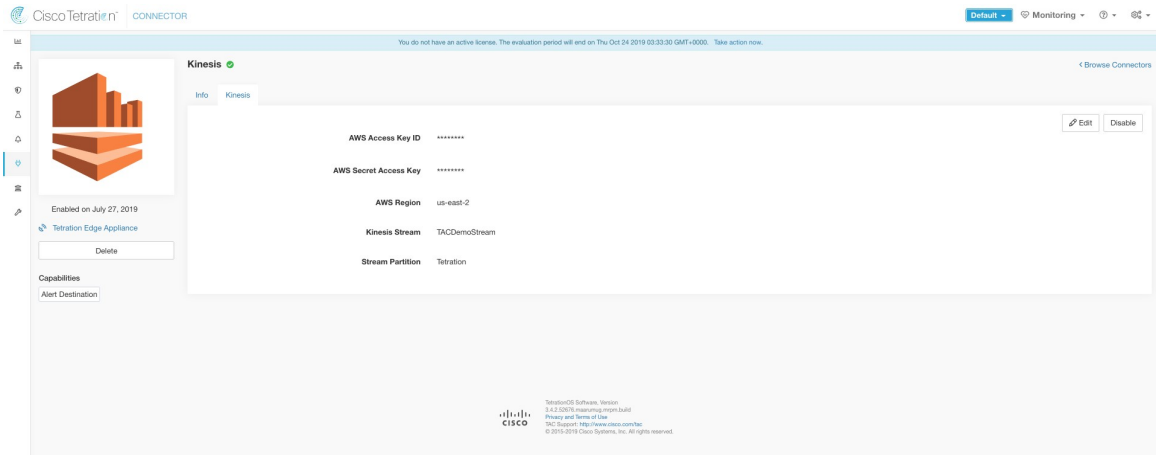
Figure 84: Kinesis connector



The following table explains the configuration details for publishing Secure Workload alerts on Amazon Kinesis. For more information, see [Kinesis Notifier Configuration](#).

| Parameter Name | Type | Description |
|------------------------------|-------------------------|---|
| AWS Access Key ID | string | AWS access key ID to communicate with AWS |
| AWS Secret Access Key | string | AWS secret access key to communicate with AWS |
| AWS Region | dropdown of AWS regions | Name of the AWS region where Kinesis stream is configured |
| Kinesis Stream | string | Name of the Kinesis stream |
| Stream Partition | string | Partition Name of the stream |

Figure 85: Sample configuration for Kinesis Connector.



Limits

| Metric | Limit |
|--|-------|
| Maximum number of Kinesis connectors on one Secure Workload Edge appliance | 1 |
| Maximum number of Kinesis connectors on one Tenant (rootscope) | 1 |
| Maximum number of Kinesis connectors on Secure Workload | 150 |

Cloud Connectors

You can use a cloud connector for Secure Workload features on cloud-based workloads.

Cloud connectors do not require a virtual appliance.

| Connector | Supported Features | Deployed on Virtual Appliance |
|------------|--|-------------------------------|
| AWS | For Amazon Web Services VPCs: <ul style="list-style-type: none"> • Collect metadata (labels) • Collect flow logs • Enforce segmentation policies From Elastic Kubernetes Service (EKS) clusters: <ul style="list-style-type: none"> • Collect metadata | N/A |

| Connector | Supported Features | Deployed on Virtual Appliance |
|--------------|---|-------------------------------|
| Azure | For Azure VNets: <ul style="list-style-type: none"> • Collect metadata (labels) • Collect flow logs • Enforce segmentation policies From Azure Kubernetes Service (AKS) clusters: <ul style="list-style-type: none"> • Collect metadata | N/A |
| GCP | For Google Cloud Platform VPCs: <ul style="list-style-type: none"> • Collect metadata (labels) • Collect flow logs • Enforce segmentation policies From Google Kubernetes Engine (GKE) clusters: <ul style="list-style-type: none"> • Collect metadata (labels) | N/A |

AWS Connector

Amazon Web Services (AWS) connector connects with [AWS](#) to perform the following high-level functions:

- Automated ingestion of inventory (and its labels) live from an AWS Virtual Private Cloud (VPC)**
 AWS allows you to assign metadata to your resources in the form of tags. Secure Workload query the tags for these resources which can then be used for inventory and traffic flow data visualization, and policy definition. This capability keeps the resource tag mapping updated by constantly synchronizing this data.

 The tags from workloads and network interfaces of an AWS VPC are ingested. If you configure both workloads and network interfaces, Secure Workload merges and displays the tags. For more information, see [Labels Generated by Cloud Connectors, on page 336](#).
- Ingestion of VPC-level flow logs** If you have set up VPC flow logs in AWS for monitoring purposes, Secure Workload can ingest flow log information by reading the corresponding S3 bucket. You can use this telemetry for visualization and segmentation policy generation.
- Segmentation** When the segmentation option is enabled, Secure Workload programs security policies using AWS native Security Groups. When enforcement is enabled for a VPC, relevant policies are automatically programmed as security groups.
- Automated ingestion of metadata from EKS clusters** When Elastic Kubernetes Services (EKS) is running on AWS, you can choose to gather all node, service, and pod metadata related to all selected Kubernetes clusters.

You can choose which capabilities to enable for each VPC.



Note We don't currently support China Regions.

Requirements and Prerequisites for AWS

For all capabilities: Create a dedicated user in AWS, or identify an existing AWS user for this connector. The connector configuration wizard generates a CloudFormation Template (CFT) that you can use to assign required privileges to this user. Make sure you have permissions in AWS to upload this CFT.

For granting cross AWS account access to the dedicated user, see [\(Optional\) Configure cross AWS account access in AWS, on page 232](#), including required access privileges.

For granting AWS account access using role, refer to role-based access to Secure Workload cluster.

Each VPC can belong to only one AWS connector. A Secure Workload cluster can have multiple AWS connectors. Gather the information described in the tables in [Configure new AWS Connector , on page 236](#).

This connector doesn't require a virtual appliance.

For gathering labels and inventory: No additional prerequisites are required.

For ingesting flow logs: VPC level flow log definitions are required in order to trigger the collection of flow logs.

Only VPC-level flow logs can be ingested.

Flow logs must be published to Amazon Simple Storage Service (S3); Secure Workload cannot collect flow data from Amazon CloudWatch logs.

Secure Workload can ingest flow logs from an S3 bucket associated with any account, if the AWS user account credentials provided during connector creation have access to both the VPC flow logs and the S3 bucket.

The following flow log attributes (in any order) are required in the flow log: Source Address, Destination Address, Source Port, Destination Port, Protocol, Packets, Bytes, Start Time, End Time, Action, TCP Flags, Interface-ID, Log status and Flow Direction. Any other attributes are ignored.

Flow logs must capture both Allowed and Denied traffic.



Note The Secure Workload AWS connector supports VPC flow logs partition on an hourly and daily basis.

For segmentation: Enabling segmentation requires Gather Labels to be enabled.

Back up your existing security groups before enabling segmentation in the connector, as all existing rules are overwritten when you enable segmentation for a VPC.

For more information, see [Best Practices When Enforcing Segmentation Policy for AWS Inventory](#).

For managed Kubernetes services (EKS): If you enable the Kubernetes option, see [Requirements and Prerequisites for EKS](#) in the Managed Kubernetes Services Running on AWS (EKS) section, including required access privileges.

(Optional) Configure cross AWS account access in AWS

If the given user credentials has access to VPCs belonging to other AWS accounts, they will be available for processing as part of the AWS connector.

1. The designated Secure Workload user should have the following AWS access permissions:

1. iam:GetPolicyVersion
2. iam:ListPolicyVersions
3. iam:ListAttachedUserPolicies
4. iam:GetUser
5. servicequotas:ListServiceQuotas

Example AWS policy JSON:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "VisualEditor0",
      "Effect": "Allow",
      "Action": [
        "iam:GetPolicyVersion",
        "iam:ListPolicyVersions",
        "iam:ListAttachedUserPolicies",
        "iam:GetUser",
        "servicequotas:ListServiceQuotas"
      ],
      "Resource": "*"
    }
  ]
}
```

2. Create an AWS IAM role in the desired AWS account of which the designated Secure Workload user is NOT part of.
3. Allow the AWS IAM role to be assumed by the Secure Workload user. This can be done by adding the Secure Workload user ARN to the AWS IAM role trust policy.

Example AWS IAM role trust policy JSON:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": <Secure Workload_user_arn>
      },
      "Action": "sts:AssumeRole",
      "Condition": {}
    }
  ]
}
```

4. Perform the steps 2 and 3 for all the desired AWS accounts which the Secure Workload user does not belong to.
5. Create a customer managed policy (NOT Inline policy) with permission to assume all the created AWS roles from different accounts.



Note In AWS connector, Customer Inline Policy is not supported.

Example Managed policy JSON:

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "VisualEditor0",
      "Effect": "Allow",
      "Action": "sts:AssumeRole",
      "Resource": [<AWS_role_cross_account_1_arn>, <AWS_role_cross_account_2_arn>...]
    }
  ]
}

```

6. [Attach](#) the created customer managed policy to the Secure Workload user.
7. The connector configuration wizard will provide a CloudFormation Template. After uploading the CFT as-is to the designated Secure Workload user, you will edit the template and upload the edited template to the CloudFormation portal to grant the required permissions to the AWS IAM roles. For details, see [Configure new AWS Connector](#) , on page 236.

Authentication Using Roles

User-based authentication requires credential keys. If the credential key is not properly managed, it can cause security threat due to their sensitive nature.

Using the role-based authentication you can configure the AWS account using roles. The connector configuration accepts the role id (ARN) and assume that role to perform specific actions on the customer's account.

Role-based authentication reduces the risk of unauthorized access.

To access the Role-based authentication, follow these steps:

Procedure

-
- Step 1** Click the **Role** tab in the connector configuration page.
 - Step 2** Register the cluster. If the cluster is not registered, it displays a message *"Cluster is not registered to use role credentials"*. *Download the provided payload and contact a customer service representative..*
 - Step 3** From the notification message, click the **download** button and download the payload file.
 - Step 4** You can use the link in the notification message to contact the **TAC** team and raise the ticket and provide the file that you have downloaded.
 - Step 5** When the cluster is registered, the **External Id** and **User ARN** gets auto populated.
- Note** Refresh the page to view the External Id and User ARN.
- Step 6** Use the generated **External Id** and **User ARN** to update the role trust relationship. It enables to assume the role.

The same part of the JSON file:

```

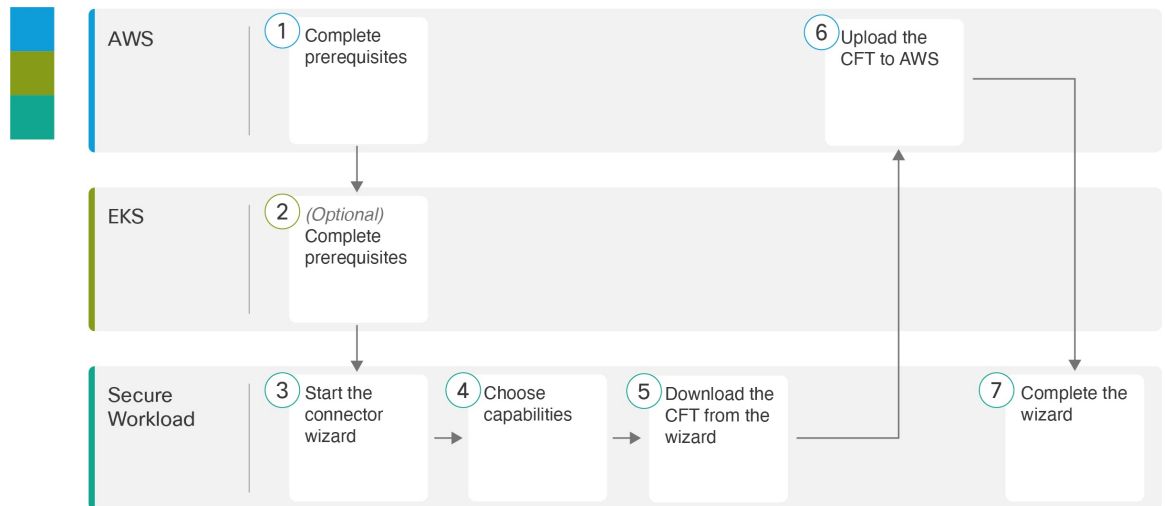
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "Statement1",
      "Effect": "Allow",
      "Principal": {
        "AWS": "<User ARN>"
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringEquals": {
          "sts:ExternalId": "<External Id>"
        }
      }
    }
  ]
}
    
```

Step 7 When the previous step is complete, you can copy the **Role ARN** from the AWS account and paste it in the AWS connector configuration page.

AWS Connector Configuration Overview

The following graphic gives a high-level overview of the connector configuration process. For essential details, see the next topic ([Configure new AWS Connector](#) , on page 236.)

Figure 86: AWS connector configuration overview



(Note that the numbers in the graphic do not correspond to step numbers in the detailed procedure.)

Configure new AWS Connector

Procedure

Step 1 In the navigation pane, choose **Manage > Workloads > Connectors**.

Step 2 Click **AWS Connector**.

Step 3 Click **Generate Template** and choose the desired capabilities.

Based on the capabilities selected, CloudFormation Template (CFT) is generated. Use the generated CFT template in your AWS CloudFormation to create the policy for the User or Role.

To enable segmentation, you must also enable **Gather Labels**.

Step 4 Download the generated CloudFormation Template (CFT). The generated CFT can be used for both the user and role.

This template has the IAM privileges required for the capabilities that you selected in the previous step.

If you enabled the Kubernetes option, you must separately configure permissions for EKS. See [Managed Kubernetes Services Running on AWS \(EKS\), on page 242](#).

Step 5 Upload the CFT to the AWS CloudFormation portal to assign privileges to the user for this connector. Ensure the AWS user have the required privileges before you can continue with the AWS connector configuration.

Note We recommend this task whether or not you are using AWS cross-account access.

You can apply the CFT using either the portal or the CLI. For more information, see:

- **Portal:** [AWS Management Console](#)
- **CLI:** [Creating a stack](#)

When you upload the CFT, AWS requires the following details:

- a. Name of the policy (This can be anything. For example, Secure WorkloadConnector)
- b. Rolename: Name of the AWS IAM role to which you are applying the CFT
- c. List of bucket ARNs And Object ARNs (Default: *)
- d. Username: Name of the AWS user to which you are applying the CFT
- e. List of VPC ARNs (Default: *)

To enter a specific list of VPC ARNs, enter the security group and network interface resources paired with the specific VPC to enable segmentation.

1. `arn:aws:ec2:<region>:<account_id>:security-group/*`
2. `arn:aws:ec2:<region>:<account_id>:network-interface/*`

Sample Code

Example 1

```
{
  "Action": [
    "ec2:RevokeSecurityGroupIngress",
    "ec2:AuthorizeSecurityGroupEgress",
```

```

    "ec2:AuthorizeSecurityGroupIngress",
    "ec2:CreateSecurityGroup",
    "ec2:RevokeSecurityGroupEgress",
    "ec2>DeleteSecurityGroup",
    "ec2:ModifyNetworkInterfaceAttribute",
    "ec2:CreateTags"
  ],
  "Resource": [
    "arn:aws:ec2:us-east-1:123456789:vpc/vpc-abcdef",
    "arn:aws:ec2:us-east-1:123456789:security-group/*",
    "arn:aws:ec2:us-east-1:123456789:network-interface/*"
  ],
  "Effect": "Allow"
},

```

Example 2

```

{
  "Action": [
    "ec2:RevokeSecurityGroupIngress",
    "ec2:AuthorizeSecurityGroupEgress",
    "ec2:AuthorizeSecurityGroupIngress",
    "ec2:CreateSecurityGroup",
    "ec2:RevokeSecurityGroupEgress",
    "ec2>DeleteSecurityGroup",
    "ec2:ModifyNetworkInterfaceAttribute",
    "ec2:CreateTags"
  ],
  "Resource": [
    "arn:aws:ec2:us-east-1:123456789:vpc/vpc-abcdef",
    "arn:aws:ec2:*:*:security-group/*",
    "arn:aws:ec2:*:*:network-interface/*"
  ],
  "Effect": "Allow"
},

```

Step 6 If you are using AWS role based authentication to connect to the Secure Workload Connector, see EKS Roles and Access privileges section.

Step 7 If you are using AWS cross-account access, follow the additional steps:

- a. You can use the same uploaded CFT to give access to role/user. If you have multiple account use the same CFT on each account.
- b. Upload the CFT to the AWS CloudFormation portal of each AWS account where the desired IAM role exists.

You can apply the CFT using either the portal or the CLI, as described in the previous step.

When you upload the CFT, AWS asks for the following:

1. Name of the policy (This can be anything. For example, Secure WorkloadConnector)
2. List of bucket ARNs And Object ARNs (Default: *)
3. Rolename: Name of the AWS IAM role to which you are applying the CFT
4. List of VPC ARNs (Default: *)

Step 8 Click **Getting started guide** (recommended) or **Configure your new connector here** button to configure the connector.

Step 9 Understand and meet the [Requirements and Prerequisites for AWS](#), EKS Roles and Access Privileges and [Best Practices When Enforcing Segmentation Policy for AWS Inventory](#), then click **Get Started**. Or if you are configuring using the **Configure your new connector** button, then click **yes**.

Step 10 Name the connector and enter the description.

Step 11 Configure settings:

You can use either of the one option to connect to AWS account.

a. Credential Keys

b. Roles

| Parameter Name | Attribute | Description |
|-----------------|---------------------|--|
| Credential Keys | Access Key | ACCESS KEY ID associated with the AWS user that has the privileges described in the CFT above. |
| | Secret Key | SECRET KEY associated with the ACCESS KEY ID above. |
| Roles | External Id | It is auto generated unique identifier for granting access to AWS resources. It is used by the user to add trust relationship to the role. |
| | User ARN | It is auto generated unique identifier assigned to an IAM. It is used by the user to add trust relationship to the role. |
| | ARN | A unique identifier assigned to each AWS resource. |
| | HTTP Proxy | (Optional) Proxy required for Secure Workload to reach AWS. |
| | Full Scan Interval | Frequency with which Secure Workload refreshes complete inventory data from AWS. Default and minimum is 3600 seconds. |
| | Delta Scan Interval | Frequency with which Secure Workload fetches incremental changes in inventory data from AWS. Default and minimum is 600 seconds. |

Step 12 Click Next.

- Step 13** The next page displays a **Resource Tree** where the user can expand to view various region and inside the region you can select or unselect the resource check boxes to obtain the list of VPCs and EKS clusters from AWS.
- Step 14** From the list of VPCs (Virtual Networks), choose the VPCs for which you want to enable your selected capabilities.
- Generally, you should enable flow ingestion as soon as possible, so that Secure Workload can begin to collect enough data required to suggest accurate policies.
- Note that since EKS only supports Gather Labels capability, no explicit capability selection has been provided. Selecting an EKS cluster will implicitly enable the supported capability. For each cluster for which you enable this capability, enter the **Assume Role ARN** (The Amazon resource number of the role to assume while connecting to Secure Workload.)
- Enable Segmentation** on VPCs will remove existing Security Group(s) and provides default access to all VPCs.
- Generally, you should not choose **Enable Segmentation** during initial configuration. Later, when you are ready to enforce segmentation policy for specific VPCs, you can edit the connector and enable segmentation for those VPCs. See the Best Practices When Enforcing Segmentation Policy for AWS Inventory.
- Step 15** For the EKS cluster, you can allow AWS IAM role access by providing the Assume Role ARN access id to connect to the AWS connector.
- Step 16** Once your selections are complete, click **Create** and wait a few minutes for the validation check to complete.
-

What to do next

If you have enabled gathering labels, ingesting flow data, and/or segmentation:

- If you enable flow ingestion, it may take up to 25 minutes for flows to begin appearing on the **Investigate > Traffic** page.
- (Optional) For richer flow data and other benefits including visibility into host vulnerabilities (CVEs), install the appropriate agent for your operating system on your VPC-based workloads. For requirements and details, see the agent installation chapter.
- After you have successfully configured the AWS connector to gather labels and ingest flows, follow the standard process for building segmentation policies. For example: Allow Secure Workload to gather sufficient flow data to generate reliable policies; define or modify scopes (typically one for each VPC); create a workspace for each scope; automatically discover policies based on your flow data, and/or manually create policies; analyze and refine your policies; ensure that your policies meet the guidelines and best practices below; and then, when you are ready, approve and enforce those policies in the workspace. When you are ready to enforce segmentation policy for a particular VPC, return to the connector configuration to enable segmentation for the VPC. For details, see [Best Practices When Enforcing Segmentation Policy for AWS Inventory, on page 240](#).

If you have enabled the Kubernetes managed services (EKS) option:

- Install Kubernetes agents on your container-based workloads. For details, see the *Kubernetes/OpenShift Agents - Deep Visibility and Enforcement* section in the agent deployment chapter.

Event Log:

The event logs can be used to know significant events happening per connector from different capabilities. We can filter them using various attributes like Component, Namespace, Messages and Timestamp.

Edit an AWS Connector

You can edit an AWS connector, for example to enable segmentation enforcement for specific VPCs or to make other changes.

Changes are not saved until you finish the wizard.

Procedure

-
- Step 1** From the navigation bar at the left side of the window, choose **Manage > Workloads > Connectors**.
 - Step 2** Click **AWS**.
 - Step 3** If you have more than one AWS connector, choose the connector to edit from the top of the window.
 - Step 4** Click **Edit Connector**.
 - Step 5** Click through the wizard again and make changes. For detailed descriptions of the settings, see [Configure new AWS Connector](#), on page 236.
 - Step 6** If you enable different capabilities (gathering labels, ingesting flows, enforcing segmentation, or gathering EKS data), you must download the revised CloudFormation Template (CFT) and upload it to AWS before continuing the wizard.
 - Step 7** To enable enforcement of segmentation policy, first make sure you have completed recommended prerequisites described in [Best Practices When Enforcing Segmentation Policy for AWS Inventory](#). On the page that lists the VPCs, choose **Enable Segmentation** for the VPCs on which you want to enable enforcement.
 - Step 8** If you have already created scopes for any of the selected VPCs, either using the wizard or manually, click **Skip this step** to complete the wizard.

You can edit the scope tree manually using the **Organize > Scopes and Inventory** page.
 - Step 9** If you have not already created any scopes for the selected VPCs and you want to keep the proposed hierarchy, choose the parent scope from above the scope tree, then click **Save**.
-

Deleting Connectors and Data

If you delete a connector, data already ingested by that connector is not deleted.

Labels and inventory are automatically deleted from active inventory after 24 hours.

Best Practices When Enforcing Segmentation Policy for AWS Inventory



-
- Warning** Before you enable segmentation enforcement on any VPC, create a backup of the security groups on that VPC. Enabling segmentation for a VPC removes existing Security Groups from that VPC. Disabling segmentation does not restore the old security groups.
-

When creating policies:

- As with all discovered policies, ensure that you have enough flow data to produce accurate policies.

- Because AWS allows only ALLOW rules in security groups, your segmentation policies should include only Allow policies, except the Catch-All policy, which should have the Deny action.

We recommend that you enable enforcement in the workspace before you enable segmentation for the associated VPC. If you enable segmentation for a VPC that is not included in a workspace that has enforcement enabled, all traffic will be allowed on that VPC.

When you are ready to enforce policy for a VPC, edit the AWS connector (see [Edit an AWS Connector](#)) and enable segmentation for that VPC.

View AWS Inventory Labels, Details, and Enforcement Status

To view summary information for an AWS connector, navigate to the connector page (Manage > Connectors), then choose the connector from the top of the page. For more details, click a VPC row.

To view information about AWS VPC inventory, click an IP address on the AWS Connectors page to see the Inventory Profile page for that workload. For more information about inventory profiles, see [Inventory Profile](#).

For information about labels, see:

- [Labels Generated by Cloud Connectors](#)
- [Labels Related to Kubernetes Clusters](#)

Concrete policies for VPC inventory are generated based on their orchestrator_system/interface_id label value. You can see this on the Inventory Profile page.

To view enforcement status, choose **Defend > Enforcement Status** from the navigation bar on the left side of the Secure Workload window. For more information, see [Enforcement Status for Cloud Connectors](#).

Troubleshoot AWS Connector Issues

Problem: The Enforcement Status page shows that a Concrete Policy was SKIPPED.

Solution: This occurs when the number of security groups exceeds the AWS limits, as configured in the AWS connector.

When a concrete policy shows as SKIPPED, the new security groups are not implemented and the previously existing security groups on AWS remain in effect.

To resolve this issue, see if you can consolidate policies, for example by using a larger subnet in one policy rather than multiple policies with smaller subnets.

If you choose to increase limits on the number of rules, you must contact Amazon before changing the limits in the AWS connector configuration.

Background:

Concrete policies are generated for each VPC when segmentation is enabled. These concrete policies are used to create security groups in AWS. However, AWS and Secure Workload count policies differently. When converting Secure Workload policies to AWS security groups, AWS counts each unique subnet as one rule.

Accounting example:

Consider the following example Secure Workload policy:

OUTBOUND: Consumer Address Set -> Provider Address Set Allow TCP port 80, 8080

AWS counts this policy as (the number of unique subnets in the Provider Address set) multiplied by (the number of unique ports).

So, if the provider address set consists of 20 Unique subnets, then this single Secure Workload policy counts in AWS as $20(\text{unique subnets}) * 2(\text{Unique ports}) = 40$ rules in security groups.

Keep in mind that because the VPCs are dynamic, the rule count is also dynamic, so the counts are approximate.

Problem: AWS unexpectedly allows all traffic

Solution: Make sure your Catch-All policy in Secure Workload is set to Deny.

Managed Kubernetes Services Running on AWS (EKS)

If you have deployed Amazon Elastic Kubernetes Service (EKS) on your AWS cloud, then you can use an AWS connector to pull in inventory and labels (EKS tags) from your Kubernetes cluster.

When an AWS connector is configured to pull metadata from managed Kubernetes services, Secure Workload connects to the cluster's API server and tracks the status of nodes, pods and services in that cluster. For the Kubernetes labels gathered and generated using this connector, see [Labels Related to Kubernetes Clusters](#).

Requirements and Prerequisites for EKS

- Verify that your Kubernetes version is supported. See <https://www.cisco.com/go/secure-workload/requirements/integrations>.
- Configure the required access in EKS, as described below.

EKS Roles and Access Privileges

User credentials and AssumeRole (if applicable) must be configured with a minimum set of privileges. The user/role must be specified in the aws-auth.yaml config map. The aws-auth.yaml config map can be edited using the following command.

```
$ kubectl edit configmap -n kube-system aws-auth
```

If AssumeRole is not used, the user must be added to the “mapUsers” section of the aws-auth.yaml config map with appropriate group. If AssumeRole ARN is specified, the role must be added to the “mapRoles” section of the aws-auth.yaml config map. A sample aws-auth.yaml config map with AssumeRole is provided below.

```
apiVersion: v1
data:
  mapAccounts: |
    []
  mapRoles: |
    - "groups":
      - "system:bootstrappers"
      - "system:nodes"
      "rolearn": "arn:aws:iam::938996165657:role/eks-cluster-2021011418144523470000000a"

      "username": "system:node:{EC2PrivateDNSName}"
    - "rolearn": arn:aws:iam::938996165657:role/BasicPrivilegesRole
      "username": secure.workload.read.only-user
      "groups":
        - secure.workload.read.only

  mapUsers: |
    []
kind: ConfigMap
metadata:
  creationTimestamp: "2021-01-14T18:14:47Z"
  managedFields:
```

```

- apiVersion: v1
  fieldsType: FieldsV1
  fieldsV1:
    f:data:
      .: {}
      f:mapAccounts: {}
      f:mapRoles: {}
      f:mapUsers: {}
  manager: HashiCorp
  operation: Update
  time: "2021-01-14T18:14:47Z"
  name: aws-auth
  namespace: kube-system
  resourceVersion: "829"
  selfLink: /api/v1/namespaces/kube-system/configmaps/aws-auth
  uid: 6c5a3ac7-58c7-4c57-a9c9-cad701110569

```

EKS specific RBAC considerations

Create a cluster role binding of the cluster role and the user/service account.

```

apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: csw-clusterrolebinding
subjects:
- kind: User
  name: csw.read.only
  apiGroup: rbac.authorization.k8s.io
roleRef:
  kind: ClusterRole
  name: csw.read.only
  apiGroup: rbac.authorization.k8s.io
kubect1 create -f clusterrolebinding.yaml
clusterrolebinding.rbac.authorization.k8s.io/csw-clusterrolebinding created

```

For information on EKS roles and access, see the EKS Roles and Access Privileges section.

Configure EKS Settings in the AWS Connector Wizard

You enable the Managed Kubernetes Services capability when you configure the AWS connector. See [Configure new AWS Connector , on page 236](#).

You will need the Assume Role ARN for each EKS cluster. For more information, see: https://docs.aws.amazon.com/STS/latest/APIReference/API_AssumeRole.html

If you are using the AWS user to access the EKS cluster, allow the user to access the Assume Role.

If you are using a cross-account IAM role, allow the IAM role to access the Assume Role.

Support for EKS Load Balancer

We add support for load balancer services in EKS. The CSW agents enforce rules on consumer hosts and provider hosts/pods.

An EKS Load Balancer has two options:

1. Preserve Client IP.
2. On provider pod, we generate
3. Target Type.

Before starting with cases, for the following policy intent:

Consumer to provider service, service protocol and port with allow action rules for various cases are generated as follows:

| Case | Preserve Client | Target Type |
|------|-----------------|-------------|
| 1 | On | IP |
| 2 | On | Instance |
| 3 | Off | IP |
| 4 | Off | Instance |

Case 1:

On consumer node we generate an egress rule with consumer to load balancer service (lb ingress ip) service protocol and port allow.

There are no host rules on the provider node, but we generate an Ingress rule on the provider pod with source as the consumer, the destination as provider pod (any), the protocol as the target protocol, the port as the target port, and the action as allow.

Case 2:

On consumer node we generate an egress rule with consumer to load balancer service (lb ingress ip) service protocol and port allow.

On provider node there is a prerouting rule generated with source as consumer and destination as all provider nodes, protocol as service protocol, port as node port of the service and an action as allow.

On provider pod, we generate an Ingress rule with source as provider nodes, destination as provider pod (any), protocol as target protocol, port as target port and action as allow.

Case 3:

On consumer node we generate an egress rule with consumer to load balancer service (lb ingress ip) service protocol and port allow.

There are no host rules on the provider node. On provider pod, we generate an Ingress rule with source as lb ingress ip's destination as provider pod (any), protocol as target protocol, port as target port and action as allow.

Case 4:

On consumer node we generate an egress rule with consumer to load balancer service(lb ingress ip) service protocol and port allow.

The provider node generates a prerouting rule that sets the lb ingress IPs as the source and all provider nodes as the destination. The rule specifies the service protocol as the protocol and the node port of the service as the port, with the action set to allow.

On provider pod, we generate an Ingress rule with source as provider nodes, destination as provider pod (any), protocol as target protocol, port as target port and action as allow.

Azure Connector

The Azure connector connects with your Microsoft Azure account to perform the following high-level functions:

- **Automated ingestion of inventory (and its tags) live from your Azure virtual networks (VNETs)**
Azure allows you to assign metadata to your resources in the form of tags. Secure Workload can ingest the tags associated with virtual machines and network interfaces, which can then be used as labels in Secure Workload for inventory and traffic flow data visualization and policy definitions. This metadata is synchronized constantly.

The tags from workloads and network interfaces of the subscription associated with the connector are ingested. If both workloads and network interfaces are configured then the tags are merged and displayed in Secure Workload. For more information, see [Labels Generated by Cloud Connectors, on page 336](#).
- **Ingestion of flow logs** The connector can ingest flow logs that you set up in Azure for your Network Security Groups (NSGs). You can then use this telemetry data in Secure Workload for visualization and segmentation policy generation.
- **Segmentation** When enforcement of segmentation policy is enabled for a virtual network, Secure Workload policies will be enforced using Azure's native Network Security Groups.
- **Automated ingestion of metadata from AKS clusters** When Azure Kubernetes Services (AKS) are running on Azure, you can choose to gather all node, service, and pod metadata related to all selected Kubernetes clusters.

You can choose which of the above capabilities to enable for each VNET.

Azure connector supports multiple subscriptions.



Note China Regions are currently not supported.

Requirements and Prerequisites for Azure

For all capabilities: A single connector can handle multiple subscriptions. You will need a subscription ID to configure the connector. This subscription ID can be one of the many subscription IDs that are being onboarded to a connector.

In Azure, create/register an application using Azure Active Directory (AD). You will need the following information from this application:

- Application (client) ID
- Directory (tenant) ID
- Client credentials (you can use either a certificate or a client secret)
- Subscription ID

The connector configuration wizard will generate an Azure Resource Manager (ARM) template that you can use to create a custom role with the permissions needed for the connector capabilities you choose to enable. These permissions will apply to all resources in the subscription you specify for the connector. Make sure you have permissions in Azure to upload this template.

If required for connectivity, ensure that you have an HTTP proxy available for this integration.

Each virtual network (VNET) can belong to only one Azure connector. An Azure account can have multiple Azure connectors.

This connector does not require a virtual appliance.

For gathering labels and inventory: No additional prerequisites are required.

For ingesting flow logs: Each virtual network (VNet) must have at least one subnet configured.

Every subnet under each VNet must have a Network Security Group (NSG) associated with it. You can associate a single NSG with multiple subnets. You can specify any Resource Group when configuring the NSG.

Only traffic that hits an NSG rule will be included in flow logs. Therefore, every NSG should have at least one rule each for inbound traffic and for outbound traffic that applies to any source, any destination – the equivalent of a catch-all rule in Secure Workload. (By default, NSGs include these rules.)

Each NSG must have flow logs enabled.

- A storage account in Azure is required. Access permissions must be included for the subscription you are using for this connector.
- The flow logs must use Version 2.
- Retention time can be 2 days (the connector pulls new flow data every minute, and two days should allow enough time for any connection failures to be remedied.)

For segmentation: Enabling segmentation requires Gather Labels to be enabled.

When you enable segmentation for a virtual network (VNet), all existing rules are removed from the NSGs associated with subnets and the network interfaces that are part of those subnets. Back up your existing NSG rules on subnet and network interface before you enable segmentation in the connector.

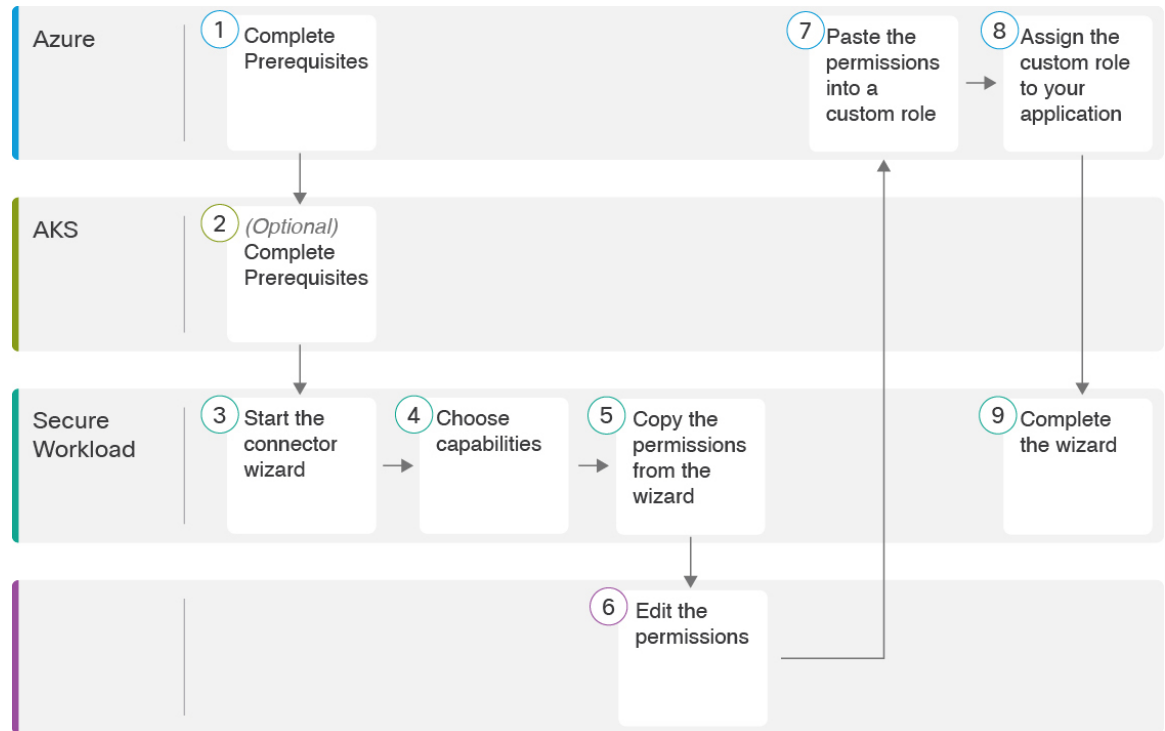
See also [Best Practices When Enforcing Segmentation Policy for Azure Inventory, on page 251](#), below.

For managed Kubernetes services (AKS): If you will enable the Kubernetes AKS option, see requirements and prerequisites in the Managed Kubernetes Services running on Azure (AKS) section below, .

Azure Connector Configuration Overview

The following graphic gives a high-level overview of the connector configuration process. For essential details, see the next topic ([Configure an Azure Connector](#).)

Figure 87: Azure connector configuration overview



(Note that the numbers in the graphic do not correspond to step numbers in the detailed procedure.)

Configure an Azure Connector

Procedure

- Step 1** From the navigation bar at the left side of the window, choose **Manage > Connectors**.
- Step 2** Click the Azure connector.
- Step 3** Click **Enable** for the first connector (in a root scope) or **Enable Another** for additional connectors in the same root scope.
- Step 4** Understand and meet requirements and prerequisites in [Requirements and Prerequisites for Azure](#), then click Get Started.
- Step 5** Name the connector and choose desired capabilities:

Selections you make on this page are used only to determine the privileges included in the Azure Resource Manager (ARM) template that will be generated in the next step, and to display the settings that you will need to configure.

In order to enable segmentation, you must also enable **Gather Labels**.

Enabling Segmentation on this page does not in itself enable policy enforcement or affect existing network security groups. Policy enforcement and deletion of existing security groups occurs only if you enable Segmentation for individual VNets later in the wizard. You can return to this wizard later to enable segmentation policy enforcement for individual VNets.

Step 6 Click **Next** and read the information on the configuration page.

Step 7 Your subscription must have the required privileges before you can continue to the next page in the wizard. To use the provided Azure Resource Manager (ARM) template to assign required permissions for the connector:

- a. Download the ARM template from the wizard.
- b. Edit the template text to replace **<subscription_ID>** with your subscription ID.

Note For a connector, you can create multiple subscription IDs in the Azure account. You can enter multiple subscription IDs where the credentials belong to the same subscription ID.
- c. In Azure, create a custom role in the applicable subscription.
- d. In the custom role form, for the Baseline permissions, choose **Start from scratch**.
- e. In the JSON tab of the custom role creation form, paste the text from the edited file you downloaded from the connector wizard.
- f. Save the custom role.
- g. Attach the custom role to the application you configured in the prerequisites for this procedure.

This template has the IAM permissions required for the capabilities that you selected in the previous step.

If you enabled the Kubernetes managed services option, you must separately configure permissions for AKS. See [Managed Kubernetes Services Running on Azure \(AKS\)](#), on page 251.

Step 8 Configure settings:

| Attribute | Description |
|-------------------------------------|---|
| SubscriptionID | The ID of the Azure subscription that you are associating with this connector. |
| ClientID | The Application (client) ID from the application that you created in Azure for this connector. |
| TenantID | The Directory (tenant) ID from the application that you created in Azure for this connector. |
| Client Secret or Client Certificate | For authentication, you can use either a client secret or a client certificate and key. Obtain either from the Client credentials link in the application that you created in Azure for this connector. If you use a certificate: The certificate should be unencrypted. Only RSA certificates are supported. Private keys can be either PKCS1 or PKCS8. |
| HTTP Proxy | Proxy required for Secure Workload to reach Azure. Supported proxy ports: 80, 8080, 443, and 3128. |
| Full Scan Interval | Frequency with which Secure Workload refreshes complete inventory data from Azure. Default and minimum is 3600 seconds. |

| Attribute | Description |
|---------------------|--|
| Delta Scan Interval | Frequency with which Secure Workload fetches incremental changes in inventory data from Azure. Default and minimum is 600 seconds. |

Step 9 Click **Next**. It may take a few minutes for the system to obtain the list of VNets and AKS clusters from Azure.

Step 10 From the list of VNets and AKS clusters for each VNet, choose the VNets and AKS clusters for which you want to enable your selected capabilities.

Generally, you should enable flow ingestion as soon as possible, so that Secure Workload can begin to collect enough data to suggest accurate policies.

Note that since AKS only supports Gather Labels capability, no explicit capability selection has been provided. Selecting an AKS cluster will implicitly enable the supported capability. Upload the client certificate and key for each cluster for which you enable this functionality.

Generally, you should not choose **Enable Segmentation** during initial configuration. Later, when you are ready to enforce segmentation policy for specific VNets, you can edit the connector and enable segmentation for those VNets. See [Best Practices When Enforcing Segmentation Policy for Azure Inventory, on page 251](#).

Step 11 Once your selections are complete, click **Create** and wait a few minutes for the validation check to complete. The View Groups page shows all VNets that you enabled for any functionality on the previous page, grouped by region. Each region, and each VNet in each region, is a new scope.

Step 12 (Optional) Choose the parent scope under which to add the new set of scopes. If you have not yet defined any scopes, your only option is the default scope.

Step 13 (Optional) To accept all settings configured in the wizard *including* the hierarchical scope tree, click **Save**.

To accept all settings *except* the hierarchical scope tree, click **Skip** this step.

You can manually create or edit the scope tree later, under **Organize > Scopes and Inventory**.

What to do next

If you have enabled gathering labels, ingesting flows data, and/or segmentation:

- If you enabled flow ingestion, it may take up to 25 minutes for flows to begin appearing on the **Investigate > Traffic** page.
- (Optional) For richer flow data and other benefits including visibility into host vulnerabilities (CVEs), install the appropriate agent for your operating system on your VNet-based workloads. For requirements and details, see the agent installation chapter.
- After you have successfully configured the Azure connector to gather labels and ingest flows, follow the standard process for building segmentation policies. For example: Allow Secure Workload to gather sufficient flow data to generate reliable policies; define or modify scopes (typically one for each VNet); create a workspace for each scope; automatically discover policies based on your flow data, and/or manually create policies; analyze and refine your policies; ensure that your policies meet the guidelines and best practices below; and then, when you are ready, approve and enforce those policies in the workspace. When you are ready to enforce segmentation policy for a particular VNet, return to the

connector configuration to enable segmentation for the VNet. For details, see [Best Practices When Enforcing Segmentation Policy for Azure Inventory, on page 251](#).

If you have enabled the Kubernetes managed services (AKS) option:

- Install Kubernetes agents on your container-based workloads. For details, see [Installing Kubernetes or OpenShift Agents for Deep Visibility and Enforcement, on page 29](#) in the agent deployment chapter.

Event Log:

The event logs can be used to know significant events happening per connector from different capabilities. We can filter them using various attributes like Component, Namespace, Messages and Timestamp.

Edit an Azure Connector

You can edit an Azure connector, for example to enable segmentation enforcement for specific VNets or to make other changes.

Changes are not saved until you finish the wizard.

Procedure

- Step 1** From the navigation bar at the left side of the window, choose **Manage > Connectors**.
 - Step 2** Click **Azure**.
 - Step 3** If you have more than one Azure connector, choose the connector to edit from the top of the window.
 - Step 4** Click **Edit Connector**.
 - Step 5** Click through the wizard again and make changes. For detailed descriptions of the settings, see [Configure an Azure Connector, on page 247](#).
 - Step 6** If you enable different capabilities (gathering labels, ingesting flows, enforcing segmentation, or gathering AKS data), you must download the revised ARM template, edit the new template text to specify the subscription ID, and upload the new template to the custom role you created in Azure before continuing the wizard.
 - Step 7** To enable enforcement of segmentation policy, first make sure you have completed recommended prerequisites described in [Best Practices When Enforcing Segmentation Policy for Azure Inventory, on page 251](#). Then, on the wizard page that lists the VNets, choose **Enable Segmentation** for the VNets on which you want to enable enforcement.
 - Step 8** If you have already created scopes for any of the selected VNets, either using the wizard or manually, click **Skip this step** to complete the wizard.

You can edit the scope tree manually using the **Organize > Scopes and Inventory** page.
 - Step 9** If you have not already created any scopes for the selected VNets and you want to keep the proposed hierarchy, choose the parent scope from above the scope tree, then click **Save**.
-

Deleting Connectors and Data

If you delete a connector, data already ingested by that connector is not deleted.

Labels and inventory are automatically deleted from active inventory after 24 hours.

Best Practices When Enforcing Segmentation Policy for Azure Inventory



Warning Before you enable segmentation enforcement on any VNet, create a backup of the network security groups on that VNet. Enabling segmentation for a VNet removes existing rules from the network security group associated with that virtual network. Disabling segmentation does not restore the old network security groups.

When creating policies: As with all discovered policies, ensure that you have enough flow data to produce accurate policies.

We recommend that you enable enforcement in the workspace before you enable segmentation for the associated VNet. If you enable segmentation for a VNet that is not included in a workspace that has enforcement enabled, all traffic will be allowed on that VNet.

When you are ready to enforce policy for a VNet, edit the Azure connector (see [Edit an Azure Connector, on page 250](#)) and enable segmentation for that VNet.

Note that if a subnet does not have a Network Security Group associated with it, Secure Workload does not enforce segmentation policy on that subnet. When you enforce segmentation policy on a VNet, the NSG at the subnet level is changed to allow all traffic, and Secure Workload policies overwrite the interface-level NSGs. An NSG for the interface is automatically created if not already present.

View Azure Inventory Labels, Details, and Enforcement Status

To view summary information for an Azure connector, navigate to the connector page (Manage > Connectors), then choose the connector from the top of the page. For more details, click a VNet row.

To view information about Azure VNet inventory, click an IP address on the Azure Connectors page to view the Inventory Profile page for that workload. For more information about inventory profiles, see [Inventory Profile](#).

For information about labels, see:

- [Labels Generated by Cloud Connectors](#)
- [Labels Related to Kubernetes Clusters](#)

Concrete policies for VNet inventory are generated based on their `orchestrator_system/interface_id` label value. You can see this on the Inventory Profile page.

To view enforcement status, choose **Defend > Enforcement Status** from the navigation bar on the left side of the Secure Workload window. For more information, see [Enforcement Status for Cloud Connectors](#).

Troubleshoot Azure Connector Issues

Problem: Azure unexpectedly allows all traffic

Solution: Make sure your Catch-All policy in Secure Workload is set to Deny.

Managed Kubernetes Services Running on Azure (AKS)

If you have deployed Azure Kubernetes Services (AKS) on your Azure cloud, then you can use an Azure connector to dynamically pull in inventory and labels (AKS tags) from your Kubernetes cluster.

When an Azure connector is configured to pull metadata from managed Kubernetes services, Secure Workload tracks the status of nodes, pods and services in that cluster.

For the Kubernetes labels gathered and generated using this connector, see [Labels Related to Kubernetes Clusters](#).

Requirements and Prerequisites for AKS

- Verify that your Kubernetes version is supported. See the [Compatibility Matrix](#) for the operating systems, external systems, and connectors for Secure Workload agents.
- Enable and configure the Managed Kubernetes Services (AKS) capability when you configure the Azure connector. See [Configure an Azure Connector](#) for details.

Support for AKS Load Balancer

AKS supports Preserve client IP.

For the following policy intent:

Consumer to provider service, service protocol and port with allow action rules for various cases generates as follows:

| Case | Preserve Client |
|------|-----------------|
| 1 | On |
| 2 | Off |

Case 1: Preserve client IP is **on**.

On the consumer node we generate an egress rule with consumer to load balancer service (lb ingress ip) service protocol and port allow.

A prerouting rule generated for provider node, which specifies the consumer as the source and all provider nodes as the destination. The rule includes the service protocol as the protocol and the node port of the service as the port, with the action set to allow.

On the provider pod, we generate an Ingress rule with src as provider nodes, dest as provider pod (any), protocol as target protocol, port as target port and action as allow.

Case 2: Preserve client IP is **off**.

On the consumer node we generate an egress rule with consumer to load balancer service (lb ingress ip) service protocol and port allow.

The provider node generates a prerouting rule that sets the lb ingress IPs as the source and all provider nodes as the destination. The rule specifies the service protocol as the protocol and the node port of the service as the port, with the action set to allow.

On the provider pod, we generate an Ingress rule with source as provider nodes, destination as provider pod (any), protocol as target protocol, port as target port and action as allow.

GCP Connector

The Google Cloud Platform connector connects with GCP to perform the following high-level functions:

- **Automated ingestion of inventory (and its tags) live from GCP Virtual Private Cloud (VPC)**

GCP allows you to assign metadata to your resources in the form of tags. Secure Workload will query the tags for these resources which can then be used for inventory and traffic flow data visualization, and

policy definition. This capability keeps the resource tag mapping updated by constantly synchronizing this data.

The tags from workloads and network interfaces of a GCP VPC are ingested. If both workloads and network interfaces are configured then the tags are merged and displayed in Secure Workload. For more information, see [Labels Generated by Cloud Connectors, on page 336](#).

- **Ingestion of flow logs from VPC** If you have set up VPC flow logs in GCP for monitoring purposes, Secure Workload can ingest flow log information by reading the corresponding Google Storage bucket. This telemetry can be used for visualization and segmentation policy generation.
- **Segmentation** Enabling this option will allow Secure Workload to program security policies using GCP native VPC firewall. When enforcement is enabled for a VPC, relevant policies will be automatically programmed to the VPC firewall.
- **Automated ingestion of metadata from GKE clusters** (K8s capabilities) when Google Kubernetes Engine (GKE) is running on GCP, you can choose to gather all node, service, and pod metadata related to all selected Kubernetes clusters.

You can choose which of the above capabilities to enable for each VPC.

Requirements and Prerequisites for GCP Connector

For all capabilities: Create a dedicated service account in GCP, or identify an existing GCP service account for this connector. The connector configuration wizard generates a IAM policy list that you can use to assign required privileges to this service account. Make sure you have permissions in GCP to upload this IAM policy list.



Note The recommended method for applying the permission in the IAM policy list to the service account is through the CLI.

Each VPC can belong to only one GCP connector. An Secure Workload cluster can have multiple GCP connectors. Gather the information described in the tables in [Configure a GCP Connector, on page 255](#), below.

This connector does not require a virtual appliance.

- **For gathering labels and inventory:** No additional prerequisites are required.
- **For ingesting flow logs:** VPC level flow log definitions are required in order to trigger the collection of flow logs.

To use the flow log ingestion, user is required to enable flow logs on the desired VPCs and setup a log router sink.

Inclusion filter for the log router sink:

1. `resource.type="gce-subnetwork"`
2. `log_name="projects/<project_id>/logs/compute.googleapis.com%2Fvpc_flows"`

Choose the sink destination as a cloud storage bucket and then choose the desired storage bucket.

While configuring the GCP connector with ingress flow logs, it is mandatory to enter the storage bucket name.

Only flow logs from VPC can be ingested.

Flow logs must be published to Google storage bucket; Secure Workload cannot collect flow data from Google Cloud Operations Suite.

Secure Workload can ingest flow logs from an Google Storage bucket associated with any account, if the GCP user account provided during connector creation have access to both the VPC flow logs and the Google storage bucket.

The following flow log attributes (in any order) are required in the flow log: Source Address, Destination Address, Source Port, Destination Port, Protocol, Packets, Bytes, Start Time, End Time, Action, TCP Flags, Interface-ID, Log status and Flow Direction. Any other attributes are ignored.

Flow logs must capture both Allowed and Denied traffic.

- **For segmentation:** Enabling segmentation requires Gather Labels to be enabled.

Back up your existing security groups before enabling segmentation in the connector, as all existing rules will be overwritten when you enable segmentation policy enforcement for a VPC.

See also [Best Practices When Enforcing Segmentation Policy for GCP Inventory](#), on page 259, below.

- **For managed Kubernetes services (GKE):** If you enable the Kubernetes option, see requirements and prerequisites in the [Managed Kubernetes Services Running on GCP \(GKE\)](#), on page 260 section below, including required access privileges.

Configure Multiple Projects Access in GCP

To configure cross multiple projects access in GCP, you can follow these steps:

Procedure

- Step 1** Sign in to your [GCP](#) console.
- Step 2** Click on the project drop-down menu in the top navigation bar and select **New Project** or you can either create a new Project or use an existing project with service Account.
- Step 3** Enter a name for your new project. Choose the organization that own the new project or select **No organization** if you do not have one.
- Step 4** Click on the **Create** button to create the new project.
- Note** You can repeat the step 2 to 4 to create as many projects as you need.
- Step 5** To link multiple projects in a single service account, navigate to **IAM & Admin** page and choose **Service Account**.
- Step 6** Click on the **Create Service Account** button. Follow the prompts to create the service account and grant it the necessary permissions.
- Note** You can either use an existing service account or create a new service account.
- Step 7** From the **Keys** tab, click **Add Key** to generate a private key in JSON file.
- Step 8** Go to the **IAM & Admin** page in the GCP console and select **IAM**.
- Note** You have to first change the project before you click on IAM & Admin and then try to grant privilege.
- Step 9** Click on the **Grant access** button to add a new project.

- Step 10** In the **New principals** field, enter the email address of the service account you want to link to the project.
- Step 11** Click on the **Save** button to associate the service account to your project.

Note Repeat these steps for each project that you want to link to your original project.

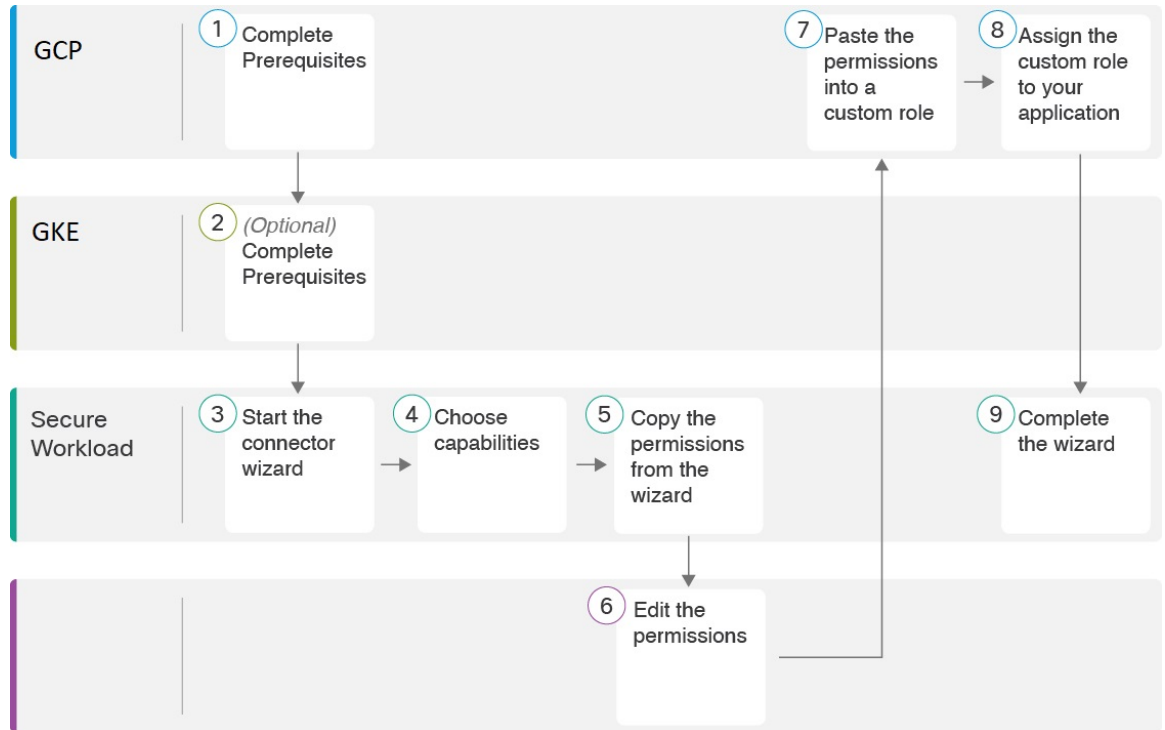
You can manage the service account permissions by going to the **IAM & Admin** page in the GCP console and selecting **IAM** for each project.

- Step 12** Make sure that the Service Account has permissions to least common ancestor (common ancestor to all the projects selected) resource level, such as a folder or organization.

GCP Connector Configuration Overview

The following graphic gives a high-level overview of the connector configuration process. For essential details, see the next topic ([Configure a GCP Connector, on page 255.](#))

Figure 88: GCP connector configuration overview



(Note that the numbers in the graphic do not correspond to step numbers in the detailed procedure.)

Configure a GCP Connector

Procedure

- Step 1** From the navigation bar at the left side of the window, choose **Manage > Connectors**.
- Step 2** Click the **GCP connector**.

- Step 3** Click **Enable** for the first connector (in a root scope) or **Enable Another** for additional connectors in the same root scope.
- Step 4** Understand and meet requirements and prerequisites in [Requirements and Prerequisites for GCP Connector](#), on page 253 and [Managed Kubernetes Services Running on GCP \(GKE\)](#), on page 260, then click **Get Started**.
- Step 5** Enter a name for the connector and choose desired capabilities, then click **Next**.

Selections you make on this page are used only to determine the privileges included in the IAM policy list that will be generated in the next step, and to display the settings that you will need to configure.

If the **Ingest Flow Logs** capabilities is checked, you must enter **Flow Log Storage Bucket Name** in the next step.

In order to enable **Segmentation**, you must check **Gather Labels**.

- Step 6** Create **Service Accounts** in the [Google Cloud console](#).
- Step 7** Download the generated IAM custom role policy list.
- This IAM custom role policy list has the IAM privileges required for the capabilities that you selected in the previous step.
- If you have enabled the Kubernetes option, you must separately configure permissions for GKE.
- For more information, see [Managed Kubernetes Services Running on GCP \(GKE\)](#), on page 260.
- Step 8** Generate a **Service Accounts** custom role in the [Google Cloud console](#); use the sample command below using [Google Cloud CLI](#):
- ```
gcloud iam roles create <Role Name> --project=<Project id> --file=<File path>
```
- Step 9** Upload the `service account json` file with required capabilities that was created as a prerequisite.
- Note** In GCP, the single connector supports multiple projects and ensure that the service account is directly linked to all projects.
- Step 10** Enter the **Flow Log Storage Bucket Name** if the Ingress Flow logs capability is checked.
- Step 11** Enter the **Root Resource Id**, which is also the GCP folder ID or organization ID.

**Note** To obtain the Root Resource ID, navigating to the [IAM & Admin, Settings](#) section and you can view it directly in the Cloud Console. Alternatively, you can also utilize the [Cloud SDK command](#) to retrieve the Root Resource ID.

**Step 12** Configure the following settings:

| Attribute           | Description                                                                                                                      |
|---------------------|----------------------------------------------------------------------------------------------------------------------------------|
| HTTP Proxy          | Proxy required for Secure Workload to reach GCP.                                                                                 |
| Full Scan Interval  | Frequency with which Secure Workload refreshes complete inventory data from GCP. Default and minimum is 3600 seconds.            |
| Delta Scan Interval | Frequency with which Secure Workload fetches incremental changes in inventory data from GCP. Default and minimum is 600 seconds. |

**Step 13** Click **Next**. It may take a few minutes for the system to obtain the list of virtual network and GKE clusters from your GCP project(s).

**Step 14** From the list of VPCs (Virtual Networks) and GKE clusters, choose the resources and their respective capabilities.

Generally, you should enable flow ingestion as soon as possible, so that Secure Workload can begin to collect enough data required to suggest accurate policies.

Generally, you should not choose **Enable Segmentation** during initial configuration. Later, when you are ready to enforce segmentation policy for specific VPCs, you can edit the connector and enable segmentation for those VPCs. See the Best Practices When Enforcing Segmentation Policy for GCP Inventory.

**Step 15** Click **Create** and wait a few minutes for the validation check to complete.

The View Groups page shows all VPCs that you enabled for any functionality on the previous page, grouped by logical\_group\_id (CSW), which is also a project\_id (GCP). Each logical\_group\_id, and each VPC in each logical\_group\_id, is a new scope.

**Step 16** Choose the parent scope under which to add the new set of scopes. If you have not yet defined any scopes, your only option is the default scope.

**Step 17** To accept all settings configured in the wizard *including* the hierarchical scope tree, click **Save**.

To accept all settings *except* the hierarchical scope tree, click **Skip** this step.

You can manually create or edit the scope tree later, under **Organize > Scopes and Inventory**.

---

### What to do next

#### If you have enabled gathering labels, ingesting flow data, and/or segmentation:

- If you enabled flow ingestion, it may take up to 25 minutes for flows to begin appearing on the **Investigate > Traffic** page.

- (Optional) For richer flow data and other benefits including visibility into host vulnerabilities (CVEs), install the appropriate agent for your operating system on your VPC-based workloads. For requirements and details, see the agent installation chapter.
- After you have successfully configured the GCP connector to gather labels and ingest flows, follow the standard process for building segmentation policies. For example: Allow Secure Workload to gather sufficient flow data to generate reliable policies; define or modify scopes (typically one for each VPC); create a workspace for each scope; automatically discover policies based on your flow data, and/or manually create policies; analyze and refine your policies; ensure that your policies meet the guidelines and best practices below; and then, when you are ready, approve and enforce those policies in the workspace. When you are ready to enforce segmentation policy for a particular VPC, return to the connector configuration to enable segmentation for the VPC. For details, see [Best Practices When Enforcing Segmentation Policy for GCP Inventory, on page 259](#).

**If you have enabled the Kubernetes managed services (GKE) option:**

- Install Kubernetes agents on your container-based workloads. For details, see [Installing Kubernetes or OpenShift Agents for Deep Visibility and Enforcement](#) in the agent deployment chapter.

**Event Log:**

The event logs can be used to know significant events happening per connector from different capabilities. We can filter them using various attributes like Component, Namespace, Messages and Timestamp.

## Edit a GCP Connector

If you want to enable gathering data from different or additional VPCs or GKE clusters, you may need to upload a service account json file with required capabilities with different permissions before you can select different VPCs or GKEs.

Changes are not saved until you finish the wizard.

### Procedure

- 
- Step 1** From the navigation bar at the left side of the window, choose **Manage > Workloads > Connectors**.
  - Step 2** Click **GCP Connector**.
  - Step 3** If you have more than one GCP connector, choose the connector to edit from the top of the window.
  - Step 4** Click **Edit Connector**.
  - Step 5** Click through the wizard again and make changes. For detailed descriptions of the settings, see [Configure a GCP Connector, on page 255](#).
  - Step 6** If you enable different capabilities (gathering labels, ingesting flows, enforcing segmentation, or gathering GKE data), you must download the revised IAM template and upload it to GKE before continuing the wizard.
  - Step 7** To enable enforcement of segmentation policy, first ensure that you have completed recommended prerequisites described in [Best Practices When Enforcing Segmentation Policy for GCP Inventory, on page 259](#). On the page that lists the VPCs, select **Enable Segmentation** for the VPCs on which you want to enable enforcement.
  - Step 8** If you have already created scopes for any of the selected VPCs, either using the wizard or manually, click **Skip this step** to complete the wizard.

You can edit the scope tree manually using the **Organize > Scopes and Inventory** page.

- Step 9** If you have not already created any scopes for the selected VPCs and you want to keep the proposed hierarchy, choose the parent scope from above the scope tree, then click **Save**.
- 

## Deleting Connectors and Data GCP

If you delete a connector, data already ingested by that connector is not deleted.

Labels and inventory are automatically deleted from active inventory after 24 hours.

## Best Practices When Enforcing Segmentation Policy for GCP Inventory



**Warning** Before you enable segmentation enforcement on any VPC, create a backup of the security groups on that VPC. Enabling segmentation for a VPC removes existing Security Groups from that VPC. Disabling segmentation does not restore the old security groups.

---

When creating policies:

- As with all discovered policies, ensure that you have enough flow data to produce accurate policies.
- Because GCP allows both ALLOW/DENY rules in firewall policy. Since GCP has very strict limitation on number of rules. So, it is better to have only ALLOW-list.

We recommend that you enable enforcement in the workspace before you enable segmentation for the associated VPC. If you enable segmentation for a VPC that is not included in a workspace that has enforcement enabled, all traffic will be allowed on that VPC.

When you are ready to enforce policy for a VPC, edit the GCP connector (see [Edit a GCP Connector, on page 258](#)) and enable segmentation for that VPC.

## GKE Inventory Labels, Details, and Enforcement Status

To view summary information for a GCP connector, navigate to **Connector >** and choose GCP Connector on the Connectors page.

To view information about inventory, click the IP address of a particular workload from the Scopes and Inventory page. You can also access the Inventory Profile from the interface tab on the VPC Profile. For more information about the Inventory profile, see [Inventory Profile](#).

Similarly, to view all Concrete Policies under the VPC profile, from the Inventory Profile Concrete Policies tab, navigate to the parent VPC Profile to see all the Concrete Policies under the VPC.

The VPC Profile is accessible from the GCP Configuration or Enforcement Status page (either global or within a workspace). You can view the Enforcement Status and Concrete Policies at the VPC level on the VPC Profile. You can also view the combined VPC Firewall Polices of all the interfaces on the VPC Firewall Policies tab.

For more information on labels, see:

- [Labels Generated by Cloud Connectors](#)
- [Labels Related to Kubernetes Clusters](#)

## Troubleshoot GCP Connector Issues

**Problem: The Enforcement Status page shows that a Concrete Policy was SKIPPED.**

**Solution:** This occurs when the number of rules in firewall policy exceeds the GCP limits, as configured in the GCP connector.

When a concrete policy shows as SKIPPED, the new security groups are not implemented and the previously existing security groups on GCP remain in effect.

To resolve this issue, see if you can consolidate policies, for example by using a larger subnet in one policy rather than multiple policies with smaller subnets.

Background:

Concrete policies are generated for each VPC when segmentation is enabled. These concrete policies are used to create firewall policy in GCP. However, GCP and Secure Workload count policies differently. When converting Secure Workload policies to GCP firewall rules in firewall policy, GCP counting mechanism is complex. For more details, see [GCP](#).

**Problem: GCP unexpectedly allows all traffic**

**Solution:** Make sure your Catch-All policy in Secure Workload is set to Deny.

## Managed Kubernetes Services Running on GCP (GKE)

You can use a cloud connector to gather metadata from Google Kubernetes Engine (GKE) clusters running on Google Cloud Platform (GCP).

The connector gathers all node, service, and pod metadata related to all selected Kubernetes clusters.

### Requirements and Prerequisites

**Secure Workload requirements:** This connector does not require a virtual appliance.

Platform requirements:

- Make sure you have permissions in GCP to configure the required access for this connector.
- Each GKE cluster can only belong to one GCP connector.
- Gather the information described in the tables in *Configure a GCP connector*, below.

GKE requirements:

- You must configure the required access privileges in GKE.
- To support Managed K8s capabilities, the roles required by the service account are:
  - Compute Network Viewer is an IAM role that gives read-only access to all network resources in GCP. <https://cloud.google.com/compute/docs/access/iam#compute.networkViewer>
  - Kubernetes Engine Viewer is a GKE cluster role that provides read-only access to resources within GKE clusters, such as nodes, pods, and GKE API objects. <https://cloud.google.com/iam/docs/understanding-roles#kubernetes-engine-roles>

## Identity Connectors

An Identity Connector serves as a bridge between Secure Workload and various identity stores, such as OpenLDAP and Active Directory. The connector enables you to synchronize the information that is stored in identity stores without the need for manual intervention. Currently, you can configure an identity connector to import user data from LDAP.

### Configure an OpenLDAP Connector

Lightweight Directory Access Protocol (LDAP) is a protocol designed for retrieving information about users, user groups, organizations, and other attributes. Its primary objective is to store data in the LDAP directory to streamline user management.



**Note** The supported version for OpenLDAP data ingestion is OpenLDAP 2.6.

### Configuration

Create an Identity Connector for LDAP in Secure Workload to establish communication with OpenLDAP.

#### Procedure

- Step 1** From the navigation pane, choose **Manage > Workloads > Connectors**.
- Step 2** Click **Identity Connector** and select **Configure your new connector here**.
- Step 3** On the **New Connection** page, enter the following details:

| Fields                | Description                                                                                                                                     |
|-----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Connector Name</b> | Enter a name for the connector.                                                                                                                 |
| <b>Description</b>    | Enter a description.                                                                                                                            |
| <b>Domain Name</b>    | Enter a domain name. The domain name must be unique in the selected scope, for example, csw.com.                                                |
| <b>Base DN</b>        | Enter the Base DN, or Distinguished Name that serves as the starting point for searches within the directory tree. For example, dc=csw, dc=com. |

| Fields                         | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
|--------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>User Filter</b>             | <p>Enter a filter to define the criteria for identifying entries that contain certain kinds of information.</p> <p>Example 1: To identify users, you can distinguish them by having two objectClass attributes—one set to 'person' and another to 'user.' The matching criteria can be</p> <pre>(&amp;(objectClass=person)(objectClass=user))</pre> <p>Example 2: To retrieve all the entries that have the objectClass=user and the cn attribute containing the word Marketing, the search filter can be</p> <pre>(&amp;(objectClass=user)(cn=*Marketing*))</pre> |
| <b>Username and Password</b>   | Enter the credentials to connect to the OpenLDAP server.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| <b>CA Certificate</b>          | Upload the CA certificate and enter the SSL server name that Secure Workload uses to authenticate. If not, <b>Disable SSL</b> .                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| <b>Server IP/FQDN and Port</b> | Enter the server IP address and port number.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| <b>Secure Connector</b>        | <p>Enable if a Secure Connector is used to tunnel connections from Secure Workload to OpenLDAP.</p> <p>Before you can enable this option, you should have deployed a Secure Connector.</p> <p>For more information, see <a href="#">Secure Connector</a>.</p>                                                                                                                                                                                                                                                                                                      |

**Step 4** Click **Create**.



Figure 89: Configure a New Connector

The screenshot shows a 'New Connection' configuration page. It has a 'Configuration' tab selected. The form contains the following fields and controls:

- Connector Name\***: Text input field with placeholder 'Connector Name (required)'.
- Description**: Text input field with placeholder 'Description'.
- Domain Name\***: Text input field with placeholder 'Domain Name (required)'.
- Base DN\***: Text input field with placeholder 'Base DN (required)'.
- User Filter\***: Text input field with placeholder 'User Filter (required)'.
- User Name\***: Text input field with placeholder 'User Name (required)'.
- Password\***: Text input field with placeholder 'Password (required)'.
- Disable SSL**: Toggle switch, currently turned off.
- CA Certificate\***: Section with an 'Upload certificate' button and a text input field with placeholder 'SSL Server Name'.
- Server IP/FQDN\***: Text input field with placeholder 'Server IP (required)' and example '172.28.171.185'.
- Port\***: Text input field with placeholder 'Port (required)' and example '443'.
- Secure Connector**: Toggle switch, currently turned on.

At the bottom right, there are 'Reset' and 'Create' buttons.

A new Identity Connector is created and the communication between Secure Workload and OpenLDAP is configured.

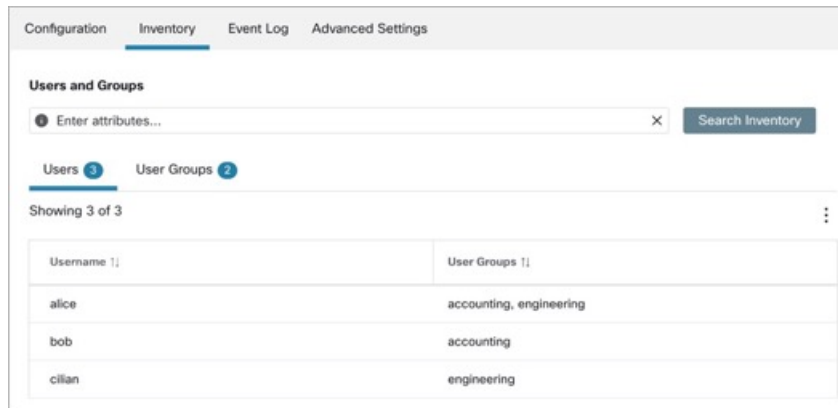
## Inventory

After the connection between Secure Workload and OpenLDAP is established, you can view a list of **Users** and **User Groups** in the **Inventory** tab. All the user groups that a user belongs to are displayed in the **Users** tab. Only unique user groups are displayed in the **User Groups** tab.

### Procedure

- Step 1** Enter the attributes to filter. Hover your cursor over the info icon to view the properties to filter.
- Step 2** Click the menu icon to download the data in JSON or CSV format.

Figure 90: Users and User Groups



**Note** The recommended limit for the number of users displayed is 300,000, while for user groups, it is 30,000.

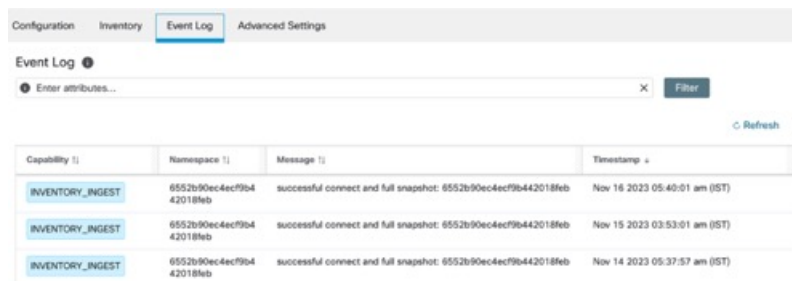
## Event Log

The **Event Log** tab displays information, warnings, and errors that occur while establishing the connection with OpenLDAP.

### Procedure

- Step 1** Enter the attributes to filter. Hover your cursor over the info icon to view the properties to filter.
- Step 2** Click the menu icon to download the data in JSON or CSV format.

Figure 91: Event Log



**Note** Color codes for the logs are Information (blue), Warning (orange), and Error (red).

## Advanced Settings

### Procedure

- Step 1** Under **Synchronize Schedule**, you can choose a time frequency at which Secure Workload synchronizes the user data from the LDAP server.
- Step 2** In the **User Attributes** field, enter up to six user attributes to be displayed.

*Figure 92: Advanced Settings*

The screenshot shows the 'Advanced Settings' tab in a web interface. Under the 'Synchronize Schedule' section, there is a text input field containing '60' and a dropdown menu set to 'minutes'. At the bottom right of the section are 'Reset' and 'Save' buttons.

## Connector Alerts

An appliance/service creates a connector alert when it experiences abnormal behavior.

### Alert Configuration

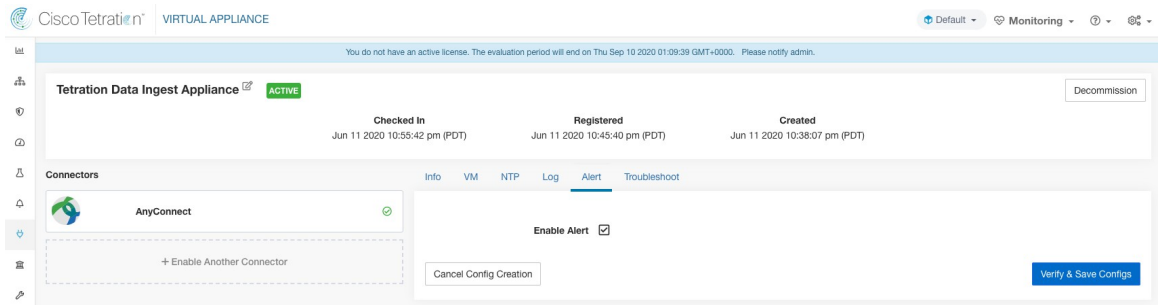
The alert configuration for appliances and connectors enables you to generate alerts for various events. In the 3.4 release, this configuration enables all types of alerts that are potentially possible for the configured appliance/connector.

| Parameter Name      | Type     | Description              |
|---------------------|----------|--------------------------|
| <b>Enable Alert</b> | checkbox | Should alert be enabled? |



**Note** The default value for *Enable Alert* is *true*.

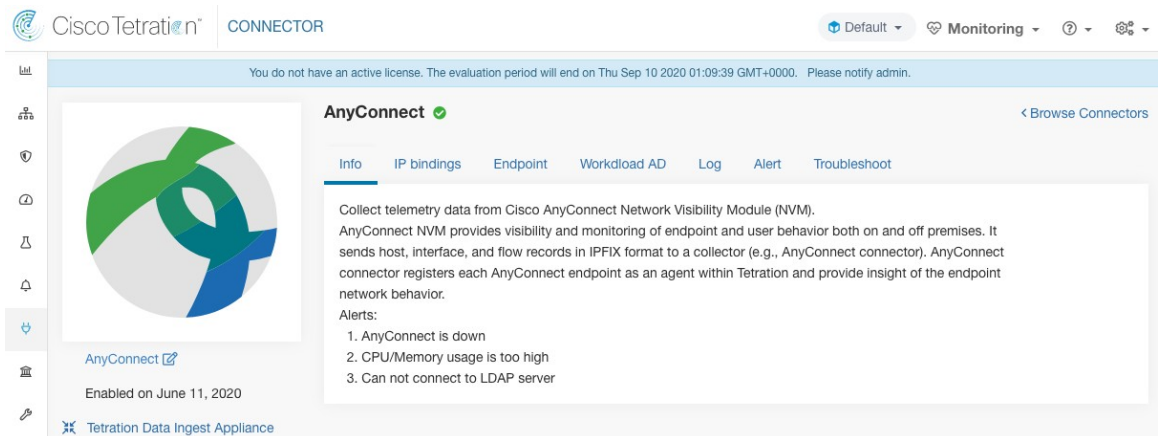
**Figure 93: Show Alert configuration on a Secure Workload Data Ingest Appliance**



## Alert Type

The Info Tab on the appliance and connector pages contains various alert types specific to each appliance and connector.

**Figure 94: Alert list info**



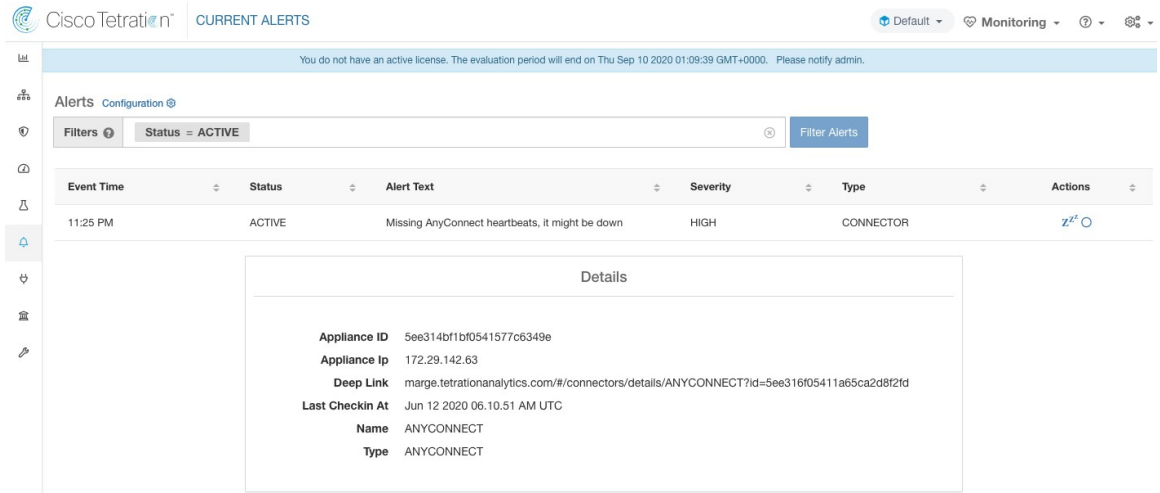
## Appliance/Connector down

An alert generates when an appliance (or a connector) is potentially down due to missing heartbeats from the appliance/connector.

Alert text: Missing <Appliance/Connector> heartbeats, it might be down.

Severity: High

Figure 95: Alert for connector down



Allowed Secure Workload virtual appliances: Secure Workload Ingest and Secure Workload Edge

Allowed connectors: All

## Appliance/Connector system usage

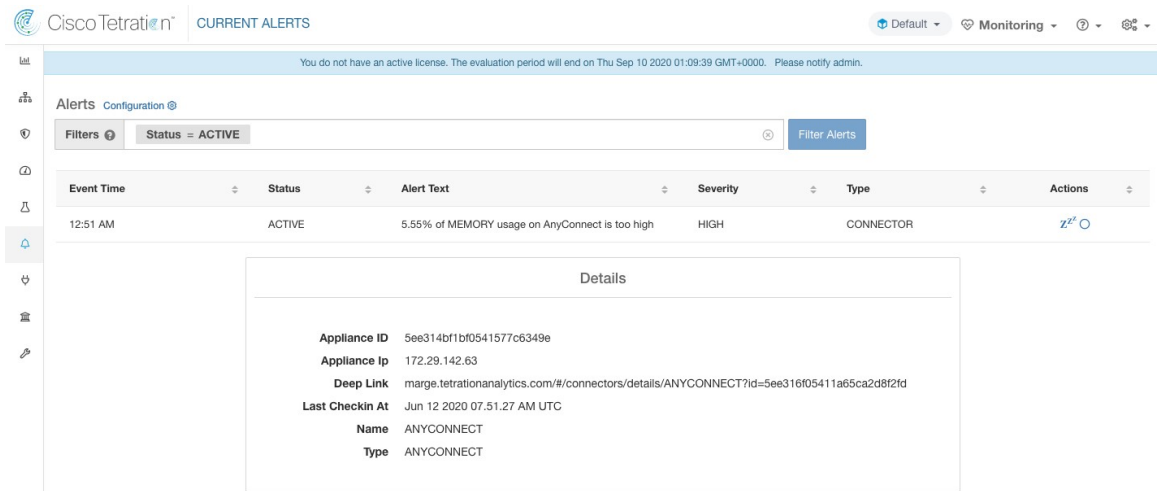
When system usage (CPU, memory, and disk) is more than 90% on an appliance (and a connector). The appliance (and/or connector) generates an informational alert to indicate that it's currently handling an increased system load.

It's normal for appliances and connectors to consume more than 90% of system resources during heavy processing activity.

Alert text: <Number> of CPU/Memory/Disk usage on <Appliance/Connector> is too high.

Severity: High

Figure 96: Alert for connector system usage too high



Allowed Secure Workload virtual appliances: Secure Workload Ingest and Secure Workload Edge

Allowed connectors: All

## Connector Configuration Error

When you try to connect a configured connector to a configured server and the configuration fails, the system generates an alert to indicate a potential issue with the configuration after accepting and deploying it.

For example, The AnyConnect connector can take an LDAP configuration, validate and accept the configuration. However, during the normal operation, it is possible that the configuration is no longer valid.

Alert captures the scenario and indicates that you have to take corrective action to update the configuration.

Alert text: Cannot connect to <Appliance/Connector> server, check <Appliance/Connector> config.

Severity: High, Low

| Server            | Connector                |
|-------------------|--------------------------|
| LDAP server       | AnyConnect, F5, ISE, WDC |
| ISE server        | ISE                      |
| ServiceNow server | ServiceNow               |

Figure 97: Alert for config status error

The screenshot shows the Cisco Tetration Alerts interface. At the top, it says "CURRENT ALERTS" and "Default Monitoring". A notification bar at the top indicates: "You do not have an active license. The evaluation period will end on Thu Sep 10 2020 01:09:39 GMT+0000. Please notify admin." Below this, the "Alerts" section is filtered by "Status = ACTIVE". A table lists the alerts:

| Event Time | Status | Alert Text                                             | Severity | Type      | Actions |
|------------|--------|--------------------------------------------------------|----------|-----------|---------|
| 11:00 PM   | ACTIVE | Can't connect to LDAP server, please check LDAP config | HIGH     | CONNECTOR | z' O    |

Below the table, a "Details" panel is expanded, showing the following information:

- Appliance ID: 5ee314bf1bf0541577c6349e
- Appliance Ip: 172.29.142.63
- Deep Link: marge.tetrationanalytics.com/#/connectors/details/ANYCONNECT?id=5ee316f05411a65ca2d8f2fd
- Last Checkin At: Jun 12 2020 06:00:51 AM UTC
- Name: ANYCONNECT
- Reason: Invalid Credentials Original Error Text LDAP Result Code 49 Invalid Credentials 80090308 Ldap Err DSID 0 C 090446 Comment Accept Security Context Error Data 52 E V 2580
- Type: ANYCONNECT

Allowed Secure Workload virtual appliances: Secure Workload Ingest and Secure Workload Edge.

Allowed connectors: AnyConnect, F5, ISE, WDC, and ServiceNow.

## Connector UI Alert Details

Figure 98: Connector UI Alert details

| Details                |                                                                                                                                                                          |
|------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Appliance ID</b>    | 5ee314bf1bf0541577c6349e                                                                                                                                                 |
| <b>Appliance Ip</b>    | 172.29.142.63                                                                                                                                                            |
| <b>Deep Link</b>       | marge.tetrationanalytics.com/#/connectors/details/ANYCONNECT?id=5ee316f05411a65ca2d8f2fd                                                                                 |
| <b>Last Checkin At</b> | Jun 12 2020 06.56.28 AM UTC                                                                                                                                              |
| <b>Name</b>            | ANYCONNECT                                                                                                                                                               |
| <b>Reason</b>          | Invalid Credentials Original Error Text LDAP Result Code 49 Invalid Credentials 80090308 Ldap Err DSID 0 C 090446 Comment Accept Security Context Error Data 52 E V 2580 |
| <b>Type</b>            | ANYCONNECT                                                                                                                                                               |

## Alert Details

See [Common Alert Structure](#) for general alert structure and information about fields. The `alert_details` fields structure contains the following subfields for connector alerts.

| Field           | Type      | Description                                                          |
|-----------------|-----------|----------------------------------------------------------------------|
| Appliance ID    | String    | Appliance ID                                                         |
| Appliance IP    | String    | Appliance IP                                                         |
| Connector ID    | String    | Connector ID                                                         |
| Connector IP    | String    | Connector IP                                                         |
| Deep Link       | Hyperlink | Redirect to appliance/connector page                                 |
| Last CheckIn At | String    | Last checkin time                                                    |
| Name            | String    | Appliance/Connector name                                             |
| Reason          | String    | The reason that Appliance/Connector can't connect to Secure Workload |
| Type            | String    | Appliance/Connector type                                             |

## Example of Alert Details

After parsing `alert_details` as JSON (unstringified), it will display as follows.

```

{
 "Appliance ID": "5f1f3d26d674b01832c6792a",
 "Connector ID": "5f1f3e47baba512a70abee43",
 "Connector IP": "172.29.142.22",
 "Deep Link":
"bingo.tetrationanalytics.com/#/connectors/details/F5?id=5f1f3e47baba512a70abee43",
 "Last checkin at": "Aug 04 2020 20.37.33 PM UTC",
 "Name": "F5",
 "Reason": "Invalid Credentials (Original error text: LDAP Result Code 49 \"Invalid
Credentials\":)",
 "Type": "F5"
}

```

## Connector UI Alert Details

Figure 99: Connector UI Alert details

| Details                |                                                                                                                                                                             |
|------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Appliance ID</b>    | 5ee314bf1bf0541577c6349e                                                                                                                                                    |
| <b>Appliance Ip</b>    | 172.29.142.63                                                                                                                                                               |
| <b>Deep Link</b>       | marge.tetrationanalytics.com/#/connectors/details/ANYCONNECT?id=5ee316f05411a65ca2d8f2fd                                                                                    |
| <b>Last Checkin At</b> | Jun 12 2020 06.56.28 AM UTC                                                                                                                                                 |
| <b>Name</b>            | ANYCONNECT                                                                                                                                                                  |
| <b>Reason</b>          | Invalid Credentials Original Error Text LDAP Result Code 49 Invalid Credentials 80090308 Ldap Err DSID 0<br>C 090446 Comment Accept Security Context Error Data 52 E V 2580 |
| <b>Type</b>            | ANYCONNECT                                                                                                                                                                  |

## Life Cycle Management of Connectors

Connectors can be enabled, deployed, configured, troubleshooted, and deleted from Secure Workload directly.

### Enabling a Connector

From the Connectors page (**Manage > Connectors**), a connector can be selected and enabled. The connector can be deployed on a new virtual appliance (which has to be provisioned first and become *Active* before a connector can be enabled on it) or an existing virtual appliance. Once the virtual appliance is chosen, Secure Workload sends the rpm package for the connector to the appliance.

When Appliance Controller on the chosen appliance receives the rpm, it does the following:

1. Construct a Docker image using the rpm package received from Secure Workload. This Docker image includes the configuration required to communicate with Kafka topic on which appliance management messages are sent. This enables the service instantiated from this image to be able to send and receive messages for managing the corresponding connector.
2. Create a Docker container from the Docker image.



3. On Secure Workload Ingest appliance, the following additional tasks are performed.
  - A free slot is identified and the corresponding IP address is determined.
  - Connector listening ports (for example, 4729 and 4739 ports on NetFlow connector to receive flow records from NetFlow V9 or IPFIX enabled switches and routers), are exposed to the host on IP corresponding to the chosen slot.
  - A Docker volume is created and added to the container.
4. The Docker container is started and it executes the connector as a *supervisord* managed service. The service starts *Service Controller* as *tet-controller* which registers with Secure Workload and spawns the actual connector service.

**Figure 100: Docker Images**

```
[root@beretta-ingest-1 tetter]# docker images
```

| REPOSITORY                                             | TAG                      | IMAGE ID     | CREATED            | SIZE  |
|--------------------------------------------------------|--------------------------|--------------|--------------------|-------|
| netflow_sensor-3.4.2.52222.maarumug.mrpm.build-netflow | 5d379fac6e37d85f2bdeff45 | 2635145b44c8 | About a minute ago | 650MB |
| tet-service-base                                       | latest                   | 6be171bbe648 | 4 days ago         | 519MB |
| artifacts.tet.wtf:6555/centos                          | 7.3.1611                 | c5d48e81b986 | 4 months ago       | 192MB |

```
[root@beretta-ingest-1 tetter]#
```

**Figure 101: Docker Volumes**

```
[root@beretta-ingest-1 tetter]# docker volume ls
```

| DRIVER | VOLUME NAME                                                      |
|--------|------------------------------------------------------------------|
| local  | 373b5b682a96547bf2526784a5943c2f110593b88485b996e7259fa4e314c439 |

```
[root@beretta-ingest-1 tetter]#
```

**Figure 102: Docker containers**

```
[root@beretta-ingest-1 tetter]# docker ps
```

| CONTAINER ID | IMAGE                                                                           | COMMAND                                                    | CREATE             |
|--------------|---------------------------------------------------------------------------------|------------------------------------------------------------|--------------------|
| D            |                                                                                 |                                                            |                    |
| 2c7a7ed4f853 | netflow_sensor-3.4.2.52222.maarumug.mrpm.build-netflow:5d379fac6e37d85f2bdeff45 | "/usr/bin/supervisor..."                                   | About a minute ago |
| Up           | About a minute                                                                  | 172.29.142.26:4729->4729/udp, 172.29.142.26:4739->4739/udp |                    |
|              |                                                                                 | nf-5d379fac6e37d85f2bdeff45                                |                    |

```
[root@beretta-ingest-1 tetter]#
```

Figure 103: Slot used by the Docker container and list of exposed ports

```
[root@beretta-ingest-1 tetter]# cat /local/tetration/appliance/appliance.conf
{
 "type": "TETRATION_DATA_INGEST",
 "slots": [
 {
 "available": false,
 "index": 0,
 "mapped_ip": "172.29.142.26",
 "share_volume": true,
 "count": 1,
 "service_containers": {
 "5d379fac6e37d85f2bdeff45": {
 "connector_id": "5d379fac6e37d85f2bdeff44",
 "service_id": "5d379fac6e37d85f2bdeff45",
 "container_id": "2c7a7ed4f853e85f3d620c663f1c7f5395b53b9dd6696276ac439d34fe142bf1",
 "image_name": "netflow_sensor-3.4.2.52222.maarumug.mrpm.build-netflow:5d379fac6e37d85f2bdeff45",
 "container_name": "nf-5d379fac6e37d85f2bdeff45",
 "service_type": "NETFLOW_SENSOR",
 "ip_bindings": [
 {
 "ip": "172.29.142.26",
 "port": "4729",
 "protocol": "udp"
 },
 {
 "ip": "172.29.142.26",
 "port": "4739",
 "label": 1,
 "protocol": "udp"
 }
]
 },
 "volume_id": "373b5b682a96547bf2526784a5943c2f110593b88485b996e7259fa4e314c439"
 }
 },
 {
 "available": true,
 "index": 1,
 "mapped_ip": "172.29.142.27",
 "share_volume": true,
 "count": 0,
 "service_containers": null
 },
 {
 "available": true,
 "index": 2,
 "mapped_ip": "172.29.142.28",
 "share_volume": true,
 "count": 0,
 "service_containers": null
 }
]
}
[root@beretta-ingest-1 tetter]#
```

Figure 104: List of ports exposed by Docker container

```
[root@beretta-ingest-1 tetter]# docker port 2c7a7ed4f853
4729/udp -> 172.29.142.26:4729
4739/udp -> 172.29.142.26:4739
[root@beretta-ingest-1 tetter]#
```

Figure 105: Docker Volume mounted to a container

```
[root@beretta-ingest-1 tetter]# docker inspect --format='{{json .Mounts}}' 2c7a7ed4f853
[{"Type": "volume", "Name": "373b5b682a96547bf2526784a5943c2f110593b88485b996e7259fa4e314c439", "Source": "/var/lib/docker/volumes/373b5b682a96547bf2526784a5943c2f110593b88485b996e7259fa4e314c439/_data", "Destination": "/local/tetration", "Driver": "local", "Mode": "z", "RW": true, "Propagation": ""}]
[root@beretta-ingest-1 tetter]#
```

*Service Controller* is responsible for the following functions:

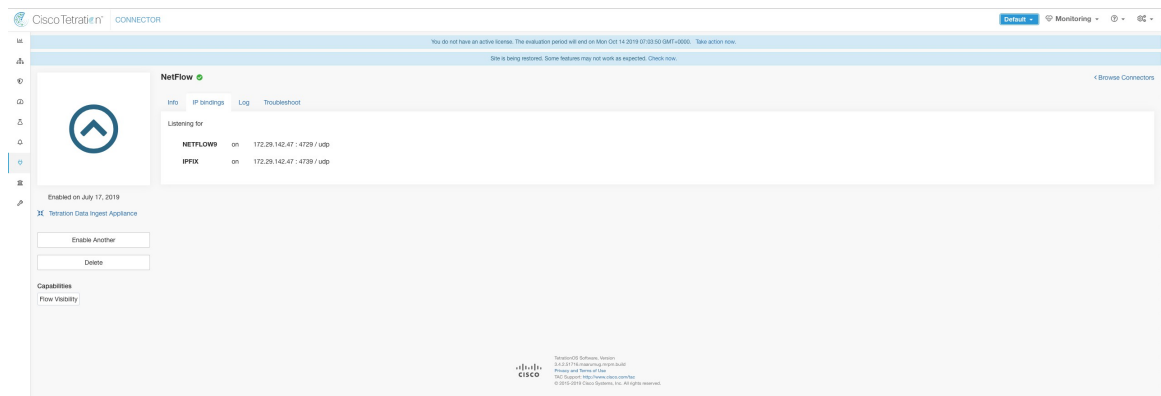
1. **Registration:** registers the connector with Secure Workload. Until the connector is registered and marked *Enabled*, no configuration updates can be pushed to the connector. When Secure Workload receives a registration request for a connector, it updates the state of the connector to *Enabled*.
2. **Configuration updates on connector:** tests and applies configuration updates on the connector. For more information, see [Configuration Management on Connectors and Virtual Appliances](#).
3. **Troubleshooting commands on connector:** executes allowed commands on the connector service for troubleshooting and debugging issues on the connector service. For more information, see [Troubleshooting](#).
4. **Heartbeats:** periodically sends heartbeats and statistics to Secure Workload to report the health of the connector. For more information, see [Monitoring a Virtual Appliance](#).

## Viewing Connector-Related Information

**Enabled Connectors:** A list of all enabled connectors can be found by clicking **Manage > Connectors** in the navigation bar at the left side of the window.

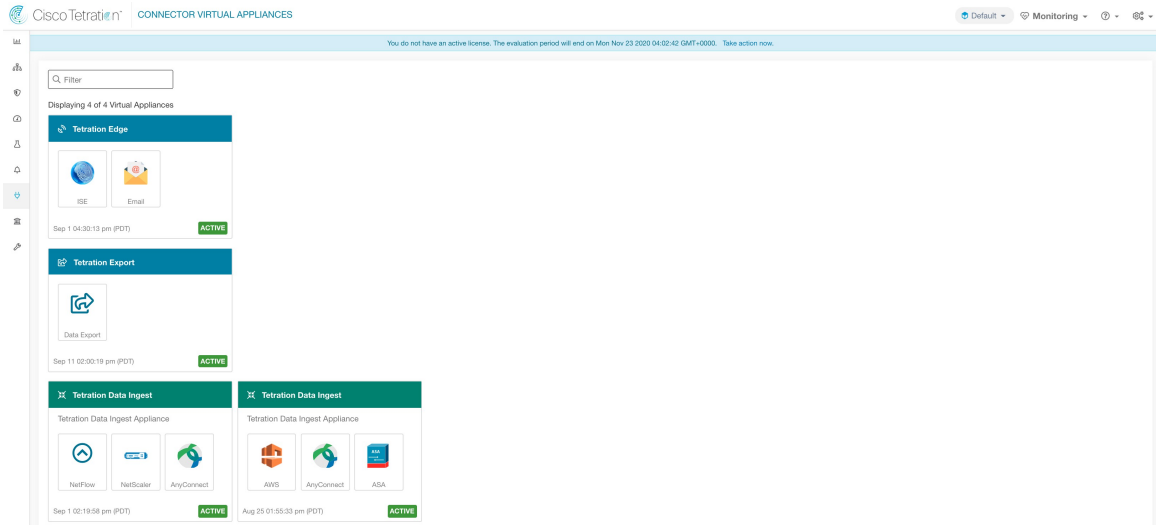
**Connector Details:** Details about the connector can be fetched by clicking on the connector. This page shows the port bindings -if any- that can be used to configure upstream network elements to send telemetry data to the correct IP and port.

**Figure 106: Connector details**



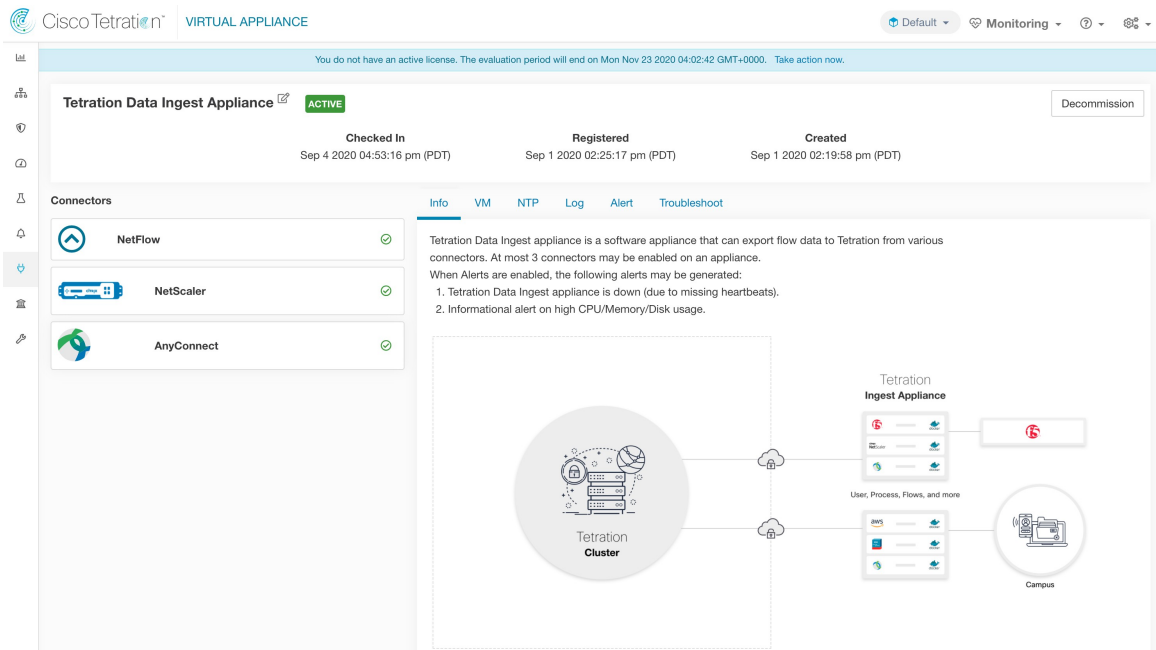
**Deployed Virtual Appliances:** A list of deployed virtual appliances can be found at: **Manage > Virtual Appliances**.

Figure 107: List of deployed virtual appliances



**Virtual Appliance Details** A detailed view of an appliance can be fetched by clicking on the appliance directly from *List of deployed virtual appliances*.

Figure 108: Appliance details and the connectors



## Deleting a Connector

When a connector is deleted, Appliance Controller on the appliance where the connector is enabled will receive a message to remove the services created for the connector. Appliance Controller does the following:

1. Stop the Docker container corresponding to the connector.

2. Remove the Docker container.
3. If the connector is deployed on a Secure Workload Ingest appliance and it exposes ports, then remove the Docker volume that was mounted to the container.
4. Remove the Docker image that was created for the connector.
5. Finally, send a message back to Secure Workload indicating the status of the delete request.

## Monitoring a Connector

Connector services periodically send heartbeats and statistics to Secure Workload. The heartbeat interval is 5 minutes. The heartbeat messages include statistics about the health of the service include system statistics, process statistics, and statistics about how many messages sent/received/error-ed over the Kafka topic that is used for the appliance management. In addition, it includes statistics exported by the connector service itself.

All metrics are available in *Digger* (OpenTSDB) and are annotated with appliance ID, connector ID, and root scope name. Additionally, Grafana dashboards for connector services are also available for important metrics from the service.

## Virtual Appliances for Connectors

Most connectors are deployed on Secure Workload virtual appliances. You will deploy required virtual appliances on an ESXi host in VMware vCenter using the OVA templates or on other KVM-based hypervisors using the QCOW2 image. The procedure to deploy virtual appliances is described in [Deploying a Virtual Appliance](#).

## Types of Virtual Appliances

Each connector that requires a virtual appliance can be deployed on one of two types of virtual appliances.

### Secure Workload Ingest

Secure Workload Ingest appliance is a software appliance that can export flow observations to Secure Workload from various connectors.

#### Specification

- Number of CPU cores: 8
- Memory: 8 GB
- Storage: 250 GB
- Number of network interfaces: 3
- Number of connectors on one appliance: 3
- Operating System: CentOS 7.9 (Secure Workload 3.8.1.19 and earlier), AlmaLinux 9.2 (Secure Workload 3.8.1.36 and later)

See important limits at [Connectors](#).



**Note** Each root scope on Secure Workload can have at most 100 Secure Workload Ingest appliances deployed.

**Figure 109: Secure Workload Ingest appliance**

**Tetration Data Ingest Appliance** ACTIVE Decommission

**Checked In**  
Sep 4 2020 04:45:59 pm (PDT)

**Registered**  
Aug 25 2020 06:47:59 pm (PDT)

**Created**  
Aug 25 2020 01:55:33 pm (PDT)

**Connectors**

- AWS** ✓
- AnyConnect** ✓
- F5** ✓

**Info** | VM | NTP | Log | Alert | Troubleshoot

Tetration Data Ingest appliance is a software appliance that can export flow data to Tetration from various connectors. At most 3 connectors may be enabled on an appliance.

When Alerts are enabled, the following alerts may be generated:

1. Tetraton Data Ingest appliance is down (due to missing heartbeats).
2. Informational alert on high CPU/Memory/Disk usage.

**Tetration Cluster** | **Tetration Ingest Appliance** | **Campus**

User, Process, Flows, and more

Secure Workload Ingest appliance allows at most 3 connectors to be enabled on an appliance. There can be more than one instance of the same connector enabled on the same appliance. For the ERSPAN Ingest appliance three ERSPAN connectors are always automatically provisioned. Many of the connectors deployed on Ingest appliance collect telemetry from various points in the network, these connectors need to listen on specific ports on the appliance. Each connector is therefore bound to one of the IP address and the default ports on which the connector should be listening to collect telemetry data. As a result, each IP address is essentially a slot that a connector occupies on the appliance. When a connector is enabled, a slot is taken (thereby, the IP corresponding to the slot). And, when a connector is disabled, the slot occupied by the connector is released (thereby, the IP corresponding to the slot). See the *Secure Workload Ingest appliance slots* for how the ingest appliance maintains the state of the slots.

Figure 110: Secure Workload Ingest appliance slots

```
[root@beretta-ingest-1 tetter]# cat /local/tetration/appliance/appliance.conf
{
 "type": "TETRATION_DATA_INGEST",
 "slots": [
 {
 "available": false,
 "index": 0,
 "mapped_ip": "172.29.142.26",
 "share_volume": true,
 "count": 1,
 "service_containers": {
 "5d379fac6e37d85f2bdeff45": {
 "connector_id": "5d379fac6e37d85f2bdeff44",
 "service_id": "5d379fac6e37d85f2bdeff45",
 "container_id": "2c7a7ed4f853e85f3d620c663f1c7f5395b53b9dd6696276ac439d34fe142bf1",
 "image_name": "netflow_sensor-3.4.2.52222.maarumug.mrpm.build-netflow:5d379fac6e37d85f2bdeff45",
 "container_name": "nf-5d379fac6e37d85f2bdeff45",
 "service_type": "NETFLOW_SENSOR",
 "ip_bindings": [
 {
 "ip": "172.29.142.26",
 "port": "4729",
 "protocol": "udp"
 },
 {
 "ip": "172.29.142.26",
 "port": "4739",
 "label": 1,
 "protocol": "udp"
 }
]
 },
 "volume_id": "373b5b682a96547bf2526784a5943c2f110593b88485b996e7259fa4e314c439"
 }
 },
 {
 "available": true,
 "index": 1,
 "mapped_ip": "172.29.142.27",
 "share_volume": true,
 "count": 0,
 "service_containers": null
 },
 {
 "available": true,
 "index": 2,
 "mapped_ip": "172.29.142.28",
 "share_volume": true,
 "count": 0,
 "service_containers": null
 }
]
}
[root@beretta-ingest-1 tetter]#
```

### Allowed Configurations

- *NTP*: Configure NTP on the appliance. For more information, see [NTP Configuration](#).
- *Log*: Configure Logging on the appliance. For more information, see [Log Configuration](#).

## Secure Workload Edge

Secure Workload Edge is a control appliance that streams alerts to various notifiers and collects inventory metadata from network access controllers such as Cisco ISE. In a Secure Workload Edge appliance, all alert notifier connectors (such as Syslog, Email, Slack, PagerDuty, and Kinesis), ServiceNow connector, Workload AD connector, and ISE connector can be deployed.

## Specification

- Number of CPU cores: 8
- Memory: 8 GB
- Storage: 250 GB
- Number of network interfaces: 1
- Number of connectors on one appliance: 8
- Operating System: CentOS 7.9 (Secure Workload 3.8.1.19 and earlier), AlmaLinux 9.2 (Secure Workload 3.8.1.36 and later)

See important limits at [Connectors](#).



**Note** Each root scope on Secure Workload can have at most one Secure Workload Edge appliance deployed.

**Figure 111: Secure Workload Edge appliance**

The connectors deployed on Secure Workload Edge appliance do not listen on ports. Therefore, the Docker containers instantiated for the connectors on Secure Workload Edge appliance do not expose any ports to the host.

## Allowed Configurations

- *NTP*: Configure NTP on the appliance. For more information, see [NTP Configuration](#).
- *Log*: Configure Logging on the appliance. For more information, see [Log Configuration](#).



## Deploying a Virtual Appliance

Deploy virtual appliances on an ESXi host in VMware vCenter or other KVM-based hypervisors such as Red Hat Virtualization. This procedure prompts you to download a virtual appliance OVA template or QCOW2 image from the [Cisco Software Download page](#).



**Attention** To deploy a Secure Workload external appliance, the ESXi host where the appliance is created should have the following specifications:

- **vSphere**: version 5.5 or better.
- **CPU**: at least 2.2 GHz per core, and has enough reservable capacity for the appliance.
- **Memory**: at least enough space to fit the appliance.

To deploy a virtual appliance to collect data from connectors:

### Procedure

- Step 1** In the Secure Workload web portal, choose **Manage > Virtual Appliances** from the navigation bar on the left.
- Step 2** Click **Enable a Connector**. The type of virtual appliance you must deploy depends on the type of connector you are enabling.
- Step 3** Click the type of connector for which you must create the virtual appliance. For example, click the NetFlow connector.
- Step 4** On the connector page, click **Enable**.
- Step 5** If you see a notice telling you that you must deploy a virtual appliance, click **Yes**. If you do not see this notice, you may already have a virtual appliance that this connector can use, in which case you do not need to perform this procedure.
- Step 6** Click the link to download the OVA template or QCOW2 image for the virtual appliance. Leave the wizard open on your screen without clicking anything else.
- Step 7** Use the downloaded:
  - OVA to deploy a new OVF template on a designated ESXi host.
    - To deploy an OVA on a vSphere Web Client, follow the instructions on how to [Deploy an OVF Template](#).
    - Ensure that the deployed VM settings match the recommended configuration for the virtual appliance type.
    - **Do not power on the deployed VM**
  - QCOW2 image to create a new VM on KVM hypervisors such as Red Hat Virtualization.
- Step 8** After the VM is deployed, but before you power it on, return to the virtual appliance deployment wizard in the Secure Workload web portal.
- Step 9** Click **Next** in the virtual appliance deployment wizard.

**Step 10** Configure the virtual appliance by providing IP addresses, gateways, hostname, DNS, proxy server settings and docker bridge subnet configuration. See the screenshot for *Configuring the VM with network parameters*.

- If the appliance must use proxy server to reach Secure Workload, check the box *Use proxy server to connect to Secure Workload*. If this is not set correctly, connectors may not be able to communicate with Secure Workload for control messages, register connectors, and send flow data to the Secure Workload collector.
- If the IP addresses and gateways of the appliance conflict with the default docker bridge subnet (172.17.0.1/16), the appliance can be configured with a customized docker bridge subnet that is specified in *Docker Bridge (CIDR format)* field. This requires appliance OVA 3.3.2.16 or later.

**Step 11** Click **Next**.

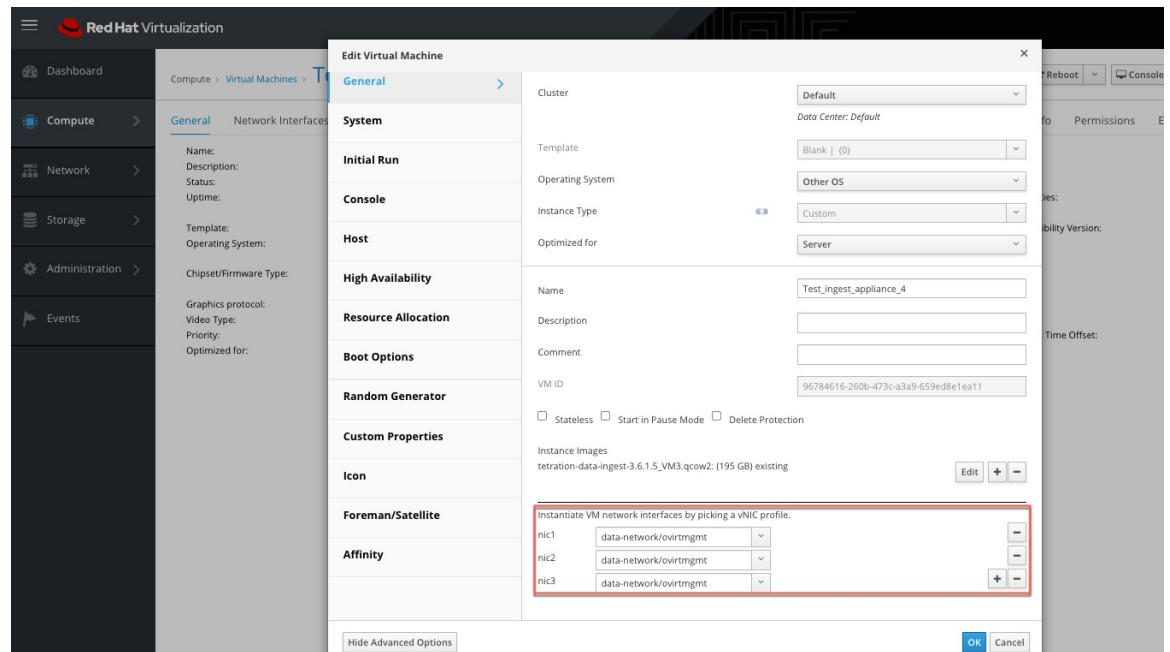
**Step 12** In the next step, a VM configuration bundle will be generated and available for download. Download the VM configuration bundle. See the screenshot for *Download the VM configuration bundle*.

**Step 13** Upload the VM configuration bundle to the datastore corresponding to the target ESXi host or other virtualization host.

**Step 14** [Applicable only when using QCOW2 image] Complete the following configurations on the other virtualization host where you have uploaded the VM configuration bundle:

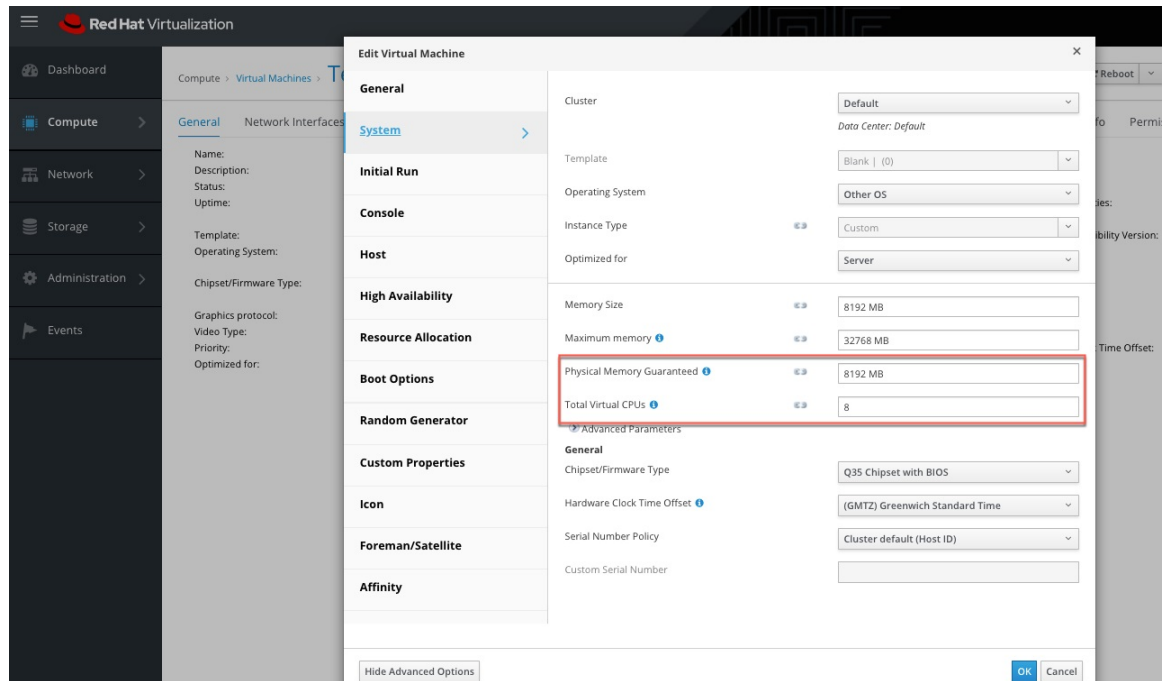
- For ingest appliances, configure three network interfaces.

**Figure 112: Example of Configuring Network Interfaces in KVM-Based Environments**



- In the memory allocation, specify the minimum requirement of 8192 MB of RAM.
- Specify the total number of virtual CPUs to be 8.

Figure 113: Example of Configuring System Resources in KVM-Based Environments



**Step 15** Edit the VM settings and mount the VM configuration bundle from the datastore to the CD/DVD drive. Make sure to select **Connect at Power On** check box.

**Step 16** Power on the deployed VM.

**Step 17** When the VM boots up and configures itself, it connects back to Secure Workload. This may take a few minutes. The appliance status on Secure Workload should transition from *Pending Registration* to *Active*. See the screenshot for *Secure Workload Ingest appliance in Pending Registration state*.

**Note** We do not recommend vMotion to be enabled for Secure Workload external appliances.

**Note** We recommend using Secure Workload external appliance OVAs as-is and to reserve 8 vCPU cores and 8192 MB of memory for QCOW2 images to deploy VMs. If sufficient resources are not available, the VM setup script would fail after the boot.

When the appliance is *Active*, connectors can be enabled and deployed on it.

Figure 114: Deploying a Secure Workload Ingest Appliance

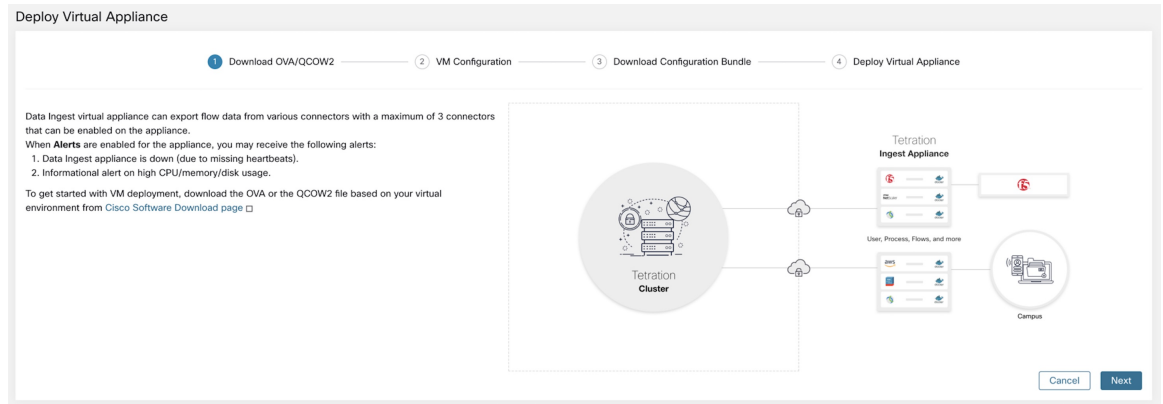


Figure 115: Configuring the VM with Network Parameters

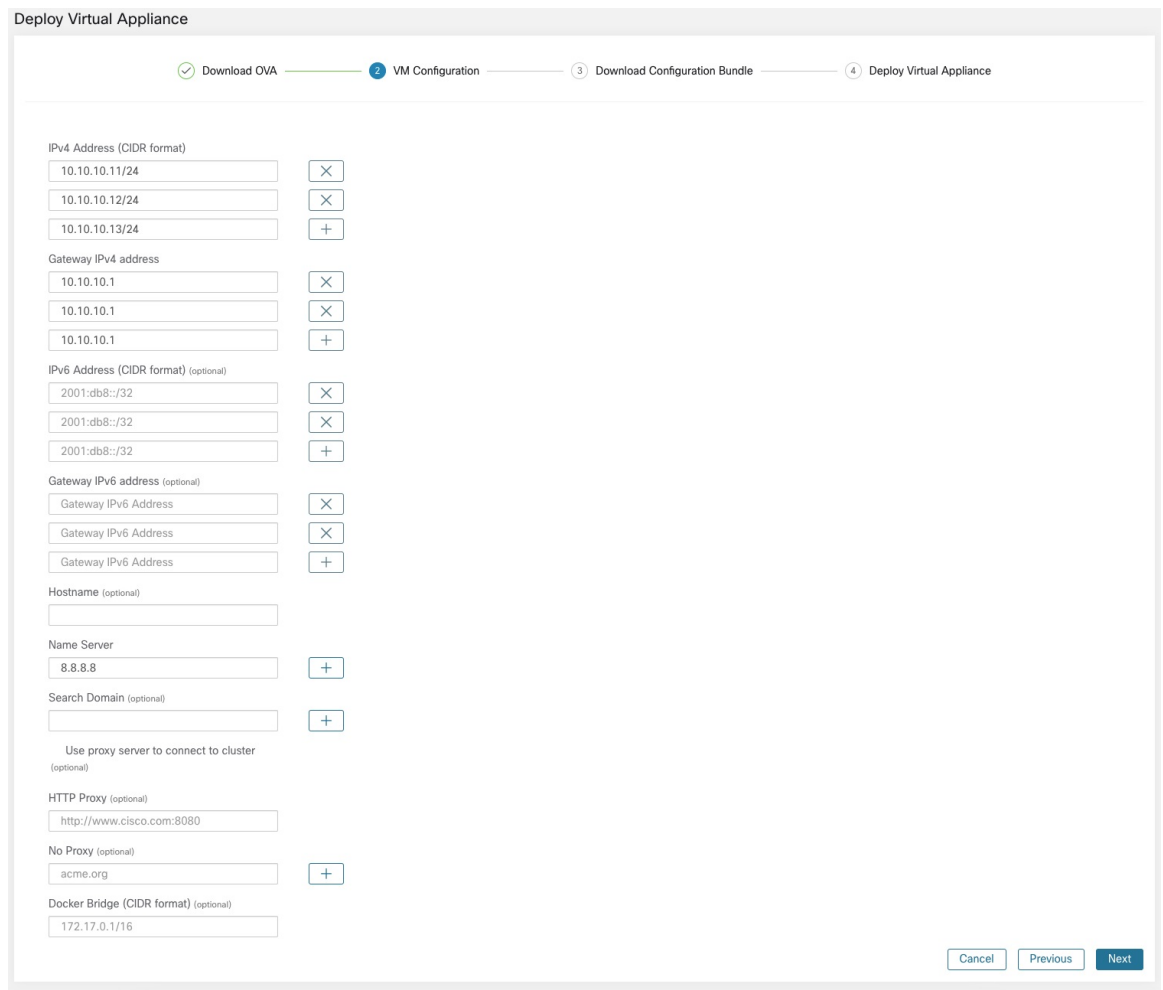


Figure 116: Download the VM Configuration Bundle

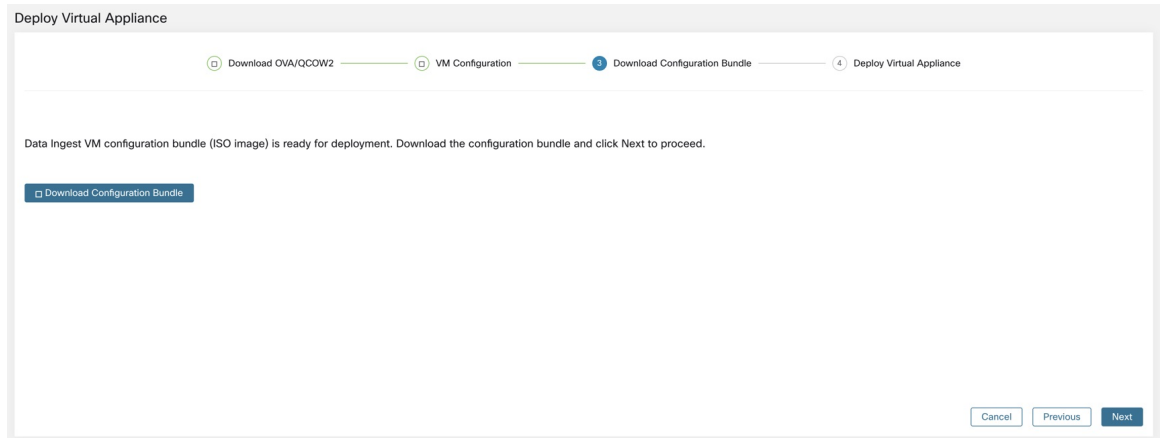


Figure 117: Deploy the VM

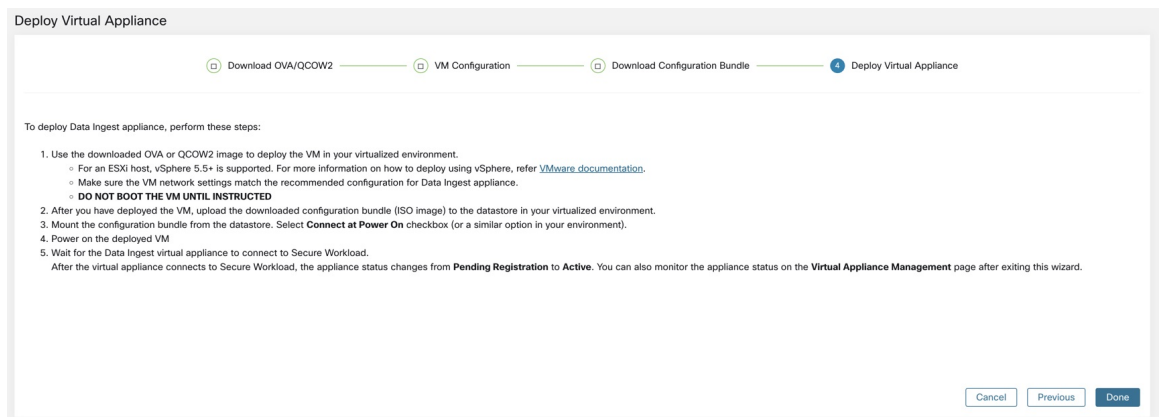
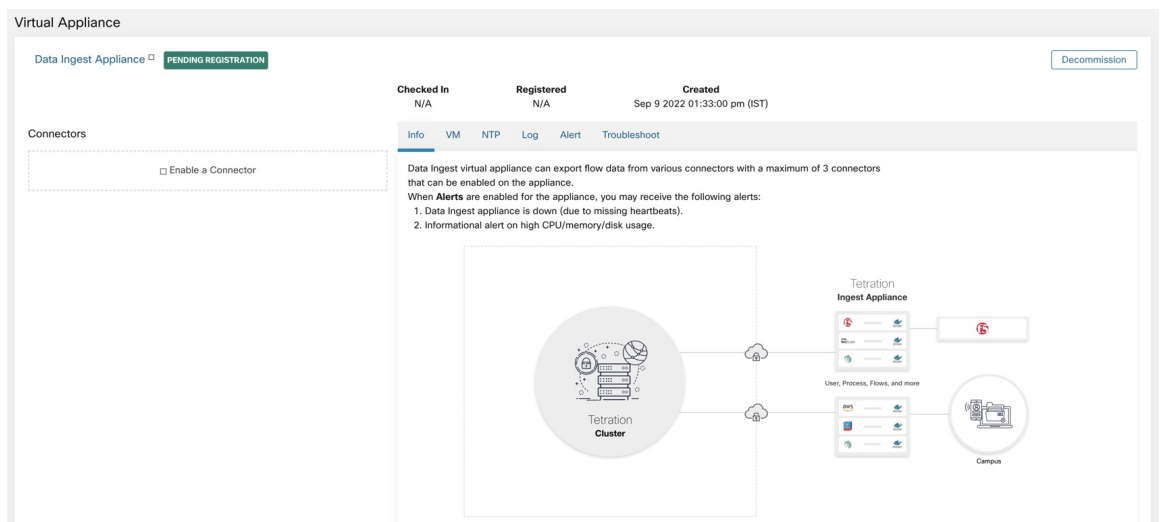


Figure 118: Secure Workload Ingest Appliance in Pending Registration State



When a virtual appliance is deployed and booted up for the first time, *tet-vm-setup* service executes and sets up the appliance. This service is responsible for the following tasks.

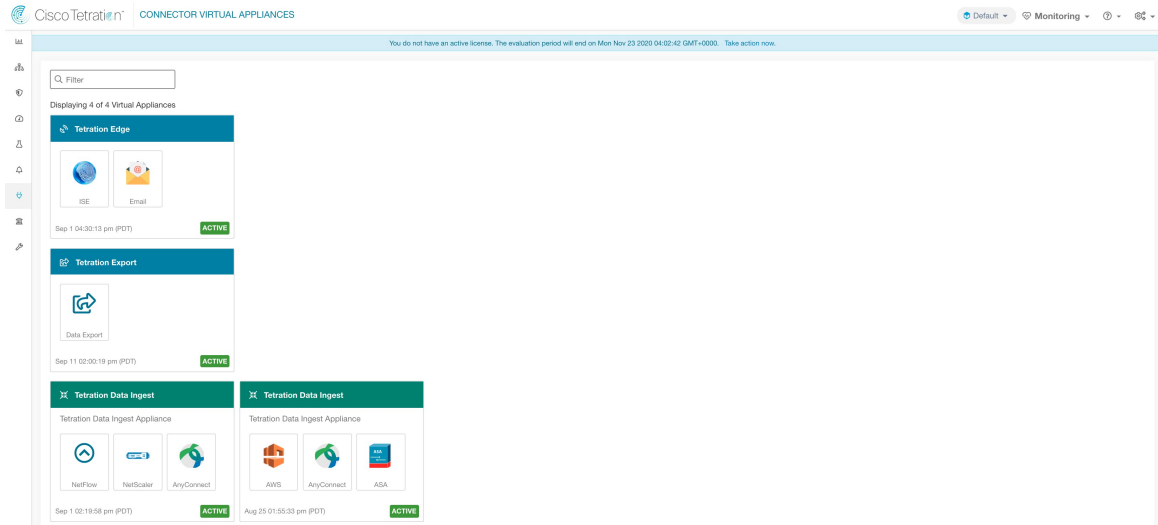
- a. **Validate the appliance:** validate the appliance for mandatory resource requirements for the type of the virtual appliance deployed.
- b. **IP address assignment:** assign IP addresses to all the network interfaces provisioned on the appliance.
- c. **Hostname assignment:** assign hostname for the appliance (if hostname is configured).
- d. **DNS configuration:** update the DNS *resolv.conf* file (if name server and/or search-domain parameters are configured).
- e. **Proxy server configuration:** update *HTTPS\_PROXY* and *NO\_PROXY* settings on the appliance (if provided).
- f. **Prepare appliance:** copies cert bundle for the Kafka topic over which appliance management messages are sent and received.
- g. **Install appliance controller:** install and bring up *Appliance Controller* which is managed by *supervisord* as *tet-controller* service.

When *tet-controller* is instantiated, it takes over the management of the appliance. This service is responsible for the following functions:

- a. **Registration:** registers the appliance with Secure Workload. Until the appliance is registered, no connectors can be enabled on the appliance. When Secure Workload receives a registration request for an appliance, it updates the state of the appliance to *Active*.
- b. **Deploying a connector:** deploys a connector as a Docker service on the appliance. For more information, see [Enabling a Connector](#).
- c. **Deleting a connector:** stops and removes the Docker service and the corresponding Docker image from the appliance. For more information, see [Deleting a Connector](#).
- d. **Configuration updates on appliances:** tests and applies configuration updates on the appliance. For more information, see [Configuration Management on Connectors and Virtual Appliances](#).
- e. **Troubleshooting commands on appliances:** executes allowed set of commands on the appliances for troubleshooting and debugging issues on the appliance. For more information, see the [Troubleshooting](#).
- f. **Heartbeats:** periodically sends heartbeats and statistics to Secure Workload to report the health of the appliance. For more information, see [Monitoring a Virtual Appliance](#).
- g. **Pruning:** periodically prune all Docker resources that are unused or dangling in order to recover storage space. This task is executed when every 24 hours.
- h. **Decommissioning the appliance:** decommissions and deletes all Docker instances from the appliance. For more information, see [Decommissioning a Virtual Appliance](#).

The list of deployed virtual appliances can be found at: **Manage > Virtual Appliances**

Figure 119: List of Deployed Virtual Appliances



## Decommissioning a Virtual Appliance

A virtual appliance can be decommissioned from Secure Workload. When an appliance is decommissioned, the following actions are triggered.

1. All configurations on the appliance and the connectors enabled on the appliance are removed.
2. All the connectors enabled on the appliance are deleted.
3. The appliance is marked *Pending Delete*.
4. When the appliance replies back with a successful delete response, appliance Kafka topic and certs are deleted.



**Note** Decommissioning an appliance cannot be undone. To restore the appliance and the connectors, a new appliance should be deployed and the connectors should be enabled on the new appliance.

## Monitoring a Virtual Appliance

Secure Workload virtual appliances periodically send heartbeats and statistics to Secure Workload. The heartbeat interval is 5 minutes. The heartbeat messages include statistics about the health of the appliance include system statistics, process statistics, and statistics about how many messages sent/received/error-ed over the Kafka topic that is used for the appliance management.

All metrics are available in *Digger* (OpenTSDB) and are labelled with appliance ID and root scope name. Additionally, Grafana dashboards for *Appliance Controller* are also available for important metrics from the appliance.

## Security Considerations

The Ingest/Edge Virtual Machine's guest Operating System is CentOS 7.9, from which OpenSSL server/clients packages were removed. Therefore, the only way to access the appliance is via its console.




---

**Note** CentOS 7.9 is the guest operating system for Ingest and Edge virtual appliances in Secure Workload 3.8.1.19 and earlier releases. Starting Secure Workload 3.8.1.36, the operating system is AlmaLinux 9.2.

---

The containers run a centos:7.9.2009 based Docker image. Most the containers are run with the base privileges (no-privileged option), except for ERSPAN container, which has the NET\_ADMIN capability.




---

**Note** Starting Secure Workload 3.8.1.36, the containers run almalinux/9-base:9.2.

---

In the unlikely case a container is compromised, the VM guest OS should not be compromisable from inside the container.

## Configuration Management on Connectors and Virtual Appliances

Configuration updates can be pushed to appliances and connectors from Secure Workload. The appliance should have registered successfully with Secure Workload and be *Active* before configuration updates can be initiated. Similarly, the connectors should have registered with Secure Workload before configuration updates can be initiated on the connector services.

There are three modes of configuration updates possible in appliances and connectors.

1. **Test and Apply:** Test the configuration and on successful test, commit the configuration.
2. **Discovery:** Test the configuration, and on successful test, discovery additional properties that can be enabled for the configuration.
3. **Remove:** Remove the configuration.




---

**Note** ERSPAN appliance and connector do not support configuration updates.

---

## Test and Apply

Configurations that support *Test and Apply* mode verify the configuration before applying (committing) the configuration on the desired appliance and/or connector.

## NTP Configuration

NTP configuration allows the appliance to synchronize the clock with the specified NTP server(s).



| Parameter Name     | Type              | Description                                                                                     |
|--------------------|-------------------|-------------------------------------------------------------------------------------------------|
| <b>Enable NTP</b>  | checkbox          | Should NTP sync be enabled?                                                                     |
| <b>NTP Servers</b> | listof<br>strings | List of NTP servers. At least one server should be given and at most 5 servers may be provided. |

**Test:** Test if a UDP connection can be made to the given NTP servers on port 123. If an error occurs for any of the NTP servers, do not accept the configuration.

**Apply:** Update `/etc/ntp.conf` and restart `ntpd` service using `systemctl restart ntpd.service`. Here is the template for generating the `ntp.conf`

```
--- GENERAL CONFIGURATION ---
server <ntp-server>
...
server 127.127.1.0
fudge 127.127.1.0 stratum 10
Drift file
driftfile /etc/ntp/drift
```



**Note** Applicable to Secure Workload 3.8.1.19 and earlier.

For Secure Workload 3.8.1.36 and later, update `/etc/chrony.conf` and restart `chronyd` service using `systemctl restart chronyd.service`. Here is the template for generating the `chrony.conf`

```
Secure Workload appliance chrony.conf.
server <ntp-server> iburst
...
driftfile /var/lib/chrony/drift
makestep 1.0 3
rtcsync
```

**Allowed Cisco Secure Workload virtual appliances:** All

**Allowed connectors:** None

Figure 120: Error while testing NTP configuration

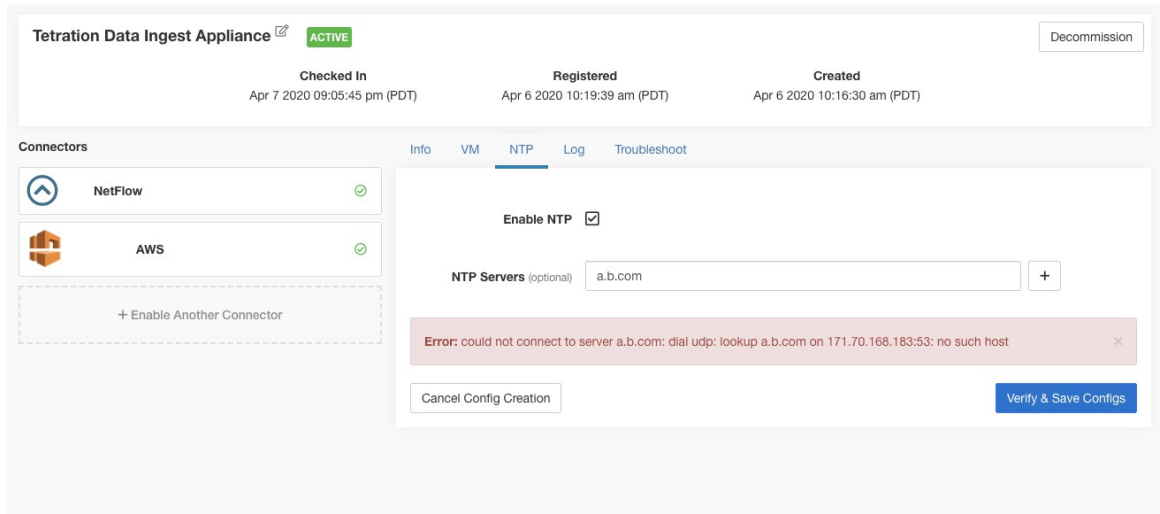


Figure 121: NTP configuration with valid NTP servers

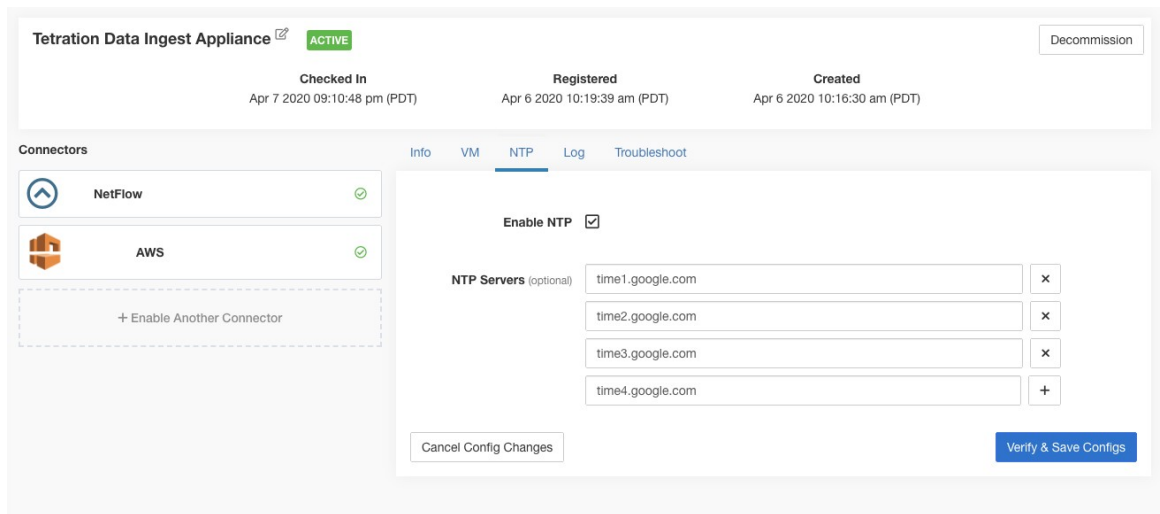
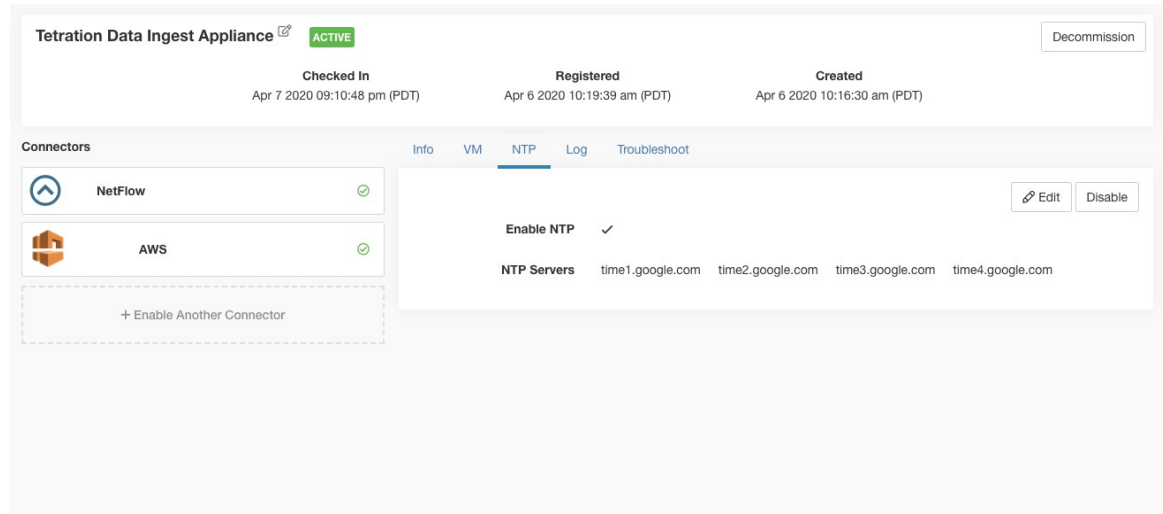


Figure 122: NTP configuration verified and applied



## Log Configuration

Log configuration updates the log levels, maximum size of the log files, and log rotation parameters on the appliance and/or connector. If the configuration update is triggered on the appliance, appliance controller log settings are updated. On the other hand, if the configuration update is triggered on a connector, service controller and service log settings are updated.

| Parameter Name                     | Type            | Description                                             |
|------------------------------------|-----------------|---------------------------------------------------------|
| <b>Logging level</b>               | dropdown        | Logging level to be set                                 |
|                                    | • <i>debug</i>  | Debug log level                                         |
|                                    | • <i>info</i>   | Informational log level                                 |
|                                    | • <i>warn</i>   | Warning log level                                       |
| • <i>error</i>                     | Error log level |                                                         |
| <b>Max log file size (in MB)</b>   | number          | Maximum size of a log file before log rotation kicks in |
| <b>Log rotation (in days)</b>      | number          | Maximum age of a log file before log rotation kicks in  |
| <b>Log rotation (in instances)</b> | number          | Maximum instances of log files kept                     |

**Test:** No op.

**Apply:** If the configuration is triggered on an appliance, update the configuration file of *tet-controller* on the appliance. If the configuration is triggered on a connector, update the configuration files of *tet-controller* and the service managed by the controller on the Docker container responsible for the connector.

**Allowed Secure Workload virtual appliances:** All

**Allowed connectors:** NetFlow, NetScaler, F5, AnyConnect, ISE, ASA, and Meraki.

**Figure 123: Log configuration on the appliance**

The screenshot shows the configuration page for a 'Tetration Data Ingest Appliance'. At the top, it indicates the appliance is 'ACTIVE' and provides dates for 'Checked In' (Apr 7 2020 09:05:45 pm (PDT)), 'Registered' (Apr 6 2020 10:19:39 am (PDT)), and 'Created' (Apr 6 2020 10:16:30 am (PDT)). A 'Decommission' button is in the top right. Below this, there are tabs for 'Connectors', 'Info', 'VM', 'NTP', 'Log', and 'Troubleshoot'. The 'Log' tab is selected. On the left, under 'Connectors', 'NetFlow' and 'AWS' are listed with green checkmarks, and there is a '+ Enable Another Connector' button. The main configuration area has a 'Logging Level' dropdown menu open, showing options: 'Choose one', 'debug' (selected), 'info', 'warn', and 'error'. Below the dropdown are fields for 'Max Log File Size (in MB)', 'Log Rotation (in days)', and 'Log Rotation (in instances)' (set to 20). At the bottom, there are 'Cancel Config Creation' and 'Verify & Save Configs' buttons.



**Note** Since all alert notifier Connectors (Syslog, Email, Slack, PagerDuty, and Kinesis) run on a single Docker service (Secure Workload Alert Notifier) on Secure Workload Edge, it is not possible to update the log config of a connector without impacting the config of another alert notifier connector. The log configurations of Secure Workload Alert Notifier (TAN) Docker service on Secure Workload Edge appliance can be updated using an allowed command.

See [Update Alert Notifier Connector Log Configuration](#) for more details.

## Endpoint Configuration

Endpoint configuration specifies the inactivity timeout for endpoints on AnyConnect and ISE connectors. When an endpoint times out, the connector stops checking in with Secure Workload and purges the local state for the endpoint on the connector.

| Parameter Name                                     | Type   | Description                                                                                                                                                         |
|----------------------------------------------------|--------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>InactivityTimeout for Endpoints(in minutes)</b> | number | Inactivity timeout for endpoints published by AnyConnect / ISE connectors. On timeout, the endpoint will not longer checkin Secure Workload. Default is 30 minutes. |

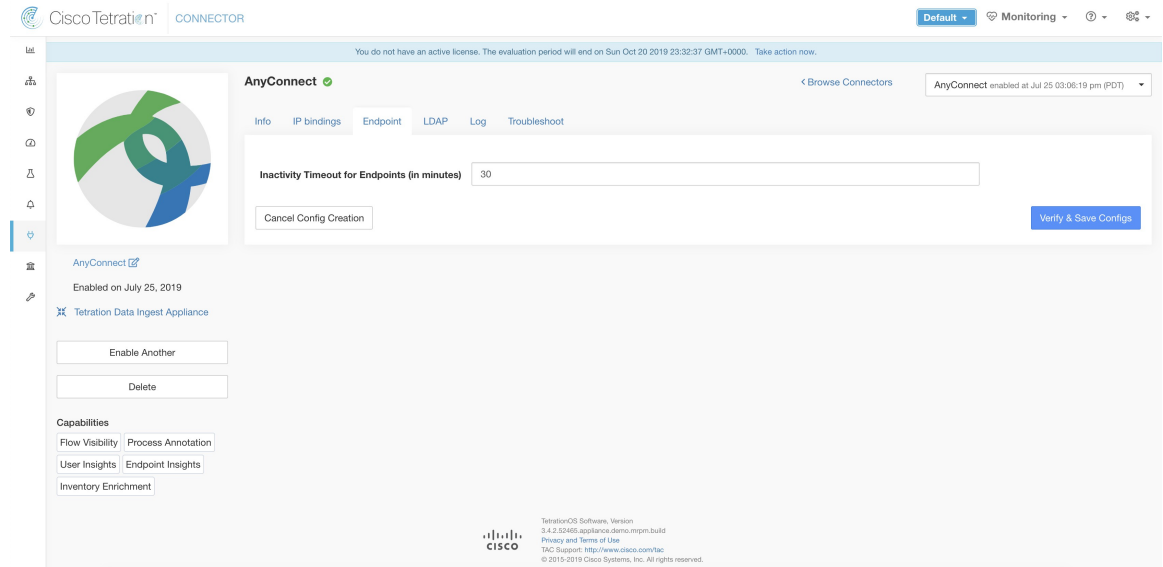
**Test** : No op.

**Apply** : Update the configuration file of the connector with the new value

**Allowed Secure Workload virtual appliances:** None

**Allowed connectors:** AnyConnect and ISE

Figure 124: Endpoint inactivity timeout configuration on AnyConnect connector



## Slack Notifier Configuration

Default configuration for publishing Secure Workload alerts on Slack.

| Parameter Name    | Type   | Description                                                       |
|-------------------|--------|-------------------------------------------------------------------|
| Slack Webhook URL | string | Slack webhook on which Secure Workload alerts should be published |

**Test:** Send a test alert to Slack using the webhook. If the alert is posted successfully, the test passes.

**Apply:** Update configuration file of the connector with the specified parameters.

**Allowed Secure Workload virtual appliances:** None

**Allowed connectors:** Slack

## PagerDuty Notifier Configuration

Default configuration for publishing Secure Workload alerts on PagerDuty.

| Parameter Name        | Type   | Description                                                           |
|-----------------------|--------|-----------------------------------------------------------------------|
| PagerDuty Service Key | string | PagerDuty service key for pushing Secure Workload alerts on PagerDuty |

**Test:** Send a test alert to PagerDuty using the service key. If the alert is published successfully, the test passes.

**Apply:** Update configuration file of the connector with the specified parameters.

**Allowed Secure Workload virtual appliances:** None

**Allowed connectors:** PagerDuty

## Kinesis Notifier Configuration

Default configuration for publishing Secure Workload alerts on Amazon Kinesis.

| Parameter Name               | Type                    | Description                                               |
|------------------------------|-------------------------|-----------------------------------------------------------|
| <b>AWS Access Key ID</b>     | string                  | AWS access key ID to communicate with AWS                 |
| <b>AWS Secret Access Key</b> | string                  | AWS secret access key to communicate with AWS             |
| <b>AWS Region</b>            | dropdown of AWS regions | Name of the AWS region where Kinesis stream is configured |
| <b>Kinesis Stream</b>        | string                  | Name of the Kinesis stream                                |
| <b>Stream Partition</b>      | string                  | Partition Name of the stream                              |

**Test:** Send a test alert to the Kinesis stream using the given configuration. If the alert is published successfully, the test passes.

**Apply:** Update configuration file of the connector with the specified parameters.

**Allowed Secure Workload virtual appliances:** None

**Allowed connectors:** Kinesis

## Email Notifier Configuration

Default configuration for publishing Secure Workload alerts on Email.

| Parameter Name            | Type     | Description                                                               |
|---------------------------|----------|---------------------------------------------------------------------------|
| <b>SMTP Username</b>      | string   | SMTP server username. This parameter is optional.                         |
| <b>SMTP Password</b>      | string   | SMTP server password for the user (if given). This parameter is optional. |
| <b>SMTP Server</b>        | string   | IP address or hostname of the SMTP server                                 |
| <b>SMTP Port</b>          | number   | Listening port of SMTP server. Default value is 587.                      |
| <b>Secure Connection</b>  | checkbox | Should SSL be used for SMTP server connection?                            |
| <b>From Email Address</b> | string   | Email address to use for sending alerts                                   |
| <b>Default Recipients</b> | string   | Comma separated list of recipient email addresses                         |

**Test:** Send a test email using the given configuration. If the alert is published successfully, the test passes.

**Apply:** Update configuration file of the connector with the specified parameters.

**Allowed Secure Workload virtual appliances:** None

**Allowed connectors:** Email

## Syslog Notifier Configuration

Default configuration for publishing Secure Workload alerts on Syslog.

| Parameter Name        | Type         | Description                                                 |
|-----------------------|--------------|-------------------------------------------------------------|
| <b>Protocol</b>       | dropdown     | Protocol to use to connect to server                        |
|                       | • <i>UDP</i> |                                                             |
|                       | • <i>TCP</i> |                                                             |
| <b>Server Address</b> | string       | IP address or hostname of the Syslog server                 |
| <b>Port</b>           | number       | Listening port of Syslog server. Default port value is 514. |

**Test:** Send a test alert to Syslog server using the given configuration. If the alert is published successfully, the test passes.

**Apply:** Update configuration file of the connector with the specified parameters.

**Allowed Secure Workload virtual appliances:** None

**Allowed connectors:** Syslog

## Syslog Severity Mapping Configuration

The following table shows the default severity mapping for Secure Workload alerts on Syslog

| Secure Workload Alerts Severity | Syslog Severity |
|---------------------------------|-----------------|
| LOW                             | LOG_DEBUG       |
| MEDIUM                          | LOG_WARNING     |
| HIGH                            | LOG_ERR         |
| CRITICAL                        | LOG_CRIT        |
| IMMEDIATE ACTION                | LOG_EMERG       |

You can modify this setting using this configuration.

| Parameter Name   | Dropdown of mappings                                                                                                                                                                                                                                       |
|------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| IMMEDIATE_ACTION | <ul style="list-style-type: none"> <li>• <i>Emergency</i></li> <li>• <i>Alert</i></li> <li>• <i>Critical</i></li> <li>• <i>Error</i></li> <li>• <i>Warning</i></li> <li>• <i>Notice</i></li> <li>• <i>Informational</i></li> <li>• <i>Debug</i></li> </ul> |
| CRITICAL         |                                                                                                                                                                                                                                                            |
| HIGH             |                                                                                                                                                                                                                                                            |
| MEDIUM           |                                                                                                                                                                                                                                                            |
| LOW              |                                                                                                                                                                                                                                                            |

**Test:** No op.

**Apply:** Update configuration file of the connector with the specified parameters.

**Allowed Secure Workload virtual appliances:** None

**Allowed connectors:** Syslog

## ISE Instance Configuration

This configuration provides the parameters required to connect to the Cisco Identity Services Engine (ISE). By providing multiple instances of this configuration, the ISE connector can connect and pull metadata about endpoints from multiple ISE appliances. Up to 20 instances of ISE configuration may be provided.

| Parameter Name            | Type   | Description                                           |
|---------------------------|--------|-------------------------------------------------------|
| ISE Client Certificate    | string | ISE client certificate to connect to ISE using pxGrid |
| ISE Client Key            | string | ISE client key to connect to ISE                      |
| ISE Server CA Certificate | string | CA certificate of ISE                                 |
| ISE Hostname              | string | FQDN of ISE pxGrid                                    |
| ISE Nodename              | string | Node name of ISE pxGrid                               |

**Test:** Connect to ISE using the given parameters. On successful connection, accept the configuration.

**Apply:** Update configuration file of the connector with the specified parameters.

**Allowed Secure Workload virtual appliances:** None

**Allowed connectors:** ISE

## Discovery

Configurations that support *Discovery* mode do the following.

1. Collect a basic configuration from the user.



2. Verify the basic configuration.
3. Discovery additional properties about the configuration and present them to the user.
4. Let the user enhance the configuration using the discovered properties.
5. Verify and apply the enhanced configuration.

In the 3.3.1.x release, LDAP configuration supports discovery mode.

## LDAP Configuration

LDAP configuration specifies how to connect to LDAP, what is the base Distinguished Name (DN) to use, what is the attribute that corresponds to username, and what attributes to fetch for each username. LDAP attributes are properties of LDAP that are specific to that environment.

Given the configuration of how to connect to LDAP and the base DN, it is possible to discover the attributes of users in LDAP. These discovered attributes can then be presented to the user in the UI. From these discovered attributes, the user selects the attribute that corresponds to the username and a list of up to six attributes to collect for each username from LDAP. As a result, this eliminates the manual configuration of the LDAP attributes and reduces errors.

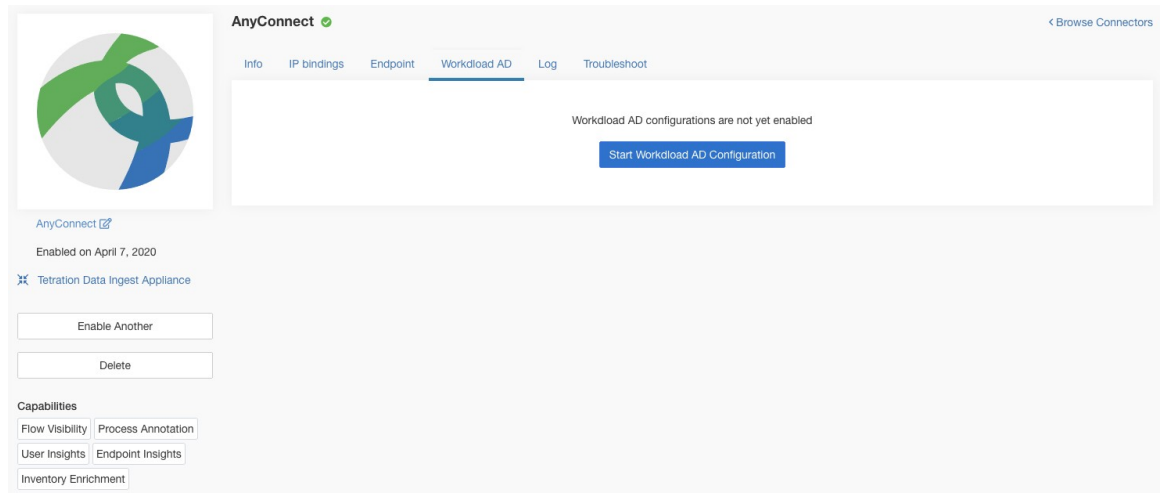
Here are the detailed steps for creating LDAP configuration through discovery.

### Procedure

#### Step 1 Start the LDAP Configuration

Initiate an LDAP configuration for the connector.

**Figure 125: Start the LDAP Configuration Discovery**



#### Step 2 Provide Basic LDAP Configuration

Specify the basic configuration for connecting to LDAP. In this configuration, the users provide the LDAP Bind DN or username to connect to LDAP server, LDAP password to use to connect to LDAP server, LDAP server address, LDAP server port, Base DN to connect to, and a filter string to fetch users that match this filter.

| Parameter Name                           | Type     | Description                                                                                    |
|------------------------------------------|----------|------------------------------------------------------------------------------------------------|
| <b>LDAP Username</b>                     | string   | LDAP username or bind DN to access LDAP server *                                               |
| <b>LDAP Password</b>                     | string   | LDAP password for the username to access LDAP server *                                         |
| <b>LDAP Server</b>                       | string   | LDAP server address                                                                            |
| <b>LDAP Port</b>                         | number   | LDAP server port                                                                               |
| <b>Use SSL</b>                           | checkbox | Should the connector connect to LDAP securely? Optional. Default is false.                     |
| <b>Verify SSL</b>                        | checkbox | Should the connector verify LDAP cert? Optional. Default is false.                             |
| <b>LDAP Server CA Cert</b>               | string   | Server CA certificate. Optional.                                                               |
| <b>LDAP Server Name</b>                  | string   | Servername for which the LDAP cert is issued (mandatory if <i>Verify SSL</i> is checked).      |
| <b>LDAP Base DN</b>                      | string   | LDAP base DN, the starting point for directory searches in LDAP                                |
| <b>LDAP Filter String</b>                | string   | LDAP filter prefix string. Filter the search result that match only this condition.            |
| <b>Snapshot Sync Interval (in hours)</b> | number   | Specify the time interval in hours to (re)create LDAP snapshot. Optional. Default is 24 hours. |
| <b>Use Proxy to reach LDAP</b>           | checkbox | Should the connector use proxy server to access LDAP server?                                   |
| <b>Proxy Server to reach LDAP</b>        | string   | Proxy server to access LDAP                                                                    |

Minimum user permissions needed to configure LDAP on Connectors is a **standard domain User**.

Figure 126: Initial LDAP configuration

The screenshot displays the 'AnyConnect' configuration page for 'Workload AD'. The interface is divided into a sidebar and a main configuration area. The sidebar includes the AnyConnect logo, a status indicator 'Enabled on April 7, 2020', and a list of capabilities: Flow Visibility, Process Annotation, User Insights, Endpoint Insights, and Inventory Enrichment. The main configuration area is titled 'Enter Configs' and contains the following fields:

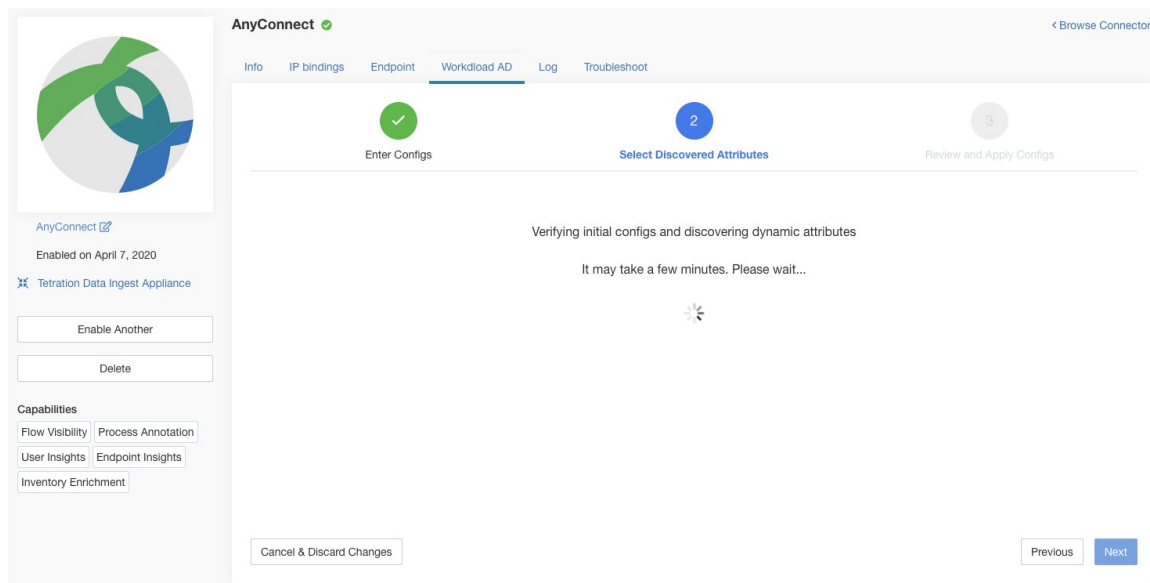
- LDAP Username:** cn=ldapadmin,dc=tetrationanalytics,dc=com
- LDAP Password:** [Redacted]
- LDAP Server:** 172.26.230.174
- LDAP Port:** 389
- Use SSL:**
- Verify SSL:**
- LDAP Server CA Cert (optional):** [Empty field]
- LDAP Server Name (optional):** Enter LDAP Server Name
- LDAP Base DN:** ou=People,dc=tetrationanalytic,dc=com
- LDAP Filter String:** (&(objectClass=organizationalPerson))
- Snapshot Sync Interval (in hours) (optional):** 24
- Use Proxy to reach LDAP:**
- Proxy Server to reach LDAP (optional):** http://1.1.1.1:8080

Navigation buttons include 'Cancel' and 'Next'.

### Step 3 Discovery in Progress

Once the user clicks *Next*, this configuration is sent to the connector. The connector establishes a connection with LDAP server using the given configuration. It fetches up to 1000 users from LDAP server and identifies all the attributes. Furthermore, it computes a list of all the single-valued attributes are common across all 1000 users. The connector returns this result back to Secure Workload.

Figure 127: Discovery in Progress



**Step 4 Enhance the Configuration with Discovered Attributes**

The user has to pick which attribute corresponds to username and select up to six attributes that the connector has to fetch and snapshot for each user in the organization (i.e., users matching the filter string). This action is performed using a dropdown of list of discovered attributes. Thus, eliminating manual errors and misconfiguration.

| Parameter Name           | Type            | Description                                               |
|--------------------------|-----------------|-----------------------------------------------------------|
| LDAP Username Attribute  | string          | LDAP attribute that contains the username                 |
| LDAP Attributes to Fetch | list of strings | List of LDAP attributes that should be fetched for a user |

**Figure 128: Discover LDAP Attributes**

The screenshot shows the AnyConnect configuration interface for a Workload AD connector. The interface is divided into three steps: 1. Enter Configs (completed), 2. Select Discovered Attributes (current step), and 3. Review and Apply Configs. In the 'Select Discovered Attributes' step, the 'LDAP Username Attribute' is set to 'cn'. The 'LDAP Attributes to Fetch' field contains a list of attributes: 'cn', 'lastLogoff', 'lastLogon', 'lastLogonTimestamp', and 'pwdLastSet'. The 'lastLogoff' attribute is currently selected. The interface includes a 'Cancel & Discard Changes' button and 'Previous' and 'Next' navigation buttons.

**Figure 129: Identify username attribute and attributes to collect for each username**

## Step 5 Finalize, Save, and Apply the Configuration

Finally, the configuration is completed by clicking *Save and Apply Changes*.

Figure 130: Complete LDAP Configuration Discovery and Commit

The figure consists of two screenshots of the AnyConnect configuration interface for LDAP. Both screenshots show the 'Workload AD' tab selected in the top navigation bar.

**Top Screenshot: Select Discovered Attributes**

- Progress indicator: Step 2 (Select Discovered Attributes) is active, with Step 1 (Enter Configs) completed and Step 3 (Review and Apply Configs) pending.
- LDAP Username Attribute:
- LDAP Attributes to Fetch:
- Buttons: 'Cancel & Discard Changes', 'Previous', and 'Next'.

**Bottom Screenshot: Review and Apply Configs**

- Progress indicator: Step 3 (Review and Apply Configs) is active, with Steps 1 and 2 completed.
- Configuration details:
  - LDAP Username: \*\*\*\*\*
  - LDAP Password: \*\*\*\*\*
  - LDAP Server: 172.26.230.174
  - LDAP Port: 389
  - Use SSL:
  - Verify SSL:
  - LDAP Server CA Cert: [Blank]
  - LDAP Server Name: [Blank]
  - LDAP Base DN: ou=People,dc=tetrationanalytics,dc=com
  - LDAP Filter String: (&(objectClass=organizationalPerson))
  - LDAP Username Attribute: cn
  - LDAP Attributes to Fetch: cn description title street displayName
  - Snapshot Sync Interval (in hours): 24
  - Use Proxy to reach LDAP:
  - Proxy Server to reach LDAP: [Blank]
- Buttons: 'Cancel', 'Previous', and 'Save & Apply Configs'.

The connector receives the completed configuration. It creates a local snapshot of all users matching the filter string and fetches only the selected attributes. Once the snapshot is completed, the connector services can start using the snapshot for annotating users and their LDAP attributes in inventories.

**Allowed Secure Workload virtual appliances:** None

**Allowed connectors:** AnyConnect, ISE, and F5.

## Remove

You can remove all the configurations that you have added from the connectors and/or appliances using the *Delete* button available for each configuration.

## Troubleshooting

Connectors and virtual appliances support various troubleshooting mechanisms to debug possible issues.



**Note** This section does not apply to the following:

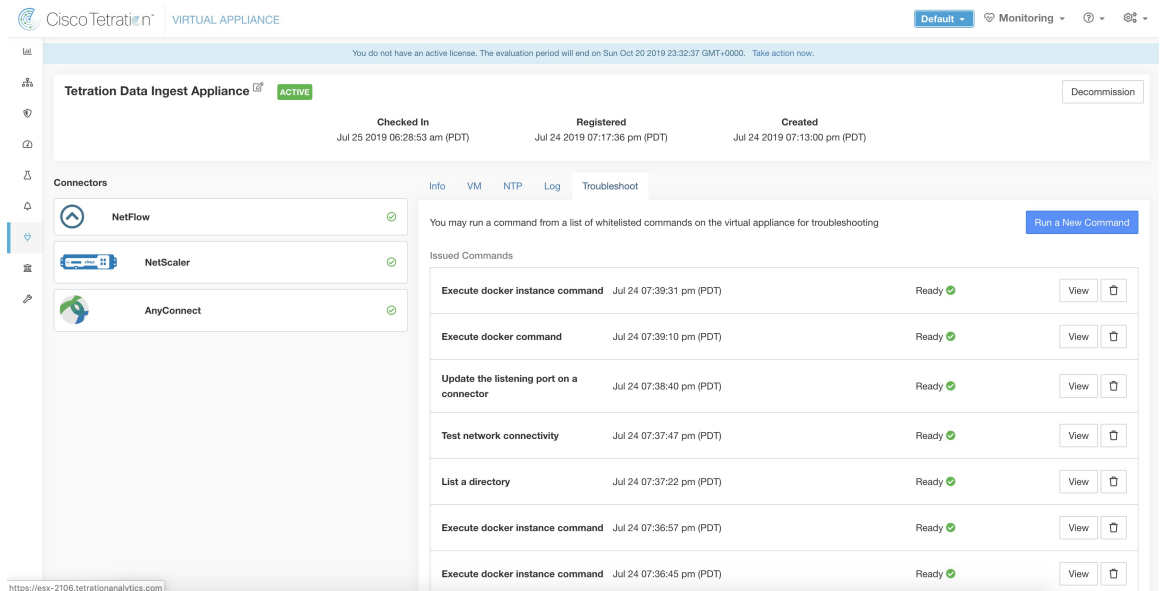
ERSPAN virtual appliance: Refer to the ERSPAN appliance page for the troubleshooting details.

Cloud connectors: To troubleshoot cloud connectors, see the section for your cloud connector, for example [Troubleshoot AWS Connector Issues](#).

## Allowed set of commands

The allowed set of commands enables you to run some debug commands on the appliances and Docker containers (for connectors). Allowed commands include the ability to retrieve logs and current running configuration, test network connectivity, and capture packets matching a specified port.

**Figure 131: Troubleshoot page on Secure Workload virtual appliance**





**Note** Troubleshooting using the allowed set of commands is available on the appliances and connectors only for users with the *Customer Support* role.

## Show Logs

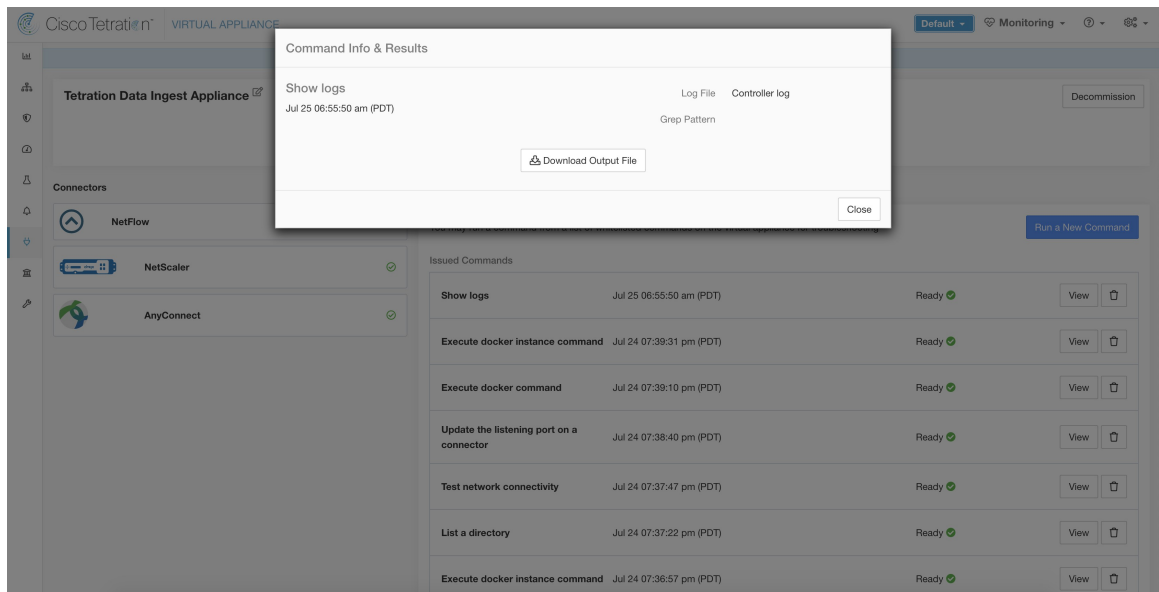
Show the contents of a controller log file and optionally grep the file for a specified pattern. Secure Workload sends the command to appliance/connector where the command was issued. The controller on the appliance/connector service returns the result (tailed for the last 5000 lines). When the result is available at Secure Workload, a download button is presented to download the file.

| Argument Name       | Type   | Description                             |
|---------------------|--------|-----------------------------------------|
| <b>Grep Pattern</b> | string | Pattern string to grep from the logfile |

**Allowed Secure Workload virtual appliances:** All

**Allowed connectors:** NetFlow, NetScaler, F5, AnyConnect, Syslog, Email, Slack, PagerDuty, Kinesis, ISE, ASA, and Meraki.

**Figure 132: Download Show Logs output from Secure Workload Ingest appliance**



## Show Service Logs

Show the contents of service log files and optionally grep the file for a specified pattern. Secure Workload sends the command to appliance/connector where the command was issued. The controller on the appliance/connector service returns the result (tailed for the last 5000 lines). When the result is available at Secure Workload, a download button is presented to download the file.

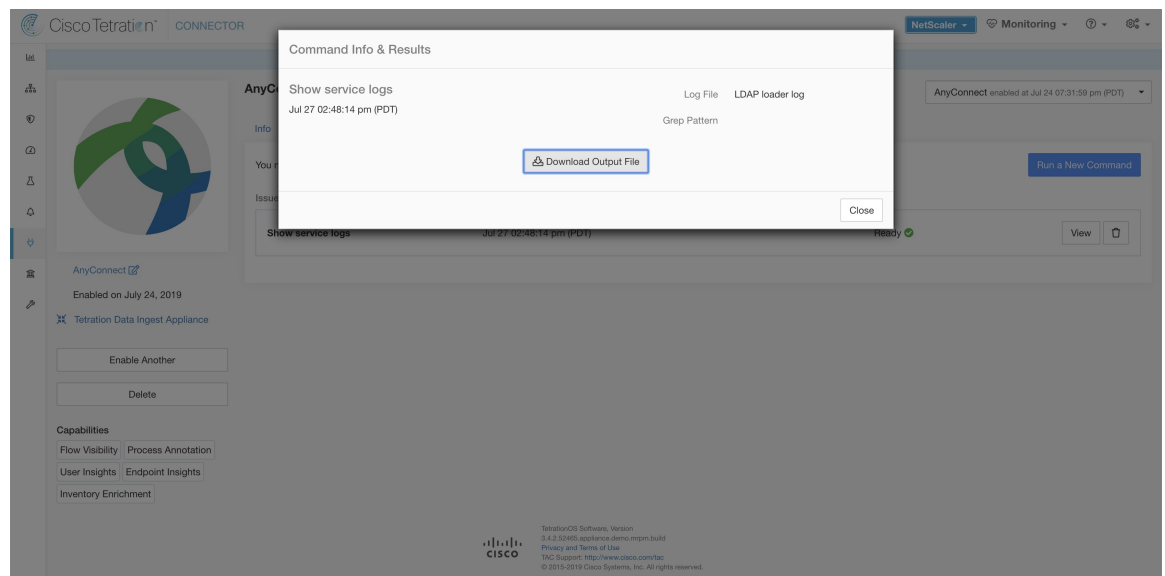


| Argument Name | Type                     | Description                                                     |
|---------------|--------------------------|-----------------------------------------------------------------|
| Log File      | dropdown                 | The name of the logfile to collect                              |
|               | • <i>Service log</i>     | Logs of the connector service                                   |
|               | • <i>Upgrade log</i>     | Upgrade logs of the service                                     |
|               | • <i>LDAP loader log</i> | Logs of the LDAP snapshot for connectors that have LDAP enabled |
| Grep Pattern  | string                   | Pattern string to grep from the logfile                         |

**Allowed Secure Workload virtual appliances:** None (only available on valid connector services)

**Allowed connectors:** NetFlow, NetScaler, F5, AnyConnect, Syslog, Email, Slack, PagerDuty, Kinesis, ISE, ASA, and Meraki.

**Figure 133: Download Show Service Logs output from AnyConnect connector for LDAP loader log log file**



## Show Running Configuration

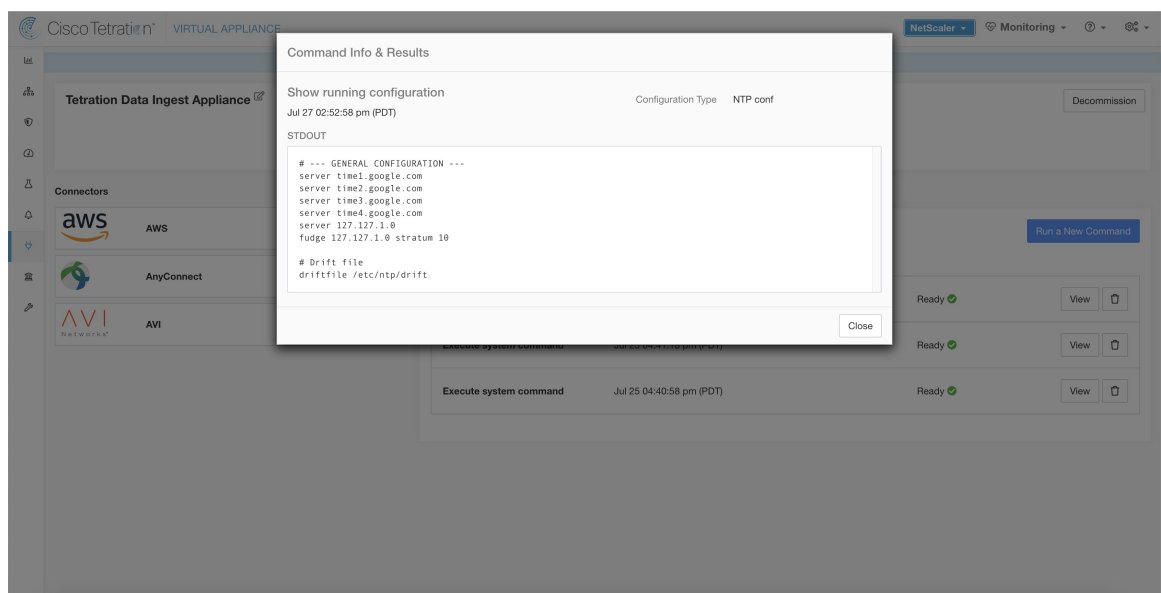
Show running configuration of an appliance/connector controllers. The controller on appliance/connector retrieves the configuration corresponding to the requested argument and returns the result. When the result is available at Secure Workload, the contents of the configuration are shown in a text box.

| Argument Name      | Type                     | Description                                                   |
|--------------------|--------------------------|---------------------------------------------------------------|
| Configuration Type | dropdown                 | Configuration file to collect                                 |
|                    | • <i>Controller conf</i> | Configuration file of the appliance controller                |
|                    | • <i>Supervisor conf</i> | Configuration file of the supervisor that runs the controller |
|                    | • <i>NTP conf</i>        | NTP configuration file                                        |
|                    | • <i>Chrony conf</i>     | /etc/chrony.conf                                              |

**Allowed Secure Workload virtual appliances:** All

**Allowed connectors:** NetFlow, NetScaler, F5, AnyConnect, Syslog, Email, Slack, PagerDuty, Kinesis, ISE, ASA, and Meraki.

**Figure 134: Show running configuration for NTP conf on a Secure Workload Ingest Appliance**



## Show Service Running Configuration

Show running configuration of an services instantiated for connectors on the appliances. The controller on the service retrieves the configuration corresponding to the requested argument and returns the result. When the result is available at Secure Workload, the contents of the configuration are shown in a text box.

| Argument Name             | Type                     | Description                                                    |
|---------------------------|--------------------------|----------------------------------------------------------------|
| <b>Configuration Type</b> | dropdown                 | Configuration file to collect.                                 |
|                           | • <i>Controller conf</i> | Configuration file of the service controller.                  |
|                           | • <i>Supervisor conf</i> | Configuration file of the supervisor that runs the controller. |
|                           | • <i>Service conf</i>    | Service configuration file.                                    |
|                           | • <i>LDAP conf</i>       | LDAP configuration for connectors that have LDAP enabled.      |

**Allowed Secure Workload virtual appliances:** None (only available on valid connector services)

**Allowed connectors:** NetFlow, NetScaler, F5, AnyConnect, Syslog, Email, Slack, PagerDuty, Kinesis, ISE, ASA, and Meraki.

## Show System Commands

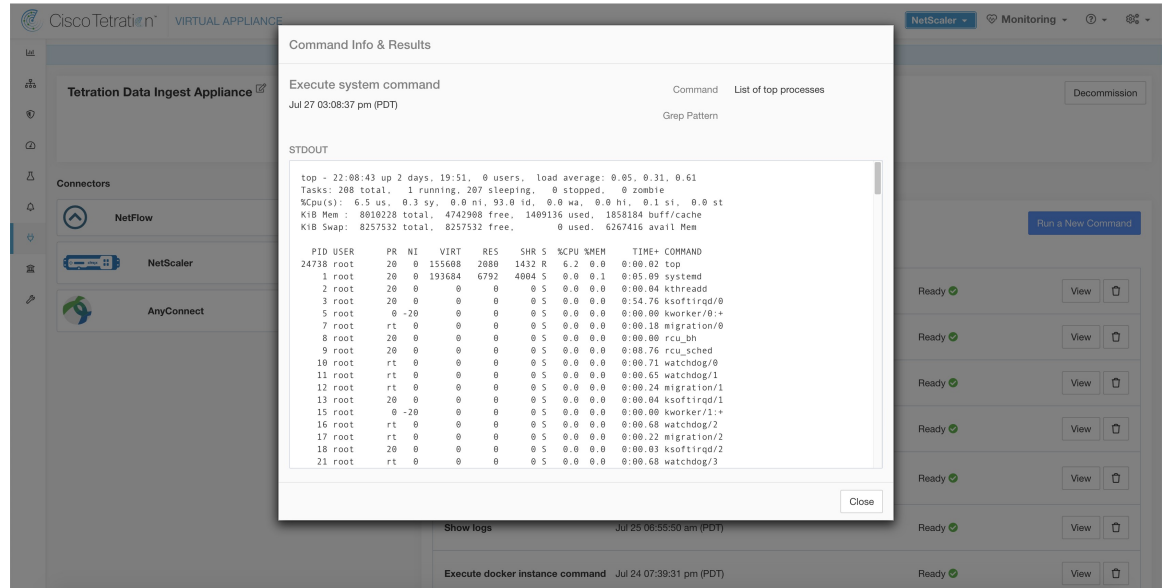
Execute a system command and optionally grep for a specified pattern. The controller on the appliance/connector service returns the result (tailed for the last 5000 lines). Optionally, a grep pattern can be provided as argument and the output is filtered accordingly. When the result is available at Secure Workload, the result is shown in a text box.

| Argument Name         | Type                               | Description                            |
|-----------------------|------------------------------------|----------------------------------------|
| <b>System Command</b> | dropdown                           | System command to execute              |
|                       | • <i>IP configuration</i>          | <b>ifconfig</b>                        |
|                       | • <i>IP route configuration</i>    | <b>ip route</b>                        |
|                       | • <i>IP packet filtering rules</i> | <b>iptables -L</b>                     |
|                       | • <i>Network status</i>            | <b>netstat</b>                         |
|                       | • <i>Network status (EL9)</i>      | <b>ss</b>                              |
|                       | • <i>Process status</i>            | <b>ps -aux</b>                         |
|                       | • <i>List of top processes</i>     | <b>top -b -n 1</b>                     |
|                       | • <i>NTP status</i>                | <b>ntpstat</b>                         |
|                       | • <i>NTP query</i>                 | <b>ntpq -pn</b>                        |
|                       | • <i>Chrony status (EL9)</i>       | <b>chronyc tracking</b>                |
|                       | • <i>Chrony query (EL9)</i>        | <b>chronyc sources</b>                 |
|                       | • <i>CPU info</i>                  | <b>lscpu</b>                           |
|                       | • <i>Memory info</i>               | <b>lsmem</b>                           |
| • <i>Disk free</i>    | <b>df -H</b>                       |                                        |
| <b>Grep Pattern</b>   | string                             | Pattern string to grep from the output |

**Allowed Secure Workload virtual appliances:** All

**Allowed connectors:** NetFlow, NetScaler, F5, AnyConnect, Syslog, Email, Slack, PagerDuty, Kinesis, ISE, ASA, and Meraki.

Figure 135: Show system command on Secure Workload Ingest appliance to retrieve list of top processes



## Show Docker Commands

Execute a Docker command and optionally grep for a specified pattern. The command is executed on the appliance by the appliance controller. The result tailed for the last 5000 lines. Optionally, a grep pattern can be provided as argument and the output is filtered accordingly. When the result is available at Secure Workload, the result is shown in a text box.

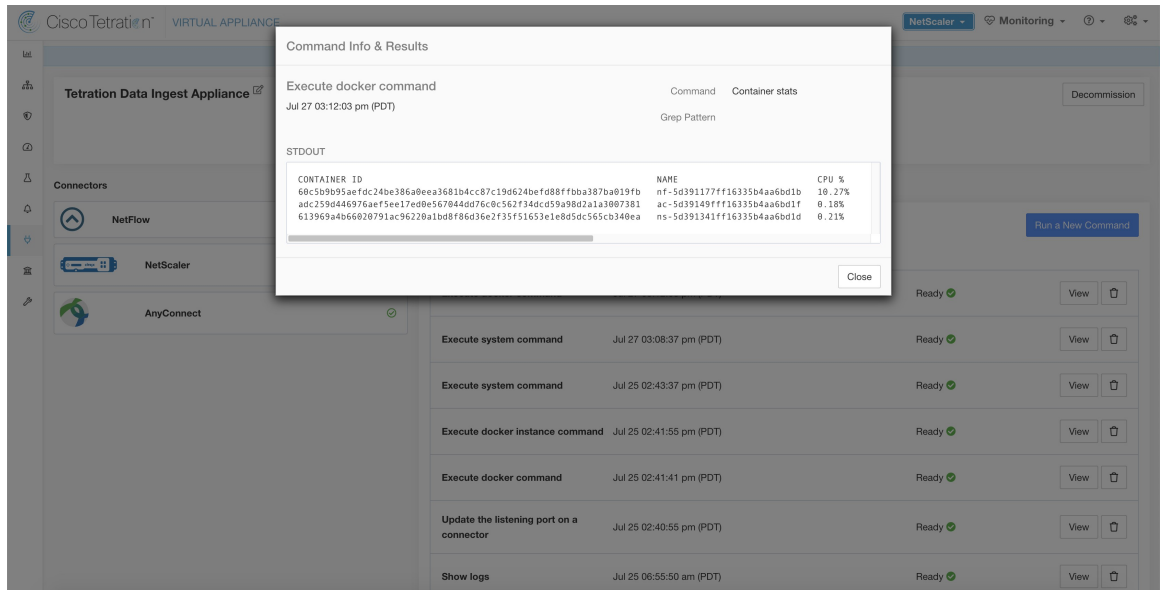
| Argument Name    | Type                          | Description                               |
|------------------|-------------------------------|-------------------------------------------|
| Docker Command   | dropdown                      | Docker command to execute                 |
|                  | • <i>Docker info</i>          | <b>docker info</b>                        |
|                  | • <i>List images</i>          | <b>docker images --no-trunc</b>           |
|                  | • <i>List containers</i>      | <b>docker ps --no-trunc</b>               |
|                  | • <i>List networks</i>        | <b>docker network ls --no-trunc</b>       |
|                  | • <i>List volumes</i>         | <b>docker volume ls</b>                   |
|                  | • <i>Container stats</i>      | <b>docker stats --no-trunc--no-stream</b> |
|                  | • <i>Docker disk usage</i>    | <code>docker system df -v</code>          |
|                  | • <i>Docker system events</i> | <b>docker system events --since '10m'</b> |
| • <i>Version</i> | <b>docker version</b>         |                                           |

| Argument Name | Type   | Description                            |
|---------------|--------|----------------------------------------|
| Grep Pattern  | string | Pattern string to grep from the output |

**Allowed Secure Workload virtual appliances:** All

**Allowed connectors:** None

**Figure 136: Execute a docker command on Secure Workload Ingest appliance to show container stats**



## Show Docker Instance Commands

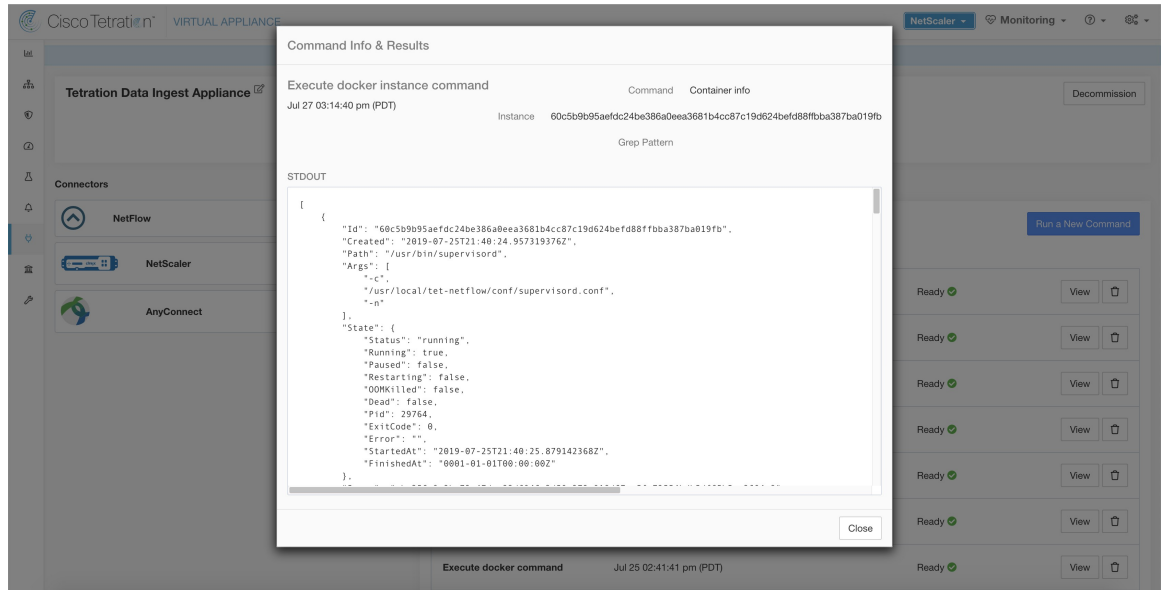
Execute a docker command on a specific instance of a Docker resource. The instance ID can be fetched using [Show Docker Commands](#). The command is executed on the appliance by the appliance controller. The result is tailed for the last 5000 lines. Optionally, a grep pattern can be provided as argument and the output is filtered accordingly. When the result is available at Secure Workload, the result is shown in a text box.

| Argument Name         | Type                                    | Description                                                                                        |
|-----------------------|-----------------------------------------|----------------------------------------------------------------------------------------------------|
| <b>Docker Command</b> | dropdown                                | Docker command to execute                                                                          |
|                       | • <i>Image info</i>                     | <b>docker images --no-trunc &lt;instance&gt;</b>                                                   |
|                       | • <i>Network info</i>                   | <b>docker network inspect &lt;instance&gt;</b>                                                     |
|                       | • <i>Volume info</i>                    | <b>docker volume inspect &lt;instance&gt;</b>                                                      |
|                       | • <i>Container info</i>                 | <b>docker container inspect--size &lt;instance&gt;</b>                                             |
|                       | • <i>Container logs</i>                 | <b>docker logs --tail 5000 &lt;instance&gt;</b>                                                    |
|                       | • <i>Container port mappings</i>        | <b>docker port &lt;instance&gt;</b>                                                                |
|                       | • <i>Container resource usage stats</i> | <b>docker stats --no-trunc--no-stream &lt;instance&gt;</b>                                         |
|                       | • <i>Container running processes</i>    | <b>docker top &lt;instance&gt;</b>                                                                 |
| <b>Instance</b>       | string                                  | Docker resource (image, network, volume, container) ID (See <a href="#">Show Docker Commands</a> ) |
| <b>Grep Pattern</b>   | string                                  | Pattern string to grep from the output                                                             |

**Allowed Secure Workload virtual appliances:** All

**Allowed connectors:** None

Figure 137: Execute a docker instance command on Secure Workload Ingest appliance to retrieve container info



## Show Supervisor Commands

Execute a `supervisorctl` command and return the result. Secure Workload sends the command to appliance/connector where the command was issued. The controller on the appliance/connector service returns the result. When the result is available at Secure Workload, the result is shown in a text box.

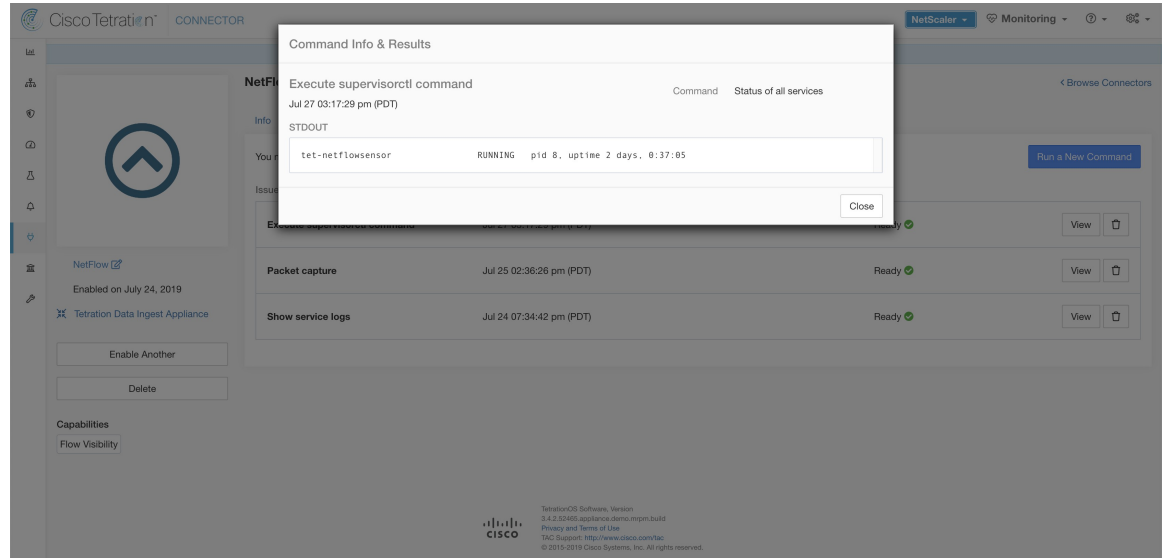
| Argument Name         | Type                            | Description                                   |
|-----------------------|---------------------------------|-----------------------------------------------|
| SupervisorCtl Command | dropdown                        | <code>supervisorctl</code> command to execute |
|                       | • <i>Status of all services</i> | <b>supervisorctl status</b>                   |
|                       | • <i>PID of supervisor</i>      | <b>supervisorctl pid</b>                      |
|                       | • <i>PID of all services</i>    | <b>supervisorctl pid all</b>                  |

**Allowed Secure Workload virtual appliances:** All

**Allowed connectors:** NetFlow, NetScaler, F5, AnyConnect, Syslog, Email, Slack, PagerDuty, Kinesis, ISE, ASA, and Meraki.



Figure 138: Execute supervisorctl command on NetFlow connector to get the status of all services

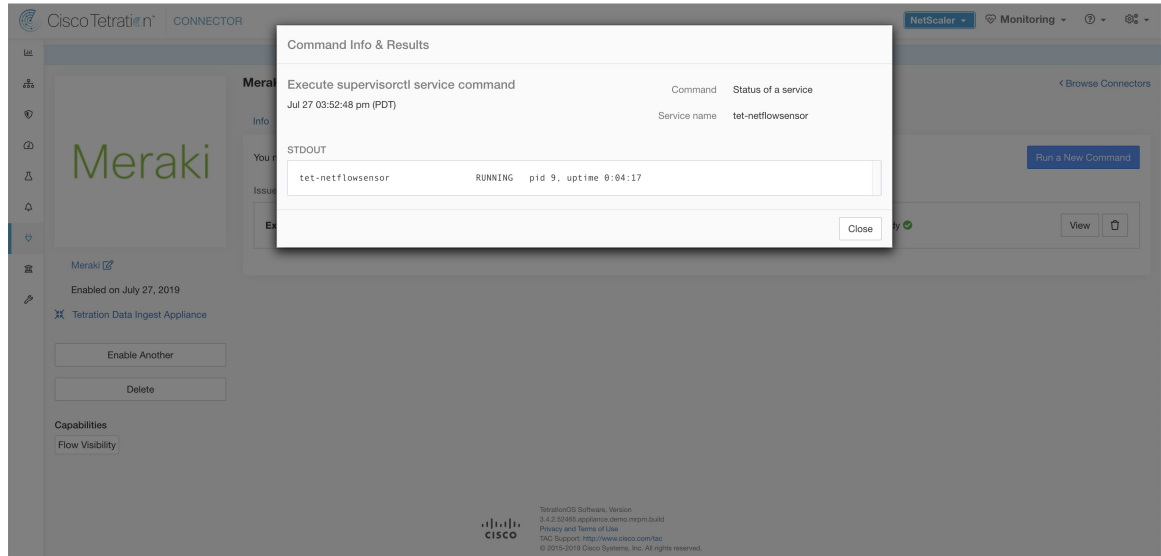


## Show Supervisor Service Commands

Execute a supervisorctl command on a specific service. The service name can be fetched using [Show Supervisor Commands](#). Secure Workload sends the command to appliance/connector where the command was issued. The controller on the appliance/connector service returns the result. When the result is available at Secure Workload, the result is shown in a text box.

| Argument Name         | Type                         | Description                                                                               |
|-----------------------|------------------------------|-------------------------------------------------------------------------------------------|
| SupervisorCtl Command | dropdown                     | <i>supervisorctl</i> command to execute                                                   |
|                       | • <i>Status of a service</i> | <b>supervisorctl status &lt;service name&gt;</b>                                          |
|                       | • <i>PID of a service</i>    | <b>supervisorctl pid &lt;service name&gt;</b>                                             |
| Service name          | string                       | Name of the supervisor controlled service (see <a href="#">Show Supervisor Commands</a> ) |

**Figure 139: Execute supervisorctl command on NetFlow connector to get the status of specified service name**



**Allowed Secure Workload virtual appliances:** All

**Allowed connectors:** NetFlow, NetScaler, F5, AnyConnect, Syslog, Email, Slack, PagerDuty, Kinesis, ISE, ASA, and Meraki.

## Network Connectivity Commands

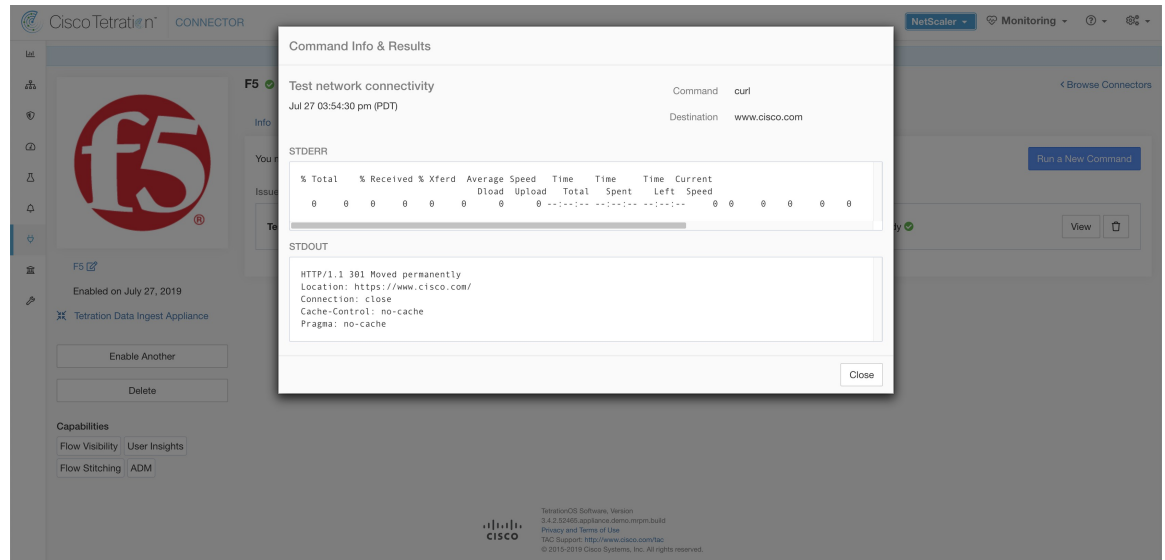
Test network connectivity from the appliance/connector. The command is executed on the appliance by the appliance controller. When the result is available at Secure Workload, the result is shown in a text box.

| Argument Name   | Type     | Description                             |
|-----------------|----------|-----------------------------------------|
| Network Command | dropdown | Network connectivity command to execute |
|                 | • ping   | <b>ping -c 5 &lt;destination&gt;</b>    |
|                 | • curl   | <b>curl -I &lt;destination&gt;</b>      |
| Destination     | string   | Destination to use for the test         |

**Allowed Secure Workload virtual appliances:** All

**Allowed connectors:** NetFlow, NetScaler, F5, AnyConnect, Syslog, Email, Slack, PagerDuty, Kinesis, ISE, ASA, and Meraki.

Figure 140: Test network connectivity on F5 connector by running a curl



## List Files

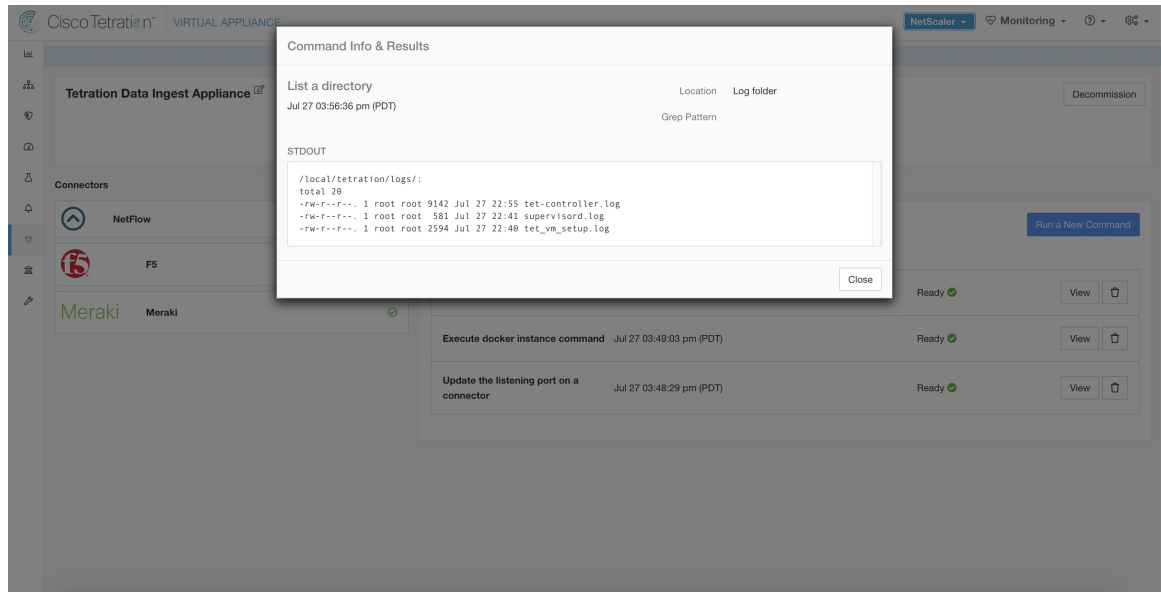
List the files in well known locations of the appliance. Optionally, grep for a specified pattern. Secure Workload sends the command to appliance where the command was issued. The controller on the appliance returns the result. When the result is available at Secure Workload, the result is shown in a text box.

| Argument Name       | Type                                     | Description                                                                    |
|---------------------|------------------------------------------|--------------------------------------------------------------------------------|
| <b>Location</b>     | dropdown                                 | List files in a target location                                                |
|                     | • <i>Controller configuration folder</i> | List the contents in the folder where controller configuration files are kept. |
|                     | • <i>Controller cert folder</i>          | List the contents in the folder where controller certs are kept.               |
|                     | • <i>Log folder</i>                      | List the contents in the folder where log files are present.                   |
| <b>Grep Pattern</b> | string                                   | Pattern string to grep from the output                                         |

**Allowed Secure Workload virtual appliances:** All

**Allowed connectors:** None

Figure 141: List the files in log folder in Secure Workload Ingest appliance



## List Service Files

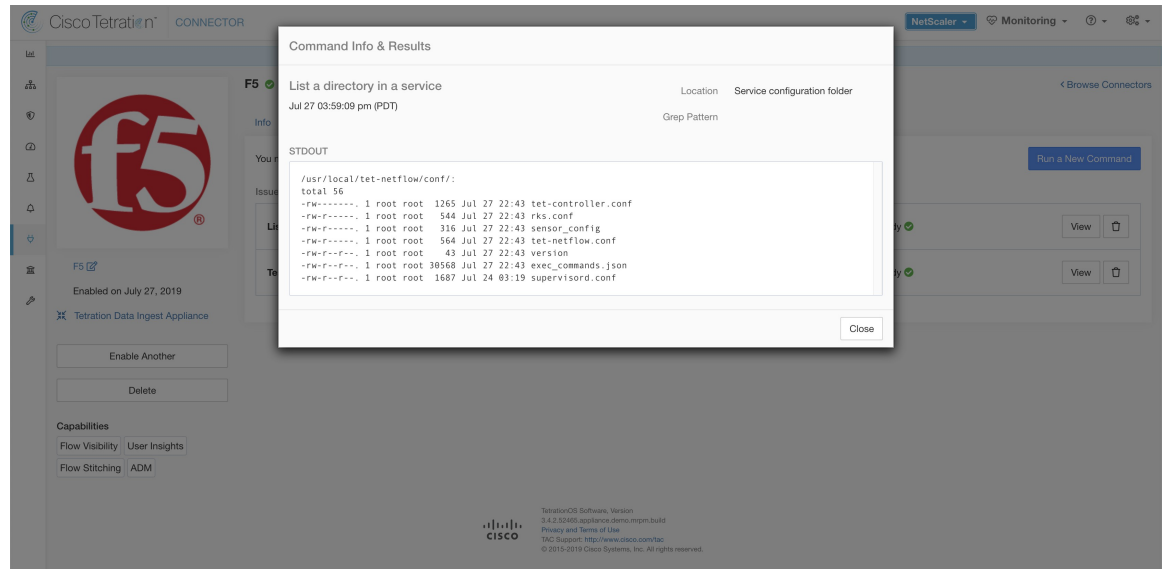
List the files in well known locations of the connector service. Optionally, grep for a specified pattern. Secure Workload sends the command to connector where the command was issued. The controller on the connector service returns the result. When the result is available at Secure Workload, the result is shown in a text box.

| Argument Name | Type                                  | Description                                                                                                 |
|---------------|---------------------------------------|-------------------------------------------------------------------------------------------------------------|
| Location      | dropdown                              | List files in a target location.                                                                            |
|               | • <i>Service configuration folder</i> | List the contents in the folder where service configuration files are kept.                                 |
|               | • <i>Service cert folder</i>          | List the contents in the folder where service certs are kept.                                               |
|               | • <i>Log folder</i>                   | List the contents in the folder where log files are present.                                                |
|               | • <i>DB folder</i>                    | List the contents in the folder where state of endpoints (esp. for AnyConnect and ISE connectors) are kept. |
| Grep Pattern  | string                                | Pattern string to grep from the output                                                                      |

**Allowed Secure Workload virtual appliances:** None

**Allowed connectors:** NetFlow, NetScaler, F5, AnyConnect, Syslog, Email, Slack, PagerDuty, Kinesis, ISE, ASA, and Meraki.

Figure 142: List the files in configuration folder of F5 connector in Secure Workload Ingest appliance



## Packet Capture

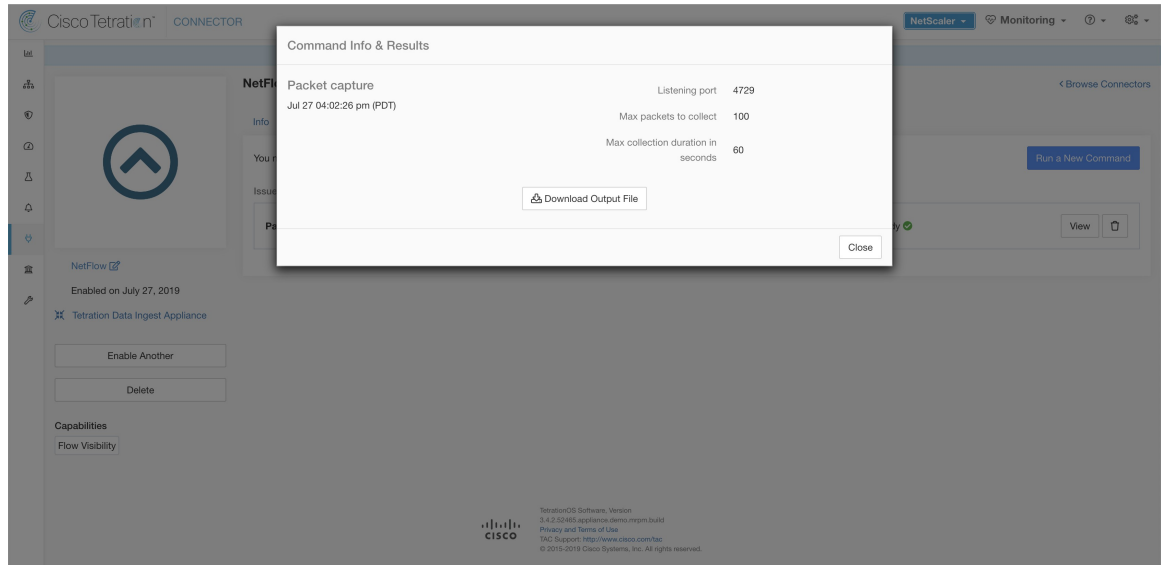
Capture incoming packets on an appliance/connector. Secure Workload sends the command to the appliance/connector where the command was issued. The controller on the appliance/connector service captures packets, encodes them and returns the result to Secure Workload. When the result is available at Secure Workload, a download button is presented to download the file in .pcap format.

| Argument Name                             | Type   | Description                                                                   |
|-------------------------------------------|--------|-------------------------------------------------------------------------------|
| <b>Listening port</b>                     | number | Capture packets that are sent/received on this port                           |
| <b>Max packets to collect</b>             | number | Maximum packets to collect before returning the result. Should be <1000       |
| <b>Max collection duration in seconds</b> | number | Maximum duration to collect before return the result. Should be <600 seconds. |

**Allowed Secure Workload virtual appliances:** All

**Allowed connectors:** NetFlow, NetScaler, F5, AnyConnect, Syslog, Email, Slack, PagerDuty, Kinesis, ISE, ASA, and Meraki.

Figure 143: Capture packets on a given port on NetFlow connector



## Update Listening Ports of Connectors

Update the listening port on a connector in Secure Workload Ingest appliance. Secure Workload sends the command to the appliance controller on the appliance where the command is issued. The controller does the following actions:

- Stops the Docker service corresponding to the connector.
- Collect the current running configuration of the service.
- Remove the Docker service.
- Update the running configuration of the service to use the new ports.
- Start a new container from the same Docker image that was used in the removed container with new exposed ports. Also, if a Docker volume was mounted to the removed container earlier, the same volume is mounted to the new container.
- Return the new IP bindings of the connector to Secure Workload.
- Secure Workload shows the result in a text box.

| Argument Name               | Type             | Description                                                                |
|-----------------------------|------------------|----------------------------------------------------------------------------|
| <b>Connector ID</b>         | string           | Connector ID of the connector for which listening ports need to be updated |
| <b>Listening port label</b> | dropdown         | The type of port that is updated.                                          |
|                             | <i>NET-FLOW9</i> | NetFlow v9 listening port                                                  |
|                             | <i>IPFIX</i>     | IPFIX listening port                                                       |

| Argument Name  | Type   | Description                |
|----------------|--------|----------------------------|
| Listening port | string | New port for the connector |

Allowed Secure Workload virtual appliances: Secure Workload Ingest

Allowed connectors: None

Figure 144: Update listening port on Meraki connector to 2055 in Secure Workload Ingest appliance

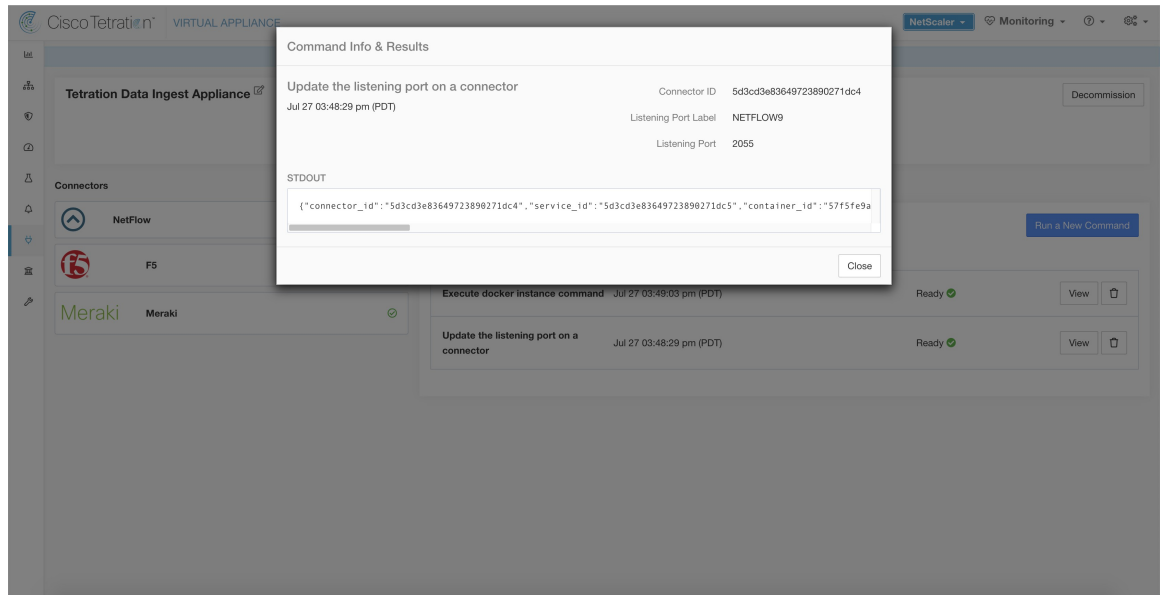
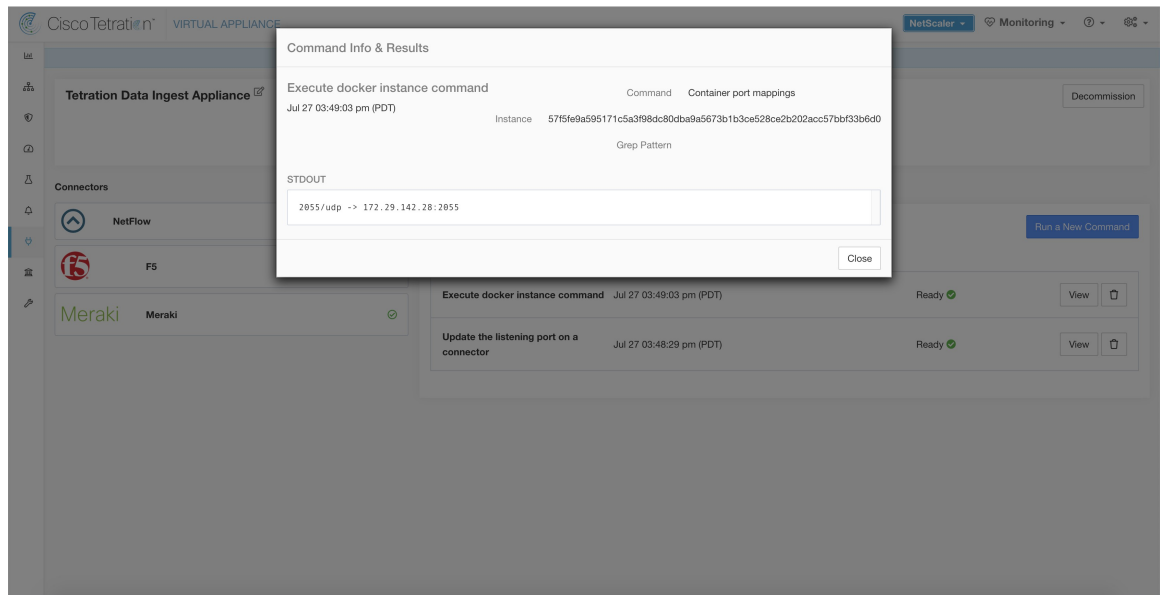


Figure 145: Retrieve the port mappings on Meraki connector in Secure Workload Ingest appliance



## Update Alert Notifier Connector Log Configuration

Update log configuration for Secure Workload Alert Notifier (TAN) service that hosts Syslog, Email, Slack, PagerDuty, and Kinesis alert notifier connectors. Since TAN hosts multiple connectors, log configuration cannot be updated from connector page directly. This allowed command allows the user to update the log configuration.

Secure Workload sends the command to the service controller on TAN Docker service of Secure Workload Edge appliance. The controller applies the configuration on the service and returns the status of the configuration update.

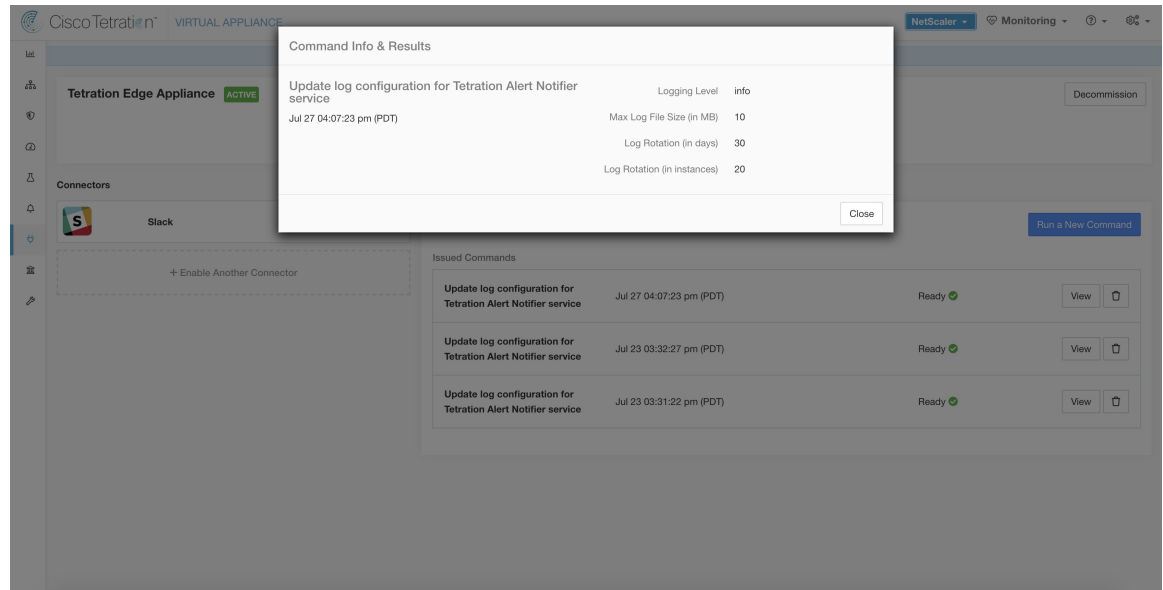
| Argument Name                      | Type           | Description                                             |
|------------------------------------|----------------|---------------------------------------------------------|
| <b>Logging level</b>               | dropdown       | Logging level to be used by the service                 |
|                                    | • <i>debug</i> | Debug log level                                         |
|                                    | • <i>info</i>  | Informational log level                                 |
|                                    | • <i>warn</i>  | Warning log level                                       |
|                                    | • <i>error</i> | Error log level                                         |
| <b>Max log file size (in MB)</b>   | number         | Maximum size of a log file before log rotation kicks in |
| <b>Log rotation (in days)</b>      | number         | Maximum age of a log file before log rotation kicks in  |
| <b>Log rotation (in instances)</b> | number         | Maximum instances of log files kept                     |

**Allowed Secure Workload virtual appliances:**Secure Workload Edge

**Allowed connectors:**None



**Figure 146: Update the log configuration on Secure Workload Alert Notifier Docker service in Secure Workload Edge appliance**



## Collect Snapshot From Appliance

Secure Workload sends the command to the appliance where the command was issued. When the controller on the appliance receives this command from Secure Workload, it collects appliance snapshot, encodes them and returns the result to Secure Workload. When the result is available at Secure Workload, a download button is presented to download the file in `.tar.gz` format.

Files included in the snapshot:

- `/local/tetration/appliance/appliance.conf`
- `/local/tetration/{logs, sqlite, user.cfg}`
- `/opt/tetration/tet_vm_setup/conf/tet-vm-setup.conf`
- `/opt/tetration/tet_vm_setup/docker/Dockerfile`
- `/opt/tetration/ova/version`
- `/usr/local/tet-controller/conf`
- `/usr/local/tet-controller/cert/{topic.txt, kafkaBrokerIps.txt}`
- `/var/run/supervisord.pid`
- `/etc/resolv.conf`

Command outputs included in the snapshot:

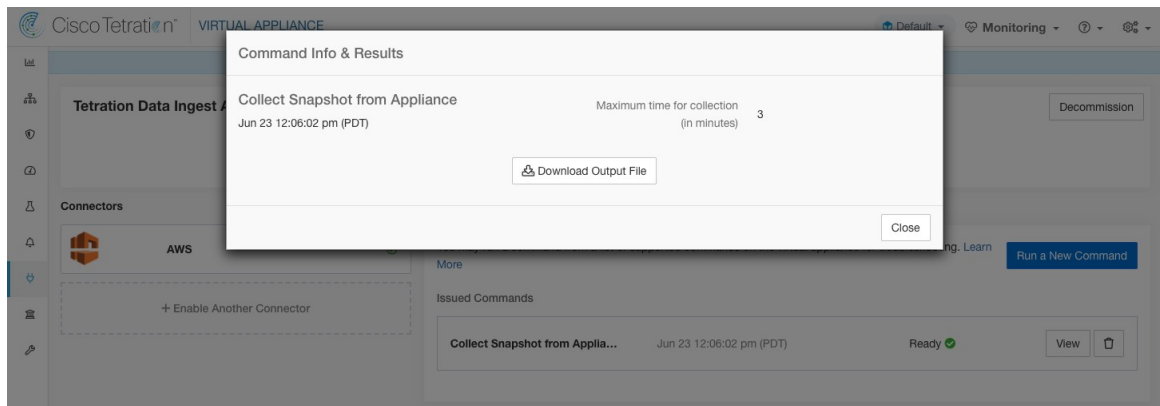
- `ps aux`
- `iptables -L`
- `netstat {-nat, -rn, -suna, -stna, -tunlp}`
- `ss {-nat, -rn, -suna, -stna, -tunlp}`

- /usr/local/tet-controller/tet-controller -version
- supervisorctl status
- rpm -qi tet-nic-driver tet-controller
- du -shc /local/tetration/logs
- ls {/usr/local/tet-controller/cert/, -l /local/tetration/sqlite/, -l /opt/tetration/tet\_vm\_setup/.tet\_vm.done, -l /opt/tetration/tet\_vm\_setup/templates/}
- docker {images, ps -a}
- blkid/ifconfig/lscpu/uptime
- free -m
- df -h

| Argument Name                      | Type   | Description                                                                      |
|------------------------------------|--------|----------------------------------------------------------------------------------|
| Max time for collection in minutes | number | Maximum duration to collect before returning the results. Should be <20 minutes. |

**Allowed Secure Workload virtual appliances :** Secure Workload Ingest and Secure Workload Edge

**Figure 147: Collect snapshot from Secure Workload appliance**



## Collect Snapshot From Connector

Secure Workload sends the command to the appliance where the connector is deployed. According to connector ID, the controller collects connector snapshot, encodes them and returns the result to Secure Workload. When the result is available at Secure Workload, a download button is presented to download the file in .tar.gz format.

Files included in the snapshot:

- /usr/local/tet-netflow/conf
- /local/tetration/{logs, sqlite}
- /var/run/{supervisord.pid, tet-netflow.pid}

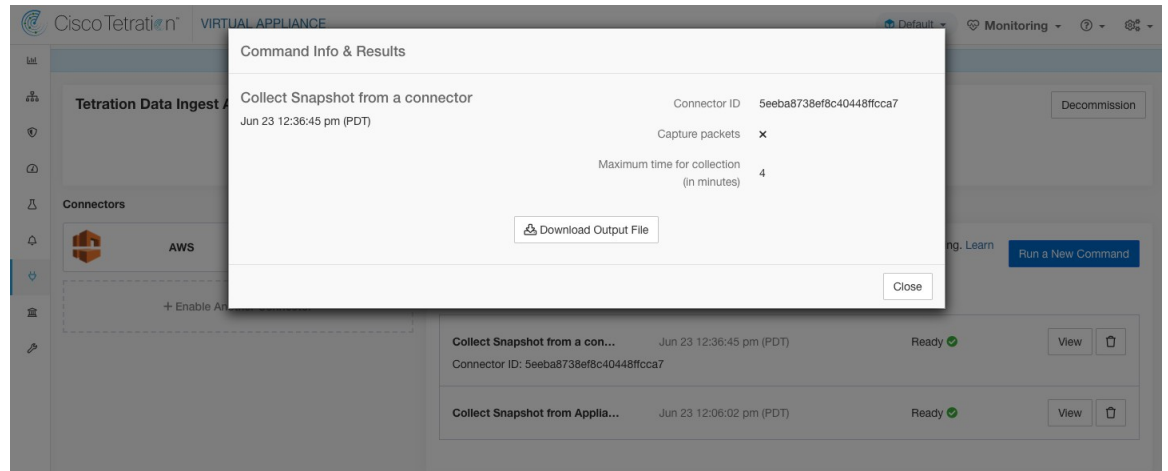
Command outputs included in the snapshot:

- ps aux
- netstat {-nat, -rn, -suna, -stna, -tunlp}
- ss {-nat, -rn, -suna, -stna, -tunlp}

| Argument Name                      | Type      | Description                                                                       |
|------------------------------------|-----------|-----------------------------------------------------------------------------------|
| Connector ID                       | string    | Connector ID of the connector for which the snapshot command is run.              |
| Capture packets                    | check-box | Should packets be captured?                                                       |
| Max time for collection in minutes | number    | Maximum duration to collect before returning the results. Should be < 20 minutes. |

**Allowed Secure Workload virtual appliances:** Secure Workload Ingest and Secure Workload Edge

**Figure 148: Collect snapshot from Secure Workload connector on designated connector ID**



## Collect Controller Profile

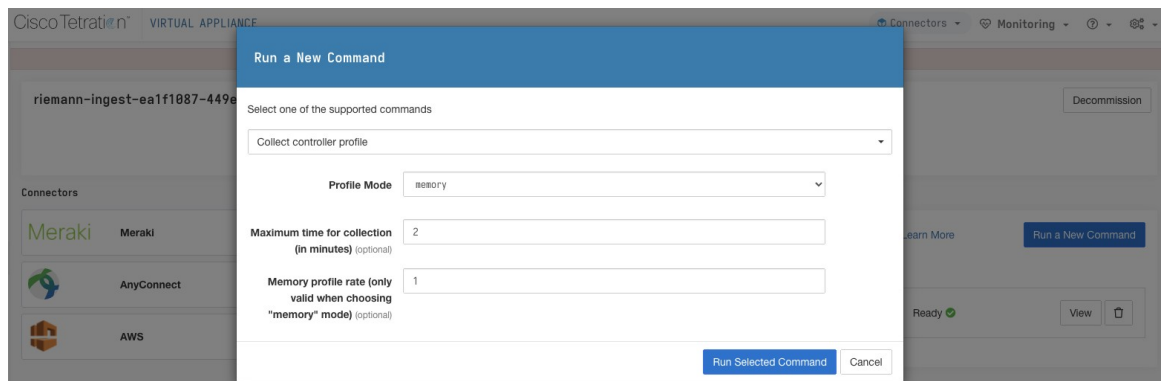
Collect controller process profiling result on appliance or connectors. Secure Workload sends the command to the connector where the command was issued. The service controller restarts the connector service in the specified profiling mode. After collecting the profiling result, service controller restarts the service in normal mode and send the result to Secure Workload. When the result is available at Secure Workload, a download button is presented to download the file in `.tar.gz` format.

| Argument Name                                                       | Type               | Description                                                                                           |
|---------------------------------------------------------------------|--------------------|-------------------------------------------------------------------------------------------------------|
| <b>Profile Mode</b>                                                 | dropdown           | Profiling mode.                                                                                       |
|                                                                     | • <i>memory</i>    | Memory profiling mode.                                                                                |
|                                                                     | • <i>cpu</i>       | CPU profiling mode.                                                                                   |
|                                                                     | • <i>block</i>     | Block profiling mode.                                                                                 |
|                                                                     | • <i>mutex</i>     | Mutex profiling mode.                                                                                 |
|                                                                     | • <i>goroutine</i> | Goroutine profiling mode.                                                                             |
| <b>Maximum time for collection (in minutes)</b>                     | number             | Maximum duration to collect before returning the result.                                              |
| <b>Memory profile rate (only valid when choosing “memory” mode)</b> | number             | Memory profiling rate. This field is optional. If not provided, default value in Golang will be used. |

**Allowed Secure Workload virtual appliances:** Secure Workload Ingest and Secure Workload Edge

**Allowed connectors:** NetFlow, NetScaler, F5, AnyConnect, Syslog, Email, Slack, PagerDuty, Kinesis, ISE, and Meraki.

**Figure 149: Collect controller profile from Secure Workload appliance**



## Collect Connector Profile

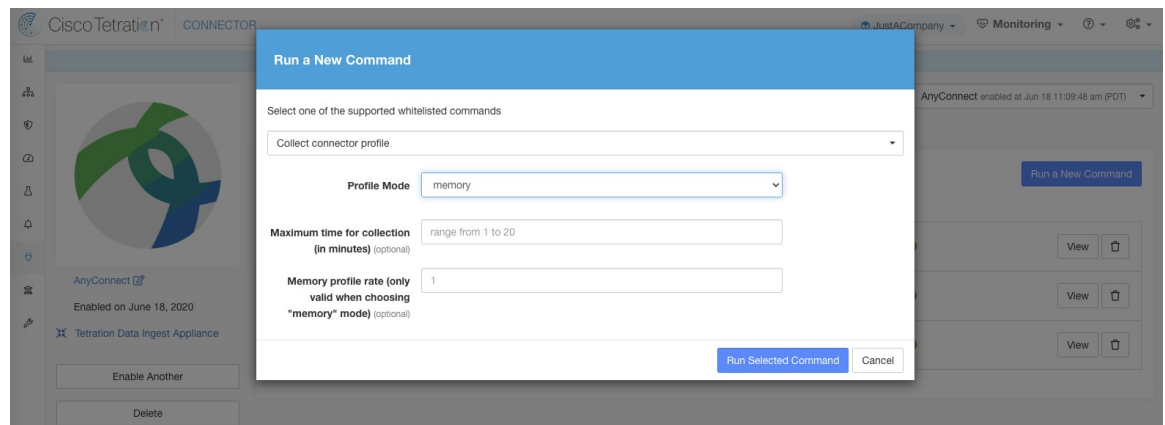
Collect connector process profiling result on connectors. Secure Workload sends the command to the connector where the command was issued. The service controller restart the connector service in the specified profiling mode. After collecting the profiling result, service controller restart the service in normal mode and send the result to Secure Workload. When the result is available at Secure Workload, a download button is presented to download the file in `.tar.gz` format.

| Argument Name                                                       | Type               | Description                                                                                           |
|---------------------------------------------------------------------|--------------------|-------------------------------------------------------------------------------------------------------|
| <b>Profile Mode</b>                                                 | dropdown           | Profiling mode.                                                                                       |
|                                                                     | • <i>memory</i>    | Memory profiling mode.                                                                                |
|                                                                     | • <i>cpu</i>       | CPU profiling mode.                                                                                   |
|                                                                     | • <i>block</i>     | Block profiling mode.                                                                                 |
|                                                                     | • <i>mutex</i>     | Mutex profiling mode.                                                                                 |
|                                                                     | • <i>goroutine</i> | Goroutine profiling mode.                                                                             |
| <b>Maximum time for collection (in minutes)</b>                     | number             | Maximum duration to collect before returning the result.                                              |
| <b>Memory profile rate (only valid when choosing “memory” mode)</b> | number             | Memory profiling rate. This field is optional. If not provided, default value in Golang will be used. |

**Allowed Secure Workload virtual appliances:** Secure Workload Ingest and Secure Workload Edge

**Allowed connectors:** NetFlow, NetScaler, F5, AnyConnect, Syslog, Email, Slack, PagerDuty, Kinesis, ISE, and Meraki.

**Figure 150: Collect connector profile from Secure Workload connector**



## Override connector alert interval for Appliance

Override default connector alert interval for appliance. Secure Workload restricts same connector alert to send only once a day in default. This command is for administrator to override interval when they think once a day is too long. When the result is available at Secure Workload, the result is shown in a text box.

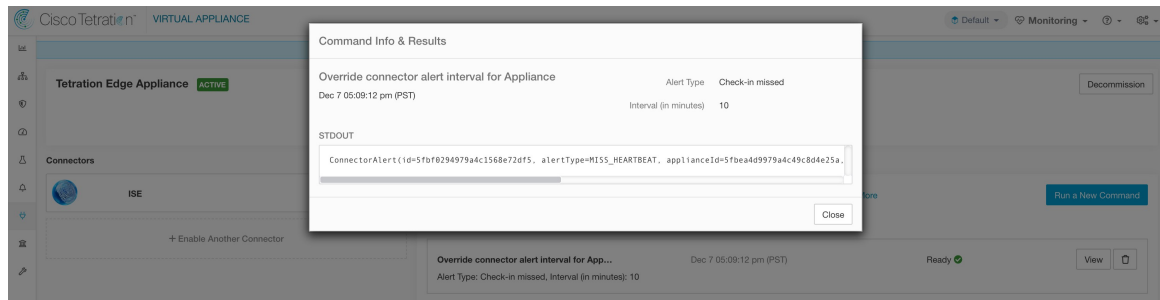
## Override connector alert interval for Connector

| Argument Name                | Type                     | Description                               |
|------------------------------|--------------------------|-------------------------------------------|
| <b>Alert Type</b>            | dropdown                 | The connector alert type to override.     |
|                              | • <i>Check-in missed</i> | Miss appliance's check-in.                |
|                              | • <i>CPU usage</i>       | High CPU usage.                           |
|                              | • <i>Memory usage</i>    | High memory usage.                        |
|                              | • <i>Disk usage</i>      | High disk usage.                          |
| <b>Interval (in minutes)</b> | number                   | Duration to override interval in minutes. |

**Allowed Secure Workload virtual appliances:** Secure Workload Ingest and Secure Workload Edge

**Allowed connectors:** None

**Figure 151: Override connector alert interval for Secure Workload appliance**



## Override connector alert interval for Connector

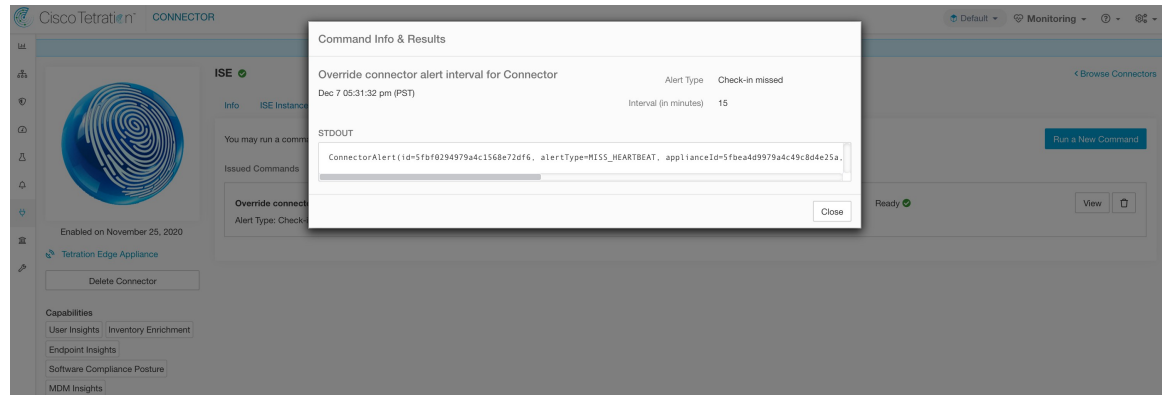
Override default connector alert interval for connector. Secure Workload restricts same connector alert to send only once a day in default. This command is for administrator to override interval when they think once a day is too long. When the result is available at Secure Workload, the result is shown in a text box.

| Argument Name                | Type                     | Description                               |
|------------------------------|--------------------------|-------------------------------------------|
| <b>Alert Type</b>            | dropdown                 | The connector alert type to override.     |
|                              | • <i>Check-in missed</i> | Miss connector's check-in.                |
| <b>Interval (in minutes)</b> | number                   | Duration to override interval in minutes. |

**Allowed Secure Workload virtual appliances:** None

**Allowed connectors:** NetFlow, NetScaler, F5, AnyConnect, Syslog, Email, Slack, PagerDuty, Kinesis, ISE, ASA, Meraki, ServiceNow, WAD.

Figure 152: Override connector alert interval for Secure Workload connector



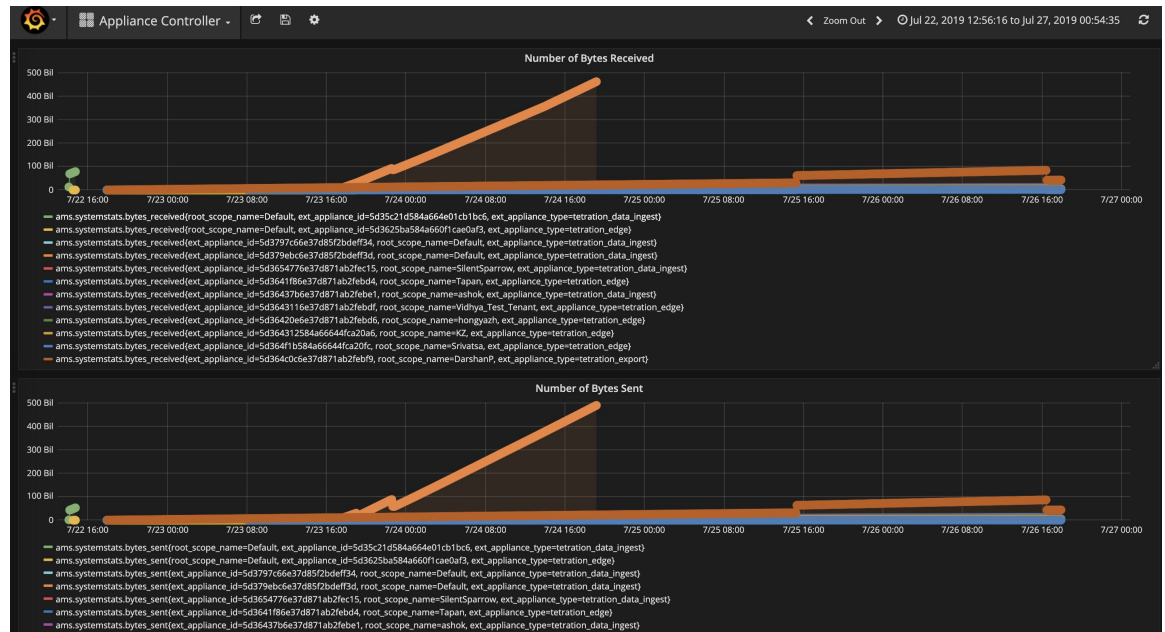
## Hawkeye Dashboards

Hawkeye dashboards provide insights about health of the connectors and virtual appliances where the connectors are enabled.

### Appliance Controller Dashboard

Appliance controller dashboard provides information about network statistics, system metrics such as CPU usage percentage, memory usage percentage, disk usage percentage, and number of open file descriptors.

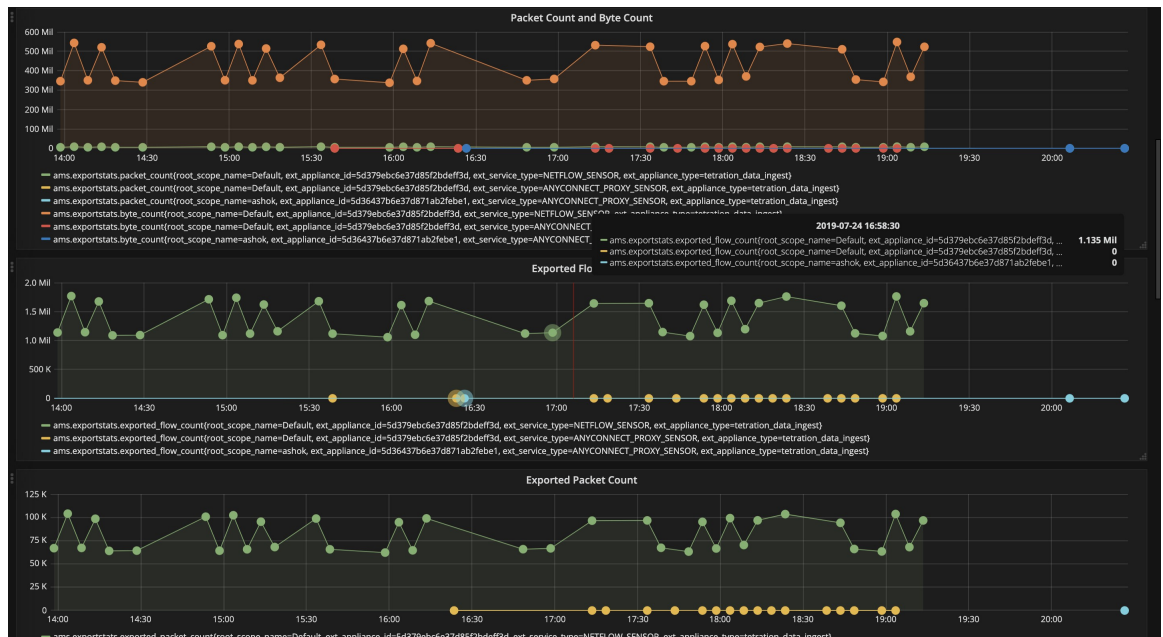
Figure 153: Appliance controller dashboard



## Service Dashboard

Service dashboard provides information about export metrics -if applicable- including number of flow observations exported to Secure Workload, number of packets exported to Secure Workload, and number of bytes exported to Secure Workload. In addition, this dashboard also provides information about protocol processing and decoding (for example, services that process NetFlow v9, and IPFIX). Metrics such as decoded count, decoded error count, flow count, packet count, and byte count are available in this dashboard. Furthermore, system metrics for the Docker container where the service is running are also included in this dashboard. Metrics such as CPU usage percentage, memory usage percentage, disk usage percentage, and number of open file descriptors are part of this dashboard.

Figure 154: Service dashboard

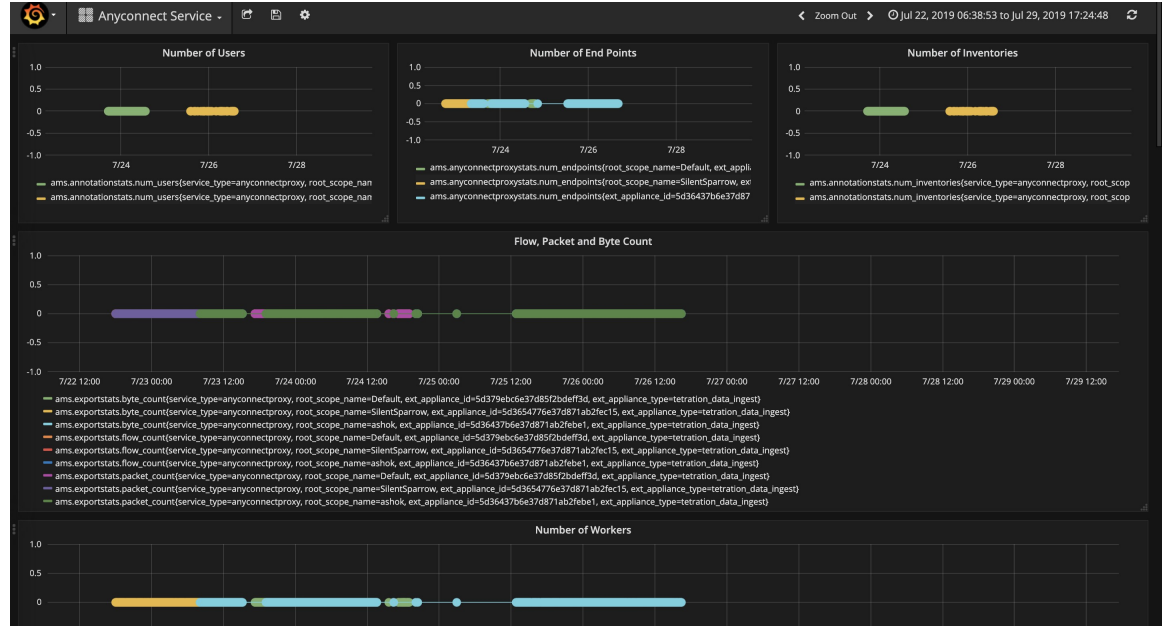


## AnyConnect Service Dashboard

AnyConnect service dashboard provides information about AnyConnect specific service information. Metrics such as number of endpoints, number of inventories, number of users reported by AnyConnect connector to Secure Workload are available in this dashboard. In addition, this dashboard also provides information about IPFIX protocol processing and decoding. Metrics such as decoded count, decoded error count, flow count, packet count, and byte count are available in this dashboard.



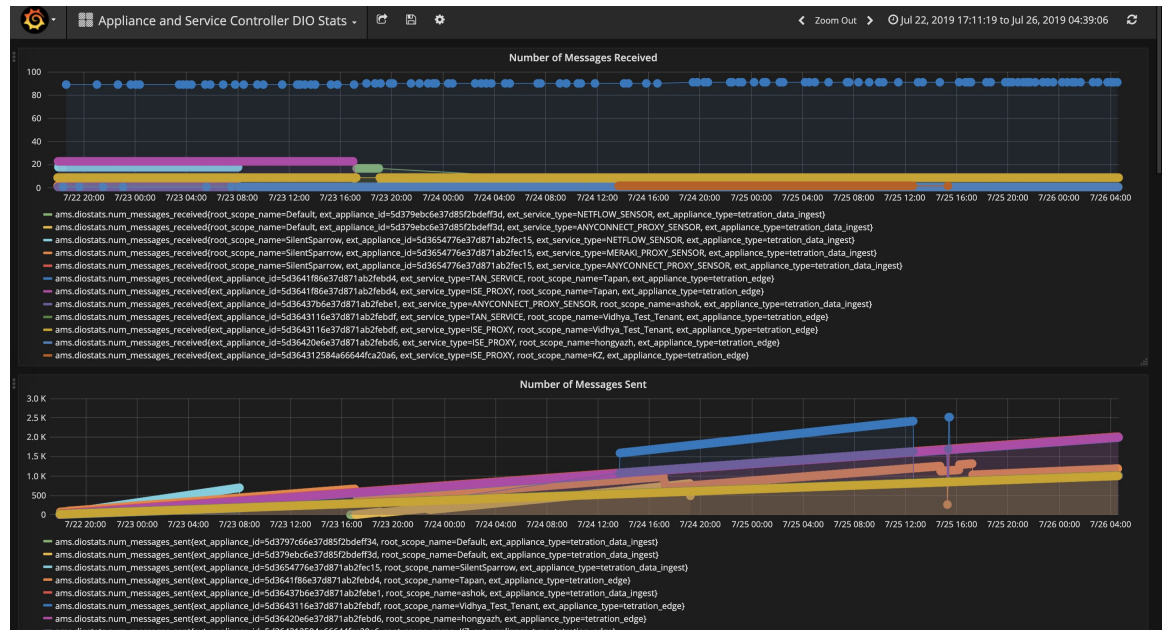
Figure 155: AnyConnect dashboard



## Appliance and Service DIO Dashboard

Appliance and service DIO dashboard provides information about number of messages exchanged in the Kafka topic on which the appliance manager and appliance/service controllers communicate. Metrics such as number of messages received, number of messages sent, number of messages failed are included in this dashboard. In addition, the last offset read by the controllers are also provided to understand whether the controller is lagging behind in processing the control messages from the manager.

Figure 156: Appliance and service DIO dashboard



## General Troubleshooting Guidelines

Once a connector shows in active state in connectors page in Secure Workload, no action is needed on the appliance where the connector is enabled; user does not need to log into it. If that is not happening, following information helps to troubleshoot such problems.

In normal conditions, on the appliance:

- `systemctl status tet_vm_setup.service` reports an *inactive* service with *SUCCESS* exit status.
- `systemctl status tet-nic-driver` reports an *active* service.
- `supervisorctl status tet-controller` reports *RUNNING* service. This indicates that the appliance controller is up and running.
- `docker network ls` reports three networks: *bridge*, *host*, and *none*.
- `docker ps` reports the containers that are running on the appliance. Typically, when a connector is enabled successfully on an appliance, a Docker container is instantiated on the appliance. For Syslog, Email, Slack, PagerDuty and Kinesis connectors, a Secure Workload alert notifier service is instantiated as a Docker container on Secure Workload edge appliance.
- `docker logs <cid>` for each container should report that tet-netflowsensor entered *RUNNING* state.
- `docker exec <cid> ifconfig` reports only one interface, besides the loopback.
- `docker exec <cid> netstat -rn` reports the default gateway.
- `cat /local/tetration/appliance/appliance.conf` on the appliance to see the list of Docker services running on the appliance. It includes details about service ID, connector ID, container, image ID and port mappings (if applicable). On a Secure Workload Ingest appliance, at most three services are running on the appliance. The port mappings and Docker volumes that are mounted on the containers are available in this file.

**Figure 157: Secure Workload appliance deployment service and status**

```
[root@esx-2106-ingest tetter]# systemctl status tet_vm_setup.service
• tet_vm_setup.service - Tetration Appliance Setup
 Loaded: loaded (/etc/systemd/system/tet_vm_setup.service; enabled; vendor preset: disabled)
 Active: inactive (dead) since Sat 2019-07-27 23:51:29 UTC; 21h ago
 Main PID: 1249 (code=exited, status=0/SUCCESS)

Jul 27 23:51:12 localhost.localdomain python[1249]: mount: /dev/sr0 is write-protected, mounting read-only
Jul 27 23:51:29 esx-2106-ingest python[1249]: Docker version 18.09.8, build 0dd43dd87f
Jul 27 23:51:29 esx-2106-ingest python[1249]: REPOSITORY TAG IMAGE ID CREATE... SIZE
Jul 27 23:51:29 esx-2106-ingest python[1249]: userPrivateKey.key
Jul 27 23:51:29 esx-2106-ingest python[1249]: intermediateCA.cert
Jul 27 23:51:29 esx-2106-ingest python[1249]: kafkaBrokerIps.txt
Jul 27 23:51:29 esx-2106-ingest python[1249]: userCA.cert
Jul 27 23:51:29 esx-2106-ingest python[1249]: kafkaCA.cert
Jul 27 23:51:29 esx-2106-ingest python[1249]: topic.txt
Jul 27 23:51:29 esx-2106-ingest python[1249]: Created symlink from /etc/systemd/system/multi-user.target.wants/s...vice.
Hint: Some lines were ellipsized, use -l to show in full.
[root@esx-2106-ingest tetter]#
```

Figure 158: Secure Workload network driver service status

```
[root@esx-2106-ingest tetter]# systemctl status tet-nic-driver.service
● tet-nic-driver.service - NIC network driver plugin for Docker
 Loaded: loaded (/etc/systemd/system/tet-nic-driver.service; enabled; vendor preset: disabled)
 Active: active (running) since Sat 2019-07-27 23:51:12 UTC; 21h ago
 Main PID: 733 (nic)
 Memory: 4.4M
 CGroup: /system.slice/tet-nic-driver.service
 └─733 /usr/local/tet/nic-driver/nic -log-level debug

Jul 27 23:51:12 localhost.localdomain systemd[1]: Started NIC network driver plugin for Docker.
Jul 27 23:51:12 localhost.localdomain systemd[1]: Starting NIC network driver plugin for Docker...
Jul 27 23:51:12 localhost.localdomain nic[733]: time="2019-07-27T23:51:12Z" level=info msg="NIC network driver started"
Hint: Some lines were ellipsized, use -l to show in full.
[root@esx-2106-ingest tetter]#
```

Figure 159: Appliance controller status

```
[root@esx-2106-ingest tetter]# supervisorctl status tet-controller
tet-controller RUNNING pid 1971, uptime 21:43:29
[root@esx-2106-ingest tetter]#
```

If any of the preceding doesn't hold true, check the deployment script logs in `/local/tetration/logs` for the reason why the appliance and/or the connector deployment failed.

You can troubleshoot any other connector registration/connectivity issues as follows.

```
docker exec <cid> ps -ef reports tet-netflowsensor-engine, /usr/local/tet/ tet-netflowsensor
-config /usr/local/tet-netflow/conf/tet-netflow.conf instances, along with the process manager
/usr/bin/supervisord -c /usr/local/tet-netflow/ conf/supervisord.conf -n instance.
```

Figure 160: Running processes on Secure Firewall ASA connector in Secure Workload Ingest appliance

```
[root@esx-2106-ingest tetter]# docker ps
CONTAINER ID IMAGE PORTS NAMES
c82decfaa877 asa_sensor-3.4.2.52465.appliance.demo.mrpm.build-asa:5d3ce5e43649723890271dd3 172.29.142.27:4729->4729/udp asa-5d3ce5e43649723890271dd3
... 22 hours ago Up 22 hours
eddd5cd59839 aws_sensor-3.4.2.52465.appliance.demo.mrpm.build-aws:5d3ce3b73649723890271dce aws-5d3ce3b73649723890271dce
... 22 hours ago Up 22 hours
[root@esx-2106-ingest tetter]# docker exec c8 ps -ef
UID PID PPID C STIME TTY TIME CMD
root 1 0 0 00:01 ? 00:00:15 /usr/bin/python /usr/bin/supervisord -c /usr/local/tet-netflow/conf/supe
rvisord.conf -n
root 8 1 0 00:01 ? 00:02:24 /usr/local/tet-netflow/tet-netflowsensor-engine -ctrl-config /usr/local/
tet-netflow/conf/tet-controller.conf -upgrade-script /usr/local/tet-netflow/scripts/check_config_update.sh -service /usr
/local/tet-netflow/tet-netflowsensor -config /usr/local/tet-netflow/conf/tet-netflow.conf
root 27002 8 0 21:31 ? 00:00:00 /usr/local/tet-netflow/tet-netflowsensor -config /usr/local/tet-netflow/
conf/tet-netflow.conf
root 27024 0 0 21:32 ? 00:00:00 ps -ef
[root@esx-2106-ingest tetter]#
```

## Log Files

The following commands can be used to view the logs from various services on the appliance.

- `/local/tetration/logs/tet-controller.log` shows the logs of the appliance controller.
- `docker exec <cid> cat /local/tetration/logs/tet-controller.log` shows the logs of the service controller on the connector.
- `docker exec <cid> cat /local/tetration/logs/tet-netflow.log` shows the logs of the connector service.
- `docker exec <cid> cat /local/tetration/logs/tet-ldap-loader.log` shows the logs of LDAP snapshot creation (if LDAP config is applicable for the connector).

- `docker exec <cid> cat /local/tetration/logs/check_conf_update.log` shows the configuration update polling logs (for connectors on the Ingest appliance).




---

**Note** There are allowed set of commands on Secure Workload that can pull these logs from the appliance and/or connectors directly. For more information, see [Allowed set of commands](#).

---

## Debug Mode

The default logging level for the appliance/service controller and connector service is set to *info* level. For troubleshooting issues, we may need to set the agent in *debug* mode. To do this, update the log configuration on the appliance/connector on Secure Workload directly for the desired appliance/connector. The log levels for both the controller and services are updated if the configuration is updated on the connector. For more information, see [Log Configuration](#).

# Cisco Secure Firewall Management Center

Combine the power of Secure Workload with the power of Cisco Secure Firewall (formerly known as Cisco Firepower) for a security solution that makes use of:

- Segmentation

Firewall-based segmentation is suitable for workloads where software agents are not installed. However, you can also use this method for agent-based workloads. You can easily and broadly apply different sets of policies for traffic entering your network, for traffic exiting your network, and for traffic between workloads within your network.

- Virtual Patching

Virtual patching adds Cisco Intrusion Prevention System (IPS) protection to workloads where software agents are installed. You can use this method to protect your application from malicious traffic. When you configure virtual patching on Secure Workload, it publishes the Common Vulnerabilities and Exposures (CVEs) to Cisco Secure Firewall for consideration while creating the IPS policies.

Virtual patching adds Cisco Intrusion Prevention System (IPS) protection to workloads where software agents are installed. You can use virtual patching rules to protect your application from malicious traffic. You can create virtual patching rules by filtering the most critical vulnerabilities in your workloads using Cisco Security Risk Score or CVSS V2 or V3 scores and the corresponding attributes. After virtual patching is configured on Secure Workload, the Common Vulnerabilities and Exposures (CVEs) are published to Cisco Secure Firewall for consideration while creating the IPS policies.

With this integration, Secure Workload automatically enforces and manages segmentation policies on the Secure Firewall Threat Defense (formerly known as Firepower Threat Defense) firewalls managed by the Secure Firewall Management Center instance. Policies are updated dynamically, and the set of workloads to which policies apply is refreshed continually as the application environment changes.

Network inventory is dynamically updated by Secure Workload inventory filters on which your segmentation policies are based; when workloads are added, changed, or removed from your network, Secure Workload automatically updates the Dynamic Objects in the Secure Firewall Management Center on which the corresponding access control rules are based. All enforced policy changes are automatically deployed to

managed Secure Firewall Threat Defense (formerly known as Firepower Threat Defense or FTD) devices; you never need to redeploy changes in Secure Firewall Management Center.

For complete information about this integration, including more details about how it works, supported platforms, limitations, setup instructions for both products, and troubleshooting information, see [Cisco Secure Workload and Cisco Secure Firewall Management Center Integration Guide](#).





## CHAPTER 5

# Manage Inventory for Secure Workload

Inventory is the IP addresses of all the workloads on your network, annotated with labels and other data that describes them. Your inventory includes workloads running on bare metal or virtual machines, in containers, and in the cloud. If applicable, it may also include workloads running on partner networks.

Collecting inventory data is an iterative process. Data from different sources for a single IP address can be merged, and new and changed IP addresses can be updated. Over time, management of your inventory should become increasingly dynamic.

You will work with and group your inventory using searches, filters, and scopes, based on the labels and annotations that are associated with each inventory item. Policies are applied to groups of workloads that are defined by the filters and scopes you define for your inventory.

Options for working with inventory vary depending on your role, but may include **Search**, **Filters**, and **Upload**.

**Table 17: Feature Information**

| Feature Name                                                                                                 | Release     | Feature Description                                                                                                                                                                                                                         | Where to Find                                                       |
|--------------------------------------------------------------------------------------------------------------|-------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------|
| Inventory Enhancements                                                                                       | 3.9         | You can manage and track the inventory of workloads, devices, and resources within the network, as well as the associated labels and subnets.                                                                                               | <a href="#">Rules for Creating Inventory Filters.</a>               |
| Integration of Cisco Vulnerability Management for Deep CVE Insights with Cisco Risk Score for Prioritization | 3.9 Patch 2 | You can use the Cisco Security Risk Scores of the CVEs to create inventory filters, microsegmentation policies to block communication from the impacted workloads, and virtual patching rules to publish the CVEs to Cisco Secure Firewall. | <a href="#">Cisco Security Risk Score-Based Filter, on page 410</a> |

| Feature Name                                                    | Release     | Feature Description                                                                                                                                                                                                                                              | Where to Find                                                 |
|-----------------------------------------------------------------|-------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------|
| Visibility and Enforcement of Well-known IPv4 Malicious Traffic | 3.9 Patch 2 | You can now identify any traffic to and from the workloads to well-known malicious IPv4 addresses. You can also create policies to block any traffic to these malicious IPs using a pre-defined read-only inventory filter titled <b>Malicious inventories</b> . | <a href="#">Malicious Inventory-Based Filter, on page 412</a> |

- [Workload Labels, on page 334](#)
- [Scopes and Inventory, on page 347](#)
- [Filters, on page 377](#)
- [Review Scope/Filter Change Impact, on page 382](#)
- [Inventory Profile, on page 387](#)
- [Workload Profile, on page 388](#)
- [Software Packages, on page 400](#)
- [Vulnerability Data Visibility, on page 403](#)
- [Service Profile, on page 412](#)
- [Pod Profile, on page 413](#)
- [Container Vulnerability Scanning, on page 414](#)

## Workload Labels

Labels (sometimes called tags, annotations, attributes, metadata, or context, though these terms are not necessarily always completely synonymous) are key to the power of Secure Workload.

Human-readable labels describe your workloads in terms of their function, location, and other criteria.

Secure Workload supports the following methods for adding user labels:

- Discovery by Secure Workload agents running on inventory items
- Manual import from uploading Comma Separated Value (CSV) files
- Manual assignment through the user interface
- Automated import through [Connectors for Endpoints](#)
- Automated import through Connectors for Inventory Enrichment
- Automated import of orchestrator generated and custom labels (See [External Orchestrators in Secure Workload](#))
- Automated import from cloud connectors (See [Cloud Connectors](#))
- You can specify inventory labels when creating the installer script. All agents installed using the script are automatically tagged with such labels. Only Linux and Windows workload deployments support this feature.



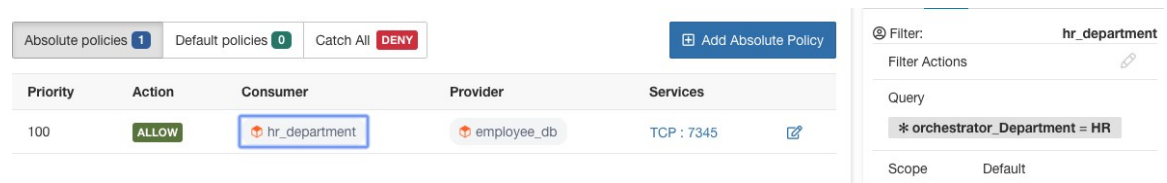
## Importance of Labels

Labels allow you to define a logical policy. For example:

*allow traffic from consumer hr\_department to provider employee\_db*

Instead of specifying the members of the consumer and provider workload groups, we can define the logical policy using the labels as shown in the following figure. Note that this allows the membership of the consumer and provider groups to be dynamically modified without the need to modify the logical policy. As workloads are added and removed from the fleet, Secure Workload is notified by services you have configured, such as external orchestrators and cloud connectors. This enables Secure Workload to evaluate the membership of the consumer group *hr\_department* and the provider group *employee\_db*.

Figure 161: Example policy with labels



## Subnet-based Label Inheritance

Subnet-based label inheritance is supported. The smaller subnets and IP addresses inherit labels from larger subnets they fall under when one of the following conditions is satisfied:

- The label is missing from the list of labels for the smaller subnet/address.
- The label value for the smaller subnet/address is empty.

Consider the following example:

| IP          | Name     | Purpose    | Environment | Spirit-Animal |
|-------------|----------|------------|-------------|---------------|
| 10.0.0.1    | server-1 | webtraffic | production  |               |
| 10.0.0.2    |          |            |             | frog          |
| 10.0.0.3    |          |            |             | eagle         |
| 10.0.0.0/24 | web-vlan |            | integration |               |
| 10.0.0.0/16 |          | webtraffic |             | badger        |
| 10.0.0.0/8  |          |            | test        | bear          |

The labels for IP address 10.0.0.3 are {"name": "web-vlan", "purpose": "webtraffic", "environment": "integration", "spirit-animal": "eagle"}.

## Label Prefixes

Labels are automatically displayed with a prefix that identifies the source of the information.

All user labels are prefixed by \* in the UI (*user\_* in OpenAPI). In addition, labels automatically imported from external orchestrators or from cloud connectors are prefixed with *orchestrator\_*. For labels imported from endpoint connectors, see details in [Connectors for Endpoints](#), but may include labels prefixed with *ldap\_*.

For example, a label with a key of *department* imported from user-uploaded CSV files appear in the UI as \**department*, and in OpenAPI as *user\_department*. A label with a key of *location* imported from an external orchestrator appear in the UI as \**orchestrator\_location*, and in OpenAPI as *user\_orchestrator\_location*.

The following figure shows an example of inventory search using the orchestrator-generated label using the prefix:

*orchestrator\_system/os\_image*:

**Figure 162: Example inventory search with orchestrator generated labels**

The screenshot shows a search interface with a filter bar containing the text: `* orchestrator_system/os_image contains Ubuntu 16.04`. To the right of the filter bar, it says "Total inventory: 196,294". Below the filter bar, it says "Showing 20 of 27 matching results" and "Results restricted to root scope".

| Hostname                   | VRF     | Address       | OS     |
|----------------------------|---------|---------------|--------|
| enforcement-scale-15-bare1 | Default | 192.168.60.21 | Ubuntu |
| enforcement-scale-15-bare2 | Default | 192.168.60.22 | Ubuntu |
| enforcement-scale-15-bare2 | Default | 192.168.10.22 | Ubuntu |
| enforcement-scale-15-bare2 | Default | 172.0.22.1    | Ubuntu |
| enforcement-scale-15-kube1 | Default | 192.168.50.11 | Ubuntu |
| enforcement-scale-15-kube1 | Default | 192.168.10.11 | Ubuntu |
| enforcement-scale-15-kube1 | Default | 172.0.1.1     | Ubuntu |
| enforcement-scale-15-kube1 | Default | 172.17.0.1    | Ubuntu |
| enforcement-scale-15-kube2 | Default | 192.168.50.12 | Ubuntu |

## Labels Generated by Cloud Connectors

These labels apply to data from AWS and Azure. The source for these labels is workloads and network interfaces of an AWS VPC or Azure VNet. The tags from the source are merged and displayed in Secure Workload. For example, if the workload tag is

```
env: prod
```

and the network interface tag is

```
env: prod
```

, the label value in Secure Workload is

```
prod, test
```

, which is displayed under the **orchestrator\_env** column on the respective connector page.

For labels specific to AKS, EKS, and GKE, see also Labels Related to Kubernetes Clusters.

**Table 18: Labels in Inventory Gathered Using a Cloud Connector**

| Key                              | Value                                                                           |
|----------------------------------|---------------------------------------------------------------------------------|
| orchestrator_system/orch_type    | AWS or Azure                                                                    |
| orchestrator_system/cluster_name | <Cluster_name is the name given by the user for this connector's configuration> |
| orchestrator_system/name         | <Name of connector>                                                             |
| orchestrator_system/cluster_id   | <Virtual network ID>                                                            |

### Instance-Specific Labels

The following labels are specific to each node:

| Key                                      | Value                                                                                               |
|------------------------------------------|-----------------------------------------------------------------------------------------------------|
| orchestrator_system/workload_type        | vm                                                                                                  |
| orchestrator_system/machine_id           | <InstanceID assigned by the platform>                                                               |
| orchestrator_system/machine_name         | <PublicDNS(FQDN) given to this node by AWS> –or–<br><InstanceName in Azure>                         |
| orchestrator_system/segmentation_enabled | <Flag to determine if segmentation is enabled on the inventory>                                     |
| orchestrator_system/virtual_network_id   | <ID of virtual network the inventory belongs to>                                                    |
| orchestrator_system/virtual_network_name | <Name of virtual network the inventory belongs to>                                                  |
| orchestrator_system/interface_id         | <Identifier of elastic network interface attached to this inventory>                                |
| orchestrator_system/region               | <Region the inventory belongs to>                                                                   |
| orchestrator_system/resource_group       | (This tag applies to Azure inventory only)                                                          |
| orchestrator_<Tag Key>                   | <Tag Value> Key-value pair for any number of custom tags assigned to inventory in the cloud portal. |

## Labels Related to Kubernetes Clusters

The following information applies to plain-vanilla Kubernetes, OpenShift, and to Kubernetes running on supported cloud platforms (EKS, AKS, and GKE).

For each object type, Secure Workload imports inventory live from a Kubernetes cluster, including labels associated with the object. Label keys and values are imported as-is.

In addition to importing the labels defined for the Kubernetes objects, Secure Workload also generates labels that facilitate the use of these objects in inventory filters. These additional labels are especially useful in defining scopes and policies.

### Generated labels for all resources

Secure Workload adds the following labels to all the nodes, pods and services retrieved from the Kubernetes/OpenShift/EKS/AKS/GKE API server.

| Key                              | Value                                                         |
|----------------------------------|---------------------------------------------------------------|
| orchestrator_system/orch_type    | kubernetes                                                    |
| orchestrator_system/cluster_id   | <UUID of the cluster's configuration in /product/>            |
| orchestrator_system/cluster_name | <Name of kubernetes cluster>                                  |
| orchestrator_system/name         | <Name of connector>                                           |
| orchestrator_system/namespace    | <The Kubernetes/OpenShift/EKS/AKS/GKE namespace of this item> |

### Node-specific labels

The following labels are generated for nodes only.

| Key                                           | Value                                                |
|-----------------------------------------------|------------------------------------------------------|
| orchestrator_system/workload_type             | machine                                              |
| orchestrator_system/machine_id                | <UUID assigned by Kubernetes/OpenShift>              |
| orchestrator_system/machine_name              | <Name given to this node>                            |
| orchestrator_system/kubelet_version           | <Version of the kubelet running on this node>        |
| orchestrator_system/container_runtime_version | <The container runtime version running on this node> |

### Pod-specific labels

The following labels are generated for pods only.

| Key                                  | Value                                                                  |
|--------------------------------------|------------------------------------------------------------------------|
| orchestrator_system/workload_type    | pod                                                                    |
| orchestrator_system/pod_id           | <UUID assigned by Kubernetes/OpenShift>                                |
| orchestrator_system/pod_name         | <Name given to this pod>                                               |
| orchestrator_system/hostnetwork      | <true/false> reflecting whether the pod is running in the host network |
| orchestrator_system/machine_name     | <Name of the node the pod is running on>                               |
| orchestrator_system/service_endpoint | [List of service names this pod is providing]                          |

### Service-specific labels

The following labels are generated for services only.

| Key                               | Value                        |
|-----------------------------------|------------------------------|
| orchestrator_system/workload_type | service                      |
| orchestrator_system/service_name  | <Name given to this service> |

- (For cloud-managed Kubernetes only) Services of ServiceType: LoadBalancer are supported only for gathering metadata, not for collecting flow data or for policy enforcement.



**Tip** Filtering items using **orchestrator\_system/service\_name** is not the same as using **orchestrator\_system/service\_endpoint**.

For example, using the filter **orchestrator\_system/service\_name = web** selects all *services* with the name **web** while **orchestrator\_system/service\_endpoint = web** selects all *pods* that provide a service with the name **web**.

### Labels Example for Kubernetes Clusters

The following example shows a partial YAML representation of a Kubernetes node and the corresponding labels imported by Secure Workload.

```
- apiVersion: v1
 kind: Node
 metadata:
 annotations:
 node.alpha.kubernetes.io/ttl: "0"
 volumes.kubernetes.io/controller-managed-attach-detach: "true"
 labels:
 beta.kubernetes.io/arch: amd64
 beta.kubernetes.io/os: linux
 kubernetes.io/hostname: k8s-controller
```

**Table 19: Label Keys Imported from Kubernetes**

| Imported label keys                                                            |
|--------------------------------------------------------------------------------|
| orchestrator_beta.kubernetes.io/arch                                           |
| orchestrator_beta.kubernetes.io/os                                             |
| orchestrator_kubernetes.io/hostname                                            |
| orchestrator_annotation/node.alpha.kubernetes.io/ttl                           |
| orchestrator_annotation/volumes.kubernetes.io/controller-managed-attach-detach |
| orchestrator_system/orch_type                                                  |
| orchestrator_system/cluster_id                                                 |
| orchestrator_system/cluster_name                                               |
| orchestrator_system/namespace                                                  |

| Imported label keys                           |
|-----------------------------------------------|
| orchestrator_system/workload_type             |
| orchestrator_system/machine_id                |
| orchestrator_system/machine_name              |
| orchestrator_system/kubelet_version           |
| orchestrator_system/container_runtime_version |

## Importing Custom Labels

You can upload or manually assign custom labels to associate user-defined data with specific hosts. This user-defined data is used to annotate associated flows and inventory.

There are limits on the number of IPv4/IPv6 addresses/subnets that can be labeled across all root scopes, regardless of label source (whether manually entered or uploaded, ingested using connectors or external orchestrators, and so on) For details, see [Label Limits](#).

## Guidelines for Uploading Label Files

### Procedure

- 
- Step 1** To view a sample file, in the left pane, select **Organize > Label Management > User Defined Label Upload**, and then click **Download a Sample**.
  - Step 2** The CSV files used to upload the user labels must include a label key (IP address).
  - Step 3** To use non-English characters in labels, the CSV file must be in UTF-8 format.
  - Step 4** Ensure the CSV files meet the guidelines described in the Label Key Schema section.
  - Step 5** All uploaded files must follow the same schema.
- 

## Label Key Schema

### Guidelines governing column names

- There must be one column with a header “IP” in the label key schema. Additionally, there must be at least one other column with attributes for the IP address.
- The column “VRF” has special significance in the label schema. If provided, it should match the root scope to which you upload the labels. It’s mandatory when uploading the CSV file using the [Scope-Independent APIs](#).
- Column names may contain only the following characters: Letters, numbers, space, hyphen, underscore, and slash.
- Column names cannot exceed 200 characters.

- Column names cannot be prefixed with “orchestrator\_”, “TA\_”, “ISE\_”, “SNOW\_”, nor “LDAP\_” since these can conflict with labels from internal applications.
- The CSV file should not contain duplicate column names.

### Guidelines governing column values

- Values are limited to 255 characters. However, they should be as short as possible while still being clear, distinctive, and meaningful to users.
- Keys and values are not case sensitive. However, consistency is recommended.
- Addresses appearing under the “IP” column should conform to the following format:
  - IPv4 addresses can be of the format “x.x.x.x” and “x.x.x.x/32”.
  - IPv4 subnets should be of the format “x.x.x.x/<netmask>”, where netmask is an integer from 0 to 31.
  - IPv6 addresses in the Long format (“x:x:x:x:x:x:x:x” or “x:x:x:x:x:x:x/x/128”) and the Canonical format (“x::x” or “x::x/128”) are supported.
  - IPv6 subnets in the Long format (“x:x:x:x:x:x:x/x/<netmask>”) and the Canonical format (“x::x/<netmask>”) are supported. Netmask must be an integer from 0 to 127.

The order of the columns does not matter. The first 32 user-defined columns will automatically be enabled for label. If more than 32 columns are uploaded, up to 32 can be enabled using the checkboxes on the right-side of the page.

## Upload Custom Labels

The following steps explain how users with **Site Admin**, **Customer Support** or a root **scope owner** role can upload labels.

### Before you begin

To upload the custom labels, create a CSV file according to the ‘Guidelines for Uploading Label Files’ section.

### Procedure

- 
- Step 1** In the left pane, select **Organize > User Defined Label Upload > CSV Upload**, and then under **Upload New Labels**, click **Select File**.
- Step 2** In the left pane, select **Organize > Label Management**, and then under **Upload New Labels**, click **Select File**.
- Step 3** Select the operation-Add, Merge, or Delete.
- **Add:** Appends labels to new and existing addresses/subnets. Resolves conflicts by selecting newer labels over existing ones. For example, if labels for an address in the database are `{"foo": "1", "bar": "2"}` and the CSV file contains `{"z": "1", "bar": "3"}`, add sets labels for this address to `{"foo": "1", "z": "1", "bar": "3"}`.
  - **Merge:** Merges labels to existing addresses/subnets. Resolves conflicts by selecting non-empty values over empty values. For example, if labels for an address in the database are `{"foo": "1", "bar": "2", "qux":`

"" , "corge": "4"} and the CSV file contains {"z": "1", "bar": "", "qux": "3", "corge": "4-updated"}, merge` sets labels for this address to {"foo": "1", "z": "1", "bar": "2", "qux": "3", "corge": "4-updated"}.

**Note** Value of “bar” in not reset to “”(empty), instead existing value of “bar”=”2” is preserved.

- **Delete:** This option removes labels for an address/subnet, which can significantly impact scopes, filters, policies, and enforced behavior. For important details, see *Delete Labels*.

**Important:** The Delete function, while uploading the custom labels, will remove ALL labels associated with the specified IP addresses/subnets, and is not limited to the columns listed in the CSV file. Therefore, the Delete operation must be used with caution.

**Step 4** Click **Upload**.

---

## Search Labels

Users with **Site Admin**, **Customer Support** or a root **scope owner** role can search for, view, and edit labels assigned to an IP address or subnet.

### Procedure

---

**Step 1** On the **Label Management** page, click **Search and Assign**.

**Step 2** In the **IP or Subnet** field, enter the IP address or subnet and click **Next**.

On the Assign Labels page, the existing labels for the entered IP address or subnet are displayed.

---

## Manually Assign or Edit Custom Labels

Users with **Site Admin**, **Customer Support**, or a root **scope owner** role can manually assign labels to a given IP address or subnet.

### Procedure

---

**Step 1** On the **Label Management** page, click **Search and Assign**.

**Step 2** In the **IP or Subnet** field, enter the IP address or subnet and click **Next**.

The Assign Labels page is displayed. Note that the existing labels will be displayed and can be edited.

**Step 3** To add a new label, in the **Labels for <IP address/subnet>** section, enter the label name and value, and then click **Confirm**. Click **Next**.

**Step 4** Review the changes and click **Assign** to commit them.

---



## Download Labels

Users with **Site Admin**, **Customer Support**, or a root **scope owner** role can download previously defined labels belonging to a root scope.

### Procedure

---

- Step 1** On the **Label Management** page, click **User Defined Label Upload**.
- Step 2** Under the **Download Existing Labels** section, click **Download Labels**.
- The labels used by Secure Workload are downloaded as a CSV file.
- 

## Change Labels



**Warning** If you need to change a label, do so cautiously, as doing so changes the membership in and effects of existing queries, filters, scopes, clusters, policies, and enforced behavior that are based on that label.

---

### Procedure

---

- Step 1** On the **Label Management** page, click the **Search and Assign** tab.
- Step 2** In the **IP or Subnet** field, enter the IP address or subnet and click **Next**.
- The labels used by Secure Workload for the entered IP address/subnet are displayed.
- Step 3** Under the **Actions** column, click the **Edit** icon to change the name and value of the required label.
- Step 4** Click **Confirm**, and then click **Next**.
- Step 5** Review the changes and click **Assign**.
- 

## Disable Labels

One way to change the schema is to disable the labels. *Proceed with caution.*

### Procedure

---

- Step 1** Navigate to the **Label Management** page.
- Step 2** For the required label, under the **Actions** column, select **Disable** and confirm to remove the label from the inventory by clicking **Yes**.
- If you decide to enable the label at a later time, click **Enable** to use the label.
-

## Review Label Change Impact

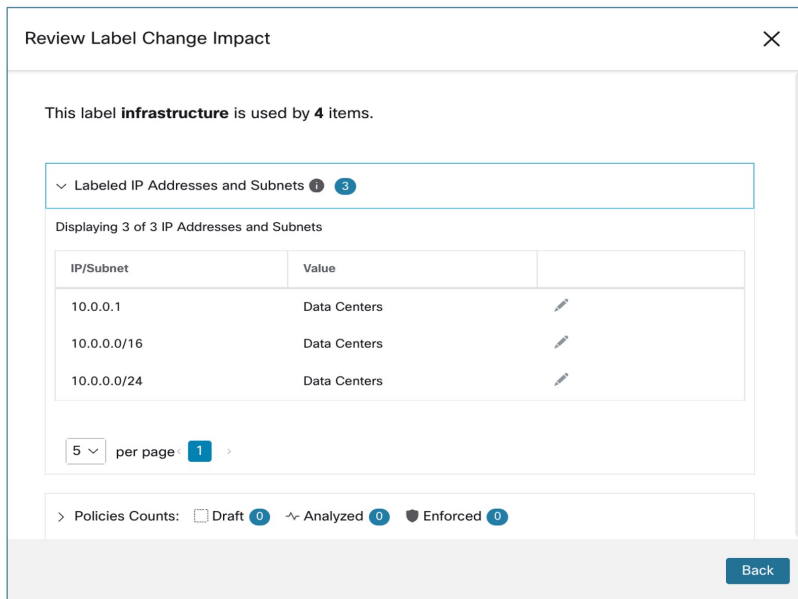
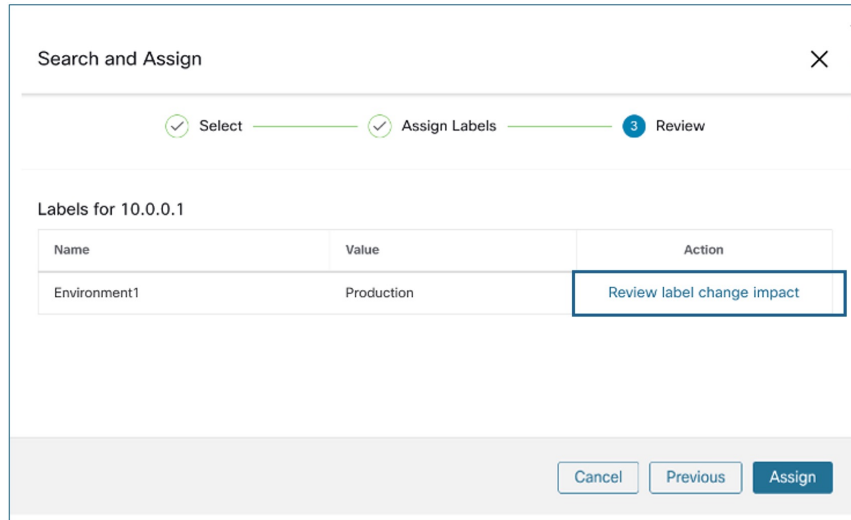
Users with **Site Admin**, **Customer Support**, or a **Root Scope Owner** role can view and edit the labels assigned to an IP address or subnet.

### Procedure

---

- Step 1** On the **Label Management** page, click **Search and Assign**.
- Step 2** In the **IP or Subnet** field, enter the IP address or subnet and click **Next**.  
Note that the existing labels inherited from the IP address or subnet are editable.
- Step 3** To add a new label, in the **Labels for IP address/subnet** section, enter the label name and value, and click **Confirm**.
- Step 4** Click **Next**.
- Step 5** Under the **Action** column, click the **Review Label Change Impact** link for the label that you want to review the details.
- Step 6** Click **Back** to close the page.
- Step 7** In the **Search and Assign** page, click **Assign** to commit the changes.

Figure 163: Search and Assign



## Delete Labels



**Caution**

One way to change the schema is to disable the labels and delete them. Proceed with caution. This action deletes the selected label that impacts all dependent **Filters** and **Scopes**. Ensure that these labels are not used. This action cannot be undone.

## Procedure

- Step 1** Disable the labels. See `disable_labels`.
- Step 2** Click the **TrashCan** icon and to confirm, click **Yes** to delete the label.

## View Labels Usage

The IP addresses/subnet inventory gets updated with the custom labels uploaded using CSV files or manually assigned by users. The labels are then used in defining the scopes and filters, and the application policies are created based on these filters. Therefore, understanding the usage of labels is critical as any modifications to the labels directly impacts the scopes, filters, and policies in Secure Workload.

To view the usage of labels:

## Procedure

- Step 1** On the **Label Management** page, the label keys, top five values of the labels in use, inventory, scopes, filters, and clusters using the custom labels are displayed.
- Step 2** Under the Usages column, click the count values against the inventory, scopes, or filters. For example, to view the scopes using the “Location” label, click the scope queries count.

**Figure 164: View Scopes of Selected Label**

| Label Management |              | User Defined Label Upload |               | Search and Assign |                |                 |         |
|------------------|--------------|---------------------------|---------------|-------------------|----------------|-----------------|---------|
| Label Source     |              | Filter by Label Key       |               |                   |                |                 |         |
| All              |              |                           |               |                   |                |                 |         |
|                  |              | Usages                    |               |                   |                |                 |         |
| Label Key        | Label Source | Inventory                 | Policy Counts | Scope Queries     | Filter Queries | Cluster Queries | Actions |
| > city           | User Defined | 0                         | 0             | 0                 | 0              | 0               | Enabled |
| > Department     | User Defined | 3                         | 0             | 0                 | 0              | 0               | Enabled |
| > location       | User Defined | 2                         | 0             | 0                 | 0              | 0               | Enabled |

The Scopes and Inventory page is displayed, and the query automatically filters the scopes with the selected label.

**Note** You can only view the usage of labels either uploaded using CSV files or those manually assigned to the IP address/subnet.

## Create a Process for Maintaining Labels

Your network and inventory will change, and you must plan to update labels to reflect those changes.

For example, if a workload is retired and its IP address is reassigned to a workload with a different purpose, you need to update the labels associated with that workload. This is true for both manually uploaded labels

and for labels maintained in and ingested from other systems such as a configuration management database (CMDB.)

Create a process to ensure that your labels are updated on a regular, ongoing basis, and add this process to your network-maintenance routine.

# Scopes and Inventory

## Scopes and Inventory Overview

This section provides visibility of the scope hierarchy, along with all the inventory it contains. Scopes categorize the available inventory using a hierarchical structure. For more information, see [Manage Inventory for Secure Workload, on page 333](#).

From the navigation pane, choose **Organize > Scopes and Inventory**, traverse down your scope hierarchy. Each scope is displayed in a scope card. The scope card displays the following:

- Scope name
- Number of children scope
- Inventory count
- (optional) Uncategorized inventory

Click on a scope card to update the pane for displaying details about the scope and the filtered list of all its inventory.

## Scope Design Principles

1. Match inventory to the scope tree according to a dynamic query match.
  - Match queries against IP or Subnet or Label (preferred)
  - Form a scope tree through conjunctive query at each layer.
2. Scope structure may be location specific-Combined Cloud vs Data Center and Cloud Specific vs Geographic location
3. Each layer of the scope tree should represent an anchor point for:
  - Policy control
  - Role Based Access Control (RBAC)
4. Keep the scope layer not too deep.
5. Ensure no overlapping of scopes:
  - Every child scope should be a subset of its parent scope.
  - Ensure nonoverlapping of sibling scopes, see [Scope Overlap](#).



**Note** Every organization is structured differently, and depending on your industry, it requires different approaches. Choose a focus area that helps in designing your scope hierarchy; location, environment, or application.



**Note** Do not use IP address or subnet to define scopes that involve Kubernetes inventory. You must use labels to define scopes and policies for these workloads. IP address alone is not sufficient to identify pod services because defining scopes using IP addresses produces unreliable results.

6. If a host has multiple interfaces, we recommend keeping all IPs belonging to the host under a single scope, so that we can discover and enforce required policies from a single location.
7. Keep the overall scope numbers within the supported limit (see the limit section)

### Key Features

Inventory count is displayed in the scopes card, providing a quick view into the number of workloads in the scope.

The filtering feature for both scopes and inventory helps to traverse down the scope tree or filter the scope hierarchy and inventory items of the selected scope.

## Scopes

Scopes are a foundational element to configuration and policy in Secure Workload. Scopes are a collection of workloads arranged in a hierarchy. Workloads labelled to serve as attributes that build a model about where it is, its role, and its function in your environment. Scopes provide a structure to support dynamic mechanisms like identification and attributes associated with an IP that may change over time.

Scopes are used to group datacenter applications and, along with the roles, enable fine grained control of their management. For example, Scopes are used throughout the product to define access to [Manage Policies Lifecycle in Secure Workload, on page 417](#), [Network Flows - Traffic Visibility](#) and [Filters](#).

Scopes are defined hierarchically as sets of trees with the root corresponding to a **VRF**. As a result, each Scope tree hierarchy represents disjoint data that does not overlap with another Scope tree, see [Scope Overlap](#).

### Scope Definition

Each individual Scope is defined with the attributes below:

| Attribute           | Description                                                                                                                           |
|---------------------|---------------------------------------------------------------------------------------------------------------------------------------|
| <b>Parent Scope</b> | The parent of the new scope defines the tree hierarchy structure.                                                                     |
| <b>Name</b>         | The name to identify the scope.                                                                                                       |
| <b>Type</b>         | This is used to specify different categories of inventory. If none are applicable, or the scope contains a mix, it can be left blank. |

| Attribute | Description                              |
|-----------|------------------------------------------|
| Query     | The Query defining the individual scope. |

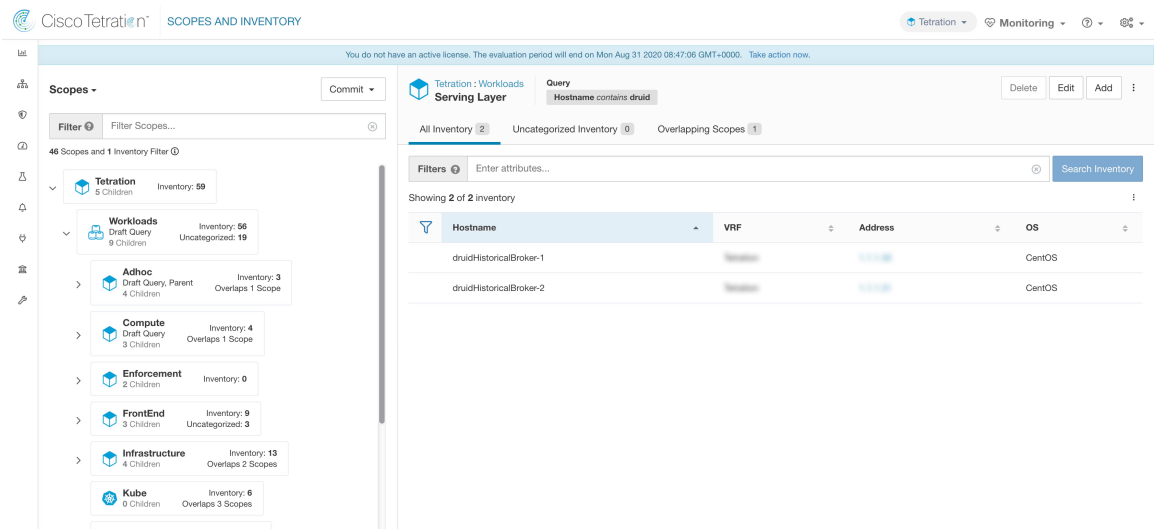


**Note** Scopes should be defined in a hierarchy that mimics the application ownership hierarchy of the organization.



**Note** Query may match against IP/Subnet or other Inventory attributes.

**Figure 165: Example of Traversing through Scope Hierarchy**



The scope directory displays the scope hierarchy and some details of each scope (for example, Inventory Count, number of child scopes, Workspaces). Clicking on a scope selects that scope and the details pane to the right updates with more information about that scope and that scope's inventory.

Figure 166: Inventory count

The screenshot shows the 'Scope Filter' interface. On the left, a sidebar lists 40 scopes with their respective inventory counts and child counts. The 'Adhoc' scope is selected, showing an inventory of 5. The main panel displays a query for 'Hostname contains adhoc' and a table of 5 inventory items:

| Hostname       | Address T1 | OS T1 |
|----------------|------------|-------|
| adhoc-1        | 4.4.1.1    | linux |
| adhoc-1        | 1.1.1.47   | linux |
| adhoc-2        | 4.4.2.1    | linux |
| adhoc-2        | 1.1.1.48   | linux |
| adhockafkaxi-1 | 1.1.1.55   | linux |

## Scope Filter

Users can use the Scope filter to quickly identify different scope details such as overlapping scopes and query. The filter feature is also helpful in identifying query changes, parent changes, etc.

| Field                         | Description                                                             |
|-------------------------------|-------------------------------------------------------------------------|
| <b>Name</b>                   | Filter by the name of the Scope or Inventory Filter.                    |
| <b>Description</b>            | Filter by text appearing in the description of a scope.                 |
| <b>Query</b>                  | Filter by fields or values used in the query.                           |
| <b>Query Change</b>           | Filter by scopes that have an uncommitted query.                        |
| <b>Parent Change</b>          | Filter by scopes that have been moved in the draft but not committed.   |
| <b>Is Inventory Filter</b>    | Show Inventory Filters that are restricted to their ownership scope.    |
| <b>Has Workspace</b>          | Filter by scopes that have a primary workspace.                         |
| <b>Has Enforced Workspace</b> | Filter by scopes that have a primary workspace that is enforced.        |
| <b>Has Overlaps</b>           | Filter by scopes that have inventory in common with a sibling scope.    |
| <b>Has Invalid Query</b>      | Filter by scopes that have a query that uses invalid or unknown labels. |

### Examples:

#### Has Overlaps



## Example of Scope Overlap

**Figure 167: Has Overlaps**

The screenshot shows the Tetraton interface with a filter 'Has Overlaps = true' applied to the Scopes section. The 'Scopes' sidebar on the left shows a tree view with 'Tetration' selected, containing 'Workloads', 'Compute', 'HDFS', and 'Namenodes'. Under 'Namenodes', 'PrimaryNamenode' and 'SecondaryNamenode' are shown with 'In Overlap' and 'Ove' (Overlap) indicators respectively. The main panel shows 'All Inventory 75' and 'Uncategorized Inventory 0'. A search bar contains 'Enter attributes...'. Below it, 'Workloads 44' and 'IP Addresses 31' are listed. A 'Load All' button is present. The inventory table below shows 20 of 44 items:

| Hostname ↑              | Address ↑  | OS ↑   |
|-------------------------|------------|--------|
| adhoc-1                 | 4.4.1.1    | linux  |
| adhoc-2                 | 1.1.1.48   | linux  |
| appServer-2             | 1.1.1.44   | linux  |
| collectorDatamover-1    | 100.64.0.1 | CentOS |
| collectorDatamover-2    | 1.1.1.27   | CentOS |
| collectorDatamover-2    | 100.64.1.1 | CentOS |
| druidHistoricalBroker-2 | 1.1.1.31   | CentOS |
| elasticsearch-1         | 1.1.1.40   | linux  |

For more information see [Scope Overlap](#)

## Parent Change

Scopes that are moved in the draft but not yet committed.

**Figure 168: Parent Change**

The screenshot shows the Tetraton interface with a filter 'Parent Changed = true' applied to the Scopes section. The 'Scopes' sidebar on the left shows a tree view with 'Tetration' selected, containing 'Workloads'. Under 'Workloads', 'Hyper Drive' is highlighted in blue, with 'Draft Changes, Parent' and 'Inventory: 0' indicators. The main panel shows 'Draft Parent Scope' and 'Tetration: Workloads: FrontEnd'. The inventory table is empty, displaying 'No Inventory'.

## Full Scope Queries

Figure 169: Example of Scope Hierarchy

The screenshot displays the 'Scopes' management interface. On the left, a tree view shows a hierarchy of scopes: Tetration (5 Children, Inventory: 75) -> Workloads (7 Children, Inventory: 75, Uncategorized: 31) -> FrontEnd (3 Children, Inventory: 12, Uncategorized: 3) -> Mongo (2 Children, Inventory: 3) -> MongoServer (6 Children, Inventory: 2). The right pane shows a query for 'MongoServer' with the query 'Hostname contains mongo'. Below the query, a table displays the results:

| Hostname  | Address  | OS    |
|-----------|----------|-------|
| mongodb-1 | 1.1.1.34 | linux |
| mongodb-2 | 1.1.1.35 | linux |

Scopes are defined hierarchically, the full query of the scope is defined as the logical ‘and’ of the scope along with all of its parents. Using the example above, assets assigned to the `Workloads:FrontEnd:Mongo`

Scope would match:

```
vrf_id = 676767 and (ip in 1.1.1.0/24) and (Hostname contains mongo).
```

Where `vrf_id = 676767` comes from the root scope query and `ip in 1.1.1.0/24` comes from the parent scope query.



**Note** It is a best practice to not have overlapping queries at the same level. This removes the importance of ordering and reduces confusion. See [Scope Overlap](#)

## Providing Access to Scopes

You can grant Read, Write, Execute, Enforce, and Owner abilities on Scopes. For more information see the **Roles** section in the *Secure Workload User Guide*.

A User is given access to a “sub-tree”. That is, the given Scope and all its children. Using the preceding example, you have the Read access to the `Workloads:FrontEnd` scope would, by inheritance, have read access to all the scopes under `Workloads:FrontEnd` including:

- `Workloads:FrontEnd:Mongo`
- `Workloads:FrontEnd:ElasticSearch`
- `Workloads:FrontEnd:Redis`
- etc. . .

It's possible to define Roles with access to multiple Scopes. For example, an “Mongo Admin” role might have Owner access to the Scopes:

- `Workloads:FrontEnd:Mongo:MongoServer`
- `Workloads:FrontEnd:Mongo:MongoDBArbiter`

Roles and Capabilities allow you to have horizontal access to the Scope hierarchy.

Scope Abilities are also inherited. For example, having the Write ability on a Scope allows one to also Read that information.

## Viewing Scope

Every user can view the scope tree they have access to. Users who have the Owner ability on the root scope have the ability to create, edit and delete scope in that tree. To access this view:

In the navigation bar on the left, click **Organize > Scopes and Inventory**.

You can traverse through the complete scope hierarchy (up to the root) for any Scopes you have access to. This complete traversal provides context as users can create policies to any Scope. Several actions can be performed on this page:

- Click the chevron in the scope hierarchy to show that scope's children.
- Clicking on a scope card will update the pane to the right to show details about that scope as well as a filterable list of all of its inventory.

**Figure 170: Example Non-Admin View**

The screenshot shows the 'Scopes' management interface. On the left, a list of scopes is displayed, with 'ResourceManagers' selected. The right pane shows the details for 'ResourceManagers', including a search bar and a table of inventory items.

| Hostname          | Address  | OS    |
|-------------------|----------|-------|
| resourceManager-1 | 1.1.1.16 | linux |
| resourceManager-2 | 1.1.1.17 | linux |

## Searching for flows referencing a scope

There are some shortcuts provided on the scopes page to help the user in scenarios they need to search for flows where one or both endpoints of the flow fall within a provided scope.

Figure 171: Searching for flows for a scope

The screenshot displays the 'Scopes' admin page. On the left, a tree view shows various scopes: Tetration (5 Children, Inventory: 75), Workloads (7 Children, Inventory: 75, Uncategorized: 31), Adhoc (2 Children, Inventory: 5), Collector (0 Children, Inventory: 7), Compute (2 Children, Inventory: 4), Enforcement (2 Children, Inventory: 0), FrontEnd (3 Children, Inventory: 12, Uncategorized: 3), Infrastructure (4 Children, Inventory: 14), and Serving Layer (2 Children, Inventory: 2). The 'Collector' scope is selected and highlighted with a red box. The main panel shows the 'Collector' scope details, including a query 'Hostname contains collector' and a table of inventory items. The 'More Scope Details' dropdown menu is open, showing options: 'Flow Search - As Consumer', 'Flow Search - As Provider', and 'Flow Search - Internal Traffic', with the first two options highlighted by a red box.

| Hostname             | Address    | OS     |
|----------------------|------------|--------|
| collectorDatamover-1 | 100.64.0.0 | CentOS |
| collectorDatamover-1 | 100.64.0.1 | CentOS |
| collectorDatamover-1 | 1.1.1.26   | CentOS |
| collectorDatamover-2 | 1.1.1.27   | CentOS |
| collectorDatamover-2 | 100.64.1.1 | CentOS |
| collectorDatamover-2 | 100.64.1.0 | CentOS |
| collectorDatamover-2 | 1.1.1.5    | CentOS |

After selecting desired scope in the scope tree (left side panel), as shown in the figure above, user can choose between the following three options:

1. *Flow Search - As Consumer* provides shortcut to the flow search page to help search for flows with selected scope as *Consumer Scope* for the flows. In other words, consumer or source endpoint in the flows belongs to the selected scope.
2. *Flow Search - As Provider* provides shortcut to the flow search page to help search for flows with selected scope as *Provider Scope* for the flows. In other words, provider or destination endpoint in the flows belongs to the selected scope.
3. *Flow Search - Internal Traffic* provides shortcut to the flow search page to help search for flows that are completely restricted to the selected scope. In other words, both endpoints of the flows (consumer and provider) belong to the selected scope.

## Creating a New Scope

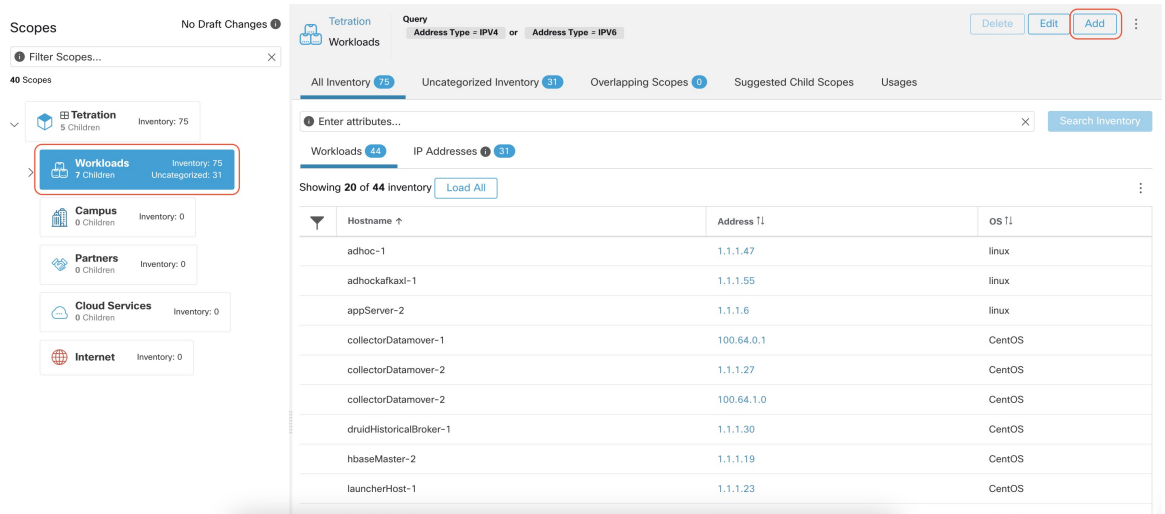
Child scopes are created on the **Scopes** admin page. This action requires the `SCOPE_OWNER` ability on the root scope. **Site Admins** are owners of all scopes.

Creating a child scope will impact the application inventory membership (member workloads) of the parent scope. As a result, the parent scope will be marked as having “draft changes”. The changes will need to be committed and dependent structures will need to be updated. See [Commit Changes](#).

### Procedure

- Step 1** In the navigation bar on the left, click **Organize > Scopes and Inventory**. The page displays the root Scopes corresponding to Tenants+VRFs already created on the system.
- Step 2** Select a child scope in the scope directory. You can filter the scopes first if necessary.
- Step 3** Click the **Add** button.

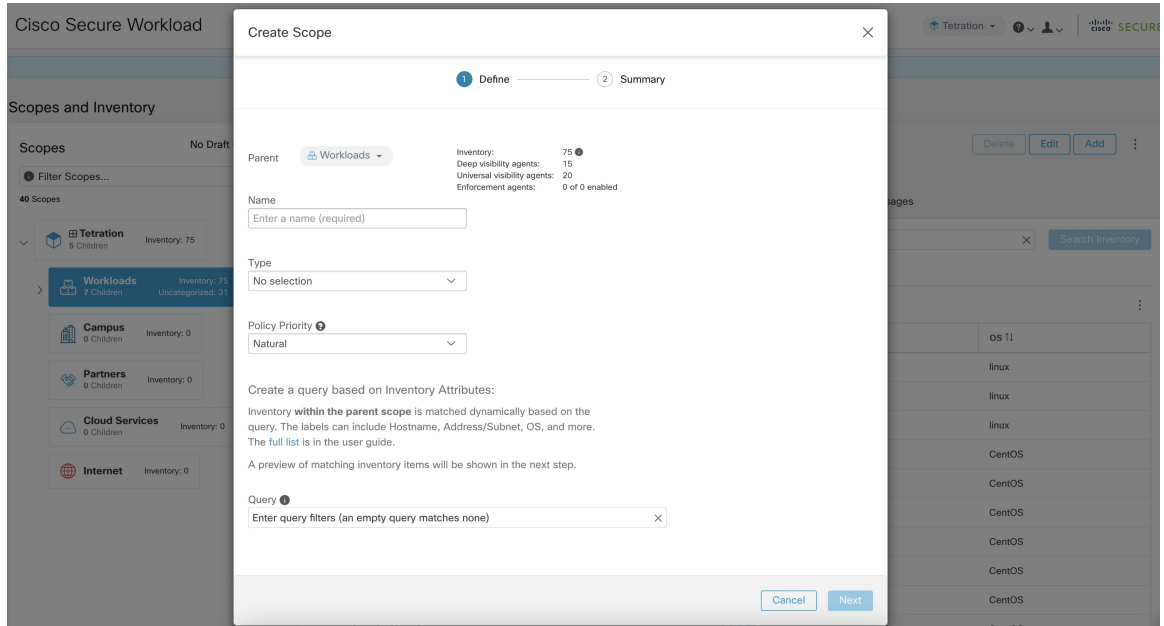
Figure 172: Scope Add Button



**Step 4** Enter the appropriate values in the following fields:

| Field         | Description                                                           |
|---------------|-----------------------------------------------------------------------|
| <b>Parent</b> | The parent of the new Scope.                                          |
| <b>Name</b>   | The name to identify the Scope. Must be unique under the parent scope |
| <b>Type</b>   | Select a category for the new Scope.                                  |
| <b>Query</b>  | The Query/Filter to match the assets.                                 |

Figure 173: Scope Create Modal

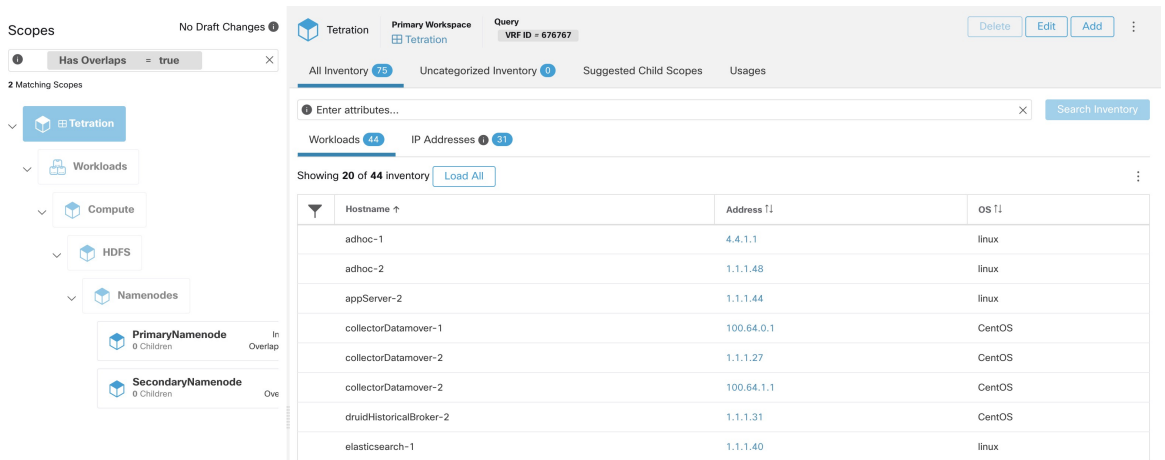


## Scope Overlap

While adding scopes, it is recommended to avoid overlapping scopes. When scopes overlap, policies generated for overlapping scopes can potentially end up confusing end users. This feature proactively notifies the user if there are any overlapping scope membership, that is, the same inventory belongs to more than one scope at the same depth in scope tree (sibling scopes). The goal is to avoid having the same workload exist in different parts of the scope tree.

To view which inventory items belong to multiple scopes, use the scope filter and enter the **Has Overlaps = true** facet.

Figure 174: Overlap facet in Scope filter



The list of overlapping scopes and the corresponding overlapping IP addresses can be viewed by traversing down the scope tree and selecting the **Overlapping Scopes** tab.

**Figure 175: Overlapping Scopes and IPs**

The screenshot shows the Cisco Tetration SCAPO AND INVENTORY interface. On the left, a 'Scopes' tree is visible with a filter 'Has Overlaps = true'. The 'Compute' scope is selected, showing its inventory of 4 items. On the right, the 'Overlapping Scopes' tab is active, displaying a table of inventory items overlapping with the 'Compute' scope. The table has columns for Hostname, VRF, Address, and OS.

| Hostname            | VRF | Address | OS     |
|---------------------|-----|---------|--------|
| namenode-1          |     |         | CentOS |
| resourceManager-1   |     |         | linux  |
| resourceManager-2   |     |         | linux  |
| secondaryNameNode-1 |     |         | linux  |

## Editing Scopes

Scopes can only be edited by users with the `SCOPE_OWNER` ability on the root scope. Site admins are owners of all scopes.

### Editing a scope name

Editing a scope name happens immediately and can take several minutes depending on the number of child scopes that need to be updated.



**Note** Flow searches by scope name will be impacted when changing the scope name.

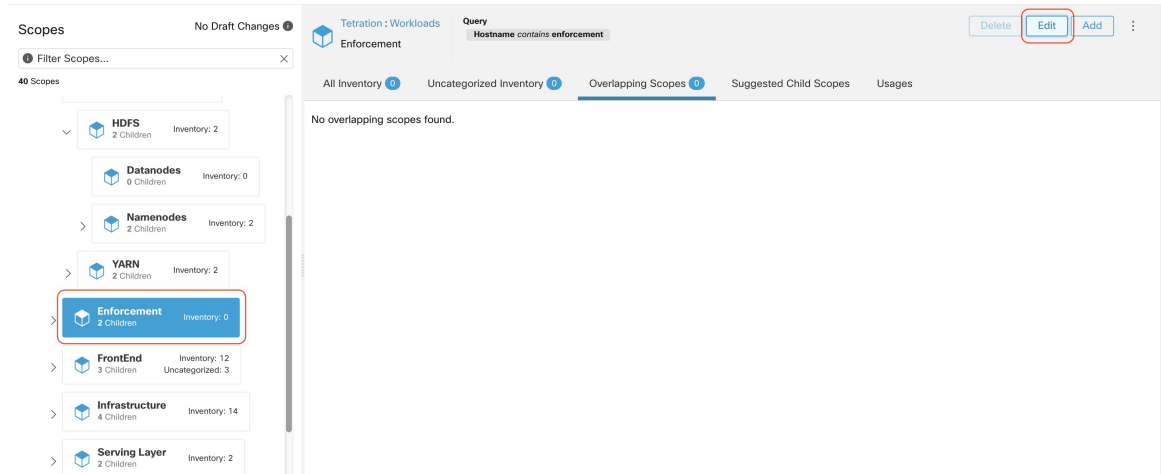
### Editing a scope query

When a scope's query is changed the direct parent and child scopes are impacted. Those scopes are marked as having 'draft changes' indicating changes have been made to the tree that have not been committed. Once all query updates have been completed, the user must click the **Commit Changes** button above the Scope Directory to make the change permanent. This will trigger a background task to update all of the scope queries and 'dynamic cluster queries' in the workspace.



**Warning** Updating a scope query can impact the scopes inventory membership (the workloads that are members of the scope). Changes will take effect during the **Commit Changes** process. To mitigate risks, you can compare membership changes for further impact analysis from the [Review Scope/Filter Change Impact](#) window. New host firewall rules will be inserted and any existing rules will be deleted on the relevant hosts.

Figure 176: Edit a Scope



To edit a scope:

## Procedure

- Step 1** Click on the **edit button** on the respective scope to be edited.
- Step 2** Edit the Name or Query for the selected scope.
- Step 3** Compare changes between the old and new Draft Query by following the **Review query change impact** link.
- Step 4** Click on **Save**. Name gets updated right away.
- Step 5** To update the Query of all scopes, Click the **Commit Changes** button.
- Step 6** You will get a popup confirmation which states the consequences of performing scope changes. The update is processed asynchronously in a background task.
- Step 7** Click on **Save**. Depending on the number of changes this can a minute or more.

Figure 177: Review query change impact

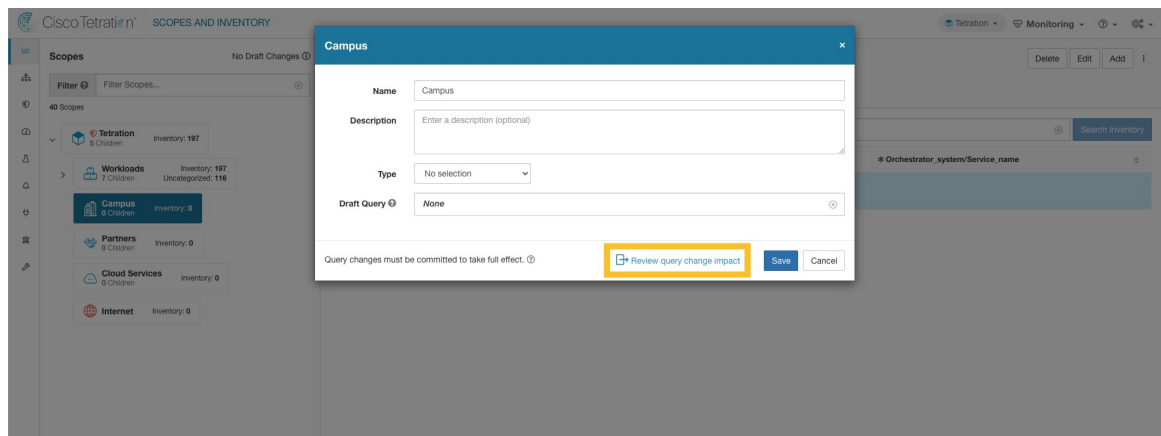
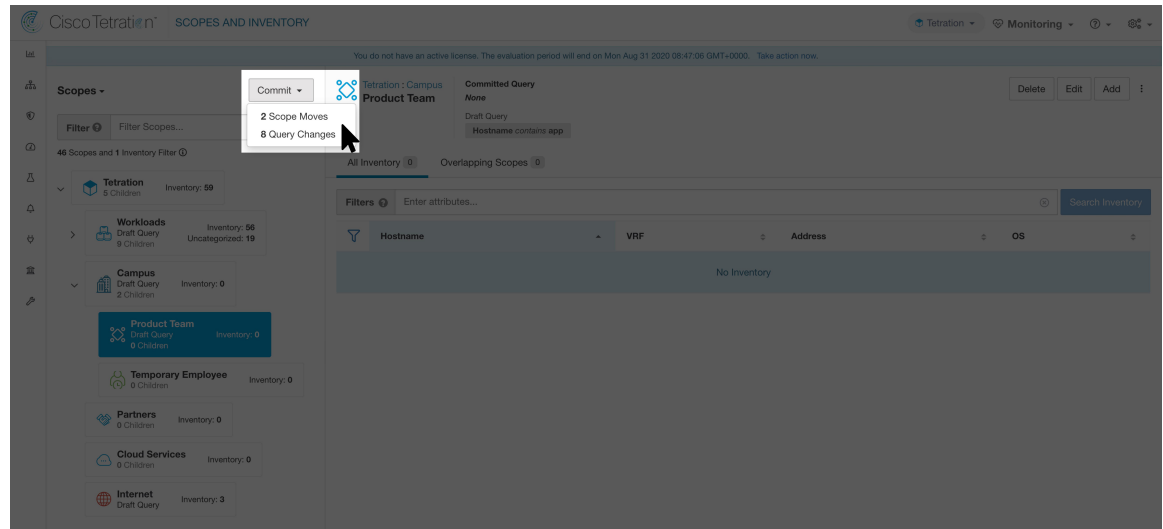




Figure 178: Commit Changes



## Editing the parent of a scope

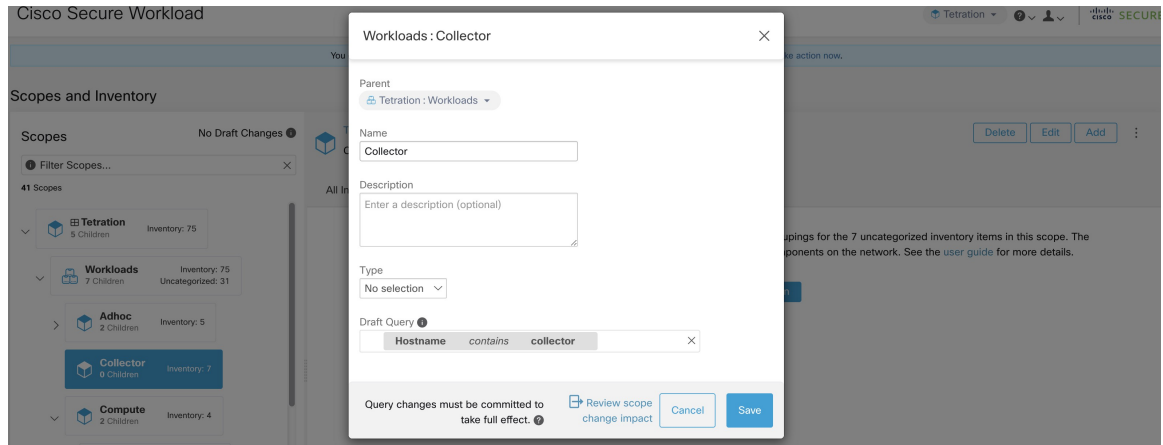
When the parent of a scope is updated, the scope query changes. This change effects the membership of both the parent and child scopes. Similar to editing the scope query, these changes are initially saved as ‘draft changes’ and will not go into effect unless they are committed. The user can validate the impact of this change before committing by clicking on “Review query change impact” on the Edit Scope modal. Once validated, the changes can be committed by clicking “Commit” and accepting the “scope moves” and “query changes”.

To edit the parent of a scope:

### Procedure

- Step 1** Click on the **edit button** on the respective scope to be edited.
- Step 2** Edit the parent for the selected scope.
- Step 3** Compare changes between the old and new Draft Query by clicking the **Review query change impact** link.
- Step 4** Click on **Save**.
- Step 5** Click on “Commit” and accept the ‘scope moves’ and ‘query changes’. The update is processed asynchronously in a background task.
- Step 6** Depending on the number of workloads this change impacts, this can take a minute or more.

**Figure 179: Changing the parent scope from Default scope to Default:ProdHosts**



## Delete a Scope

You can delete a scope only if you have the root scope owner privileges. Site Administrators are the owners of all scopes.

Deleting a scope impacts the application inventory membership of the parent scope (the workloads that are members of the parent scope). After you delete a scope, the status of the parent scope changes to *draft* changes. Commit the changes along with the dependencies such as workspaces, policies, and config intent. For more information, see [Commit Changes](#).

You cannot delete scopes with dependent objects. An error message is displayed under the following circumstances:

- A workspace is defined for the scope.
- An inventory filter is assigned to the scope.
- A policy exists that uses the scope to define its consumers or providers.
- An agent config intent is defined in the scope.
- An interface config intent is defined in the scope.
- A forensics config intent is defined on the scope.

For more information on scope dependencies in the [Review Scope/Filter Change Impact](#) page, click the **Dependencies** tab. Delete redundant objects before you delete the scope.



### Note

- Delete scopes that are without any child scopes.
- To remove a root scope, remove the VRF first from the **Tenants** page.

1. From the navigation pane, click **Organize > Scopes and Inventory**.
2. Click the **Scope** that you want to view the corresponding child scopes.

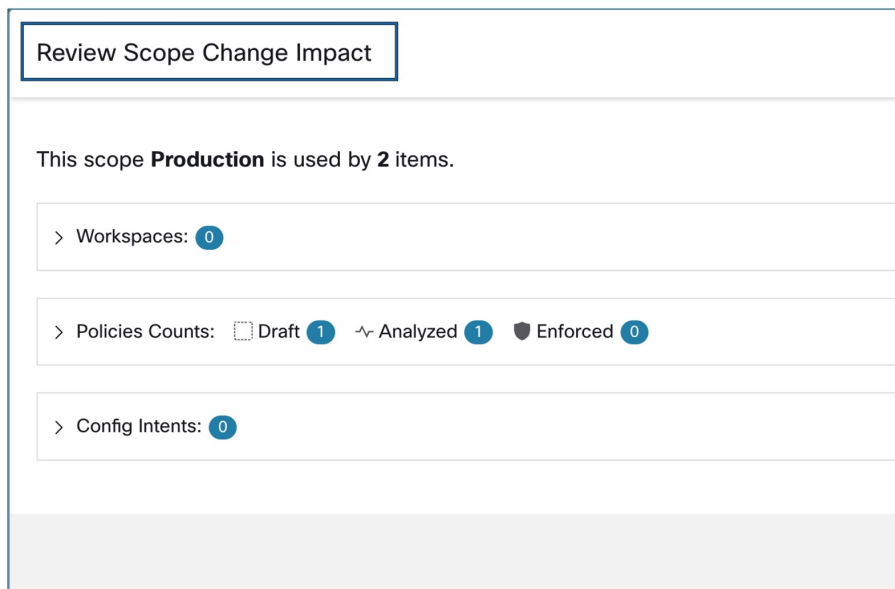
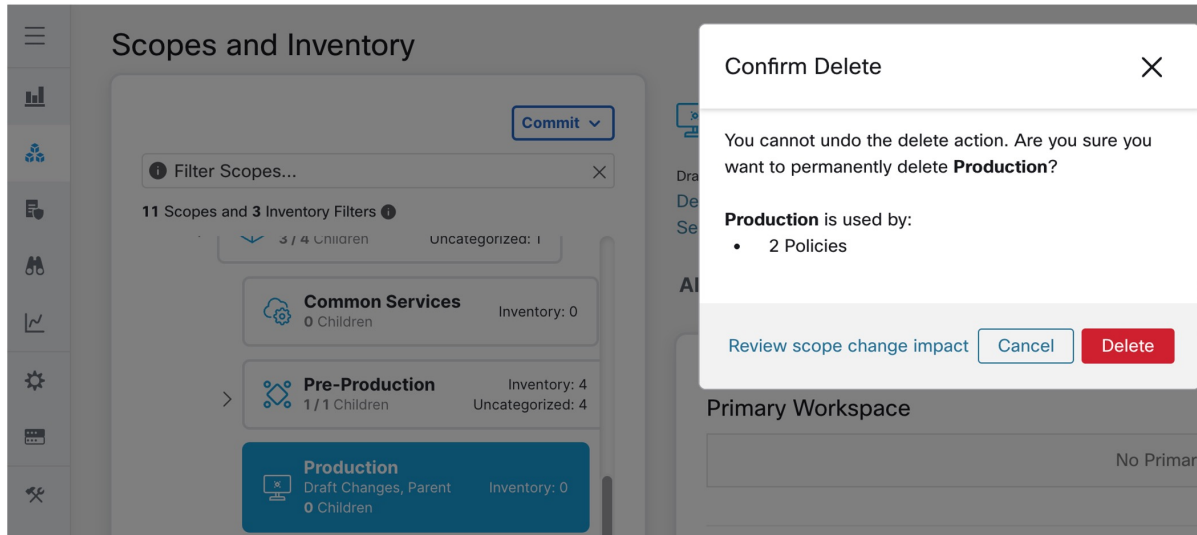
3. Choose the child scope that you want to delete.
4. Click the **Delete** button next to the **Edit** and **Add** buttons.
5. Click the **Review Scope Change Impact** link to review the scope changes before deleting the scope.
6. Click **Back** to close the page.

**Figure 180: Delete a Scope**

The screenshot displays the 'Scopes' management interface. On the left, a tree view shows the hierarchy of scopes: Tetration (5 Children, Inventory: 77), Workloads (7 Children, Uncategorized: 33, Inventory: 77), Adhoc (2 Children, Inventory: 5), AdhocKafka (0 Children, Inventory: 1), AdhocServers (0 Children, Inventory: 4), Collector (0 Children, Inventory: 7), and Compute (2 Children, Inventory: 4). The 'AdhocKafka' scope is highlighted in blue. On the right, the 'AdhocKafka' scope details are shown, including a query 'Hostname contains adhocKafka' and a table of inventory items. The 'Delete' button is highlighted with a red box.

| Hostname      | Address T1 | OS T1 |
|---------------|------------|-------|
| adhockafka1-1 | 1.1.1.55   | linux |

Figure 181: Delete a Scope



- Click **Delete** to delete the scope.

## Reset the Scope Tree

If any of the above configurations exist, you must delete them before you can reset the scope tree. The Reset button is not available until you do so.

To reset the scope tree:

### Before you begin

You can delete the entire scope tree and start over.

Resetting the scope tree deletes all scopes, labels, workspaces, and collection rules. It does not delete any ingested data.

Only a user with the `SCOPE_OWNER` ability on the root scope can reset the scope tree.

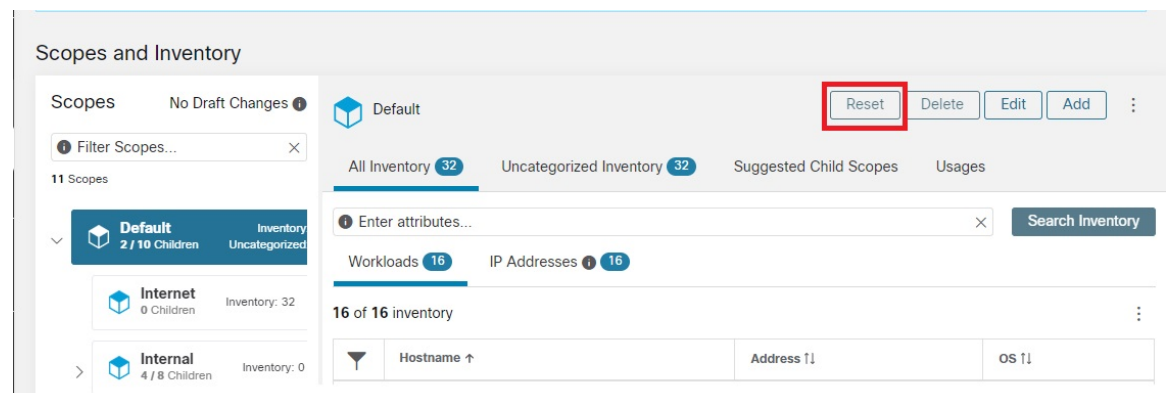
However, you cannot reset the scope tree if any of the following are defined for any scope in the tree:

- Workspaces (except the single workspace created if you created the scope tree using the wizard)
- Inventory filters
- Policies
- Agent Config Intents
- Interface Config Intents
- Forensics Config Intents

## Procedure

- Step 1** From the navigation menu on the left, choose **Organize > Scopes and Inventory**.
- Step 2** Click the scope at the top of the tree.
- Step 3** Click **Reset**.
- Step 4** Confirm your choice.
- Step 5** If necessary, refresh the browser page to continue.

**Figure 182: Reset Scope Tree**



## Commit Changes

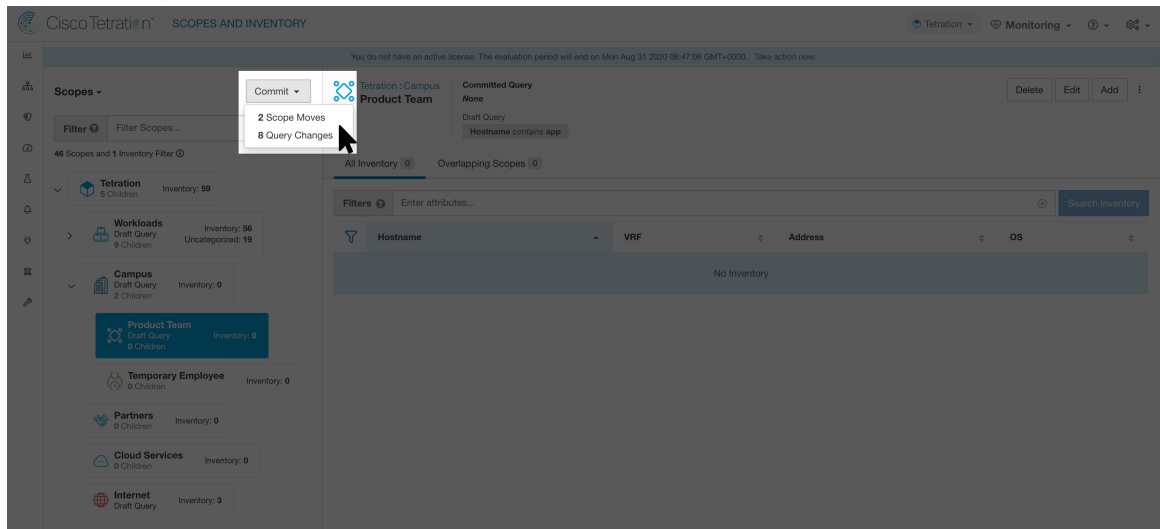
A scope's application inventory query definition is defined by its query and those of its direct children. When this happens the scope is marked as having 'draft changes' and the scope's query, workspaces, and clusters will not be changed until the **Commit Changes** background task is run. When a scope is in draft, the caution triangle is shown by the affected scopes icons, and the 'Commit Changes' button is shown on the Scopes page (top right) and should be clicked to run the **Commit Changes** background task.

Events that can mark a scope as in draft:

- Query update
- The query of any parent is updated.
- Direct child is added.
- Direct child is deleted.
- Direct child's query is updated.

Changing the name of a scope does not change the draft state of the scope.

**Figure 183: Commit Changes**



**Note** The **Commit Changes** task is asynchronous. It usually takes several seconds but large scope trees can take several minutes.

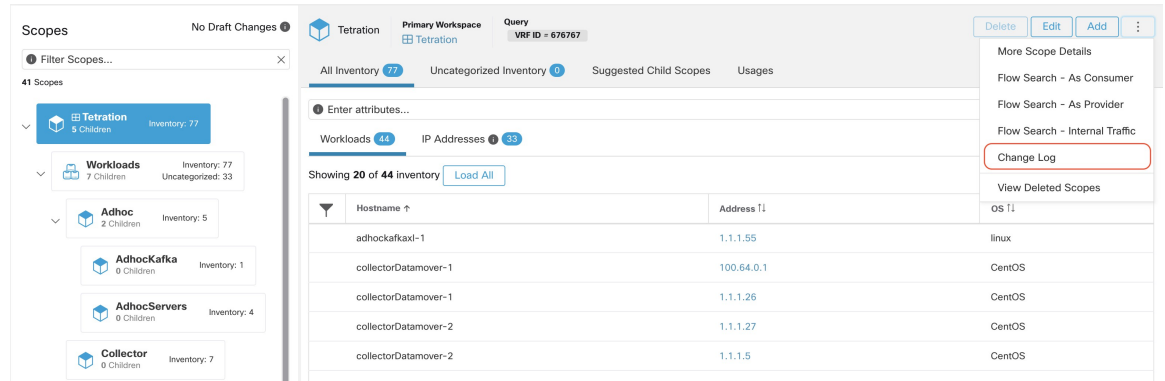


**Note** The scope update task will be completed when the root scope is no longer in draft. Refresh the page to get the latest state.

## Change Log

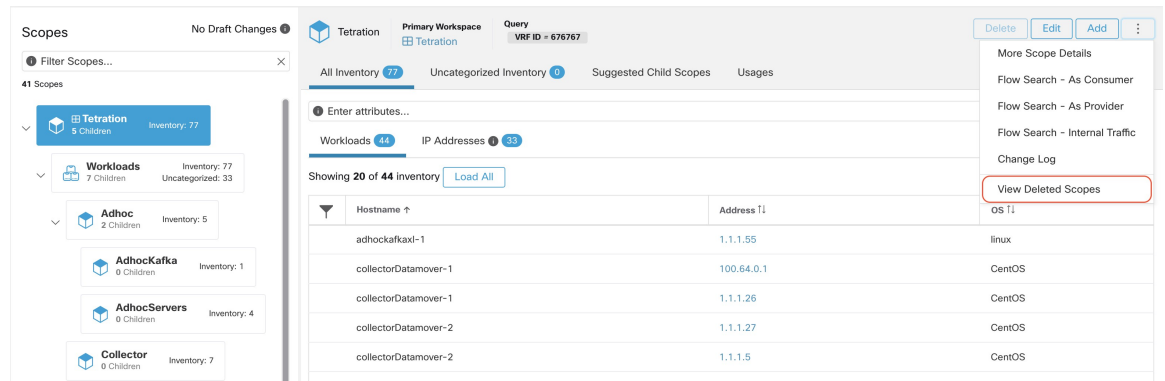
**Site Admins** and users with the `SCOPE_OWNER` ability on the root scope can view the change logs for each scope by clicking change log in the overflow menu in the upper right.

Figure 184: Change Log



These users can also view a list of deleted scopes by clicking on the **View Deleted Scopes** link in the overflow menu in the upper right corner.

Figure 185: View Deleted Scopes



## Creating a New Tenant

Root level scopes map to VRFs that are created under Tenants or through the **Scopes** admin page. This action is only available to **Site Admins** and **Customer Support users**.

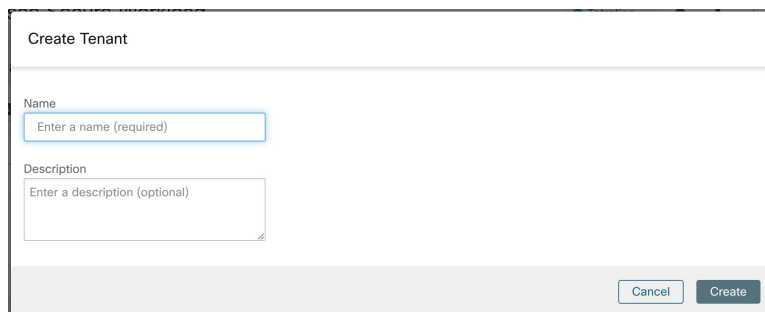
### Procedure

- Step 1** In the navigation bar on the left, click **Platform > Tenants**.
- Step 2** Click the **Create New Tenant** button.
- Step 3** Enter the appropriate values in the following fields:

| Field              | Description                                                            |
|--------------------|------------------------------------------------------------------------|
| <b>Name</b>        | The name to identify the Scope. Must be unique under the parent Scope. |
| <b>Description</b> | An optional description.                                               |

**Step 4** Click the **Create** button.

*Figure 186: Create Tenant*



The screenshot shows a web form titled "Create Tenant". It contains two text input fields. The first field is labeled "Name" and has a placeholder text "Enter a name (required)". The second field is labeled "Description" and has a placeholder text "Enter a description (optional)". At the bottom right of the form, there are two buttons: "Cancel" and "Create".

## Inventory

To work with inventory, click **Organize > Scopes and Inventory** in the left navigation bar.

### Inventory Search

All inventory detected on the network is searchable. To search inventory, use the **Search Inventory** button. Each inventory item is uniquely identifiable by IP and VRF and can be used for performing a search. A service inventory item is not searchable using its IP Address. Use any of the User Labels associated to the service such as `user_orchestrator_system/service_name` for searching a service inventory. After a host has been found, you can view detailed information about the host on the host profile page.

### Inventory Building Blocks

1. Root Scope
  - Root of the scope hierarchy under a given tenant
  - Provides a logical separation for L3 address domains
2. Scope
  - Inventory container defined by dynamic query
  - Foundation for hierarchical policy model
  - Anchor point for policy, RBAC, and filter configuration
3. Filter
  - Flexible construct based on dynamic inventory query
  - Anchor point for intent definition, provided services, and policy definition





**Note** Includes all IP addresses from partners and anything that is communicating in your environment. Whether they have an agent on them or not, you should define what they are through label.

### Label Planning Considerations

1. Source of data
  - Networks - IPAM? Routing tables? Spreadsheet?
  - Hosts - CMDB, Hypervisor, Cloud, App Owners?
2. Accuracy of data
3. How dynamic the data is and how it will be updated.
  - Manual Upload?
  - API Integration?
4. Start with the basics and grow.
  - Use network labels to build high-level scope structure.
  - Use host labels to build more detailed scope structure at app level.

## Searching Inventory

Searching for inventory displays information about specific inventory items.

**Figure 187: Inventory Search**

The screenshot shows the Tetration inventory search interface. On the left, there is a 'Scopes' navigation pane with a search bar and a list of scopes: Tetration (Inventory: 77), Workloads (Inventory: 77, Uncategorized: 33), Campus (Inventory: 0), Partners (Inventory: 0), Cloud Services (Inventory: 0), and Internet (Inventory: 0). The main area shows 'All Inventory' (77) and 'Uncategorized Inventory' (0). A search bar is present with the text 'Enter attributes...' and a 'Search Inventory' button. Below the search bar, there are filters for 'Workloads' (44) and 'IP Addresses' (633). A 'Showing 20 of 44 inventory' indicator and a 'Load All' button are also visible. The table below shows the following data:

| Hostname             | Address T1 | OS T1  |
|----------------------|------------|--------|
| adhoc-1              | 1.1.1.47   | linux  |
| adhoc-2              | 1.1.1.48   | linux  |
| appServer-2          | 1.1.1.6    | linux  |
| collectorDatamover-1 | 100.64.0.1 | CentOS |
| collectorDatamover-2 | 1.1.1.27   | CentOS |

### Procedure

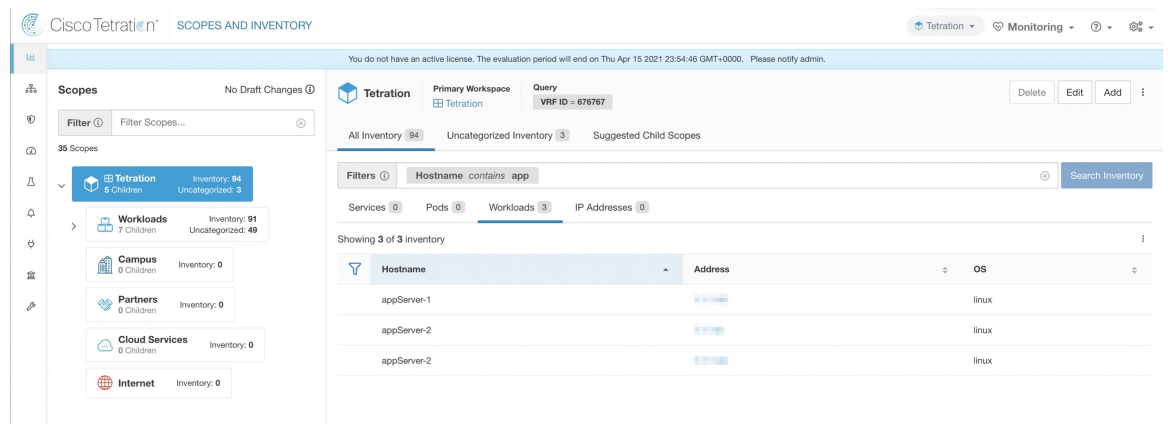
- Step 1** From the navigation pane, choose **Organize > Scopes and Inventory**.
- Step 2** Enter the attributes in the **Filters** field for the inventory items that you are looking for. The attributes include the following:

| Attributes                                                | Description                                                                                                                     |
|-----------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------|
| <b>Hostname</b>                                           | Enter a full or partial hostname.                                                                                               |
| <b>VRF Name</b>                                           | Enter a VRF name.                                                                                                               |
| <b>VRF ID</b>                                             | Enter a VRF ID (numeric).                                                                                                       |
| <b>Address</b>                                            | Enter a valid IP address or subnet (IPv4 or IPv6).                                                                              |
| <b>Address Type</b>                                       | Enter either IPv4 or IPv6.                                                                                                      |
| <b>OS</b>                                                 | Enter an OS name (e.g. CentOS).                                                                                                 |
| <b>OS Version</b>                                         | Enter an OS version (e.g. 6.5).                                                                                                 |
| <b>Interface Name</b>                                     | Enter an interface name (e.g. eth0).                                                                                            |
| <b>MAC</b>                                                | Enter a MAC address.                                                                                                            |
| <b>In Collection Rules?</b>                               | Enter true or false.                                                                                                            |
| <b>Process Command Line</b>                               | Enter the substring of a command that is running on host (Note: this facet cannot be saved as part of inventory filter).        |
| <b>Process Binary Hash</b>                                | Enter the process hash of a command that is running on the host (Note: this facet cannot be saved as part of inventory filter). |
| <b>Package Info</b>                                       | Enter the package name optionally followed by a package version (prefixed by #).                                                |
| <b>Package CVE</b>                                        | Enter part of or a complete CVE ID.                                                                                             |
| <b>CVE Score v2</b>                                       | Enter a CVSSv2 (Common Vulnerability Scoring System) score (numeric).                                                           |
| <b>CVE Score v3</b>                                       | Enter a CVSSv3 (Common Vulnerability Scoring System) score (numeric).                                                           |
| <b>Cisco Security Risk Score</b>                          | Enter a Cisco Security Risk Score (numeric).                                                                                    |
| <b>Severity (Cisco Security Risk Score)</b>               | Enter a Cisco Security Risk Score severity: High, Medium, or Low.                                                               |
| <b>Active Internet Breach (Cisco Security Risk Score)</b> | Indicates whether CVE is part of Active Internet Breach activity across organisations. Enter true or false.                     |
| <b>Easily Exploitable (Cisco Security Risk Score)</b>     | Indicates whether CVE has known exploit kits. Enter true or false.                                                              |
| <b>Fix Available (Cisco Security Risk Score)</b>          | Indicates whether a fix is available for the CVE. Enter true or false.                                                          |

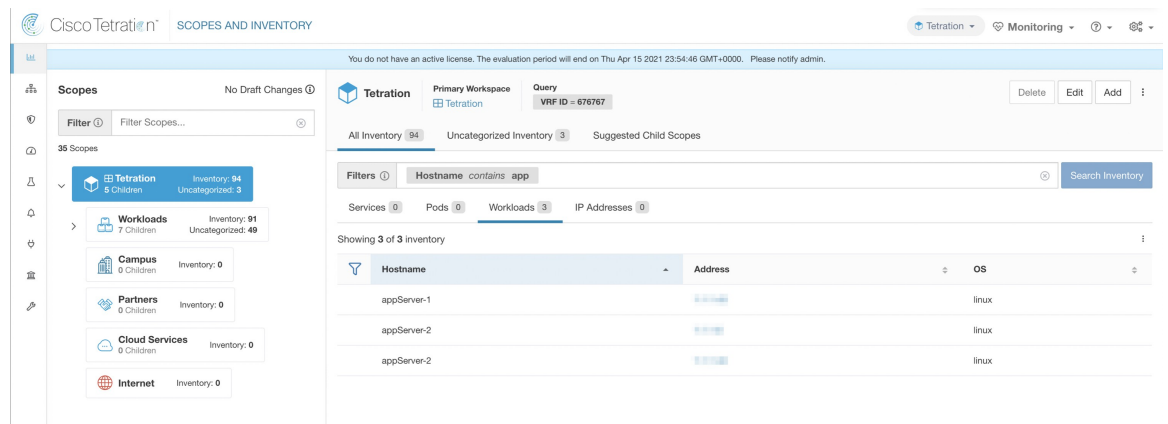
| Attributes                                               | Description                                                                                                                         |
|----------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------|
| <b>Malware Exploitable (Cisco Security Risk Score)</b>   | Indicates whether CVE can be actively exploited with malware including trojans, worms, ransomware, and others. Enter true or false. |
| <b>Popular Targets (Cisco Security Risk Score)</b>       | Indicates whether CVE is detected in high volume by other Cisco Vulnerability Management clients. Enter true or false.              |
| <b>Predicted Exploitable (Cisco Security Risk Score)</b> | Indicates whether CVE is expected to have an Active Internet Breach in the future. Enter true or false.                             |
| <b>User Labels</b>                                       | Attributes prefixed with come from user labels.                                                                                     |

**Step 3** Click **Search Inventory**. The results are displayed below the **Filters** field that is grouped into four tabs. Each tab displays a table with the relevant columns. Additional columns can be displayed by clicking on the funnel icon in the table header. If any user labels are available, they will be prefixed with and can be toggled here.

**Figure 188: Inventory Search Results**



**Figure 189: Inventory Search Results**



The search results are grouped into four tabs:

| Tab          | Description                                                                                                                                                                                                                                                                                                                                       |
|--------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Services     | Lists the Kubernetes services and load balancers discovered through External Orchestrators. This tab is hidden unless a related external orchestrator is configured.                                                                                                                                                                              |
| Pods         | Lists the Kubernetes pods. This tab is hidden unless a related external orchestrator is configured.                                                                                                                                                                                                                                               |
| Workloads    | Lists the inventory items reported by Secure Workload agents.                                                                                                                                                                                                                                                                                     |
| IP Addresses | <p>Lists the inventory items discovered through:</p> <ul style="list-style-type: none"> <li>• inventory upload</li> <li>• learning from flows</li> <li>• manually uploaded labels</li> <li>• labels ingested through connectors and external orchestrators</li> </ul> <p>Additionally, the lists from subnets reported from the same sources.</p> |

**Note** By default, the catch all subnets for IPv4 and IPv6 addresses display in each tenant.

There is also a mention of the inventory count next to each tab. The immediately available information in a search includes hostname, IP Addresses with subnets, OS, OS Version, Service Name and Pod Name. The list of displayed columns can be toggled by clicking the funnel icon in the table header. Search results are restricted to the currently selected scope shown in the scope directory. More information can be seen on the respective profile page by clicking on an item in the search results.

More details about each host is displayed on the **Workload Profile**, which is accessible by clicking on the IP address field of a search result row. See the [Workload Profile](#) for more information.

To create Inventory Filters via the sidebar: Choose **Organize > Inventory Filters** from the top-level menu. Click the **Create Filter** button. A modal dialog appears where you can name your saved filter.

## Suggest Child Scopes

Suggest Child Scopes is a tool that uses machine learning algorithms (such as community detection in networks) to discover groupings that could serve as scopes. This tool is helpful when building a scope hierarchy, and facilitates the process of defining more granular child scopes for a given scope. Candidate child scopes are shown as suggestions that can then be selected and added.

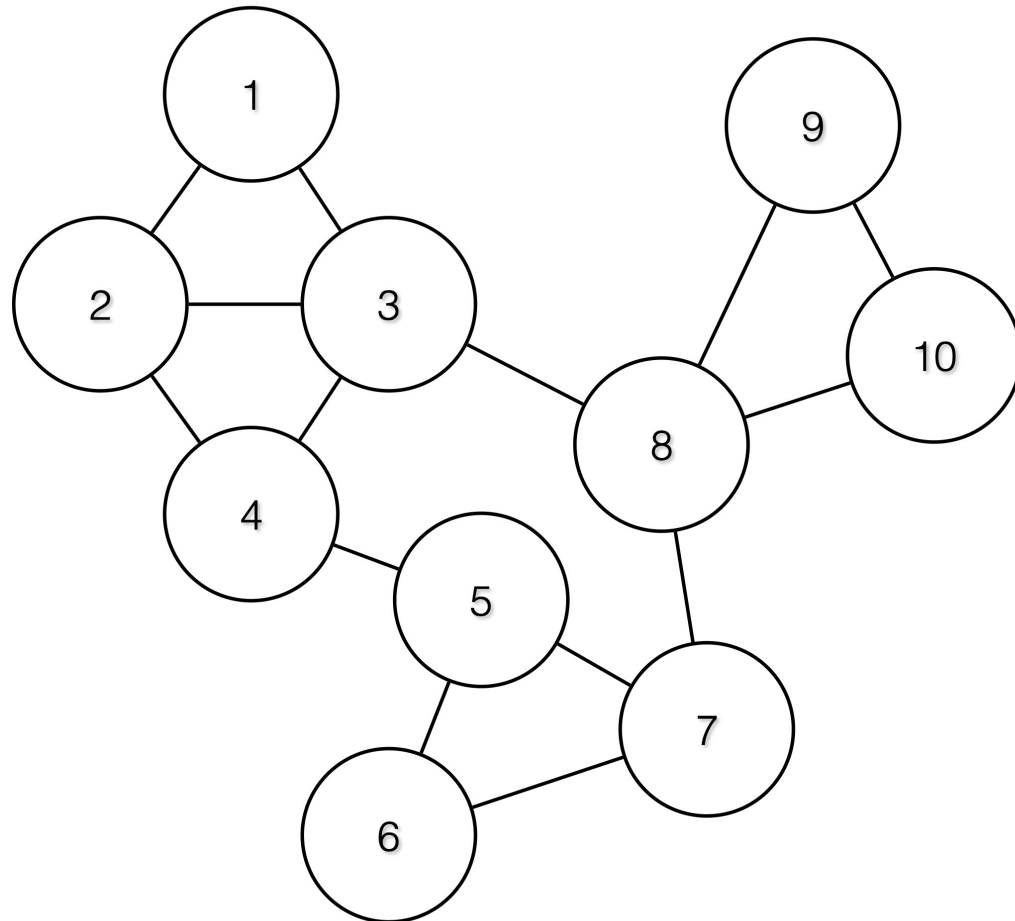
**Description of Algorithms:** A graph based on the communications among the unclaimed members of the parent scope is first created (note: unclaimed members are those that do not belong to any child scope of the parent), and the graph is preprocessed, for example the algorithms attempt to identify endpoints that communicate with sufficiently high proportion of other endpoints in the graph. Such a group of endpoints, if found, is displayed to the user as a candidate **common services** grouping. The rest of the graph is processed

to detect groups that behave as **communities**, meaning roughly that the endpoints disproportionately communicate with one another more often (or on more provider ports) than to endpoints outside the group. Each such grouping may correspond to an application or a department within the organization. Such a partitioning can also lead to sparser policies among scopes.

**Example:**

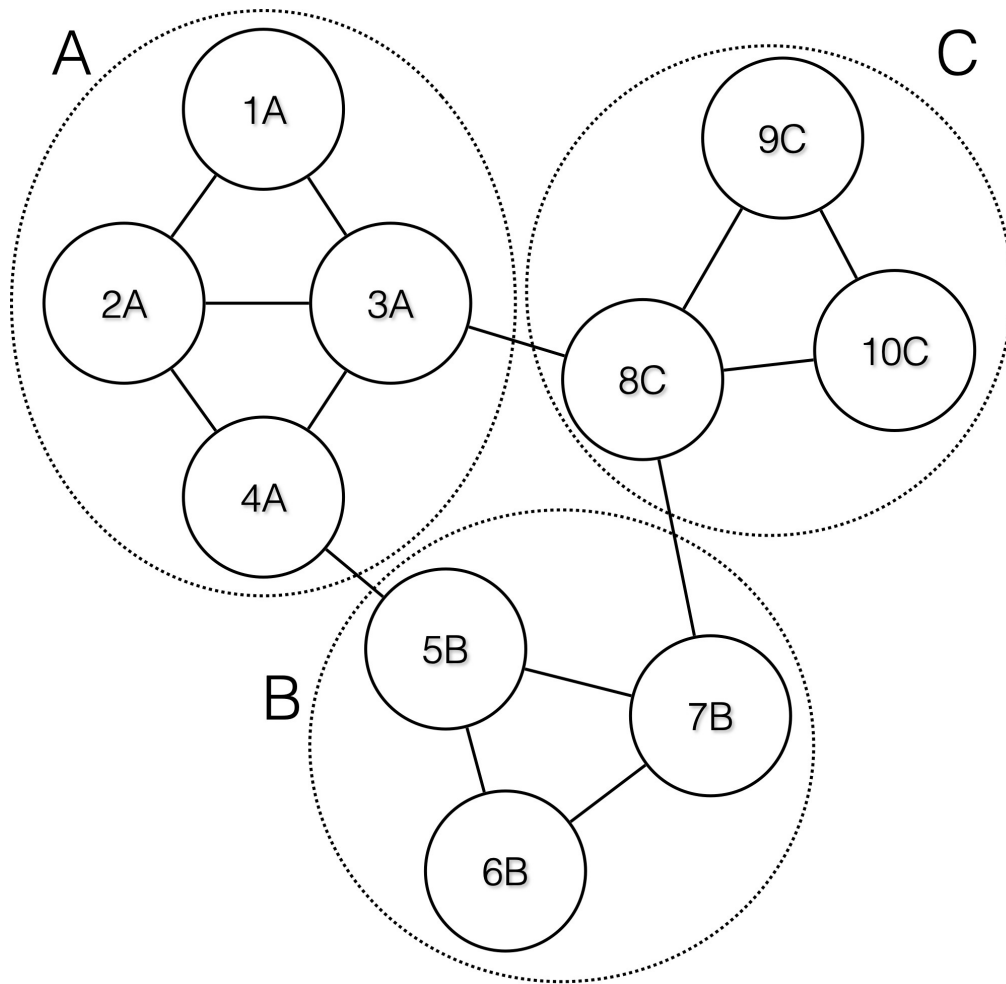
Let 1 through 10 be individual endpoint IPs. Assume the input (communications) graph is as follows:

*Figure 190: Input graph*



Then the endpoints 1 - 4, 5 - 7 and 8 - 10 will be grouped together because they have relatively high degree of communication (number of edges) among one another, and relatively low communications to other endpoints.

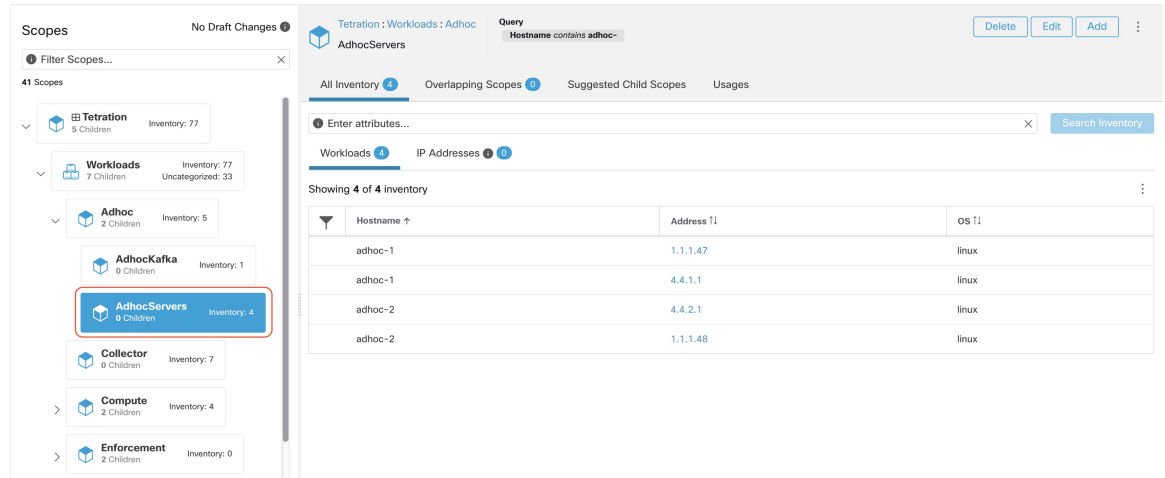
Figure 191: Output Groups



## Steps to perform scope suggestion

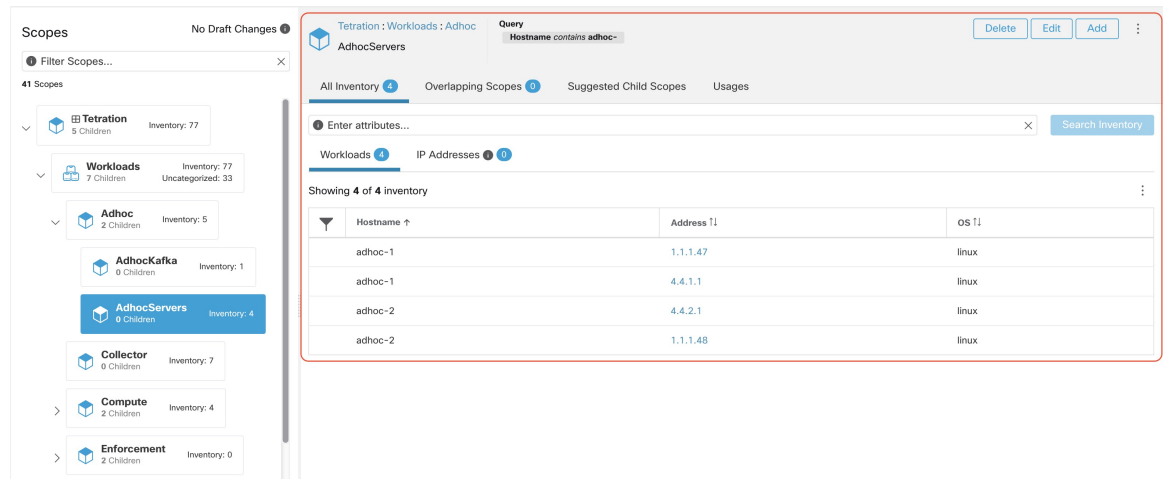
To invoke scope suggestion for a desired scope user should locate on the scopes page and select it.

Figure 192: Selecting a Scope



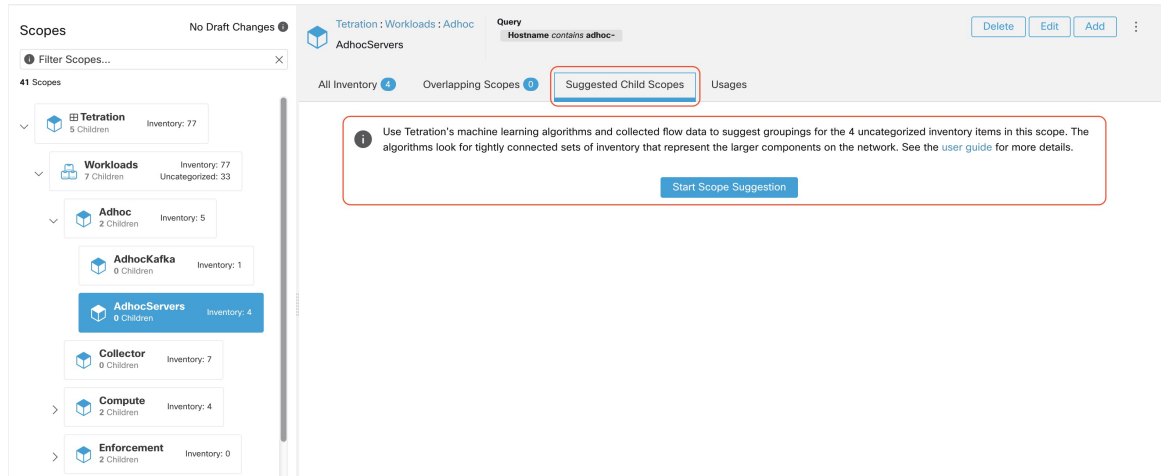
In the window, user can browse the inventory, *uncategorized inventory items*, i.e. those items that belong to the current selected scope and that do not belong to any of the current selected scope's child scopes. Clicking on the **uncategorized inventory items** allows one to view this list.

Figure 193: Scope Window



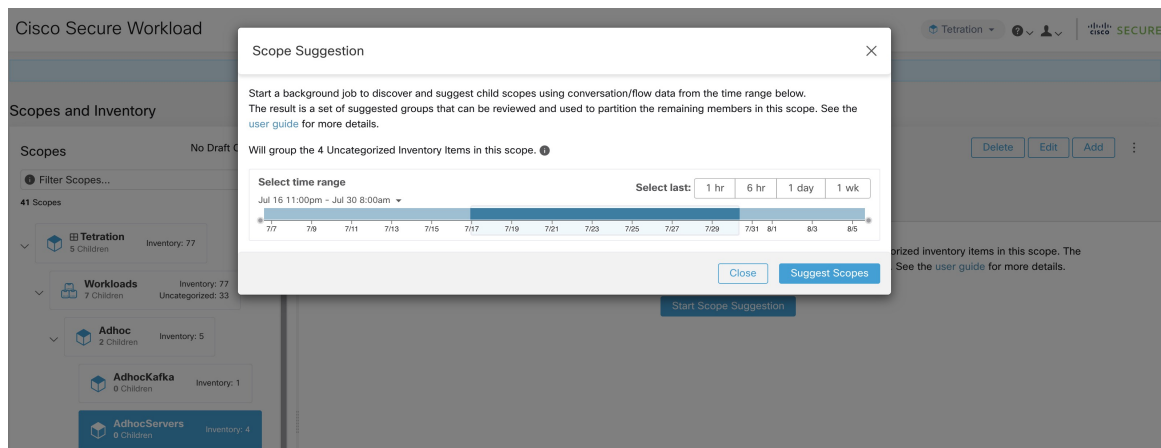
After selecting the scope user can click on **Suggest Child Scopes**, and click on **Start Scope Suggestion** (or click on Rerun, in case this is not the first time). Note that the input for a scope suggestion run will be the uncategorized inventory items.

Figure 194: Child Scopes



User can set the date range as input for scope suggestion and click on **Suggest Scopes**. A scope suggestion run is often fast under medium overall load, and takes only a few minutes for processing ten to thousands of endpoints, with tens of thousands of conversations.

Figure 195: Scope Suggestion Data Range Selector



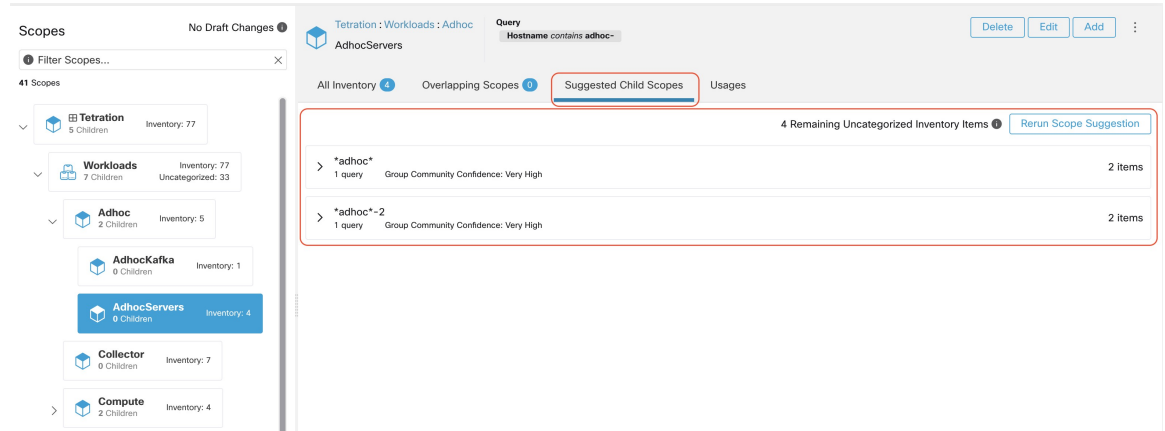
The output is shown to the user as a list of candidates, currently up to 20 groups (shown), each accompanied with information such as group confidence (quality), a candidate scope name, and queries. Each discovered group has an associated **Group Community Confidence**, the possible values being: **Very High**, **High**, **Medium** and **Low**. This is a measure of the **Community** property of the group: the higher the confidence, the higher the community property of the given group of endpoints (many edges inside the group, relatively few edges to outside). Currently, the subset of groups picked to be shown are selected based on the Group Community Confidence. The groups discovered can currently fall under one of these four group types:

- **Generic Group**: Any group discovered via machine learning based on the community property. Note that any group that is not explicitly designated with the special types below is a generic group.
- **Common Service**: This group consists of endpoints that communicate with much of the input inventory. These endpoints could be running some kind of shared service(s).



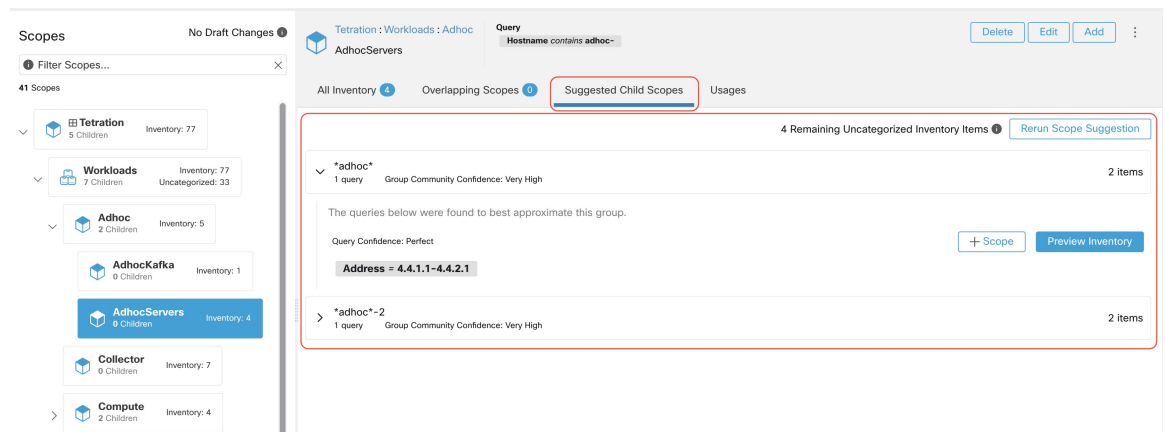
- **Common Service Clients:** This group consists of endpoints that only communicate with the **Common Service** group.
- **Ungrouped:** This group consists of endpoints that cannot be grouped since they don't have sufficient communications.

Figure 196: Scope Suggestion Output



The user can click on a discovered group to view the list of queries generated for the selected group. The user can preview the inventory covered by the query which will closely define the discovered group. The queries consist of IP-ranges, subnets, host names and user uploaded labels. There is a confidence measure associated with each group called **Query confidence** which can have one of the following range of values **Perfect, Very High, High, Medium** and **Low**. For query generation, first the groups are discovered via graph processing and machine learning, then the queries are generated for each group. **Query Confidence** is a measure of how well the query can cover the endpoints. A query confidence of **Perfect** indicates that the query exactly covers the suggested (discovered) group. On the other end of the spectrum, a **Low** query confidence indicates that the query significantly misses out on exactly capturing the suggested group, which means that the query covers many **Extra IPs** (not part of the discovered group) and/or has many **Missing IPs** (not covered by the query).

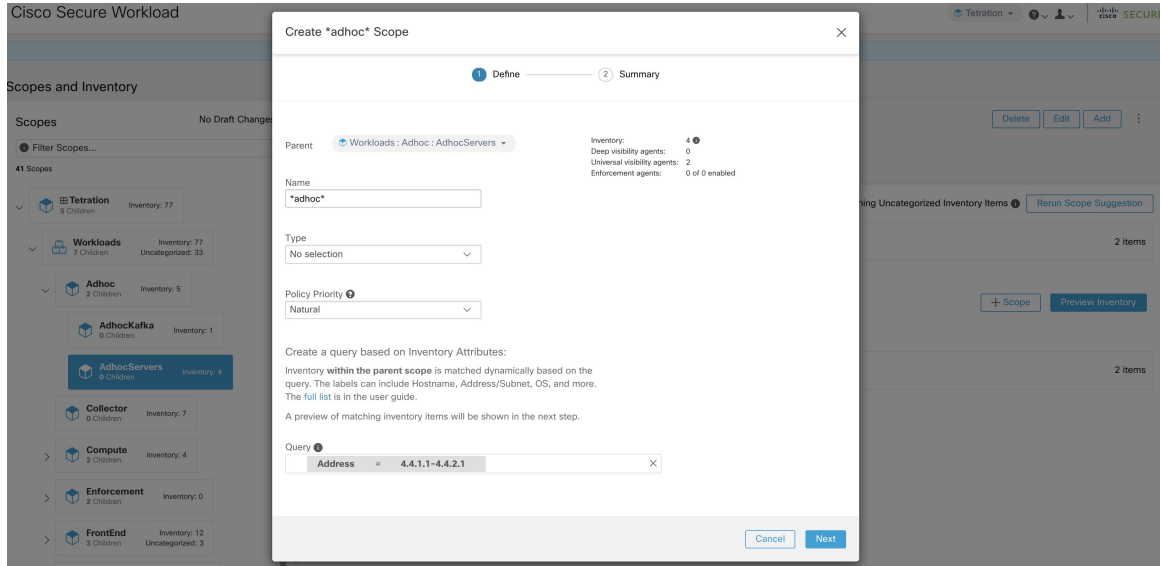
Figure 197: Scope Suggestion Output Queries



The user can click on **+ Scope** button which will take the user to an edit window where the user can edit the group name and group query. The user can examine a query, the IPs that it matches, and decide whether some

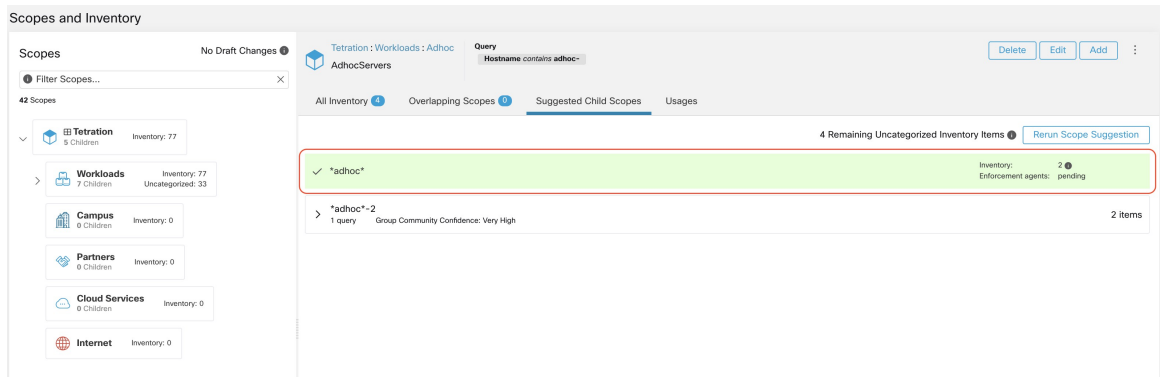
IPs need to be added or removed by adjusting the query. Once satisfied, the user can then click on **Next**, to review and convert the group to a scope on the draft view canvas.

**Figure 198: Scope Suggestion Edit Window**



After the user has converted a suggested group to a scope, the group slot turns green and the **Uncategorized Inventory Items** count decreases.

**Figure 199: Example of scope suggestion output after converting one suggested group to a scope**



The user can repeat the process of scope creation from the remaining list of groups. The recommended workflow is to create one or more scopes and then re-run **scope suggestion**. A zero count for **Uncategorized Inventory Items** indicates that there is no inventory left to be further scoped (for the currently selected parent scope).

Figure 200: Scope Suggestion Output from Multiple Scope Creations

The screenshot shows the 'Scopes and Inventory' interface. On the left, a tree view shows the hierarchy: Tetration (5 Children, Inventory: 77) > Workloads (7 Children, Inventory: 77, Uncategorized: 33) > Adhoc (2 Children, Inventory: 5). The 'Adhoc' scope is expanded to show 'AdhocKafka' (0 Children, Inventory: 1) and 'AdhocServers' (2 Children, Inventory: 4, Uncategorized: 2). The 'AdhocServers' scope is further expanded to show '\*adhoc\*' (0 Children, Inventory: 2) and '\*adhoc\*-2' (0 Children, Inventory: 2). On the right, the 'Suggested Child Scopes' tab is active, showing a message: 'It is a best practice to rerun grouping after creating a few new scopes. This allows the machine learning algorithm to better suggest groups for the remaining items.' Below this, two suggested scopes are listed: '\*adhoc\*' (Inventory: 2, Enforcement agents: 0 of 0 enabled) and '\*adhoc\*-2' (Inventory: 2, Enforcement agents: pending). A 'Rerun Scope Suggestion' button is visible.

After the scope creation process is done (the uncategorized count is 0), user can repeat this process on the newly created child scopes in order to generate a deeper scope tree as desired.

Figure 201: Scopes List after Initial Scope Suggestion and Creation

The screenshot shows the 'Scopes and Inventory' interface after the initial scope suggestion and creation. The left tree view now includes 'AdhocServers' (2 Children, Inventory: 4, Uncategorized: 2) expanded to show '\*adhoc\*' (0 Children, Inventory: 2) and '\*adhoc\*-2' (0 Children, Inventory: 2). The right pane shows the 'Suggested Child Scopes' tab with the same message and two suggested scopes: '\*adhoc\*' (Inventory: 2, Enforcement agents: 0 of 0 enabled) and '\*adhoc\*-2' (Inventory: 2, Enforcement agents: pending). A 'Rerun Scope Suggestion' button is visible.



**Note** There is also a possibility that the uncategorized items in a scope do not partition well (e.g., do not form communities). In that case, the algorithm may return no groupings (an empty result).

## Filters

Filters are saved inventory searches used in defining policies, configuration intents, and so on. Avoid any filter that is associated with a scope, which defines the filter's ownership scope.

To view existing filters, click **Organize > Inventory Filters** on the navigation bar. You can also view inventory filters specific to any workspace for any scope.

The list of filters are restricted based on the root of the currently selected scope.

The filters also display the number of members, number of policies it is involved in, the sum of draft analysed and enforced policies.

Figure 202: Inventory filters

| Name       | Query                                                                                  | Ownership Scope | Restricted? | Members | Policies | Configs | Created At           | Actions |
|------------|----------------------------------------------------------------------------------------|-----------------|-------------|---------|----------|---------|----------------------|---------|
| Everything | Address = 0.0.0.0/0 or Address = ::/0                                                  | All Root Scopes | No          |         |          |         | AUG 30, 2023 6:45 AM |         |
| Test ana   | CVE Score v2 = 233 and CVE Score v3 = 2332 or<br>CVE Score v2 = 234423<br>show more... | Default         | No          |         |          |         | AUG 31, 12:29 PM     |         |
| filter-1   | Address = 10.0.0.1                                                                     | Default         | No          |         |          |         | SEP 1, 11:14 PM      |         |
| filter-2   | Address = 10.0.0.2                                                                     | Default         | No          |         |          |         | SEP 1, 11:14 PM      |         |

You can review inventory membership changes with respect to the selected parent scope by visiting the [Review Scope/Filter Change Impact](#) window.

## Create an Inventory Filter

Create inventory filters to:

- Create or discover policies specific to subsets of workloads within a scope.

For example, create a group of API servers within the scope, the servers must be accessible through the API interface. Create policies to allow only the permissible traffic, but block access to all other workloads for that application.

- Create policies for workloads that exist across many scopes.

For example, to create a policy that applies to all workloads on the network running a particular operating system, create an inventory filter that spans across multiple or all scopes.



**Tip** To convert an existing cluster to an inventory filter, see [Convert a Cluster to an Inventory Filter, on page 493](#).

### Procedure

- Step 1** Navigate to one of the following locations:
  - Choose **Organize > Inventory Filters**.
  - Navigate to any workspace in a scope for which you want to create an inventory filter and click **Manage Policies > Filters > Inventory Filters**.
- Step 2** Click **Create Filter** or **Add Inventory Filter**.
- Step 3** Add a name, description, and query that includes all, and only those workloads to include in the filter.
- Step 4** Click **Show Advanced Options**.
- Step 5** Specify the scope for the filter.
  - To modify the filter, you must have write access to the specified scope or any of its ancestors.

- (Depending on other settings in this procedure) The workloads included in the filter.

**Step 6** Configure options:

| To                                                                                                                                                                                                  | Do This                                                                                                                                                                                                                                                                                   |
|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Include workloads that meet the filter query criteria, whether they are members of the scope that is specified in this filter.                                                                      | Deselect <b>Restrict Query to Ownership Scope</b>                                                                                                                                                                                                                                         |
| Include only workloads that are members of the scope that is specified in this filter.                                                                                                              | Choose <b>Restrict query to ownership scope.</b>                                                                                                                                                                                                                                          |
| Allow automatic policy discovery to suggest policies specific to the set of workloads defined by this filter.<br><br>These workloads must be a subset of the scope that is specified in the filter. | Select <b>Restrict Query to Ownership Scope</b> and <b>Provides a Service External of its Scope.</b><br><br>To use this filter, you must configure external dependencies.<br><br>For more information, see <a href="#">Fine-Tune External Dependencies for a Workspace</a> , on page 449. |

**Step 7** Click **Next**.

**Step 8** Review the details and click **Create**.

## Review Filter Change Impact

Prior to modifying the filter, note that changes to the filter will modify the membership and impact the existing queries, inventories, and dependencies such as filters, scopes, policies, and enforced behavior that are based on that label.

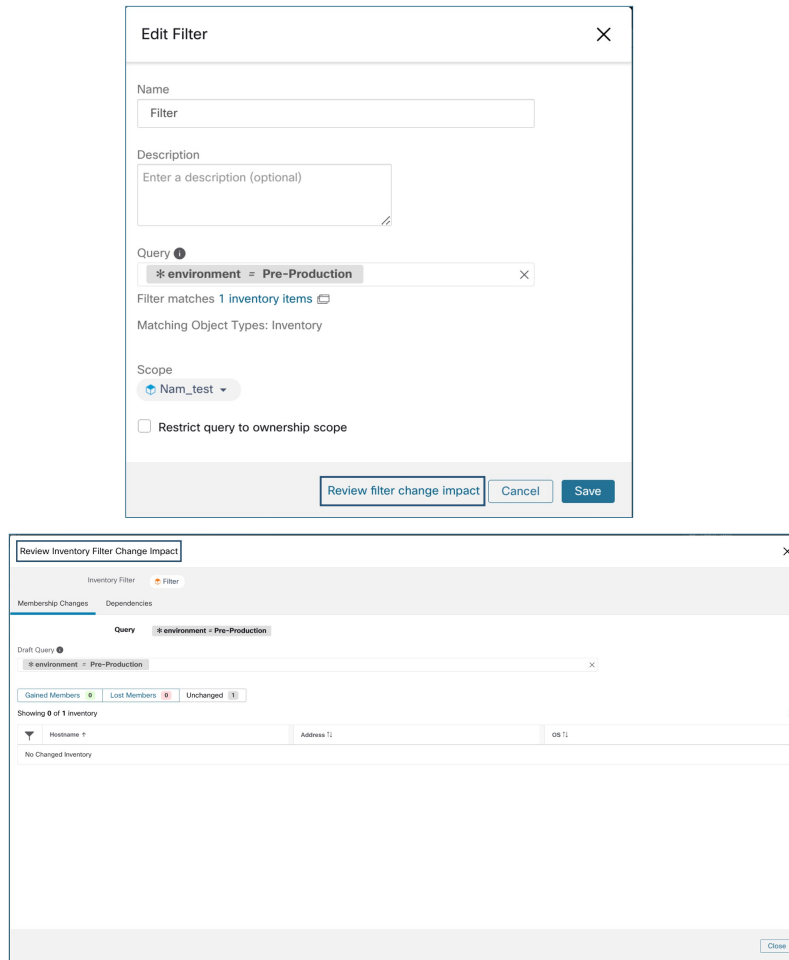
### Procedure

**Step 1** On the **Inventory Filter** page, under the **Actions** tab, click the **Edit** (pencil) icon.

**Step 2** On the **Edit Filter** page, before you make any changes, click **Review filter change impact** to review the inventory filters.

**Step 3** On the **Review Inventory Filter Change Impact** page, review the filters for policies and configuration intent under the **Membership Changes** and **Dependencies** tabs.

Figure 203: Edit a Filter



**Step 4** After you verify the inventory filters, to return to the **Edit Filter** page, click **Close**.

## Create a Domain Filter

Use a domain filter to group domains and identify flows where a consumer or provider domain name matches the filter that is defined in your environment.

In conversation mode, only certain types of proxies are supported for domain enforcement, such as HTTP proxy and TCP. In case of TCP, when a domain is blocked with intent, the first packet may pass through; however, the connection is blocked even before a handshake is complete.

### Rules for Domain Filters

- You can enter only two domain names in the **Query** field such as mail.cisco.com or domain name=\*cisco.com. Domain names, such as .com, .org, .net are not supported.
- Each label in the domain name can have only letters, numbers, or a hyphen.

- Use the wildcard \* in the domain name and only for the first label, for example .amazon.com, but do not use aws.com. Also, do not combine wildcards with any other characters using regex, for example, do not use aws\*.com.
- A wildcard matches any number of labels (subdomains), for example, .yahoo.com matches finance.yahoo.com, web.finance.yahoo.com and all its subdomains. However, it does not match yahoo.com.
- www prefix is treated as a subdomain, and is therefore not treated as the domain itself, for example, google.com and www.google.com are separate domains.
- Do not limit the scope of inventory filters. If an object matches the filter, apply it to the entire tenant by entering DOMAIN.

## Procedure

---

- Step 1** Navigate to one of these locations:
- Choose **Organize > Inventory Filters**.
  - Navigate to a workspace in the scope to create an inventory filter, click **Manage Policies > Filters > Inventory Filters**.
- Step 2** Click **Create Filter** or **Add Inventory Filter** to display the Inventory Filter page.
- Step 3** Check the **Domain Filter** check box.
- Step 4** Enter a name and query for the domain filter, and click **Next**
- Step 5** Review the details and click **Create** to create a domain filter.
- 

For every new inventory filter, create a corresponding new object type that defines the kind of objects for filter matches. The possible values are:

- **INVENTORY** encompasses workloads, services, pods, and IP addresses.
- **DOMAIN** refers to domains. Domain name is the only facet available to match domains; all other facets match only the **INVENTORY** type.

You can create a heterogenous filter using the domain name and another facet with an **OR** operator, for example domain name=\* .google .com OR hostname that contains mach. However, it is not possible to use **AND** to combine such facets using the **AND** operator, for example domain name=\* .google .com AND hostname that contains mach.

## Restrict to Ownership Scope

Check the **Restrict to Ownership Scope?** checkbox to determine if the scope impacts the inventory that matches with a filter. For example, in the following structure:

1. Tenant with query

```
VRF ID = 3
```

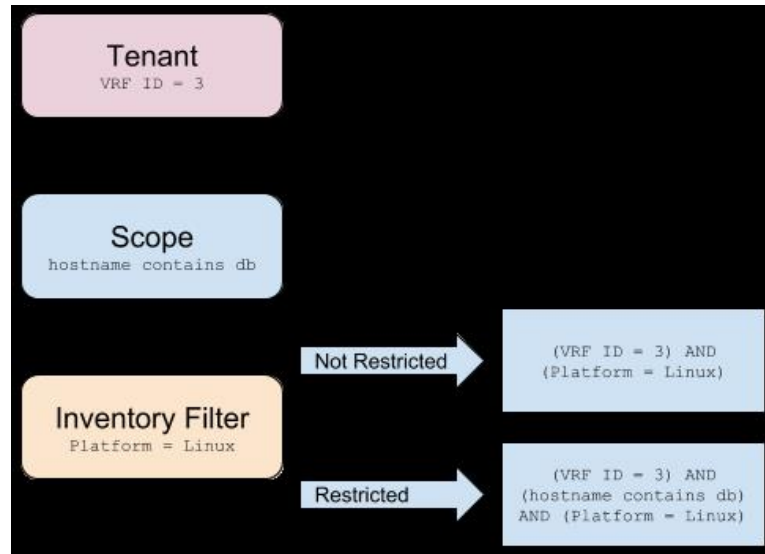
2. Scope within the tenant with the query

```
hostname contains db
```

- Inventory filter with the following query that is attached to a scope.

```
Platform = Linux
```

**Figure 204: Tenant, Scope, and Inventory Filter Structure**



- If you do not choose **Restrict to Ownership Scope**, the filter will match all hosts within the tenant that also matches the filter. Enter the following query:

```
(VRF ID = 3) AND (Platform = Linux)
```

- If you choose **Restrict to Ownership Scope**, the filter will match only those hosts within the tenant and the scope that also matches the filter. Enter the following query:

```
(VRF ID = 3) AND (hostname contains db) AND (Platform = Linux)
```

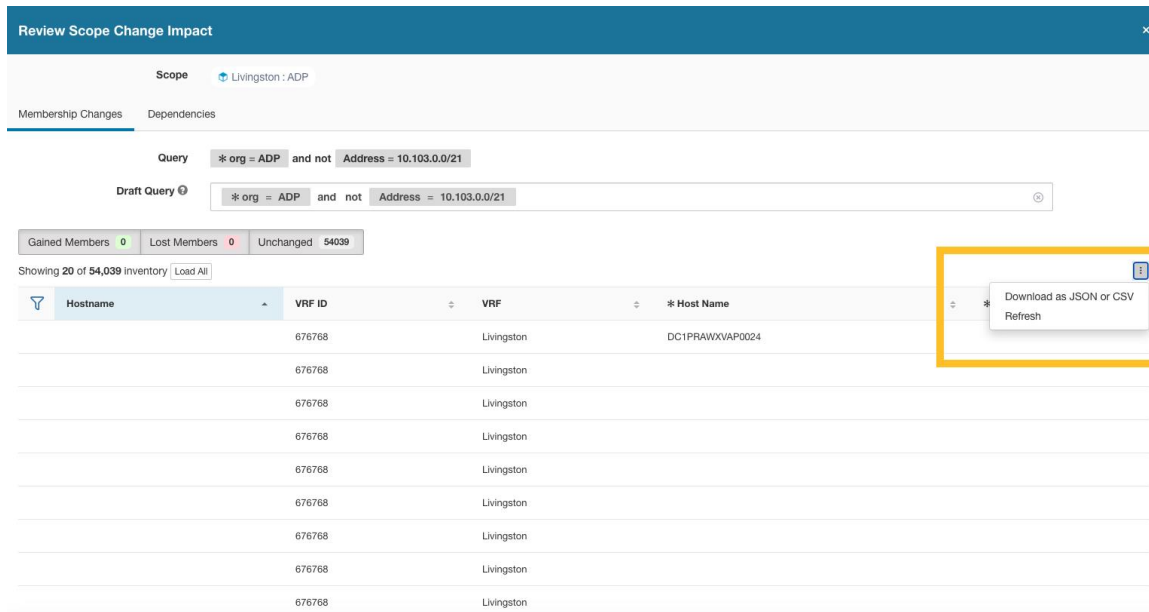
## Review Scope/Filter Change Impact

Updating a scope query can impact the scope's inventory membership after it gets committed. Likewise filter query change, which gets saved directly, can also impact the scope inventory memberships. You can identify membership changes between the new and old queries by following the **Review query change impact** link on either Scope or Filter Edit modals. In addition, knowing the scope or filter dependencies can be helpful for impact analysis and removing all necessary objects preventing Scope deletion. Visit the **Dependencies** tab as well, to traverse the Scope Dependencies tree for further information.





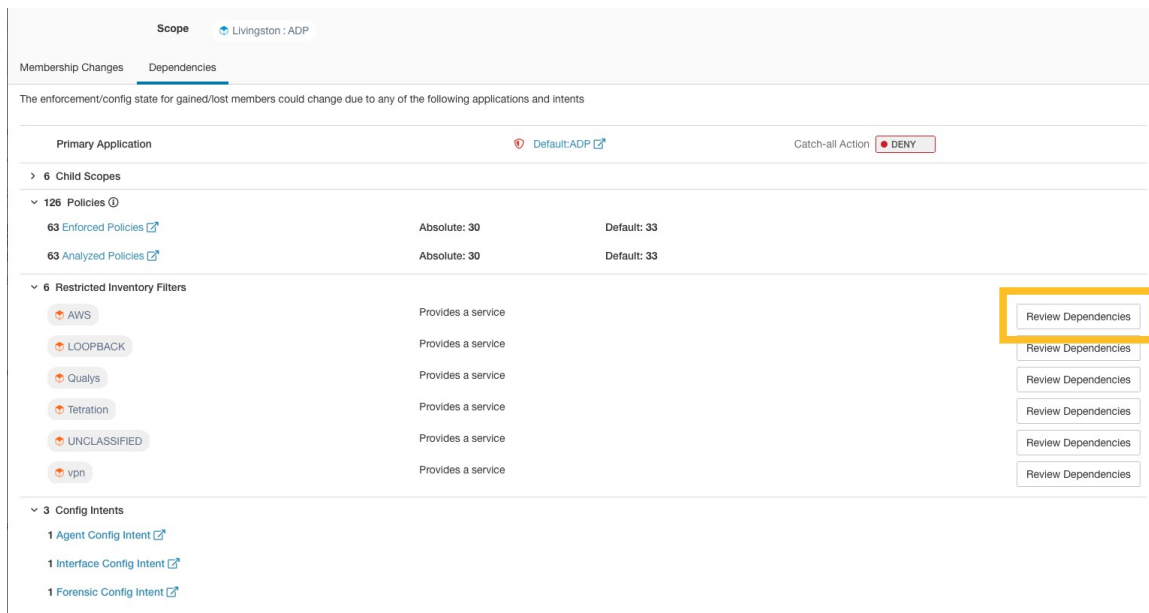
Figure 206: Scope Membership Changes



## Dependencies

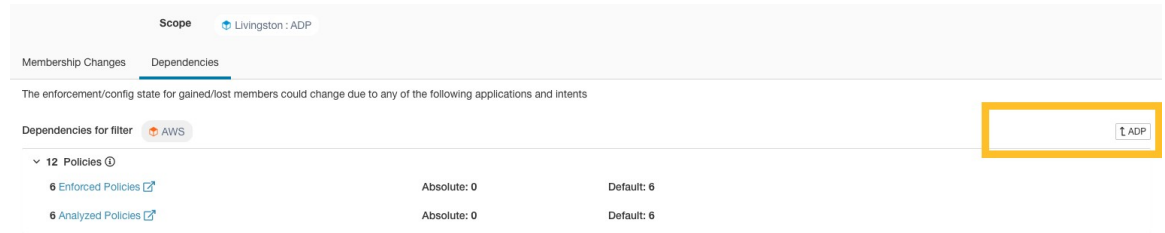
You can traverse down to nested dependencies by further selecting **Review Dependencies**

Figure 207: Review Dependencies



You can traverse back up the dependencies tree by selecting the selected Parent link:

**Figure 208: Parent Link**



The following are Scope Dependencies which may exist:

**Table 20: The following are Scope Dependencies which may exist**

| Type                                | Description                                                                                                       |
|-------------------------------------|-------------------------------------------------------------------------------------------------------------------|
| <b>Application</b>                  | Has primary and secondary application names and links to the specific workspaces under Segmentation.              |
| <b>Child Scopes</b>                 | Has names and links to child Scope Detail views. Allows drill down to lower level Dependencies.                   |
| <b>Policies</b>                     | Has analyzed and enforced policies counts and links to respective Global Policy Views filtered by selected scope. |
| <b>Restricted Inventory Filters</b> | Has names and links to child Filter Detail views. Allows drill down to lower level Dependencies.                  |
| <b>Config Intents</b>               | Has names and links to Agent, Interface and Forensics Config Intents views.                                       |

## Filter Query Change Impact Modal

Both **Membership Changes** and **Dependencies** tab can be accessed by following the link to **Review query change impact** on Inventory Filter Edit window.

## Membership Changes

Figure 209: Inventory Filter Membership Changes

**Edit Filter** ✕

**Name**

**Description**

**Query**  ✕

Filter matches 12 inventory items

**Scope** ADP ▾

- Restrict query to ownership scope
- Provides a service external of its scope

Review query change impact **Save** Cancel

## Dependencies

The following are Filter Dependencies which may exist:

| Type                  | Description                                                                                                      |
|-----------------------|------------------------------------------------------------------------------------------------------------------|
| <b>Policies</b>       | Has analyzed and enforced policies counts and links to respective Global Policy Views filtered by selected scope |
| <b>Config Intents</b> | Has names and links to Agent, Interface and Forensics Config Intents views                                       |

# Inventory Profile



**Note** An inventory profile page is linked from various places. One of the ways to see an inventory profile is to perform a search for inventory, then click an IP address to go to its profile. If you are working in the Scopes and Inventory page, click an IP address in the IP addresses tab, not an IP address in the Workloads tab. (Clicking an IP address in the Workloads tab displays the Workload Profile, not the Inventory Profile.)

The following information is available for the inventory:

| Field          | Description                                                                                                                                                                                                                                                                                                                                                                                                          |
|----------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Scopes         | List of scopes that the inventory belongs to.                                                                                                                                                                                                                                                                                                                                                                        |
| Inventory Type | <ul style="list-style-type: none"> <li>• <b>Flow Learnt</b> inventory was registered based on the observed flows.</li> <li>• <b>Labeled</b> inventory was manually uploaded using the inventory upload utility.</li> <li>• <b>Agent</b> inventory was reported by the software agent installed on a host.</li> <li>• <b>Tagged</b> inventory was either reported by connectors or external orchestrators.</li> </ul> |
| User Labels    | The list of user uploaded attributes for this inventory. See <a href="#">Workload Labels</a> for more details.                                                                                                                                                                                                                                                                                                       |

**Additional information is available only if both of the following are true:**

1. Inventory has been ingested through a cloud connector.
2. Segmentation is enabled for the virtual network in which the inventory resides.

| Field              | Description                                                                                                                                   |
|--------------------|-----------------------------------------------------------------------------------------------------------------------------------------------|
| Enforcement Health | The status information of the host software agent. See <a href="#">Agent Health Tab</a> for more details.                                     |
| Concrete Policies  | This tab shows Secure Workload concrete enforcement policies applied on the host. See <a href="#">Concrete Policies Tab</a> for more details. |
| Security Groups    | The list of security groups and their policies applied to this inventory.                                                                     |

## Inventory Profile Information

| Field                      | Description                                                                                                                                                                                                                  |
|----------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Experimental</b> Groups | A list of cluster or user-defined inventory filters that are used for policy live analysis.                                                                                                                                  |
| <b>Enforcement</b> Groups  | A list of cluster or user-defined inventory filters that are used for policy enforcement. They can be different from experimental groups depending on the versions of policies being analyzed and/or enforced in the system. |



- 
- Note** The inventory profile details may not be available for an IP address when:
- The inventory is excluded from collection rules.
  - In a unidirectional flow, the inventory is available only for two minutes, and then it is removed.
  - In a bidirectional flow, the inventory is available for 30 days. If no more flows are observed during these 30 days then the inventory details are removed.
- 

## Workload Profile

Workload profile displays detailed information about a host where Secure Workload software agent is installed. This section explains how to view a workload profile and the information it contains.



- 
- Note** A workload profile page is linked from various places. One of the ways to see a workload profile is to perform a search for host as described in search
- 

From the results of inventory search, click on IP address of the host to go to its profile. Based on the type of agent installed on the host, the following tabs are available on the page. Note that you may end up on inventory profile page if Secure Workload software agent is not installed on the host that this inventory belongs to.

## Labels and Scopes Tab

This tab includes the enforcement and experimental groups, scopes that the host belongs to. The experimental groups are inventory filters that are used for policy live analysis, while the enforcement groups are the filters that are used for policy enforcement. They can be different depending on the versions of policies being analyzed and/or enforced in the system.

Figure 210: Workload Labels and Scopes

- LABELS AND SCOPES
- AGENT HEALTH
- LONG LIVED PROCESSES
- PROCESS SNAPSHOTS
- INTERFACES
- PACKAGES
- VULNERABILITIES
- CONFIG
- STATS
- ENFORCEMENT HEALTH
- CONCRETE POLICIES
- CONTAINER POLICIES
- NETWORK ANOMALIES
- FILE HASHES
- DOWNLOAD LOGS

### Labels

Labels Key and Value for each Workload interface and the label source. See [User Guide](#) for more details.

Synced 3 Addition Pending 2 Deletion Pending 0

| Label Key <span style="font-size: small;">↑</span> | Label Value <span style="font-size: small;">↑</span> | 10.103.1.3 <span style="font-size: small;">↑</span> |
|----------------------------------------------------|------------------------------------------------------|-----------------------------------------------------|
| * org                                              | internal                                             | <span style="color: green;">●</span> cmdb           |
| * app                                              |                                                      | <span style="color: orange;">○</span> cmdb          |
| * env                                              |                                                      | <span style="color: orange;">○</span> cmdb          |
| * orchestrator_system/cluster_name                 | vCenter-alpine-vc01.tetrationanalytics.com           | <span style="color: green;">●</span> orchestrator   |
| * orchestrator_system/workload_type                | vm                                                   | <span style="color: green;">●</span> orchestrator   |

Rows per page 5 < 1 2 3 >

### Scopes and Applications

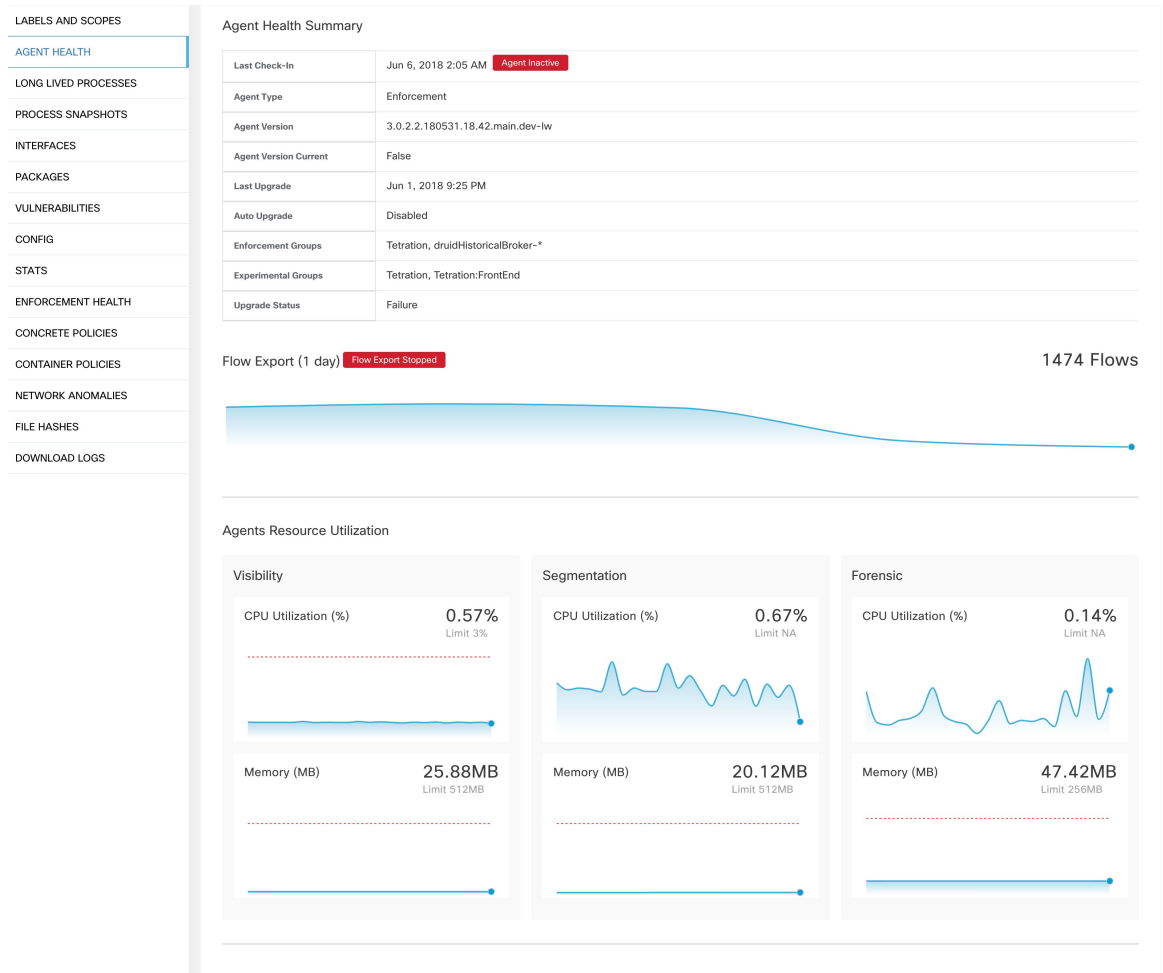
| <span style="font-size: small;">↑</span>                                                   | Primary Application <span style="font-size: small;">↑</span> | Analysis <span style="font-size: small;">↑</span>                                           | Enforcement <span style="font-size: small;">↑</span> |
|--------------------------------------------------------------------------------------------|--------------------------------------------------------------|---------------------------------------------------------------------------------------------|------------------------------------------------------|
| <span style="background-color: #e0e0e0; padding: 2px;">wildfire</span>                     | wildfire                                                     | Disabled                                                                                    | Disabled                                             |
| <span style="background-color: #e0e0e0; padding: 2px;">wildfire:internal</span>            | N/A                                                          | N/A                                                                                         | N/A                                                  |
| <span style="background-color: #e0e0e0; padding: 2px;">wildfire:internal:datacenter</span> | wildfire:internal:datacenter                                 | Version: p6<br>Policies: 17<br>Catch-All-Action: <span style="color: green;">●</span> ALLOW | Disabled                                             |

Rows per page 5 < 1 >

## Agent Health Tab

The status information of the host software agent such as it's type, OS platform, agent version and last check-in time are also shown in the **Agent Health** tab. See [Software Agent Config](#) for more details. This tab also shows detailed time series data for traffic bytes and packets occurred per one day.

Figure 211: Workload Agent Health Details



For users with root scope owner privileges, summary page also includes a section to collect and download agent logs for deep visibility and enforcement agents (versions 3.3 or later) within that root scope. Also note that this feature is not available for agents running on platforms AIX and SUSE Linux Enterprise Server (s390x-Linux on IBM Z architectures). Use “Initiate Log Collection” button to collect logs from the agent and then logs are available for download in a few minutes. If the download fails, retry collection of logs, and then attempt download again.



Figure 212: Agent Logs

## Process List Tab

This tab shows list of processes running on the host. A filter is also available to narrow down the list of processes based on the attributes of a process shown in table header below.

Figure 213: Workload Process List

| Process Command Line                      | User Name  | PID   | Parent PID | Libraries Count | Last Exec Content Change       | Last Exec Content/Attr Change | Last |
|-------------------------------------------|------------|-------|------------|-----------------|--------------------------------|-------------------------------|------|
| (flush-8.0)                               | root       | 12920 | 2          | 0               |                                |                               | May  |
| sshd: tetinstall@notty                    | tetinstall | 30783 | 30780      | 49              | Mar 27 2020 10:28:58 pm (EET)  | May 4 2020 03:04:23 pm (EEST) | May  |
| sshd: tetinstall                          | root       | 30780 | 17838      | 49              | Mar 27 2020 10:28:58 pm (EET)  | May 4 2020 03:04:23 pm (EEST) | May  |
| pickup                                    | postfix    | 865   | 6509       | 36              | Apr 3 2017 11:05:15 pm (EEST)  | May 4 2020 03:04:24 pm (EEST) |      |
| smtpd                                     | postfix    | 28513 | 6509       | 37              | Apr 3 2017 11:05:15 pm (EEST)  | May 4 2020 03:04:24 pm (EEST) |      |
| smtpd                                     | postfix    | 13098 | 6509       | 37              | Apr 3 2017 11:05:15 pm (EEST)  | May 4 2020 03:04:24 pm (EEST) | May  |
| /usr/sbin/anacron                         | root       | 31440 | 1          | 9               | Nov 23 2013 02:43:14 pm (EET)  | Mar 6 2018 08:58:09 pm (EET)  | May  |
| /usr/bin/atop                             | root       | 19529 | 1          | 7               | Aug 6 2019 05:59:40 pm (EEST)  | May 4 2020 03:01:24 pm (EEST) |      |
| /usr/bin/atop                             | root       | 27289 | 1          | 7               | Aug 6 2019 05:59:40 pm (EEST)  | May 4 2020 03:01:24 pm (EEST) | May  |
| pickup                                    | postfix    | 27381 | 6509       | 36              | Apr 3 2017 11:05:15 pm (EEST)  | May 4 2020 03:04:24 pm (EEST) | May  |
| java metrics_tsdbsdb.jar pipeline-#.xi... | tetter     | 14488 | 28926      | 19              | Dec 11 2019 12:41:47 pm (EET)  | May 4 2020 03:06:27 pm (EEST) |      |
| java metrics_tsdbsdb.jar pipeline-#.xi... | tetter     | 14431 | 28925      | 19              | Dec 11 2019 12:41:47 pm (EET)  | May 4 2020 03:06:27 pm (EEST) |      |
| java metrics_tsdbsdb.jar pipeline-#.xi... | tetter     | 29308 | 28926      | 19              | Dec 11 2019 12:41:47 pm (EET)  | May 4 2020 03:06:27 pm (EEST) | May  |
| python /opt/tetration/itm/itm.py          | root       | 9671  | 15821      | 27              | Aug 18 2016 06:14:31 pm (EEST) | Mar 6 2018 08:59:54 pm (EET)  |      |
| /opt/tetration/efe/tet-efe-efe.conf...    | tetter     | 13500 | 13362      | 52              | May 4 2020 09:21:21 am (EEST)  | May 4 2020 09:20:41 pm (EEST) |      |
| /opt/tetration/collector/tet-collec...    | tetter     | 13414 | 28030      | 53              | May 4 2020 08:36:24 am (EEST)  | May 4 2020 09:19:47 pm (EEST) |      |
| /opt/tetration/efe/tet-efe-relay ef...    | tetter     | 13362 | 30934      | 4               | May 4 2020 07:27:16 pm (EEST)  | May 4 2020 09:20:37 pm (EEST) |      |
| tet-sensor                                | tet-sensor | 2817  | 2807       | 14              | Apr 30 2020 02:52:26 am (EEST) | May 4 2020 10:16:21 pm (EEST) |      |
| tet-main                                  | root       | 2809  | 2805       | 4               | Apr 30 2020 02:52:26 am (EEST) | May 4 2020 10:16:21 pm (EEST) |      |
| tet-engine                                | root       | 2805  | 1          | 5               | Apr 30 2020 02:52:26 am (EEST) | May 4 2020 10:16:21 pm (EEST) |      |

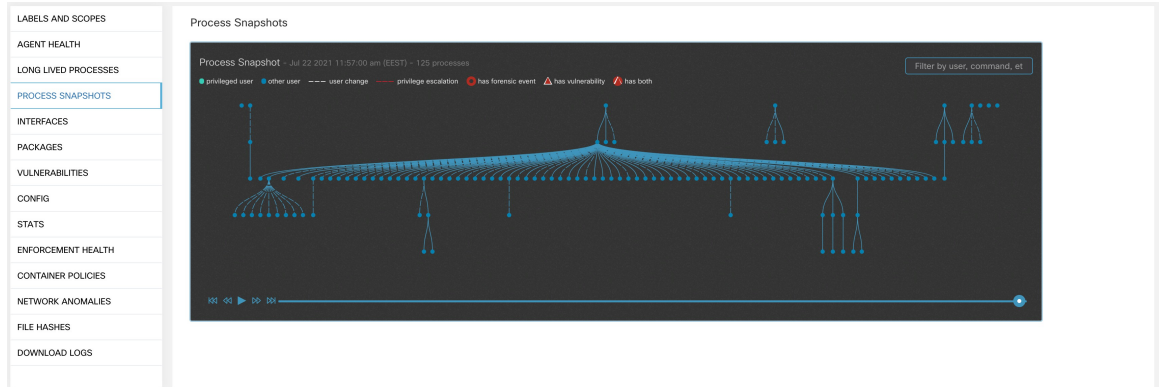
### Attribute Descriptions:

| Attribute                | Description                                                                                                                                                                                                                                                                      |
|--------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Last Exec Content Change | Similar to mtime in linux. It is the timestamp when only the file content changes.                                                                                                                                                                                               |
| Last Exec Content Change | Similar to ctime in linux. It is the timestamp when either the file content or attribute changes.                                                                                                                                                                                |
| Last Seen                | Last time when the process is observed. Available when the process is dead.                                                                                                                                                                                                      |
| CPU Usage                | CPU usage trend by the process in the past hour.                                                                                                                                                                                                                                 |
| Memory Usage             | Memory usage trend by the process in the past hour.                                                                                                                                                                                                                              |
| Process Binary Hash      | SHA256 hash of the process binary in hex string, also known as process hash for short. Not available for kernel processes.                                                                                                                                                       |
| Anomaly Score            | Process hash (anomaly) score. See <a href="#">Process Hash Anomaly Detection</a> for more information.                                                                                                                                                                           |
| Verdict                  | Verdict of the process hash (either Malicious or Benign). The verdict is determined based on whether the process hash belongs to any user-defined hash list or known threat-intelligence hash database. See <a href="#">Process Hash Anomaly Detection</a> for more information. |
| Verdict Source           | Source of the verdict. The verdict source can be either User Defined, or Secure Workload Cloud, or NIST. This attribute is known as Hash DB Source in previous releases. See <a href="#">Process Hash Anomaly Detection</a> for more information.                                |

## Process Snapshot Tab

This tab shows searchable process tree observed on the workload.

**Figure 214: Workload Process Snapshot**



## Interfaces Tab

This tab shows details about the network interfaces installed on the host. It's available for all types of software agents.

**Figure 215: Workload Interface List**

The screenshot shows the 'Interfaces' tab in the application. It features a table with columns for Name, Mac Address, VRF, Family Type, IP Address, and Netmask. Below the table are filter sections for Enforcement Groups, Experimental Groups, User Labels, and Scopes.

| Name   | Mac Address       | VRF     | Family Type | IP Address               | Netmask                 |
|--------|-------------------|---------|-------------|--------------------------|-------------------------|
| lo     | 00:00:00:00:00:00 | Default | IPV4        | 127.0.0.1                | 255.0.0.0               |
| lo     | 00:00:00:00:00:00 | Default | IPV6        | ::1                      | fff:fff:fff:fff:fff:fff |
| ens192 | 00:50:56:88:1a:aa | Default | IPV4        | 10.103.4.105             | 255.255.248.0           |
| ens192 | 00:50:56:88:1a:aa | Default | IPV6        | fe80::250:56ff:fe88:1aaa | fff:fff:fff:fff::       |

Filter sections below the table:

- Enforcement Groups: Default ...2 more
- Experimental Groups: Default ...2 more
- User Labels: App = App1
- Scopes: Default ...2 more

## Software Packages Tab

This tab shows the list of packages installed on the host. You can selectively view software packages based on package attributes in the table header.

Figure 216: Software Packages List

| Name ↓            | Version ↑   | Architecture ↑ | Publisher ↑ |
|-------------------|-------------|----------------|-------------|
| PyYAML ▲          | 3.10        |                |             |
| MAKEDEV           | 3.24        |                |             |
| bzip2             | 1.0.5       |                |             |
| bridge-utils      | 1.2         |                |             |
| binutils          | 2.20.51.0.2 |                |             |
| bind-utils        | 9.8.2       |                |             |
| bash              | 4.1.2       |                |             |
| basesystem        | 10.0        |                |             |
| b43-openfwfwf     | 5.2         |                |             |
| avahi-libs        | 0.6.25      |                |             |
| authconfig        | 6.1.12      |                |             |
| audit-libs-python | 2.4.5       |                |             |
| audit-libs        | 2.4.5       |                |             |
| audit             | 2.4.5       |                |             |
| attr              | 2.4.44      |                |             |
| atop              | 1.27        |                |             |
| atk               | 1.30.0      |                |             |
| at                | 3.1.10      |                |             |
| ansible           | 1.9.6       |                |             |
| alsa-lib          | 1.0.22      |                |             |

< 1 2 >

## Vulnerabilities Tab

The **VULNERABILITIES** tab displays the identified CVEs on the workload according to Common Vulnerability Scoring System (CVSS V2, V3) and Cisco Security Risk Score.

Figure 217: Vulnerabilities Tab

| CVE #          | Package Name           | Package Version | Score (V2) | Score (V3) | Severity (V2) | Base Severity (V3) | Access Vector (V2) | Access Complexity (V2) | Authentication (V2) | Confidentiality Impact (V2) |
|----------------|------------------------|-----------------|------------|------------|---------------|--------------------|--------------------|------------------------|---------------------|-----------------------------|
| CVE-2019-1389  | msserver2016datacenter | 1607-14393.3300 | 7.7        | 8.4        | HIGH          | HIGH               | ADJACENT_NETWORK   | LOW                    | SINGLE              | COMPLETE                    |
| CVE-2019-1388  | msserver2016datacenter | 1607-14393.3300 | 7.2        | 7.8        | HIGH          | HIGH               | LOCAL              | LOW                    | NONE                | COMPLETE                    |
| CVE-2019-1384  | msserver2016datacenter | 1607-14393.3300 | 6.5        | 9.9        | MEDIUM        | CRITICAL           | NETWORK            | LOW                    | SINGLE              | PARTIAL                     |
| CVE-2019-1383  | msserver2016datacenter | 1607-14393.3300 | 4.6        | 7.8        | MEDIUM        | HIGH               | LOCAL              | LOW                    | NONE                | PARTIAL                     |
| CVE-2019-1382  | msserver2016datacenter | 1607-14393.3300 | 2.1        | 5.5        | LOW           | MEDIUM             | LOCAL              | LOW                    | NONE                | PARTIAL                     |
| CVE-2019-1381  | msserver2016datacenter | 1607-14393.3300 | 2.1        | 5.5        | LOW           | MEDIUM             | LOCAL              | LOW                    | NONE                | PARTIAL                     |
| CVE-2019-1380  | msserver2016datacenter | 1607-14393.3300 | 4.6        | 7.8        | MEDIUM        | HIGH               | LOCAL              | LOW                    | NONE                | PARTIAL                     |
| CVE-2019-1374  | msserver2016datacenter | 1607-14393.3300 | 4.3        | 5.5        | MEDIUM        | MEDIUM             | NETWORK            | MEDIUM                 | NONE                | PARTIAL                     |
| CVE-2019-1371  | Internet Explorer      | 11.0.155        | 7.8        | 7.5        | HIGH          | HIGH               | NETWORK            | HIGH                   | NONE                | COMPLETE                    |
| CVE-2019-1367  | Internet Explorer      | 11.0.155        | 7.6        | 7.5        | HIGH          | HIGH               | NETWORK            | HIGH                   | NONE                | COMPLETE                    |
| CVE-2019-1357  | Internet Explorer      | 11.0.155        | 4.3        | 4.3        | MEDIUM        | MEDIUM             | NETWORK            | MEDIUM                 | NONE                | NONE                        |
| CVE-2019-1238  | Internet Explorer      | 11.0.155        | 7.1        | 6.4        | HIGH          | MEDIUM             | NETWORK            | HIGH                   | SINGLE              | COMPLETE                    |
| CVE-2019-1192  | Internet Explorer      | 11.0.155        | 4.3        | 4.3        | MEDIUM        | MEDIUM             | NETWORK            | MEDIUM                 | NONE                | PARTIAL                     |
| CVE-2019-11335 | msserver2016datacenter | 1607-14393.3300 | 2.1        | 6.5        | LOW           | MEDIUM             | LOCAL              | LOW                    | NONE                | PARTIAL                     |
| CVE-2019-0719  | msserver2016datacenter | 1607-14393.3300 | 9          | 9.1        | HIGH          | CRITICAL           | NETWORK            | LOW                    | SINGLE              | COMPLETE                    |
| CVE-2019-0712  | msserver2016datacenter | 1607-14393.3300 | 6.8        | 6.8        | MEDIUM        | MEDIUM             | NETWORK            | LOW                    | SINGLE              | NONE                        |
| CVE-2019-0608  | Internet Explorer      | 11.0.155        | 4.3        | 4.3        | MEDIUM        | MEDIUM             | NETWORK            | MEDIUM                 | NONE                | NONE                        |
| CVE-2018-12207 | msserver2016datacenter | 1607-14393.3300 | 4.9        | 6.5        | MEDIUM        | MEDIUM             | LOCAL              | LOW                    | NONE                | NONE                        |

Figure 218: Workload Profile: Vulnerabilities Tab

| CVE #          | Package Name       | Package Version              | Score (V2) | Score (V3) | Cisco Security Risk Score | Severity (V2) | Base Severity (V3) | Severity (Cisco Security Risk Score) | Access |
|----------------|--------------------|------------------------------|------------|------------|---------------------------|---------------|--------------------|--------------------------------------|--------|
| CVE-2024-25062 | libxml2            | 2.9.10+dfsg-Subuntu0.20.04.6 | 7.5        | 50.1       | 50.1                      | HIGH          | HIGH               | HIGH                                 |        |
| CVE-2024-24806 | libw1              | 1.34.2-1ubuntu1.3            | 7.3        | 50.1       | 50.1                      | HIGH          | HIGH               | HIGH                                 |        |
| CVE-2024-22365 | libpam-modules     | 1.3.1-Subuntu4.6             | 5.5        | 22.3       | 22.3                      | MEDIUM        | LOW                | LOW                                  |        |
| CVE-2024-22365 | libpam-modules-bin | 1.3.1-Subuntu4.6             | 5.5        | 22.3       | 22.3                      | MEDIUM        | LOW                | LOW                                  |        |
| CVE-2024-22365 | libpam-runtime     | 1.3.1-Subuntu4.6             | 5.5        | 22.3       | 22.3                      | MEDIUM        | LOW                | LOW                                  |        |
| CVE-2024-22365 | libpam0g           | 1.3.1-Subuntu4.6             | 5.5        | 22.3       | 22.3                      | MEDIUM        | LOW                | LOW                                  |        |
| CVE-2024-22195 | python3-jinja2     | 2.10.1-2                     | 6.1        | 44.6       | 44.6                      | MEDIUM        | MODERATE           | MODERATE                             |        |
| CVE-2024-0727  | libssl1.1          | 1.1.1f-1ubuntu2.19           | 5.5        | 25.0       | 25.0                      | MEDIUM        | MODERATE           | MODERATE                             |        |
| CVE-2024-0727  | openssl            | 1.1.1f-1ubuntu2.19           | 5.5        | 25.0       | 25.0                      | MEDIUM        | MODERATE           | MODERATE                             |        |
| CVE-2024-0567  | libgnutls30        | 3.6.13-2ubuntu1.8            | 7.5        | 50.1       | 50.1                      | HIGH          | HIGH               | HIGH                                 |        |
| CVE-2024-0553  | libgnutls30        | 3.6.13-2ubuntu1.8            | 7.5        | 50.1       | 50.1                      | HIGH          | HIGH               | HIGH                                 |        |
| CVE-2023-7104  | libsqlite3-0       | 3.31.1-4ubuntu0.5            | 7.3        | 18.6       | 18.6                      | HIGH          | LOW                | LOW                                  |        |
| CVE-2023-6918  | libssh-4           | 0.9.3-2ubuntu2.3             | 5.3        | 41.8       | 41.8                      | MEDIUM        | MODERATE           | MODERATE                             |        |

# Agent Configuration Tab

This tab shows software agent settings. It is only available for Deep Visibility and Enforcement Agents. These settings can be modified using Agent Configuration Intents via the agent config page. See [Software Agent Config](#)

Figure 219: Applied Workload Configuration

The screenshot displays the 'Applied Workload Configuration' interface. On the left is a sidebar menu with the following items: LABELS AND SCOPES, AGENT HEALTH, LONG LIVED PROCESSES, PROCESS SNAPSHOTS, INTERFACES, PACKAGES, VULNERABILITIES, CONFIG (highlighted), STATS, ENFORCEMENT HEALTH, CONTAINER POLICIES, NETWORK ANOMALIES, FILE HASHES, and DOWNLOAD LOGS. The main panel is titled 'Config' and contains the following settings:

- Config Intent
- Apply profile **enforcer** to filter **Enf-Workloads**
- Config Profile
- Enforcement
  - Enforcement
  - Windows Enforcement Mode - WFP
  - Preserve Rules
  - Allow Broadcast
  - Allow Multicast
  - Allow Link Local Addresses
  - CPU Quota Mode - Adjusted (3%)
  - Memory Quota Limit - 512MB
- Flow Visibility
  - Flow Analysis Fidelity - Detailed
  - Data Plane
  - Auto-Upgrade
  - PID Lookup
  - CPU Quota Mode - Adjusted (3%)
  - Memory Quota Limit - 512MB
- Process Visibility and Forensics
  - Forensics
  - Meltdown Exploit Detection
  - CPU Quota Mode - Adjusted (3%)
  - Memory Quota Limit - 256MB

## Agent Statistics Tab

This tab shows statistics about the Secure Workload agent installed on the host. It's only available for Deep Visibility and Enforcement Agents.

Figure 220: Agent Statistics



## Concrete Policies Tab

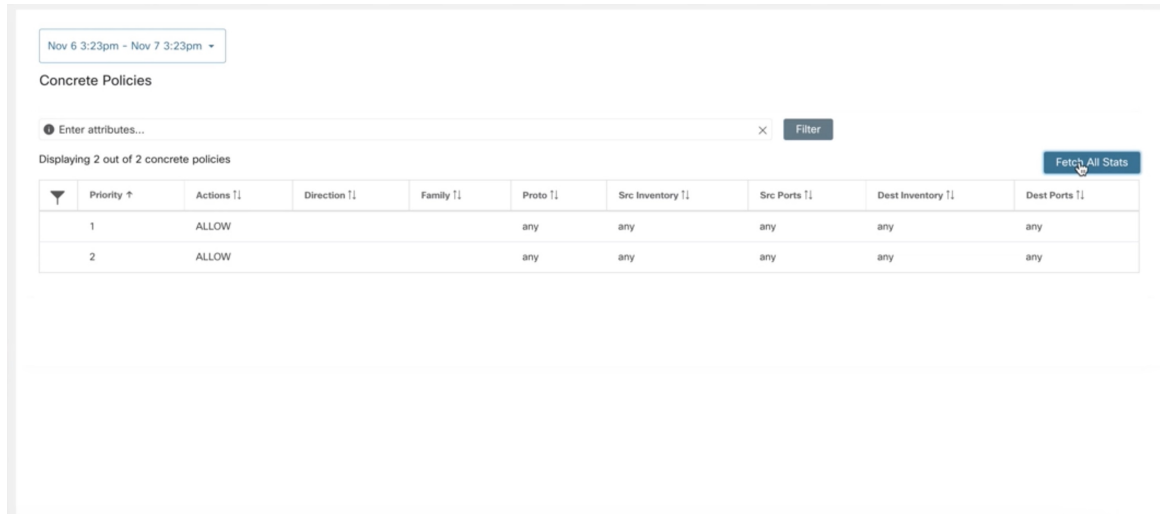
When a workspace is enforced, each workload receives only the policies in that workspace that are specific to that workload. These policies that are actually programmed on each workload are called *concrete policies*.

For example, suppose the provider specified in a policy with action ALLOW includes all inventory in the subnet 1.1.1.0/24. When this policy is installed on a workload with a Secure Workload agent and having IP address 1.1.1.2, the firewall rules look like this:

1. For incoming traffic firewall rules allow traffic destined to 1.1.1.2 specifically, not to the whole subnet 1.1.1.0/24.
2. For outgoing traffic firewall rules allow traffic sourced from 1.1.1.2 specifically, not from the whole subnet 1.1.1.0/24.

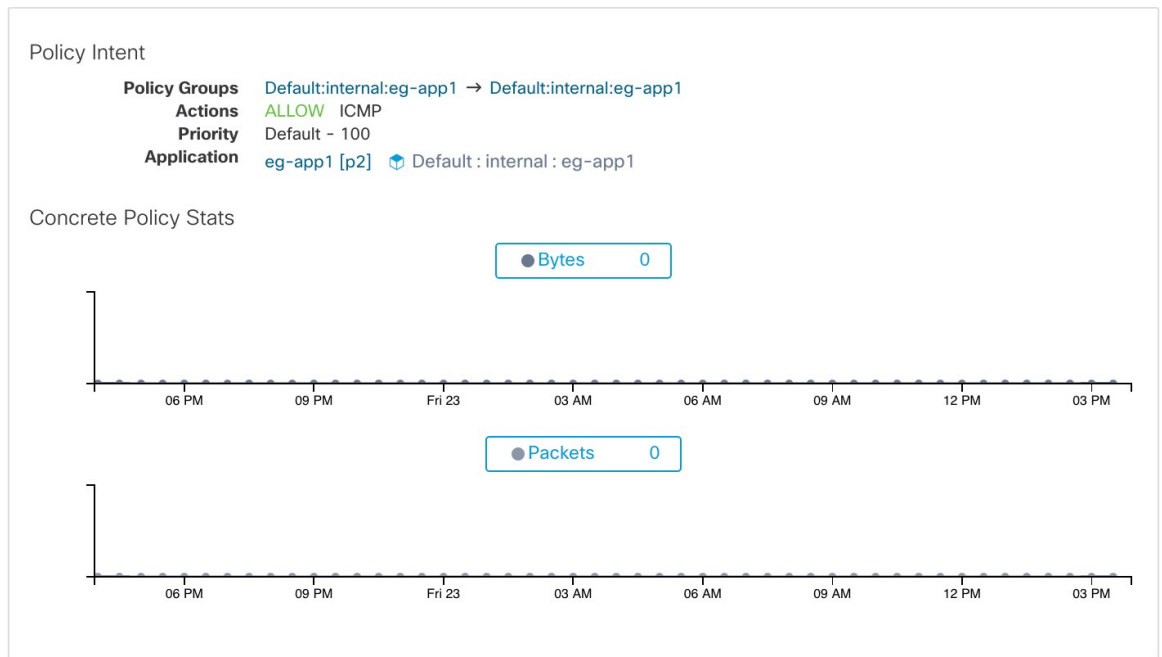
The CONCRETE POLICIES tab in the Workload Profile shows Secure Workload concrete enforcement policies applied on the host. Each row in this table corresponds to a firewall rule implemented on the host. Each policy row can be further expanded to display the logical intent from which this concrete policy derived. Packet and byte count time series view is also available for each rule. Click the **Fetch All Stats** button to view packets and bytes count for each rule. A filter is also available in this tab to narrow the list of enforced policies based on attributes of a policy shown in table header below. This tab is only available when the installed agent is enabled for enforcement.

Figure 221: Concrete Policy List



In the image below, **Policy Groups** shows the consumer and provider:

Figure 222: Concrete Policy Row



## Container Policies Tab

This tab shows Secure Workload concrete enforcement policies applied on the containers. Each row in this table corresponds to a firewall rule implemented on the container pod.



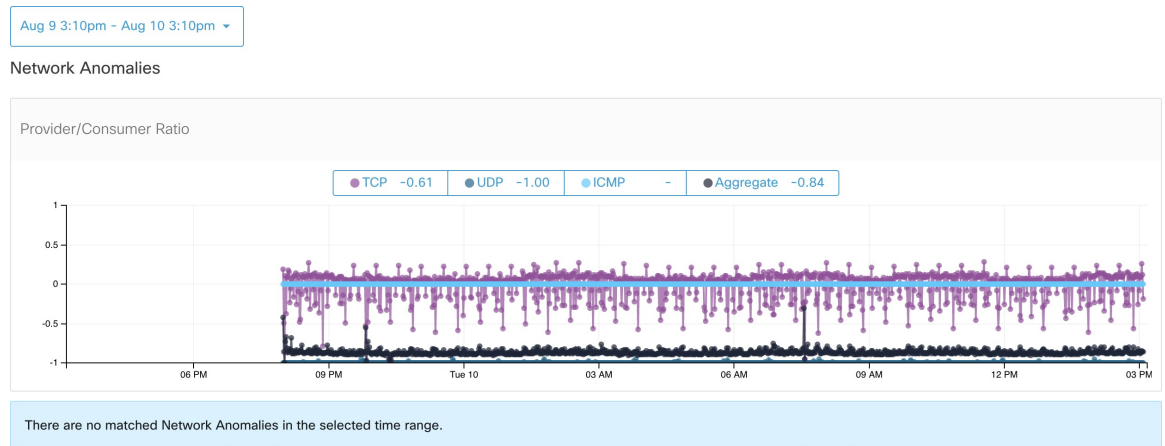
Figure 223: Container Concrete Policy List

| Pod ID          | Priority | Packets | Bytes | Actions | Direction | Family | Proto | Src Inventory | Src Ports | Dest Inventory | Dest Ports |
|-----------------|----------|---------|-------|---------|-----------|--------|-------|---------------|-----------|----------------|------------|
| 7abc1d87-27d... | 27       | N/A     | N/A   | ALLOW   | INGRESS   | IPV4   | TCP   | 172.0.2.4     | any       | 172.0.1.6/32   | 10000      |
| 7abc239a-27d... | 28       | N/A     | N/A   | ALLOW   | EGRESS    | IPV4   | TCP   | 172.0.1.5/32  | 10000     | 172.0.2.4      | any        |
| 117113c6-26f... | 28       | N/A     | N/A   | ALLOW   | EGRESS    | IPV4   | TCP   | 172.0.1.4/32  | 10000     | 172.0.2.4      | any        |
| 7abc1d87-27d... | 28       | N/A     | N/A   | ALLOW   | EGRESS    | IPV4   | TCP   | 172.0.1.6/32  | 10000     | 172.0.2.4      | any        |
| 7abc239a-27d... | 29       | N/A     | N/A   | ALLOW   | INGRESS   | IPV4   | TCP   | 172.0.2.4     | any       | 172.0.1.5/32   | 10001      |
| 117113c6-26f... | 29       | N/A     | N/A   | ALLOW   | INGRESS   | IPV4   | TCP   | 172.0.2.4     | any       | 172.0.1.4/32   | 10001      |
| 7abc1d87-27d... | 29       | N/A     | N/A   | ALLOW   | INGRESS   | IPV4   | TCP   | 172.0.2.4     | any       | 172.0.1.6/32   | 10001      |
| 7abc239a-27d... | 30       | N/A     | N/A   | ALLOW   | EGRESS    | IPV4   | TCP   | 172.0.1.5/32  | 10001     | 172.0.2.4      | any        |
| 117113c6-26f... | 30       | N/A     | N/A   | ALLOW   | EGRESS    | IPV4   | TCP   | 172.0.1.4/32  | 10001     | 172.0.2.4      | any        |
| 7abc1d87-27d... | 30       | N/A     | N/A   | ALLOW   | EGRESS    | IPV4   | TCP   | 172.0.1.6/32  | 10001     | 172.0.2.4      | any        |

## Network Anomalies Tab

This tab helps to identify the events with large data movements in or out of this workload. See [PCR-Based Network Anomaly Detection](#) for more information.

Figure 224: Workload Network Anomalies



## File Hashes Tab

This tab detects process hash anomalies by assessing the consistency of process binary hashes across the system. See [Process Hash Anomaly Detection](#) for more info.

Figure 225: Workload File Hashes

| Begin                               | SHA1 Hash | SHA256 Hash | File Path                                         | Anomaly Score | Reason  | Links            |
|-------------------------------------|-----------|-------------|---------------------------------------------------|---------------|---------|------------------|
| <input checked="" type="checkbox"/> | 6b6e65d   | 74654b5     | c:\program files\vmware\vmware tools\vmtoolsd.exe | 0.00          | Flagged | Inventory Search |

## Software Packages

The **Software Packages** feature set allows viewing packages installed on hosts and the vulnerabilities affecting them. Specifically, it allows to:

- View packages registered with the following package managers:
  - Linux: Redhat Package Manager (RPM) and Debian Package Manager (dpkg)
  - Windows: Windows Registry Service
- View Common Vulnerabilities and Exposures (CVEs) affecting packages installed on a host.
- Define inventory filters using the package name and version.

## Packages Tab

To view packages installed on a host, navigate to the packages tab on the workload profile [Workload Profile](#) page.

Figure 226: Workload profile packages

Packages

Enter attributes...

Displaying 22 of 22

| Name ↓            | Version ↑   | Architecture ↑ | Publisher ↑ |
|-------------------|-------------|----------------|-------------|
| PyYAML ▲          | 3.10        |                |             |
| MAKEDEV           | 3.24        |                |             |
| bzip2             | 1.0.5       |                |             |
| bridge-utils      | 1.2         |                |             |
| binutils          | 2.20.51.0.2 |                |             |
| bind-utils        | 9.8.2       |                |             |
| bash              | 4.1.2       |                |             |
| basesystem        | 10.0        |                |             |
| b43-openfwfwf     | 5.2         |                |             |
| avahi-libs        | 0.6.25      |                |             |
| authconfig        | 6.1.12      |                |             |
| audit-libs-python | 2.4.5       |                |             |
| audit-libs        | 2.4.5       |                |             |
| audit             | 2.4.5       |                |             |
| attr              | 2.4.44      |                |             |
| atop              | 1.27        |                |             |
| atk               | 1.30.0      |                |             |
| at                | 3.1.10      |                |             |
| ansible           | 1.9.6       |                |             |
| alsa-lib          | 1.0.22      |                |             |

< 2 >

## Common Vulnerabilities and Exposures

Along with the packages, the **VULNERABILITIES** tab provides details on the CVEs identified on your workloads. Each vulnerability contains a link to the Nation Vulnerability Database (NVD) which provides more information on the specific vulnerability. In addition to displaying the CVE ID, we also display the impact score (on a scale of 10) based on the scoring methods—Cisco Security Risk Score, CVSS V2, and CVSS V3, and the severity of the vulnerability.

Along with the packages, the **VULNERABILITIES** tab provides details on the CVEs identified on your workloads. Each vulnerability contains a link to the Nation Vulnerability Database (NVD) which provides more information on the specific vulnerability. In addition to displaying the CVE ID, we also display the impact score (on a scale of 10) based on the scoring methods—CVSS V2 and CVSS V3, and the severity of the vulnerability.

## Windows Packages and CVEs

Following section lists the behavior of Windows agent with regard to reporting package information to Secure Workload.

- Windows applications, PowerShell, IE are reported as packages. .net framework is also reported as a package.
- Other Windows applications like notepad.exe, cmd.exe, mstsc.exe, and so on are not reported.

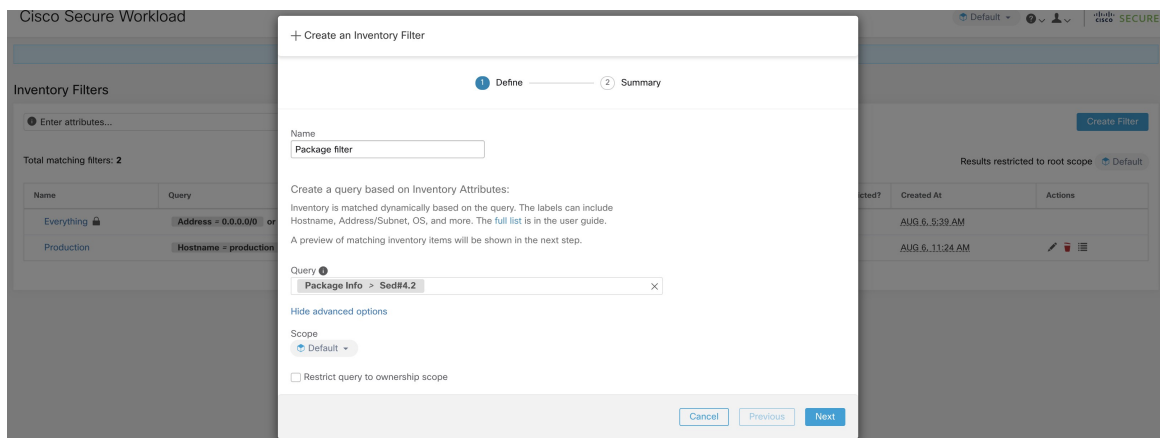
- Windows server configured roles and features are reported as packages but the version may be incorrect. For example: If the DNS server is configured, reported version will either 0 or 8.
- Windows agent reports 3rd party products installed using MSI installer or exe installer:
  - For MSI installers, MSI APIs are used to retrieve package information. For example, version, publisher, package name.
  - If the exe installer is used to install the package, package information is retrieved from the registry.
  - Package installer fields like version, publisher is optional. If version is missing, the package will not be reported.
  - If a product is extracted from zip file or installed as an app, it will not be reported in the package list.

## Inventory Filters

Package related information can be searched by defining an inventory filter with the package name and version (optional).

**The syntax for this filter is as follows:** `PackageName#PackageVersion`

**Figure 227: Inventory package**



The following operations are supported:

- Equality - returns hosts with packages matching PackageName and the PackageVersion (if provided).
- Inequality - returns hosts with packages matching PackageName but not the PackageVersion (if provided).
- Greater Than - returns hosts with packages matching PackageName and with version greater than PackageVersion.
- Greater Than or Equal To - returns hosts with packages matching PackageName and with version greater than or equal to PackageVersion.
- Less Than - returns hosts with packages matching PackageName and with version less than PackageVersion.

- Less Than or Equal To - returns hosts with packages matching PackageName and with version less than or equal to PackageVersion.

## Vulnerability Data Visibility

Using vulnerability data visibility, you can detect and view vulnerabilities affecting packages and processes on a host. Define inventory filters using:

- CVE IDs
- CVSS V2 Scores
- CVSS V3 Scores
- Cisco Security Risk Scores
- CVSS V2 Attributes
- CVSS V3 Attributes
- Cisco Security Risk Score Attributes

## Workload Profile Page

Vulnerability related information affecting packages and processes on a system is displayed on the [Workload Profile](#) page.

### Packages Tab

The packages tab lists packages installed on a host and vulnerabilities affecting them.

Figure 228: Workload profile packages

LABELS AND SCOPES

AGENT HEALTH

LONG LIVED PROCESSES

PROCESS SNAPSHOTS

INTERFACES

PACKAGES

VULNERABILITIES

CONFIG

STATS

ENFORCEMENT HEALTH

CONCRETE POLICIES

CONTAINER POLICIES

NETWORK ANOMALIES

FILE HASHES

DOWNLOAD LOGS

Packages

Enter attributes... Filter

Displaying 22 of 22

| Name ↓            | Version ↑   | Architecture ↑ | Publisher ↑ |
|-------------------|-------------|----------------|-------------|
| PyYAML ▲          | 3.10        |                |             |
| MAKEDEV           | 3.24        |                |             |
| bzip2             | 1.0.5       |                |             |
| bridge-utils      | 1.2         |                |             |
| binutils          | 2.20.51.0.2 |                |             |
| bind-utils        | 9.8.2       |                |             |
| bash              | 4.1.2       |                |             |
| basesystem        | 10.0        |                |             |
| b43-openfwfwf     | 5.2         |                |             |
| avahi-libs        | 0.6.25      |                |             |
| authconfig        | 6.1.12      |                |             |
| audit-libs-python | 2.4.5       |                |             |
| audit-libs        | 2.4.5       |                |             |
| audit             | 2.4.5       |                |             |
| attr              | 2.4.44      |                |             |
| atop              | 1.27        |                |             |
| atk               | 1.30.0      |                |             |
| at                | 3.1.10      |                |             |
| ansible           | 1.9.6       |                |             |
| alsa-lib          | 1.0.22      |                |             |

< 1 2 >

## Process List Tab

Long-lived processes are displayed under the process list tab.

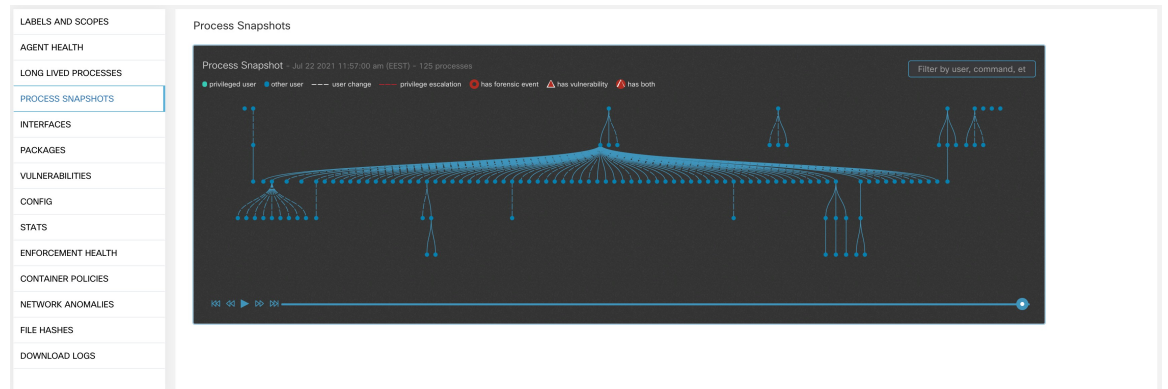
Figure 229: Workload profile process list

| Process Command Line                    | User Name  | PID   | Parent PID | Libraries Count | Last Exec Content Change       | Last Exec Content/Attr Change | Last |
|-----------------------------------------|------------|-------|------------|-----------------|--------------------------------|-------------------------------|------|
| (flush-8.0)                             | root       | 12920 | 2          | 0               |                                |                               | May  |
| sshd: tetinstall@notty                  | tetinstall | 30783 | 30780      | 49              | Mar 27 2020 10:28:58 pm (EET)  | May 4 2020 03:04:23 pm (EEST) | May  |
| sshd: tetinstall                        | root       | 30780 | 17838      | 49              | Mar 27 2020 10:28:58 pm (EET)  | May 4 2020 03:04:23 pm (EEST) | May  |
| pickup                                  | postfix    | 865   | 6509       | 36              | Apr 3 2017 11:05:15 pm (EEST)  | May 4 2020 03:04:24 pm (EEST) |      |
| smtpd                                   | postfix    | 28513 | 6509       | 37              | Apr 3 2017 11:05:15 pm (EEST)  | May 4 2020 03:04:24 pm (EEST) |      |
| smtpd                                   | postfix    | 13098 | 6509       | 37              | Apr 3 2017 11:05:15 pm (EEST)  | May 4 2020 03:04:24 pm (EEST) | May  |
| /usr/sbin/anacron                       | root       | 31440 | 1          | 9               | Nov 23 2013 02:43:14 pm (EET)  | Mar 6 2018 08:58:09 pm (EET)  | May  |
| /usr/bin/atop                           | root       | 19529 | 1          | 7               | Aug 6 2019 05:59:40 pm (EEST)  | May 4 2020 03:01:24 pm (EEST) |      |
| /usr/bin/atop                           | root       | 27289 | 1          | 7               | Aug 6 2019 05:59:40 pm (EEST)  | May 4 2020 03:01:24 pm (EEST) | May  |
| pickup                                  | postfix    | 27381 | 6509       | 36              | Apr 3 2017 11:05:15 pm (EEST)  | May 4 2020 03:04:24 pm (EEST) | May  |
| java metrics_tsdb.jar pipeline-#t.xi... | tetter     | 14488 | 28926      | 19              | Dec 11 2019 12:41:47 pm (EET)  | May 4 2020 03:06:27 pm (EEST) |      |
| java metrics_tsdb.jar pipeline-#t.xi... | tetter     | 14431 | 28925      | 19              | Dec 11 2019 12:41:47 pm (EET)  | May 4 2020 03:06:27 pm (EEST) |      |
| java metrics_tsdb.jar pipeline-#t.xi... | tetter     | 29308 | 28926      | 19              | Dec 11 2019 12:41:47 pm (EET)  | May 4 2020 03:06:27 pm (EEST) | May  |
| python /opt/tetration/itm/itm.py        | root       | 9671  | 15821      | 27              | Aug 18 2016 06:14:31 pm (EEST) | Mar 6 2018 08:59:54 pm (EET)  |      |
| /opt/tetration/efe/tet-efe-efe.conf...  | tetter     | 13500 | 13362      | 52              | May 4 2020 09:21:21 am (EEST)  | May 4 2020 09:20:41 pm (EEST) |      |
| /opt/tetration/collector/tet-collec...  | tetter     | 13414 | 28030      | 53              | May 4 2020 08:36:24 am (EEST)  | May 4 2020 09:19:47 pm (EEST) |      |
| /opt/tetration/efe/tet-efe-relay ef...  | tetter     | 13362 | 30934      | 4               | May 4 2020 07:27:16 pm (EEST)  | May 4 2020 09:20:37 pm (EEST) |      |
| tet-sensor                              | tet-sensor | 2817  | 2807       | 14              | Apr 30 2020 02:52:26 am (EEST) | May 4 2020 10:16:21 pm (EEST) |      |
| tet-main                                | root       | 2809  | 2805       | 4               | Apr 30 2020 02:52:26 am (EEST) | May 4 2020 10:16:21 pm (EEST) |      |
| tet-engine                              | root       | 2805  | 1          | 5               | Apr 30 2020 02:52:26 am (EEST) | May 4 2020 10:16:21 pm (EEST) |      |

## Process Snapshot Tab

Vulnerability information is displayed for all processes in the process tree under the process snapshot tab.

Figure 230: Workload profile process snapshot tab



## Vulnerabilities Tab

The **VULNERABILITIES** tab displays the CVEs identified by Secure Workload on the workload.

For each CVE, besides basic impact metrics, exploit information based on our threat intelligence is displayed:

- Exploit Count: Number of times CVE was seen exploited in the wild in the last year
- Last Exploited: Last time CVE was seen exploited in the wild by our threat intelligence

Figure 231: Workload Profile: Vulnerabilities Tab

| CVE ID         | Package Name           | Package Version | Score (V2) | Score (V3) | Severity (V2) | Base Severity (V3) | Access Vector (V2) | Access Complexity (V2) | Authentication (V2) | Confidentiality Impact (V2) |
|----------------|------------------------|-----------------|------------|------------|---------------|--------------------|--------------------|------------------------|---------------------|-----------------------------|
| CVE-2019-1389  | msserver2016datacenter | 1607-14393.3300 | 7.7        | 8.4        | HIGH          | HIGH               | ADJACENT_NETWORK   | LOW                    | SINGLE              | COMPLETE                    |
| CVE-2019-1388  | msserver2016datacenter | 1607-14393.3300 | 7.2        | 7.8        | HIGH          | HIGH               | LOCAL              | LOW                    | NONE                | COMPLETE                    |
| CVE-2019-1384  | msserver2016datacenter | 1607-14393.3300 | 6.5        | 9.9        | MEDIUM        | CRITICAL           | NETWORK            | LOW                    | SINGLE              | PARTIAL                     |
| CVE-2019-1383  | msserver2016datacenter | 1607-14393.3300 | 4.6        | 7.8        | MEDIUM        | HIGH               | LOCAL              | LOW                    | NONE                | PARTIAL                     |
| CVE-2019-1382  | msserver2016datacenter | 1607-14393.3300 | 2.1        | 5.5        | LOW           | MEDIUM             | LOCAL              | LOW                    | NONE                | PARTIAL                     |
| CVE-2019-1381  | msserver2016datacenter | 1607-14393.3300 | 2.1        | 5.5        | LOW           | MEDIUM             | LOCAL              | LOW                    | NONE                | PARTIAL                     |
| CVE-2019-1380  | msserver2016datacenter | 1607-14393.3300 | 4.6        | 7.8        | MEDIUM        | HIGH               | LOCAL              | LOW                    | NONE                | PARTIAL                     |
| CVE-2019-1374  | msserver2016datacenter | 1607-14393.3300 | 4.3        | 5.5        | MEDIUM        | MEDIUM             | NETWORK            | MEDIUM                 | NONE                | PARTIAL                     |
| CVE-2019-1371  | Internet Explorer      | 11.0.155        | 7.6        | 7.5        | HIGH          | HIGH               | NETWORK            | HIGH                   | NONE                | COMPLETE                    |
| CVE-2019-1367  | Internet Explorer      | 11.0.155        | 7.6        | 7.5        | HIGH          | HIGH               | NETWORK            | HIGH                   | NONE                | COMPLETE                    |
| CVE-2019-1357  | Internet Explorer      | 11.0.155        | 4.3        | 4.3        | MEDIUM        | MEDIUM             | NETWORK            | MEDIUM                 | NONE                | NONE                        |
| CVE-2019-1238  | Internet Explorer      | 11.0.155        | 7.1        | 6.4        | HIGH          | MEDIUM             | NETWORK            | HIGH                   | SINGLE              | COMPLETE                    |
| CVE-2019-1192  | Internet Explorer      | 11.0.155        | 4.3        | 4.3        | MEDIUM        | MEDIUM             | NETWORK            | MEDIUM                 | NONE                | PARTIAL                     |
| CVE-2019-11135 | msserver2016datacenter | 1607-14393.3300 | 2.1        | 6.8        | LOW           | MEDIUM             | LOCAL              | LOW                    | NONE                | PARTIAL                     |
| CVE-2019-0719  | msserver2016datacenter | 1607-14393.3300 | 9          | 9.1        | HIGH          | CRITICAL           | NETWORK            | LOW                    | SINGLE              | COMPLETE                    |
| CVE-2019-0712  | msserver2016datacenter | 1607-14393.3300 | 6.8        | 6.8        | MEDIUM        | MEDIUM             | NETWORK            | LOW                    | SINGLE              | NONE                        |
| CVE-2019-0608  | Internet Explorer      | 11.0.155        | 4.3        | 4.3        | MEDIUM        | MEDIUM             | NETWORK            | MEDIUM                 | NONE                | NONE                        |
| CVE-2018-12207 | msserver2016datacenter | 1607-14393.3300 | 4.9        | 6.5        | MEDIUM        | MEDIUM             | LOCAL              | LOW                    | NONE                | NONE                        |

Figure 232: Workload Profile: Vulnerabilities Tab

| CVE ID         | Package Name       | Package Version              | Score (V2) | Score (V3) | Cisco Security Risk Score | Severity (V2) | Base Severity (V3) | Severity (Cisco Security Risk Score) | Access |
|----------------|--------------------|------------------------------|------------|------------|---------------------------|---------------|--------------------|--------------------------------------|--------|
| CVE-2024-25082 | libxml2            | 2.9.10+dfsg-Subuntu0.20.04.6 | 7.5        | 50.1       |                           | HIGH          | HIGH               |                                      |        |
| CVE-2024-24806 | libvpx             | 1.34.2-1ubuntu1.3            | 7.3        | 50.1       |                           | HIGH          | HIGH               |                                      |        |
| CVE-2024-22365 | libpam-modules     | 1.3.1-Subuntu4.6             | 5.5        | 22.3       |                           | MEDIUM        | LOW                |                                      |        |
| CVE-2024-22365 | libpam-modules-bin | 1.3.1-Subuntu4.6             | 5.5        | 22.3       |                           | MEDIUM        | LOW                |                                      |        |
| CVE-2024-22365 | libpam-runtime     | 1.3.1-Subuntu4.6             | 5.5        | 22.3       |                           | MEDIUM        | LOW                |                                      |        |
| CVE-2024-22365 | libpam0g           | 1.3.1-Subuntu4.6             | 5.5        | 22.3       |                           | MEDIUM        | LOW                |                                      |        |
| CVE-2024-22195 | python3-jinja2     | 2.10.1-2                     | 6.1        | 44.6       |                           | MEDIUM        | MODERATE           |                                      |        |
| CVE-2024-0727  | libssl1.1          | 1.1.1f-1ubuntu2.19           | 5.5        | 25.0       |                           | MEDIUM        | MODERATE           |                                      |        |
| CVE-2024-0727  | openssl            | 1.1.1f-1ubuntu2.19           | 5.5        | 25.0       |                           | MEDIUM        | MODERATE           |                                      |        |
| CVE-2024-0567  | libgnutls30        | 3.6.13-2ubuntu1.8            | 7.5        | 50.1       |                           | HIGH          | HIGH               |                                      |        |
| CVE-2024-0553  | libgnutls30        | 3.6.13-2ubuntu1.8            | 7.5        | 50.1       |                           | HIGH          | HIGH               |                                      |        |
| CVE-2023-7104  | libsqlite3-0       | 3.31.1-4ubuntu0.5            | 7.3        | 18.6       |                           | HIGH          | LOW                |                                      |        |
| CVE-2023-6918  | libssh-4           | 0.9.3-2ubuntu2.3             | 5.3        | 41.8       |                           | MEDIUM        | MODERATE           |                                      |        |

## Inventory Filters

The following types of inventory filters can be defined to identify hosts with vulnerable packages:

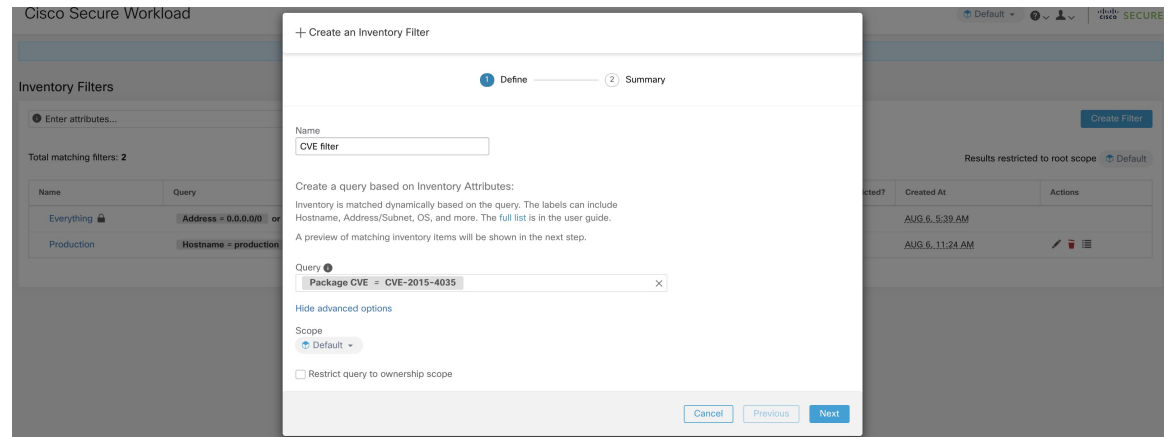
### CVE ID Based Filter

The inventory filter created with CVE ID allows searching hosts affected by a specific CVE.



To search for a host affected by a specific CVE, enter the CVE ID in the format: `CVE-XXXX-XXXX`

**Figure 233: Inventory filter CVE**



The following operations are supported:

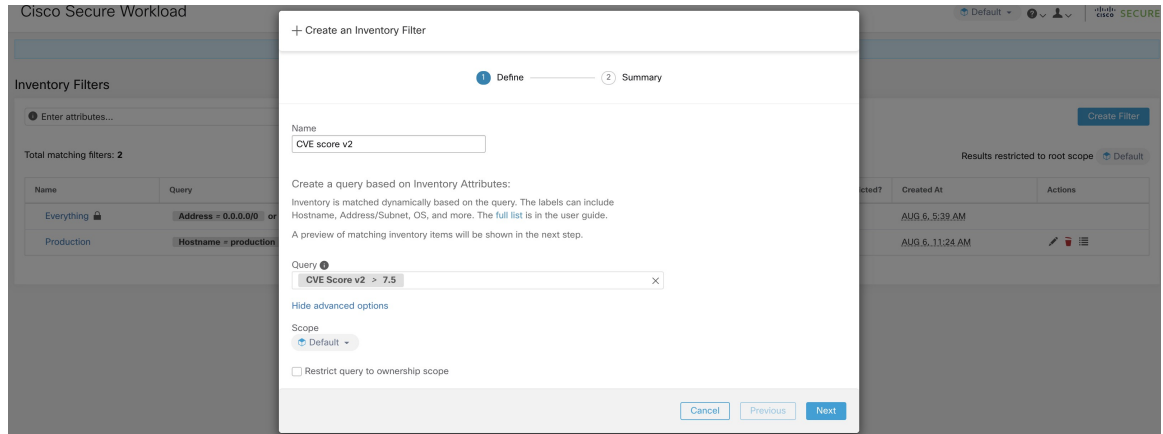
- `=`: Returns hosts with packages affected by a CVE ID.
- `≠`: Returns hosts with packages not affected by a CVE ID.
- **contains**: Returns hosts with packages affected by a CVE present in the input string (entering **cve** returns hosts affected by a CVE).
- **doesn't contain**: Returns hosts with packages not affected by a CVE present in the input string (entering **cve** returns hosts not affected by a CVE).
- **matches**: Returns hosts with packages affected by a CVE that matches the input string.

## Common Vulnerability Scoring System Impact Score Based Filter

The Common Vulnerability Scoring System (CVSS) based filter allows searching for hosts that have CVE with the specified CVSS V2 or CVSS V3 impact score by entering the score in numeric format.

For example, to search for hosts with CVEs of CVSS V2 impact score greater than 7.5, the query is `CVE Score v3 > 7.5`.

Figure 234: Inventory Filter CVSS



The following operations are supported:

- =: Returns hosts which have CVE with the specified CVSS V2 or V3 impact scores.
- ≠: Returns hosts which do not have CVE with the specified CVSS V2 or V3 impact scores.
- >: Returns hosts which have CVE with CVSS V2 or V3 impact scores greater than the specified CVSS V2 or V3 impact scores respectively.
- ≥: Returns hosts which have CVE with CVSS V2 or V3 impact scores greater than or equal to the specified CVSS V2 or V3 impact scores respectively.
- <: Returns hosts which have CVE with CVSS V2 or V3 impact scores less than the specified CVSS V2 or V3 impact scores respectively.
- ≤: Returns hosts which have CVE with CVSS V2 or V3 impact scores less than or equal to the specified CVSS V2 or V3 impact scores respectively.

## CVSS V2 Attributes Based Filters

Inventory filters can be created using access vectors and access complexities to identify vulnerable hosts. The following operations are supported in the filter:

- =: Returns hosts with packages affected by vulnerabilities matching the filter.
- ≠: Returns hosts with packages not affected by vulnerabilities matching the filter.

### Access Vector

Access vector reflects how the vulnerability is exploited. The farther the attacker can get from the vulnerable system, the higher the base score. The table below lists different access vectors with their access requirements:

| Value            | Type of access             |
|------------------|----------------------------|
| LOCAL            | Physical or local (shell). |
| ADJACENT_NETWORK | Broadcast or collision.    |
| NETWORK          | Remotely exploitable.      |

### Access Complexity

This metric measures the complexity in exploiting a vulnerability once the attacker is able to access the target system. The base score is inversely proportional to the access complexity. The different types of access complexities are as follows:

| Value  | Description                                 |
|--------|---------------------------------------------|
| HIGH   | Specialized access conditions exist.        |
| MEDIUM | Access conditions are somewhat specialized. |
| LOW    | Specialized access conditions do not exist. |

## CVSS V3 Attributes Based Filters

Attack vectors, attack complexities, and privilege required to influence the CVSS V3 score can be used in inventory filters. The following operations are supported in the filter:

- =: Returns hosts with packages affected by vulnerabilities matching the filter.
- ≠: Returns hosts with packages not affected by vulnerabilities matching the filter.

### Attack Vector

This metric reflects the context by which vulnerability exploitation is possible. The farther an attacker can get from the vulnerable component, the higher the base score. The table below lists different attack vectors with their access requirements:

| Value            | Type of access                             |
|------------------|--------------------------------------------|
| LOCAL            | Local (keyboard, console) or remote (SSH). |
| PHYSICAL         | Physical access is needed.                 |
| ADJACENT_NETWORK | Broadcast or collision.                    |
| NETWORK          | Remotely exploitable.                      |

### Attack Complexity

This metric describes the conditions that must exist in order to exploit the vulnerability. The base score is greatest for least complex attacks. The different types of access complexities are as follows:

| Value | Description                                                       |
|-------|-------------------------------------------------------------------|
| HIGH  | Significant effort needed in setting up and executing the attack. |
| LOW   | Specialized access conditions do not exist.                       |

### Privileges Required

This metric describes the level of privileges an attacker must possess before successfully exploiting the vulnerability. The base score is highest when privileges aren't needed to carry out an attack. The different values of privilege required are as follows:

| Value | Privileges required                                                     |
|-------|-------------------------------------------------------------------------|
| HIGH  | Privileges providing significant control over the vulnerable component. |
| LOW   | Low privileges that grant access to non-sensitive resources.            |
| NONE  | Privileges aren't needed to carry out an attack.                        |

### Cisco Security Risk Score-Based Filter

The Cisco Security Risk Score based-filter enables you to search for hosts that have CVEs with the specified Cisco Security Risk Score by entering the score in numeric format. For example, to search for hosts with CVEs of Cisco Security Risk Score greater than 67, the query is Cisco Security Risk Score  $\geq$  67.

The severities of the Cisco Security Risk Score ranges are:

- **High** for score range from 67 to 100
- **Medium** for score range from 34 to 66
- **Low** for score range from 0 to 33

Figure 235: Creating Inventory Filter with Cisco Security Risk Score

Create an Inventory Filter

1 Define ————— 2 Summary

Name

Create a query based on Inventory Attributes:  
 Inventory is matched dynamically based on the query. The labels can include Hostname, Address/Subnet, OS, and more. The [full list](#) is in the user guide.  
 A preview of matching inventory items will be shown in the next step.

Query ⓘ

Matching Object Types: Inventory

[Show advanced options](#)

Cancel Previous Next

The following operations are supported in the filter query:

- =: Returns hosts that have CVE with the specified Cisco Security Risk Score.
- ≠: Returns hosts that do not have CVE with the specified Cisco Security Risk Score.
- >: Returns hosts that have CVE with Cisco Security Risk Score greater than the specified Cisco Security Risk Score.
- ≥: Returns hosts that have CVE with Cisco Security Risk Score greater than or equal to the specified Cisco Security Risk Score.
- <: Returns hosts that have CVE with Cisco Security Risk Score less than the specified Cisco Security Risk Score.
- ≤: Returns hosts that have CVE with Cisco Security Risk Score less than or equal to the specified Cisco Security Risk Score.

## Cisco Security Risk Score Attributes-Based Filters

The Cisco Security Risk Score attributes can be used in inventory filters. The following operations are supported in the query filter:

- =: Returns hosts with packages that are affected by vulnerabilities matching the filter.
- ≠: Returns hosts with packages that are not affected by vulnerabilities matching the filter.

**Table 21: Cisco Security Risk Score Attributes with Description**

| Attribute              | Description                                                                                                    |
|------------------------|----------------------------------------------------------------------------------------------------------------|
| Active Internet Breach | Indicates whether CVE is part of Active Internet Breach activity across organisations.                         |
| Easily Exploitable     | Indicates whether CVE has known exploit kits.                                                                  |
| Fix Available          | Indicates whether a fix is available for the CVE.                                                              |
| Malware Exploitable    | Indicates whether CVE can be actively exploited with malware including trojans, worms, ransomware, and others. |
| Popular Target         | Indicates whether CVE is detected in high volume by other Cisco Vulnerability Management clients.              |
| Predicted Exploitable  | Indicates whether CVE is expected to have an Active Internet Breach in the future.                             |

The values of the attributes are booleans; enter either true or false to filter the CVEs based on the attributes.

## Malicious Inventory-Based Filter

By default, the feature to identify well-known malicious IP addresses is disabled. After you enable the feature, you can identify all the well-known malicious IPv4 addresses using the **Malicious inventories** filter. Use the read-only filter to create and enforce policies on workloads to block the communication from workloads to well-known malicious IPv4 addresses.

By default, the query of the **Malicious inventories** filter is set to `* Is_malicious = true`.

For more information about the following topics, refer to the corresponding sections:

- Enabling detection of malicious consumer and provider IP addresses, see [Visibility of Well-Known Malicious IPv4 Addresses, on page 645](#).
- To create microsegmentation policies, see [Create and Discover Policies, on page 427](#).
- To enforce policies on your workloads, see [Enforce Policies, on page 538](#).

## Service Profile

Secure Workload provides visibility of all Kubernetes services and other Load Balancers ingested through an external orchestrator. Service profile page shows the details for a given service.



**Note** Service profile page is linked from various places. One of the ways to see a service profile is to perform a search for service as described in search

From the results of search, click on a Service Name under the Services tab to go to its profile. The following information is available for the service:

### Header

Header consists of:

- **Orchestrator Name:** Name of the external orchestrator which reported this service.
- **Orchestrator Type:** Type of the external orchestrator.
- **Namespace:** Namespace of the service.
- **Service Type:** Type of the service. Possible values include ClusterIP, Node, Port, and LoadBalancer.

### IP and Ports

This table lists all the possible IP and port combinations through which this service is accessible. For services of type NodePort, this table shows both ClusterIP:Port and NodeIp:NodePort association.

### User Labels

The list of user uploaded and orchestrator system generated labels for this service.

### Scopes

List of scopes that the pod belongs to.

## Pod Profile

Secure Workload provides visibility of all Kubernetes pods ingested through a Kubernetes external orchestrator. Pod profile page shows the details for a given pod.



---

**Note** Pod profile page is linked from various places. One of the ways to see a pod profile is to perform a search for pod as described in search

---

From the results of search, click on a Pod Name under the Pods tab to go to its profile. The following information is available for the pod:

### Header

Header consists of:

- **Orchestrator Name:** Name of the external orchestrator which reported this pod.
- **Orchestrator Type:** Type of the external orchestrator.
- **Namespace:** Namespace of the pod.
- **IP Address:** Pod's IP Address.

**User Labels**

The list of user uploaded and orchestrator system generated labels for this pod.

**Scopes**

List of scopes that the service belongs to.

## Container Vulnerability Scanning

To maintain health and identify potential security weaknesses, we recommend scanning the Kubernetes pods regularly.

**Prerequisites**

- Ensure that a Kubernetes cluster is on board.
- Install the CSW Kubernetes daemonset agent as part of the Kubernetes cluster. For more information, see [Installing Kubernetes or OpenShift Agents for Deep Visibility and Enforcement](#).

**Procedure**


---

**Step 1** Navigate to **Manage > Workloads > Kubernetes**.

**Note** The **Clusters** tab displays a list of all on-boarded clusters along with the associated inventory, such as services and pods.

**Step 2** Click **Pod Vulnerability Scanning**.

**Step 3** To start the scan, enable the toggle under **Actions**. By default, the toggle is disabled.

**Step 4** Click the edit icon to modify the query and select a subset of pods running on the cluster.

**Note**

- A pod query is populated by default to scan all pod inventories in the cluster. However, you can edit pod queries to select the pods to scan.
- Currently, scanning Windows container images is not supported.

**Step 5** Expand a cluster to view the **Health Status Summary**.

- Click on a Kubernetes Node Name to view the Workload Profile.
- Enable the toggle to automatically download additional information to the host so that the scanner can execute.



Figure 236: Pod Vulnerability Scanning

**Kubernetes**

Clusters **Pod Vulnerability Scanning**

To secure your Kubernetes workloads and to keep clusters healthy, regularly scan clusters for any known vulnerabilities and to identify potential security

**Scanners**

| Cluster Name ↑↓         | Pod Queries ↑↓    | Health |
|-------------------------|-------------------|--------|
| ▼ Kubernetes Cluster #1 | Scanning all pods |        |

Health Status Summary

| Kubernetes Node Name | Last Reported                |
|----------------------|------------------------------|
| node-1               | Sep 5 2023 03:43:57 pm (PDT) |

Rows per page 5 < 1 >

**Registry List**

Enter attributes... × Filter

| Registry URL ↑↓   | Registry Type ↑↓ | Kubernetes Cluster ↑↓ | Last Scanned ↑↓               | C |
|-------------------|------------------|-----------------------|-------------------------------|---|
| 192.168.51.1:5000 | Other            | Kubernetes Cluster #1 | Aug 30 2023 03:29:18 pm (PDT) |   |
| 192.168.51.1:5001 | Other            | Kubernetes Cluster #1 | Aug 30 2023 02:59:18 pm (PDT) |   |
| docker.io         | Other            | Kubernetes Cluster #1 | Aug 30 2023 03:43:59 pm (PDT) |   |
| quay.io           | Other            | Kubernetes Cluster #1 | Aug 30 2023 03:58:55 pm (PDT) |   |
| registry.k8s.io   | Other            | Kubernetes Cluster #1 | Aug 30 2023 02:43:54 pm (PDT) |   |

Rows per page 5 < 1 >

**Step 6** Verify the connection status and enter the credentials, if necessary. The **Registry List** displays all detected registries.

**Note** Credentials vary based on the registry type.

| Registry Type | Credentials                        |
|---------------|------------------------------------|
| Azure         | Tenant ID, Client ID, Secret Key   |
| AWS           | Access Key, Secret Key             |
| GCP           | Service account key in JSON format |
| Other         | Username, Password                 |

### Troubleshooting

Follow these steps to ensure a successful connection:

- a. The scanner pod is able to connect to the registry.
  - b. The required network policies are in place.
  - c. Credentials are entered, if necessary.
-



## CHAPTER 6

# Manage Policies Lifecycle in Secure Workload

- [Segmentation Policy Basics, on page 417](#)
- [Use Workspaces to Manage Policies, on page 418](#)
- [About Policies, on page 425](#)
- [Create and Discover Policies, on page 427](#)
- [Grouping Workloads: Clusters and Inventory Filters, on page 487](#)
- [Address Policy Complexities, on page 498](#)
- [About Deleting Policies, on page 520](#)
- [Review and Analyze Policies, on page 520](#)
- [Enforce Policies, on page 538](#)
- [Modify Enforced Policies, on page 551](#)
- [About Policy Versions \(v\\* and p\\*\), on page 553](#)
- [Conversations, on page 559](#)
- [Automated Load Balancer Config for Automatic Policy Discovery \(F5 Only\), on page 566](#)
- [Policies Publisher, on page 571](#)

## Segmentation Policy Basics

The purpose of segmentation and microsegmentation policies is to allow only the traffic your organization needs to conduct business, and block all other traffic. The goal is to reduce your network's attack surface without disrupting business operations.

Secure Workload segmentation policies allow or block traffic based on its source, destination, port, protocol, and a few other attributes that are typically platform-specific.

You can create some policies manually, and use Secure Workload's powerful automatic policy discovery feature to generate other policies based on existing network traffic.

You can review, refine, and analyze your policies, then enforce them when you are confident that they allow only the traffic that your organization needs.



---

**Important** Microsegmentation essentially creates a firewall around each workload.

Therefore, for traffic to pass between each consumer-provider pair, both ends of the conversation must allow the conversation to happen: The consumer and the provider must each have a policy that allows the traffic.

---



---

**Note** The terms *firewall rule*, *edge*, and *cluster edge* are sometimes used to mean "policy."

---

## Use Workspaces to Manage Policies

Workspaces (formerly called “Application workspaces” or “Applications”) are where you work with and manage policies.

You can perform all policy-related activities for a particular scope, such as creating, analyzing, and enforcing policies, in the workspace or workspaces associated with that scope.

Each workspace provides an isolated environment, allowing experimentation with no effect on other workspaces.

### Controlling User Access to Workspaces

Workspaces are meant to be used by multiple users from the same team as shared documents.

To control access to a workspace, assign user roles for the scope associated with the workspace. For more information, see the Roles section.

## Working with Policies: Navigating to the Workspaces Page

- **To work with policies, or to view existing application workspaces or create new ones:**

Choose **Defend** > **Segmentation** from the navigation bar at the left side of the window.

- **To view a particular workspace:**

In the list of scopes at the left side of the Workspaces page, navigate to the scope associated with the workspace, then click the workspace. The current active workspace is highlighted in the list.

- **If you are looking at a workspace and want to return to the list of workspaces:**

Click the **Workspaces** link near the left side of the page you are looking at.

Figure 237: Workspace Management Page

The screenshot shows the 'Segmentation' page with the 'Workspaces' tab selected. On the left, a list of workspaces is shown for the 'Furong:jumphost' scope. The 'Furong:jumphost' workspace is highlighted as the primary workspace. The main content area displays a table of policies for this workspace:

| Type         | Version | Absolute Policies | Default Policies | Catch All |
|--------------|---------|-------------------|------------------|-----------|
| Enforced     | p10     | 1                 | 10               | ALLOW     |
| Analyzed     | p10     | 1                 | 10               | ALLOW     |
| Latest Draft | v3      | 1                 | 10               | ALLOW     |

Below the table, the 'Parent Scopes' section shows a table with columns for Scope, Primary Workspace, Analysis, and Enforcement:

| Scope  | Primary Workspace | Analysis                                              | Enforcement |
|--------|-------------------|-------------------------------------------------------|-------------|
| Furong | test              | Version: p6<br>Policies: 4<br>Catch-all Action: ALLOW | Disabled    |

## Create a Workspace

To create policies for a scope, first create a workspace for that scope.

To create a workspace:

1. From the navigation menu on the left side of the window, choose **Defend** > **Segmentation**.
2. In the scope listing on the left side of the page, search for or scroll to the scope for which you want to create policies.
3. Hover over the scope until you see a blue plus sign, then click it.
4. Complete the form and click **Create** when done.

If a workspace exists for the scope, any additional workspaces is created as a secondary workspaces.

## Primary and Secondary Workspaces

For each scope, you can create one Primary workspace and multiple secondary workspaces.

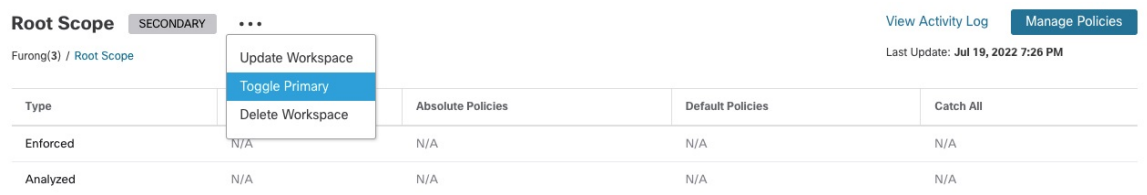
Only a primary workspace can be enforced. Other features that are available only for primary workspaces include the ability to manage policies in which consumer and provider reside in different scopes; live policy analysis; compliance reporting; and collaborative security policy definition.

Use secondary workspaces to experiment with policies when you want to preserve the existing policies in the primary workspace.

### To change a workspace to primary or secondary:

You can switch a workspace from primary to secondary and conversely at any time by clicking the menu icon next to the workspace name at the top of the page and selecting **Toggle Primary**.

Figure 238: Switching a Workspace Between Primary and Secondary



## Rename a Workspace

To rename a workspace:

Click **...** beside the workspace type (Primary or Secondary) shown near the top of the page and choose **Update Workspace**.

## View Workloads in a Scope

In any workspace, click the **Matching Inventories** tab.

## Search Within a Workspace

To search within a workspace for workloads, clusters, or policies:

1. Select **Defend > Segmentation**.
2. From the list of scopes on the left, click the scope and workspace of interest.
3. Click **Manage Policies**.
4. Click the magnifying glass.
5. Enter search criteria.

### Search Criteria

Multiple criteria are treated as logical AND.

For IP addresses and numeric values:

- Indicate logical OR using a comma: 'port: 80,443'.
- Range queries are also supported for number values: 'port: 3000-3999'.

| Filters            | Description                                                                 |
|--------------------|-----------------------------------------------------------------------------|
| <b>Name</b>        | Enter a cluster or workload name. Performs case-sensitive substring search. |
| <b>Description</b> | Searches cluster descriptions.                                              |
| <b>Approved</b>    | Matches approved clusters using the values 'true' or 'false'.               |

| <b>Filters</b>          | <b>Description</b>                                                                                                                      |
|-------------------------|-----------------------------------------------------------------------------------------------------------------------------------------|
| <b>Address</b>          | Enter a subnet or IP address using CIDR notation (for example, 10.11.12.0/24). Matches workloads or clusters which overlap this subnet. |
| <b>Supernet</b>         | Enter a subnet using CIDR notation (for example, 10.11.12.0/24) to match clusters whose workloads are fully contained in this subnet.   |
| <b>Process</b>          | Searches workload processes using case-sensitive substring search.                                                                      |
| <b>Process UID</b>      | Searches workload process usernames.                                                                                                    |
| <b>Port</b>             | Searches both workload provider port and policy port.                                                                                   |
| <b>Protocol</b>         | Searches both workload provider protocol and policy protocol.                                                                           |
| <b>Consumer Name</b>    | Matches a policy's consumer cluster name. Performs a case-sensitive substring match.                                                    |
| <b>Provider Name</b>    | Matches a policy's provider cluster name. Performs a case-sensitive substring match.                                                    |
| <b>Consumer Address</b> | Matches policies whose consumer address overlaps with the provided IP or subnet.                                                        |
| <b>Provider Address</b> | Matches policies whose provider address overlaps with the provided IP or subnet.                                                        |

### Search Example

The screenshot shows a search interface with a search bar containing the query "Address = 0.0.0.0/0". Below the search bar, there is a "Search" button and the text "over workloads, clusters.". The results section shows "Found 81 results page 1". Two cluster results are displayed:

- Cluster: OTHER: rcdn9-dci13n-g
- Description: [edit icon]
- View Cluster Details
- Workloads ?
- IP Addresses ?
- Neighbors 13
- Subnets 2

The second cluster result is partially visible:

- Cluster: OTHER: rtp1-dcm02n-b
- Description: [edit icon]

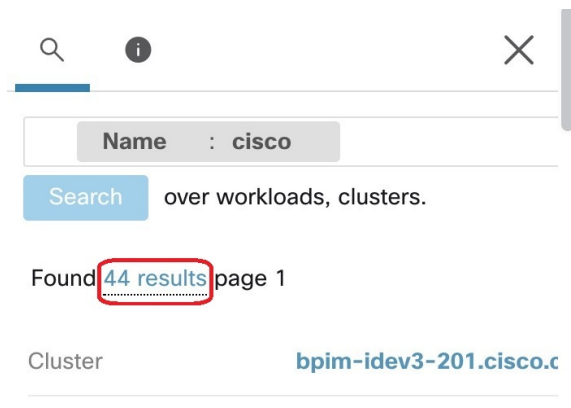
### Filtering Search Results by a Specific Type

Search results may include multiple types of objects, for example workloads and clusters.

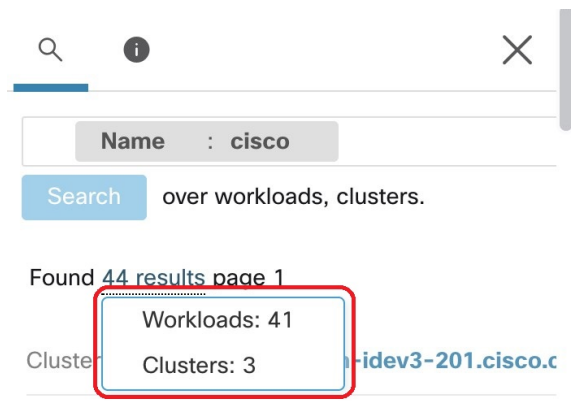
To filter search results by a specific type:

1. Click the result total:





2. Select the type from the dropdown:



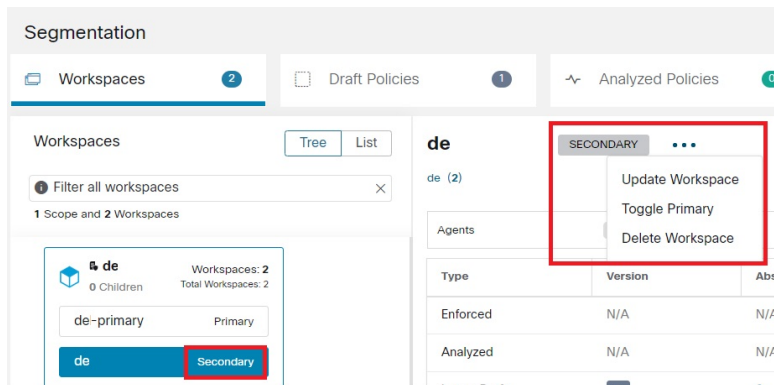
3. A type filter will be added and the search will be rerun.

## Deleting Workspaces

Only secondary (nonprimary) workspaces can be deleted. To switch a workspace to secondary, see [Primary and Secondary Workspaces, on page 419](#).

To delete a workspace:

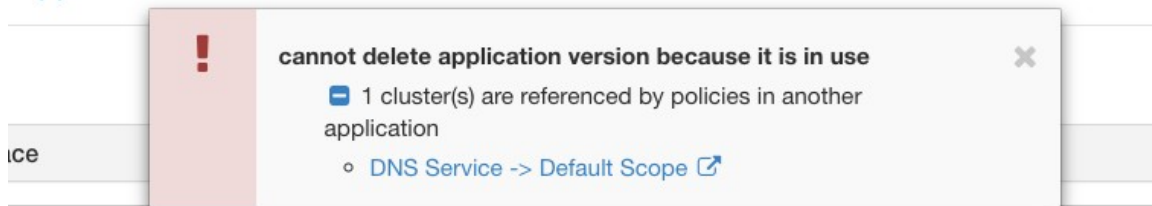
1. Choose **Defend > Segmentation**.
2. In the list of scopes at the left side of the page, navigate to the scope containing the workspace to delete and click it.
3. Click the workspace to delete.
4. Click **...** beside **Secondary** and choose **Delete Workspace**.



If a workload or cluster in a workspace is referenced by a policy in another workspace as a result of a Provided Service, the dependent workspace cannot be deleted, and a list of the dependencies will be returned. This information can be used to fix the dependency.

Figure 239: List of Items Preventing the Deletion of the Workspace

### Applications



In rare conditions there may be a cross dependency where Workspace A depends on a cluster in Workspace B and a Workspace B depends on a cluster in Workspace A. In this case, the individual policies or published policy versions (p\*) must be deleted. The “delete restrictions” error provides links to all the policies so this can be accomplished.

To delete p\* versions, see [View, Compare, and Manage Analyzed Policy Versions, on page 536](#) or [View, Compare, and Manage Enforced Policy Versions, on page 551](#).

# About Policies

## Policy Attributes

Table 22: Policy Properties

| Security Policy Property              | Description                                                                                                                                                                                                                                                                                                                |
|---------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Scope for which the policy is defined | A policy generally affects only workloads that are members of the scope associated with the workspace in which the policy is defined.<br><br>(However, see also the topics under <a href="#">Address Policy Complexities, on page 498.</a> )<br><br>For more information, see <a href="#">Policy Example, on page 427.</a> |
| Consumer                              | The client of a service or the initiator of a connection.<br><br>Any scope, cluster, or inventory filter can be used as the consumer in a policy.<br><br>See important information in <a href="#">About Consumer and Provider in Policies, on page 426.</a>                                                                |
| Provider                              | The server or the recipient of a connection.<br><br>Any scope, cluster, or inventory filter can be used as the provider in a policy.<br><br>See important information in <a href="#">About Consumer and Provider in Policies, on page 426.</a>                                                                             |
| Protocols and Ports                   | The server (listening) port and IP protocol of the service made available by the provider that should be permitted or blocked.                                                                                                                                                                                             |
| Action                                | ALLOW or DENY: Whether to allow or drop traffic from consumer to provider on the given service port/protocol.                                                                                                                                                                                                              |
| Rank and Priority                     | For more information the rank and priority of policies in a workspace, see <a href="#">Policy Rank: Absolute, Default, and Catch-All, on page 425.</a>                                                                                                                                                                     |

## Policy Rank: Absolute, Default, and Catch-All

Policy rank determines whether a policy is overridden by a more specific policy lower in the priority list (or in a scope lower in the scope tree). The lowest priority policy in every scope is always the Catch-all rule.

| Policy Rank     | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
|-----------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Absolute</b> | Absolute policies take effect even if they contradict application-specific policies lower in the policy list (and thus, lower priority) or in scopes lower in the scope tree. Generally, use Absolute policies to enforce best practices, protect different zones, or quarantine-specific workloads. For example, use absolute policies to control traffic to DNS or NTP servers, or to meet regulatory requirements.<br><br>Absolute policies are listed above default policies in the policy priority list. |

| Policy Rank      | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
|------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Default</b>   | <p>Default policies can be overridden by policies lower in the policy list or in scopes lower in the scope tree. Generally, fine-grained policies are Default policies.</p> <p>Default policies are listed below absolute policies in the policy priority list.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| <b>Catch-All</b> | <p>Each workspace has a catch-all policy that handles traffic in each direction that does not match any explicitly specified policies in the workspace. The catch-all action can be Allow or Deny.</p> <p>In general, set the Catch-All policy as follows:</p> <ul style="list-style-type: none"> <li>• <b>Allow</b> traffic in scopes higher in the scope tree, so that policies in scopes lower in the tree can evaluate the traffic.</li> <li>• <b>Deny</b> traffic at the most specific leaf at the bottom of the scope tree.</li> </ul> <p>This gives policies in all scopes in the tree the opportunity to match the traffic, while blocking traffic that does not match any policy in any scope.</p> <p>The catch-all rule is applied to all interfaces on each workload in the workspace.</p> |

## Policy Inheritance and the Scope Tree

Because your workloads are organized into a hierarchical scope tree, you can create general policies once in a scope at or near the top of the tree, and the policies can optionally apply to all workloads in all scopes below that scope in the tree.

You specify whether the general policies can be overridden by more specific policies lower in the tree.

See [Policy Rank: Absolute, Default, and Catch-All, on page 425](#).

## About Consumer and Provider in Policies

The consumer and provider that is specified in a policy serve the following purposes:

- They specify the workloads or Secure Workload agents that receive policy or firewall rules.
- They specify the set of IP addresses to which the firewall rules that are installed on the workloads apply.

If a host has multiple interfaces (IP addresses), policies apply to all interfaces.



### Important

The above is the default behavior of how firewall rules are programmed on the workloads. If the IP addresses specified in the firewall rules differ from the IP addresses of the workloads that the policy is installed to, you need to separate the two purposes of consumers and providers in a policy. See [Effective Consumer or Effective Provider, on page 517](#).

## Policy Example

The following example policy illustrates the importance of the scope in which a policy is defined, the impact of policy inheritance, and the use of inventory filters to create precise policies or policies that apply to workloads in multiple scopes.

Consider the following example involving three scopes:

- **Apps**  
and its child scopes
  - **Apps : HR** and
  - **Apps : Commerce**

In addition, the inventory filters PRODUCTION and NON-PRODUCTION specify production and nonproduction hosts, respectively. (You can define an inventory filter to apply to hosts within a scope or across scopes.)

Assume that the following policy is defined in the **Apps** scope:

```
DENY PRODUCTION -> NON-PRODUCTION on TCP port 8000 (Absolute)
```

Since this policy is an absolute policy that is defined in the primary workspace under the **Apps** scope, it affects all PRODUCTION/NONPRODUCTION hosts that are members of the **Apps** scope, including members of its descendant scopes (hosts that belong to the **Apps : HR** and **Apps : Commerce** scopes).

Now consider the case where the exact same policy is defined under the workspace that is associated with the **Apps : HR** scope. In this scenario, the policy can only affect PRODUCTION/NONPRODUCTION hosts that are members of the **Apps : HR** scope. More precisely, this policy results in inbound rules on NONPRODUCTION HR hosts (if any) denying connections on TCP port 8000 from **any** PRODUCTION host, and outbound rules on PRODUCTION HR hosts (if any) dropping connection requests to **any** NONPRODUCTION host.

## Create and Discover Policies

### Best Practices for Creating Policies

- For an overview of the entire segmentation process, see [Get Started with Segmentation and Microsegmentation, on page 2](#) and subtopics.
- Manually create policies that apply broadly across your network.

For example, block unwanted traffic to your workloads from outside your network, or quarantine vulnerable hosts.

- Create manual policies in scopes at or near the top of your scope tree.

For example, to block all traffic from outside your network to every host in your network, put the policy into the scope at the top of the tree.

- If you want to be able to override the general policy for some workloads (for example, following the example above, you want to block general access from outside your network but you want some

workloads to be accessible from outside the network), create the high-level policies as Default policies. Then create specific policies for the applicable workloads.

- Consider using templates to speed policy creation.
- See [Manually Create Policies, on page 428](#), [Policies for Specific Purposes, on page 430](#), and [Policy Templates, on page 432](#).

- (Optional) Initially, automatically discover policies at a scope near the top of your tree, for all scopes in a branch of the tree, to create coarse policies that allow all existing traffic and limit future unwanted traffic. You can then build granular policies that protects your network from unnecessary or unwanted traffic.

See [Discover Policies for One Scope or for a Branch of the Scope Tree, on page 439](#) and [Discover Policies Automatically, on page 436](#) for information.

- When you are ready to discover more granular policies, automatically discover policies for scopes at or near the bottom of your scope tree, especially in the scopes for individual applications.

See [Discover Policies for One Scope or for a Branch of the Scope Tree, on page 439](#) and [Discover Policies Automatically, on page 436](#) for information.

- Be sure you have policies that address uncommon or infrequent activities and scenarios, such as failover, restoration from backup, once-yearly activities, and so on.
- After you have identified and allowed the traffic that your applications require, then look for any traffic that shouldn't be occurring and block such instances.

Look first at traffic to and from your most sensitive applications.

For example, if you see traffic from your customer-facing web app to your top-secret research and development app's database, you want to investigate.

- Work with your colleagues to ensure that the correct policies are applied to the correct workloads.
- Initially, when you enforce policies, consider setting the catch-all to Allow. Then, monitor traffic to see what matches the catch-all rule. When no necessary traffic is matching the catch-all rule, you can set the catch-all to Deny.

## Manually Create Policies

Typically, you can manually create policies that apply broadly across your network.

For example, you can manually create policies to:

- Allow access from all internal workloads to your NTP, DNS, Active Directory, or vulnerability scanning servers.
- Deny access from all hosts outside your organization to hosts inside your network unless explicitly permitted.
- Quarantine vulnerable workloads.

You can create absolute policies that cannot be overridden by more granularly applied policies, and default policies that can be overridden if a more specific policy exists.

You can create manual policies for scopes nearer the top of your tree.

### Before you begin

- (Optional) Consider using one of the templates available from **Defend > Policy Templates**.
- (Optional) If you know you have a set of workloads that receive the same policies, use an inventory filter to group them so you can easily apply policies to the set. The inventory filter can apply to only one scope, or to workloads in any scope. See [Create an Inventory Filter, on page 378](#).
- Make sure that the workloads in this scope are the workloads that you expect to be in this scope. See [View Workloads in a Scope, on page 420](#).

### Procedure

---

- Step 1** Click **Defend > Segmentation**.
- Step 2** In the list on the left, search for or navigate to the scope in which you want to create the policy.
- Step 3** Click the scope and workspace in which you want to create the policy.
- If you haven't yet created the workspace for this scope, see [Create a Workspace, on page 419](#).
- Step 4** Click **Manage Policies**.
- Step 5** Click the **Policies** tab if it is not already selected.
- Step 6** Click **Add Policy**.
- If you don't see an Add Policy button, see [If the Add Policy Button Is Not Available, on page 429](#).
- Step 7** Enter information.
- For information about the **Absolute** checkbox, see [Policy Rank: Absolute, Default, and Catch-All, on page 425](#). Generally, if you are creating policies that you don't expect exceptions for, enable this checkbox.
  - **Priority** sets the order of the policy in the list. For more information about setting policy order, see [Policy Priorities, on page 498](#) and subtopics. (You can set policy order later.)
  - Consumer and provider can be entire scope, or, if you have created groups of workloads using inventory filters (or less optimally, clusters in the same workspace), you can choose those.

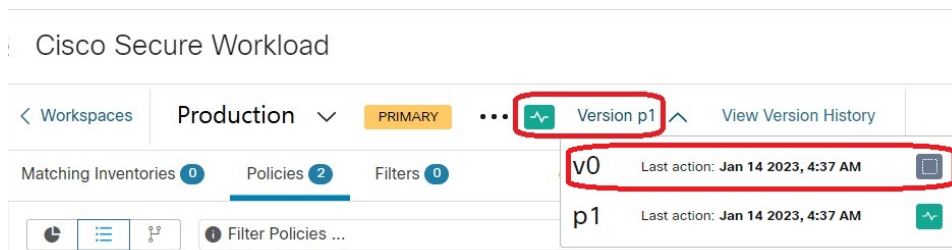
---

### What to do next

Make sure the **Catch-all** action is appropriate for the workspace. See [Policy Rank: Absolute, Default, and Catch-All, on page 425](#).

## If the Add Policy Button Is Not Available

If you are trying to create a policy and the **Add Policy** button is not available, click the version showing at the top of the page and choose the latest "v" version, which is indicated with a gray square:



## Policies for Specific Purposes

### Create InfoSec Policies to Block Traffic from Outside Your Network

Use this procedure to quickly create a complete set of policies to control traffic entering your network from outside the network. The default set of policies allows only traffic using common ports and protocols and denies all other traffic. You can modify the default policy set to meet your needs.

#### Before you begin

Use this procedure if the following criteria are met:

- Your scope tree has a scope that is named **Internal** immediately below the root scope.  
This scope's members include, or will include, subnets encompassing all workloads on your internal network.
- The Internal scope does not yet have any policies defined in it.



**Note** Alternatively, you can use the **InfoSec** template available from **Defend > Policy Templates** to accomplish this with a few additional steps.

#### Procedure

- Step 1** Choose **Defend > Segmentation**,
- Step 2** Click the **Internal** scope and click the primary workspace.  
If the Primary workspace does not yet exist, click the + button to create it.
- Step 3** Click **Manage Policies**.
- Step 4** Click **Add InfoSec Policies**.
- Step 5** Verify that all the policies in the list, including protocols and ports, are policies you want and delete and modify policies as desired.
- Step 6** Click **Create**.



### What to do next

(Optional) Add any additional policies to your Internal scope, such as policies that allow certain external traffic to specific workloads.

Place any specific policies below the more general policies in the list.

## Create Policies to Address Immediate Threats

If you must address an immediate threat, you can manually add a narrowly focused Absolute policy to a scope at or near the top of your scope tree, then enforce the primary workspace for that scope.

After you remediate the threat, you can remove that policy and reenforce the workspace.

## Create a Policy to Quarantine Vulnerable Workloads

You can:

- Create policies in advance, to automatically quarantine workloads with specific known vulnerabilities or a vulnerability severity threshold you specify.
- Create policies, to immediately quarantine workloads with detected known vulnerabilities that you deem sufficiently problematic.

This topic outlines the process for doing either.

### Before you begin

Look at the [View Vulnerability Dashboard, on page 697](#) to see what policies are required.

### Procedure

- 
- Step 1** Create an inventory filter that defines the vulnerabilities or the vulnerability severity threshold that you want to quarantine:
- a) From the navigation bar at the left of the window, choose **Organize > Inventory Filters**.
  - b) Click **Create Inventory Filter**
  - c) Click the **(i)** button beside **Query** and enter **CVE** to see the relevant filter options.
  - d) Enter filter criteria that determine which workloads you wish to quarantine.
  - e) Be sure **Restrict query to ownership scope** is NOT selected.
- Step 2** Create a policy to quarantine affected workloads:
- For general instructions, see [Manually Create Policies, on page 428](#).
- Recommendations:
- Create the policy in your **Internal** or other scope near the top of your scope tree.
  - The policy should be an Absolute policy unless you want to allow exceptions. Be sure to create policies to address any exceptions.
  - Create separate policies for consumer and provider.
  - Set the priority of each policy to a low number so it will be hit before other policies in the list.
  - Set the action to **Deny**.

**Step 3** Review, analyze, and enforce the policy or policies.

---

#### What to do next

Create an alert so you are notified when traffic hits this policy so you can remediate the problem and restore traffic to the vulnerable workload. See [Configure Alerts, on page 647](#).

## Policy Templates

Policy Templates are used to apply similar sets of policies to multiple workspaces.

Secure Workload includes some predefined templates, and you can create your own templates.

Policy templates require the scope owner capability on the root scope.

### System-Defined Policy Templates

To view available policy templates, navigate to **Defend > Policy Templates**.

To use a policy template, see [Applying a Template, on page 435](#).

To modify a system-defined template, download the JSON file, edit it, then upload it.

### Create Custom Policy Templates

#### JSON Schema for Policy Templates

The policy template JSON schema is designed to mimic the schema of [Export a Workspace](#). You can create a set of policies in a workspace, export it as JSON, modify the JSON, then import as a policy template.

| Attribute         | Type                    | Description                                                                 |
|-------------------|-------------------------|-----------------------------------------------------------------------------|
| name              | string                  | (optional) Used as the name of the template during import.                  |
| description       | string                  | (optional) Template description that is displayed during the apply process. |
| parameters        | parameters object       | Template parameters, see below.                                             |
| absolute_policies | array of policy objects | (optional) Array of absolute policies.                                      |
| default_policies  | array of policy objects | (required) Array of default policies, can be empty.                         |

#### Parameters Object

The parameters object is optional but can be used to dynamically define filters as parameters to the template. The parameters are referenced using the `consumer_filter_ref` or `provider_filter_ref` policy attributes.

The keys of the parameters object are the reference names. The values are an object with a required "type": "Filter" and an optional description. An example Parameters object is shown below:

```
{
 "parameters": {
 "HTTP Consumer": {
 "type": "Filter",
 "description": "Consumer of the HTTP and HTTPS service"
 },
 "HTTP Provider": {
 "type": "Filter",
 "description": "Provider of the HTTP and HTTPS service"
 }
 }
}
```

The parameters can be referenced in the policy objects, for example: "consumer\_filter\_ref": "HTTP Consumer" or "provider\_filter\_ref": "HTTP Provider".

### Special Parameter References

A few special references automatically map to a filter and do not need to be defined as parameters.

| Ref             | Description                                                                    |
|-----------------|--------------------------------------------------------------------------------|
| _workspaceScope | Resolves to the scope of the workspace to which the template is being applied. |
| _rootScope      | Resolves to the root/top level scope.                                          |

### Policy Object

To maintain compatibility with the workspace export JSON, the policy object contains multiple keys for consumers and providers. They are resolved as follows:

```
if *_filter_ref is defined
 use the filter resolved by that parameter
else if *_filter_id is defined
 use the filter referenced by that id
else if *_filter_name is defined
 use the filter that has that name
else
 use the workspace scope.
```

If a filter cannot be resolved as defined above, an error is returned both at the time of application and at the time of upload.

| Attribute           | Type    | Description                                                      |
|---------------------|---------|------------------------------------------------------------------|
| action              | string  | (optional) Action of the policy, ALLOW, or DENY (default ALLOW). |
| priority            | integer | (optional) The priority of the policy (default 100).             |
| consumer_filter_ref | string  | Reference to a parameter.                                        |

| Attribute            | Type              | Description                                                                             |
|----------------------|-------------------|-----------------------------------------------------------------------------------------|
| consumer_filter_name | string            | Reference to a filter by name.                                                          |
| consumer_filter_id   | string            | ID of a defined Scope or Inventory Filter.                                              |
| provider_filter_ref  | string            | Reference to a parameter.                                                               |
| provider_filter_name | string            | Reference to a filter by name.                                                          |
| provider_filter_id   | string            | ID of a defined Scope or Inventory Filter.                                              |
| l4_params            | array of l4params | List of allowed ports and protocols.                                                    |
| Attribute            | Type              | Description                                                                             |
| proto                | integer           | Protocol integer value (NULL means all protocols).                                      |
| port                 | integer           | Inclusive range of ports, for example, [80, 80] or [5000, 6000] (NULL means all ports). |

#### L4param object

| Attribute | Type    | Description                                                                             |
|-----------|---------|-----------------------------------------------------------------------------------------|
| proto     | integer | Protocol integer value (NULL means all protocols).                                      |
| port      | integer | Inclusive range of ports, for example, [80, 80] or [5000, 6000] (NULL means all ports). |

#### Template Sample

```
{
 "name": "Allow HTTP/HTTPS and SSH",
 "parameters": {
 "HTTP Consumer": {
 "type": "Filter",
 "description": "Consumer of the HTTP and HTTPS service"
 },
 "HTTP Provider": {
 "type": "Filter",
 "description": "Provider of the HTTP and HTTPS service"
 }
 },
 "default_policies": [
 {
 "action": "ALLOW",
 "priority": 100,
 "consumer_filter_ref": "__rootScope",
 "provider_filter_ref": "__workspaceScope",
 }
]
}
```

```

 "l4_params": [
 { "proto": 6, "port": [22, 22] },
]
 },
 {
 "action": "ALLOW",
 "priority": 100,
 "consumer_filter_ref": "HTTP Consumer",
 "provider_filter_ref": "HTTP Provider",
 "l4_params": [
 { "proto": 6, "port": [80, 80] },
 { "proto": 6, "port": [443, 443] }
]
 }
]
}

```

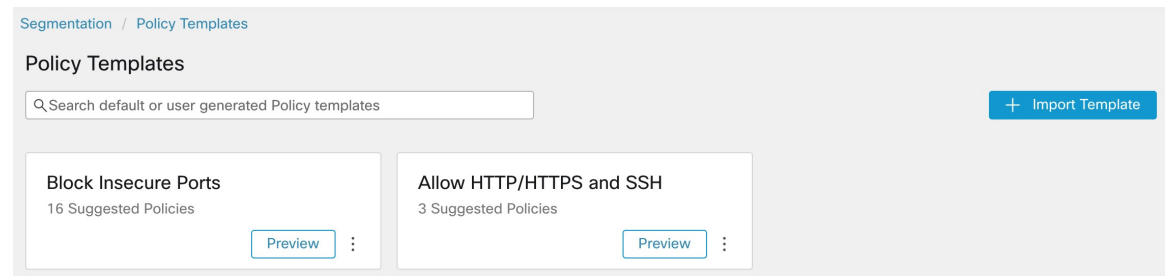
## Template Import

Policy Templates shown on the Policy Templates page that can be accessed from the main Segmentation page. This is where templates can be imported/uploaded using the “Import Template” button.

Templates are validated for correctness when they are uploaded. A helpful list of errors is provided to debug any issues.

Once a template is uploaded, it can be applied, downloaded, or have its name and description updated.

**Figure 240: Display of Available Templates**



## Applying a Template

Applying a template to a workspace takes several steps:

1. Select a template to preview.
2. Select a workspace to apply the template to.
3. Fill in parameters, if necessary.
4. Review the policies.
5. Apply the policies.

The policies will be added to the latest version of the selected workspace. Policies created via a template can be filtered using the `From Template? = true` filter.

Figure 241: Applying a Policy Template

Segmentation / Policy Templates / Allow HTTP/HTTPS and SSH

Allow HTTP/HTTPS and SSH Apply Policies

Select workspace

Default  
Primary Workspace Default X

Parameters

HTTP Consumer ?  
Select a scope

HTTP Provider ?  
My HTTP/HTTPS Service X

Policies

3 Suggested Policies

| Rank ↑↓ | Priority ↑↓ | Action ↑↓          | Consumer ↑↓              | Provider ↑↓           | Protocol ↑↓ | Port ↑↓     |
|---------|-------------|--------------------|--------------------------|-----------------------|-------------|-------------|
| Default | 100         | <span>ALLOW</span> | Default                  | Default               | TCP         | 22 (SSH)    |
| Default | 100         | <span>ALLOW</span> | Defined by HTTP Consumer | My HTTP/HTTPS Service | TCP         | 80 (HTTP)   |
| Default | 100         | <span>ALLOW</span> | Defined by HTTP Consumer | My HTTP/HTTPS Service | TCP         | 443 (HTTPS) |

## Discover Policies Automatically

Automatic policy discovery, sometimes referred to as policy discovery, and formerly known as Application Dependency Mapping (ADM), uses existing traffic flows and other data to do the following:

- Suggest a set of “allow” policies based on existing successful network activity.  
The goal of these policies is to identify the traffic that your organization needs, and block all other traffic.
- Group workloads into clusters based on similarity of their computing behavior  
For example, if an application includes multiple web servers, those might be clustered together.  
For more information, see [Clusters, on page 488](#).

You can discover policies for each scope. Typically, you discover policies for scopes at or near the bottom of your scope tree, for example at the application level. However, for initial deployment, you might want to discover policies at a higher-level scope, so you have general, temporary policies in place while you create more refined policies.

You can discover policies as often as desired, to refine the suggested policies based on additional information.

You can manually modify suggested policies and clusters, and/or approve any of them so they are carried forward and not modified by subsequent discovery runs.

You can include both manually created policies and discovered policies in a workspace.

After you discover policies, you will review and analyze them before enforcing them.

To get started discovering policies, see [How to Automatically Discover Policies, on page 437](#).

For more information, see [Policy Discovery Details, on page 437](#).

**Figure 242: Example: Automatically Discovered Policies**

| Rank    | Priority | Action | Consumer                                      | Provider                                       | Protocols And Ports                     |
|---------|----------|--------|-----------------------------------------------|------------------------------------------------|-----------------------------------------|
| Default | 10       | ALLOW  | ... : internal : datacenter : non-prod : app2 | jumpshot                                       | TCP : 12345 (trend-micro-av) ... 1 more |
| Default | 10       | ALLOW  | ... : internal : datacenter : non-prod : app2 | ... : internal : datacenter : non-prod : app2  | TCP : 443 (HTTPS)                       |
| Default | 100      | ALLOW  | wildfire : internal : datacenter : non-prod   | wildfire                                       | ICMP ... 5 more                         |
| Default | 100      | ALLOW  | ... : internal : datacenter : non-prod : app2 | wildfire : internal                            | UDP : 53 (DNS) ... 2 more               |
| Default | 100      | ALLOW  | jumpshot                                      | wildfire : internal : datacenter : non-prod    | TCP : 22 (SSH)                          |
| Default | 100      | ALLOW  | wildfire : internal : datacenter : non-prod   | wildfire : internal : datacenter : non-prod    | TCP : 22 (SSH)                          |
| Default | 100      | ALLOW  | ... : internal : datacenter : non-prod : app2 | wildfire : internal : datacenter : prod : app1 | TCP : 22 (SSH)                          |
| Default | 100      | ALLOW  | wildfire                                      | ... : internal : datacenter : non-prod : app2  | TCP : 3389 (Remote Desktop)             |
| Default | 100      | ALLOW  | wildfire : internal                           | ... : internal : datacenter : non-prod : app2  | TCP : 22 (SSH)                          |
| Default | 100      | ALLOW  | ... : internal : datacenter : non-prod : app2 | ... : internal : datacenter : non-prod : app2  | TCP : 21 (FTP Control) ... 1 more       |

## Policy Discovery Details

Additional information about automatic policy discovery:

- Automatic policy discovery considers conversations in which at least one end is a member workload of the scope within the time range selected. Membership in the scope is based only on the most current scope definition; former membership is not considered.
- By default, policy discovery produces policies and clusters by analyzing communication flows ("conversations"), but optionally can consider other information such as processes running on workloads or load balancer configurations.

See [Include Data From Load Balancers and Routers When Discovering Policies, on page 452](#).

- You can discover policies in any workspace within the scope. Discovery results in each workspace are independent of the results in other workspaces in the scope.
- For detailed discussions of complex concepts that are related to automatic policy discovery, see [Advanced Features of Automatic Policy Discovery, on page 445](#) and [Address Policy Complexities, on page 498](#).

## How to Automatically Discover Policies

Perform the following steps. At any point, you can decide to discover policies again.

Work with colleagues as needed to complete these steps.

| Step | Do This                                                                                                                                                                                                                                          | More Information                                                                                                                                                                                                                                                                                                      |
|------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1    | Upload and label your workload inventory, and gather flow data that inform policy discovery.                                                                                                                                                     | See <a href="#">Get Started with Segmentation and Microsegmentation</a> , on page 2 and subtopics.                                                                                                                                                                                                                    |
| 2    | Choose whether you discover policies for: <ul style="list-style-type: none"> <li>• Workloads in a single scope</li> <li>• Workloads in all of the scopes in a branch of the scope tree</li> </ul>                                                | See <a href="#">Discover Policies for One Scope or for a Branch of the Scope Tree</a> , on page 439.<br><br>(You can always discover policies again at any time.)                                                                                                                                                     |
| 3    | Choose the scope in which you discover policies.                                                                                                                                                                                                 | This depends in part on whether you discover policies for a single scope or for a branch of the scope tree.                                                                                                                                                                                                           |
| 4    | Choose the workspace in which you discover policies.                                                                                                                                                                                             | Generally, you will discover policies in the scope's primary workspace, because you can only analyze policies in a primary workspace. (However, you can always change a workspace to primary later.)<br><br>If your chosen scope does not yet have a workspace, see <a href="#">Create a Workspace</a> , on page 419. |
| 5    | Confirm the inventory that you expect to include in policy discovery.                                                                                                                                                                            | <a href="#">Verify the Workloads That Policy Discovery Will Apply To</a> , on page 441                                                                                                                                                                                                                                |
| 6    | (Optional) Create inventory filters to group workloads that you want to treat as a group.                                                                                                                                                        | See <a href="#">Create an Inventory Filter</a> , on page 378.                                                                                                                                                                                                                                                         |
| 7    | Set the <b>Catch-all</b> action for the workspace.                                                                                                                                                                                               | See <a href="#">Policy Rank: Absolute, Default, and Catch-All</a> , on page 425                                                                                                                                                                                                                                       |
| 8    | Discover Policies                                                                                                                                                                                                                                | <a href="#">Discover Policies Automatically</a> , on page 436<br><br>Be sure to complete the prerequisites in the "Before You Begin" section.                                                                                                                                                                         |
| 9    | View and manage the clusters (groups of workloads) that policy discovery creates.<br><br>(This step applies only when you discover policies for a single scope; clusters are not generated when you discover policies for a branch of the tree.) | See <a href="#">Clusters</a> , on page 488 and subtopics.<br><br>Evaluate the suggested clusters, optionally edit cluster membership as needed, and approve (or better, convert to inventory filters) any clusters that you want to make permanent.                                                                   |
| 10   | Consider complexities such as policy inheritance and cross-scope policies.                                                                                                                                                                       | See <a href="#">Address Policy Complexities</a> , on page 498.                                                                                                                                                                                                                                                        |
| 11   | Review generated policies.                                                                                                                                                                                                                       | See <a href="#">Review Automatically Discovered Policies</a> , on page 520 and subtopics                                                                                                                                                                                                                              |



| Step | Do This                                                                                                                  | More Information                                                                                                                                                                                                                                                                                     |
|------|--------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 12   | Approve policies that you want to keep.                                                                                  | <a href="#">Approve Policies, on page 463</a>                                                                                                                                                                                                                                                        |
| 13   | Discover policies again as desired, to reflect additional flow data, changes in scope membership, or other changes.      | <p><b>Important: Before You Re-run Automatic Policy Discovery, on page 466</b></p> <p>You can rerun policy discovery at any time.</p> <p>Review and approve policies and clusters each time you discover policies.</p>                                                                               |
| 14   | Run live analysis to see how your policies affect your actual traffic.                                                   | <p>When you believe that your policies do what you expect them to do, start <a href="#">Live Policy Analysis, on page 527</a>.</p> <p>If you change policies or rediscover policies, restart policy analysis (to analyze the current policies).</p>                                                  |
| 15   | If you re-discover policies or make other changes, restart live analysis.                                                | See <a href="#">After Changing Policies, Analyze Latest Policies, on page 535</a> .                                                                                                                                                                                                                  |
| 16   | When you are confident that the policies will not block essential traffic, enforce the workspace.                        | See <a href="#">Enforce Policies</a> and subtopics.                                                                                                                                                                                                                                                  |
| 17   | Verify that enforcement is working as expected.                                                                          | See <a href="#">Verify That Enforcement Is Working as Expected, on page 547</a>                                                                                                                                                                                                                      |
| 18   | (Optional) Configure default policy discovery settings that optionally apply when discovering policies in any workspace. | <p>See <a href="#">Default Policy Discovery Config, on page 461</a> and linked topics.</p> <p>Because these are advanced settings, we recommend that you change them only if you have a specific need to change them. You can change them at any time during your process as you realize a need.</p> |

## Discover Policies for One Scope or for a Branch of the Scope Tree

If either option is not possible when you discover policies for a particular scope, the selection is made for you and you will not see a choice of options.

**Table 23: Discovering Policies For**

| A Branch of the Scope Tree                                                                                                                                                                                                        | A Single Scope                                                                                                                                                                                              |
|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Use this method as a starting point, when you are beginning to use Secure Workload, to quickly generate a temporary set of coarse policies that allow existing traffic while helping to protect your network from future threats. | Use this method to fine-tune segmentation policies and ensure that all allowed flows are expected; the smaller number of policies makes it easier to see any existing anomalies that require investigation. |

| A Branch of the Scope Tree                                                                                                                                                                                 | A Single Scope                                                                                                                                                                                                                                                                                                                                                                 |
|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <p>Typically, you use this method for scopes nearer the top of your scope tree.</p> <p>The top of the branch can be any scope in the tree.</p>                                                             | <p>Typically, you use this method for scopes at or near the bottom of your scope tree, for example for scopes dedicated to a single application.</p>                                                                                                                                                                                                                           |
| <p>Discover policies only in one scope – the scope at the top of the branch that you choose.</p>                                                                                                           | <p>Discover policies for each scope in the branch as needed.</p>                                                                                                                                                                                                                                                                                                               |
| <p>All workloads in the chosen scope and all child and descendant scopes are included in discovery.</p>                                                                                                    | <p>Workloads that are also members of any child scope are not included in discovery for this scope.</p> <p>Policies are generated only for workloads that appear in the <b>Uncategorized Inventory</b> tab for that scope on the <b>Organize &gt; Scopes and Inventory</b> page.</p> <p>You can discover policies for workloads in child and descendant scopes separately.</p> |
| <p>All policies for workloads in all scopes in the branch reside in the scope at the top of the branch.</p>                                                                                                | <p>Assuming you also create policies for workloads in child and descendant scopes, policies reside in multiple scopes.</p>                                                                                                                                                                                                                                                     |
| <p>This method typically generates a large number of policies.</p>                                                                                                                                         | <p>This method generates fewer policies in any individual scope.</p>                                                                                                                                                                                                                                                                                                           |
| <p>Discovered policies apply to entire scopes; this option cannot create policies specific to subsets of workloads within scopes.</p>                                                                      | <p>This option can generate policies that apply to subsets of workloads within the consumer and/or provider scope. (Workloads can be grouped by generated clusters and/or by configured inventory filters, and policies applied just to these subsets.)</p>                                                                                                                    |
| <p>All policies are created in a single scope at the top of the branch, so extra steps are not required when a policy's consumer and provider are in different scopes.</p>                                 | <p>Allowing traffic between consumers and providers in different scopes requires extra steps.</p> <p>See <a href="#">When Consumer and Provider Are in Different Scopes: Policy Options</a>, on page 504.</p>                                                                                                                                                                  |
| <p>Discovery can run even if a scope does not have any member workloads with installed agents, as long as descendant scopes have agents or external orchestrators or connectors that gather flow data.</p> | <p>The scope must have member workloads with installed agents or external orchestrators or connectors that gather flow data.</p>                                                                                                                                                                                                                                               |
| <p>This option is available to root scope owners and site admins only.</p>                                                                                                                                 | <p>You must have privileges to create policies for this scope.</p>                                                                                                                                                                                                                                                                                                             |
| <p>The maximum number of agents and conversations is different for each option. See <a href="#">Limits Related to Policies</a>, on page 954.</p>                                                           |                                                                                                                                                                                                                                                                                                                                                                                |
| <p>This option was formerly the Deep Policy Generation advanced configuration option for automatic policy discovery. The behavior has not changed.</p>                                                     | <p>This was formerly the default behavior for automatic policy discovery.</p>                                                                                                                                                                                                                                                                                                  |

| A Branch of the Scope Tree                                                                                                             | A Single Scope |
|----------------------------------------------------------------------------------------------------------------------------------------|----------------|
| For additional details, see <a href="#">Discovering Policies for a Branch of the Scope Tree: Additional Information, on page 441</a> . | --             |

### Discovering Policies for a Branch of the Scope Tree: Additional Information

- All workloads that are conversation endpoints, whether or not they are members of the scope in which policy discovery is run, are assigned the highest matching scope label according to the top-down order given in the external dependencies list.
- For advanced configuration options available when you generate policies for a branch of the scope tree, see:
  - [Enable redundant policy removal, on page 458](#)
  - [Policy Compression, on page 454](#) and related subtopic, [Hierarchical policy compression, on page 454](#)
- Currently, the count of workloads shown for automatic policy discovery includes only those that are not also members of a subscope.

## Verify the Workloads That Policy Discovery Will Apply To

Before you automatically discover policies, verify that the workloads on which policy discovery will be based are in fact the set of workloads you expect. Discovered policies will be generated from flow data captured by agents on these workloads.

### Before you begin

Decide which of the options in [Discover Policies for One Scope or for a Branch of the Scope Tree, on page 439](#) you can use.

### Procedure

- 
- Step 1** From the navigation menu on the left, choose **Defend > Segmentation**.
- Step 2** Click the scope for which you want to discover policies.
- Step 3** Click the workspace in which you want to discover policies.
- Step 4** Click **Manage Policies**.
- Step 5** Click **Matching Inventories**.
- Step 6** If you discover policies for a single scope:
- Click **Uncategorized Inventory**

This page shows workloads that are not also members of child scopes. (In standard automatic policy discovery, policies and clusters are generated in this scope only for workloads that are not also members of child scopes.)
  - Click **IP addresses**.
 

IP addresses on this page do not have Secure Workload agents installed.

Because they do not have agents that are installed, these IP addresses are not considered during automatic policy discovery for this scope UNLESS:

- Policy is being managed via a cloud connector
- The IP addresses are container-based inventory, in which case individual workloads appear on the **Pods** tab, or
- The workloads happen to communicate with a workload in this scope that is considered during policy discovery.

Before discovering policies, consider installing agents on workloads that need them and allowing some time to pass for flow data to accumulate.

c) Click **Workloads**.

Policies and clusters are generated only for workloads on this page and for IP addresses on the IP addresses tab that meet the criteria specified above for consideration.

d) If you have Kubernetes or OpenShift inventory, you will see a **Services** tab and a **Pods** tab.

If you have installed agents on your Kubernetes/OpenShift workloads, check the inventory on those tabs as well.

e) If you have load-balancer inventory, that inventory appears on the **Services** tab.

**Step 7** If you discover policies for a branch of the tree:

a) Click **All Inventory**

This process generates policies (but not clusters) for all workloads in this scope, whether they are also members of child scopes.

b) Click **IP addresses**.

IP addresses on this page do not have Secure Workload agents installed.

Because they do not have agents installed, these IP addresses will not be considered during automatic policy discovery for this scope unless:

- Policy is managed via a cloud connector
- The IP addresses are container-based inventory, in which case individual workloads appear on the **Pods** tab, or
- The workloads happen to communicate with a workload in this scope that is considered during policy discovery.

Before discovering policies, consider installing agents on these workloads and allowing some time to pass for flow data to accumulate.

c) Click **Workloads**.

Policies are generated only for workloads on this page and for IP addresses on the IP addresses tab that meet the criteria specified above for consideration.

d) If you have Kubernetes or OpenShift inventory, you will see a **Services** tab and a **Pods** tab.

If you have installed agents on your Kubernetes/OpenShift workloads, check the inventory on those tabs as well.

e) If you have load-balancer inventory, that inventory appears on the **Services** tab.

**Step 8** Verify that the workloads are the set you expect.

## Automatically Discover Policies

Use this procedure to generate suggested Allow policies based on existing traffic on your network.

You can rediscover policies at any time.

### Before you begin

- Gather flow data before you can effectively automatically discover policies.

Typically, this means you have installed agents on the workloads in the scope, or have configured and gathered data using a cloud connector or external orchestrator.

Flow summary data that is used by automatic policy discovery is computed every 6 hours. Thus, upon initial deployment of Secure Workload, automatic policy discovery is not possible until such data is available.

More flow data generally produces more accurate results.

Before you enforce a policy, you should gather enough data to include traffic that occurs only periodically (monthly, quarterly, annually, and so on.) For example, if an application generates a quarterly report that gathers information from sources that the application does not access at other times, ensure that the flow data includes at least one instance of that report-generation process.

- Complete the steps up to this point in [How to Automatically Discover Policies, on page 437](#).
- Meet the policy discovery-related [Limits Related to Policies, on page 954](#).

If necessary, break larger scopes into smaller child scopes.

- Commit any scope changes before discovering policies, or any configured exclusion filters may not match (exclude) flows as expected. See [Commit Changes, on page 363](#).



**Important** If you are rerunning policy discovery, see the important considerations first: [Important: Before You Re-run Automatic Policy Discovery, on page 466](#).

### Procedure

- Step 1** Choose **Defend > Segmentation**.
- Step 2** In the scope tree or list of scopes in the pane on the left, scroll to or search for the scope for which you want to generate policies.
- Step 3** Click a workspace (primary or secondary) in the scope.
- Step 4** Click **Manage Policies**.
- Step 5** Click **Automatically Discover Policies**.
- Step 6** If you see an option to discover policies for a branch or an entire scope, choose an option.

If you don't see an option, only one option is possible for the scope for which you are discovering policies. For more information, see [Discover Policies for One Scope or for a Branch of the Scope Tree, on page 439](#).

**Step 7** Choose the time range for the flow data that you want to include.

Experiment to find the right time range; you can generate policies as often as needed to get optimal results.

A shorter time range generates results faster, and may generate fewer results.

In general, a longer time range produces more accurate policies. However, if the scope definition has changed, do not include dates before the change is made.

Your time range should include traffic that occurs only periodically (monthly, quarterly, annually, and so on.) if applicable. For example, if an application generates a quarterly report that gathers information from sources that it does not access at other times, be sure that the time range includes at least one instance of that report-generation process.

To configure a time range beyond the last 30 days, select the **custom** range, and fill the required start and end times under the drop-down time selection widget.

**Step 8** (Optional) Specify advanced settings.

Generally, we suggest that you don't change advanced settings for initial discovery runs, then make changes only as needed to address specific issues.

For details, see [Advanced Configurations for Automatic Policy Discovery, on page 451](#).

**Step 9** Click **Discover Policies**. Generated policies appear on this page.

### What to do next

- View [Stop Automatic Policy Discovery in Progress, on page 444](#).
- Return to [How to Automatically Discover Policies, on page 437](#) and continue with the next step in the table.
- You can rediscover policies at any time. For actions you should take first, see [Important: Before You Re-run Automatic Policy Discovery, on page 466](#).

## Stop Automatic Policy Discovery in Progress

Progress of automatic policy discovery is always visible in the header. Navigating to other workspaces does not affect the progress.

To stop the run while it is in progress, click the **abort** button.

Once the run is complete, a message is displayed. If successful, **Click to see results** navigates to a different view showing the changes before and after the run. If automatic policy discovery fails, it is indicated with a different message and a reason.

**Figure 243: Automatic Policy Discovery Progress**



## Advanced Features of Automatic Policy Discovery

You must specify a time range for the discovery run. If necessary, you can configure advanced options.

You can configure advanced options for each workspace, or set defaults for all workspaces (the entire root scope), then modify the settings for individual workspaces as needed.

**Table 24: Configure Advanced Options for Automatic Policy Discovery**

| For a Workspace                                                                                      | For All Workspaces                                           |
|------------------------------------------------------------------------------------------------------|--------------------------------------------------------------|
| Option descriptions for individual workspaces (in column 1) apply also for all workspaces (column 2) |                                                              |
| <a href="#">External Dependencies, on page 448</a>                                                   | <a href="#">Default Policy Discovery Config, on page 461</a> |
| <a href="#">Advanced Configurations for Automatic Policy Discovery, on page 451</a>                  | <a href="#">Default Policy Discovery Config, on page 461</a> |
| <a href="#">Exclusion Filters, on page 445</a>                                                       | <a href="#">Default Exclusion Filters, on page 461</a>       |

### Exclusion Filters

If certain flows are generating unwanted policies, you can exclude those flows from automatic policy discovery using exclusion filters.

For example, to disallow certain protocols like ICMP in the final allow list model, you can create an exclusion filter with a protocol field set to ICMP.



#### Note

- Conversations that match exclusion filters are excluded for the purposes of policy generation and clustering, but remain in the Conversations View with red 'excluded' icon (see the Table View in [Conversations](#)). Likewise, workloads of the workspace incident on such conversations remain viewable as well.
- An exclusion filter that uses a cluster or a filter definition from a workspace is effective only in primary workspaces (otherwise, its cluster definitions are not visible to the label system, and any matching conversations are not excluded).
- Exclusion filters are versioned; to track modifications, see [Activity Logs and Version History](#).
- For limits on the number of exclusion filters, see [Limits Related to Policies, on page 954](#).

You can create one or both of the following, then enable either or both when discovering policies:

- A list of exclusion filters for each workspace.
- A list of default exclusion filters that is available to all workspaces in your tenant.

You can also enable or disable either or both lists for the Default Policy Discovery Config.


For instructions, see [Configure, Edit, or Delete Exclusion Filters, on page 446](#) and [Enable or Disable Exclusion Filters, on page 447](#).

*Configure, Edit, or Delete Exclusion Filters*

You can use this procedure to create a list of exclusion filters for a single workspace, or a list of default exclusion filters that are available to all workspaces.

**Procedure**

**Step 1** Do one of the following:

| To                                                                      | Do This                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
|-------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Configure exclusion filters for a specific workspace                    | Navigate to the workspace, then do one of the following: <ul style="list-style-type: none"> <li>• Click <b>Manage Policies</b>, then click  near the top right of the page and select <b>Exclusion Filters</b>.</li> <li>• From the automatic policy discovery configuration page, click the <b>Exclusion filters</b> link in the Advanced Configurations section.</li> <li>• Delete a discovered policy; you will see an option to create an exclusion filter.</li> </ul> |
| Configure default exclusion filters that are available to any workspace | <ol style="list-style-type: none"> <li>a. Choose <b>Defend &gt; Segmentation</b>,</li> <li>b. Click the caret at the right side of the page to expand the Tools menu, then choose <b>Default Policy Discovery Config</b>.</li> <li>c. Scroll to the bottom of the page.</li> <li>d. Click <b>Default Exclusion Filters</b>.</li> </ol>                                                                                                                                                                                                                        |

**Step 2** To create an exclusion filter, click **Add Exclusion Filter**.

**Step 3** Specify parameters for the flows to exclude from consideration during policy discovery:

You do not need to enter values for all of the fields. Any empty field is treated as a wildcard for matching flows.

Any conversation that matches all the fields of any exclusion filter is ignored for the purposes of policy creation and clustering.

| Option          | Description                                                                                                                                                                                                                                     |
|-----------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Consumer</b> | Matches conversations where the consumer address is a member of the selected scope, inventory filter, or (for workspace-specific exclusion filters only, cluster). You can specify any arbitrary address space by creating a new custom filter. |
| <b>Provider</b> | Matches conversations where the provider address is a member of the selected scope, inventory filter, or (for workspace-specific exclusion filters only, cluster). You can specify any arbitrary address space by creating a new custom filter. |
| <b>Protocol</b> | Matches conversations with specified protocol.                                                                                                                                                                                                  |



| Option      | Description                                                                                                                                                    |
|-------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Port</b> | Matches conversations with provider (server) port matching the specified port, or port range. Enter port ranges using a dash separator, for example, "100-200" |

**Step 4** To edit or delete an exclusion filter, hover over the applicable row to see the **Edit** and **Delete** buttons.

**Step 5** If you are configuring default exclusion filters:

When the configured filters are ready to use, return to the **Default Policy Discovery Config** page, and click **Save** to make the changes available to individual workspaces.

### What to do next



**Important** Exclusion filters are enabled by default in the workspace in which they are configured.

Default exclusion filters are enabled by default in all workspaces.

Both types of exclusion filters are enabled by default in the Default Policy Discovery Config.

Before discovering policies:

- Enable or disable exclusion filters and default exclusion filters.
  - In each workspace
  - On the Default Policy Discovery Config page

For instructions, see [Enable or Disable Exclusion Filters, on page 447](#).

- Commit any scope changes, or the filters may not match (and therefore exclude) the expected flows. See [Commit Changes, on page 363](#).

### Enable or Disable Exclusion Filters

You can create exclusion filters in each workspace and/or create a set of default exclusion filters that you can apply to all workspaces.

By default, both types of exclusion filters are enabled.

To make changes

- To enable or disable exclusion filters for a single workspace:

In the workspace, click **Manage Policies**, then click **Automatically Discover Policies**, then click **Advanced Configurations**. You can enable exclusion filters and/or default exclusion filters for this workspace.

- To enable or disable exclusion filters in the Default Policy Discovery Config:

Choose **Defend > Segmentation**, then click the caret at the right side of the page to expand the Tools menu. Then choose **Default Policy Discovery Config**. Scroll to or click **Advanced Configurations**. You can enable exclusion filters and/or default exclusion filters.

## External Dependencies

External dependencies are relevant only when you use the process that is described in [\(Advanced\) Create Cross-Scope Policies, on page 505](#).

External Dependencies settings apply to automatically discovered policies involving communications to and from workloads that are members of a scope other than the scope in which policies are discovered. (That is, communications involving "external workloads.")

A workload that is not a member of the scope in which the policy exists is an *external workload*. Such workloads are the other end of a conversation with a *target workload* (which is a member of the scope in which the policy exists).

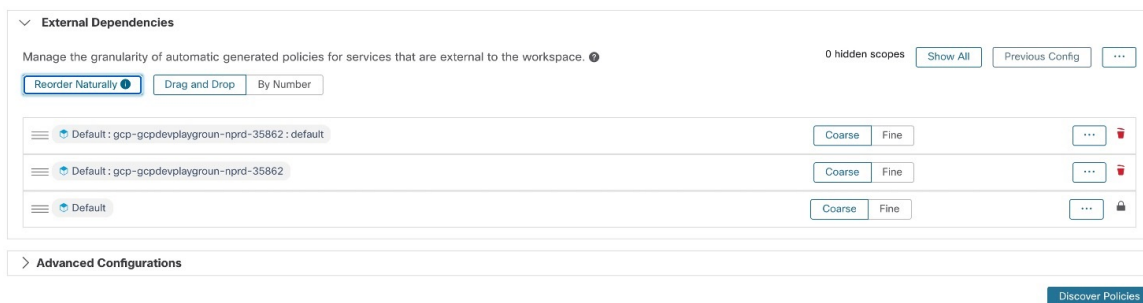
The External Dependencies list is an ordered list of all scopes in your hierarchy. Each scope in the list is set to one of the following:

- Generate specific or refined policies (more secure), OR
- Generate coarse policies in higher scopes, which may generalize better (that is, be more likely to allow legitimate flows that were not seen in the time range that is specified when discovering policies).

During policy discovery, the first scope (or cluster, or inventory filter – see below) that matches the workload will be used to generate the “allow” policy, where the matching order (and consequent granularity level) is determined by the top-down ranking that is displayed in the External Dependencies section.

A default scope order is configured for you, with all scopes set to "Coarse" by default.

**Figure 244: Default External Dependencies**



| To                                                                | Do This                                                                                                                                                                                                                                                                      |
|-------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| View or fine-tune external dependencies for a workspace:          | Navigate to the workspace and click <b>Automatically Discover Policies</b> , then click <b>External Dependencies</b> .<br><br>To reorder the scopes and choose granular options for each, see <a href="#">Fine-Tune External Dependencies for a Workspace, on page 449</a> . |
| Configure default external dependencies for an entire root scope: | See <a href="#">Default Policy Discovery Config, on page 461</a> .                                                                                                                                                                                                           |

### External Dependencies: Granular Policies Involving Subsets of Scopes

You can optionally discover policies at a more granular level than scope-to-scope, to control traffic to a specified subset of the workloads in a scope.

For example, you may want to create policies specific to a certain type of host within an application, such as API servers; you can group those workloads into a subset within the application scope.

To generate policies specific to a subset of workloads within a scope, see [Fine-Tune External Dependencies for a Workspace](#), on page 449.

### *Tips for Exploring External Dependencies*

Use the following tips to explore the behavior of automatic policy discovery for policies involving workspaces that are not members of the scope that is associated with the workspace in which the policies reside.



#### **Tip**

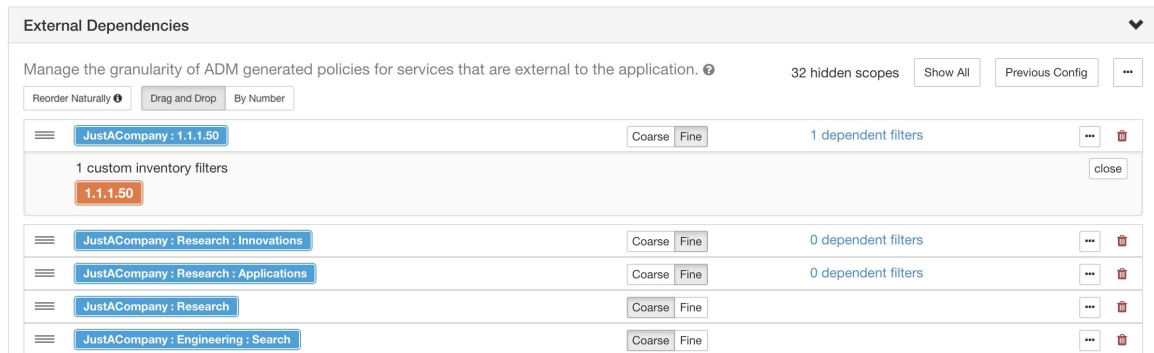
- You can remove and rearrange the list to generate policies at a desired granularity. For example, removing all Company: RTP subsopes will help generate wide policies to the whole Company:RTP scope, but not its individual components, while maintaining the higher granularity for Company: SJC scope. Furthermore, you can click on the **Fine** button next to any scope and see if there are finer grain candidates defined under that scope.
- By default, the root scope is configured as the lowest entry in the External Dependencies list, so that automatic policy discovery always generates policies to more specific scopes whenever possible. Initially, to view relatively few coarse-grained policies, you can temporarily place the root scope on the top of the external dependencies. This way, after automatic policy discovery, you will see all external policies of the workspace connecting to only one scope, the root scope (as every external workload maps to the root scope). The resulting number of generated policies are smaller and easier to examine and comprehend.
- You can also temporarily bundle all workloads that are members of the scope associated with the workspace ("internal workloads") into one cluster, approve the cluster, and then discover policies. Again, this results in a reduced set of policies, as no clustering (subpartitioning of the workspace/scope) takes place, so you can view policies that are either internal (connect to internal workloads), or external (connect an internal to an external workload). Later, you can view progressively more refined policies by unbundling internal workloads and/or placing one or a few external scopes of interest above the root.
- **Important** Always carefully examine policies involving the root scope, since these policies allow all traffic to and from the entire network. This is especially important when the root scope is placed low in the External Dependencies list and it is not your intention to generate coarse policies. Such policies may not have resulted from network-wide traffic in or out of the workspace scope. Rather they can be triggered by a few external endpoints which failed to receive finer scopes or inventory filter assignments beyond simply the root scope.

While auditing these policies, you should examine the associated conversations (See [Conversations](#)) to identify these endpoints and subsequently categorize them into finer scopes or inventory filters, to avoid less-secure policies at the root-scope level.

### *Fine-Tune External Dependencies for a Workspace*

Use this procedure to create policies between specified subsets of workloads within scopes (rather than between entire scopes) during automatic policy discovery, when the provider of a policy belongs to a different scope than the scope in which policies are being discovered.

Figure 245: Fine-tuning External Dependencies



### Before you begin

- Configure an inventory filter for each subset of workloads for which you want to generate specific policies. You can create any number of inventory filters, in any scope.

There are several ways to create inventory filters:

- Convert clusters of interest to inventory filters.  
(See [Convert a Cluster to an Inventory Filter, on page 493](#)),  
and/or
- Create new inventory filters.  
See [Create an Inventory Filter, on page 378](#).

These filters must have the following options enabled:

- **Restrict query to ownership scope**  
**Provides a service external of its scope**

- See also [Tips for Exploring External Dependencies, on page 449](#).

### Procedure

- 
- Step 1** Navigate to the workspace in which you will discover policies.
  - Step 2** Click **Automatically Discover Policies**.
  - Step 3** Click **External Dependencies**.
  - Step 4** If necessary, click **Show All** scopes.
  - Step 5** (Optional) Leverage previous configurations:
    - To reuse the changes you made to the list the last time you discovered policies, click **Previous Config**.
    - If you have set up external dependencies in the global “Default Policy Discovery Config”, you can use the global list by clicking **Default Config**. Or, after obtaining the default list, you can modify it as desired (for that workspace only), and then use the customized version on subsequent runs by clicking **Previous Config** once.

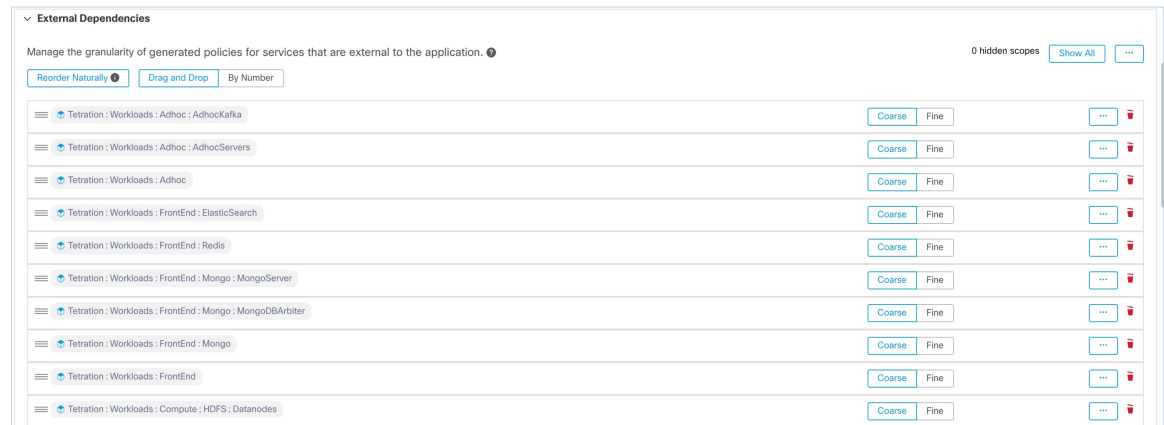
**Step 6** Reorder scopes (and inventory filters, if applicable) as needed.

Policy is applied based on the first scope or inventory filter in the list (starting from the top) that matches the traffic. For this purpose, you generally want to apply the most specific policy that matches traffic, so you want child scopes (more specific) above their parents (less specific).

- If you have recently created new child scopes, which by default are added to the bottom of the list, reorder the entire list to place child scopes above their parents:

(Recommended) Click **Reorder Naturally**.

**Figure 246: Reorder naturally**



- (If you have a specific reason) To reorder the list manually:

- Click **Drag and Drop**.
- Click **By Number**:

The external dependencies will be assigned priority values in multiples of 10. Change the values to change the order.

Once numbers are modified, click **View** to update the list order and reassign multiples of 10 to each of the priorities.

**Step 7** Specify granularity for each row:

- Click **Fine** for each row for which you want to generate policies specific to configured inventory filters or clusters.

Click **Coarse** to generate policies that apply to the entire scope.

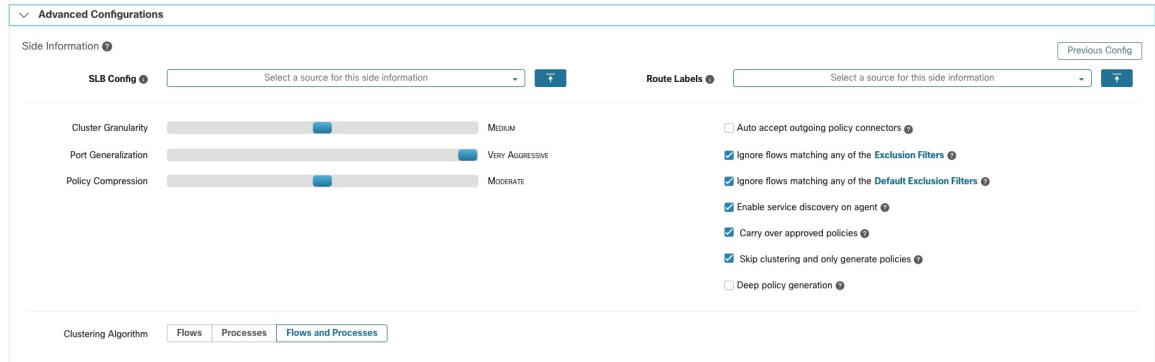
- To apply granularity to all subscopes of a scope: Click the  button at the end of the scope's row.

## Advanced Configurations for Automatic Policy Discovery

Use advanced settings to include additional information when discovering policies or to adapt to a particular environment.

- To access these settings for a specific workspace, click **Automatically Discover Policies** in the applicable workspace.
- To change the defaults for all workspaces, see [Default Policy Discovery Config](#), on page 461.

Figure 247: Advanced Automatic Policy Discovery Configurations



*Include Data From Load Balancers and Routers When Discovering Policies*

You can upload data from load balancers and routers to inform automatic policy discovery.

To access the following options, click **Advanced Configurations** in the automatic policy discovery settings and look at the "Side Informaton" or "sideinfo" section.

| Option                                                             | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
|--------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <p><b>SLB Config</b><br/>(Upload load balancer configurations)</p> | <p>To download data from your load balancer in the correct format, see <a href="#">Retrieving LoadBalancer Configurations for Advanced Policy Discovery Configuration</a>.</p> <p>Supported formats for uploading loadbalancer configs:</p> <ul style="list-style-type: none"> <li>• <b>F5 BIG-IP</b></li> <li>• <b>Citrix Netscaler</b></li> <li>• <b>HAProxy</b></li> <li>• Others:</li> </ul> <p>Use the <b>Normalized JSON</b> schema.</p> <p>You must convert any unsupported load balancer config into this schema.</p> <p>This simple schema includes basic information on Virtual IPs (VIPs) and backend IPs.</p> <p>To download a sample JSON file, click the info button beside <b>SLB Config</b>.</p> |

| Option                     | Description                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|----------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Upload <b>Route Labels</b> | <p>You can upload a list of provisioned subnets/routes from the routers to help partition hosts based on pre-provisioned set of subnets. The clustering results generated by automatic policy discovery never span the subnet boundaries as defined by the uploaded data. You can modify the results after automatic policy discovery is complete.</p> <p>To download a sample JSON file, click the info button beside <b>Route Labels</b>.</p> |

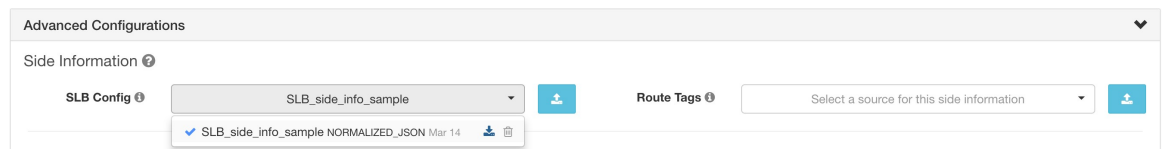


**Note** Clusters do not span partition boundaries, meaning a cluster computed by automatic policy discovery does not contain target workloads from two different partitions. Partitions are computed from the uploaded load balancer or router data. However, you can freely move workloads from one cluster to another, for example by changing cluster query definitions (manual cluster editing), or disable the upload of any side info.

#### To view or delete a previously uploaded Load Balancer (SLB Config) or Route Labels file:

1. Click into the respective box labeled **Select a source for this side information**.  
A list of uploaded files will appear.
2. Click the download or trash icon beside the file to view or delete.

**Figure 248: Uploaded Side Information**



### Cluster Granularity

Clustering Granularity allows you to control the size of the clusters generated by automatic policy discovery.

- **Fine** results in more but smaller clusters
- **Coarse** results in fewer but larger clusters



**Note** You may not observe a significant change in the results due to many other signals that our algorithms take into account. For example, if there is a very high confidence in the generated clusters, changing this control will make little change in the results.

### Port Generalization

The **Port Generalization** option in **Advanced Configurations** for automatic policy discovery controls the level of statistical significance required when performing port generalization, i.e., replacing numerous ports being used as server ports on a single workload, with a port interval.

This setting can affect accuracy, number, and compactness of policies and the time required to generate them.

To disable port generalization, move the slider to the extreme left. Note that if disabled, automatic policy discovery and/or automatic policy discovery UI rendering time may be slowed substantially, in case many server ports are used by the workloads.

As the slider is moved to the right toward more aggressive generalization, less evidence is required to create port-intervals and also the criterion for replacing original policies (involving single ports) with port-intervals is relaxed.

### Background

Some applications such as Hadoop use and change many server ports in some interval, for instance in 32000 to 61000. Automatic policy discovery attempts to detect such behavior for each workload, using the workload's server port usages in the observed flows: by observing only a fraction of total possible ports (but numerous ports, eg 100s), automatic policy discovery may 'generalize' that any port in, say 32000 to 61000, could be used as a server port by the workload. Ports that fall within intervals are replaced with such intervals (when certain criteria on minimum observed counts are met). This results in fewer, more compact policies. Interval estimation is important for computing accurate policies: without sufficient generalization many legitimate future flows would be dropped if the policy is enforced. By merging numerous ports into one or a few intervals, the rendering time of the UI is sped up significantly as well.

You can control the degree of port generalization including disabling it.

### Policy Compression

When policy compression is enabled, if policies in multiple clusters in the workspace are similar, then those policies can be replaced with one or more policies applicable to the entire parent scope. For example, if all or almost all clusters in the workspace provide the same port to the same consumer, then all of those cluster-specific policies are replaced with one policy in the parent scope. This reduces the number of policies significantly, minimizes clutter, and may also allow legitimate future flows that would have been dropped (accurate generalization).

The more aggressive the compression setting, the smaller is the required threshold on policy frequency in order to replace cluster-specific policies with a policy applicable to the entire parent.

#### When generating policies for a branch of the scope tree:

This knob can be used to alter the level of aggressiveness in [Hierarchical policy compression](#).




---

**Note** Currently, the automatic policy discovery conversations page does not support showing the conversations that led to a compressed policy (you may need to disable compression or use flow search).

---

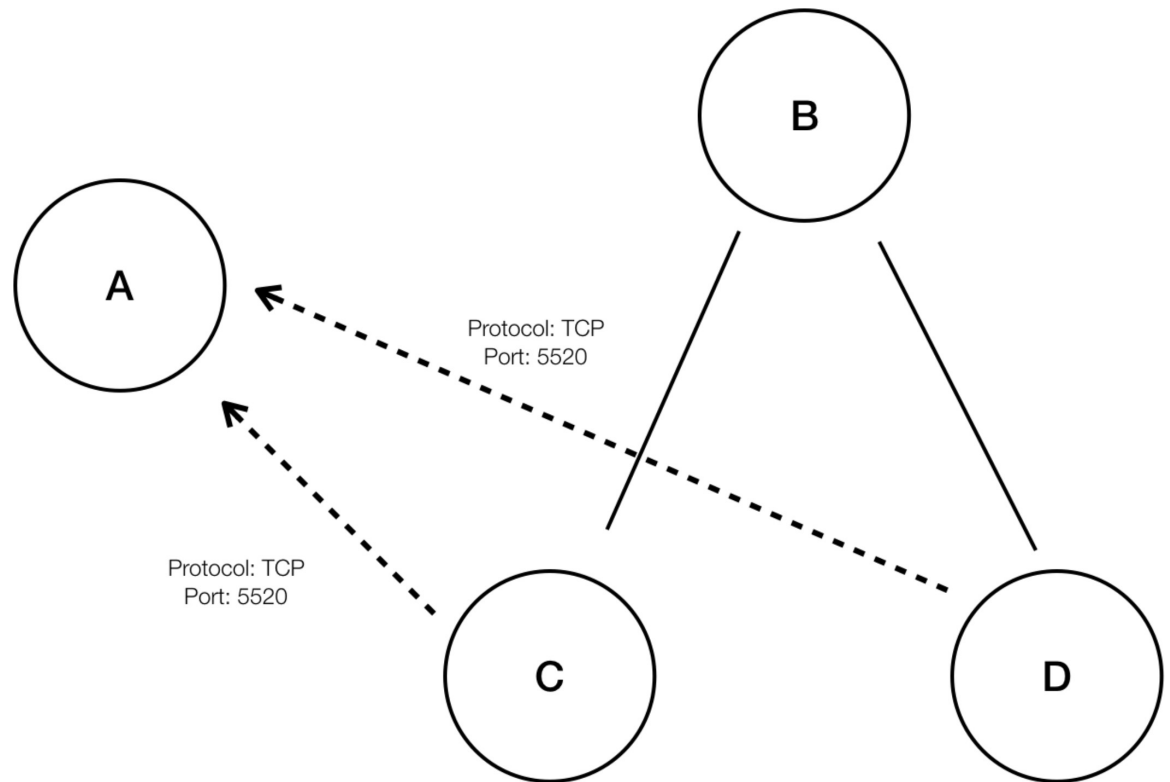
### Hierarchical policy compression

Policy compression can also be done when generating policies for a branch of the scope tree. The [Policy Compression](#) knob can be used to alter the level of aggressiveness in hierarchical policy compression. An example of hierarchical policy compression is illustrated below.

- Let A, B, C and D be scopes part of a scope tree, where "C" and "D" are the child scopes of "B". Let "C" → "A" be a TCP "ALLOW" policy on port 5520 and "D" → "A" be TCP "ALLOW" policy on port 5520.

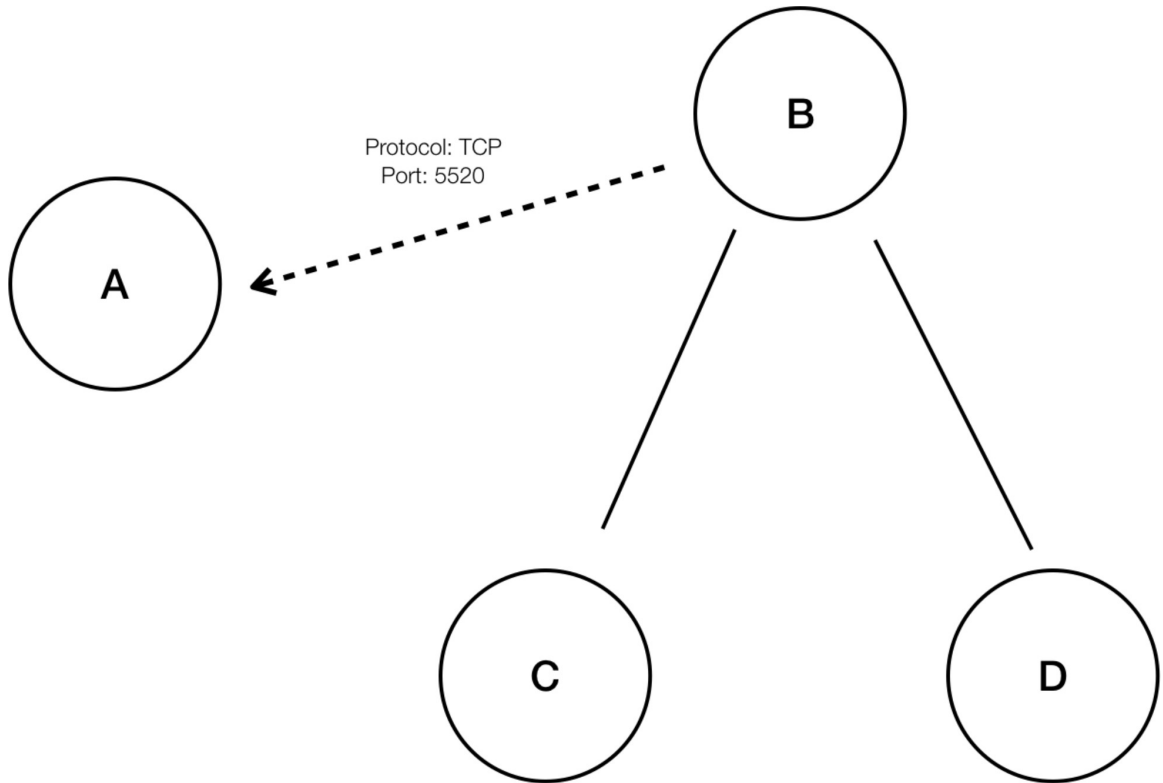


Figure 249: Before hierarchical policy compression



- With hierarchical policy compression if a sufficiently large group child scopes involves in policies sharing the same port, protocol and destination or source, these policies will be replaced by a generalized policy that connects the parent scope to the common source or destination. In the above mentioned case “C” and “D” are child scopes of “B” and the policies “C” → “A” and “D” → “A” share the same destination, port and protocol. Since 100% of child scopes of “B” contain the similar policy the policy will be promoted to be “B” → “A”, resulting in the following. Furthermore, hierarchical compression can be repeated so a generalized policy can go all the way to the root of the subtree (branch of the scope tree) .

Figure 250: After hierarchical policy compression



- The policy compression knob allows you to tune the aggressiveness of such compression, by changing the minimum required proportion of the policy-sharing child scopes (usually measured as the fraction of total number of child scopes) to trigger the compression. When disabled, each policy is generated between highest priority scopes based on the External Dependencies list. Subsequently, if you choose to impose the naturally ordered External Dependencies list, the policies generated will be the most granular policies among scopes.

### Clustering Algorithm (Input to Clustering)

Advanced users can choose the main source of data for clustering algorithms, that is, live network flows, or running processes, or both.

### Auto accept outgoing policy connectors

This option is applicable only when you use automatic policy discovery to create cross-scope policies using the method described in [\(Advanced\) Create Cross-Scope Policies, on page 505](#).

Any outgoing policy requests created during automatic policy discovery will be automatically accepted.

For complete information, see [Auto Accept Policy Connectors, on page 514](#) and [Policy Requests](#).




---

**Note** This option is only available for root scope owners and site admins.

---

### Auto Approve Generated Policies

This option is applicable if you want to approve all policies generated by policy discovery.



**Note** Be aware that if you choose this option, and later on if you need to modify or undo any changes, you can only do so manually.

For more information, see [Auto Accept Policy Connectors, on page 514](#) and [Policy Requests](#).



**Note** This option is available for root scope owners and site administrators.

### Ignore Flows Matching Exclusion Filters

To ignore conversation flows that you specify, enable the applicable option. To view or modify either filters list, click the applicable **Exclusion Filters** link. For more information, see [Exclusion Filters, Default Exclusion Filters, on page 461](#), and [Configure, Edit, or Delete Exclusion Filters, on page 446](#).

### Enable service discovery on agent

In certain applications, a large range of ports might be designated for use, but actual traffic might use only a subset of those ports during the time period included in policy discovery. This option allows the entire designated pool of ports for these applications to be included in policies for those applications, rather than just the ports seen in actual traffic.

Enabling this option allows ephemeral port-range information regarding services present on the agent node to be gathered. Policies are then generated based on this port-range information.

#### Example:

- Windows Active Directory Domain Server uses default Windows ephemeral port-range **49152-65535** to serve requests. When this flag is set this port range information is reported by the agent and policies are generated based on this information.

**Figure 251: Service discovery enabled on the agent**

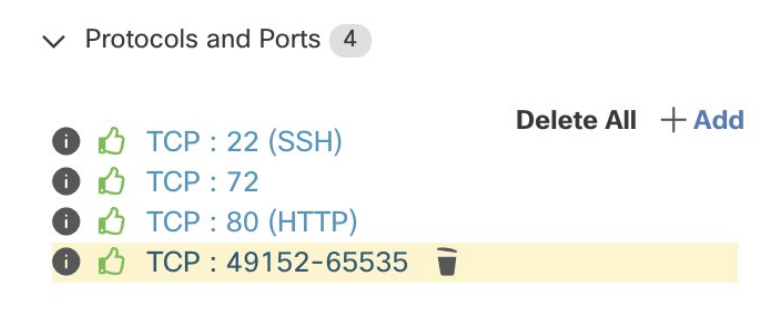
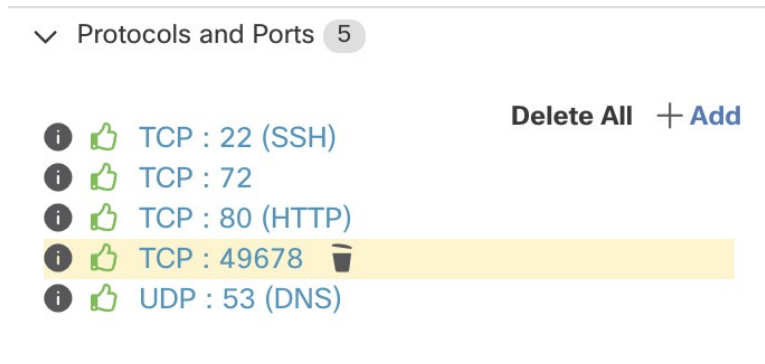


Figure 252: Service discovery not enabled on the agent



### Carry over Approved Policies

This option is enabled by default.

When this flag is set, all the policies that you have marked as approved (including those approved using OpenAPI) will be preserved. This helps you to not have to re-define a particular broad DENY rule that should take effect regardless of the “allow” policies that are discovered by automatic policy discovery.

For details, see [Approved Policies, on page 463](#).

### Skip clustering and only generate policies

If this option is selected, no new clusters are generated, and policies are generated from any existing approved clusters or inventory filters and otherwise involve the entire scope associated with the workspace (in effect, treating the entire scope as a single cluster). This option can result in substantially fewer (but coarser) policies.

### Enable redundant policy removal

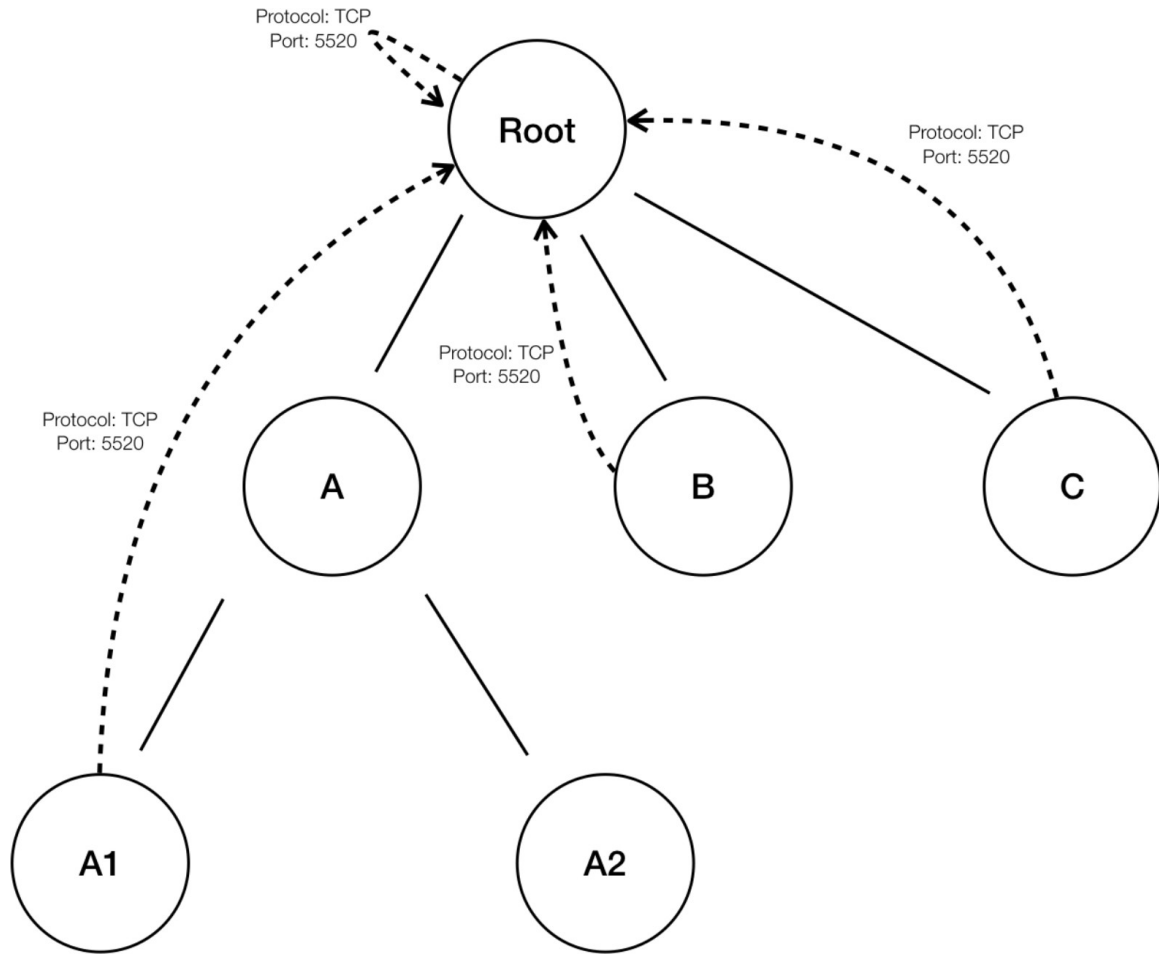
This option is only available when generating policies for a branch of the scope tree.

This option enables/disables removal of redundant granular policies.

#### Example:

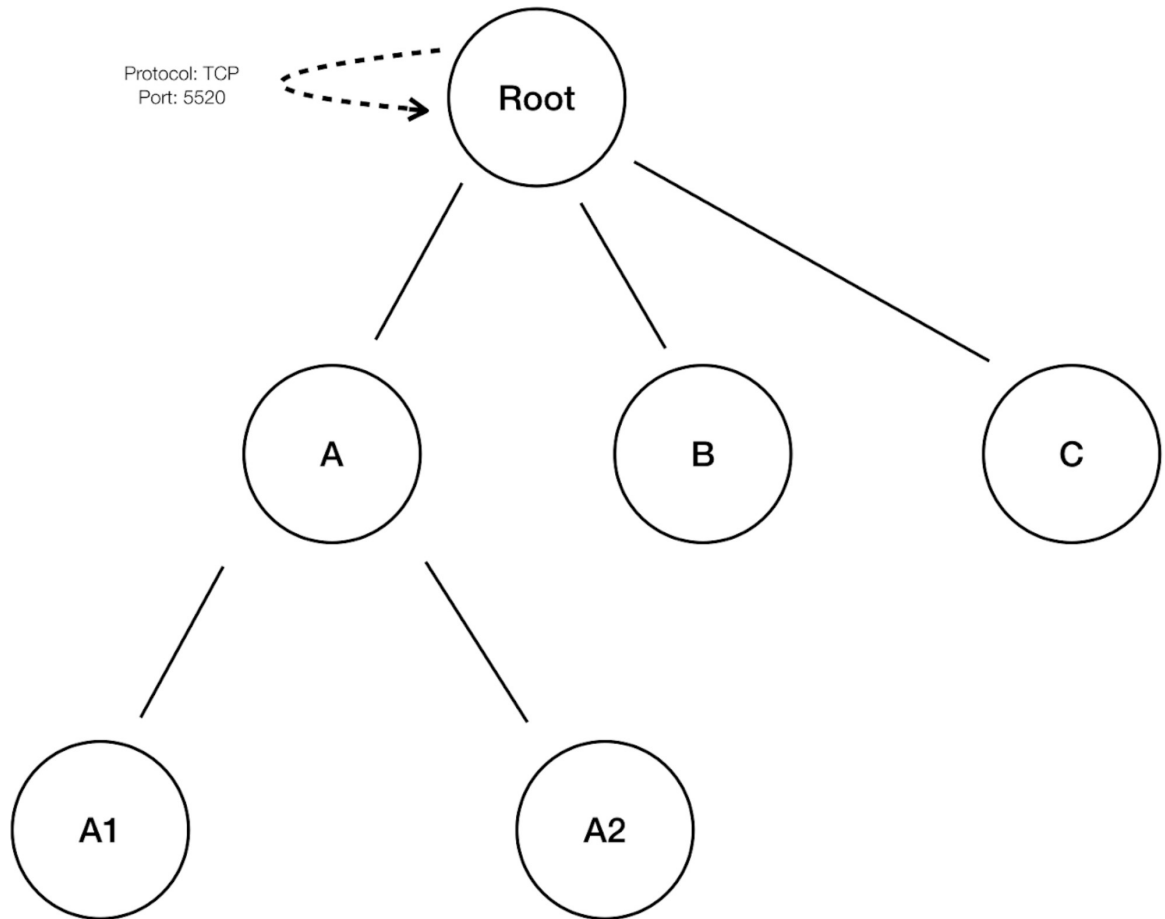
- Let Root, A, B, C, A1 and A2 be scopes part of a scope tree. Let the following be the policies:
  1. “Root” → “Root”
  2. “B” → “Root”
  3. “C” → “Root”
  4. “A1” → “Root”

Figure 253: Before removal of redundant policies



- The policies “B” → “Root”, “C” → “Root” and “A1” → “Root” are redundant as the policy “Root” → “Root” covers these policies. The remove redundant policies feature will check and remove such policies resulting in only one policy “Root” → “Root” as follows.

Figure 254: After removal of redundant policies



Redundant policy removal can be very useful in maintaining a succinct set of interpretable policies. The reduced policy set contains the minimal number of policies at the chosen compression level to cover all the workload traffic. However, you should always audit the policy through policy analysis and examine the corresponding conversations to evaluate the tightness of the resulting policies. This is especially important when there exists traffic to or from endpoints that are not categorized into finer scopes or inventory filters. Such endpoints may trigger the generation of coarser policies than intended, such as policies involving the root scope. If at the same time, redundant policy removal is enabled, more granular policies will be removed and will not be presented to you. To diagnose the source of (compressed) policies and to view finer level policies, turn off policy compression and redundant policy removal. Also note that currently, the automatic policy discovery conversations page may fail to show the conversations that lead to a compressed/generalized policy; so to get around this, you can turn off compression and redundant policy removal, so it is easier to find the conversations that lead to the generated policies.



**Tip** Since discovering policies for a branch of the scope tree discovers all policies for the scope subtree rooted at the workspace scope, these policies will cover all the legal traffic seen by automatic policy discovery for all the workloads under the subtree. When analyzing these policies using tools such as Policy Analysis (See [Policies](#)), you should turn off Policy Analysis in all the workspaces associated with the subscopes. This way, the policies (if any) residing in the subscope workspaces (usually receive a high priority due to more specific scope definition) will not take priority and interfere with the results. However, exceptions apply when the policies in the subscope workspaces are configured to cover different sets of traffic that usually involve finer inventory filters or clusters specific to the subscopes.

## Default Policy Discovery Config

You can configure default automatic policy discovery settings that can optionally be used in any workspace in the entire root scope.

To configure default options for policy discovery:

Choose **Defend > Segmentation**, then click the caret at the right side of the page to expand the Tools menu. Then choose **Default Policy Discovery Config**.

**Figure 255: Navigating to the Default Policy Discovery Config page**

| Type         | Version | Absolute Policies | Default Policies | Catch All |
|--------------|---------|-------------------|------------------|-----------|
| Enforced     | N/A     | N/A               | N/A              | N/A       |
| Analyzed     | N/A     | N/A               | N/A              | N/A       |
| Latest Draft | V5      | 0                 | 23               | ALLOW     |

For information about options on the Default Policy Discovery Config page, see:

- [External Dependencies](#), on page 448 and subtopics
- [Advanced Configurations for Automatic Policy Discovery](#), on page 451 and subtopics
- [Default Exclusion Filters](#), on page 461



**Important** When your default configurations are complete and ready to use in individual workspaces, click **Save**.

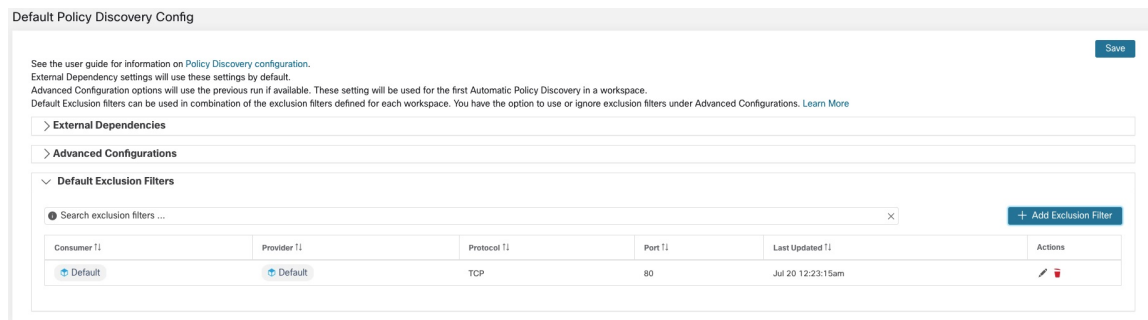
## Default Exclusion Filters

Exclusion Filters help you fine-tune policies and clusters suggested by automatic policy discovery by specifying traffic flows to exclude from discovery input.

For details, see [Exclusion Filters](#).

You can make a global Default Exclusion Filters list that is available to all workspaces in your tenant, then specify for each workspace whether or not to use this default list when discovering policies.

Figure 256: Default Exclusion Filters



To configure default exclusion filters, see [Configure, Edit, or Delete Exclusion Filters, on page 446](#).

To enable or disable default exclusion filters, see [Enable or Disable Exclusion Filters, on page 447](#).

## Retrieving LoadBalancer Configurations for Advanced Policy Discovery Configuration

Below are the instructions for retrieving supported load balancer configuration files in a format that can be directly uploaded to Secure Workload for use in policy discovery. For more information, see [Advanced Configurations for Automatic Policy Discovery](#) and [Include Data From Load Balancers and Routers When Discovering Policies, on page 452](#).

Note that all files must be encoded as ASCII.

### Citrix Netscaler

Concatenate the output of `show run` in your console and upload the file.

See [Sample config file](#)

### F5 BIG-IP

Upload the `bigip.conf` file.



**Note** If you have a file with a .UCS extension, unzip the archive folder and upload only the `bigip.conf` file within the configuration dump. If there are multiple `bigip.conf` files, concatenate and then upload the files.

See [Sample config file](#)

### HAProxy

Upload your `haproxy.cfg` file. The path is typically `/etc/haproxy/haproxy.cfg`.

See [Sample config file](#)

### Normalized JSON

If you find the above options limiting, convert your configs to the following JSON schema and upload them directly. The example JSON file can be directly downloaded by clicking the **i** icon next to SLB Config in Advanced Run Configurations for automatic policy discovery.

See [Sample config file](#)



## Approve Policies

As you review policy discovery results, approve discovered policies that you want to keep, to carry them forward intact when you discover policies in future. For complete details, see [Approved Policies, on page 463](#).

To approve a policy:

1. On the Policies page, for the policy you want to protect, click the value in the **Protocols and Ports** column.
2. In the panel that opens on the right, select the checkbox to the left of each protocol-and-port for which you want to retain the policy during future policy discovery.

**Figure 257: Approve Policies**

The screenshot shows the 'Policies' page in the Cisco Secure Workload interface. The main table lists policies with columns for Rank ID, Priority ID, Action ID, Consumer ID, Provider ID, Protocol ID, Port ID, and Confidence ID. The 'Action ID' column shows 'ALLOW' for all listed policies. The 'Port ID' column contains links to protocol and port details, such as '53 (DNS)', '80 (HTTP)', '123 (NTP)', '137 (NETBIOS Name Service)', '443 (HTTPS)', and '5660 (Secure Workload Enforcement)'. The right-hand panel, titled 'Policy Actions', shows the configuration for a selected policy, including Priority (100), Action (ALLOW), Consumer (Default), and Provider (Default). A tooltip is visible over the 'Port ID' column, stating: 'Policies marked as 'approved' will be carried over during the next Automatic Policy Discovery ONLY IF there are matching consumer and provider filters. This policy is not approved click to toggle'. Below the tooltip, there are checkboxes for specific protocol and port combinations: TCP - 6443, UDP - 53 (DNS), UDP - 123 (NTP), and UDP - 137 (NETBIOS Name Service).

You can also use this procedure to remove approval from a policy.

## Approved Policies

In general, approved policies are not changed during automatic policy discovery, and automatic policy discovery does not suggest policies that would duplicate or overlap the effects of approved policies.

The following are approved policies:

- Manually created policies.
- Discovered policies that are manually approved.  
(When you are satisfied that a policy behaves as intended, you approve it to protect it from changes during future automatic policy discovery. See [Approve Policies, on page 463](#).)
- Uploaded policies, unless explicitly marked as `approved: false`.
- Approved policies that are defined in parent and ancestor scopes (specifically, from the latest versions of their primary workspaces) that apply to workloads in this scope.
- Policies created when policy requests are accepted from another workspace when cross-scope policies are handled using the advanced method that is described in [When Consumer and Provider Are in Different Scopes: Policy Options, on page 504](#). For example, this includes policies that are included from the [Provided Services, on page 516](#) tab.

Approved policies are shown with a thumbs-up icon next to the protocol type when you click a policy's ports or protocols link and view details in the panel at the right side of the page.

### Exceptions to Approved Policy Protections

Approved policies are preserved during future automatic policy discovery if *both* ends of the policy are any of: approved cluster; inventory filter; accepted policy request (for cross-scope policies); or a cluster that doesn't significantly change membership. (However, the cluster membership may have changed in the last case.)

Approved policies may not be protected during future automatic policy discovery runs if either end of the policy is a cluster that is not approved, and if, upon automatic policy discovery, no newly generated cluster has sufficiently high overlap with such cluster.

To protect a policy that involves an unapproved cluster, you should explicitly approve the clusters at each end of the policy.

There is also an advanced configuration for automatic policy discovery that is enabled by default. If you do not want to protect approved policies from changes, you can deselect this option for a workspace or for the global default policy discovery configuration. See [Carry over Approved Policies, on page 458](#).

## Troubleshoot Approved Policies

### Approved policies are not being carried forward

If approved policies are not being carried forward as expected, make sure the **Carry over approved policies** option is selected in the advanced and/or default configuration settings for automatic policy discovery.

### Finding conversations that are excluded from policy generation

During automatic policy discovery, any conversations that match the criteria for an existing approved policy are excluded from the policy generation. This omission prevents redundant policies covering the same conversations from being generated. (This process differs from the exclusion filters (See [Exclusion Filters](#)), in which you define matching filters instead of policies. Exclusion filters prevent matching conversations from being visible to all parts of automatic policy discovery. )

Note that while redundant policies are not generated from these conversations, the conversations are still considered when automatic policy discovery analyzes and generates clusters.

To see which conversations are excluded from automatic policy discovery by existing approved policies:

In the conversations view (See [Conversations](#)), use the **excluded** flag to filter conversations. You can also explore which existing approved policies result in the exclusion of these conversations in the policy details view that opens on the right side of the page when you click the ports and protocols link in a policy, then click the exclusion icon next to the conversation. (Hover over the icons to find the right icon.)

## Iteratively Revise Policies

Defining and refining policies, for a single scope and for an entire network, will be an iterative process.

You can expect to revise both discovered and manually created policies.

### Re-running Automatic Policy Discovery

You can rerun automatic policy discovery at any time. The main reasons to rerun automatic policy discovery are to include additional information that was not included in the previous run, or to exclude information that is not helpful. For example, you can:

- Install additional agents or configure additional connectors, and allow some flow data to accumulate.

- Increase the timespan used for discovery, to include more data.
- Approve clusters (with or without editing them first), which can improve the clustering of other workloads upon rerun. See [Approving Clusters, on page 496](#).
- Exclude flows that you know you don't want to influence policy so you don't have to edit them out. See [Exclusion Filters, on page 445](#).
- Change advanced settings (for details, see [Advanced Configurations for Automatic Policy Discovery, on page 451](#).)
- Capture changes after you have made changes to [Address Policy Complexities, on page 498](#).

Automatically discovering policies again on an existing workspace may generate different clusters and policies in the workspace.

If a host is no longer in the scope of the workspace, upon a subsequent automatic policy discovery run, that host will not appear in any cluster; if it were in an approved cluster, it will no longer appear in that cluster. Even with the same set of member workloads but with a different timeframe or configuration, automatic policy discovery may result in different clusters.



---

**Note** For a list of the types of policies that are not modified during policy discovery, see [Approved Policies, on page 463](#).

---



---

**Note** *Removal of Redundant Policies* On subsequent automatic policy discovery, approved policies in primary workspaces will remove matching conversations for policy generation, so redundant policies are not generated. Note that, as is the case for exclusion filters, this functionality may not work perfectly on non-primary workspaces if the policy uses a Cluster filter defined in the workspace. Cluster filters from a non-primary workspaces are not active, and will not match any flows, thus redundant policies may still be generated in non-primary workspaces during automatic policy discovery.

---

## Important: Before You Re-run Automatic Policy Discovery



- Important** Address each of the following before re-discovering policies in a workspace:
- By default, each time you discover policies in a particular workspace, the previous set of discovered policies and clusters are overwritten based on the data included in the new discovery period. If you want to keep some policies and clusters but not others, approve those policies and clusters.
  - If you want to preserve any existing generated clusters, see [Preventing Cluster Modification During Automatic Policy Discovery Reruns](#) or [Approving Clusters, on page 496](#).
  - If you want to preserve any existing generated policies, see [Approve Policies, on page 463](#).
  - Any existing **Advanced Configuration** settings configured in the previous discovery run are used unless you change them.  
However, any configured *default* External Dependencies will be used over those of the previous run.
  - If the currently displayed version of the discovered policies is not the latest version, and you want to keep previously discovered versions, click the version displayed at the top of the page and choose the latest v\* version.  
If a previous version is displayed, any versions between that version and the new discovered version will be deleted.  
For details, see [View, Compare, and Manage Discovered Policy Versions, on page 466](#).

To re-run policy discovery, see [Automatically Discover Policies, on page 443](#). After you have addressed the points in this topic, the process is the same each time you discover policies.

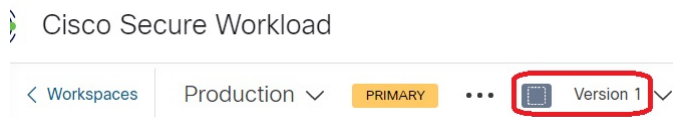
## View, Compare, and Manage Discovered Policy Versions

Each time you discover policies in a workspace, the version number (v\*) assigned to the set of policies increments.

For information, see [About Policy Versions \(v\\* and p\\*\), on page 553](#).

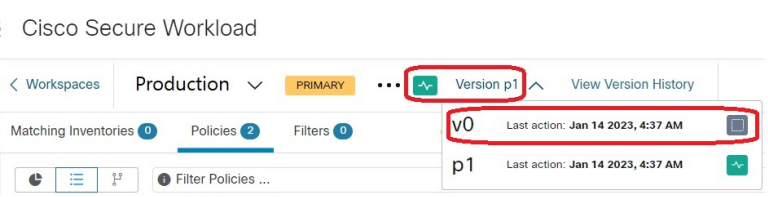
### Procedure

- Step 1** Click **Defend > Segmentation**.
- Step 2** Navigate to the workspace.
- Step 3** Click **Manage Policies**.
- Step 4** The currently displayed version of the policies generated by automatic policy discovery is shown at the top of the page:



If you have already analyzed or enforced policies, the displayed version may be a policy discovery version, an analyzed policy version, or an enforced version.

**Step 5** Do one of the following:

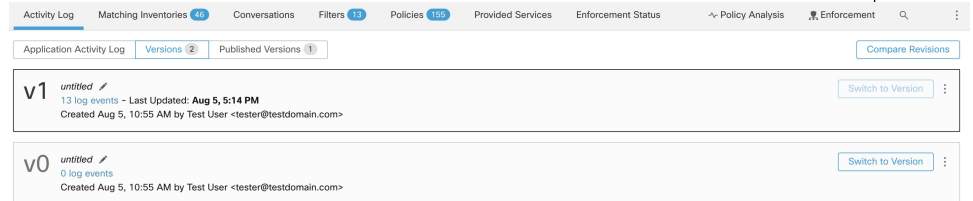
|                                                                                             |                                                                                                                                                                                                                                                                                                                                                                              |
|---------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <p>Display a different version of the policies generated by automatic policy discovery:</p> | <p>Click the current version and choose a different v* version.<br/>(If you see p* versions, those are analyzed and/or enforced versions, not versions of discovered policies.)</p>  <p><b>Important!!</b> See the caveat in the What To Do Next section at the end of this procedure.</p> |
|---------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

View details about a version

- a. Click **View Version History** at the top of the page beside the current version.
- b. Click the **Versions** tab to see the versions of discovered policies. (Not the Published Versions tab.)

The list of versions displays:

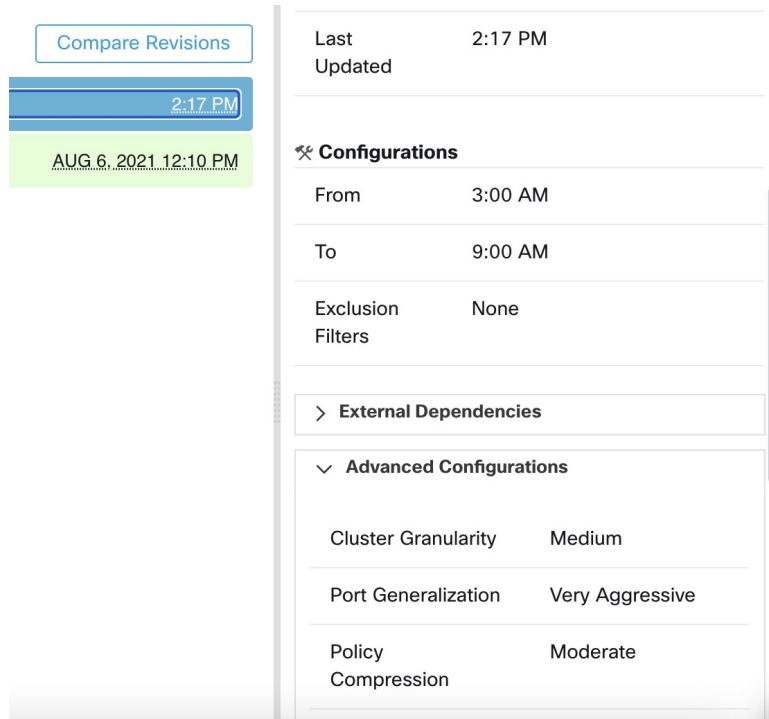
**Figure 258: List of generated policy versions with summary information**



- c. Click the **log events** link in the version.
- d. Click a link in an event row.



Available details include statistics, exclusion filters, external dependencies, and configurations for the run.

**Figure 259: Configurations used for particular automatic policy discovery runs**



Compare two versions to see what has changed:

- a. Click **Compare Revisions**.
- b. Choose the versions to compare.
- c. For result details, see [Comparison of Policy Versions: Policy Diff](#), on page 556.

|                             |                                                                                                                                                                                                                                          |
|-----------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Delete an unwanted version: | Click  for the version and choose <b>Delete</b> .<br>You cannot delete the last remaining version generated by automatic policy discovery (v* version). |
| Export a version:           | Click  for the version and choose <b>Export...</b>                                                                                                      |

### What to do next



- Important** If you want to preserve previous versions of the discovered policies, always display the current version of the discovered policies when you are done working with older versions.
- If the most current version of the discovered policies is not displayed the next time you discover policies for this workspace, older versions may be deleted.
- For example, if the most current version of discovered policies is v4, and v2 is displayed when you discover policies again, then the existing v3 and v4 will be deleted and the new discovered policy version will be v3.
- This behavior ensures a linear version history, which simplifies reverting to a previous version if desired.
- In addition, you can manually create policies only if the latest v\* version is displayed.

## Policy Discovery Kubernetes Support

Policy discovery uses the information on pods and services from Kubernetes configuration to create clusters for both pods and services and the respective policies are generated.

If the Cluster Granularity is set to COARSE or VERY COARSE, then the services and the pods backing them is clustered together.

The screenshot shows a policy discovery interface for a cluster named 'replicaset-zeta'. The main view displays a diagram of the cluster structure with a central 'replicaset-zeta' node connected to several other components: 'deployment-alpha-78d9f9865f', 'rc-epsilon', 'daemonset-gamma', and 'statefulset-beta'. The right-hand panel provides details for the cluster, including its name, description, confidence level, and a query. Below the query, there is a table of pods.

| Namespace | Pod Name              | Address       |
|-----------|-----------------------|---------------|
| standard  | replicaset-zeta-xkmb  | 172.16.84.132 |
| standard  | replicaset-zeta-gnddc | 172.16.247.39 |
| standard  | replicaset-zeta-7kb7z | 172.16.247.36 |

If the Cluster Granularity is set to Medium or Fine or very fine, then the services, and the pods backing them is clustered separately.

This screenshot shows a more granular view of the 'replicaset-zeta' cluster. The diagram includes additional components like 'service-alpha-http', 'service-alpha-tcp', 'service-gamma', 'service-beta', and 'service-epsilon'. The right-hand panel shows the same cluster details as the previous screenshot, but with a different set of pods listed in the table.

| Namespace | Pod Name              | Address       |
|-----------|-----------------------|---------------|
| standard  | replicaset-zeta-7kb7z | 172.16.247.36 |
| standard  | replicaset-zeta-xkmb  | 172.16.84.132 |
| standard  | replicaset-zeta-gnddc | 172.16.247.39 |

For pod clusters, the source information is added as part of the cluster description and each cluster in the description contain the information of which entity has caused the cluster to be formed.

For example, **Description:** “The cluster was formed from the following sources: ReplicaSet name: replicaset-zeta”.



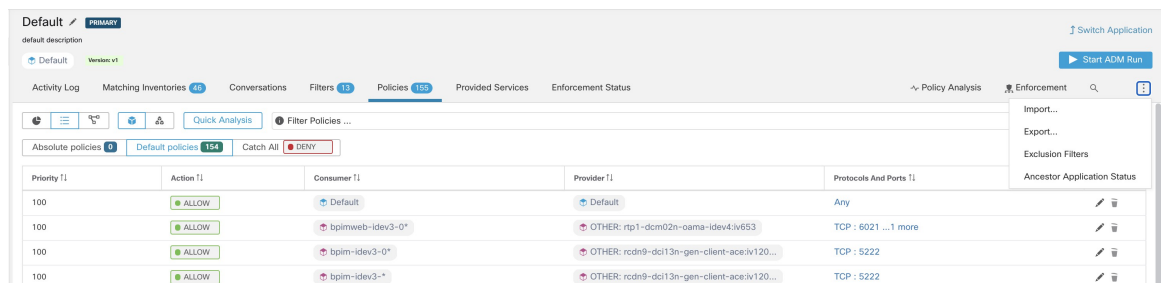
# Import/Export

## Export a Workspace

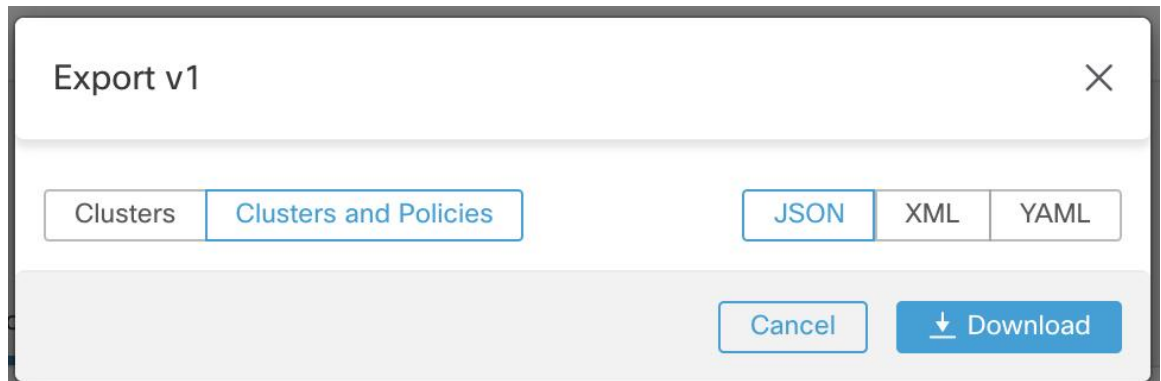
All the relevant contents of clusters and policies in each workspace can be downloaded as a single file in a number of popular structured document formats like JSON, XML and YAML. One can use such files for further in-house processing or ingestion by other policy enforcement or analysis tools.

Navigate to the **...** menu item on the workspace header and click on the **export** item. This will show the export dialog. You can choose whether the exported file should include only the cluster contents or cluster contents as well as the security policies among the clusters generated by automatic policy discovery based on real network flows. Choose the desired format and click download to download the file into the local file system.

**Figure 260: Import/Export menu items**



**Figure 261: Exporting Policies of a workspace**



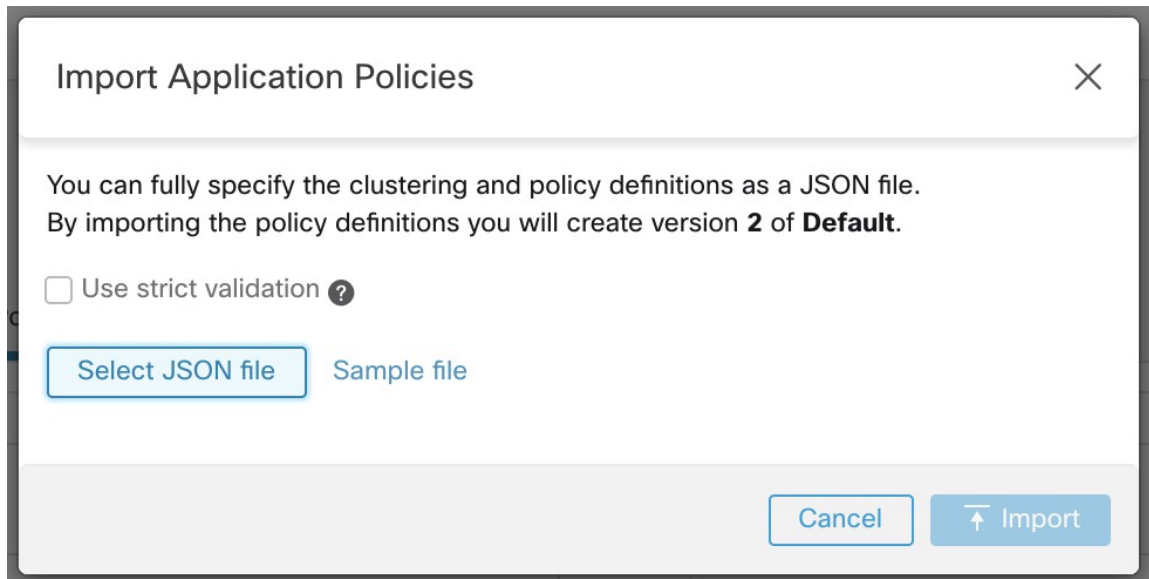
When you export a workspace, the "Auto accept outgoing policy connectors" setting in the automatic policy discovery configuration is included and will be active in the imported workspace.

## Import

You can import known cluster and policy definitions into a workspace by directly uploading a JSON file. Similar to automatic policy discovery, uploading policies into an existing workspace creates a new version and places the cluster and policy definitions under the new version. Missing filters and incorrect property values will return an error.

Click on the **Import** menu item from the **...** menu in the workspace header. In the import dialog, you can select a JSON file with a valid format. A small sample JSON file demonstrating the schema for policies and clusters can be found by clicking on the **Sample** button.

Figure 262: Importing Clusters/Policies



**Strict Validation** if enabled, will return an error if the JSON contains unrecognized attributes. This is useful for locating typos or incorrectly identified optional fields.



**Note** All imported policies are marked as approved by default unless explicitly marked as `approved: false`. You have the option to maintain such approved policies during automatic policy discovery to generate a new set of policies. See [Approved Policies, on page 463](#) for more info.

**Pro Tip:** The schema of the JSON file retrieved by exporting an application workspace is schema-compatible with the expected format for importing policies into a workspace. Therefore, you can clone policies from one application workspace to another using an export followed by an import. Note that many features may not work the same when exporting and then importing policies. For example, the conversations backing the policies are not included in the export and will not be present when importing the policies either.

## Platform-Specific Policies

For important details about how agents enforce policy on each platform, see [Policy Enforcement with Agents, on page 41](#). For Kubernetes/OpenShift, see [Enforcement on Containers, on page 546](#).

### Windows

#### Recommended Windows OS-Based Policy Configuration

Always specify ports and protocols in policies when possible; we recommend not to allow ANY port, ANY protocol.

For example, a generated policy with port and protocol restrictions might look like this:

```
dst_ports {
```

```

 start_port: 22
 end_port: 22
 consumer_filters {
 application_name: "c:\\test\\putty.exe"
 }
 }}
ip_protocol: TCP

```

In contrast, if you allow network connections that are initiated by iperf.exe with ANY protocol and ANY port, the generated policy looks like this:

```

match_set {
 dst_ports {
 end_port: 65535
 consumer_filters {
 application_name: "c:\\test\\iperf.exe"
 }
 }
 address_family: IPv4
 inspection_point: EGRESS
 match_comment: "PolicyId=61008290755f027a92291b9d:61005f90497d4f47cedacb86:"
}

```

For the above filter, Secure Workload creates a policy rule to allow the network traffic on the provider as follows:

```

match_set {
 dst_ports {
 end_port: 65535
 }
 address_family: IPv4
 inspection_point: INGRESS
 match_comment: "PolicyId=61008290755f027a92291b9d:61005f90497d4f47cedacb86:"
}

```

This network rule opens all the ports on the Provider. We strongly recommend not to create OS-based filters with *Any* protocol.

## Configure Policies for Windows Attributes

For more granularity when enforcing a policy on Windows-based workloads, you can filter network traffic by:

- Application Name
- Service Name
- User Names with or without User Groups

This option is supported in both WAF and WFP modes. Windows OS-based filters are categorized as *consumer filters* and *provider filters* in the generated network policy. The Consumer filters filter the network traffic that is initiated on the consumer workload and Provider filters filter the network traffic that is destined for the provider workload.

### Before you begin

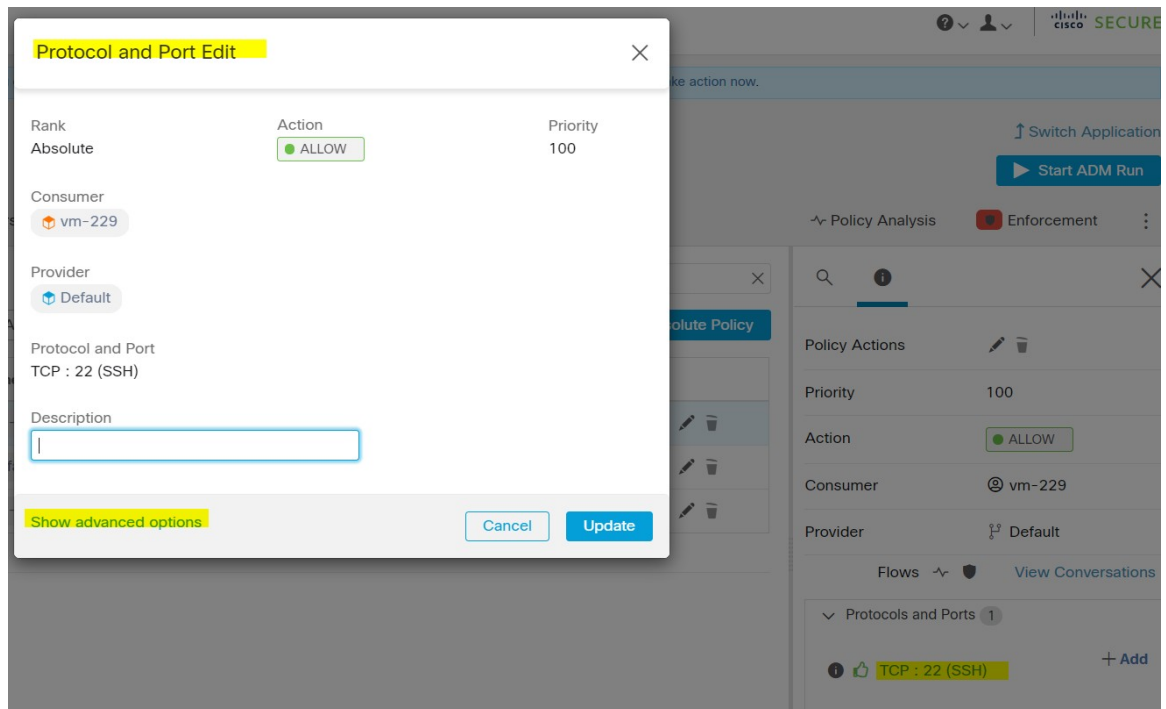
This procedure assumes you are modifying an existing policy. If you have not yet created the policy to which you want to add a Windows OS-based filter, create that policy first.



**Important** See [Caveats, on page 52](#) and [Known limitations, on page 51](#) for policies involving Windows attributes.

## Procedure

- Step 1** In the navigation pane, click **Defend > Segmentation**.
- Step 2** Click the scope that contains the policy for which you want to configure Windows OS-based filters.
- Step 3** Click the workspace in which you want to edit the policy.
- Step 4** Click **Manage Policies**.
- Step 5** Choose the policy to edit.
- Important** Consumer and Provider must include only Windows workloads.
- Step 6** In the table row for the policy to edit, click the existing value in the **Protocols and Ports** column.
- Step 7** In the pane on the right, click the existing value under **Protocols and Ports**.
- In the example, click **TCP : 22 (SSH)**.



- Step 8** Click **Show advanced options**.

While using process level controls a consumer/provider scope or filter should only contain Windows agents. Otherwise, non-Windows OSs (Linux, AIX) will skip the policy and report a sync error in Enforcement Status. See the [user guide](#) for more information.

Consumer Service

Consumer Binary Path

Consumer Users or User Groups ⓘ

Provider Service

Provider Binary Path

Provider Users or User Groups ⓘ

Hide advanced options

- Step 9** Configure consumer filters based on Application name, Service name, or User name.
- The application name must be a full pathname.
  - Service name must be a short service name.
  - User name can be a local user name (For example, `tetter`) or domain user name (For example, `sensor-dev@sensor-dev.com` or `sensor-dev\sensor-dev`)
  - User group can be local user group (For example, `Administrators`) or domain user group (For example, `domain users\\sensor-dev`)
  - Multiple user names and/ or user group names can be specified, separated by ",".(For example, `sensor-dev\@sensor-dev.com,domain users\\sensor-dev`)
  - Service name and User name cannot be configured together.
- Step 10** Configure provider filters based on Application name, Service name, or User name.  
 Follow the same guidelines as given for consumer filters in the previous step.
- Step 11** Enter the paths to the binary, as applicable.  
 For example, enter `c:\test\putty.exe`
- Step 12** Click **Update**.

### Known limitations

- Windows 2008 R2 does not support Windows OS based filtering policies.

- Network policy can be configured with a single user name whereas MS Firewall UI supports multiple users.

### Caveats

- While using the Windows OS-based policies, a consumer/ provider scope or filter should only contain Windows agents. Otherwise, non-Windows OSs (Linux, AIX) skip the policy and report a sync error in Enforcement Status.
- Avoid creating Windows OS filters with *loose* filtering criteria. Such criteria may open unwanted network ports.
- If OS filters are configured for consumer, then the policies are applicable only to consumer, similarly if it is configured for provider then it is applicable only to provider.
- Due to limited or no knowledge of the process context, user context or service context of the network flows, there will be discrepancy in the policy analysis if the policies have Windows OS-based filters.

### Verify and Troubleshoot Policies with Windows OS-Based Filtering Attributes

If you use Windows OS-based filtering attributes, the following topics provide you with verification and troubleshooting information.

Cisco TAC can use this information as needed to troubleshoot such policies.

### Policies Based on Application Name

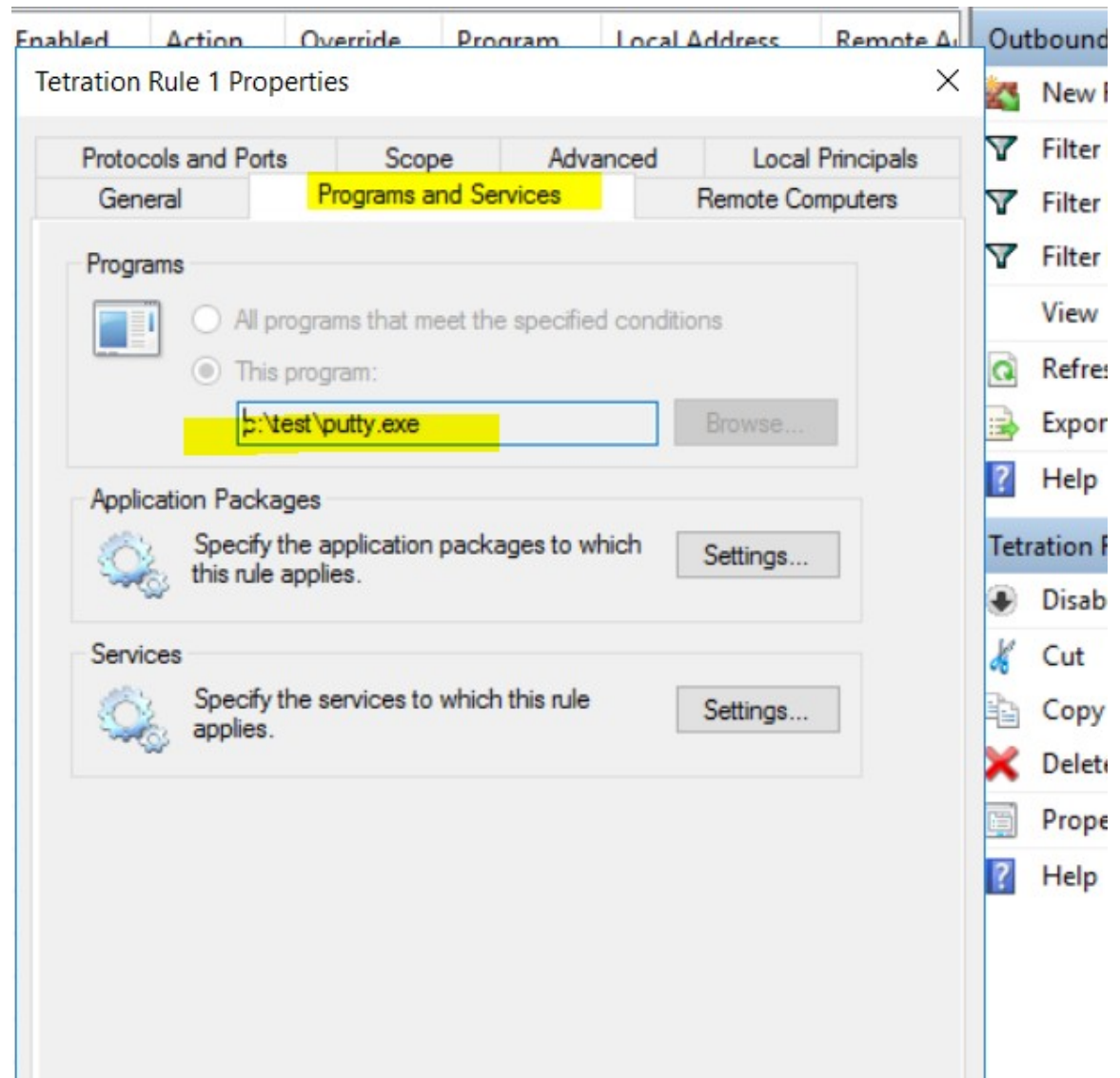
Use the following information to verify and troubleshoot policies based on application name on Windows OS workloads.

The following sections describe the way policies should appear on the workload for an application binary entered as **c:\test\putty.exe**.

#### Sample Policy Based on Application Name

```
dst_ports {
 start_port: 22
 end_port: 22
 consumer_filters {
 application_name: "c:\test\putty.exe"
 }
}
ip_protocol: TCP
address_family: IPv4
inspection_point: EGRESS
```

## Generated Firewall Rule



## Generated Filter Using netsh

To verify, using native Windows tools, that a filter has been added to an advanced policy:

- With administrative privileges, run `cmd.exe`.
- Run `netsh wfp show filters`.
- The output file, **filters.xml**, is generated in the current directory.
- Check `FWPM_CONDITION_ALE_APP_ID` for the application name in the output file: `filters.xml`.

```
<fieldKey>FWPM_CONDITION_ALE_APP_ID</fieldKey>
 <matchType>FWP_MATCH_EQUAL</matchType>
 <conditionValue>
 <type>FWP_BYTE_BLOB_TYPE</type>
```

```

 <byteBlob>
 <data>
 .→5c006400650076006900630065005c0068006100720064006400690073006b0076006f006
 .→</data>
 <asString>\device\harddiskvolume2\temp\putty.exe</
 .→asString>
 </byteBlob>
 </conditionValue>

```

### Generated WFP Filter Using `tetenf.exe -l -f`

```

Filter Name: Secure Workload Rule 1

EffectiveWeight: 18446744073709551592
LayerKey: FWPM_LAYER_ALE_AUTH_CONNECT_V4
Action: Permit
RemoteIP: 10.195.210.15-10.195.210.15
Remote Port: 22
Protocol: 6
AppID: \device\harddiskvolume2\test\putty.exe

```

### Invalid Application Name

- In WAF mode, Firewall rule is created for an invalid application name.
- In WFP mode, the WFP filter is not created for an invalid application name but the NPC is not rejected. The agent logs a warning message and configures the rest of the policy rules.

## Policies Based on Service Name

Use the following information to verify and troubleshoot policies based on Service name on Windows OS workloads.

The following sections describe the way that the policies should appear on the workload.

### Sample Policy Based on Service Name

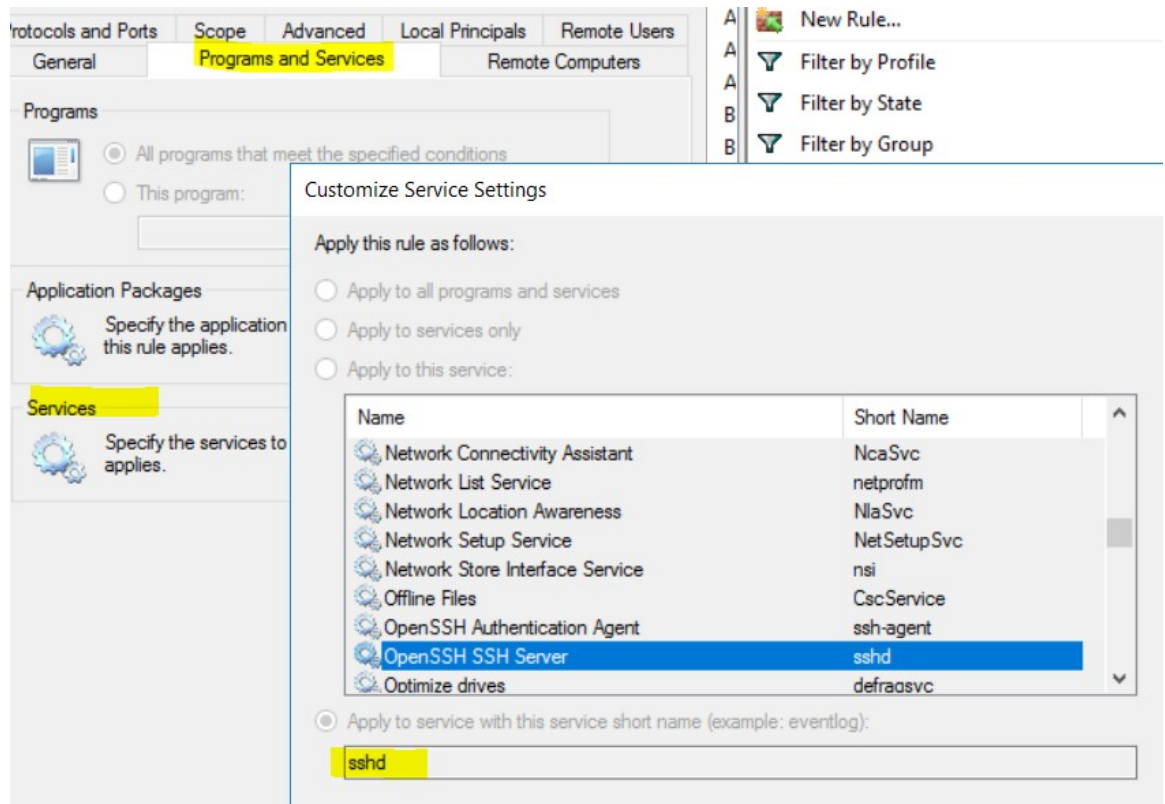
```

dst_ports {
 start_port: 22
 end_port: 22
 provider_filters {
 service_name: "sshd"
 }
}
ip_protocol: TCP
address_family: IPv4
inspection_point: INGRESS

```



### Generated Firewall Rule



### Generated Filter Using netsh

To verify using native Windows tools, that a filter has been added for an advanced policy:

- With administrative privileges, run `cmd.exe`.
- Run `netsh wfp show filters`.
- The output file, **filters.xml**, is generated in the current directory.
- Check `FWPM_CONDITION_ALE_USER_ID` for user name in the output file: filters.xml.

```
<item>
 <fieldKey>FWPM_CONDITION_ALE_USER_ID</fieldKey>
 <matchType>FWP_MATCH_EQUAL</matchType>
 <conditionValue>
 <type>FWP_SECURITY_DESCRIPTOR_TYPE</type>
 <sd>O:SYG:SYD: (A;;CCRC;;;S-1-5-80-3847866527-469524349-687026318-
 →516638107)</sd>
 </conditionValue>
</item>
```

### Generated WFP Filter Using tetenf.exe -l -f

```
Filter Name: Secure Workload Rule 3

EffectiveWeight: 18446744073709551590
```

```
LayerKey: FWPM_LAYER_ALE_AUTH_RECV_ACCEPT_V4
Action: Permit
Local Port: 22
Protocol: 6
User or Service: NT SERVICE\sshd
```

### Invalid Service Name

- In WAF mode, the Firewall rule is created for a nonexistent service name.
- In WFP mode, the WFP filter is not created for a nonexistent service name.
- Service SID type must be *Unrestricted* or *Restricted*. If the service type is *None*, the Firewall Rule and WFP filter can be added but they have no effect.

To verify the SID type, run the following command:

```
sc qsidtype <service name>
```

### Policies Based on User Group or User Name

Use the following information to verify and troubleshoot policies based on user name (with and without user group name) on Windows OS workloads.

Sections in this topic describe the way that the policies should appear on the workload.

Examples in this topic are based on policies that are configured with the following information:

Figure 263: Policies Based on User Group or User Name

Description

While using process level controls, a consumer/provider scope or filter should only contain Windows agents. Otherwise, non-Windows OSs (Linux, AIX) will skip the policy and report a sync error in Enforcement Status. See the [user guide](#) for more information.

Consumer Service

Consumer Binary Path

Consumer Users or User Groups ⓘ  
sensor-dev\domain users,sensor-dev@se

Provider Service

Provider Binary Path

Provider Users or User Groups ⓘ

### Sample Policy Based on User Name

```
dst_ports {
 start_port: 30000
 end_port: 30000
 provider_filters {
 user_name: "sensor-dev\sensor-dev"
 }
}
ip_protocol: TCP
address_family: IPv4
inspection_point: EGRESS
```

### Sample Policy Based on User Group and User Name

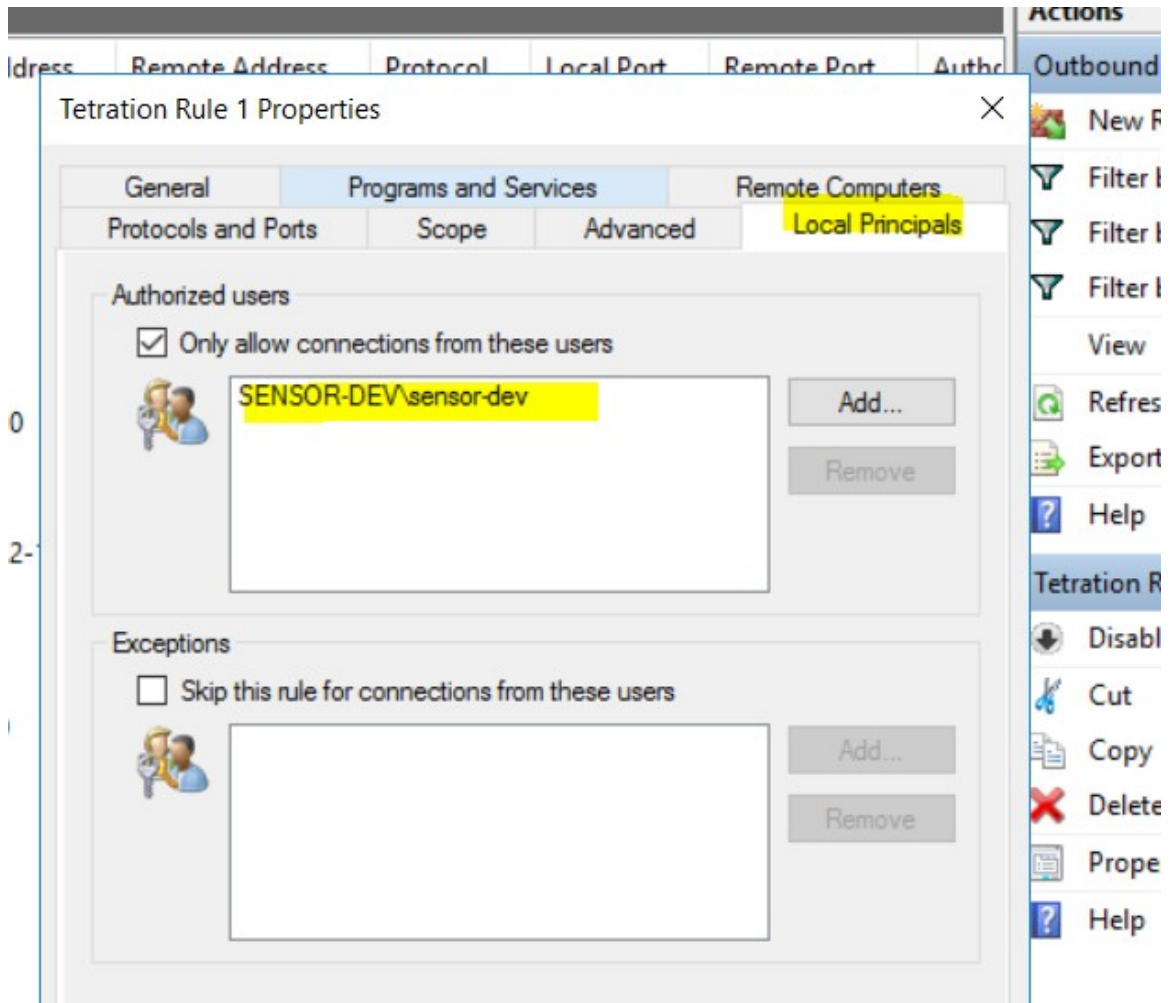
```
dst_ports {
 start_port: 30000
 end_port: 30000
 provider_filters {
 user_name: "sensor-dev\domain users,sensor-dev\sensor-dev"
 }
}
ip_protocol: TCP
```

```
address_family: IPv4
inspection_point: EGRESS
```

### Generated Firewall Rule

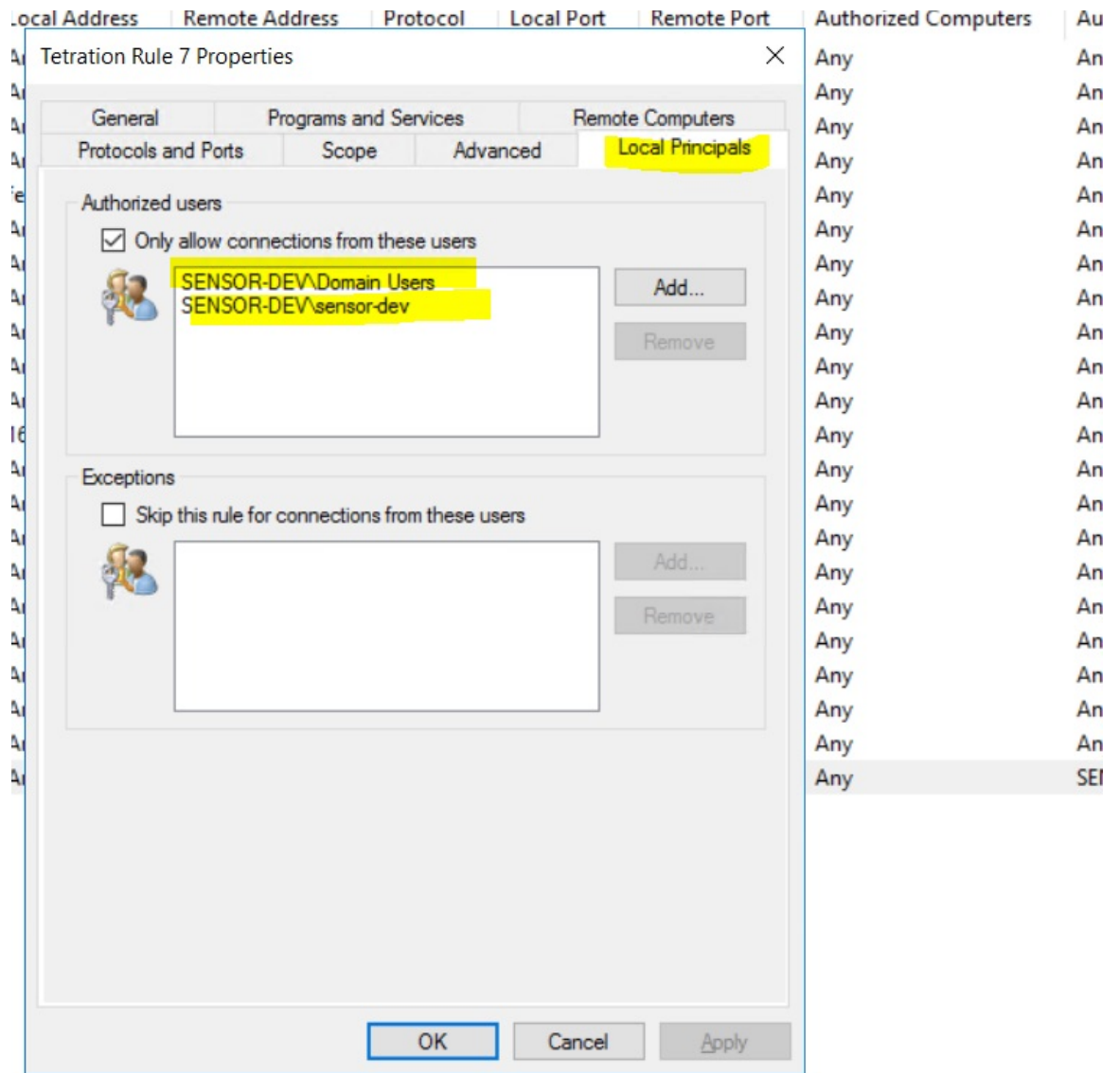
#### Firewall Rule Based on User Name

Example: Firewall rule based on User Name, sensor-dev\\sensor-dev



#### Firewall Rule Based on User Group and User Name

Example: Firewall rule based on User Name, sensor-dev\\sensor-dev and user group, domain users\\sensor-dev



### Generated Filter Using netsh

To verify using native Windows tools that a filter has been added for an advanced policy:

- With administrative privileges, run `cmd.exe`.
- Run `netsh wfp show filters`.
- The output file, **filters.xml**, is generated in the current directory.
- Check `FWPM_CONDITION_ALE_USER_ID` for user name in the output file: `filters.xml`.

```
<item>
 <fieldKey>FWPM_CONDITION_ALE_USER_ID</fieldKey>
 <matchType>FWP_MATCH_EQUAL</matchType>
 <conditionValue>
 <type>FWP_SECURITY_DESCRIPTOR_TYPE</type>
```

```

 <sd>O:LSD: (A;;CC;;;S-1-5-21-4172447896-825920244-2358685150) </sd>
 </conditionValue>
</item>

```

### Generated WFP Filters Using `tetenf.exe -l -f`

#### Filter based on User Name

Example: WFP Rule based on User Name, SENSOR-DEV\sensor-dev

```

Filter Name: Secure Workload Rule 1

EffectiveWeight: 18446744073709551590
LayerKey: FWPM_LAYER_ALE_AUTH_CONNECT_V4
Action: Permit
RemoteIP: 10.195.210.15-10.195.210.15
Remote Port: 30000
Protocol: 6
User or Service: SENSOR-DEV\sensor-dev

```

#### Filter based on User Group and User Name

Example: WFP Rule based on User Name, SENSOR-DEV\sensor-dev and User Group name, SENSOR-DEV\Domain Users

```

Filter Name: Secure Workload Rule 1

EffectiveWeight: 18446744073709551590
LayerKey: FWPM_LAYER_ALE_AUTH_CONNECT_V4
Action: Permit
RemoteIP: 10.195.210.15-10.195.210.15
Remote Port: 30000
Protocol: 6
User or Service: SENSOR-DEV\Domain Users, SENSOR-DEV\sensor-dev

```

*Service name and user name cannot be configured for a Network policy rule.*




---

**Note** The network policy is rejected by the Windows agent if the user name or the user group is invalid.

---

## Kubernetes and OpenShift

### (Optional) Additional Policies for Kubernetes Workloads

The following procedures are optional, depending on your Kubernetes environment.

#### *Policies for Kubernetes Nginx Ingress Controller Running in Host-network Mode*

Secure Workload enforces policies both at the nginx ingress controller and at the backend pods when the pods are exposed to the external clients using Kubernetes ingress object.




---

**Note** If the ingress controller is not running in host network mode refer IngressControllerAPI

---



**Note** IBM-ICP uses Kubernetes Nginx Ingress controller by default and runs on control plane nodes in host network mode.

Following are the steps to enforce the policy using the Kubernetes Nginx Ingress controller.

### Procedure

**Step 1** Create an external orchestrator for Kubernetes/OpenShift as described here.

```
→ ~
→ ~ k8s get ingress
NAME HOSTS ADDRESS PORTS AGE
test-ingress * 192.168.60.100 80 7s
```

**Step 2** Create an ingress object in the Kubernetes cluster. A snapshot of the yaml file used to create the ingress object is provided in the following picture.

```
▶ k8s get ingress
NAME HOSTS ADDRESS PORTS AGE
svc-ce2e-teeksitlbiwlc * 192.168.10.13 80 74s
```

```

~
▶ k8s get ingress -o yaml
apiVersion: v1
items:
- apiVersion: extensions/v1beta1
 kind: Ingress
 metadata:
 annotations:
 virtual-server.f5.com/ip: 192.168.10.13
 virtual-server.f5.com/partition: k8scluster
 creationTimestamp: "2020-06-26T21:31:01Z"
 generation: 1
 labels:
 e2e-test: "yes"
 name: svc-ce2e-teeksitlbwlc
 namespace: default
 resourceVersion: "1074475"
 selfLink: /apis/extensions/v1beta1/namespaces/default/ingresses/svc-ce2e-teeksitlbwlc
 uid: 5526b4a3-b7f4-11ea-aa09-525400d58002
 spec:
 backend:
 serviceName: svc-ce2e-teeksitlbwlc
 servicePort: 80
 status:
 loadBalancer:
 ingress:
 - ip: 192.168.10.13
 kind: List
 metadata:
 resourceVersion: ""
 selfLink: ""

```

**Step 3** Deploy Kubernetes Nginx Ingress controller in the Kubernetes cluster. IBM-ICP Ingress controller pods are running on control plane nodes by default.

```

~
▶ k8s get pods -o wide -n ingress-nginx
NAME READY STATUS RESTARTS AGE IP NODE NOMINATED NODE
nginx-ingress-controller-6bc9c6745c-scfzs 1/1 Running 0 2m11s 192.168.10.13 enforcement-scale-16-kube3 <none>

~
▶ k8s get node enforcement-scale-16-kube3 -o wide
NAME STATUS ROLES AGE VERSION INTERNAL-IP EXTERNAL-IP OS-IMAGE KERNEL-VERSION CONTAINER-RUNTIME
enforcement-scale-16-kube3 Ready <none> 7d5h v1.12.3 192.168.10.13 <none> Ubuntu 16.04.5 LTS 4.4.0-139-generic docker://18.6.1

```

**Step 4** Create a backend service which will be accessed by the consumers outside the cluster. In the example provided below we have created a simple *svc-ce2e-teeksitlbwlc* (http-echo) service.

```

~
▶ k8s get svc svc-ce2e-teeksitlbwlc
NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE
svc-ce2e-teeksitlbwlc ClusterIP 10.102.30.231 <none> 80/TCP 6m11s

```

**Step 5** Create a policy between external consumer and backend service.



| Priority | Action | Consumer                      | Provider | Protocols And Ports |
|----------|--------|-------------------------------|----------|---------------------|
| 100      | ALLOW  | OTHER: RCDN9-DCI03N-ACE-Clien | Default  | TCP : Any           |

Scope: **Default**

Full Name: Default

Primary App: Tetration

Query: VRF ID = 1

[View Scope Details](#)

> Workloads ?

> IP Addresses ?

**Step 6** When you are ready, enforce the policy.

**Step 7** In case of Nginx ingress controller Secure Workload software applies the appropriate allow/drop rule where the source will be consumer specified in the above step and destination will be corresponding Ingress controller pod IP. In case of backend pods, Secure Workload software will apply the appropriate allow/drop rule where the source will be Ingress pod and destination will be the backend pod IP.

### *Policies for Kubernetes Nginx/Haproxy Ingress controller running as Deployment/Daemonset*

Secure Workload will enforce policies both at the ingress controller and at the backend pods when the pods are exposed to the external clients using Kubernetes ingress object.

Following are the steps to enforce policies on Ingress controller.

#### **Procedure**

- Step 1** Create/Update an external orchestrator for Kubernetes/OpenShift using OpenAPI. See [Orchestrators](#) for information on creating the external orchestrator using OpenAPI. Add information of Ingress Controllers for External Orchestrator config.
- Step 2** Create an ingress object in the Kubernetes cluster.
- Step 3** Deploy Ingress controller in the Kubernetes cluster.
- Step 4** Create a backend service which will be accessed by the consumers outside the cluster
- Step 5** Create a policy between external consumer and backend service.
- Step 6** When you are ready, enforce the policy.
- Step 7** In case of Ingress controllers Secure Workload software will apply the appropriate allow/drop rule where the source will be consumer specified in the above step and destination will be corresponding Ingress controller pod IP. In case of backend pods, Secure Workload software will apply the appropriate allow/drop rule where the source will be Ingress pod and destination will be the backend pod IP.

## **Grouping Workloads: Clusters and Inventory Filters**

Clusters and inventory filters serve similar purposes, but have some important differences:

Table 25: Comparison of Clusters and Inventory Filters

| Clusters                                                                                                                                                                                                                                                                                                                                   | Inventory Filters                                                                                                                                                                                                                    |
|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Are used to apply policy to a subset of the workloads in a scope.                                                                                                                                                                                                                                                                          | Can be used to apply policy to a subset of the workloads in a scope.<br><br>Can also be used to apply policy to workloads regardless of scope (for example, to apply policy to all workloads running a particular operating system.) |
| Are defined by a query                                                                                                                                                                                                                                                                                                                     | Are defined by a query.                                                                                                                                                                                                              |
| Can include only workloads in a single scope.                                                                                                                                                                                                                                                                                              | Can have membership restricted to a single scope or include workloads in any scope (for example, if the filter is based on operating system.)                                                                                        |
| Can only be used by policies in the same workspace and workspace version.                                                                                                                                                                                                                                                                  | Can be used by policies in any scope and any workspace.                                                                                                                                                                              |
| Can be automatically created during automatic policy discovery.                                                                                                                                                                                                                                                                            | Must be manually created or converted from an existing cluster.                                                                                                                                                                      |
| Can be overwritten during automatic policy discovery if not approved. Approving known good clusters can improve accuracy of other clusters in future discovery runs.                                                                                                                                                                       | Are never modified by automatic policy discovery.                                                                                                                                                                                    |
| Benefit from important features of automatic policy discovery. They: <ul style="list-style-type: none"> <li>• Have a confidence rating that helps you evaluate whether the workloads in the group belong together.</li> <li>• Can be compared with clusters generated during other policy discovery runs on the same workspace.</li> </ul> | --                                                                                                                                                                                                                                   |
| Cannot be used when configuring <a href="#">External Dependencies, on page 448</a> and other features related to cross-scope policies and policy discovery.                                                                                                                                                                                | Can be used to configure granular policies involving external dependencies and other features related to cross-scope policies, such as auto-pilot rules.                                                                             |
| See <a href="#">Clusters, on page 488</a> and subtopics.                                                                                                                                                                                                                                                                                   | See <a href="#">Create an Inventory Filter, on page 378</a> and <a href="#">Convert a Cluster to an Inventory Filter, on page 493</a>                                                                                                |

## Clusters

A cluster is a set of workloads that are grouped together within a workspace. (A Secure Workload deployment can also be called a cluster, but the two usages are unrelated.)

For example, if your application scope includes several web servers among many other types of servers and hosts that comprise your application, you might want a cluster of web servers within this application scope, so you can assign specific policies only to these web servers.

Automatic policy discovery groups workloads into clusters based on the signals observed in the timeframe that is specified during the run configuration.

### Each cluster is defined by a query

Cluster queries are dynamic unless you define them with specific IP addresses. With dynamic queries, cluster membership can change over time to reflect changes in your inventory: More, fewer, or different workloads can match the query.

For example, if a cluster query is based on hostname containing the substring 'HR', and more hosts with hostname containing HR are added to the workspace, the cluster automatically includes the additional hosts.

Automatic policy discovery examines the hostnames and labels that are associated with workloads. For each cluster, automatic policy discovery generates a short list of candidate queries based on the hostnames and these labels. From these queries, you can select one, possibly edit it, and associate it with the cluster. Note that, in certain cases, when automatic policy discovery cannot formulate simple enough queries based on the hostnames and labels, no (alternate) queries are suggested.

### Workloads in approved clusters are not affected by future policy discovery

Only workloads that are not already members of an approved cluster in the relevant workspace are affected by policy discovery. An **approved cluster** is a cluster that you have manually approved. For details, see [Approving Clusters, on page 496](#).

### Edit clusters to improve grouping

In the following sections, we describe a few workflows to edit, enhance, and approve the clustering results. Note that one can change/approve clusters only in the latest version of a workspace (see [Activity Logs and Version History](#)).

See [Making Changes to Clusters, on page 491](#).

### Clusters involving Kubernetes inventory



---

**Note** If your workspace includes inventory from multiple Kubernetes namespaces, each cluster query must filter by namespace. Add the namespace filter to each query if it is not already present. If you change any query, then automatically discover the policies again.

---

### A cluster may consist of a single workload.

You may want to create policies involving just a single workload.

### Clusters may be converted to inventory filters

Like approved clusters, clusters promoted to inventory filters are not changed during subsequent policy discovery.

Unlike clusters, inventory filters are not tied to a workspace, but are globally available within your Secure Workload deployment.

For a comparison of clusters and inventory filters, see [Grouping Workloads: Clusters and Inventory Filters, on page 487](#).

See [Convert a Cluster to an Inventory Filter, on page 493](#).

## Cluster Confidence

Use the confidence or quality score of a cluster to identify clusters needing improvement.

The confidence for a cluster is the average of the confidences for member workloads. In general, the more similar a workload is to other members of the cluster it was assigned, and the more dissimilar it is to the workloads of the closest (most similar) alternative cluster, the higher the confidence for that workload.

When flows are used for clustering, two workloads are similar when they have a similar pattern of conversations (such as similar sets of neighbors in the conversation graph, i.e., similar sets of consumer and provider workloads and ports).



- 
- Note**
- Cluster confidence is not computed (undefined) for:
    - clusters containing only one workload
    - approved clusters
    - workloads in the scope for which no communication was observed (or no process information is available, if process-based clustering was chosen)
  - Clusters do not span partition boundaries (such as subnet boundaries, see route labels in the advanced automatic policy discovery configurations). However, in computing confidence and alternate cluster, such boundaries are ignored. This indicates the potential existence of workloads or clusters that behave very similarly even though they are in different subnets.
  - After editing clusters, the confidence scores may become inaccurate as they are NOT recomputed until you discover policies again.
- 

To view cluster confidence, see [View Clusters, on page 490](#).

## View Clusters

The clusters view supports query-to-cluster association and query editing.

In the clusters view, you can click a table column heading to sort the clusters based on that column (such as name, the number of workloads, or confidence).

For each cluster, by clicking on its row, you can view further cluster information such as description, suggested or approved queries, and the member workloads in the right panel. Several of these fields are editable.

To view clusters and details about them:

1. Navigate to the scope and workspace of interest.
 

Clusters are specific to a workspace; each workspace in a scope can have different clusters. To make clusters available outside their current workspace, see [Convert a Cluster to an Inventory Filter, on page 493](#).
2. Click **Manage Policies**.
3. Click **Filters**.
4. Click **Clusters**.
5. To view information about a cluster, click a cluster.
  - a. Look in the panel that opens on the right.
  - b. For more details, click **View cluster details**.

The Cluster Details page opens in a separate browser tab.

**Figure 264: Clusters View**

The screenshot displays the 'Clusters View' interface. At the top, there is a navigation bar with tabs for 'Activity Log', 'Matching Inventories' (46), 'Conversations', 'Filters' (13), 'Policies' (154), 'Provided Services', and 'Enforcement Status'. Below the navigation bar is a search bar and a 'Create Cluster' button. The main area shows a table of clusters with columns for 'Name', 'Matching Inventory', 'Confidence', 'Dynamic', and 'Approved'. The cluster 'bpim\* 2' is highlighted. To the right, a sidebar shows details for the selected cluster, including 'Name', 'Description', 'Confidence', and a list of associated Workloads, IP Addresses, Neighbors, and Subnets.

| Name                     | Matching Inventory | Confidence | Dynamic | Approved |
|--------------------------|--------------------|------------|---------|----------|
| bpim*                    | 4                  | N/A        |         |          |
| bpim* 2                  | 4                  | Low        |         |          |
| bpim-idev3-*             | 3                  | N/A        |         |          |
| bpim-idev3-* 2           | 3                  | N/A        |         |          |
| bpim-idev3-0*            | 2                  | Low        |         |          |
| bpim-idev3-07.cisco.com  | 1                  | N/A        |         |          |
| bpim-idev3-201.cisco.com | 1                  | N/A        |         |          |
| bpim-idev3-203.cisco.com | 1                  | N/A        |         |          |
| bpim-idev4-*             | 3                  | N/A        |         |          |
| bpim-idev4-* 2           | 2                  | N/A        |         |          |

## Making Changes to Clusters

Automatic policy discovery creates one or more candidate queries for each cluster.

If clustering results do not completely match your expectations, you can improve the grouping by editing the query.

To browse and edit clusters: Click on the **clusters** box at the top of the page. To change a cluster (e.g. change the members of a cluster or select/change its query), select/edit the cluster's query, as shown below.

Figure 265: Edit Cluster

You can add or remove explicit IP addresses, or pick another query from the list of alternatives provided and edit that query. A cluster's query can be any query filter expressed in terms of addresses, hostnames, and labels. If you define a query based on labels rather than explicit IP addresses, the cluster will be dynamic, and new, changed, or removed inventory that is properly labeled will automatically be included in or excluded from the cluster.

After query selection and possible editing is done, click save. Note that once the SAVE button is clicked, the cluster is automatically marked approved, the approved thumbs-up icon turns blue (whether or not a change was made). The approved icon can be toggled to change the approved status as desired. See details at [Approving Clusters, on page 496](#).



### Important

When a cluster's membership is changed, it may be necessary to discover policies again to get an updated policy accurately reflecting the changes in flows among the changed clusters. This is because cluster memberships may have changed (such as new nodes added to a cluster). A similar situation can occur if the scope corresponding to the workspace is edited or in general when workspace membership changes. Similarly, cluster confidence scores may no longer be accurate with changes to cluster memberships. In all these cases, automatically discovering policies again is useful to get updated policies and cluster confidence scores (updated confidences on unapproved clusters).

If you edit cluster queries, it is possible that clusters associated with queries may overlap.

## Convert a Cluster to an Inventory Filter

Convert a cluster to an inventory filter if:

- You do not want the cluster to be modified by future automatic policy discovery runs, as a more versatile alternative to approving the cluster.
- You want the cluster to be independent of the workspace and workspace version.
- You are creating or discovering policies in which the consumer and provider belong to different scopes, and you want to create policies specific to a subset of workloads in a scope, not just policies involving the entire scope.

You must use inventory filters instead of clusters for this purpose if you create cross-scope policies using the advanced method described in [When Consumer and Provider Are in Different Scopes: Policy Options](#), on page 504 and you want policies to be more granular than scope-to-scope.

### Procedure

---

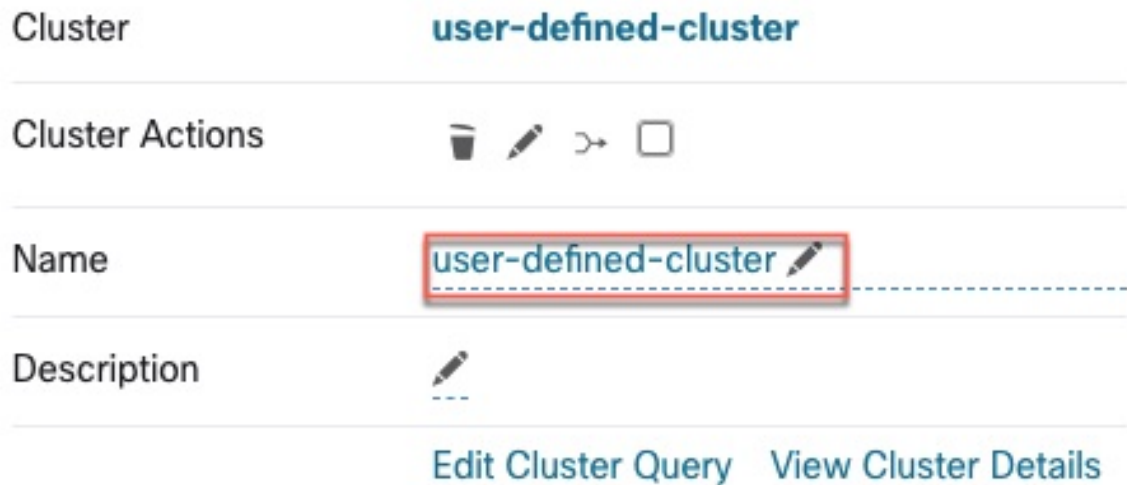
- Step 1** Navigate to the workspace that contains the cluster to promote.
- Step 2** Click **Manage Policies**.
- Step 3** Click **Filters**.
- Step 4** Click **Clusters**.
- Step 5** Click the cluster you want to use in the cross-scope policy.
- Step 6** In the panel on the right, in the **Cluster Actions** section, click ➤ (Promote to Inventory Filter.)
- Step 7** Verify that the name, description, and query are as expected.
- Step 8** Select **Restrict Query to Ownership Scope**.
- (Inventory filters can cross scope boundaries, but you do not want this behavior for this purpose; you want this filter to include only workloads in this scope.)
- Step 9** If you want the application defined by this inventory filter to be the provider in policies generated during automatic policy discovery, select **Provides a service external of its scope**.
- If this application is a consumer rather than a provider, or if you will use this inventory filter only for manually created policies, you don't need to enable this option.
- Step 10** Click **Promote Cluster**.
- Step 11** Verify that the cluster has moved to the **Inventory Filters** tab.
- You may need to refresh the page to see this change.
- 

## Creating or Deleting Clusters

Click the **Create Cluster** button on the clusters page to create a new empty cluster. Alternatively, you can also create a cluster from the automatic policy discovery page by clicking on **Create Filter** button in Get Started sidebar and selecting Clusters in the modal.

**Figure 266: Creating a new Cluster**

The new user defined cluster will show up on the side panel to be renamed, if necessary.

**Figure 267: Renaming a Cluster**

An empty cluster may be deleted by selecting the cluster in any of the views so that the details appear on the side panel and clicking the trash button on the header of cluster detail view. See figure above.

## Comparing Versions of Generated Clusters: Diff Views

After you have automatically discovered policies at least twice for a workspace, you can compare the clusters generated in different discovery runs.

### Procedure

**Step 1** Navigate to the clusters diff view using one of the following paths:

- After successfully discovering policies, a message will appear indicating the success with a link that navigates to the diff view showing discovery results. Click the results link.

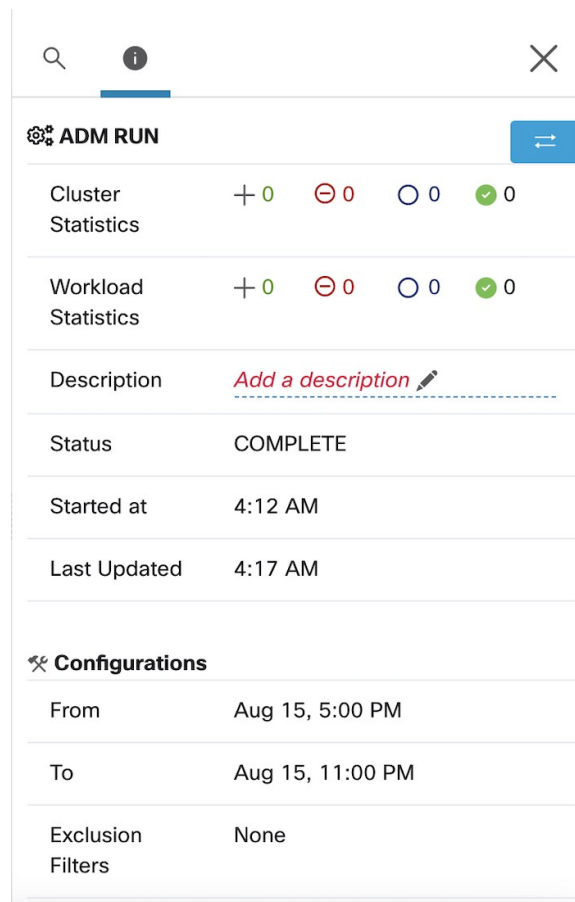
**Figure 268: Successful automatic policy discovery run**

- Compare revisions from the versions view:
  - a. Follow the steps in [View, Compare, and Manage Discovered Policy Versions](#), on page 466.
  - b. After you click **Compare Revisions**, click **Clusters**.



- From the version details side panel:
  - a. Follow the steps to view version details in [View, Compare, and Manage Discovered Policy Versions](#), on page 466.
  - b. From the side panel, when it is showing context information for an automatic policy discovery run, click the double-arrow button on the top right corner of the side panel:

**Figure 269: Showing Context Information**



**Step 2** Choose the versions to compare.

**Step 3** Review the comparison results:

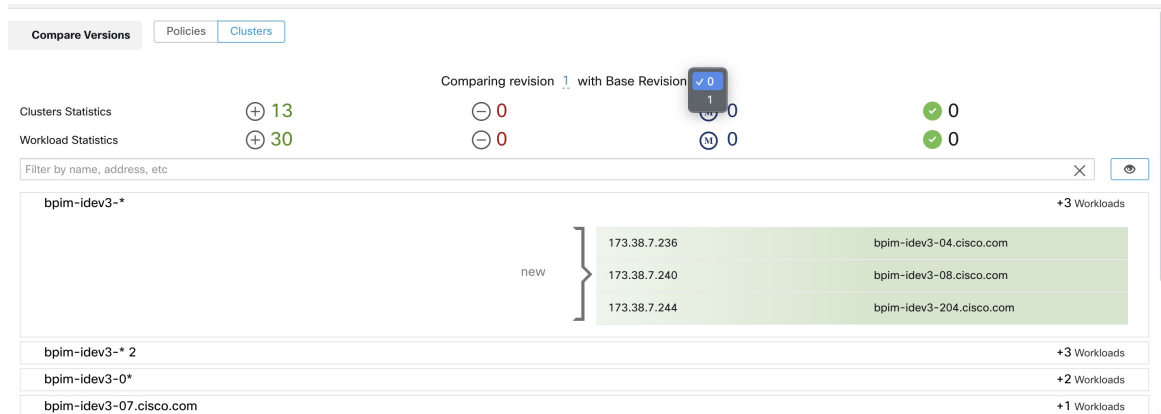
At the top level, the diff view for automatically discovered policies shows high level statistics about changes in clusters and workloads showing the number of added, deleted, modified, and unchanged clusters and workloads.

The rest of the view is organized as a list of clusters in the order of added, deleted, modified and unchanged, each color coded to reflect the status as well as the number of workloads added to or removed from the cluster.

You may search for a particular cluster or workload by name or IP address. To see how the contents of a cluster have changed, click any of the rows representing a cluster to expand that row.

**Note** By default, unchanged clusters are hidden. To display unchanged clusters, click the button with the eye icon.

Figure 270: Cluster Diff View



### What to do next

To view a similar comparison for policies, see [Comparison of Policy Versions: Policy Diff](#).

## Preventing Cluster Modification During Automatic Policy Discovery Reruns

If you do not want automatic policy discovery (formerly known as ADM) to modify a cluster when you automatically discover policies for the workspace in future, approve the cluster.

For example, approve the cluster if you have edited the cluster query, and now you need to add new workloads to the scope and cluster them without affecting the existing policies. Approving the cluster freezes the cluster contents and attributes in the current state. Automatic policy discovery does not change approved clusters.

See [Approving Clusters, on page 496](#).

Alternatively, you can promote the cluster to an inventory filter, which will never be modified by policy discovery. See [Convert a Cluster to an Inventory Filter, on page 493](#).

## Approving Clusters



**Note** See also [Convert a Cluster to an Inventory Filter, on page 493](#), which may be a more appropriate option for your needs.

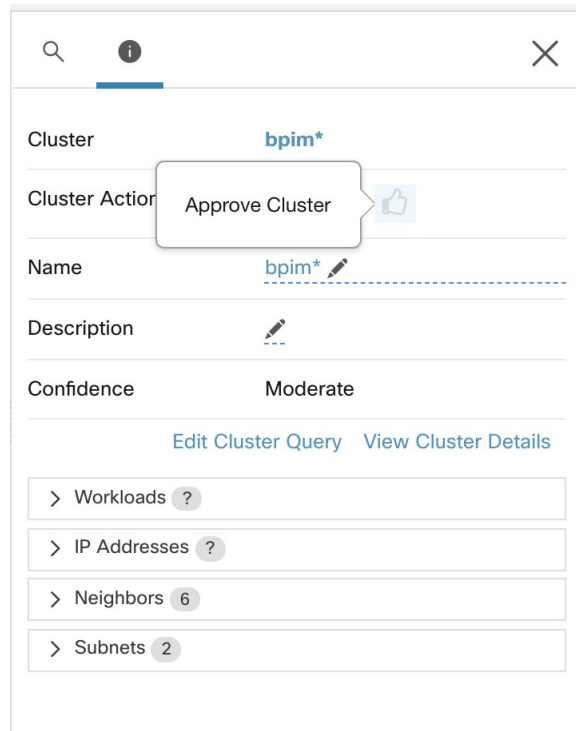
After you approve a cluster, subsequent automatic policy discovery does not change that cluster's query. Memberships of approved clusters can change only if the members of the workspace change.

Workloads that are members of an approved cluster may be referred to as "approved workloads."

To approve a cluster:

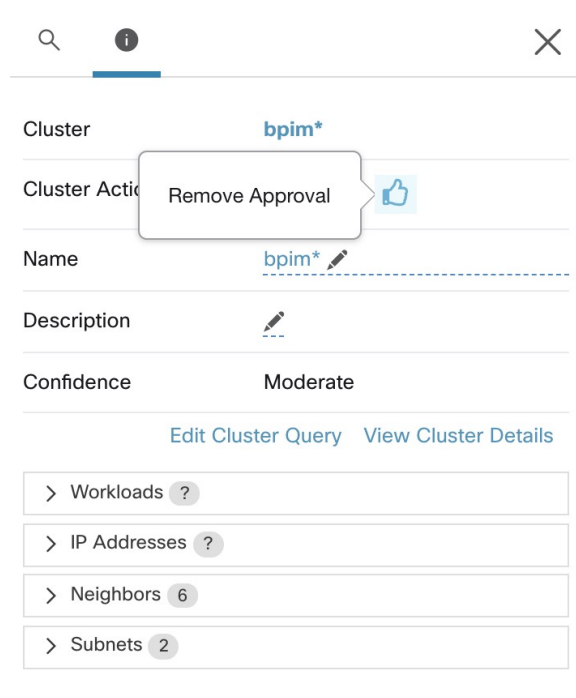
Make sure the cluster of interest is shown on the side panel. You can accomplish this via searching for the cluster, or clicking on the desired cluster on the chart in any of the views. Then select the check box on the top-right corner of the cluster info on the side panel as illustrated below. After a cluster is approved, it indicates that it will remain unchanged by future automatic policy discovery.

Figure 271: Approving Clusters



To remove approval of a cluster, click the approval icon.

Figure 272: Removing Approval of a Cluster



# Address Policy Complexities

Enforcement results are impacted by factors including:

- Rule type and rank:
  - Absolute vs Default policies
  - The catch-all setting for the workspace

See [Policy Rank: Absolute, Default, and Catch-All, on page 425](#).

- Order of policies within the workspace

See [Policy Priorities, on page 498](#).

- Policies inherited from parent or ancestor scopes, including the catch-all rule

You will want to ensure that a higher-priority policy is not hitting traffic before the policy that you expect to hit that traffic.

To see the impact of policies in ancestor scopes, run live policy analysis on all involved scopes. See [Live Policy Analysis, on page 527](#).

When you are ready to enforce the policies in a workspace, a wizard shows you which inherited policies impact workloads in the workspace. For information, see [Policy Enforcement Wizard, on page 544](#).

- Cross-scope policy interactions

(When consumer and provider are in different scopes, or one end of the conversation is in a different scope than the policy)

See [When Consumer and Provider Are in Different Scopes: Policy Options, on page 504](#).

- Situations in which the actual consumer or provider in a policy may differ from the default configured consumer and provider, for example in failover scenarios.

See [Effective Consumer or Effective Provider, on page 517](#).

## Policy Priorities

Traffic handling is affected by:

- The priority of policies within the scope, and
- [Policy Global Ordering and Conflict Resolution, on page 499](#)

### Policy priorities within a scope

Within a workspace, the order of the policies in the list reflects the relative priority of each policy, with the highest priority policy at the top of the list, and the lowest priority policy at the bottom of the list.

In each workspace, Absolute policies have priority over Default policies, and the Catch-All policy is the lowest priority policy in the workspace.

For details, see [Policy Rank: Absolute, Default, and Catch-All, on page 425](#).

## Policy Global Ordering and Conflict Resolution

Conflicts can arise between different policies defined under different scopes. More specifically, conflicts arise for workloads (inventory items) that belong to multiple scopes, such as parent/child, when those scopes have contradictory policies).

It is not feasible to resolve such conflicts manually due to the dynamic nature of scope membership; workloads can enter and leave scopes as their properties change. Therefore, the system imposes a global order, as described below, for all policies according to the scope under which they are defined. For each workload, the list of relevant policies (according to consumer/provider/scope) is identified and sorted by the global order. The decision to permit or drop a flow is made based on the *first* matching policy in the sorted list.

By understanding the global ordering scheme of security policies, network admins can define the correct scopes and their priorities to apply the overall desired policies on workloads. Within each scope, application owners maintain their ability to enforce fine-grained policies on their respective workloads.

A global network policy has the following characteristics:

- A set of scopes ordered by priority (highest priority first).
- Each scope's primary workspace has absolute policies, default policies and a catch-all action.
- Each group of absolute or default policies within each workspace is sorted according to their local priorities (highest first).

The global order of policies is defined as follows:

- Groups of absolute policies from the primary workspaces of all scopes (arranged from highest to lowest priority).
- Groups of default policies from the primary workspaces all scopes (arranged from lowest to highest priority).
- Catch-all policies from all scopes (arranged from lowest to highest priority).

Note that the scope order applies to groups of policies in category 1 and 2, rather than individual policies. Within each group, individual policies with lower policy priority numbers taking precedence.

For a specific workload, first the subset of scopes it belongs to is determined, then the above order is applied. The catch-all policy from the lowest priority (enforced) workspace to which this workload belongs is the applicable catch- all (but an absolute or default policy may override). For a given flow on that workload, the action of the highest matching policy is applied.

**Note**

- If a workspace has neither Absolute nor Default policies defined, the workspace is ignored. The workspace's catch-all policy will not be included in the global order.
- The order of Default policies in the global order is the reverse of the scope priorities. This lets you define broad policies for all scopes to secure the perimeter of all workspaces including those that do not have policy enforcement enabled. At the same time application owners who have enabled enforcement on their scopes have the ability to override these default policies.
- Overlapping scopes are not recommended; see [Scope Overlap, on page 356](#) for details. However, if a workload has two or more interfaces, in overlapping or disjoint scopes, the catch-all policy of the lowest priority workspace with enforcement enabled will apply (among all the applicable catch-all policies).

We expand our previous three-scope example to illustrate this ordering scheme. Assume that the three scopes are assigned the following priorities (See [Use Workspaces to Manage Policies](#) for instruction on how to change scope priorities):

1. Apps
2. Apps:HR
3. Apps:Commerce

The primary workspace of each of these scopes has absolute policies, default policies and a catch-all action. Each group of absolute or default policies within each workspace is sorted according their local priorities.

The global ordering of the policies are as follows:

1. Apps Absolute policies
2. Apps:HR Absolute policies
3. Apps:Commerce Absolute policies
4. Apps:Commerce Default policies
5. Apps:HR Default policies
6. Apps Default policies
7. Apps:Commerce Catch-all
8. Apps:HR Catch-all
9. Apps Catch-all

A workload that belongs to the *Apps* scope will receive only the following policies in the given order:

1. Apps Absolute policies that match the workload
2. Apps Default policies
3. Apps Catch-all

A workload that belongs to the *Apps* and *Apps:Commerce* scopes receive only the following policies in the given order:

1. Apps Absolute policies
2. Apps:Commerce Absolute policies
3. Apps:Commerce Default policies
4. Apps Default policies
5. Apps:Commerce Catch-all

A workload that belongs to the *Apps* and *Apps:HR* scopes will receive only the following policies in the given order:

1. Apps Absolute policies
2. Apps:HR Absolute policies
3. Apps:HR Default policies
4. Apps Default policies
5. Apps:HR Catch-all

### Policy Order and Overlapping Scopes



---

**Important** The following scenario involves overlapping scopes. You should avoid having overlapping sibling scopes – workloads should not be members of multiple branches of the scope tree. For more information, see [Scope Overlap, on page 356](#).

---

A workload that belongs to all three *Apps*, *Apps:HR* and *Apps:Commerce* scopes will receive the following policies in the given order:

1. Apps Absolute policies
2. Apps:HR Absolute policies
3. Apps:Commerce Absolute policies
4. Apps:Commerce Default policies
5. Apps:HR Default policies
6. Apps Default policies
7. Apps:Commerce Catch-all

Note that the relative ordering of the *Apps:HR* and *Apps:Commerce* scopes only matters if the two scopes overlap (that is, there are workloads that belong to both sibling scopes.) This is because policies are always defined under a scope. A workload belonging to one scope only will not be affected by policies from the other scope, thus the order does not matter.

## Validate the Order and Priority of Policies

To validate the order and priority of policies in parent/ancestor workspaces, click the **Analyzed Policies** or **Enforced Policies** tab at the top of the Defend > Segmentation page. These views provide a global view of the analyzed and enforced policies respectively.

**Figure 273: Example: List of enforced policies in their policy priority order**

The screenshot shows the 'Enforced Policies' view in the Segmentation section. It displays a list of policies grouped into Absolute, Default, and Catch-All categories. A search filter is applied to the Absolute Policies section, showing results for '10.103.1.1'. The table lists policy details such as name, version, and last enforcement event.

| Category           | Policy Name        | Version     | Last Enforcement Event |
|--------------------|--------------------|-------------|------------------------|
| Absolute Policies  | Furong:jumphost    | Version p10 | March 3, 2022          |
| Default Policies   | Furong:ipv6-domain | Version p10 | September 10, 2021     |
| Default Policies   | Furong:jumphost    | Version p10 | March 3, 2022          |
| Default Policies   | Furong:App1        | Version p10 | May 13, 2021           |
| Catch-All Policies | Furong:ipv6-domain | Version p10 | DENY                   |

- To limit the list of policies to only those which include a particular scope or filter as a consumer or provider, select a scope or enter a filter.
- Available filters:

| Filter Name      | Definition                                                                    |
|------------------|-------------------------------------------------------------------------------|
| <b>Port</b>      | Policy port to match, e.g. 80.                                                |
| <b>Protocol</b>  | Policy protocol to match, e.g. TCP.                                           |
| <b>Approved</b>  | Matches policies that have been marked as <a href="#">Approved Policies</a> . |
| <b>External?</b> | Policies in which the consumer and provider are in different scopes.          |
| <b>Action</b>    | Policy action: Allow or Deny                                                  |

## (Advanced) Change Policy Priorities



**Caution** Scope policy priority order rarely needs to be changed. Since changing policy priorities can affect enforcement results on all workspaces, change with caution.

Access to this feature is limited to users with very high privilege roles such as site admin.



## Before you begin

Before changing scope priority order:

- Understand policy sorting logic and how policy priorities on scopes translate to ordering of individual policy intents. See [Policy Priorities, on page 498](#).
- Make changes in a secondary workspace until you are confident that your new order will be as expected.
- Plan your changes, considering the following guidelines:

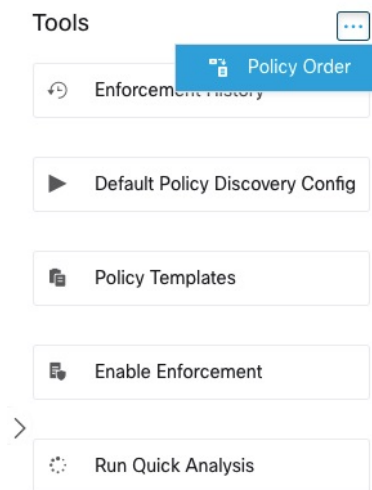
When reordering, keep a parent-first ordering (parent scopes above child scopes) to take advantage of the hierarchical structure of your scope tree.

(If you have overlapping sibling scopes, it may be necessary to reorder sibling scopes and their children. Overlapping sibling scopes are not recommended. Fix these by updating scope queries. See [Scope Overlap, on page 356](#).)

## Procedure

**Step 1** To reorder policy priority, click the menu icon next to **Tools** and select **Policy Order**:

*Figure 274: Navigating to Policy Priorities page*

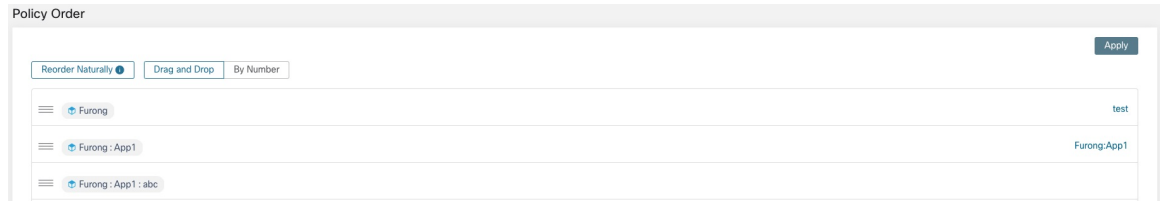


Once on the Policy Order page, you can see the list of all scopes and their corresponding primary workspaces according to the current policy priority.

**Step 2** There are several ways to reorder the scopes:

- To reorder the entire list to place parent scopes above child scopes ("pre-order"): Click **Reorder Naturally**. This is the recommended order and any deviation from this should be done with care.
- To reorder the list manually:
  - Drag the rows up and down.
  - Click **By Number** to set a number for each scope to be used for sorting. This can be easier for large lists.

Figure 275: Setting Policy Priorities for Scopes



### What to do next

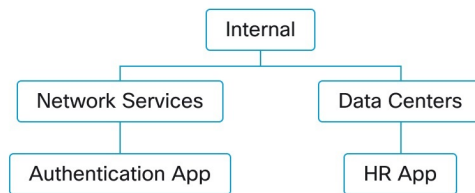
Run Quick Analysis to see the results of your changes.

## When Consumer and Provider Are in Different Scopes: Policy Options

### Example Scenario

The following situation is an example showing cross-scope traffic:

Your scope hierarchy includes a Network Services scope that includes an authentication application (the provider). An HR application, which is a member of a scope on a different branch of the scope hierarchy, is a consumer of the service provided by the authentication application.



### Policy Options

Secure Workload offers several ways to address this situation:

| Option                                                                                                                  | Instructions                                                                                                                                                                                                                                                                                                                                                                                                                                                    | Pros and Cons                                                                                                                                                                                                                                                                                                                                                                      |
|-------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Create these policies in a parent or ancestor scope that includes both consumer and provider as children or descendants | <ul style="list-style-type: none"> <li>Manually create one or more policies in the common-ancestor scope.</li> </ul> <p>(Optional) For more precise policies, group workloads using inventory filters. For examples and instructions, see <a href="#">Create an Inventory Filter, on page 378</a>.</p> <ul style="list-style-type: none"> <li>Automatically discover policies in the common-ancestor scope, for the entire branch of the scope tree.</li> </ul> | <p>These methods are the simplest way to address cross-scope policies.</p> <p>These methods require only one policy per consumer-provider pair.</p> <p>If you are considering using automatic policy discovery, see important considerations in <a href="#">Discover Policies for One Scope or for a Branch of the Scope Tree, on page 439</a>.</p>                                |
| Use the advanced method for creating cross-scope policies                                                               | <p>Automatically discover policies for each individual scope.</p> <p>See <a href="#">(Advanced) Create Cross-Scope Policies, on page 505</a>.</p> <p>(This procedure applies to both manually created policies and discovered policies.)</p>                                                                                                                                                                                                                    | <p>This method requires two policies for each consumer-provider pair: A policy for the consumer and a policy for the provider.</p> <p>This method allows policy creation when consumer and provider policies are owned by different people.</p> <p>See other considerations in <a href="#">Discover Policies for One Scope or for a Branch of the Scope Tree, on page 439</a>.</p> |

## (Advanced) Create Cross-Scope Policies

This procedure describes the advanced method of creating cross-scope policies (policies in which consumer and provider are in different scopes.) It applies to both manually created policies and automatically discovered policies.

This method requires two policies for each consumer-provider pair, because both ends of the conversation must allow the conversation to happen:

- A policy in the consumer's scope must allow conversations with the provider, and
- A policy in the provider's scope must allow conversations with the consumer.

This procedure includes the steps that must be taken by the owner of each scope in order to create cross-scope policies. If your access privileges allow you to modify both scopes, you can perform all steps.

### Before you begin

- Consider simpler options for handling cross-scope traffic. See [When Consumer and Provider Are in Different Scopes: Policy Options, on page 504](#).
- Policies using this method must be created in the primary workspace of both consumer and provider.

If the provider scope to be specified in the policy does not yet have a primary workspace, create it before creating cross-scope policies using this method.

- The policies must have ALLOW action in order for policy requests to be created.
  - For some additional details related to these requirements, see [Policy Requests, on page 506](#).
  - (Optional) Consider options for automatic handling of cross-scope policy requests. See [Automate Handling of Cross-Scope Policy Requests, on page 511](#).
  - (Optional) If you want cross-scope policies to apply only to the workloads in a cluster within the consumer or provider scope, and not to the entire scope, see [Convert a Cluster to an Inventory Filter, on page 493](#). Clusters cannot be used in cross-scope policies created using this procedure.
- If you are discovering policies automatically, see also [External Dependencies, on page 448](#) and [Fine-Tune External Dependencies for a Workspace, on page 449](#).

### Procedure

---

- Step 1** In the consumer's primary workspace, create the desired policy, either manually or using automatic policy discovery.
- For each cross-scope policy created, a policy request will automatically be created for the provider.
- To view the policy requests, see [Viewing, Accepting, and Rejecting Policy Requests, on page 507](#).
- Note: If an existing policy in the provider application's workspace matches this traffic, a new policy is not needed and a request is not created. This situation is indicated as described in [Resolved Policy Requests, on page 515](#).
- Step 2** You (or the owner of the provider application) must respond to each policy request:
- See [Viewing, Accepting, and Rejecting Policy Requests, on page 507](#).
- Accepting a policy request automatically creates the required policy in the primary workspace of the provider, allowing traffic between the two applications.
- If you do not want to allow traffic from the requesting application, reject the request.
- Step 3** (Optional) If you are automatically discovering policies, you may want to [Fine-Tune External Dependencies for a Workspace, on page 449](#).
- Step 4** Review and analyze both primary workspaces.
- 

### What to do next

When you are ready to enforce these policies, you must enforce both primary workspaces.

## Policy Requests

Policy requests are generated when you create cross-scope policies using the method described in [\(Advanced\) Create Cross-Scope Policies, on page 505](#). Each time a policy is created in a consumer scope's primary workspace when the provider is a member of a different scope, if the policy does not yet exist in the primary workspace associated with the provider's scope, a policy request is generated.

This policy request alerts the owner of the provider application to allow dependent applications to access necessary services.

See options for viewing and responding to policy requests at [Viewing, Accepting, and Rejecting Policy Requests, on page 507](#) and [Automate Handling of Cross-Scope Policy Requests, on page 511](#).

#### Additional details about policy requests

- The provided services page (on which policy requests appear) is only available to primary workspaces. This is to ensure that isolated experiments on secondary workspaces do not create notifications on other primary workspaces.
- If an external scope (when the provider specified in the policy belongs to a different scope than the consumer) does not have a primary workspace, no requests are sent (for example, this could be the case for the root scope, or any scope defined for workloads outside the organization). If an external scope has not published any policy, policy analysis and enforcement are carried out on the consumer end only.
- Clusters are not supported when the provider is in a different scope than the consumer. If the policy's consumer is a cluster, the policy request will be made as if the policy request were from the consumer application's scope. Multiple policies consuming the same service from a provider could be grouped together.
- Policy requests are generated only for providers, not for consumers. If a consumer workspace is analyzing or enforcing policies, it has to explicitly include policies that allow all its legitimate consuming flows, either through automatic policy discovery or by explicitly manually crafting policies (no policy requests from external provider workspaces are generated to it).

#### Viewing, Accepting, and Rejecting Policy Requests

When creating cross scope policies using the method described in [\(Advanced\) Create Cross-Scope Policies, on page 505](#), a policy is required in the primary workspace of the provider's scope in addition to the policy in the consumer's scope. When a cross-scope policy is created in the primary workspace of the consumer's scope, a policy request is automatically created in the primary workspace of the provider's scope.

Use the information in this topic to accept the request (to create the required policy in the provider scope) or reject the request (in which case the cross-scope policy will not take effect.)

#### To view, accept, or reject policy requests:

| To                       | Do This                                                                                                                                                                                                                                  |
|--------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| View all policy requests | <ol style="list-style-type: none"> <li>1. Choose <b>Defend &gt; Segmentation</b>.</li> <li>2. Click <b>Policy Requests</b> at the top of the page.</li> <li>3. Click a consumer scope to see policy requests from that scope.</li> </ol> |

| To                                                                                           | Do This                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|----------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| View policy requests for a particular scope                                                  | <p>To view pending policy requests for a provider scope:</p> <ol style="list-style-type: none"> <li>1. Choose <b>Defend &gt; Segmentation</b>.</li> <li>2. Click the primary workspace of the applicable scope.</li> <li>3. Click <b>Manage Policies</b>.</li> <li>4. Click <b>Provided Services</b>.</li> </ol> <p>If the tab does not display a number, there are no policy requests pending for this workspace.</p> <ol style="list-style-type: none"> <li>5. Click <b>Policy Requests</b>.</li> <li>6. Click a consumer scope to see policy requests from that scope.</li> </ol> <p>Or</p> <p>To view a policy request from the consumer scope:</p> <p>In the Policies tab of the primary workspace of the consumer scope, click the value in the <b>Protocols and Ports</b> column, then look at the panel that opens on the right side of the page. In the <b>Protocols and Ports</b> section, click a yellow dot to see pending policy requests.</p> |
| Manually accept a request and automatically create the required policy in the Provider scope | From either of the locations above, click <b>Accept</b> next to the policy request.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| Manually reject a request                                                                    | From either of the locations above, click <b>Reject</b> next to the policy request.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |

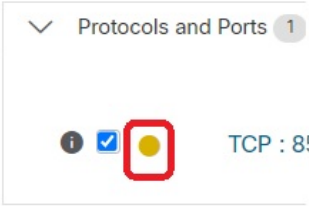

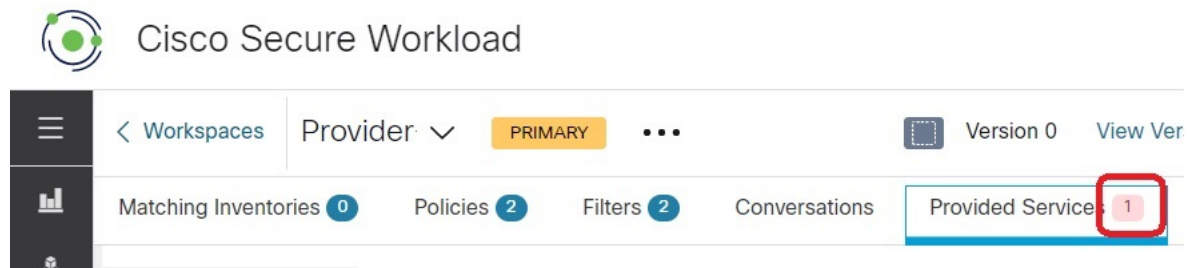
| To                                                                    | Do This                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
|-----------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| View policy request status from the consumer workspace                | <p>On the Policies page of the primary consumer workspace, click the policy, then click the port/protocol value. Status is shown in the panel that opens on the right.</p> <p>A pending request is shown with a yellow dot:</p>  <p>When the request is accepted, the dot changes to a green check mark:</p>  <p>mark:</p> <p>Click the indicator for details.</p> |
| View policy request status from the provider's workspace              | View request status in the <b>Provided Services</b> tab described above.                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| Allow policy discovery to create the required policy for the provider | Automatically discover policies in the provider scope's primary workspace, using a time range that ensures that the corresponding flows are seen, then publish the policy.                                                                                                                                                                                                                                                                                                                                                             |
| See also options for automating handling of policy requests           | <a href="#">Automate Handling of Cross-Scope Policy Requests, on page 511</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                          |

Figure 276: Pending policy requests in the provider's workspace



### Accepting Policy Requests: Details

Accepting a policy request on a service is equivalent to creating a policy from the requested filter as the consumer to the service as the provider. Additionally, upon accepting a policy request, the original policy

from the consumer application’s workspace (in the example, FrontEnd App and Serving Layer) will be marked as accepted (see figures below)

Figure 277: Accepting/Rejecting policy requests

The screenshot shows the 'Provided Services' section for the 'Tetration' provider. It displays a table of policy requests with columns for 'Consumer Application's Scope', 'Status', and 'Time'. The 'Tetration : FrontEnd' scope shows 1 pending, 0 accepted, and 0 rejected requests. The 'Tetration : Serving Layer' scope shows 0 pending, 1 accepted, and 1 rejected request. Below the table, there are details for two requests: one for TCP:90 (Accepted) and one for TCP:92 (Rejected), both originating from 'Tetration : Serving Layer' at 2:27 PM.

Figure 278: Policy status shown as Accepted

The screenshot shows the 'Serving Layer' workspace for the 'Tetration' provider. It displays a table of policies with columns for 'Priority', 'Action', 'Consumer', 'Provider', and 'Services'. A tooltip is visible over the first row, indicating that a policy request was accepted. The tooltip text reads: 'Policy request accepted', 'Request sent at: 2:27 PM', 'to Application: Tetration Workspace', 'with Scope: Tetration', 'Accepted at: 2:35 PM', 'By: You'.

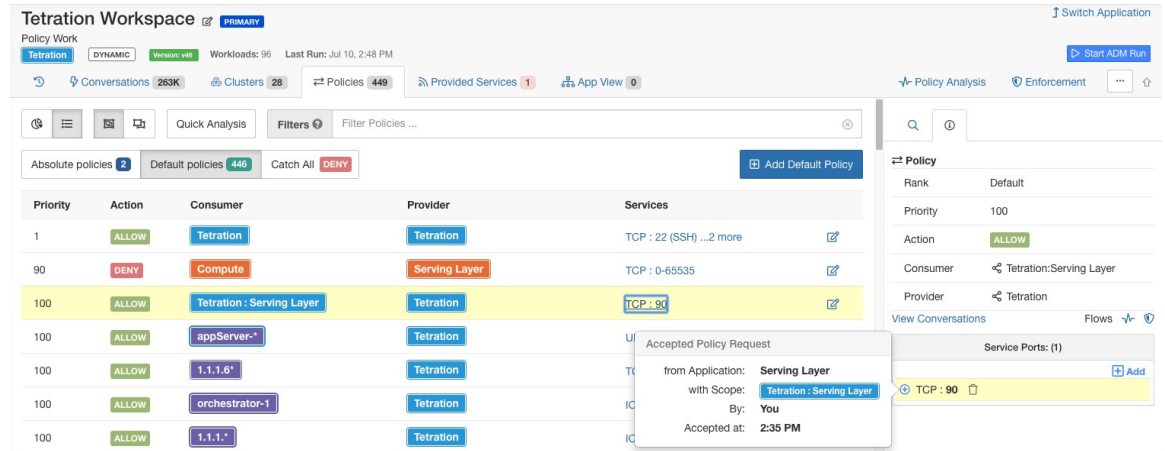
The new policy created on the provider application’s workspace (in this example, the workspace is named Tetration) is marked with a **plus** icon indicating that this policy was created due to an external policy request.



**Note** If the original policy on the consumer side is deleted after the policy request is accepted, the policy on provider side will not be deleted. However, the tooltip next to the policy shows the original policy as deleted with the timestamp of the event:



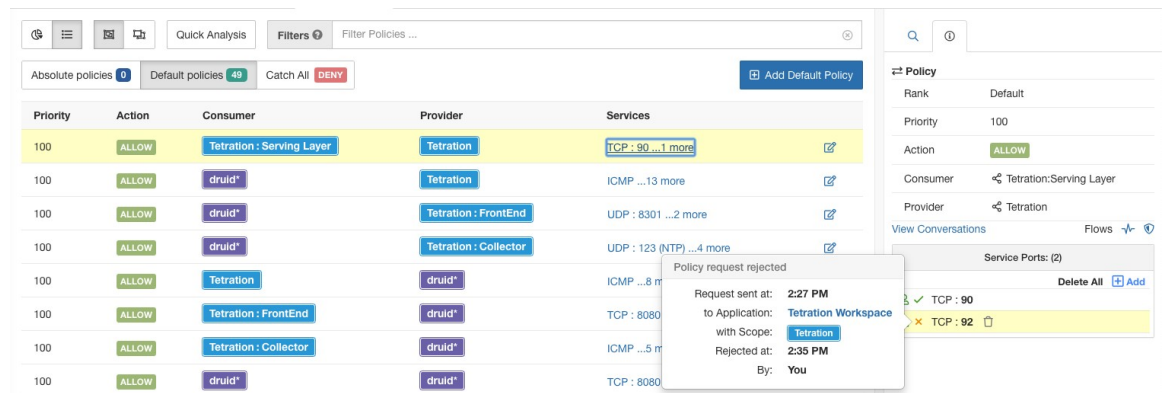
Figure 279: Provider side policy, created by accepting a policy request



### Rejecting Policy Requests: Details

Rejecting a policy request does not create or update any policies. The original policy from the consumer application’s workspace (in the example, Serving Layer App) will be marked as rejected, but the policy remains in effect, i.e., outbound traffic still will be allowed. The tooltip next to the reject policy has information about the provider application, the user that rejected the policy request as well as the time of the rejection.

Figure 280: Policy status shown as Rejected



### Automate Handling of Cross-Scope Policy Requests

Policy requests are generated when you create cross-scope policies using the method described in [\(Advanced\) Create Cross-Scope Policies](#), on page 505.

There are several options to reduce the number of policy requests that are generated when creating cross-scope policies:

Table 26: Options for Automatically Handling Policy Requests

| To                                                                                                                                              | Do This                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
|-------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Specify handling of policy requests between specific consumer-provider pairs                                                                    | See <a href="#">Auto-pilot Rules, on page 512</a> .<br>You must have the required privileges.                                                                                                                                                                                                                                                                                                                                                                           |
| Automatically create all required policies for providers for all cross-scope policies created during policy discovery in a particular workspace | When you start an automatic policy discovery run, enable the <b>Auto accept outgoing policy connectors</b> option in the Advanced Configurations section.<br>This option is available to root scope owners and site admins only.<br>For details, see:<br><a href="#">Advanced Configurations for Automatic Policy Discovery, on page 451</a> and <a href="#">Auto Accept Policy Connectors, on page 514</a>                                                             |
| Specify default handling for all policy requests from all workspaces                                                                            | On the Default Policy Discovery Config page, enable the <b>Auto accept outgoing policy connectors</b> option in the Advanced Configurations section.<br>This option is available to root scope owners and site admins only.<br>For details, see:<br><a href="#">Default Policy Discovery Config, on page 461</a> and <a href="#">Advanced Configurations for Automatic Policy Discovery, on page 451</a> and <a href="#">Auto Accept Policy Connectors, on page 514</a> |

## Auto-pilot Rules

This feature is applicable only if you create cross-scope policies using the method described in [\(Advanced\) Create Cross-Scope Policies, on page 505](#).

Infrastructure applications that provide services to many other applications in a datacenter may receive a large number of policy requests from other applications.

You can reduce the volume of policy requests by creating auto-pilot rules to automatically accept or reject future matching policy requests.




---

**Note** Auto-pilot rules do not apply to existing policy requests. They affect only future policy requests.

---

### Automatically accept or reject policy requests using Auto-Pilot Rules

Configure auto-pilot rules to automatically accept or reject policy requests between a specified consumer-provider pair, on specified ports. Auto-pilot rules can be broad (scope-to-scope), or apply only to a subset of workloads within each scope (as configured by inventory filters. You can use an inventory filter for the consumer, for the provider, or for each.)

1. If you want your auto-pilot rule to apply to a subset of workloads within a scope rather than to the entire scope:

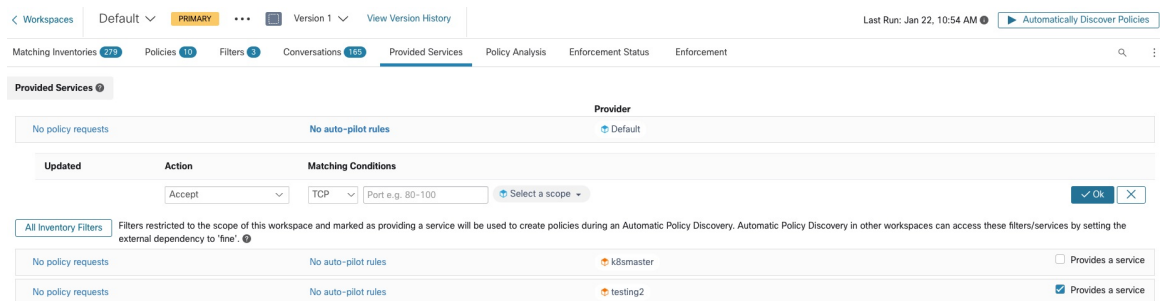
Create an inventory filter in the relevant scope(s) to group the workloads. Be sure the **Restrict Query to Ownership Scope** option is selected in each inventory filter, to ensure that the filter only includes workloads that are members of the scope.

2. Choose **Defend > Segmentation**.
3. Click the primary workspace of the consumer scope for which you want to automatically accept or reject policy requests related to a specific provider.
4. Click **Manage Policies**.
5. Click **Provided Services**.
6. If you are creating this rule for an inventory filter, perform the following steps for the desired inventory filter (inventory filters are identified by an orange icon.)  
Otherwise perform these steps for the scope (scopes are identified by a blue icon.)  
Make sure you are clicking in the correct place.
7. Click **No Auto-Pilot Rules** or **auto-pilot rules**, whichever is displayed.
8. Click **New Auto-Pilot Rule**.
9. Configure the auto-pilot rule. Select the scope or inventory filter that represents the provider.
10. Click **OK**.

### Example Auto-Pilot Rule

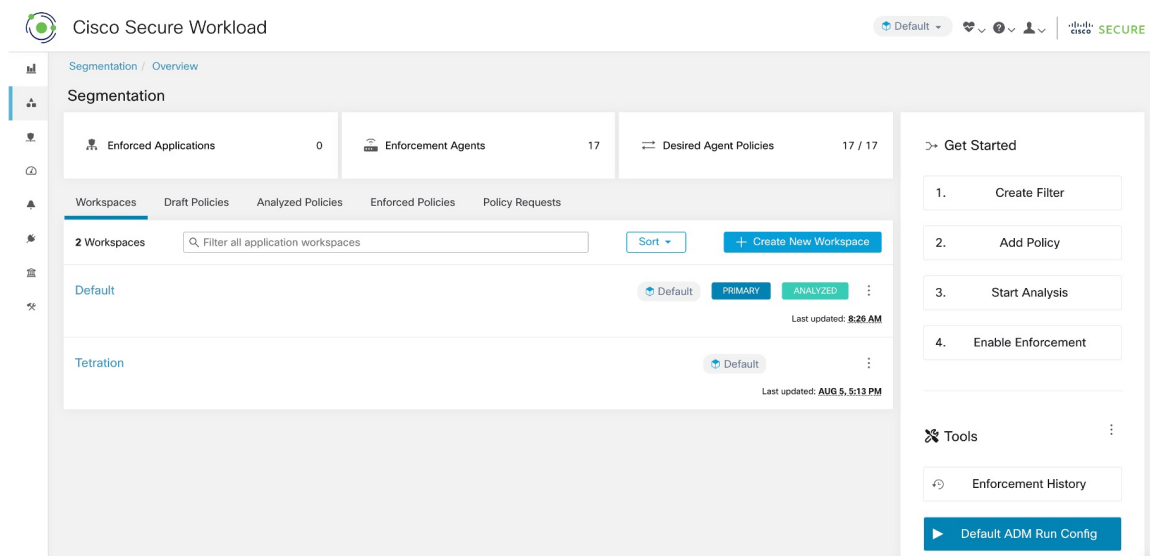
In the example below, we create a new auto-pilot rule to reject TCP policy requests in port range 1-200 from any consumer contained in Tetration:Adhoc to the provider service Tetration

**Figure 281: Creating/Updating Auto-pilot rules**



Then we create a new policy in the workspace for the *FrontEnd App* on TCP port 23. Since the policy is a match for the auto-pilot rule, it will be automatically rejected. The status and reason for policy rejection is indicated on the tooltip next to the rejected policy.

Figure 282: Policy automatically getting rejected by Auto-pilot rule



### View a count of policies recently created by auto-pilot rules

To view the number of policies created in a workspace by auto-pilot rules since live policy analysis was last initiated (or re-initiated) for the workspace:

Navigate to the Provided Services page for the relevant primary workspace and look for the count of “Auto Created” policies.

### Auto Accept Policy Connectors

You can set this option as the default policy discovery configuration, or set it in the automatic policy discovery advanced options for each workspace.

The **Auto accept outgoing policy connectors** option on the automatic policy discovery configuration page allows you to automatically accept any policy requests created as part of automatic policy discovery.

If this option is enabled in Default automatic policy discovery config, policy requests created manually or by importing a workspace will be automatically accepted as well.

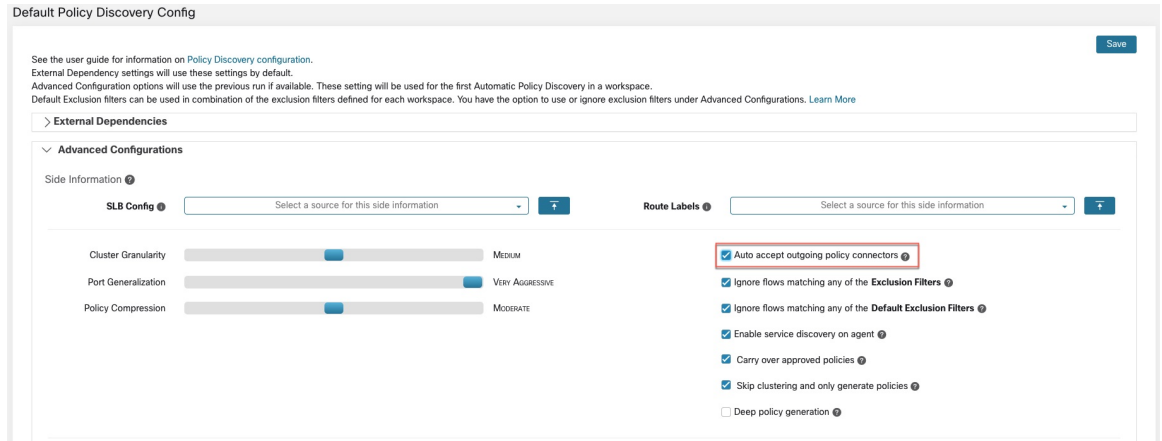



---

**Note** This option is only available for root scope owners or Site Admins.

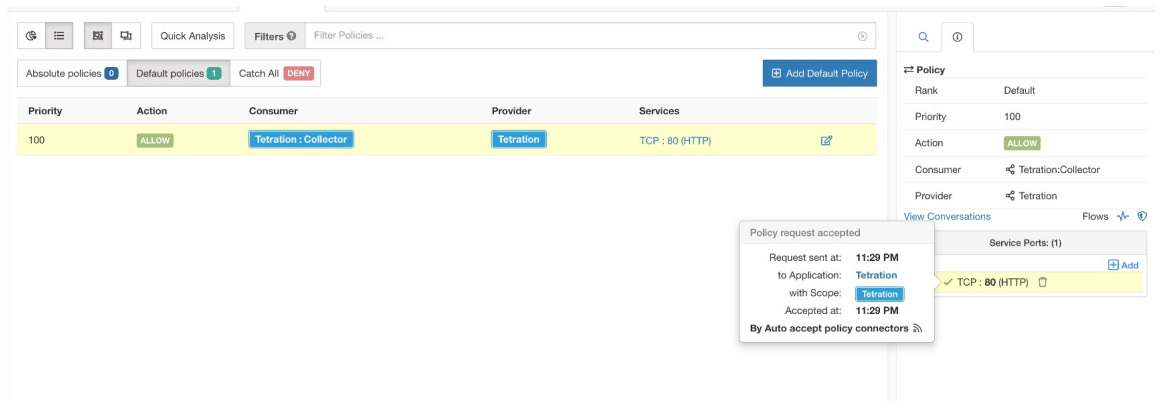
---

**Figure 283: The Auto accept outgoing policy connectors option**



After this option is set, any policy request created in any workspace the root scope or in the applicable workspace will be automatically accepted.

**Figure 284: Policy is automatically accepted by Auto accept policy connectors**



**Resolved Policy Requests**

If all conditions for creating a policy request are met, but there is already an existing matching policy on the provider application’s workspace, the policy created on the consumer application’s workspace will be marked as resolved indicating that the provider application’s workspace is already allowing the traffic through the requested port.

Figure 285: Policy status shown as Resolved

The screenshot displays the 'Provided Services' page in the Cisco Secure Workload console. The main table lists policies with columns for Priority, Action, Consumer, Provider, and Services. A tooltip is visible over the 'Resolved' status, showing the request details: 'Request sent at: 2:19 PM', 'to Application: Tetration Workspace', 'with Scope: Tetration', and 'Resolved at: 2:19 PM'. The right sidebar shows a detailed view of a policy, including its Rank, Priority, Action, Consumer, and Provider.

| Priority | Action | Consumer             | Provider             | Services                  |
|----------|--------|----------------------|----------------------|---------------------------|
| 100      | ALLOW  | Tetration : FrontEnd | Tetration            | TCP : 22 (SSH) ... 1 more |
| 100      | ALLOW  | appServer-*          | Tetration            | ICMP ... 35 more          |
| 100      | ALLOW  | mongodb*             | Tetration            | UDP : 53 (DNS) ... 7 more |
| 100      | ALLOW  | redis-*              | Tetration            | ICMP ... 6                |
| 100      | ALLOW  | elasticsearch-*      | Tetration            | UDP : 53 (DNS) ... 7 more |
| 100      | ALLOW  | Tetration            | Tetration : FrontEnd | TCP : 22 (SSH) ... 1 more |
| 100      | ALLOW  | 4.4.2.5              | Tetration : FrontEnd | TCP : 5000                |
| 100      | ALLOW  | 1.1.1.6*             | Tetration : FrontEnd | TCP : 8000 ... 11 more    |
| 100      | ALLOW  | 1.1.1.* [2]          | Tetration : FrontEnd | UDP : 514                 |

## Provided Services

This page is used only for creating policies in which the consumer and provider are in different scopes, and only if you are using the method described in [\(Advanced\) Create Cross-Scope Policies, on page 505](#).

For more information about options on this page, see:

- [Policy Requests, on page 506](#)
- [Auto-pilot Rules, on page 512](#)
- [Create an Inventory Filter, on page 378](#) and [External Dependencies, on page 448](#) (for information about the **Provides a service** option)

To access this page, navigate to a primary workspace, then click **Manage Policies**, then click **Provided Services**.

## Troubleshoot Cross-Scope Policies

If cross-scope policies were created using the method described in [\(Advanced\) Create Cross-Scope Policies, on page 505](#), the primary workspaces for the consumer and provider workloads must each have a policy that allows the traffic. Ensure that the required policies exist in both workspaces.

No notification is given if one of the policies is deleted or modified.

If the policy pair was generated during policy discovery, see information about approving policies to protect them from subsequent discovery runs. See [Approve Policies, on page 463](#).

Verify that other requirements are still being met, as listed in [\(Advanced\) Create Cross-Scope Policies, on page 505](#).

Both consumer and provider workspaces that have the required policies must be enforced.

### Useful tools for cross-scope policies

- Use the **External?** filter option to find policies in which the provider is in a different scope from the scope in which you discovered policies.
- The policy visual view has an option to display external policies. See [Policy Visual Representation, on page 523](#).

### If you are using Default Policy Discovery Config

Make sure you have clicked **Save** on the **Default Policy Discovery Config** page after making changes to make default external dependency configurations available to individual workspaces.

## Effective Consumer or Effective Provider

The consumer and provider specified in a policy determine:

- The set of workloads with Secure Workload agents that receive the policy.
- The set of IP addresses that are affected by the installed firewall rules.

By default, these are the same.

However, you may need to specify a group of IP addresses in the firewall rules that is different from the IP addresses of the workloads that receive the policy. (See an example below.)

To address this need, you can configure effective consumer and/or effective provider.

### Default behavior for consumer and provider

By default, when a Secure Workload agent receives a policy, the firewall rules are specific to that workload. This is best illustrated with the following example:

Consider an ALLOW policy with provider filter specifying 1.1.1.0/24 subnet. When this policy is programmed on a workload with IP address 1.1.1.2, the firewall rules look like the following:

- For incoming traffic firewall rules allow traffic destined to 1.1.1.2 specifically and not to the whole subnet 1.1.1.0/24.
- For outgoing traffic firewall rules allow traffic sourced from 1.1.1.2 specifically and not from the whole subnet 1.1.1.0/24 (to prevent spoofing).

As a corollary, any agent workloads belonging to the workspace that do not have IP address within 1.1.1.0/24 subnet will not receive the above firewall rules.

### Example: Effective Consumer or Effective Provider

In this example, suppose you are configuring policies for a fleet of workloads behind a virtual IP (VIP), similar to keepalive or windows failover clustering solutions. You will use effective consumer and /or effective provider to ensure that traffic is not disrupted during a failover event.

Consider a fleet of workloads with IP addresses (172.21.95.5 and 172.21.95.7) that provide a service sitting behind a VIP - 6.6.6.6. This VIP is a floating VIP and only one workload owns the VIP at any point in time. The goal is to program firewall rules on all the workloads in the fleet to allow traffic to 6.6.6.6.

In this setup, we have a scope and a corresponding workspace that contain a cluster of workloads that represents the fleet (172.21.95.5 and 172.21.95.7) as well as the VIP (6.6.6.6).

**Figure 286: Scopes including VIP and cluster of workloads**

| Name       | Query                                                               | Ability | Total Children |
|------------|---------------------------------------------------------------------|---------|----------------|
| WinClients | Address = 172.21.95.1 or Address = 172.21.95.3                      | Owner   | 0              |
| WinServers | Address = 172.21.95.5 or Address = 172.21.95.7 or Address = 6.6.6.6 | Owner   | 0              |

The VIP is exposed in this workspace as a provided service as shown below:

**Figure 287: VIP exposed as a provided service**

The screenshot shows the 'Provided Services' section in a management console. It features a table with columns for 'No policy requests', 'No auto-pilot rules', and 'Provider'. A service named 'Tetration' is listed with a 'Test' button and a checked 'Provides a service' checkbox. A detailed view of the 'Tetration' service is shown on the right, including filter actions, query, scope, and restricted status.

If we were to add a policy from the clients of this service to the service VIP, then (by default) firewall rules allowing traffic to the VIP will only be programmed on the workload that owns the VIP. However, in case of a failover event, it may take some time for the new workload that subsequently owns the service VIP to get the right firewall rules and traffic may be disrupted for a brief while.

**Figure 288: Policy allowing traffic from clients to service VIP**

The screenshot displays the 'Policies' section of the management console. A table lists various policies with columns for Priority, Action, Consumer, Provider, and Protocols And Ports. Most policies have an 'ALLOW' action. A detailed view of a policy action is shown on the right, highlighting the 'ALLOW' action and the associated consumer and provider information.

To address this issue, we configure the Effective Provider (using the procedure below.) Specifically, we set Effective Provider to include the group of workloads where firewall rules allowing traffic to the service VIP need to be programmed – it does not matter if any of these workloads own the VIP or not.

When Effective Provider is set, we can see on the workloads that firewall rules allowing traffic to 6.6.6.6 are programmed even when a workload does not own the VIP. When all workloads backing the service are programmed with these rules, traffic will not be disrupted during a failover event because the new primary workload (that owns the VIP) will have the necessary firewall rules programmed.



Figure 289: Firewall rules on the host allowing traffic to service VIP

```

$
$ hostname -I | awk '{print $1}' IP Address of
172.21.95.7 the server
$ part of cluster
$
$ sudo iptables -n --list TA_INPUT ← Ingress rules
Chain TA_INPUT (1 references)
target prot opt source destination
ACCEPT tcp -- 0.0.0.0/0 0.0.0.0/0 match-set ta_6c6b4133313438ff5429ca8c14b6 src match-set ta_ac2618d307e4e7dbb76b96c0df3f dst mul
tiport dports 1443 ctstate NEW,ESTABLISHED /* PolicyId=DEFAULT:100:ALLOW:5ed53fe8497d4f26444d50b3:5ed5435b497d4f26414d50b1:6 */
RETURN all -- 0.0.0.0/0 0.0.0.0/0
$
$ sudo iptables -n --list TA_OUTPUT ← Egress rules
Chain TA_OUTPUT (1 references)
target prot opt source destination
ACCEPT tcp -- 0.0.0.0/0 0.0.0.0/0 match-set ta_ac2618d307e4e7dbb76b96c0df3f src match-set ta_6c6b4133313438ff5429ca8c14b6 dst mul
tiport sports 1443 ctstate ESTABLISHED /* PolicyId=DEFAULT:100:ALLOW:5ed53fe8497d4f26444d50b3:5ed5435b497d4f26414d50b1:6 */
RETURN all -- 0.0.0.0/0 0.0.0.0/0
$
$ sudo ipset list ta_ac2618d307e4e7dbb76b96c0df3f
Name: ta_ac2618d307e4e7dbb76b96c0df3f
Type: hash:net
Revision: 3
Header: family inet hashsize 1024 maxelem 65536
Size in memory: 16816
References: 2
Members:
6.6.6.6 ← VIP
$ sudo ipset list ta_6c6b4133313438ff5429ca8c14b6
Name: ta_6c6b4133313438ff5429ca8c14b6
Type: hash:net
Revision: 3
Header: family inet hashsize 1024 maxelem 65536
Size in memory: 16848
References: 2
Members:
172.21.95.1
172.21.95.3 ← Client IPs
$

```

## How to Configure Effective Consumer or Effective Provider

1. Click the policy to edit.
2. Click the Edit button at the top right side of the policy to go to advanced policy options.
3. Click **Effective Consumer** or **Effective Provider**.
4. Specify the desired addresses.
5. You may need to specify addresses for both effective consumer and effective provider.

# About Deleting Policies



## Important

Before you delete a policy, check to be sure it is not one of a pair of policies required when consumer and provider are in different scopes.

To determine this: Click the link for the policy in the Protocols and Ports column. In the panel that opens on the right side of the page, look at the Protocols and Ports section. Policies created by accepting a cross-scope policy request are indicated by a plus sign beside the port and protocol:



Click the plus sign to see the creator of the cross-scope policy and a link to the corresponding consumer policy.



## Note

Policies suggested by automatic policy discovery that have not been approved may not be present after a subsequent policy discovery run, if the traffic flows that produced them are not seen during the subsequent run. To preserve suggested policies, see [Approve Policies, on page 463](#).

## Review and Analyze Policies

It is essential that you ensure that your policies will have the intended effects (and not have any unintended effects) before you enforce them.

### Review Automatically Discovered Policies

Review policy discovery results on the Policies page of the workspace in which you discovered policies.

#### Start your review here


We recommend that you start by checking to see if policies address each of the following areas, in this suggested order:

- Critical, common ports
- Internet-facing traffic
- Traffic between different applications (These flows may involve workloads in different scopes)
- Traffic within the same application (These flows are likely to involve workloads in the same scope)

### Helpful tools for reviewing policies

- To make this effort more manageable, filter and sort the policies so you can review related policies as a group.
  - Click table headings to sort the columns, for example by consumer, provider, or port/protocol.
  - Use the filter at the top of the policies list to view specific subsets.
    - To see a list of properties that you can filter on, click the (i) button in the Filter Policies box.

- Look at the graphical representation of the generated policies:

Click the Policy Visual View button ()

For more information, see [Policy Visual Representation, on page 523](#).

- To search or filter the rows based on ports, click the **Ungrouped** button.
- By default, the policies are grouped by consumer/provider/action. To return to this view, click the **Grouped** button.
- Use the **External?** filter option to find policies in which the provider is in a different scope from the scope in which you discovered policies.
  - Create policies for this traffic using one of the methods described in [When Consumer and Provider Are in Different Scopes: Policy Options, on page 504](#).
- Look at the confidence level of the generated policies. See [Address Low-Confidence Policies, on page 522](#).
- Look at the Workload Profile for detailed information about a workload. Click the IP address, then click **View Workload Profile** in the pane on the right.
- To view the traffic flows that were used to produce a particular policy, click the value in the **Protocols and Ports** column for that policy, then click **View Conversations** in the side panel that opens.
  - See [Conversations, on page 559](#) for more information.
  - If needed, you can drill down further by clicking **Flow Search** to view the flows for a conversation.

### Other things to do and check

- Identify unknown IP addresses (such as failover or other floating IPs) and tag them with labels so you know what they are.
  - You may find helpful details on the Inventory Profile page. Click the IP address, then click **View Inventory Profile** in the pane on the right.
- Look for anything that is obviously not desirable or does not make sense.
- Group workloads using inventory filters so a single policy can address multiple workloads. See [Create an Inventory Filter, on page 378](#).
- Investigate and contact other network administrators as needed to understand the need for the policies you see.
- See the topics under [Address Policy Complexities, on page 498](#), which can involve manual and approved policies as well as automatically discovered policies.

- In general, it is recommended that the maximum number of policies in a scope is not larger than about 500. If you have many more than this, see if you can consolidate similar policies or consider splitting the scope.
- As you review, approve any policies that you know are correct as-is to preserve them in future discovery runs.

## Address Low-Confidence Policies

After automatic policy discovery, confidence ratings indicate the accuracy and appropriateness of each discovered policy for each service (port and protocol) specified in the policy.

### To identify low-confidence discovered policies:

1. Navigate to the applicable scope and workspace and click **Manage Policies**.
2. Click the **Policies** tab.
3. Click the **Ungrouped Policy List View** button.
4. Click the **Confidence** column heading to sort the list of policies by confidence level.
5. Click the value in the **Protocols and Ports** column to open a panel on the right side of the window.
6. In the **Protocols and Ports** section, the color of each **C** indicates the confidence for each service (port and protocol) specified in the policy.  
To interpret the confidence level, hover over the **C**.
7. Look for low-confidence indicators for any services in the list.
8. If applicable, delete or edit unwanted policies, or add additional policies.

### To view the confidence levels for a particular policy:

1. In the Policies tab, click the value in the **Protocols and Ports** column for that policy.  
The Policy Side View panel opens at the right side of the window.
2. In the **Protocols and Ports** section, the color of each **C** indicates the confidence for each service (port and protocol) specified in the policy.  
To interpret the confidence level, hover over the **C**.

### Flow Direction and Policy Confidence

The accuracy of discovered policies depends on correct identification of the flow direction. If flow direction is incorrectly identified, the confidence rating of automatic policy discovery results may be reduced. For information about determination of flow direction for the conversation(s) analyzed for the creation of the policy, see [Client-Server Classification](#).

## Troubleshoot Automatic Policy Discovery Results

If automatic policy discovery results are not what you expect, check the following:

**Extend the selected time range to include more data**

Extend the time window to include more data and to capture events that happen infrequently. For example, if an application generates a complex quarterly report using data drawn from several provider applications, be sure to include a time range that includes that traffic.

**Avoid data gathered before certain changes**

If the scope definition has changed, or data gathered before a certain time has become invalid for some other reason, be sure your time range does NOT include data before that point.

**Exclude misleading traffic flows**

Exclusion filters may need to be configured or modified.

There are multiple places exclusion filters can be configured, and multiple places they can be enabled or disabled. Check each location:

- Check the exclusion filters configured for the workspace.
- Check the default exclusion filters configured at the bottom of the Default Policy Discovery Config page.
- Check which exclusion filters are enabled in the Advanced Configurations section in the workspace's settings for automatic policy discovery.
- Check which exclusion filters are enabled in Advanced Configurations section on the Default Policy Discovery Config page.
- If you are using Default Exclusion Filters, make sure you have clicked **Save** on the **Default Policy Discovery Config** page to make those configurations available to individual workspaces.

For details, see [Exclusion Filters, on page 445](#) and subtopics.

**Troubleshoot Policies in which Consumer and Provider Are in Different Scopes**


See [Troubleshoot Cross-Scope Policies, on page 516](#).

**Check Status of Approved Policies**

See [Troubleshoot Approved Policies, on page 464](#).

## Policy Visual Representation

Policy visual representation provides a graphical view of the policies.

To navigate to the policy visual representation page: On the Policies page, click the graph icon () to the right of the list icon.

**Policy View Elements**

The visual elements on the policy view are:

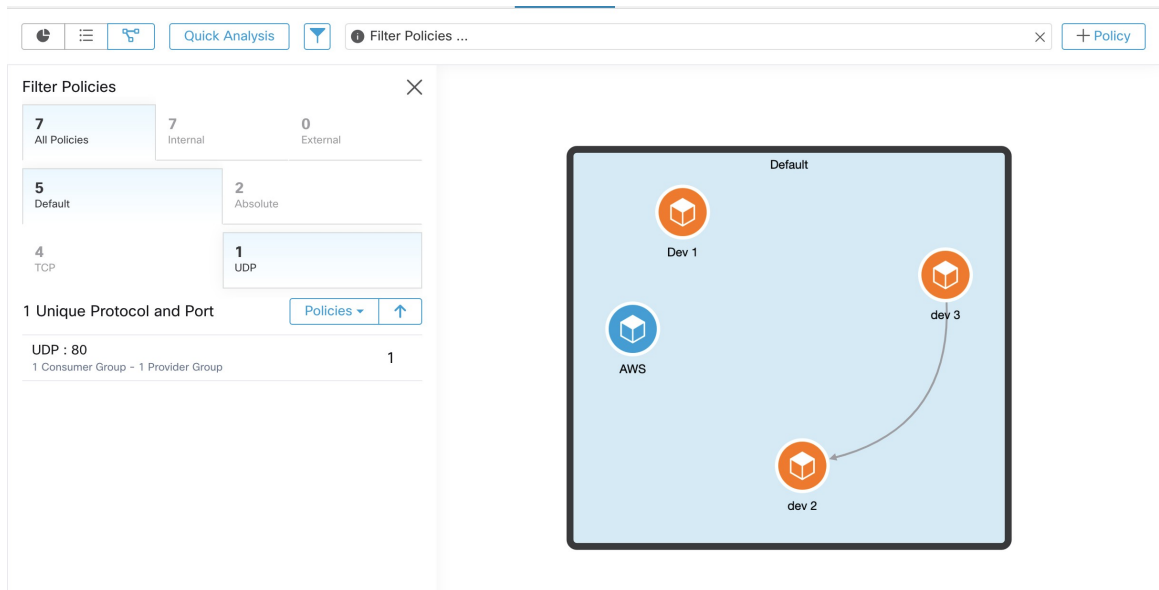
| This Element                   | Represents This                               |
|--------------------------------|-----------------------------------------------|
| A blue, orange, or purple icon | A node (the consumer or provider of a policy) |

| This Element              | Represents This       |
|---------------------------|-----------------------|
| Blue icon                 | A scope               |
| Orange icon               | An inventory filter   |
| Purple icon               | A cluster             |
| Line connecting two icons | One or more policies. |

**Policy View Options**

| To                                                                                                               | Do This                                                                                                       |
|------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------|
| View the list of workloads included in a consumer or provider node                                               | Double-click the node's icon.                                                                                 |
| View policy specifics such as services (ports), action (Allow/Deny) and protocol between a consumer and provider | Double-click the line connecting them. Details appear in the pane on the right.                               |
| View the policies entering and leaving a node                                                                    | Click the icon.                                                                                               |
| View only the policies between workloads within the scope                                                        | Click the <b>Internal</b> button.                                                                             |
| View only the policies in which the provider is in a different scope from the consumer                           | Click the <b>External</b> button.                                                                             |
| Use advanced filtering options                                                                                   | Click the (i) button to the left of the filter text input box to see the options, then enter filter criteria. |

**Figure 290: Filtering policies in graphical view**



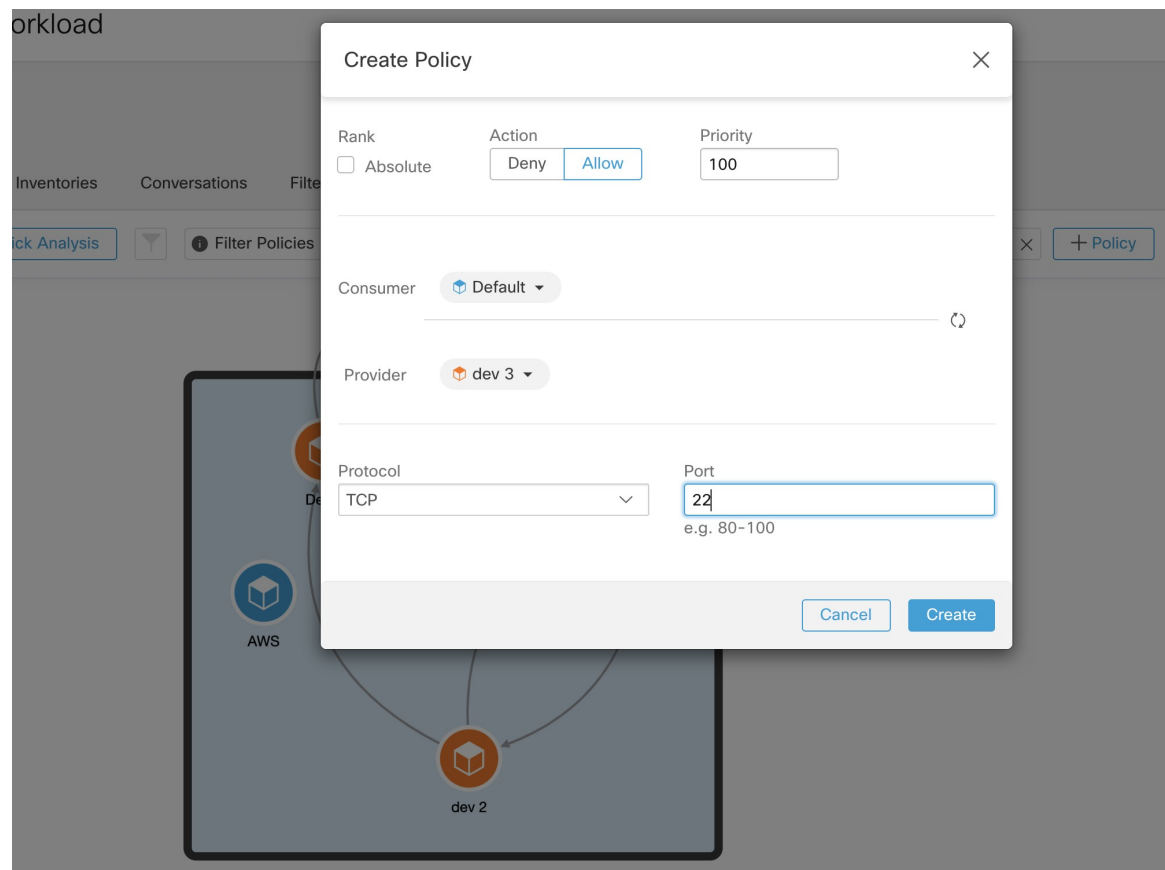
To download a high resolution image of the graphical view of the policies:

1. In the lower-right corner of the graph, click the ellipsis icon, and then click **Export Image**.
2. Select the required resolution and image type.
3. Click **Download**.

### Add a Policy (Policy View Page)

To create a policy, hover over the consumer until you see a “+” sign and then hold and drag the policy onto the provider. To create an Absolute policy, toggle the Absolute checkbox in the modal. Otherwise, the policy is created as a Default policy. Policies can also be managed by clicking a line and selecting a policy from the pop-up list. Policies will be displayed in the sidebar.

*Figure 291: Policy creation in graphical view*



## Quick Analysis

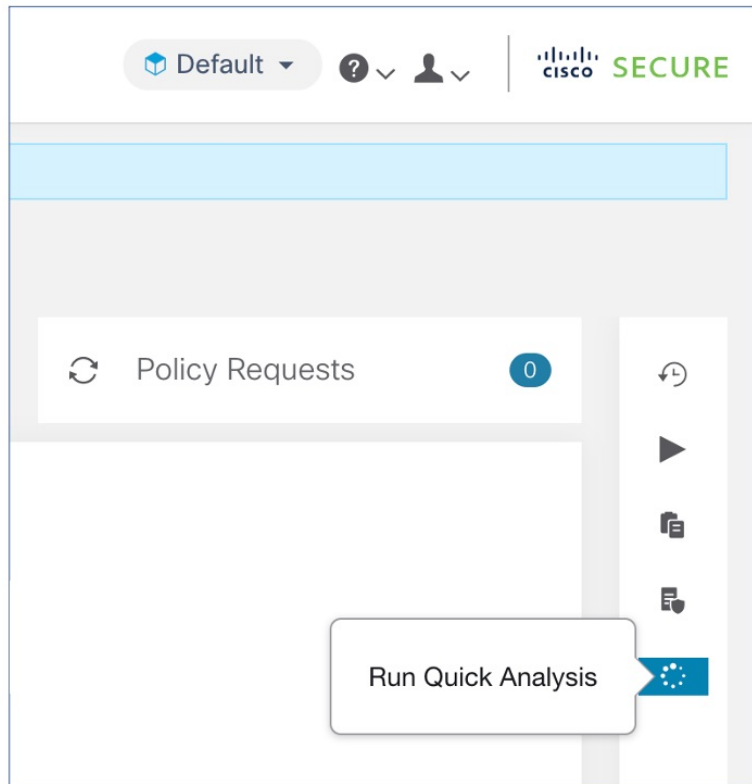
Quick analysis enables testing a hypothetical flow against all the policies in the current workspace and all other relevant policies from other workspaces. Quick analysis facilitates debugging and experimentation with different security policies, without the need to run live policy analysis for the workspace.

**Restriction**

- You can run Quick Analysis only on primary workspaces.
- Quick Analysis is not currently supported on flows from Kubernetes services.

Click the **Run Quick Analysis** tab on the right navigation pane to view the dialog.

*Figure 292: Quick Analysis Tab*



Enter the Consumer (client) IP, Provider (server) IP, port, and protocol for the hypothetical flow, then click **Find Matching Policies** button.

A policy decision will be shown indicating whether the hypothetical flow would be allowed or denied given the policy definitions in the latest version of the workspace and all other policies from relevant workspaces that are already pushed for live policy analysis.

At the bottom of the dialog, we show the matching outbound and inbound policies separately, and in their globally sorted order. It is only the first row on either side that has any effect. For a connection to successfully get established, we need both the top outbound rule on consumer and the top inbound rule on the provider side to be ALLOW rules.

Showing all other matching policies in their order, provides a valuable debugging tool to help sort out issues in policy definitions when a certain policy seems to not be taking any effect. You can add, update, or delete policies from the workspace, and repeat the analysis immediately without the need to run live policy analysis on the workspace.



Figure 293: Quick Policy Analysis

Quick Hypothetical Flow Analysis ?

Match this Hypothetical Flow against

Replace this application's policies with

Version: v1

Consumer Address: 173.38.45.96

Provider Address: RCON9-DC-Internal

Protocol: TCP

Provider Port: 80

Policy Decision: ✔ ALLOW

| Consumer Outbound Policies                                                                            | Provider Inbound Policies                                                                             |
|-------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------|
| <p>OTHER: unknown → bpimdmgr-idev3-0*</p> <p>ALLOW TCP : 22 Default</p> <p>Tetration [v1] Default</p> | <p>OTHER: unknown → bpimdmgr-idev3-0*</p> <p>ALLOW TCP : 22 Default</p> <p>Tetration [v1] Default</p> |
| <p>OTHER: unknown → bpimdmgr-idev3-0*</p> <p>ALLOW TCP : 22 Default</p> <p>Tetration [v1] Default</p> | <p>OTHER: unknown → bpimdmgr-idev3-0*</p> <p>ALLOW TCP : 22 Default</p> <p>Tetration [v1] Default</p> |

## Live Policy Analysis

After you have reviewed and approved the set of network security policies generated by automatic policy discovery, and before you enforce the policies, you should use live policy analysis to observe how the policies would affect actual traffic on your network.

Some questions that live policy analysis can help you answer:

- What would be the impact on this scope's application(s) if the policies in this workspace are enforced now?
- Could we have prevented a previously known security attack/risk by enforcing the new set of policies?  
See [Run Policy Experiments to Test Current Policies Against Past Traffic](#), on page 534.
- Are our policies working the way we expect them to?

You should run policy analysis on any workspace that has policies. Because workloads in any particular scope can be affected by policies in other scopes, you should not run policy analysis only for a single scope before enforcing policy for that scope. Consider analyzing policies for all scopes that may affect traffic in a particular scope.

For example:

- Policies defined in scopes above this scope in the tree may apply to workloads in this scope.

- If workloads in this scope communicate with workloads in a different scope, policies in that scope may affect these communications. When policy analysis is started in that scope (or latest policies are analyzed after a policy change there), this can affect this scope's policy analysis results.

You should perform policy analysis any time you revise policies, to ensure that changes don't break applications.

Running live policy analysis on a workspace is sometimes referred to as "publishing" a workspace.

## Start Live Policy Analysis

Once you have reviewed the policies generated in a workspace by automatic policy discovery, and you believe that they are as you want them, you can start policy analysis.

### Before you begin

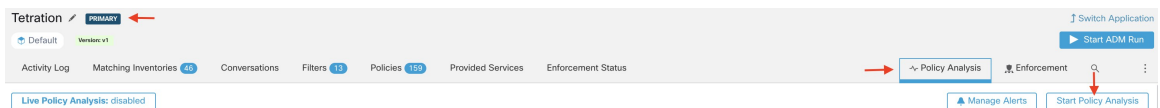


**Important** Live analysis includes the effects of policies in other workspaces that are also running live analysis. If you have enabled enforcement on any workspace, but analysis is not running on that workspace or the enforced version of the policies is not the same as the analyzed version of the policies, your live analysis results for this workspace may not be accurate.

### Procedure

- Step 1** Toggle the workspace to **Primary** by clicking **...** to the right of "Secondary" next to the workspace name in the header.
- Step 2** Navigate to the **Policy Analysis** tab.
- Step 3** Click **Start Policy Analysis** on the right.

**Figure 294: Enable Policy Analysis**



### What to do next

- Because policies in other scopes can apply to workloads in this scope, consider simultaneously analyzing policies in other scopes that could affect analysis results in this scope. See [Example: Impact of Policies Analyzed in Other Scopes, on page 530](#).
- If you want to be notified when escaped flows are detected, click **Manage Alerts**.
- Use the tools on the page to filter the data. To see the available filter criteria, click the (i) button in the filter box.
- If you add or change policies after initiating policy analysis, you must re-start analysis in order to include the changes in the analysis. See [After Changing Policies, Analyze Latest Policies, on page 535](#).

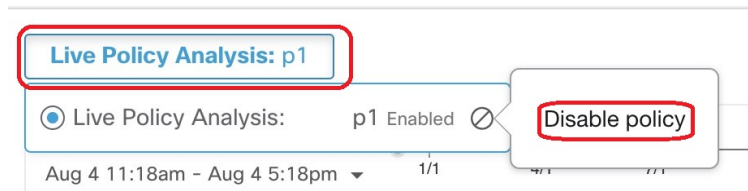
## Stop Live Policy Analysis

In general, you should let policy analysis continue to run, even after you enforce policies, since the policies in this workspace may affect policy analysis results in other workspaces you are analyzing.

To stop live policy analysis:

Click the **Live Policy Analysis: P<number>** button, then click **Disable policy**:

**Figure 295: Disable Live Policy Analysis**

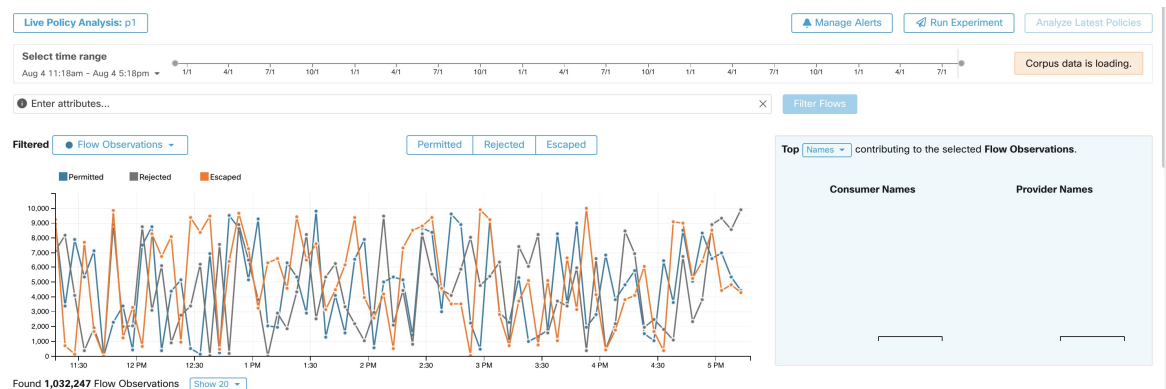


## Policy Analysis Results: Understand the Basics

During policy analysis, all flows traveling into, out of, and within the scope associated with the workspace are assigned one of the following results:

- **Permitted:** Flow was allowed by the network, and also by the analyzed policies.
- **Escaped:** Flow was allowed by the network, but should have been dropped according to the analyzed policies.
- **Rejected:** Flow was dropped by the network, and also by the analyzed policies.

**Figure 296: Policy Analysis Page**



Some things to look at to get oriented:

- You may filter the flow information presented in this page via a faceted filter bar. Clicking the **Filter Flows** button updates all the charts accordingly.
- Hovering on the chart shows the percentage of the aggregate observed flows at that timestamp.
- Clicking on a timestamp reveals a list of all filtered flows in a table below for further analysis.
- You can limit the interactions to one of the three result types by selecting or deselecting the types at the top of the time series charts.

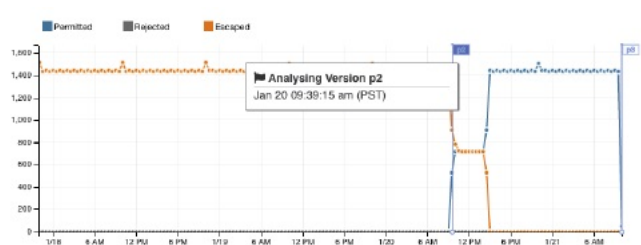
## Example: Impact of Policies Analyzed in Other Scopes

- The Top N chart on the right shows the top Hostnames, Addresses, Ports, and so on contributing to the data shown in the time series chart on the left.

You can limit the time series chart to escaped flows and select “Ports” in the Top N chart to see the top ports contributing to escaped flows.

## Example: Impact of Policies Analyzed in Other Scopes

In the following example, there are permitted flows up to about 12pm. At that time, policy analysis was started in a workspace associated with a different scope, affecting traffic with workloads in this scope and causing flows to be marked as escaped. (You know that this change did not result from policy changes newly analyzed in this workspace, because this would have created a label flag.)



### Analysis without Policies

The flows into, out of, and within the scope associated with the workspace may be affected by policies in other workspaces that are being analyzed. If live policy analysis is not enabled on this workspace, the flows will be marked with those of the other workspaces in the system that do have live policy analysis enabled.



**Note** If no workspaces are running live policy analysis, the timeseries chart will be empty.

## Policy Analysis Details

### Flow Disposition

In policy live analysis, to decide on whether a flow is **Permitted**, **Escaped**, or **Rejected**, we have to first determine the **Disposition** of the flow from the network perspective. Each flow will receive an **ALLOWED**, **DROPPED** or **PENDING** disposition, based on the signals and observations given by Secure Workload agents. There are a number of scenarios based on the agent configurations along the path of the flow and the flow types.

First, regardless of flow types, if any agent along the path of a flow reports that the flow is **DROPPED**, the flow will receive a **DROPPED** disposition.

When there is no **DROP** reported by any agents along the path of the flow, we consider the case of bidirectional flows and unidirectional flows separately. When bidirectional flows are observed, we look at flows in pairs (forward and reverse) based on their source, destination ports and protocol, and timings. The same cannot be done for unidirectional flows.

For bidirectional flows, if there are agents installed and data plane enabled on both ends, a forward flow will receive an **ALLOWED** disposition if both the source and the destination agent report that the flow is observed. Otherwise, the forward flow will get a **PENDING** disposition. If an agent is installed on either the source or the destination workload, but not both, then the forward flow will received an **ALLOWED** disposition if and

only if the agent observes subsequent reverse flow within a **60**second window. Otherwise a PENDING status will be assigned to the forward flow. The disposition of the reverse part of the bidirectional flow follows the same logic except that now the source and the destination are reversed. For example, if only one side has an agent, whether a reverse flow disposition is PENDING or ALLOWED depends on the observation and timing of its subsequent forward flow based on the same logic.

Note that we assume firewalls implement silent drop. If a reject message is sent on the *same* flow (for example, rejecting a TCP SYN with RST + ACK), a reverse flow will be detected, and the previous forward flow will be marked as ALLOWED. However if the reject message is sent on a *different* flow (for example, rejecting a TCP SYN with an ICMP message), the forward flow will remain as PENDING.

For a unidirectional flow, the flow will be considered DROPPED if it is reported as DROPPED by any agent as in the case of bidirectional flows. However, since there is no matching reverse flow, the flow will have PENDING disposition status if both agents observe the flow.

### Violation Types

The flow dispositions are checked against the policies being analyzed to determine the final violation types.

A flow's violation type will be

- **Permitted**, if its disposition is ALLOWED or PENDING, and its deciding policy action is ALLOW,
- **Escaped**, if its disposition is ALLOWED, and its deciding policy action is DENY,
- **Rejected**, if its disposition is DROPPED or PENDING, and its deciding policy action is DENY,

A DROPPED status is assigned only to flows whose relevant agents explicitly report their DROPPED status. When there is no explicit report of dropping for agents, the flow receives PENDING status.

When disposition is PENDING:

- and policy action is DENY, then violation type is set to Rejected.
- and policy action is ALLOW, then violation type is set to Permitted.

For a bidirectional flow, if the policy violation types of forward and reverse part of the flow agree, only a single type is shown in the policy analysis or enforcement analysis page. Otherwise, forward and reverse are shown separately, such as PERMITTED:REJECTED.

### Example scenarios:

- Packets are dropped at the source-side enforcement.
  - In this case, the source side Secure Workload egress agent will report that the flow is DROPPED.
- Packets leave the source.
  - If there is only an agent on the source side, the flow will be reported as ALLOWED by the egress agent if a reverse packet is also observed by the agent within 60 seconds.
  - If there is a visibility-only agent on both the source and the destination side, the flow will be given a DROPPED disposition status, if and only if the ingress agent reports that the flow is DROPPED. Otherwise, the flow will be reported as ALLOWED.
  - Flow packets are received at the destination, but no reverse traffic.
    - If there is no destination side agent, the flow will receive a PENDING status. Otherwise, it will be assigned ALLOWED status.

## Suggested Steps for Investigating Flows

When drilling into specific flows when examining policy results, the following suggestions and filters may be helpful:

### 1. Focus on *ESCAPED FLOWS* initially:

**Escaped** flows require special attention as their actual flow dispositions differ from the intended actions based on the currently analyzed policies. Investigate to ensure that enforcing these policies does not block needed flows and adversely impact your applications.

Click the violation type, such as **Escaped**.

(Later, you can look at rejected and permitted flows as needed.)

Escaped flows can occur for many reasons, including but not limited to:

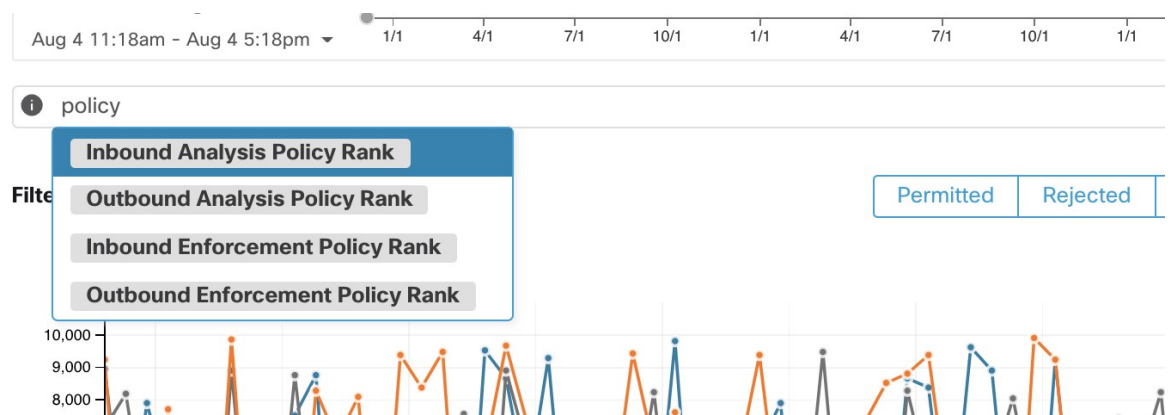
- Another policy higher in the priority order is taking effect
- The traffic is taking a different path than the route that your policies address, or
- The policy that you expect the traffic to hit is in a workspace that is not being analyzed (if you are looking at escaped flows on the Policy Analysis page) or enforced (if you are looking at escaped flows on the Enforcement page), for example in an ancestor scope or even in a secondary workspace in the same scope.

### 2. Identify flows that matched the catch-all policy (inbound and outbound) :

It is important to understand what flows are matched to catch-all policies, especially in an allow-list policy model. If these flows are legitimate but do not have explicit allow policies configured for them, you may want to add appropriate explicit policies in the corresponding inbound or outbound scopes. If they are suspicious flows, you want to quickly identify them and further investigate their details.

To focus on these flows, apply filters based on the *catch-all* value of **inbound\_policy\_rank** or **outbound\_policy\_rank**, depending whether you are looking at the inbound, outbound or both sides, shown below.

**Figure 297: Policy Analysis Filtering Options for Rank**



### 3. Filter out TCP flows with RST: *Fwd flags do not contain RST, Rev flags do not contain RST*

Some escaped TCP flows have RST flags set. These flows are reset by either their consumers or providers. They are unestablished connections without data exchange, but may be reported as **ALLOWED** because the agents see their handshaking packets. Since they do not have established connections to begin with,

they will not be affected when currently analyzed policies are enforced. Filtering out TCP flows that have the RST flag on either side allows you to focus on more meaningful and important escaped flows whose established connection will be blocked by the currently analyzed policies.

4. If most traffic is using IPv4, focus only on IPv4 flows:

Filter using *address type = IPv4, address type != IPv6*. It is also helpful to filter out *link-local* address.

5. Prioritize which flows to focus on in the next diagnostic step by identifying the most frequent hostnames, ports, addresses, scopes, and so on involved in the escaped traffic:

Select *Hostname, Ports, or Addresses* from the TopN feature pane. You can usually combine these with other filters to drill down to a particular type of traffic when diagnosing policies.

6. Search flow data for the hostnames, ports, protocols, etc. identified in the previous step

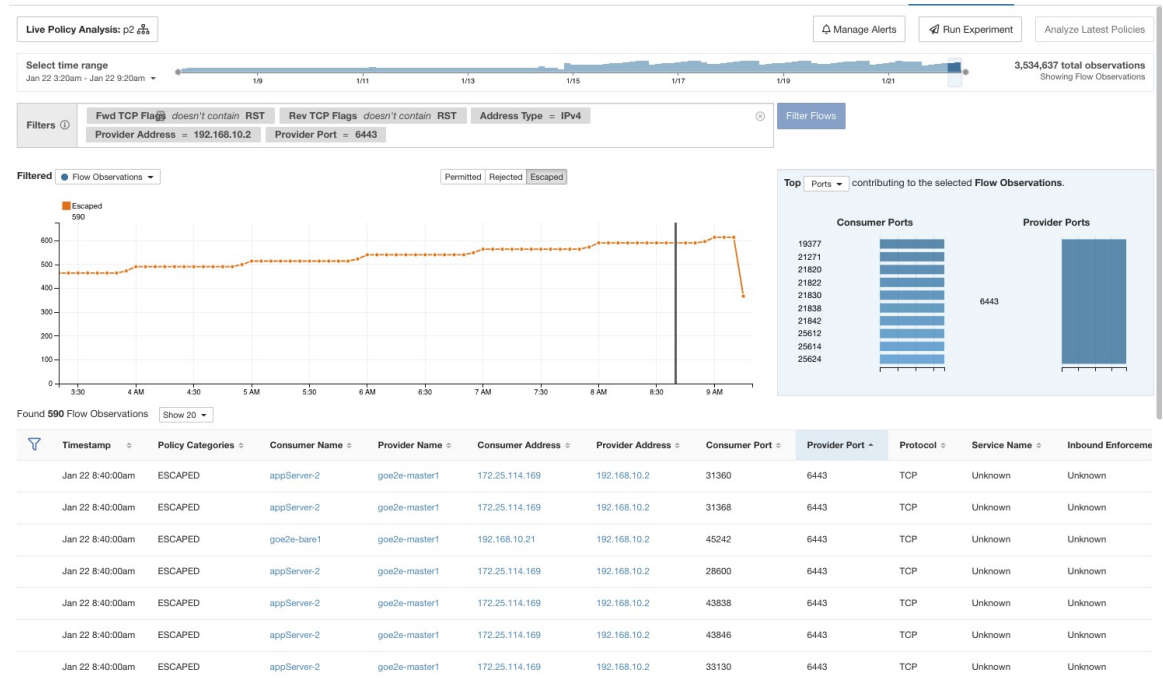
Once you have an idea about the top candidates based on the targeted flows' hostnames, port and and so on, you can choose to drill down into the flows by either applying drill-down filters directly from values given in the top N query window, or by manually entering relevant filters into the flow search filters bar. For example, *Consumer Hostname contains {something}, Provider Hostname contains {something}, Provider Port = {some port number}, Protocol = TCP Protocol != ICMP*

7. Check individual flows and quick analysis:

Finally, you can focus on a specific flow to examine its policy result by clicking the table row corresponding to the flow. Pay attention to the policies matched to the flow and the scopes of both the consumer and the provider addresses. If the policy action does not match your intended action, you need to create appropriate policies in workspaces associated with the consumer's and/or the provider's scopes to change the policy action.

The figure below shows an example workflow of narrowing down escaped flows using some of the filtering described above. The search input also supports “,” and “-” for Port, Consumer Address and Provider Address, by translating “-” into range queries.

Figure 298: Policy analysis diagnosis example



## Run Policy Experiments to Test Current Policies Against Past Traffic

If a known attack or other significant short-term traffic pattern occurred in the past, and you want to see how your current policies (or another versioned policy set) would have handled that traffic, you can use the Run Experiments feature.

### Before you begin



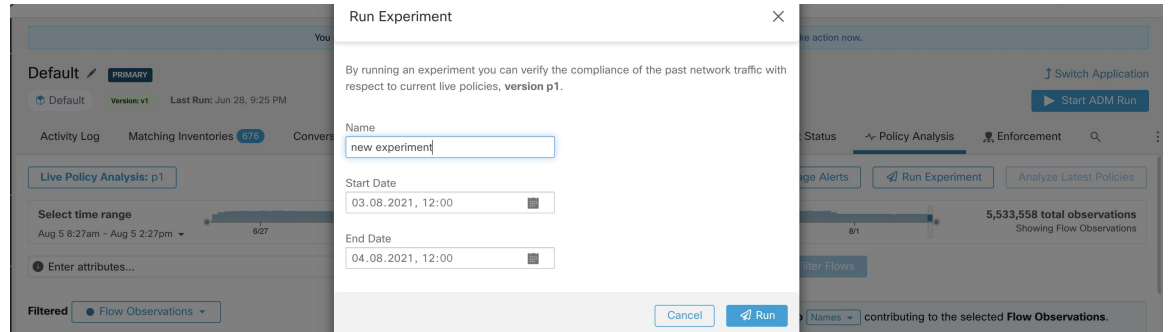
**Tip** As an alternative to this procedure, you can run automatic policy discovery again, including the relevant time range, and see what different policies are suggested.

### Procedure

- Step 1** Navigate to the Policy Analysis page of your selected workspace.
- Step 2** From the top of the page, select the policy version to test.
- Step 3** Click **Run Experiment**.
- Step 4** Enter a name and a duration for the policy experiment.



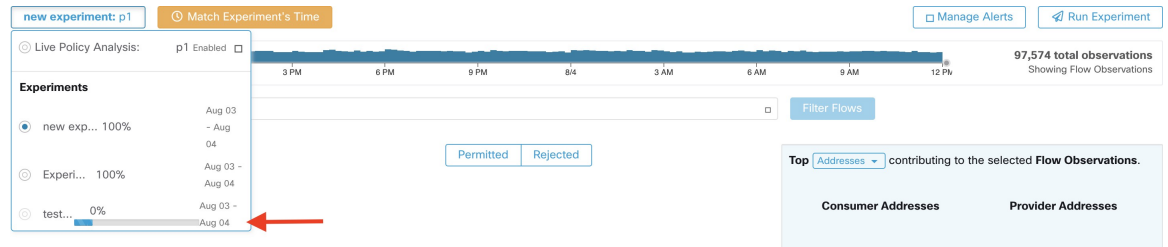
Figure 299: Run Experiment Form



This will start a new policy analysis job which goes back in time and re-analyzes all the flows in the selected duration against the selected versioned policy.

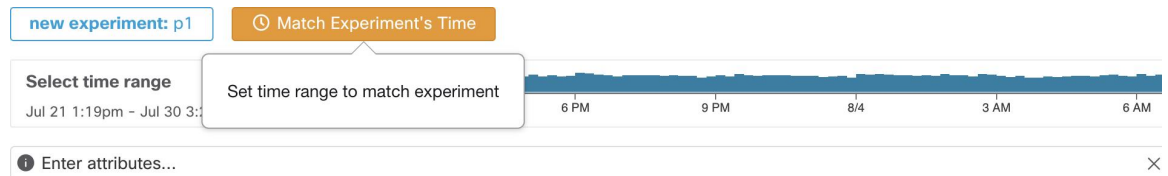
This job may take a few minutes, depending on the selected duration. The progress is shown in the policy selector menu. When the results are ready to be presented, you should be able to select the policy experiment like any other versioned policy and the time series charts showing different flow categories will be updated accordingly.

Figure 300: View Experiment Status



**Note** If you cannot see any flows when selecting a policy experiment, it might be due to time range mismatch, for example, the current time range of the charts is the past 1 hour, but the experiment duration is 6 hours in the past. To reset the time range to the duration of the experiment, click the clock icon next to the policy selector.

Figure 301: Match Time Range



## After Changing Policies, Analyze Latest Policies

Policy analysis does not automatically reflect policy changes in the workspace. When you are ready to analyze the current set of policies after making changes, click **Analyze Latest Policies** so policy analysis reflects the changes.

If the policies in the workspace have not changed since policy analysis was last initiated, or if policy analysis is not currently enabled, the Analyze Latest Policies button is not available. If the button is clickable, there are policy changes that have not yet been included in analysis.

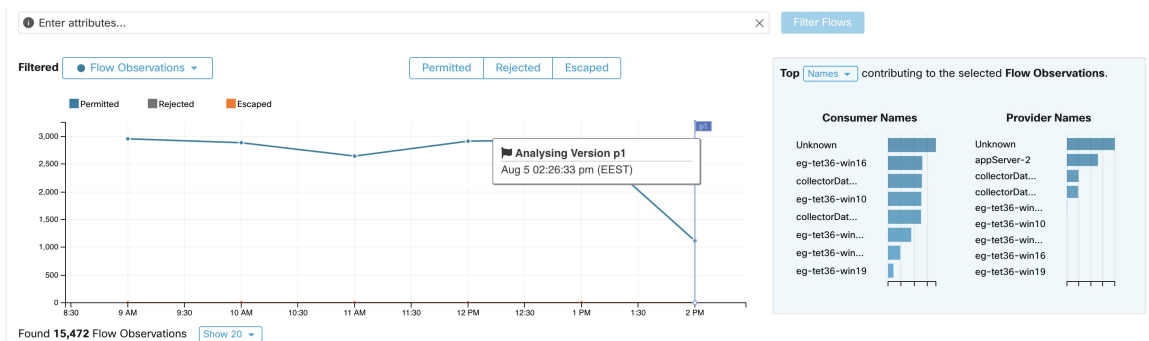
See also [View, Compare, and Manage Analyzed Policy Versions](#), on page 536.

## Policy Label Flags

On the policy analysis timeseries chart, policy label flags mark the point at which analysis was initiated and at each point analysis was re-initiated to reflect the latest policy and cluster changes.

Click a flag to view the version of the policies associated with that flag:

**Figure 302: Policy label flag in timeseries chart**



Clicking a policy label flag opens the corresponding version of the Policies page and displays the policies analyzed by that policy analysis version.

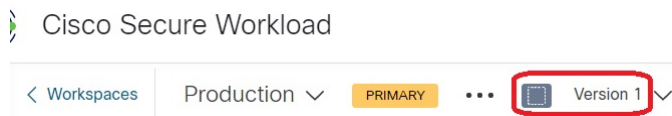
## View, Compare, and Manage Analyzed Policy Versions

Each time you analyze or re-analyze policies in a workspace after making changes, a new analysis version (p\*) is created.

For details about versioning, see [About Policy Versions \(v\\* and p\\*\)](#), on page 553.



### Procedure

- Step 1** Click **Defend > Segmentation**.
- Step 2** Navigate to the relevant scope and primary workspace.
- Step 3** Click **Manage Policies**.
- Step 4** The currently displayed version of the policies is shown at the top of the page:



The displayed version may be a policy discovery version, an analyzed policy version, or an enforced version.

- Step 5** You can:

|                                                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
|--------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| To display a different version of the policies:  | <p>Click the current version and choose a different version.</p> <p>For descriptions of the versions, see <a href="#">About Policy Versions (v* and p*)</a>, on page 553.</p> <p><b>Important!</b> If you choose a v* version, see <a href="#">View, Compare, and Manage Discovered Policy Versions</a>, on page 466 instead of this topic, including the important caveat at the end of the topic.</p>                                                                                       |
| To view details about the analyzed versions:     | <ol style="list-style-type: none"> <li>a. Click <b>View Version History</b> at the top of the page beside the current version.</li> <li>b. Click the <b>Published Versions</b> tab to see the versions of analyzed and enforced policies.</li> <li>c. To view log entries for a version, click the link in the version. <ul style="list-style-type: none"> <li>Pale green rows represent analysis activity.</li> <li>Bright green rows represent enforcement activity.</li> </ul> </li> </ol> |
| To compare two versions to see what has changed: | <ol style="list-style-type: none"> <li>a. Click <b>Compare Revisions</b>.</li> <li>b. Choose the versions to compare. <ul style="list-style-type: none"> <li>You can compare the latest draft version, analyzed and enforced versions.</li> </ul> </li> <li>c. For result details, see <a href="#">Comparison of Policy Versions: Policy Diff</a>, on page 556.</li> </ol>                                                                                                                    |
| To delete an unwanted version:                   | <p>Click  for the version and choose <b>Delete</b>.</p> <p>Published policy versions (p* versions) can be deleted as long as the version is not being actively analyzed or enforced.</p>                                                                                                                                                                                                                   |
| To export a version:                             | <p>Click  for the version and choose <b>Export...</b></p> <p>See also <a href="#">Export a Workspace</a>, on page 471.</p>                                                                                                                                                                                                                                                                                 |

### What to do next

When you are done working with versions, change the version at the top of the workspace page to the latest discovered policy version (v\*).

This avoids unintentional deletion of discovered policy versions and allows you to manually create policies in the workspace.

## Activity Logs of Policy Analysis

All workspace users may view activity logs associated with changes done on the policy analysis page in the workspace history (see [Activity Logs and Version History](#) ).

- Enable policy analysis

**Figure 303: Enable policy analysis**

You started policy analysis to version p1

2:28 PM

- Disable policy analysis

**Figure 304: Disable policy analysis**

You stopped policy analysis

2:32 PM

- Update policy analysis

**Figure 305: Update policy analysis**

You updated policy analysis to version p1

2:24 PM

## Enforce Policies

Secure Workload can enforce policies using:

- [Deploy Software Agents on Workloads, on page 9](#) installed on individual workloads:
  - Linux
  - Windows
  - Kubernetes/OpenShift

For technical details about how agents work on each platform, see [Policy Enforcement with Agents, on page 41](#) and [Enforcement on Containers, on page 546](#).

- Cloud connectors:
  - AWS through [AWS Connector, on page 231](#)
  - Azure through [Azure Connector, on page 244](#)
- Integrate load balancers through an external orchestrator:
  - [F5 BIG-IP, on page 155](#)
  - [Citrix Netscaler, on page 161](#)
- Integration with [Cisco Secure Firewall Management Center, on page 330](#)
- Streaming to third-party orchestrators for enforcement in third-party infrastructure



### Caution

When you enforce policies, the system inserts new firewall rules on affected hosts and deletes any existing rules on the relevant hosts.

## Check Agent Health and Readiness to Enforce

Some of these checks can be done before or after enforcing policy.

Permissions may be required to modify agent or connector capability; see the requirements and prerequisites in the relevant chapters.

You do not need to perform these checks for any workloads on which you do not intend to enforce policies.

| Verify That:                                                                                       | More Information                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
|----------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Agents are installed on all workloads in the scope that are associated with the enforced workspace | <p>Click <b>Defend &gt; Segmentation</b> and navigate to the relevant scope and workspace. Click <b>Matching Inventories</b>, then click <b>IP Addresses</b>.</p> <p>IP addresses on this tab generally do not have agents that are installed, and agents generally must be installed to enforce policy.</p> <p>Exceptions: Enforcement occurs for the following types of inventory that appears on the IP Addresses tab:</p> <ul style="list-style-type: none"> <li>• Cloud-based inventory on which policy is enforced using a cloud connector. (Installing agents on individual workloads is optional.)</li> <li>• Kubernetes addresses appear in the IP Addresses list if agents are installed on individual workload pods; Kubernetes inventory with installed agents appears on the Pods tab.</li> </ul> |
| The installed agent version is current and supported                                               | <p>For an overview of installed agent versions, click <b>Manage &gt; Agents</b>, then click <b>Distribution</b> and look at the <b>Agent Software Version Distribution</b> chart.</p> <p>For details, click <b>Manage &gt; Agents</b>, then click <b>Agents List</b>.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| Installed agents have enforcement capability                                                       | <p>Click <b>Manage &gt; Agents</b>, then click <b>Convert to Enforcement Agent</b>.</p> <p>In the <b>Filter</b> box, enter <b>Agent Type = Deep Visibility</b></p> <p>Convert any agents that must enforce policy.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| Enforcement is enabled for all agents                                                              | <p>(This requirement is distinct from ensuring that agents have enforcement capability and from enabling enforcement in the workspace.)</p> <p><b>Important!</b> Depending on your deployment, this may need to be done before or after you enforce the workspace.</p> <p>See that the Verify Enforcement is Enabled for Agents section.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| Enforcement is enabled for nonagent enforcement mechanisms                                         | <p><b>Important!!!</b> Do not enable enforcement on cloud connectors without agents until AFTER you enforce policy on the workspace.</p> <p>External orchestrators that support enforcement must also be enabled before they can enforce.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |

| Verify That:                                                                                           | More Information                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
|--------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| The <b>Preserve Rules</b> setting in the Agent Config Profile is appropriate for the workload platform | <ul style="list-style-type: none"> <li>• For Kubernetes/OpenShift, see Enforcement on Containers section.</li> <li>• For other platforms, see information for each platform in Software Agents section.</li> </ul> <p>Tip: Search this document for "Preserve Rules" to find useful information.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| (After the workspace is enforced) All agents have received the applicable policies for the workload    | See the Verify Enforced Policies are being pushed to Agents section.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| Agents are healthy                                                                                     | <p>In addition to the sources above, the following locations have information about agent health:</p> <ul style="list-style-type: none"> <li>• Click <b>Manage &gt; Agents</b>, then click <b>Monitor</b>.</li> </ul> <p>Look at the information under <b>Enforcement Agents</b>. For details, see the Agent Monitoring section.</p> <li>• Click <b>Manage &gt; Agents</b>, then click <b>Distribution</b>.</li> <p>Choose the agent type from the top of the page.</p> <p>For information about this page, see the Agent Status and Statistics.</p> <li>• Click <b>Organize &gt; Scopes and Inventory</b>, filter to find a specific workload of interest, and click the IP address.</li> <p>The <b>Workload Profile</b> page opens in a separate browser window including an Agent Health panel.</p> <p>For details, see Workload Profile section.</p> |

## Enable Policy Enforcement



**Caution** Enforcing policies delete existing firewall rules and write new firewall rules on every workload in the scope that is affected by this workspace.

If you have not properly verified that your policies are working correctly, enforcing policies can change the way that your applications work and disrupt business operations.

### Before you begin

- Initially, when you enforce policies, consider setting the catch-all to Allow. Then, monitor traffic to see what matches the catch-all rule. When no necessary traffic is matching the catch-all rule, you can set the catch-all to Deny.
- If you enforce workspaces in multiple scopes at once, you can enforce only analyzed workspaces. If you enforce a single workspace using the second method that is described in the procedure below, analyzing the policies in the workspace before you enforce them is recommended but not required.

See [Live Policy Analysis](#) and subtopics.

- The wizard for enforcing a single scope is more detailed than the wizard that offers the option to enforce multiple scopes simultaneously. If you require the features in [Policy Enforcement Wizard, on page 544](#), use the second method that is described in the procedure below.

- **IMPORTANT!** Verify that the policies are correct.

Policy results in any workspace may be affected by enforced policies in other scopes. Before policy enforcement is enabled on a workspace, the Policy Enforcement page shows how flows are affected by enforced policies in the workspaces associated with other scopes. For example, a broad “Production hosts should not talk with Non-Production hosts” policy in the enforced workspace of a parent scope may impact traffic on workloads belonging to an application in a child scope.

If no new information is being shown in the Enforcement charts, make sure that the correct time range is selected.

For information about the information you see on the Enforcement page, see [Live Policy Analysis](#) and subtopics. (The same information for Live Analysis also applies to the Policy Enforcement page.)

If live analysis results differ from results on the Enforcement page, make sure the scopes, policy versions, and time range being analyzed are the same as the scopes, policy versions, and time range being used to generate results on the Enforcement page.

- Understand how agents enforce policies on each platform. See:
  - For Windows and Linux workloads, see [Policy Enforcement with Agents, on page 41](#) and subtopics.
  - For Kubernetes and OpenShift, see [Enforcement on Containers, on page 546](#).
  - For load balancers, see [Policy enforcement for Citrix Netscaler, on page 163](#) and [Policy enforcement for F5 BIG-IP, on page 157](#).
  - For cloud-based workloads configured using cloud connectors, see:
    - [Best Practices When Enforcing Segmentation Policy for AWS Inventory, on page 240](#) and linked topics.
    - [Best Practices When Enforcing Segmentation Policy for Azure Inventory, on page 251](#) and linked topics.

- You must have the required permissions to enforce policies:

You must have the Enforce ability or higher for the scope. Users with other abilities on the scope can still view this page but will not be able to enforce (or disable) new policies.

- Verify that all relevant installed agents and other enforcement endpoints such as cloud connectors are ready to enforce policy. For a list of agent health and readiness checks, see [Check Agent Health and Readiness to Enforce, on page 539](#).



---

**Note** Some of these checks must wait until after enforcement; for example, you should enable enforcement on cloud connectors only after you have enabled enforcement in the workspace. For installed agents, you will typically enable enforcement in the agent configuration before you enforce the workspace.

---

### Procedure

---

**Step 1** From the navigation pane, choose **Defend > Segmentation**.

**Step 2** You can enforce policies for one scope or for multiple scopes at the same time:

To enforce policy for multiple scopes at the same time:

(Only workspaces that have been analyzed can be enforced using this process.)

- a) Click the caret on the right side of the page to display the Tools pane:
- b) Click **Enable Enforcement**.
- c) Click **Next** to start the wizard.
- d) Select one workspace to enforce.  
(The option to enforce workspaces for additional scopes is on the last page of the wizard.)
- e) Click **Next**.
- f) Choose the version of that workspace to enforce, then click **Next**.
- g) To simultaneously enforce policies for another scope, click + **Add Another Workspace** and complete the steps.

Repeat as needed for additional scopes.

- h) Click **Accept and Enforce**.

To enforce policies for a single scope:

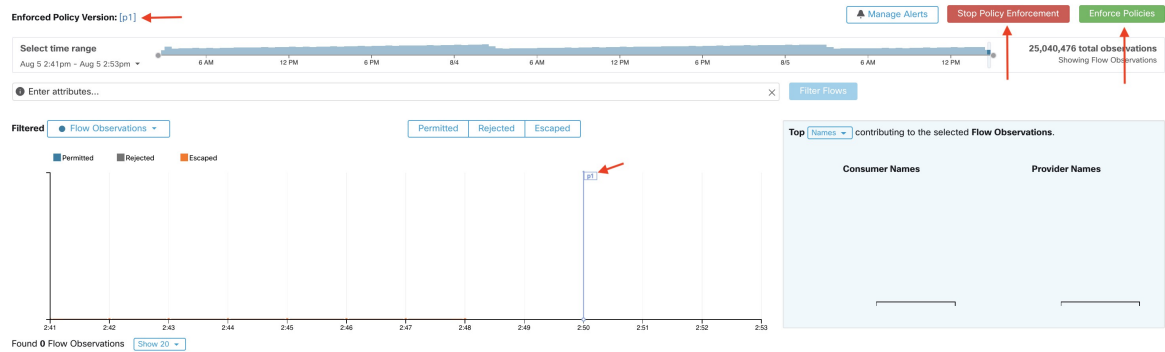
- a) Navigate to the primary workspace for the scope for which you want to enforce policy.
- b) Click **Manage Policies**.
- c) Click **Enforcement**.
- d) Click **Enforce Policies**.
- e) Step through the wizard.

For wizard details, see [Policy Enforcement Wizard, on page 544](#).

**Step 3** Click **Accept and Enforce** on the final page of the wizard to push new firewall rules to the assets that are affected by policies in this workspace. A label flag is created at the time of enforcement:



Figure 306: Policy Enforcement Page with Enforcement Enabled



You may need to refresh the page to see the flag.

### What to do next

- If you enforced policy for a single workspace, consider whether workspaces for other scopes must also be enforced to achieve the expected enforcement outcomes.

For example, workspaces for ancestor scopes or scopes that include workloads that are involved in cross-scope policies may also need to be enforced.

- Enforcement will not occur until enforcement is enabled for the agents, cloud connectors, and/or external orchestrators that enforce the policies:
  - For workloads with installed agents, enforce policy in the agent config for the relevant scopes and inventory filters. See [Software Agent Config, on page 65](#) and subtopics.
  - For cloud-based workloads configured using cloud connectors, see:
    - [Best Practices When Enforcing Segmentation Policy for AWS Inventory, on page 240](#) and linked topics.
    - [Best Practices When Enforcing Segmentation Policy for Azure Inventory, on page 251](#) and linked topics.
  - For Kubernetes and OpenShift, see:
    - [Enforcement on Containers, on page 546](#)
    - [Software Agent Config, on page 65](#)
  - For load balancers, see:
    - [Policy enforcement for F5 BIG-IP, on page 157](#)
    - [Policy Enforcement for F5 Ingress Controller, on page 158](#)
    - [Policy enforcement for Citrix Netscaler, on page 163](#)
- Check to be sure enforcement is working as expected. See [Verify That Enforcement Is Working as Expected, on page 547](#).

- Configure alerts so you are notified of any issues, for example if flows are rejected after enforcement is enabled.

## Policy Enforcement Wizard

When you enforce policies for a single workspace from the Enforcement page of the workspace, the policy enforcement wizard lets you:

- Review policies before they are implemented on the workloads.  
This includes policies that are inherited from ancestor scopes.
- Download policy changes for review.
- Compare policy versions.
- Choose which analyzed version of the workspace to enforce.
- Roll back policies to a previous version.

Steps in the policy enforcement wizard:

### 1. Select Policy Updates

You can select which version of policies to be enforced on the workloads.

The difference between the currently enforced policies and policies in the selected version is displayed.

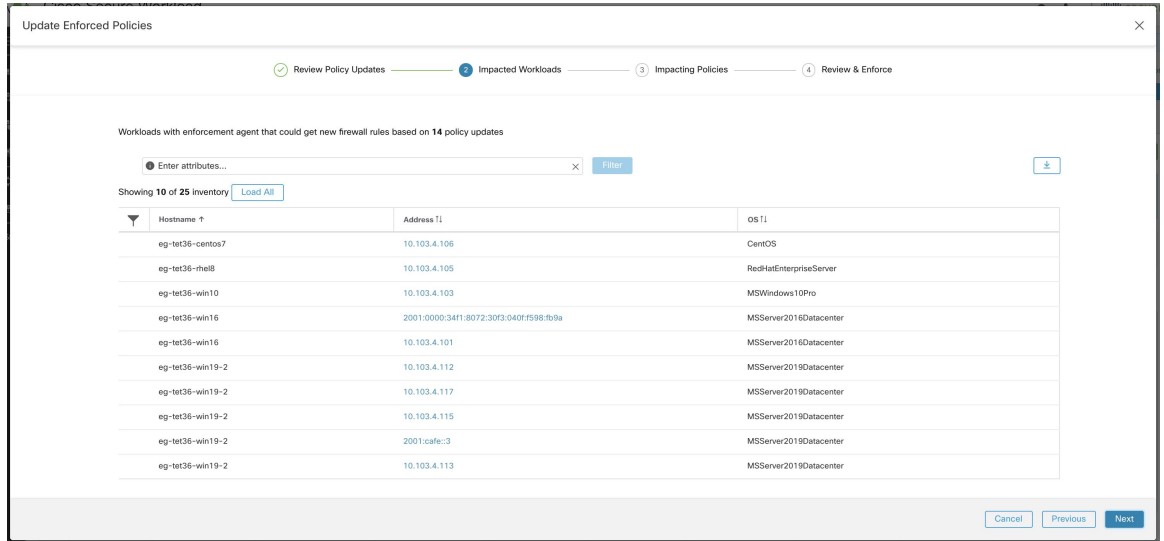
Similarly to the [Comparison of Policy Versions: Policy Diff](#), you can filter and review the policy changes and download them as CSV.

### 2. Impacted Workloads

This step shows the workloads that will be affected by the new firewall rules generated from the selected policy changes. The result comes from searching all the workloads that have enforcement agents within the union of the consumers/providers of the selected policy changes.

The number of potentially impacted workloads cannot exceed the total number of workloads in the scope. However, the actual impacted workloads might be smaller due to other factors such as agent config intents.

Figure 307: List of Impacted Workloads

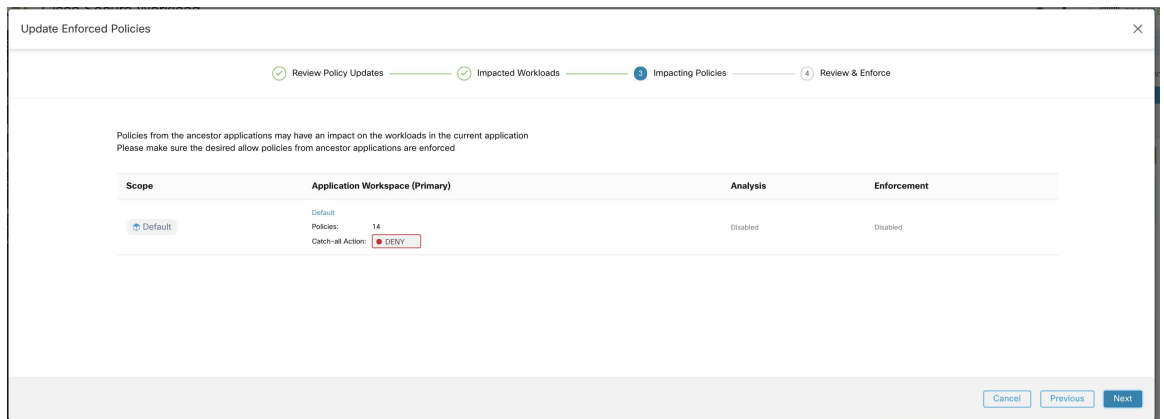


For more details on viewing, filtering, and downloading inventory items, see [Manage Inventory for Secure Workload, on page 333](#).

### 3. Impacting Policies

Policies from the ancestor workspaces may impact workloads in the current workspace. Therefore, you should make sure the desired allow policies from ancestor workspaces are enforced.

Figure 308: List of ancestor workspaces and enforced versions



### 4. Review & Accept

This final step summarizes the policy changes to be enforced, the number of potentially impacted workloads, and the catch-all action that will be enforced. When you click **Accept and Enforce**, the policies in the workspace will be used to calculate the new firewall rules that will be configured on the relevant workloads.

You have the option to provide a name, description, and reason for action for the newly enforced policies for future reference. In case of rollback, you can provide only the reason, as name and description for a past version cannot be changed.

Figure 309: Review the Summary and Enforce Policy Changes

## Enforcement on Containers

For an overview of the steps that are required to set up segmentation on container-based workloads that are managed by Kubernetes and OpenShift, see [Set Up Microsegmentation for Kubernetes-Based Workloads](#), on page 7.



**Attention** Agents running on Kubernetes/OpenShift hosts must be configured to preserve existing rules.

In order to prevent enforcement from interfering with iptables rules added by Kubernetes, the agent must be configured with a profile that has the **Preserve Rules** option enabled. See [Create an Agent Configuration Profile](#)

When enforcing policies on containers, Secure Workload allows Kubernetes/OpenShift service abstractions to be used as providers. Internally, the policies for service abstractions are transformed into rules for the provider pods and the nodes they are running on. This transformation depends on the type of the Kubernetes/OpenShift service, and it is dynamically updated whenever changes are received from the API server.

The following example illustrates the flexibility made possible by this feature. Consider the following policy which allows traffic from all hosts and pods with the label *environment = production* to a Kubernetes service of type *NodePort* with the name *db* which exposes TCP port 27017 on a set of pods.

| Consumer                                                                      | Provider                              | Protocol/Port | Action |
|-------------------------------------------------------------------------------|---------------------------------------|---------------|--------|
| environment =<br>production<br>OR<br>orchestrator_environment<br>= production | orchestrator_system/service_name = db | TCP 27017     | Allow  |

This policy would result in the following firewall rules:

- On hosts and pods that are labeled with *environment = production*, allow outgoing connections to all Kubernetes nodes of the cluster to which the service belongs. This rule uses the node port that is assigned to this service by Kubernetes.
- On pods with the label *environment = production*, allow outgoing connections to the ClusterIP assigned to this service by Kubernetes. This rule uses the port that is exposed by the service (TCP 27017).
- On Kubernetes nodes of the cluster to which the service belongs, allow outgoing connections to the provider pods. This rule uses the target port that is exposed by the service (TCP 27017).
- On pods providing the service db, all incoming connections from all kubernetes nodes and consumer hosts and pods. This rule uses the target port that is exposed by the service (TCP 27017).

Changes to the type of the service, ports, and set of provider pods will immediately be picked up by Secure Workload rule generator and used to update the generated firewall rules.



---

**Caution** Policies including Kubernetes/OpenShift inventory must be designed carefully to avoid conflicting with the internal operation of the kubernetes cluster.

Kubernetes/OpenShift items imported by Secure Workload include the pods and services constituting the kubernetes cluster (for example, pods in the kube-system namespace). This allows precise policies to be defined to secure the kubernetes cluster itself, but it also means that badly designed policies can affect the operation of the cluster.

---

## Verify That Enforcement Is Working as Expected

### Check Agents

See [Check Agent Health and Readiness to Enforce, on page 539](#).

### Check for Escaped and Rejected Flows

In the menu on the left side of the screen, click **Overview**.

On the **Security Dashboard** page, look at the **Segmentation Compliance Score**.

If this is less than 100, you may have escaped or rejected flows, either of which indicates a policy configuration issue.

For details, see [Segmentation Compliance Score, on page 694](#).

For more information about investigating these situations, see [Policy Analysis Results: Understand the Basics, on page 529](#) and subtopics. (The information in these topics applies to enforced policies shown on the Enforcement tab and to analyzed policies shown on the Policy Analysis tab.)

Add any missing policies, or modify existing policies, for example, by adding additional protocols/ports, to allow required legitimate traffic.

Then reanalyze before reenforcing.

## View Enforced Policies for a Specific Workload (Concrete Policies)

Use this procedure to view all enforced policies for a specific workload (that is, the *concrete policies* for that workload). This view is useful because all policies in a workspace may not apply to every workload in the workspace, and because policies in multiple workspaces may apply to a particular workload (for example, inherited policies in parent or ancestor scopes).

Concrete policies are listed in priority order. For more information about the effects of priority, see the Policy Priorities section.

### Before you begin



**Note** Concrete policies include only policies in enforced workspaces. If a workspace is not enforced, any policies that would apply to the workload if the workspace were enforced do not appear in the list.

### Procedure

**Step 1** You can navigate to the Concrete Policies page for a workload from the Inventory page or from the workspace:

To navigate from the Scopes and Inventory page:

- a) Choose **Organize > Scopes and Inventory**.
- b) Search for the IP address of the workload of interest and click it.

The Workload Profile opens in a separate tab.

In general, except for cloud-based workloads that are managed without agents, Kubernetes, and OpenShift workloads, if the IP address appears in the **IP Addresses** tab and not in the **Workloads** tab, this means that an agent is not installed on the workload, so policies cannot be enforced, and there is no concrete policies list.

To navigate from the Segmentation page:

- a) Choose **Defend > Segmentation**.
- b) Click the scope.
- c) Click the Primary workspace.
- d) Click **Manage Policies**.
- e) Click the **Matching Inventories** tab.
- f) Search for the IP address of the workload of interest and click it.
- g) In the panel that opens on the right, click **View Workload Profile**.

The Workload Profile opens in a separate tab.

**Step 2** From the menu on the left side of the Workload Profile page, click **CONCRETE POLICIES**.

**Step 3** Click a row to view details.

For more information, see the Concrete Policies tab.

**Step 4** To see the amount of traffic that has hit each policy:

- a) Click **Fetch All Stats**.
- b) Click each policy of interest.

- Step 5** To view information about Kubernetes or OpenShift workloads, click **CONTAINER POLICIES**.
- 

#### What to do next

Choose **Monitor > Enforcement Status** for status of concrete policies, for example to see if any policies have been skipped. For details, see the Enforcement Status section.

## Verify That Enforcement Is Enabled for Agents

### Procedure

---

- Step 1** Click **Defend > Enforcement Status**.
- Step 2** To view only the enforcement status for a specific scope, toggle the **Filter by Scope** control and select a scope.
- Step 3** Look at the **Agent Enforcement Enabled** chart.  
If the chart shows that any agents are **Not Enforced**, continue with this procedure.  
Otherwise, skip the rest of this procedure, as all agents are enabled for enforcement.
- Step 4** Click the orange **Not Enforced** section of the chart to display the affected workloads in the table below the chart.
- Step 5** Enable enforcement on these workloads by modifying the Agent Config profile.  
See [Create an Agent Configuration Profile, on page 67](#).
- 

## Verify That Enforced Policies Are Being Pushed to Agents

For enforcement to occur, policies specific to each workload must be successfully pushed to the agent installed on that workload. Status is also shown for policy enforcement that is managed by cloud connectors even if agents are not installed.

### Before you begin

Enforce policies for at least one scope.

### Procedure

---

- Step 1** Click **Defend > Enforcement Status**.
- Step 2** To view only the enforcement status for a specific scope, toggle the **Filter by Scope** control and select a scope.
- Step 3** Look at the **Agent Concrete Policies** chart.  
If the chart shows that any are **Skipped**, continue with this procedure.  
Otherwise, skip the rest of this procedure.

**Step 4** To display the list of workloads affected by this issue, click the red **Skipped** slice of the chart. The affected workloads are listed in the table below the charts.

**Step 5** To see the reasons for this issue:  
For each workload in the search results, click the **(i)** button beside **Skipped** in the **Concrete Policies** column.

| Error Message                               | More Information                                                                                                                                                                         |
|---------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Agent doesn't have Windows OS               | At least one policy that is applicable only to Windows workloads includes consumers and/or providers that are not running Windows OS.<br><br>Remove those workloads from those policies. |
| Maximum number of policies has been reached | See <a href="#">If There Are Too Many Policies for the Agent, on page 550</a> .                                                                                                          |

#### What to do next

(Optional) Configure an alert so you are notified if this situation occurs in future. See [Configure Alerts, on page 647](#).

## If There Are Too Many Policies for the Agent

If the complete set of applicable concrete policies cannot be pushed to a particular agent, the latest version of the policies is not pushed.

Background: There is a limit to the number of policies supported on each agent. Limits also apply to policies enforced using cloud connectors. You may find the information in [Configuration Limits in Secure Workload, on page 951](#) helpful.

#### Before you begin

Use this procedure to resolve this problem if [Verify That Enforced Policies Are Being Pushed to Agents, on page 549](#) indicates that the agent cannot accommodate the full set of enforced policies.

#### Procedure

**Step 1** Navigate to the primary workspace for an affected scope.

**Step 2** Modify the policies in the primary workspace:

Try to reduce the number of policies and reduce any long lists of IP addresses in consumer or provider.

For example, consolidate existing policies, and/or base policies on subnets rather than on huge lists of IP addresses.

For policies enforced using a cloud connector, you may also be able to increase any limits that are imposed by the platform. See the documentation for your cloud platform.

**Step 3** After you have made changes, enforce the latest version of the workspace and check again for skipped policies.



**Step 4** Repeat this procedure for any other scopes experiencing this issue.

## Modify Enforced Policies

### Enforce New and Revised Policies

If you must revise policies after enforcement, generally you will make the changes in the same primary workspace. Then, review your changes carefully, and analyze the workspace again to be sure it has the effect you expect. When you are confident that the changes will have the desired effect, click the **Enforce Latest Policies** button in the upper right of the page.

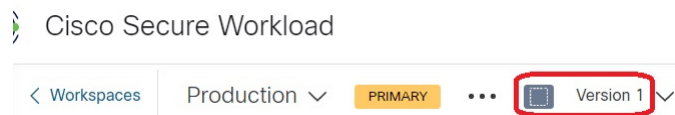
### View, Compare, and Manage Enforced Policy Versions

Each time that you enforce or reenforce policies in a workspace after making changes, a new version (p\*) is created.

For detailed information about versioning, see [About Policy Versions \(v\\* and p\\*\)](#), on page 553.

#### Procedure



- Step 1** Click **Defend > Segmentation**.
- Step 2** Navigate to the relevant scope and primary workspace.
- Step 3** Click **Manage Policies**.
- Step 4** The currently displayed version of the policies is shown at the top of the page:



The displayed version may be a policy discovery version, an analyzed policy version, or an enforced version.

- Step 5** Do one of the following:

|                                                 |                                                                                                                                                                                                                                                                                                                                                                                                         |
|-------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| To display a different version of the policies: | <p>Click the current version and choose a different version.</p> <p>For descriptions of the versions, see <a href="#">About Policy Versions (v* and p*)</a>, on page 553.</p> <p><b>Important!</b> If you choose a v* version, see <a href="#">View, Compare, and Manage Discovered Policy Versions</a>, on page 466 instead of this topic, including the important caveat at the end of the topic.</p> |
|-------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

|                                                  |                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|--------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| To view details about the analyzed versions:     | <ol style="list-style-type: none"> <li>Click <b>View Version History</b> at the top of the page beside the current version.</li> <li>Click the <b>Published Versions</b> tab to see the versions of analyzed and enforced policies.</li> <li>To view log entries for a version, click the link in the version.<br/>Pale green rows represent analysis activity.<br/>Bright green rows represent enforcement activity.</li> </ol> |
| To compare two versions to see what has changed: | <ol style="list-style-type: none"> <li>Click <b>Compare Revisions</b>.</li> <li>Choose the versions to compare.<br/>You can compare the latest draft version, analyzed and enforced versions.</li> <li>For result details, see <a href="#">Comparison of Policy Versions: Policy Diff, on page 556</a>.</li> </ol>                                                                                                               |
| To delete an unwanted version:                   | <p>Click  for the version and choose <b>Delete</b>.</p> <p>Published policy versions (p* versions) can be deleted as long as the version is not being actively analyzed or enforced.</p>                                                                                                                                                        |
| To export a version:                             | <p>Click  for the version and choose <b>Export...</b></p> <p>See also <a href="#">Export a Workspace, on page 471</a>.</p>                                                                                                                                                                                                                      |

### What to do next

When you are done working with versions, change the version at the top of the workspace page to the latest discovered policy version (v\*).

This avoids unintentional deletion of discovered policy versions and allows you to manually create policies in the workspace.

## Revert Enforced Policies to an Earlier Version

To roll back enforced policies to a previous version, follow one of the processes that are described in [Enable Policy Enforcement, on page 540](#) and choose an earlier version to enforce.

## Disable Policy Enforcement

- **To disable policy enforcement for multiple scopes simultaneously:**

Follow the procedure for enforcing policy in multiple scopes simultaneously, as described in [Enable Policy Enforcement, on page 540](#). On the Select Version page of the wizard, click **Select a version** and choose **Disable enforcement**.

- **To disable policy enforcement for a single scope:**

Navigate to the Policy Enforcement page for the scope's primary workspace and click the red **Stop Policy Enforcement** button. This writes new firewall rules to assets in the scope based on enforced policies in ancestor workspaces. A Label Flag with an 'x' will be created on the time series chart.

## Enforcement History

Enforcement History provides a list of changes to the list of enforced workspaces and their version.

To view the enforcement history:

1. Click the caret at the right side of the Segmentation page to expand the Tools menu.

2. Click **Enforcement History**.

Each section describes an event and displays a summary of what has changed.

3. Click an event for details about all the policies that were enforced at that time.

**Figure 310: Enforcement history view**

The screenshot shows the 'Enforcement History' interface. At the top, there is a search bar with the placeholder 'Enter attributes...' and a 'Filter' button. To the right, there is a date and time selector set to '03.08.2021, 15:03:48' with an 'Apply' button. Below this, there are three event entries:

- Jul 1 2021 01:50:22 am (EEST)**: Change by: Site Admin <team-x-all@tetrationanalytics.com>. Action: UPDATE (eg-app1) p1 → p2.
- Jun 29 2021 12:03:04 am (EEST)**: Change by: Site Admin <team-x-all@tetrationanalytics.com>. Action: ENABLE (eg-app1) p1.
- Jun 28 2021 11:35:08 pm (EEST)**: Change by: Site Admin <team-x-all@tetrationanalytics.com>. Action: The enforcement profile was refreshed to notify the agents.

At the bottom, there are 'Back to Start' and 'Next »' buttons.

## About Policy Versions (v\* and p\*)

Policy versions are sometimes called workspace versions.

### Displayed Version

The version of the policies (and clusters) that you are currently working with is shown at the top of the workspace page:

The screenshot shows the 'Cisco Secure Workload' workspace page. At the top, there is a breadcrumb navigation: '< Workspaces App1-primary PRIMARY ... Version 0'. The 'Version 0' text is highlighted with a red box.

- V\* versions are generated by automatic policy discovery




For details, see below

- P\* versions are analyzed and/or enforced versions

For details, see below

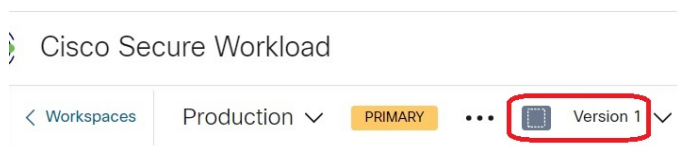
The following icons may appear beside the version number:

Table 27: Version Icons

|                                                                                   |                                                                        |
|-----------------------------------------------------------------------------------|------------------------------------------------------------------------|
|  | Indicates the version of the policies that is currently being analyzed |
|  | Indicates the version of the policies that is currently being enforced |
|  | Indicates the latest version of automatically discovered policies      |
| (no icon)                                                                         | Indicates that the version is not the latest version of its type       |

Examples:

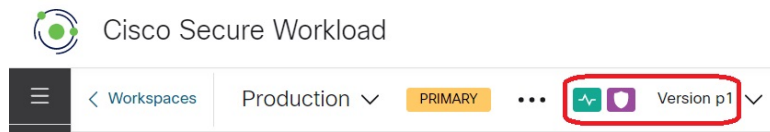
- Displayed version is the latest discovered version of the policies:



- Displayed version is the version of the policies that is currently being analyzed:



- Displayed version is the version of the policies currently being analyzed and enforced:



**Policy Discovery Version (v\*)**

Each time you automatically discover policies for a workspace, the version (v\*) increments.

The first time you automatically discover policies, version 1 is generated, and all modifications after that run, such as editing or approving clusters (but not a rerun), are also grouped under version 1. When you subsequently automatically discover policies, a new version is generated (unless discovery failed).

The v\* version also increments if you import policies.

To work with v\* versions, see [View, Compare, and Manage Discovered Policy Versions, on page 466](#).

**Published Policy Version (p\*)**

The term "published" policy version (p\*) for a workspace can refer to either:

- The version of the policies that was analyzed, or
- The version of the policies that was enforced

These are two separate but parallel versions that depend on the context:

- Policy version for analysis:

Each time you analyze policies in a workspace, or click **Analyze Latest Policies** after making a change, the system takes a snapshot of all the clusters and policies that are defined in that workspace, and the "published" policy version (p\*) number for analysis increments. The latest **Live Policy Analysis** version is shown at the top-left side of the page on the Policy Analysis tab of the primary workspace.



- Policy version for enforcement:

Each time you enable enforcement of the policies in a workspace, or enable enforcement again after making changes, the "published" policy version (p\*) for enforcement becomes the number of the analyzed version that you choose in the enforcement wizard. So, if you enforce analyzed version 5, the enforced version is also version 5, even if it is, for example, the first time policy has been enforced for the workspace. The current **Enforced Policy Version** is shown at the top-left side of the page on the Enforcement tab of the primary workspace.



### Managing Published (p\*) Versions

Published policy versions cannot be edited, only deleted entirely.



- Note** Published policy versions (p\*) are limited to 100 total. Once this limit is reached, you must delete old versions. To manage and delete p\* versions, see [View, Compare, and Manage Analyzed Policy Versions, on page 536](#), or [View, Compare, and Manage Enforced Policy Versions, on page 551](#). You can also use the API to delete published versions.

## Comparison of Policy Versions: Policy Diff

To compare policies, see any of the following topics: [View, Compare, and Manage Discovered Policy Versions, on page 466](#), [View, Compare, and Manage Analyzed Policy Versions, on page 536](#), or [View, Compare, and Manage Enforced Policy Versions, on page 551](#)

Policy changes will be displayed in three categories: Absolute, Default and Catch All. In the comparison table:

- Different services that belongs to the same policy are grouped together
- Filter policy changes by facet or by diff type
- Policy changes and services are paginated
- Download filtered policy changes as CSV

**Table 28: Facet filter properties**

| Property | Description           |
|----------|-----------------------|
| Priority | e.g. 100              |
| Action   | e.g. ALLOW, DENY      |
| Consumer | e.g. Consumer Cluster |
| Provider | e.g. Provider Cluster |
| Port     | e.g. 80               |
| Protocol | e.g. TCP              |

**Table 29: CSV output columns**

| Column        | Description                                                   |
|---------------|---------------------------------------------------------------|
| Rank          | The category of the policy. e.g. ABSOLUTE, DEFAULT, CATCH_ALL |
| Diff          | The diff type of the change. e.g. ADDED, REMOVED, UNCHANGED   |
| Priority      | e.g. 100                                                      |
| Action        | e.g. ALLOW, DENY                                              |
| Consumer Name | The name of the consumer cluster.                             |
| Consumer ID   | The ID of the consumer cluster.                               |
| Provider Name | The name of the provider cluster.                             |
| Provider ID   | The ID of the provider cluster.                               |
| Protocol      | e.g. TCP                                                      |

| Column | Description |
|--------|-------------|
| Port   | e.g. 80     |

In the figure below, policy versions p1 and v1 are compared.

**Figure 311: Policy Diff View**

The screenshot shows the 'Compare' section with two tabs: 'Policies' and 'Clusters'. Under 'Policies', there are two input fields: 'Base Version' containing 'p1' and 'Compare Version' containing 'v0'. Below these are statistics: 'Name: untitled', '9 log events', and 'Last Updated: Aug 5, 5:14 PM'. A search bar 'Filter Policies ...' is also visible.

Absolute No matching changes

Default Added 0 Removed 153 Unchanged 0

| Priority | Action | Consumer         | Provider                                    | Service                                                                      |
|----------|--------|------------------|---------------------------------------------|------------------------------------------------------------------------------|
| 100      | ALLOW  | bpimweb-idev4-0* | OTHER: rcdi9-dcl13n-gen-client-ace:lv120... | TCP : 5222                                                                   |
| 100      | ALLOW  | bpimweb-idev4-0* | OTHER: RTP-DC-Internal                      | UDP : 53 (DNS)<br>TCP : 80 (HTTP)<br>TCP : 111 (SunRPC)<br>TCP : 443 (HTTPS) |
| 100      | ALLOW  | bpimweb-idev4-0* | OTHER: unknown                              | UDP : 53 (DNS) ...1 more                                                     |

**Figure 312: Policy Diff View Download Button**

A button labeled 'Download Policy Changes as CSV' with a download icon (a square with a downward arrow).

**Figure 313: Filtering Policy Diff View**

The dropdown menu lists the following filterable properties:

- Priority: e.g. 100
- Action: e.g. ALLOW, DENY
- Consumer: e.g. Consumer Cluster
- Provider: e.g. Provider Cluster
- Port: e.g. 80
- Protocol: e.g. TCP

**Figure 314: Policy Diff View Diff Type Filter**

The filter shows: Default Added 15 Removed 4 Unchanged 149

**Figure 315: Policy Diff View Grouping**

The screenshot shows a single policy entry with a green background, indicating it is a change. The entry details are: Priority 100, Action ALLOW, Consumer bpimweb-idev4-0\*, Provider OTHER: RTP-DC-Internal, and Services UDP : 53 (DNS), TCP : 80 (HTTP), TCP : 111 (SunRPC), and TCP : 443 (HTTPS).

Figure 316: Policy Diff View CSV Output

| Rank    | Diff  | Priority | Action | Consumer Name    | Consumer ID              | Provider Name                            | Provider ID              | Protocol | Port |
|---------|-------|----------|--------|------------------|--------------------------|------------------------------------------|--------------------------|----------|------|
| DEFAULT | ADDED | 100      | ALLOW  | bpimweb-idev4-0* | 610bcda7a51e713db909da40 | OTHER: rcdn9-dci13n-gen-client-ace:iv120 | 610bcda7a51e713db909d9f1 | TCP      | 5222 |
| DEFAULT | ADDED | 100      | ALLOW  | bpimweb-idev4-0* | 610bcda7a51e713db909da40 | OTHER: RTP-DC-Internal                   | 610bcda7a51e713db909d9fe | UDP      | 53   |
| DEFAULT | ADDED | 100      | ALLOW  | bpimweb-idev4-0* | 610bcda7a51e713db909da40 | OTHER: RTP-DC-Internal                   | 610bcda7a51e713db909d9fe | TCP      | 80   |
| DEFAULT | ADDED | 100      | ALLOW  | bpimweb-idev4-0* | 610bcda7a51e713db909da40 | OTHER: RTP-DC-Internal                   | 610bcda7a51e713db909d9fe | TCP      | 111  |
| DEFAULT | ADDED | 100      | ALLOW  | bpimweb-idev4-0* | 610bcda7a51e713db909da40 | OTHER: RTP-DC-Internal                   | 610bcda7a51e713db909d9fe | TCP      | 443  |
| DEFAULT | ADDED | 100      | ALLOW  | bpimweb-idev4-0* | 610bcda7a51e713db909da40 | OTHER: unknown                           | 610bcda7a51e713db909da45 | UDP      | 53   |
| DEFAULT | ADDED | 100      | ALLOW  | bpimweb-idev4-0* | 610bcda7a51e713db909da40 | OTHER: unknown                           | 610bcda7a51e713db909da45 | TCP      | 443  |
| DEFAULT | ADDED | 100      | ALLOW  | bpimweb-idev3-0* | 610bcda7a51e713db909da26 | OTHER: rcdn9-dci13n-gen-client-ace:iv120 | 610bcda7a51e713db909d9f1 | TCP      | 5222 |



**Tip** See also [Comparing Versions of Generated Clusters: Diff Views, on page 494](#).

## Activity Logs and Version History

Activity logs record the history of modifications that are applied to a workspace by you. The events that are shown include adding, removing, and renaming workloads and clusters, moving workloads between clusters, uploading side information, submitting and aborting automatic policy discovery, and so on. The view shows which user has made each modification.

To view the modification history for a workspace, click any **Activity Log** link in the workspace.

For example:

1. Click **Defend > Segmentation**
2. Click the relevant scope and workspace.
3. Click the **View Activity Logs** link.
4. Click the **Workspace Activity Log** tab.

Figure 317: Log of Events Applicable to Version v1 of this Workspace

| Activity Log                                                                   | Matching Inventories <span>46</span> | Conversations | Filters <span>13</span> | Policies <span>155</span> | Provided Services | Enforcement Status | Policy Analysis | Enforcement | Compare Revisions |
|--------------------------------------------------------------------------------|--------------------------------------|---------------|-------------------------|---------------------------|-------------------|--------------------|-----------------|-------------|-------------------|
| Application Activity Log                                                       |                                      |               |                         |                           |                   |                    |                 |             | Compare Revisions |
| You stopped policy enforcement                                                 |                                      |               |                         |                           |                   |                    |                 |             | AUG 5, 5:14 PM    |
| You started policy enforcement on version p1                                   |                                      |               |                         |                           |                   |                    |                 |             | AUG 5, 4:59 PM    |
| You stopped policy enforcement                                                 |                                      |               |                         |                           |                   |                    |                 |             | AUG 5, 2:50 PM    |
| You started policy enforcement on version p1                                   |                                      |               |                         |                           |                   |                    |                 |             | AUG 5, 2:50 PM    |
| You stopped policy analysis                                                    |                                      |               |                         |                           |                   |                    |                 |             | AUG 5, 2:39 PM    |
| You started policy experiment on version p1 named s                            |                                      |               |                         |                           |                   |                    |                 |             | AUG 5, 2:39 PM    |
| You updated policy analysis to version p1                                      |                                      |               |                         |                           |                   |                    |                 |             | AUG 5, 2:38 PM    |
| You stopped policy analysis                                                    |                                      |               |                         |                           |                   |                    |                 |             | AUG 5, 2:38 PM    |
| You started policy analysis to version p1                                      |                                      |               |                         |                           |                   |                    |                 |             | AUG 5, 2:38 PM    |
| You deleted exclusion filter OTHER: RTP-DC-Internal → Default : TCP port 80    |                                      |               |                         |                           |                   |                    |                 |             | AUG 5, 2:05 PM    |
| You updated exclusion filter to Default → OTHER: RTP-DC-Internal : on any port |                                      |               |                         |                           |                   |                    |                 |             | AUG 5, 2:05 PM    |

For information about the version-related tabs and options on the page, see:

- [About Policy Versions \(v\\* and p\\*\)](#), on page 553
- [View, Compare, and Manage Discovered Policy Versions](#), on page 466



- [View, Compare, and Manage Analyzed Policy Versions, on page 536](#)
- [View, Compare, and Manage Enforced Policy Versions, on page 551](#)

## Automatic Deletion of Old Policy Versions

Every week, the following are automatically deleted: Workspace versions that have not been accessed for six months and policy experiments that have not been accessed in the last 30 days.

## Conversations

A conversation is defined as a service provided by one host on a particular port and consumed by another host. Such a conversation is materialized from many flows over different times. Automatic policy discovery takes all such flows, ignores the ephemeral/client ports, and deduplicates them to generate the conversation graph. For any given conversation between host A and host B on server (provider) port N, there has been at least one flow observation from A to B on port N in the timeframe for which automatic policy discovery has been performed.

Use flow data to better understand what flows are associated with what process while evaluating clusters generated during automatic policy discovery.

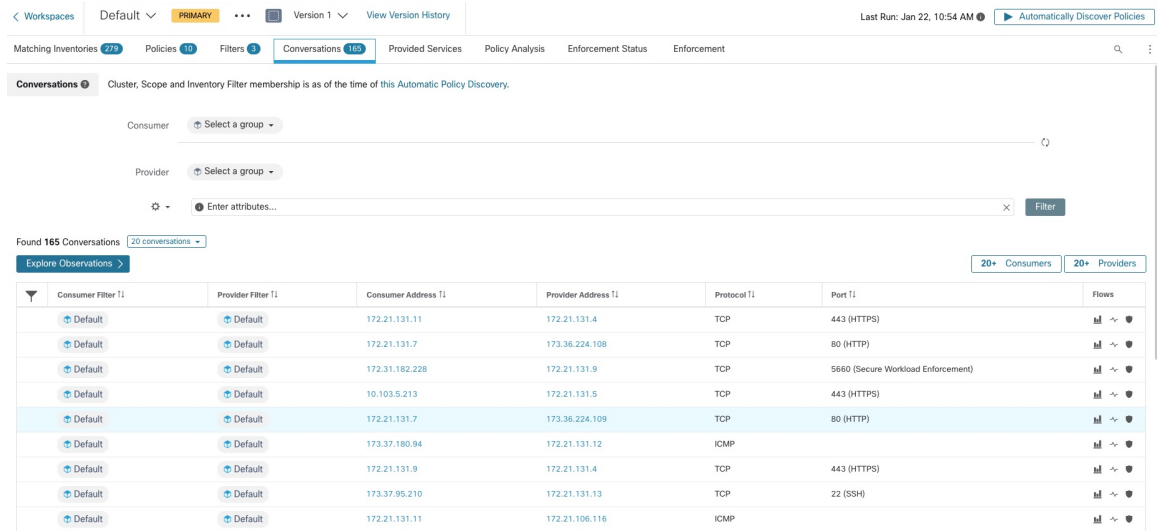
In addition, information that is collected by agents provides visibility of unused L4 ports. Unused ports are the ones for which no communication was seen for the interval selected for automatic policy discovery. This information can be used to open up policies for communication on those ports OR to close those applications binding to the unused ports, thereby reducing the attack surface of the workload.

Note that client-server classification affects the automatic policy discovery conversation view – it dictates which port is dropped (is deemed ephemeral) in the aggregation: See [Client-Server Classification](#).

## Conversations Table View

The Conversations Table view provides a simple way to view aggregated flows from the duration of automatic policy discovery where the consumer port is removed and there is only one record for all time. While policies go from filter to filter, conversations are from IP address to IP address.

Figure 318: Conversations Table View

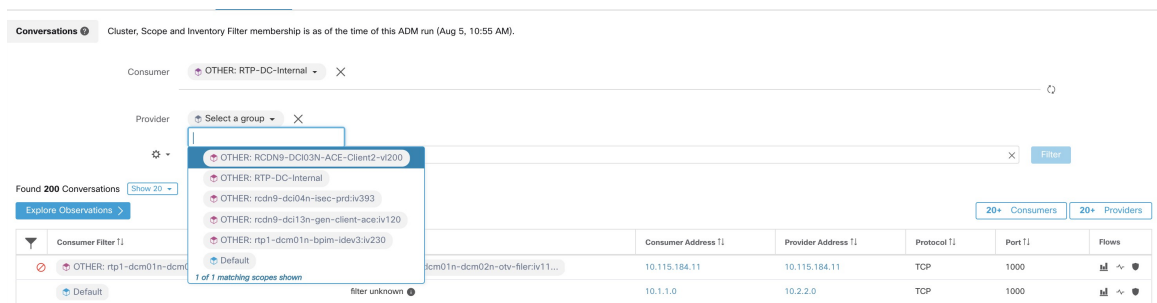


## Choosing Consumer or Provider

Consumers and Providers can be selected by a typeahead dropdown selector which allows one to choose Inventory Filters, Scopes, and Clusters as shown in the example below. All conversations between the chosen Consumer and Provider are displayed. Note: to delete an existing filter, click on the ‘x’ icon (erasing the filter may not work).

By default, the Consumer and Provider match against all of the inventory filters an IP address is a member of during automatic policy discovery. For example, searching for the “root scope” will match all the conversations even though some IPs may be better matched by more specific scopes. To perform a more specific match, select “Restrict scope filtering to an IP’s best match” from the settings dropdown to the left of the faceted filter input.

Figure 319: Choosing Consumer or Provider



## Conversation Filters

Figure 320: Conversation Filters



This is where you define filters to narrow-down the search results. All the possible dimensions can be found by clicking on the (?) icon next to the word Filters. For any User Labels data, those columns will also be

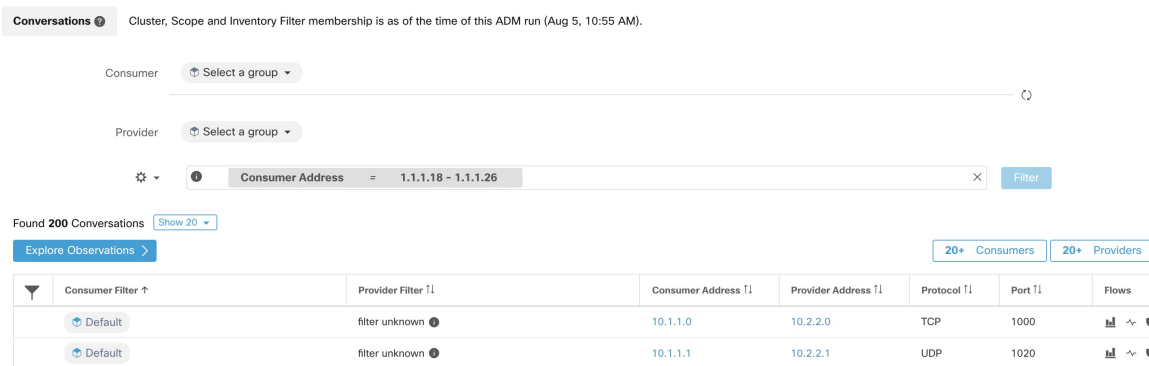
available for the appropriate intervals. This input also supports and, or, not, and parenthesis keywords, use these to express more complex filters. For example, a direction-agnostic filter between IP 1.1.1.1 and 2.2.2.2 can be written:

Consumer Address = 1.1.1.1 and Provider Address = 2.2.2.2 or Consumer Address = 2.2.2.2 and Provider Address = 1.1.1.1 And to additionally filter on Protocol = TCP:

(Consumer Address = 1.1.1.1 and Provider Address = 2.2.2.2 or Consumer Address = 2.2.2.2 and Provider Address = 1.1.1.1) and Protocol = TCP

The filter input also supports “,” and “-” for Port, Consumer Address and Provider Address, by translating “-” into range queries. The following are examples of a valid filter:

**Figure 321: Filter input Supports Range Query for Consumer Address**



Available filters:

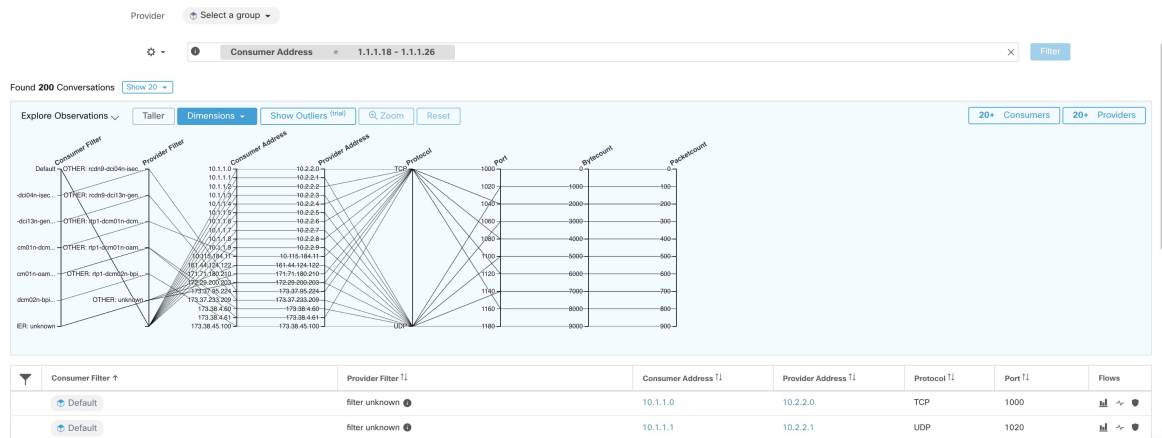
| Filters                 | Description                                                                                                                                                                                   |
|-------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Consumer Address</b> | Enter a subnet or IP Address using CIDR notation (for example, 10.11.12.0/24). Matches conversation flow observations whose consumer address overlaps with the provided IP Address or subnet. |
| <b>Provider Address</b> | Enter a subnet or IP Address using CIDR notation (for example, 10.11.12.0/24) Matches conversation flow observations whose provider address overlaps with provided IP address or subnet.      |
| <b>Port</b>             | Matches conversation flow observations whose port overlaps with provided port.                                                                                                                |
| <b>Protocol</b>         | Filter conversation flow observations by Protocol type (TCP, UDP, ICMP).                                                                                                                      |
| <b>Address Type</b>     | Filter conversation flow observations by Address type (IPv4, IPv6, DHCPv4).                                                                                                                   |
| <b>Confidence</b>       | Indicated the confidence in the direction of flow. Possible values: High, Very High, Moderate.                                                                                                |
| <b>Excluded?</b>        | Match conversations that are excluded by an exclusion filter or approved policy.                                                                                                              |

| Filters     | Description                                                                                      |
|-------------|--------------------------------------------------------------------------------------------------|
| Excluded By | Match conversations excluded by a specific filter.<br>Possible values: Exclusion Filter, Policy. |

## Explore Observations

Clicking on the Explore Observations button enables a chart view that allows quick exploration of the high-dimensional data via a “Parallel Coordinates” chart. A bit overwhelming at first, this chart can be useful when enabling only the dimensions you’re interested in (by unchecking items in the Dimensions dropdown), and when rearranging the order of the dimensions. A single line in this chart represents a single observation, and where that line intersects with the various axes indicates the value of that observation for that dimension. This can become clearer when hovering over the list of observations below the chart to see the highlighted line representing that observation in the chart:

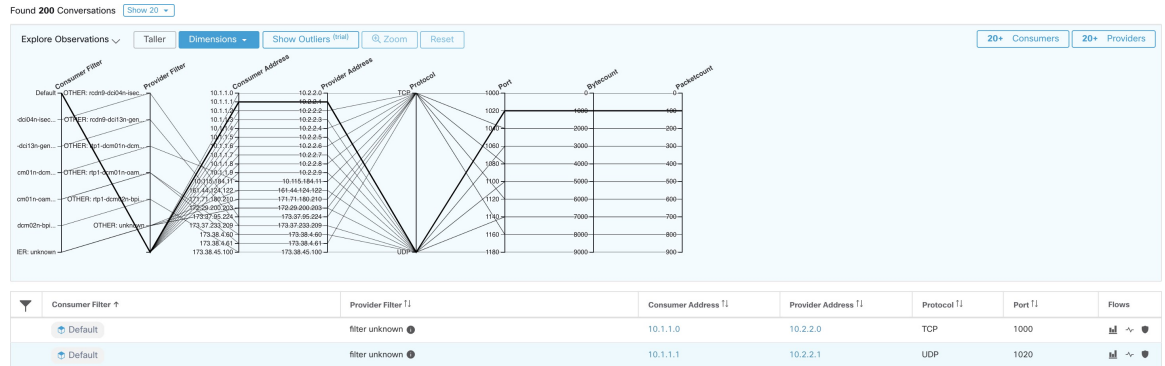
Figure 322: Explore Observations



## Conversation Observation Hovered

Due to the high-dimensional nature of the conversations data, this chart is wide by default, and requires scrolling right to see the entire chart. For this reason, it’s useful to disable all but the dimensions you are interested in. Hover state in Explore Conversations is provided to map (hover) each conversation with the table list view.

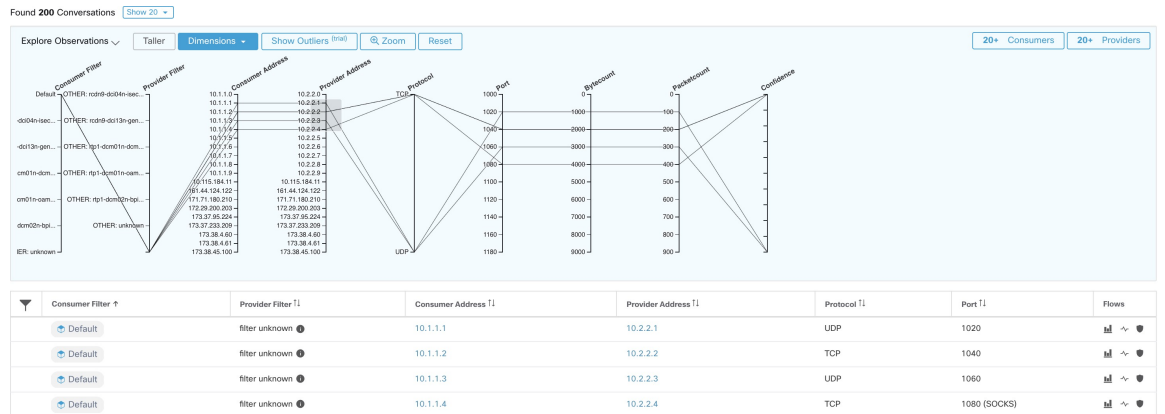
Figure 323: Conversation Observation Hovered



## Filtering

Dragging the cursor along any of the axes creates a selection that will show only observations that match that selection. Click again on the axis to remove the selection at any time. Selections can be made on any number of axes at a time. The list of observations will update to show only the selected conversations.

Figure 324: Filtering

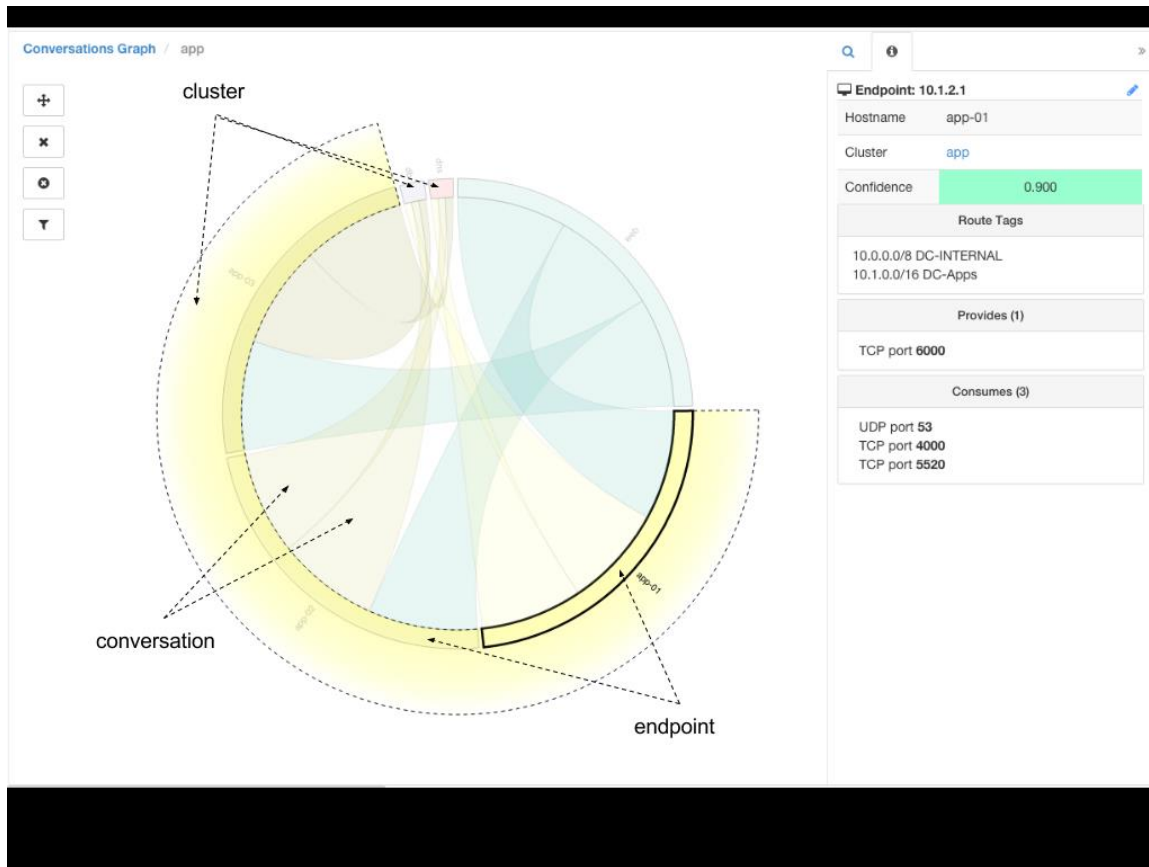


## Conversations Chart View

Conversation chart view has a similar look and feel to the policy view page, except that instead of partitions/clusters/policies, it focuses on clusters/workloads/conversations. As illustrated in the figure below, the outer arcs at the high level represent clusters and can be expanded to show the member hosts/workloads as inner arcs. The chords represent the conversations or connections.

The controls and side panel on conversation view behave similarly to the policy view, except for the fact that the side panel information also shows detailed information about selected workloads such as consumed/provided services as well as link to parent cluster and process information, if available.

Figure 325: Conversations Chart View



## Top Consumers/Providers of Conversations

The number of top Consumers or Providers based on total conversations reflecting chosen filters can be seen from two buttons on top of the Conversations table. Click on each one to see a dialog containing a table with the Conversation Count column along with each Consumer/Provider’s Address, Hostname, and other User Annotated columns.

Figure 326: Above the Conversations Table



Figure 327: Top Consumers Modal

| Top Consumers <span style="float: right;">×</span>                                |                         |            |                      |
|-----------------------------------------------------------------------------------|-------------------------|------------|----------------------|
| Showing 20 of <span style="border: 1px solid #ccc; padding: 2px;">Top 20 ▾</span> |                         |            |                      |
|                                                                                   | Hostname ↑↓             | Address ↑↓ | Conversation Count ↓ |
|                                                                                   | appServer-2             |            | 38                   |
|                                                                                   | appServer-1             |            | 37                   |
|                                                                                   | orchestrator-1          |            | 10                   |
|                                                                                   |                         |            | 8                    |
|                                                                                   | orchestrator-2          |            | 6                    |
|                                                                                   | orchestrator-3          |            | 6                    |
|                                                                                   | tsdbBosunGrafana-1      |            | 6                    |
|                                                                                   | zookeeper-2             |            | 5                    |
|                                                                                   | collectorDatamover-1    |            | 5                    |
|                                                                                   | collectorDatamover-2    |            | 5                    |
|                                                                                   | druidHistoricalBroker-2 |            | 5                    |
|                                                                                   | tsdbBosunGrafana-2      |            | 5                    |
|                                                                                   | namenode-1              |            | 5                    |
|                                                                                   | zookeeper-1             |            | 4                    |

Figure 328: Top Providers Modal

The screenshot shows a modal window titled "Top Providers" with a close button (X) in the top right corner. Below the title, it says "Showing 20 of" followed by a dropdown menu set to "Top 20". The table below has four columns: a filter icon, "Hostname ↑↓", "Address ↑↓", and "Conversation Count ↓". The table lists 20 providers with their respective hostnames, addresses, and conversation counts.

|  | Hostname ↑↓             | Address ↑↓ | Conversation Count ↓ |
|--|-------------------------|------------|----------------------|
|  | appServer-2             | 1.1.1.44   | 38                   |
|  | appServer-1             | 1.1.1.43   | 37                   |
|  | orchestrator-1          | 1.1.1.252  | 10                   |
|  |                         | 1.1.1.4    | 8                    |
|  | orchestrator-2          | 1.1.1.253  | 6                    |
|  | orchestrator-3          | 1.1.1.254  | 6                    |
|  | tsdbBosunGrafana-1      | 1.1.1.32   | 6                    |
|  | zookeeper-2             | 1.1.1.14   | 5                    |
|  | collectorDatamover-1    | 1.1.1.26   | 5                    |
|  | collectorDatamover-2    | 1.1.1.27   | 5                    |
|  | druidHistoricalBroker-2 | 1.1.1.31   | 5                    |
|  | tsdbBosunGrafana-2      | 1.1.1.33   | 5                    |
|  | namenode-1              | 1.1.1.7    | 5                    |
|  | zookeeper-1             | 1.1.1.13   | 4                    |
|  | launcherHost-1          | 1.1.1.23   | 4                    |

## Automated Load Balancer Config for Automatic Policy Discovery (F5 Only)



**Important** This is an experimental feature.

This feature and its APIs are in **ALPHA** and are subject to changes and enhancements in future releases.

Automatic policy discovery generates policies from configuration for load balancers connected to an external orchestrator. Generating policies from configuration minimizes reliance on flow data and improves the accuracy of discovered clusters and policies.

It relies on clients to report flows to the load balancer for generating policies to permit this traffic.



## Terminology

**VIP** Virtual IP: IP to which the client sends traffic that is destined for a service.

**SNIP** SNAT IP: IP used by the load balancer for sending traffic to backend hosts.

**BE** Backend Endpoint: IP of the backend host.

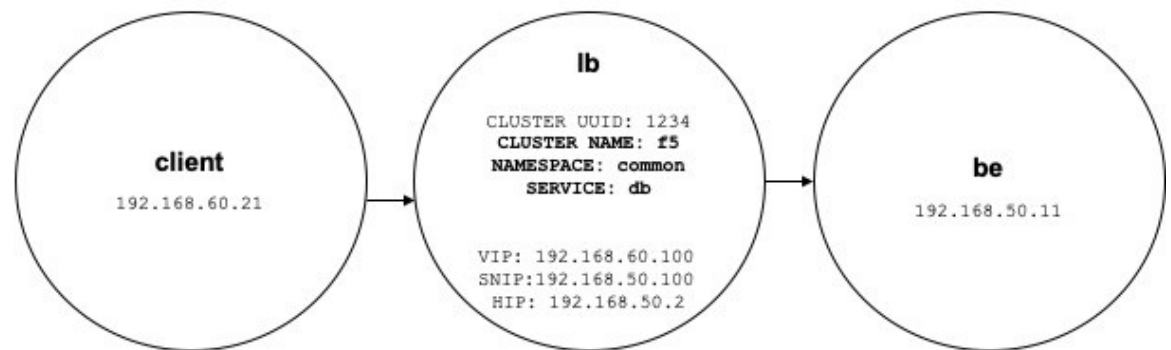
**HIP** Health-check IP: Source IP used by the load balancer for sending health-check traffic to backend hosts.



**Note** HIPs are the same as SNIPs in automap mode. However, HIPs and SNIPs can differ when a SNAT pool is configured.

## Deployment

Figure 329: Deployment



Consider the following deployment where load balancer VIPs, SNIPs, and HIPs are part of the *lb* scope, and BEs are part of the *be* scope. Scopes are created as follows.

- Client

The client scope includes clients communicating with the load balancer. For the example above, the *client* scope query is as follows:

```
address eq 192.168.60.21 or address eq 192.168.60.22
```

- lb

The F5 external orchestrator labels VIPs, SNIPs, HIPs, and BEs used by the load balancer. These labels can be used to construct scope queries, where *orchestrator\_system/service\_name* is used for selecting VIPs, *orchestrator\_system/service\_startpoint* SNIPs, and *orchestrator\_system/service\_healthcheck\_startpoint* HIPs for the service. For the example above, a scope query that includes VIPs, SNIPs, and HIPs for service *db* is as follows:

```
user_orchestrator_system/cluster_id eq 1234 and
(user_orchestrator_system/service_name eq db or
user_orchestrator_system/service_startpoint eq db or
user_orchestrator_system/service_healthcheck_startpoint eq db)
```




---

**Note** It is required that SNIPs and VIPs be part of the same scope.

---

- **Be**

`user_orchestrator_system/service_endpoint` selects BEs for a service. For the example above, a scope query that includes BEs for service `db` is as follows:

```
user_orchestrator_system/cluster_id eq 1234 and
user_orchestrator_system/service_endpoint eq db
```

## Clusters

Each service generates up to four discovered clusters, of which only the service cluster is visible to the user. SNIP, HIP, and BE clusters appear as related clusters for the service cluster. HIP and BE clusters are generated only when HIPs and BEs are present in the `lb` scope.

For the example above, automatic policy discovery generates a SNIP cluster and HIP cluster in the `lb` scope that include SNIPs and HIPs for service. Since BEs lie outside the `lb` scope, automatic policy discovery does not generate a backend cluster but instead adds the `be` scope to the list of related clusters for `db`.

Clusters are generated as follows:

- **Service**

The service cluster includes VIPs for service. The query for the service cluster as follows:

```
user_orchestrator_system/cluster_id eq 1234 and
user_orchestrator_system/namespace eq common and
user_orchestrator_system/service_name eq db
```

- **SNIP**

SNIPs for a service are included in the SNIP cluster. The query for the SNIP cluster is as follows:

```
user_orchestrator_system/cluster_id eq 1234 and
user_orchestrator_system/service_startpoint eq db
```

- **HIP**

HIPs for a service are included in the HIP cluster. The query for the HIP cluster is as follows:

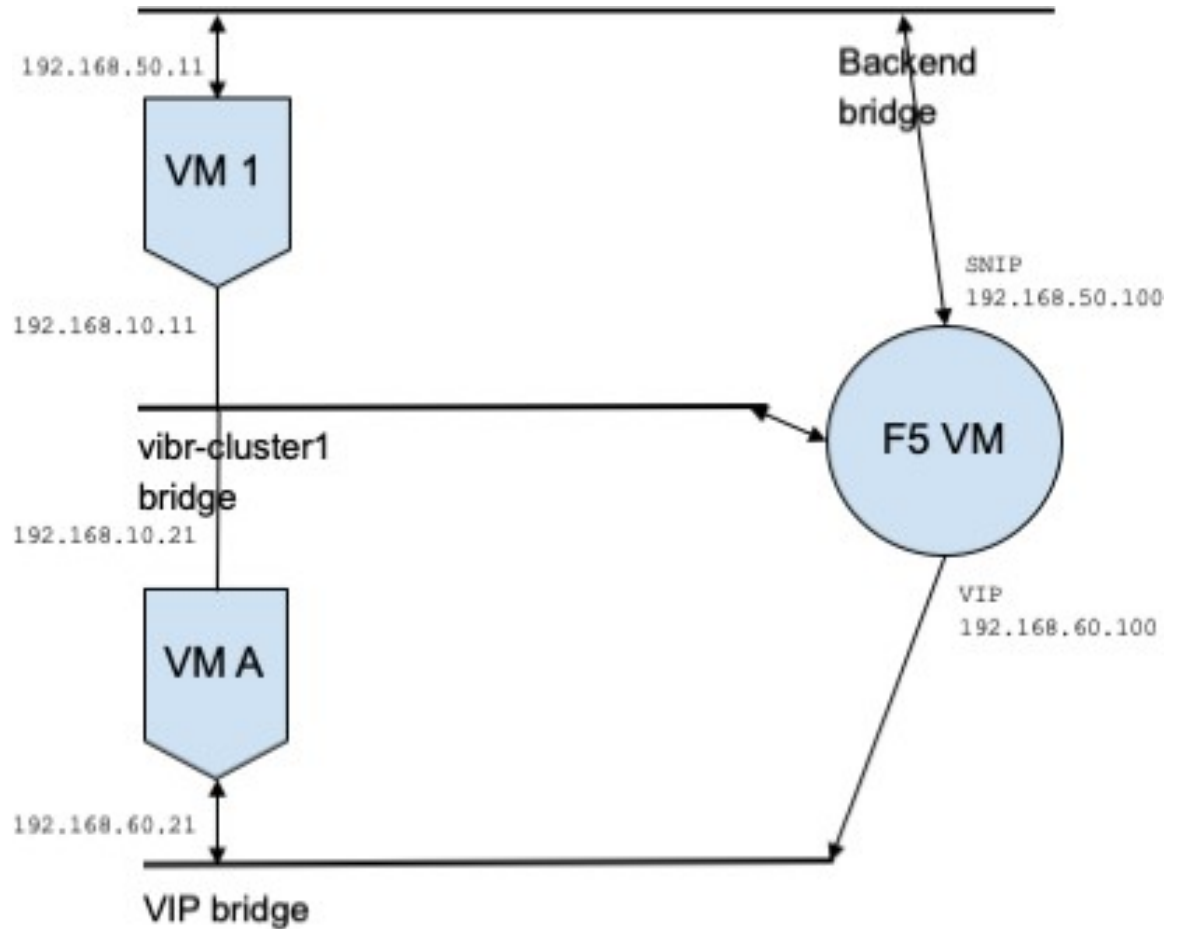
```
user_orchestrator_system/cluster_id eq 1234 and
user_orchestrator_system/service_healthcheck_startpoint eq db
```

- **Backend**

A backend cluster for the service is generated when one or more BEs are part of the `lb` scope. This doesn't apply to the example above, resulting in a backend cluster not being generated in the `lb` scope.

## Policies

Figure 330: Policy Generation



Assume we have a service *db* with VIP `192.168.60.100`, SNIP `192.168.50.100`, and a backend VM with IP `192.168.50.11` listening on port 10000. Traffic from client VM `192.168.60.21` to *db* results in the following policies:

- Policy from client to VIP

The following policy permits from the client VM to service *db*.

```
{
 "src": "<uuid of client scope>",
 "dst": "<uuid of service cluster>",
 "l4_params": [
 {
 "port": [
 10000,
 10000
],
 "proto": 6,
 }
]
}
```

- Policy from SNIP to BE.

A policy permitting traffic from the SNIP to the BE is autogenerated from configuration, and shows up as a related policy for *db*.

```
{
 "src": "<uuid of SNIP cluster>",
 "dst": "<uuid of be scope>",
 "l4_params": [
 {
 "port": [
 10000,
 10000
],
 "proto": 6,
 }
]
}
```

A policy connector from the *lb* scope to the *be* scope pushes the following policy to it.

| Consumer | Provider | Port  | Protocol | Action |
|----------|----------|-------|----------|--------|
| SNIP     | be       | 10000 | TCP      | Allow  |

This generates firewall rules on BE host 192.168.50.11 allowing incoming traffic from LB SNIP 192.168.50.100 on port 10000.

- Policy from HIP to BE.

A policy permitting traffic from the HIP to the BE is autogenerated from configuration, and shows up as a related policy for *db*.

```
{
 "src": "<uuid of HIP cluster>",
 "dst": "<uuid of be scope>",
 "l4_params": [
 {
 "port": [
 0,
 0
],
 "proto": ICMP,
 }
]
}
```

A policy connector from the *lb* scope to the *be* scope pushes the following policy to it.

| Consumer | Provider | Port | Protocol | Action |
|----------|----------|------|----------|--------|
| HIP      | be       | 0    | ICMP     | Allow  |

This generates firewall rules on BE host 192.168.50.11 allowing incoming ICMP traffic from LB HIP 192.168.50.2.

## Caveats

- When multiple services from the same load balancer instance have the same name, backend rules generated for any of these services will include backend pools for all of them, i.e. rules will be more permissive than needed.

## Policies Publisher

*Policies Publisher* is an advanced Cisco Secure Workload feature allowing third-party vendor to implement their own enforcement algorithms that are optimized for network appliances such as load balancers or firewalls. This feature is realized by publishing defined policies to a Kafka instance residing within Secure Workload cluster and by providing customers with Kafka client certificates, which allows third-party vendor code to retrieve policies from Kafka and to translate them into their network appliances configuration appropriately.

This section aims to describe the procedure third-party vendors, in short users in the following, must perform to exploit the *Policies Publisher* feature with Java on Linux.

## Prerequisites

The following software packages are installed on a Linux system, such as Ubuntu 16.04.

- Java 8 JDK
- [Apache Kafka Clients](#): kafka-clients-1.0.0.jar
- [Protocol Buffers Core](#): protobuf-java-3.4.1.jar
- [Apache Log4j](#): log4j-1.2.17.jar
- [Simple Logging Facade for Java](#): slf4j-api-1.7.25.jar, slf4j-log4j12-1.7.25.jar
- [Snappy compressor/decompressor for Java](#): snappy-java-1.1.4.jar

## Getting Kafka Client Certificates

- Create a user role with capability “*Owner*” and assign it to a user account of choice:

Figure 331: User Role Configuration to Receive Policies from Kafka

The screenshot shows the 'Role Details' configuration page for a role named 'Policies Subscription'. The page includes a 'Name' field with the value 'Policies Subscription', a 'Description' field with the placeholder text 'Enter a description (optional)', and a 'Scope' dropdown menu currently set to 'Policies Subscription'. Below these fields are two buttons: a blue 'Update' button with a checkmark icon and an orange 'Delete Role' button with a trash icon. At the bottom right, there is a blue 'Add Capability' button. Below the form is a table titled 'Capabilities' with three columns: 'Scope', 'Ability', and 'Action'. The table contains two rows, both with 'Policies Subscription' in the 'Scope' column. The first row has 'Enforce' in the 'Ability' column and a trash icon in the 'Action' column. The second row has 'Owner' in the 'Ability' column and a trash icon in the 'Action' column.

- Perform policies enforcement as described in [Enforce Policies](#). This first step is necessary as it creates a Kafka topic that is associated with active scope.
- Navigate to **Manage > Data Tap Admin**
- Select the tab “Data Taps” and download Kafka client certificates by clicking on the download button under column “Actions”. Make sure to select the *Java Keystore* format in the download dialog.

Figure 332: Data Taps View

The screenshot shows the 'Data Tap Admin - Data Taps' view. At the top right, there is a blue '+ New Data Tap' button. Below it is a table with the following columns: Name, Topic, Description, Kafka Broker, Type, Status, and Actions. The table contains three rows of data taps.

| Name                                                                           | Topic                             | Description                            | Kafka Broker                  | Type     | Status | Actions |
|--------------------------------------------------------------------------------|-----------------------------------|----------------------------------------|-------------------------------|----------|--------|---------|
| Alerts                                                                         | topic-611847e5497d4f628667761f    | DataTap Managed by Tetration           | 172.31.178.25:4... and 2 more | Internal | Active | ⬇       |
| DataExport                                                                     | DataExportTopic-611847e5497d4f628 | DataTap Managed by Tetration           | 172.31.178.25:4... and 2 more | Internal | Active | ⬇       |
| Policy Stream 676767 <span style="color: red; font-weight: bold;">ALPHA</span> | Policy-Stream-676767              | Tetration Network policy for Tenant676 | 172.31.178.25:4... and 2 more | Internal | Active | ⬇       |

- The downloaded clients certificates file usually has a name like *Policy-Stream-10-Policies-Subscription.jks.tar.gz*. Create a directory and unpack it underneath the created directory as below:

```
mkdir Policy-Stream-10-Policies-Subscription
tar -C Policy-Stream-10-Policies-Subscription -zxf
Policy-Stream-10-Policies-Subscription.jks.tar.gz
```

## Protobuf Definition File

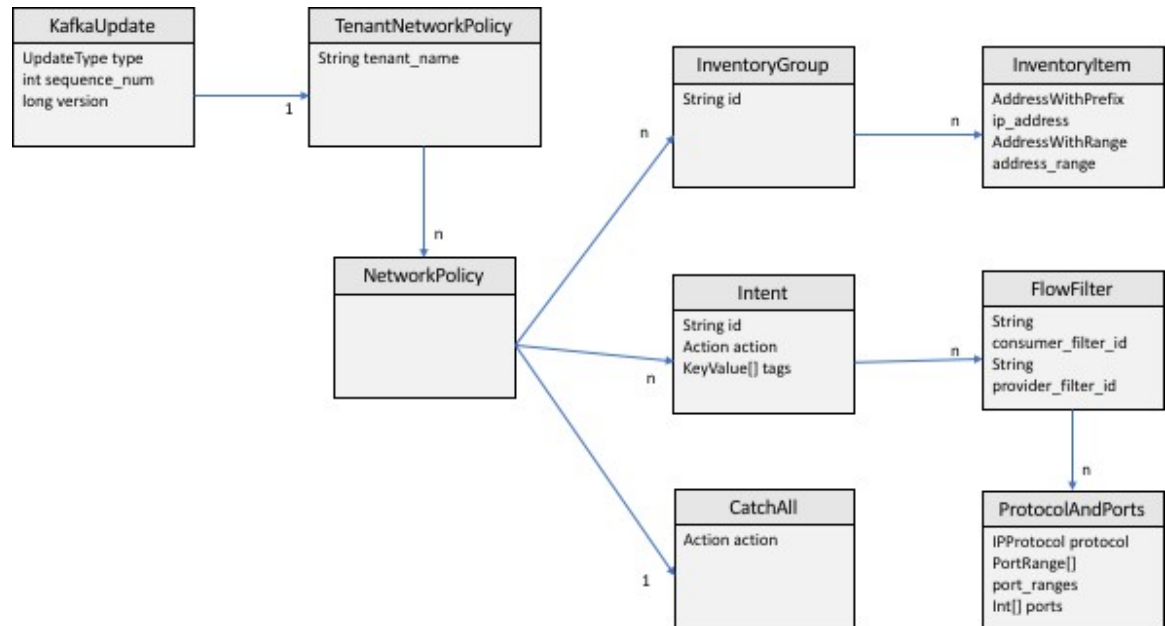
The network policies exposed by Secure Workload backend to Kafka are encoded in [Google Protocol Buffers](#) format. Refer to [this guide](#) for instructions how to download and install it on your Linux system.

The proto file of Secure Workload network policy can be downloaded from [here](#).

## Data Model of Secure Workload Network Policy

Picture below shows a simplified UML diagram of Secure Workload entities exposed to Kafka:

**Figure 333: Data Model of Secure Workload Network Policy**



A *Secure Workload Network Policy* as modeled in protobuf consists of a list of *InventoryGroups*, a list of *Intents* and a *CatchAll* policy. Each policy contains all the items belonging to one root scope. An *InventoryGroup* contains a list of *InventoryItems*, which represent Secure Workload entities such as servers or appliances by specifying their network address, be it a singular network address, subnet or address range. An *Intent* describes action (allow or deny) to be taken when a network flow matches with the given consumer's *InventoryGroup*, provider's *InventoryGroup* and network protocols and ports. The *CatchAll* represents the catch-all action that is defined for the root scope inside Secure Workload. If no workspace with enforcement enabled exists for the root scope, a default policy of *ALLOW* is written to the produced policy.

When an enforcement is triggered by the users or by a change of inventory groups, Secure Workload backend sends a full snapshot of defined network policies to Kafka as a sequence of messages that are represented as *KafkaUpdates*. Refer to *KafkaUpdate*'s comments in *tetration\_network\_policy.proto* file for details how to reconstruct those messages to a full snapshot and how to handle error conditions.

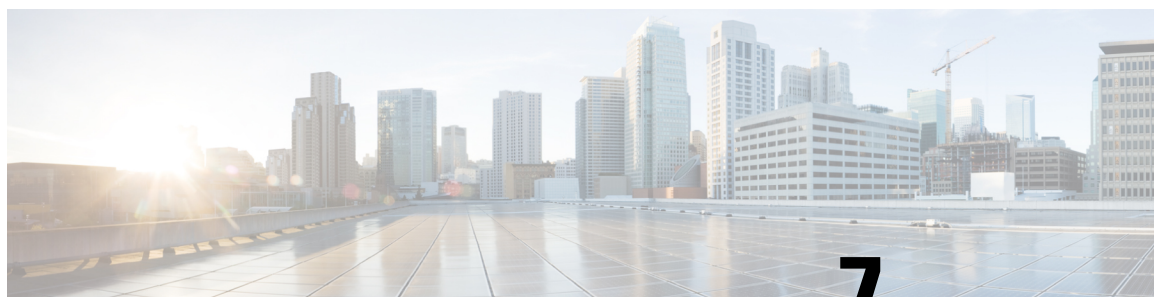
In case *KafkaUpdate* message size is greater than 10MB, Secure Workload backend splits this message into multiple fragments, each of size 10MB. If there is multiple fragments, only the first fragment has the *ScopeInfo* field of *TenantNetworkPolicy*. The *ScopeInfo* will be set to nil in the remaining fragments of *KafkaUpdate* message.

## Reference Implementation of Secure Workload Network Policies Client

For implementation and instructions on how to compile and run a demo client, see [tnp-enforcement-client](#) in Java.

This implementation provides common code to read network policies from Secure Workload policy stream via Kafka only. Vendor-specific code to program the actual policies to a network device can be plugged in by implementing the required interface [PolicyEnforcementClient](#).





## CHAPTER 7

# Configure and Monitor Forensic Events

The **Forensics** feature set enables monitoring and alerting for possible security incidents by capturing real-time forensic events and applying user-defined rules. Specifically, it enables:

- Defining of rules to specify forensic events of interest
- Defining trigger actions for matching forensic events
- Searching for specific forensic events
- Visualizing event-generating processes and their full lineages



---

**Warning** When the **Forensics** feature is enabled, the software agents may consume additional host resources depending on the agent configuration. See the Software Agent Config section.

---

- [Compatibility](#), on page 575
- [Forensics Signals](#), on page 576
- [Forensic Configuration](#), on page 581
- [Forensic visualization](#), on page 593
- [Fields Displayed in Forensic Events](#), on page 597
- [Forensic Analysis - Searchable Fields](#), on page 602
- [Search Terms in Forensic Analysis](#), on page 603
- [Forensics alerts](#), on page 609
- [Forensics Score](#), on page 611
- [PCR-Based Network Anomaly Detection](#), on page 613
- [Process Hash Anomaly Detection](#), on page 620

## Compatibility

The forensics signals are reported by the deep visibility agents on all platforms, except Solaris. Currently, only a few forensic signals are supported for AIX. For more information, see the Forensics signals section.

Forensics information is provided through Linux kernel APIs, Audit and syslog, Windows kernel APIs, Windows events, AIX audit system, and others. In general, OS vendors guarantee compatibility within a major release. However, it is possible that APIs could differ slightly across platforms and minor releases, as OS vendors may backport features and fixes. As a result, some forensic event types might not be available on

some platforms. Also, the agent does not attempt to recover or enable any OS services that are disabled when the agent starts.

For example, there are number of forensic signals that use the Linux Audit Framework. If forensics are enabled, a deep visibility agent will insert Secure Workload audit rules into the system after the agent starts. The rule insertion requires the system to have the augenrules utility that is installed and `/etc/audit/rules.d` directory. If any of these prerequisites are not satisfied, Secure Workload audit rules will not be inserted. As a result, Forensics signals including File Access and Raw Socket Creation will not be reported.

If a user has enabled forensics previously and disables it, the agent removes the audit rules that are inserted by Secure Workload. On Red Hat 7.3 and CentOS 7.3, we observed an operating system bug that may impact the rule removal process. The agent removes the audit rules by: 1. Removing the `taau.rules` in `/etc/audit/rules.d/` 2. Running `$service auditd restart`. The OS regenerates the ruleset based on the `audit.rules` and `*.rules` files in `/etc/audit/rules.d/`. Then `auditd` will load the rules into the system.

The operating system adds `-D` at the beginning of `/etc/audit/rules.d/audit.rules` file to clear all the rules before inserting the new ruleset. However, on Red Hat 7.3 and CentOS 7.3 machines the `/etc/audit/rules.d/audit.rules` may not have `-D`. This is because the OS creates an empty `/etc/audit/rules.d/audit.rules` file if this file does not exist and a default rule file in the subdirectory of `/usr/share/doc/audit-<version>/` does not exist either, for example, `/usr/share/doc/audit-2.8.4/rules/10-base-config.rules` is one possible default rule location. The exact OS behavior can be observed from the RPM update script by running `$rpm -qf -scripts /etc/audit/rules.d`

In Linux, some forensics signals rely on the observation of 64-bit system calls. 32-bit Linux system calls are not supported in the current release.

## Forensics Signals

The **Forensics** feature must be enabled for software agents to capture and report forensic events. The feature can be enabled in Software Agent Config. For more information, see the [Software Agent Config](#) section.

When the **Forensics** feature is enabled, the agent reports the following forensic events.

| Signal               | Description                                                                                                                                                       |
|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Privilege Escalation | Privilege escalations, such as commands executed with <code>sudo</code> .                                                                                         |
| User Logon           | User login events.                                                                                                                                                |
| User Logon Failed    | User login failed attempts.                                                                                                                                       |
| Shellcode            | Suspicious shell executions resembling shellcode attempts.                                                                                                        |
| File Access          | Accesses on sensitive files such as password files.                                                                                                               |
| User Account         | Adding or removing user accounts.                                                                                                                                 |
| Unseen Command       | New commands that the agent has not seen. Users can use the command anomaly score to tune results based on scope. See <a href="#">Unseen Command</a> for details. |
| Unseen Library       | New library that agent have not seen process that is loaded before.                                                                                               |

| Signal              | Description                                                                                                                                               |
|---------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------|
| Raw Socket Creation | Processes creating raw sockets. For example, port knocking.                                                                                               |
| Binary Changed      | Changes to hash values or modification times of known binaries.                                                                                           |
| Library Changed     | Changes to hash values or modification times of known libraries.                                                                                          |
| Side Channel        | Side channel attack attempts (Meltdown).                                                                                                                  |
| Follow User Logon   | Descendant processes forked or executed after the login events.                                                                                           |
| Follow Process      | Follow Process events report processes that match user forensic config rules based on process attributes such as binary path, command string, and others. |
| Network Anomaly     | Anomalies in network traffic of the workload, see <a href="#">PCR-Based Network Anomaly Detection</a> for more information.                               |

**Table 30: Forensic Signals Supported on AIX**

| Signal               | Description                                                 |
|----------------------|-------------------------------------------------------------|
| Privilege Escalation | Privilege escalations, such as commands executed with sudo. |
| Raw Socket Creation  | Processes creating raw sockets. For example, port knocking. |
| User Account         | Adding or removing user accounts.                           |

## Privilege Escalation

When the process changes its privilege from low to high, it is considered a Privilege Escalation. In Linux, this means the user-id of the process has changed from non zero to zero. There are legitimate cases such as changing the password for a normal user and other special-purpose binaries such as sudo. This event is currently not available in Windows. Privilege escalation in Windows is typically done through other mechanisms rather than changing the privilege of the process itself, i.e., integrity level. Privilege escalations on Windows are covered by other types of forensics events, such as unseen commands or binary changes.

## User Log on

User log on events including SSH, RDP, and other types of logons. Whenever available, sensors captures who, when, and how a user logs in. For example, for SSH in Linux, sensors report username, authentication type (password, public), and source IP.

## User Log on Failed

Similar to User Log on events above, sensors report failed attempts to log in with similar information whenever available.

## Shellcode

Shellcode events have different interpretations in Linux and Windows. In Linux, sensors identify processes running as interactive shell without a login session or terminal. (There are no good reasons for interactive shell running outside of a login session.) In this release, detection of shellcode events is limited in that it assumes the attack will utilize a shell already available in the system. If an attack uploads new binaries, sensors flag these binaries as either unseen commands or binary changes, if they replace existing binaries. In Windows, every process that is linked with the PowerShell DLL will be labeled as shellcode. Users can create rules to filter out legitimate cases.

## File Access

File Access events report accesses to sensitive files, such as password files. In this release, the list of files to be monitored cannot be changed by users. In Linux, the sensor monitors write access to `/etc/passwd`. Sensor also monitors read and write accesses to `/etc/shadow`. Windows will not trigger this event in this release.

## User Account

User Account events report the creation of local user accounts whenever the information is available.

## Unseen Command

Unseen Command events report commands that the sensor has not seen before. Unseen command is defined as an unseen transition/edge from a parent to a child process. For example, assuming a web server (httpd) is executing a CGI script that is called `abc.sh`, when the sensor sees it for the first time, it will report `abc.sh` as unseen command. Subsequent executions of `abc.sh` by the web server will not result in forensic events since the sensor has seen and reported it before. If a service or process never executes any binary, an unseen command event from that service/process indicates a possible compromise. Note that sensors are stateless across restarts, so a previously seen command will be reported again after a sensor restart.

Since 3.4, for SaaS clusters, each Unseen Command event is associated with a command anomaly score ranging from 0.0 to 1.0. The lower the score, the more anomalous the transition is. The command transitions, that is, the tuples (parent command line, command line), are cross-checked for anomalous transitions among those events having the same tuple below:

- The narrowest scopes that the sensor belongs to. For example, the unseen command event is observed on workload W which belongs to the following scope lineages: `Root Scope -> A -> B -> C` and `Root Scope -> D -> E`. Then, the command is cross-checked among all workloads in scopes C and E (Note that C and E can be either overlapping or nonoverlapping). The anomaly score of the event is the maximum of the anomaly scores of the event regarding those 2 scopes.
- The execution path of the running process.
- The execution path of the parent process.
- The binary hash of the running process.

A score 1.0 means the same command transition having the same tuple (narrowest scope, execution path, parent execution path, binary hash) has been seen. A score 0.0 means such command transition with such execution path, parent execution path and binary hash of the running process has never been observed on any hosts within the same scopes. The anomaly score can be used to suppress similar unseen command alerts from firing within the same scope and reduce false positives. See [Default Secure Workload Rules](#) for an example of how this score can be used.



---

**Note** The anomaly score is only available for SaaS clusters from 3.4 and later.

---

## Unseen Library

Unseen Library events report libraries that the sensor has not seen a process that is loaded before. An unseen library is defined as an unseen pair of binary execution path and library path. For example, an application usually loads a relatively stable list of libraries. An attacker who has access to the machine may restart the application and LD\_PRELOAD malicious libraries. When the sensor sees the newly loaded malicious libraries in this application binary execution path for the first time, it reports unseen library events. Subsequent load of the malicious libraries will not result in forensic events since the sensor has seen and reported it before. Legitimate cases include application loads new libraries after upgrade or applications dynamically load new libraries. Note that sensors might report a previously seen library again after restart.

Note that this is an experimental feature and is subject to change in future releases.

## Raw Socket Creation

Raw Socket Creation events are only supported on Linux in this release. Raw sockets are typically used to snoop or inject/spoof traffic. There are legitimate uses of raw sockets, such as in diagnosis tools like tcpdump, or when crafting special IP packets like ping or arp. Malicious uses include stealth scans to avoid logging by target/victim machines, malware port knocking, and so on. Secure Workload sensors also create raw sockets for collecting flow-related information. (For consistency, sensors do not suppress events that are triggered by their own flow information collection.)

## Binary Changed

Binary Changed events report changes to the file contents and attributes of binaries for running processes. Sensors record the file attributes of every running process. If a process runs a binary at the same path, but with different file attributes (ctime, mtime, size, or hash), the sensor flags the process as a binary change. Legitimate cases include application upgrade.

## Library Changed

Library Changed events report changes to the file contents and attributes of libraries for running processes. Sensors record the file attributes of loaded libraries. If a process loads a library at the same path, but with different file attributes (ctime, mtime, size, or hash), the sensor will flag the process with a library change. Legitimate cases include library upgrade.

Note that this is an experimental feature and is subject to change in future releases.

## Side Channel

Side Channel events report running software that exploits side channel vulnerabilities. This release provides one side channel detection capability on selected Linux platform: Meltdown. See the details below for supported machine configurations. These are advanced security features and therefore disabled by default. Users should expect to see increased CPU usage when side channel reporting is enabled. The CPU quota that is configured in the UI will still be honored. If the forensic collection subprocess of the sensor determines that its CPU usage is too high for too long, it shuts down, and the parent sensor process will restart it with a small delay. Enabling this feature on old or unsupported kernels could lead to system instability. Testing in similar nonproduction environments is recommended.

This feature can be turned on/off from the agent config page in the UI and they can be turned on/off in each agent config profile.

Meltdown is a side channel attack that abuses the speculative execution and cache features in the CPU (<https://meltdownattack.com/>). It allows an attacker to read privileged-domain data from an unprivileged domain, for example, reading kernel memory from a user space application without ring 0 privileges. Meltdown detection currently supports CentOS 7 and Ubuntu 16.04.

## Follow User Logon

Follow User Logon events report descendant processes (up to 4 levels) that are executed after a User Logon event process (SSH, RDP, and so on.). Processes reported under this Follow User Logon event are for auditing purposes and not necessary having any security events.

## Follow Process

Follow Process events report processes that match user forensic config rules based on process attributes such as binary path, command string, and so on. Processes that are reported under this Follow Process event are for auditing purposes and not necessary having any security events.

Example 1: Report processes that are run by cmd.exe or powershell.exe

Event Type = Follow Process AND (Process Info - Exec Path contains cmd.exe OR Process Info - Exec Path contains powershell.exe)

Example 2: Report any processes which are created by winword.exe or excel.exe or powerpnt.exe.

Event Type = Follow Process with\_ancestor (Process Info - Exec Path contains winword.exe OR Process Info - Exec Path contains excel.exe OR Process Info - Exec Path contains powerpnt.exe)

Note: Follow Process events can be tracked by one of the following process signals:

- Process Info - Exec Path
- Process Info - Command String
- Process Info - Username
- Follow Process - Parent Exec Path
- Follow Process - Parent Command String
- Follow Process - Parent Username

# Forensic Configuration

Forensics feature uses intent-based configuration. Intents specify how to apply forensic profiles to inventory filters. Forensic profile consists of multiple forensic rules. Profiles in an intent are applied in order from top to bottom.

## Forensic Rules



**Note** The maximum number of rules per root scope is 100.

## Adding a Forensic Rule

This section explains how to add new forensic rules.

### Before You Begin

You must login as **Site Admin**, **Customer Support** or **Scope Owner** in the system.

### Procedure

- Step 1** In the navigation bar on the left, click **Defend > Forensic Rules**.
- Step 2** Click **Create Rule**.
- Step 3** Enter the appropriate values in the following fields.

| Field                  | Description                                                                                                                                                                                                           |
|------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Rule Name</b>       | Enter a name for the rule. Name cannot be empty.                                                                                                                                                                      |
| <b>Ownership scope</b> | Enter an ownership scope for this rule.                                                                                                                                                                               |
| <b>Actions</b>         | Select actions when this rule is triggered. <b>Record</b> means matching security events persist for further analysis. <b>Alert</b> action means to publish matching security events to Secure Workload Alert system. |
| <b>Severity</b>        | Select severity level of this rule: <b>LOW</b> , <b>MEDIUM</b> , <b>HIGH</b> , <b>CRITICAL</b> or <b>REQUIRES IMMEDIATE ACTION</b>                                                                                    |
| <b>Clause</b>          | Enter a rule clause. A clause must contain security event signals from either a process forensic event or a workload event. A clause is invalid if it contains both process and workload signals.                     |

Figure 334: Create rule

**Step 4** Click **Save**.

## Basic Forensic Rule Composition

A forensic rule must contain **exactly one** forensic event type (for example, **Event Type == Unseen Command**). The following optional clauses uses attributes of that event (for example, **Unseen Command - Parent Uptime**).

Below is an example of using **Unseen Command** event type. For more examples, see the default rules and MITRE rules.

**EventType = Unseen Command and Unseen Command - Parent Uptime (microseconds) >= 60000000.**

## Default Secure Workload Rules

Default Secure Workload rules are provided to help the users to construct rules that are meaningful in their environment. These rules are displayed in the forensic config page and they are not editable. The rules are available in all root scopes.

Figure 335: Default Rules

|                         |         |                                                                         |               |      |  |
|-------------------------|---------|-------------------------------------------------------------------------|---------------|------|--|
| Tetration - Privileg... | Default | A pre-defined rule that alerts and records Privilege Escalation events. | ALERT, RECORD | HIGH |  |
| Tetration - Raw Sock... | Default | A pre-defined rule that alerts and records Raw Socket Creation events.  | ALERT, RECORD | HIGH |  |
| Tetration - Unseen C... | Default | A pre-defined rule that alerts and records Unseen Command events.       | ALERT, RECORD | LOW  |  |

The Secure Workload forensic rules:

### 1. Name Secure Workload - Privilege Escalation

**Clause** **EventType = Privilege Escalation and ( ProcessInfo - ExecPath doesn't contain sudo and ProcessInfo - ExecPath doesn't contain ping and Privilege Escalation Is≠ Type - Suid Binary)**

**Description.** This rule reports privilege escalation events that are not generated by setuid binaries. To reliably filter out the setuid binaries, it also filters out **sudo** and **ping** based on “ProcessInfo - ExecPath”. Secure Workload users can also filter out other setuid binaries by defining their own rules.

### 2. Name Tetration - Unseen Command



**Clause EventType = Unseen Command and Unseen Command - Parent Uptime (microseconds) >= 60000000 or ProcessInfo - ExecPath contains /bash or ProcessInfo - ExecPath contains /sh or ProcessInfo - ExecPath contains /ksh or Parent - ExecPath contains httpd or Parent - ExecPath contains apache or Parent - ExecPath contains nginx or Parent - ExecPath contains haproxy**

**Description.** This rule reports unseen command events that match one of the following criteria:

- a. Process parent is alive for more than **60,000,000** microseconds.
- b. Process ExecPath contains some type of shell, for example, **/bash**, **/sh**, and **/ksh**.
- c. Process parent ExecPath contains some type of server application, for example, **httpd**, **apache**, **nginx**, and **haproxy**.

3. **Name** Tetration - Raw Socket

**Clause EventType = Raw Socket Creation and (Raw Socket - ExecPath doesn't contain ping and Raw Socket - ExecPath doesn't contain iptables and Raw Socket - ExecPath doesn't contain xtables-multi)**

**Description** This rule reports raw socket creation events that are not generated by **ping** and **iptables**. Secure Workload users can also filter out other binaries by defining their own rules.

4. **Name** Tetration - Network Anomaly with Unseen Command

**Clause EventType = Network Anomaly and Network Anomaly - Unseen Command Count > 3 and Network Anomaly - Non-seasonal Deviation > 0**

**Description** This rule reports network anomaly events that match the following criteria:

- a. There are more than 3 Unseen Command events on the same workload within 15 minutes.
- b. The [Rule Attributes](#) is greater than 0 (which also means it is greater than or equal to 6.0 because 6.0 is the minimum reported deviation for all network anomaly events).

5. **Name** Tetration - Anomalous Unseen Command

**Clause EventType = Unseen Command and Unseen Command - Anomaly - Score < 0.6**

**Description** This rule reports unseen command events whose anomaly score is less than 0.6. This means that only highly anomalous events whose commands do not look similar to previously observed commands are reported. The threshold 0.6 is decided based on Secure Workload's experiments on how similar commands are at different thresholds. See [Unseen Command](#) for a detailed explanation of the score.

6. **Name** Tetration - Unusual Parent of smss

**Clause EventType = Follow Process and ProcessInfo - ExecPath contains smss.exe and (Follow Process - ParentExecPath doesn't contain smss.exe and Follow Process - ParentExecPath doesn't contain System)**

**Description** This rule is specific for windows. This rule alerts if smss.exe has a parent that is different from another instance of smss.exe or the System process.

7. **Name** Tetration - Unusual Parent of wininit

**Clause EventType = Follow Process and ProcessInfo - ExecPath contains wininit.exe and Follow Process - ParentExecPath doesn't contain smss.exe**

**Description** This rule is specific for windows. This rule alerts if wininit.exe has a parent that is different from smss.exe.

8. **Name** Tetration - Unusual Parent of RuntimeBroker

**Clause** **EventType = Follow Process and ProcessInfo - ExecPath contains RuntimeBroker.exe and Follow Process - ParentExecPath doesn't contain svchost.exe**

**Description** This rule is specific for windows. This rule alerts if RuntimeBroker.exe has a parent that is different from svchost.exe.

9. **Name** Tetration - Unusual Parent of services

**Clause** **EventType = Follow Process and ProcessInfo - ExecPath contains services.exe and Follow Process - ParentExecPath doesn't contain wininit.exe**

**Description** This rule is specific for windows. This rule alerts if services.exe has a parent that is different from wininit.exe.

10. **Name** Tetration - Unusual Parent of lsaid

**Clause** **EventType = Follow Process and ProcessInfo - ExecPath contains lsaid.exe and Follow Process - ParentExecPath doesn't contain wininit.exe**

**Description** This rule is specific for windows. This rule alerts if lsaid.exe has a parent that is different from wininit.exe.

11. **Name** Tetration - Unusual Child of lsass

**Clause** ( **EventType = Follow Process and ProcessInfo - ExecPath doesn't contain efsui.exe and ProcessInfo - ExecPath doesn't contain werfault.exe** ) **with ancestor Process Info - ExecPath contains lsass.exe**

**Description** This rule is specific for windows. This rule alerts if lsass.exe has any descendants that are not efsui.exe or werfault.exe.

## Default MITRE ATT&CK Rules

Default MITRE ATT&CK rules are provided to alert techniques from the MITRE ATT&CK Framework (<https://attack.mitre.org/>). There are 24 rules pertaining to adversarial behaviour and most of them are mapped to a particular MITRE technique. The complete list of the rules is below.

1. **Name** Suspicious MS Office behavior

**Clause** ( **Event type = Follow Process and (Process Info - Exec Path doesn't contain Windowssplwow64.exe ) and (Process Info - Exec Path doesn't contain chrome.exe ) and (Process Info - Exec Path doesn't contain msip.executionhost.exe ) and (Process Info - Exec Path doesn't contain msip.executionhost32.exe ) and (Process Info - Exec Path doesn't contain msosync.exe ) and (Process Info - Exec Path doesn't contain ofcccaupdate.exe ) with ancestor (Process Info - Exec Path contains winword.exe or Process Info - Exec Path contains excel.exe or Process Info - Exec Path contains powerpnt.exe )** )

**Description** This rule alerts and records if Microsoft Office processes (WIN-WORD.exe/EXCEL.exe/POWERPNT.exe) create any child processes. Based on our research we have allowed a few common child processes known to be created by these MS Office binaries, to reduce the number of false positives.

2. **Name** T1015 - Accessibility features 1

**Clause Event type = Follow Process (Process Info - Exec Path contains cmd.exe or Process Info - Exec Path contains powershell.exe or Process Info - Exec Path contains cscript.exe or Process Info - Exec Path contains wscript.exe) and (Follow Process - Parent Exec Path contains winlogon.exe or Follow Process - Parent Exec Path contains atbroker.exe or Follow Process - Parent Exec Path contains utilman.exe)**

**Description** This rule alerts and records if any of the Accessibility features binaries (On-screen Keyboard, Magnifier, Sticky keys, and so on.) are abused and are tricked into opening cmd/powershell/cscript/wscript. The invocation of accessibility binaries is controlled by either winlogon, atbroker or utilman processes depending on from where they are invoked (from the logon screen or after a user logs in). This rule captures suspicious child processes (cmd.exe, powershell.exe, cscript.exe, wscript.exe) of the accessibility processes (winlogon.exe, utilman.exe, and atbroker.exe). Use this with **T1015 - Accessibility features 2** to also catch the additional child processes of these four suspicious child processes\*\*.

3. **Name** T1015 - Accessibility features 2

**Clause Event type = Follow Process with ancestor (( Process Info - Exec Path contains cmd.exe or Process Info - Exec Path contains powershell.exe or Process Info - Exec Path contains cscript.exe or Process Info - Exec Path contains wscript.exe) and (Follow Process - Parent Exec Path contains winlogon.exe or Follow Process - Parent Exec Path contains atbroker.exe or Follow Process - Parent Exec Path contains utilman.exe))**

**Description** This rule alerts and records if any of the Accessibility features binaries (On-screen Keyboard, Magnifier, Sticky keys, and so on.) are abused and are tricked into opening cmd.exe/powershell.exe/cscript.exe/wscript.exe. The invocation of accessibility binaries is controlled by either winlogon, atbroker or utilman processes depending on from where they are invoked (from the login screen or after a user logs in). This rule captures child processes of the suspicious child processes of these processes (winlogon, utilman, and atbroker). One should use this with **T1015 - Accessibility features 1** which alerts the suspicious child processes of accessibility binaries.

4. **Name** T1085 - rundll32

**Clause (Event type = Follow Process and Process Info Exec Path does not contain msixec.exe and Process Info Exec Path does not contain WindowsSystem32SystemPropertiesRemote.exe with ancestor (Process Info - Exec Path contains rundll32.exe and Follow Process - Parent Exec Path does not contain msixec.exe and not ( Process Info -command string contains Windowssystem32shell32.dll or ( Process Info -command string contains Windowssystem32wow64shell32.dll or ( Process Info -command string contains WindowsSystem32migrationWinInetPlugin.dll ))**

**Description** This rule alerts and records if rundll32.exe creates child processes. This binary can be called to execute arbitrary binary/dll or used by control.exe to install malicious control panel items. However, we have allowed if msixec.exe is either the parent or the descendant of rundll32.exe. We have also permitted some of the common rundll32 commands that make use of well-known dlls.

5. **Name** T1118 - InstallUtil

**Clause Event type = Follow Process with ancestor Process Info - Exec Path contains installutil.exe**

**Description** This rule alerts and records if InstallUtil.exe creates child processes.

6. **Name** T1121 - Regsvcs/Regasm

**Clause Event type = Follow Process and ( Process Info - Exec path does not contain fondue.exe or Process Info - Exec path does not contain regasm.exe or Process Info - Exec path does not contain**

**regsvr32.exe with ancestor (Process Info - Exec Path contains regasm.exe or Process Info - Exec Path contains regsvcs.exe)**

**Description** This rule alerts and records if regsvcs.exe or regasm.exe create child processes. However, we have permitted if fondue.exe/regasm.exe/regsvr32.exe is spawned by regasm.exe or regsvcs.exe to reduce the number of false positives.

7. **Name** T1127 - Trusted Developer Utilities - msbuild.exe

**Clause** ( Event type = Unseen Command with ancestor Process Info - Exec Path contains MSBuild.exe ) and ( Process Info - Exec Path does not contain Tracker.exe ) and ( Process Info - Exec Path doesn't contain csc.exe ) and ( Process Info - Exec Path does not contain Microsoft Visual Studio ) and ( Process Info - Exec Path does not contain al.exe ) and ( Process Info - Exec Path does not contain lc.exe ) and ( Process Info - Exec Path does not contain dotnet.exe ) and ( Process Info - Exec Path does not contain cvtres.exe ) and ( Process Info - Exec Path does not contain conhost.exe ) and not ( Event type = Unseen Command with ancestor ( Process Info - Exec Path contains Tracker.exe or Process Info - Exec Path contains csc.exe or Process Info - Exec Path contains Microsoft Visual Studio or Process Info - Exec Path contains al.exe or Process Info - Exec Path contains lc.exe or Process Info - Exec Path contains dotnet.exe or Process Info - Exec Path contains cvtres.exe ) )

**Description** This rule alerts and records if msbuild.exe creates child processes which do not belong to an allowlist of child processes it usually creates. This rule is currently Unseen Command based, as opposed to Follow Process, since Follow Process does not yet support allowing process subtrees. The current rule allows the following processes and their descendants: Tracker.exe, csc.exe, any process from “Microsoft Visual Studio” path, al.exe, lc.exe, dotnet.exe and cvtres.exe. The rule also allows conhost.exe. These processes can be seen during regular usage of MSBuild.exe (for example, compiling a project via Visual Studio). All the other descendants (not usual behavior) of MSBuild.exe are alerted.

8. **Name** T1127 - Trusted Developer Utilities - rcsi.exe

**Clause** Event type = Follow Process with ancestor Process Info - Exec Path contains rcsi.exe

**Description** This rule alerts and records if rcsi.exe creates child processes.

9. **Name** T1127 - Trusted Developer Utilities - tracker.exe

**Clause** (Event type = Unseen Command with\_ancestor Process Info - Exec Path contains tracker.exe) and not (Event type = Unseen Command with\_ancestor Process Info - Exec Path contains MSBuild.exe)

**Description** This rule alerts and records if tracker.exe creates child processes and tracker itself is not a descendant of MSBuild.exe. Thus legitimate invocations of tracker via Visual Studio are approved, but other invocations are alerted. One limitation with the Tracker.exe and the previous MSBuild.exe rules is that if an attacker uses the MSBuild technique to create Tracker, and then make Tracker create a malicious child, it would not be alerted by either of the rules since Tracker having MSBuild as an ancestor is considered legitimate.

10. **Name** T1128 - Netsh Helper DLL

**Clause** Event type = Follow Process with ancestor Process Info - Exec Path contains netsh.exe

**Description** This rule alerts and records if netsh.exe creates child processes.

11. **Name** T1136 - Create Account

**Clause** Event type = User Account

**Description** This rule alerts and records if a new user is created.

- 12. Name** T1138 - Application Shimming  
**Clause Event type = Follow Process Info - Exec Path** *contains* **sdbinst.exe**  
**Description** This rule alerts and records if sdbinst.exe is invoked.
- 13. Name** T1180 - Screensaver  
**Clause Event type = Follow Process AND with ancestor Process Info - Exec Path** *contains* **.scr**  
**Description** This rule alerts and records if a process is created with “.scr” in the exec path.
- 14. Name** T1191 - CMSTP  
**Clause Event type = Follow Process with ancestor Process Info - Exec Path** *contains* **cmstp.exe**  
**Description** This rule alerts and records if cmstp.exe creates child processes.
- 15. Name** T1202 - Indirect Command Execution - forfiles.exe  
**Clause Event type = Follow Process with ancestor Process Info - Exec Path** *contains* **forfiles.exe**  
**Description** This rule alerts and records if forfiles.exe creates child processes.
- 16. Name** T1202 - Indirect Command Execution - pcalua.exe  
**Clause Event type = Follow Process with ancestor Process Info - Exec Path** *contains* **pcalua.exe**  
**Description** This rule alerts and records if pcalua.exe creates child processes.
- 17. Name** T1216 - Signed Script Proxy Execution - pubprn.vbs  
**Clause Event type = Follow Process with ancestor (( Process Info - Exec Path** *contains* **cscript.exe** **or Process Info - Exec Path** *contains* **wscript.exe)** **and Process Info - Command String** *contains* **.vbs** **and Process Info - Command String** *contains* **script** )  
**Description** This rule alerts and records if any vbs script is run using wscript.exe or cscript.exe, to create a new process, with a parameter “script”. This technique could be used by an attacker to execute pubprn.vbs with a script parameter pointing to a malicious set file which then gives code execution.
- 18. Name** T1218 - Signed Binary Proxy Execution - msixexec.exe  
**Clause Event type = Follow Process with ancestor Process Info - Exec Path** *contains* **msixexec.exe**  
**Description** This rule alerts and records if msixexec.exe creates child processes.
- 19. Name** T1218 - Signed Binary Proxy Execution - odbconf.exe  
**Clause Event type = Follow Process with ancestor Process Info - Exec Path** *contains* **odbconf.exe**  
**Description** This rule alerts and records if odbconf.exe creates child processes.
- 20. Name** T1218 - Signed Binary Proxy Execution - Register-CimProvider  
**Clause Event type = Follow Process with ancestor Process Info - Exec Path** *contains* **Register-CimProvider.exe**  
**Description** This rule alerts and records if Register-CimProvider.exe creates child processes.
- 21. Name** T1220 - XSL Script Processing - msxsl.exe  
**Clause Event type = Follow Process with ancestor Process Info - Exec Path** *contains* **msxsl.exe**  
**Description** This rule alerts and records if msxsl.exe creates child processes.

22. **Name** T1220 - XSL Script Processing - wmic  
**Clause Event type = Follow Process and (Process Info - Exec Path contains wmic.exe and Process Info - Command String contains .xsl)**  
**Description** This rule alerts and records if an xsl script is used by wmic. This can be used to launch arbitrary binaries.
23. **Name** T1223 - Compiled HTML Files  
**Clause Event type = Follow Process with ancestor Process Info - Exec Path contains hh.exe**  
**Description** This rule alerts and records if hh.exe creates child processes.
24. **Name** T1003 - Credential Dumping - Lsass  
**Clause Event type = Follow Process and Process Info - Exec Path contains procdump.exe and Process Info - Command String contains lsass**  
**Description** This rule alerts and records if procdump.exe is used to dump the memory of lsass processes.
25. **Name** T1140 - Deobfuscate/Decode Files or Information  
**Clause Event type = Follow Process and Process Info - Exec Path contains certutil.exe and (Process Info - Command String matches .\*encode\s.\* or Process Info - Command String matches .\*decode\s.\***  
**Description** This rule alerts and records if certutil.exe is used to either encode or decode a file. This technique is often used by attackers to decode their encoded payload on the victim machine.
26. **Name** T1076 - Remote Desktop Protocol  
**Clause Event type = Follow Process and Process Info - Exec Path contains tscon.exe**  
**Description** This rule alerts and records if tscon.exe is executed. Attackers can use tscon.exe to hijack existing RDP sessions.
27. **Name** T1197 - BITS Jobs - Powershell  
**Clause Event type = Follow Process and Process Info - Exec Path contains powershell.exe and Process Info - Command String contains Start-BitsTransfer**  
**Description** This rule alerts and records if the powershell.exe is used to run the cmdlet Start-BitsTransfer to copy/move files.
28. **Name** T1170 - MSHTA  
**Clause Event type = Follow Process with ancestor Process Info - Exec Path contains mshta.exe**  
**Description** This rule alerts and records if mshta.exe is used to run malicious HTA scripts that spawn child processes.
29. **Name** T1158 - Hidden Files and Directories  
**Clause Event type = Follow Process and (Process Info - Exec Path contains attrib.exe and Process Info - Command String contains +h)**  
**Description** This rule alerts and records if attrib.exe is used to set a file/directory as hidden.
30. **Name** T1114 - Email Collection  
**Clause Event type = Follow Process (Process Info - Command String matches .\*(ost|pst)(\s|'|').\* or Process Info - Command String matches .\*(ost|pst)\$ ) Process Info - Exec Path doesn't contain outlook.exe**

**Description** This rule alerts and records if email files (.ost and .pst) are accessed from any other process other than outlook.exe.

31. **Name** T1070 - Indicator Removal on Host - Event Log

**Clause Event type = Follow Process and Process Info - Exec Path contains wevtutil.exe and Process Info - Command String matches .\*\s(c|clear-log)\s.\***

**Description** This rule alerts and records if wevtutil.exe is used to clear event logs.

32. **Name** T1070 - Indicator Removal on Host - USN

**Clause Event type = Follow Process and Process Info - Exec Path contains fsutil.exe and Process Info - Command String matches .\*\susn\s.\* and Process Info - Command String matches .\*\sdeletejournal.\***

**Description** This rule alerts and records if fsutil.exe is used to delete USN journals.

33. **Name** T1053 - Scheduled Task

**Clause Event type = Follow Process and Process Info - Exec Path contains schtasks.exe and Process Info - Command String contains create**

**Description** This rule alerts and records if schtasks.exe is used to create new scheduled tasks.

34. **Name** T1003 - Credential Dumping - Vaultcmd

**Clause Event type = Follow Process and Process Info - Exec Path contains vaultcmd.exe and Process Info - Command String matches .\*\list.\***

**Description** This rule alerts and records if vaultcmd.exe is used access Windows Credentials vault.

35. **Name** T1003 - Credential Dumping - Registry

**Clause Event type = Follow Process and Process Info - Exec Path contains reg.exe and ((Process Info - Command String contains save or Process Info - Command String contains export) and (Process Info - Command String contains hklm or Process Info - Command String contains hkey\_local\_machine) and (Process Info - Command String contains sam or Process Info - Command String contains security or Process Info - Command String contains system))**

**Description** This rule alerts and records if reg.exe is used dump certain registry hives.

36. **Name** T1201 - Password Policy Discovery 1

**Clause Event type = Follow Process and Process Info - Exec Path contains change and Process Info - Command String contains -l**

**Description** This rule alerts and records if change utility is used to list the password policy (password age policy) on a linux machine.

37. **Name** T1081 - Credentials in Files - Linux

**Clause Event type = Follow Process and (Process Info - Exec Path contains cat or Process Info - Exec Path contains grep) and (Process Info - Command String contains .bash\_history or Process Info - Command String contains .password or Process Info - Command String contains .passwd)**

**Description** This rule alerts and records if attempts are made to search for passwords stored in files on a linux machine.

38. **Name** T1081 - Credentials in Files - Windows

**Clause Event type = Follow Process and Process Info - Exec Path contains findstr.exe and Process Info - Command String contains password**

**Description** This rule alerts and records if attempts are made to search for passwords stored in files on a windows machine.

**39. Name** T1089 - Disabling Security Tools

**Clause Event type = Follow Process and ( (Process Info - Exec Path contains fltmc.exe and Process Info - Command String contains unload sysmon) or (Process Info - Exec Path contains sysmon.exe and Process Info - Command String contains lu) )**

**Description** This rule alerts and records if attempts are made to unload sysmon driver using fltmc.exe or sysmon.exe

## Forensic profiles

### Add a Profile

This section explains how to add new forensic profiles.

Before You Begin

You must log in as **Site Admin**, **Customer Support** or **Scope Owner** in the system.

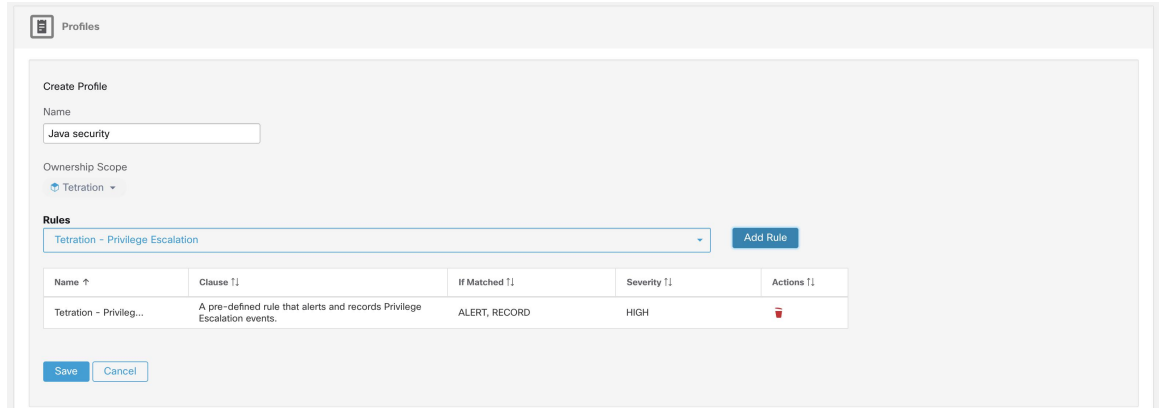
#### Procedure

- Step 1** In the navigation bar on the left, click **Defend > Forensic Rules**.
- Step 2** Click **Create Profile**.
- Step 3** Enter the appropriate values in the following fields.

| Field                  | Description                                         |
|------------------------|-----------------------------------------------------|
| <b>Name</b>            | Enter a name for the profile. Name cannot be empty. |
| <b>Ownership scope</b> | Enter an ownership scope for this profile.          |
| <b>Rules</b>           | Add rules into this profile.                        |



Figure 336: Create Profile



**Step 4** Click **Save**.

## Edit a Profile

This section explains how a user edit forensic profiles.

Before You Begin

You must login as **Site Admin**, **Customer Support** or **Scope Owner** in the system.

### Procedure

- Step 1** In the navigation bar on the left, click **Defend > Forensic Rules**.
- Step 2** Find the profile you want to edit and click the **pencil** icon in the column on the right.
- Step 3** Enter the appropriate values in the following fields.

| Field                  | Description                                          |
|------------------------|------------------------------------------------------|
| <b>Name</b>            | Update a name for the profile. Name cannot be empty. |
| <b>Ownership scope</b> | Update an ownership scope for this profile.          |
| <b>Rules</b>           | Add/remove rules into this profile.                  |

**Step 4** Click **Save**.

## Clone a Profile

This section explains how a user clones forensic profiles.

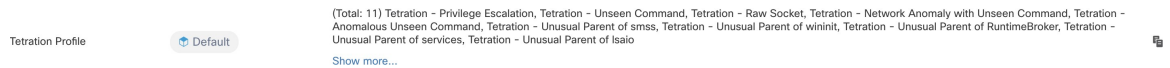
## Procedure

- 
- Step 1** In the navigation bar on the left, click **Defend > Forensic Rules**.
  - Step 2** Find the profile that you want to clone and click the **clone** icon in the column on the right.
  - Step 3** Enter the name for the cloned profile.
  - Step 4** Click **Save**.
- 

## Default Profile - Secure Workload Profile

The Secure Workload profile contains 11 default forensic rules and can be added to intents. It is not editable by the user but it can be cloned. The cloned default forensic profile is editable.

**Figure 337: Default profiles**



## Default Profile - MITRE ATT&CK Profile

The MITRE ATT&CK Profile contains 39 MITRE ATT&CK rules and can be added to intents. It is not editable by the user but it can be cloned. The cloned profile is editable. MITRE ATT&CK Profile includes the following rules:

1. Suspicious MS Office behavior
2. T1015 - Accessibility features 1
3. T1015 - Accessibility features 2
4. T1085 - rundll32
5. T1118 - InstallUtil
6. T1121 - Regsvcs/Regasm
7. T1127 - Trusted Developer Utilities - msbuild.exe
8. T1127 - Trusted Developer Utilities - rcsi.exe
9. T1127 - Trusted Developer Utilities - tracker.exe
10. T1128 - Netsh Helper Dll
11. T1136 - Create Account
12. T1138 - Application Shimming
13. T1180 - Screensaver
14. T1191 - CMSTP
15. T1202 - Indirect Command Execution - forfiles.exe
16. T1202 - Indirect Command Execution - pcalua.exe
17. T1216 - Signed Script Proxy Execution - pubprn.vbs

18. T1218 - Signed Binary Proxy Execution - msiexec.exe
19. T1218 - Signed Binary Proxy Execution - odbconf.exe
20. T1218 - Signed Binary Proxy Execution - Register-CimProvider
21. T1220 - XSL Script Processing - msxsl.exe
22. T1220 - XSL Script Processing - wmic
23. T1223 - Compiled HTML Files
24. T1003 - Credential Dumping - Lsass
25. T1140 - Deobfuscate/Decode Files or Information
26. T1076 - Remote Desktop Protocol
27. T1197 - BITS Jobs - Powershell
28. T1170 - MSHTA
29. T1158 - Hidden Files and Directories
30. T1114 - Email Collection
31. T1070 - Indicator Removal on Host - Event Log
32. T1070 - Indicator Removal on Host - USN
33. T1053 - Scheduled Task
34. T1003 - Credential Dumping - Vaultcmd
35. T1003 - Credential Dumping - Registry
36. T1201 - Password Policy Discovery 1
37. T1081 - Credentials in Files - Linux
38. T1081 - Credentials in Files - Windows
39. T1089 - Disabling Security Tools

## Forensic visualization

### Accessing Forensic Page

This section explains how to access forensic page.

Before You Begin

You must log in as **Site Admin**, **Customer Support** or **Scope Owner** in the system.

## Procedure

---

- Step 1** Click on **Security** link on the left panel.
- Step 2** Click on **Forensics** item. Forensic page appears.

*Figure 338: Security Forensic*

---

## Browsing Forensic Events

This section explains how to browse matching forensic events.

Before You Begin

You must log in as **Site Admin**, **Customer Support** or **Scope Owner** in the system and navigate to the forensic page.

### Procedure

---

**Step 1** Choose a specific range in the **Time Range Picker** at the top of the page.

**Step 2** Select **Severity** drop-down.

**Step 3** In **Filters**, enter filters for matching forensic events and click on **Filter Forensic Events**.

**Step 4** Table of matching forensic events is updated, according to the selected time range, severity, and filters.

**Note** Forensic events are visible under the root scope level and will not be visible upon switching to sub/child scopes.

---

## Inspecting a Forensic Event

This section explains how to inspect forensic events.

Before You Begin

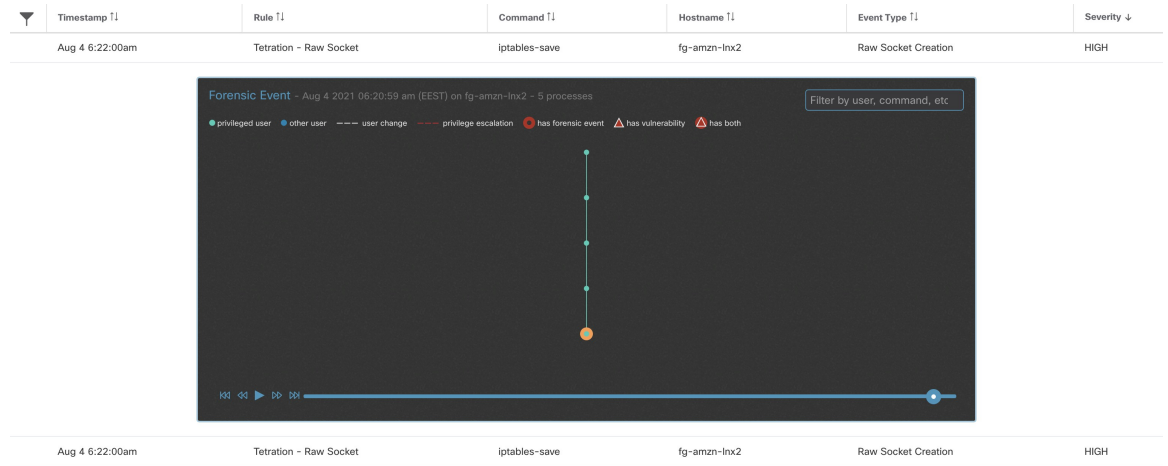
You must log in as **Site Admin**, **Customer Support** or **Scope Owner (Root Scope)** in the system.

### Procedure

---

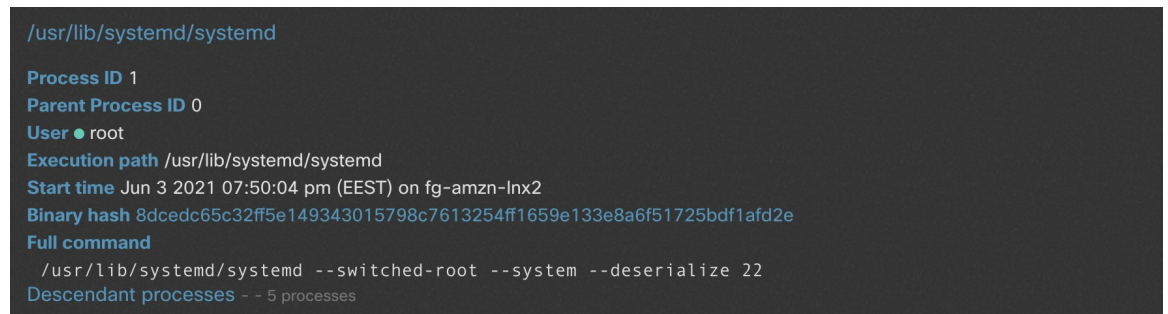
**Step 1** Click on the event to be inspected. The **Process detail** pane appears.

Figure 339: Forensic Event Table



**Step 2** On lineage tree, click on process to be inspected for details.

Figure 340: Forensic Process Details



## Fields Displayed in Forensic Events

Each Forensic Event has several fields which provide useful data. There are a few fields common to all the different types of forensic events, and there are a few fields unique to a particular forensic event. Below is a list of the fields that are part of the UI. The first table describes the fields common to all forensic events, followed by a table that describes process information that is displayed with each alert and then the tables with unique fields per forensic event. Some of the fields may be present in multiple tables, because of the way the data is stored and exported.

### Common Fields

| Field          | Description                                                 |
|----------------|-------------------------------------------------------------|
| Bin attr ctime | Changed time in linux/ Create time in windows of the binary |

| Field           | Description                                         |
|-----------------|-----------------------------------------------------|
| Bin attr hash   | Sha256 hash of the binary                           |
| Bin attr mtime  | Modified time of the binary                         |
| Bin attr name   | Name of the binary on the file system               |
| Bin attr size   | Size of the binary on the file system               |
| Bin exec path   | Full path of the binary                             |
| Cmdline         | Full command line of the process that gets executed |
| Event time usec | Time (in microseconds) when this event is observed  |

## Process Info

| Field             | Description                                              |
|-------------------|----------------------------------------------------------|
| Process ID        | Process ID of the process                                |
| Parent Process ID | Process ID of the parent of the process                  |
| User              | User that executed the process                           |
| Execution path    | Full path of the binary that corresponds to the process. |
| Start time        | Time when the process was started                        |
| Full command      | Full command line of the process that gets executed      |

## Privilege Escalation

| Field                        | Description                                       |
|------------------------------|---------------------------------------------------|
| Parent cmdline               | Full command line of the parent of the process    |
| Parent exe                   | Full path of the parent of the process            |
| Parent Uptime (microseconds) | Time since the parent of the process was executed |
| Parent Username              | User that executed the parent of the process      |
| Types bitmap suid binary     | Indicates whether the binary has the suid bit set |

## User Logon

| Field              | Description                       |
|--------------------|-----------------------------------|
| Auth type password | Indicates password authentication |



| Field                             | Description                                                         |
|-----------------------------------|---------------------------------------------------------------------|
| Auth type pubkey                  | Indicates key based authentication                                  |
| Type login ssh                    | Indicates that a user logged in via ssh                             |
| Type login win batch              | Indicates windows batch login (Type 4, eg schtasks)                 |
| Type login win cached             | Indicates logon via cached credentials (Type 11, CachedInttractive) |
| Type login win interactive        | Indicates interactive logon (Type 2, eg RDP)                        |
| Type login win network cleartext  | Indicates logon via ssh (Type 8)                                    |
| Type login win network            | Indicates network login (Type 3, eg Psexec)                         |
| Type login win new cred           | Indicates the usage of new credentials (Type 9, eg Runas command)   |
| Type login win remote interactive | Indicates remote logon (Type 10, eg RDP)                            |
| Type login win service            | Indicates that a service was started by SCM (Type 5)                |
| Type login win unlock             | Indicates that the workstation was unlocked (Type 7)                |
| Src IP                            | The source IP from which the login event was generated              |
| Src Port                          | The source port from which the login event was generated            |
| Username                          | Username associated with the log in event                           |

## User Logon Failed

| Field                            | Description                                                         |
|----------------------------------|---------------------------------------------------------------------|
| Auth type password               | Indicates password authentication                                   |
| Auth type pubkey                 | Indicates key based authentication                                  |
| Type login ssh                   | Indicates that a user logged in via ssh                             |
| Type login win batch             | Indicates windows batch login (Type 4, eg schtasks)                 |
| Type login win cached            | Indicates logon via cached credentials (Type 11, CachedInttractive) |
| Type login win interactive       | Indicates interactive logon (Type 2, eg RDP)                        |
| Type login win network cleartext | Indicates logon via ssh (Type 8)                                    |
| Type login win network           | Indicates network login (Type 3, eg Psexec)                         |

| Field                             | Description                                                       |
|-----------------------------------|-------------------------------------------------------------------|
| Type login win new cred           | Indicates the usage of new credentials (Type 9, eg Runas command) |
| Type login win remote interactive | Indicates remote logon (Type 10, eg RDP)                          |
| Type login win service            | Indicates that a service was started by SCM (Type 5)              |
| Type login win unlock             | Indicates that the workstation was unlocked (Type 7)              |
| Src IP                            | The source IP from which the login event was generated            |
| Src Port                          | The source port from which the login event was generated          |
| Username                          | Username associated with the log in event                         |

## Shellcode

| Field                                  | Description                                                                         |
|----------------------------------------|-------------------------------------------------------------------------------------|
| Signal sources bitmap cmd as sh no tty | Indicates that a shell process has no tty that is associated with it                |
| Signal sources bitmap powershell       | Indicates that the process has powershell dll loaded (System.Management.Automation) |

## File Access

| Field                | Description                                            |
|----------------------|--------------------------------------------------------|
| File                 | Full path of the file that was accessed                |
| Perm read perm       | Indicates that the file had Read permission            |
| Perm read write perm | Indicates that the file had Read and Write permissions |
| Perm write perm      | Indicates that the file had Write permission           |

## User Account

| Field        | Description                            |
|--------------|----------------------------------------|
| Username     | Username of the user that was created  |
| Ops acct add | Indicates that a new account was added |

## Unseen Command

| Field                         | Description                                                                                                                             |
|-------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------|
| Anomaly - Score               | Score (0 to 1.0) indicating how frequently the command line was seen previously, lower score implies that the command is more anomalous |
| Anomaly - Similarity - High   | True if the anomaly score is larger than 0.8 and is smaller than 1                                                                      |
| Anomaly - Similarity - Medium | True if the anomaly score is larger than 0.6 and is smaller than or equal to 0.8                                                        |
| Anomaly - Similarity - Low    | True if the anomaly score is larger than 0 and is smaller than or equal to 0.6                                                          |
| Anomaly - Similarity - Seen   | True if the anomaly score is 1, i.e. the same command has been seen before                                                              |
| Anomaly - Similarity - Unique | True if the anomaly score is 0, i.e. the command has never been seen before                                                             |
| Parent cmdline                | Full command line of the parent process                                                                                                 |
| Parent exepath                | Binary path of the parent process                                                                                                       |
| Parent uptime                 | Time since the parent process was executed                                                                                              |
| Parent username               | Username of the user that executed the parent process                                                                                   |
| Sensor uptime                 | Uptime of the sensor                                                                                                                    |

## Unseen Library

| Field    | Description                                                                         |
|----------|-------------------------------------------------------------------------------------|
| Lib Path | The full path of the library file that was previously not associated to the process |

## Raw Socket Creation

| Field    | Description                                          |
|----------|------------------------------------------------------|
| Exe Path | Full path of the process that created the raw socket |

## Library Changed

| Field                | Description                                   |
|----------------------|-----------------------------------------------|
| Library changed name | The full path of the Library that was changed |

## Side Channel

| Field                          | Description                           |
|--------------------------------|---------------------------------------|
| Signal sources bitmap meltdown | Indicates the use of Meltdown exploit |

## Follow User Logon

| Field    | Description                        |
|----------|------------------------------------|
| Username | Username that executed the process |

## Follow Process

| Field                        | Description                                                                         |
|------------------------------|-------------------------------------------------------------------------------------|
| Parent cmdline               | Full command line of the parent process                                             |
| Parent exeopath              | Binary path of the parent process                                                   |
| Parent uptime usec           | Time since the parent process was executed                                          |
| Parent username              | Username of the user that executed the parent process                               |
| Time since last changed usec | Time elapsed between the process start time and its binary file change time (mtime) |
| Username                     | Username of the user that executed the process                                      |

## Network Anomaly

For more information, see [Forensic Rules for Network Anomaly Events](#) for the list of attributes associated with Network Anomaly events.

## Forensic Analysis - Searchable Fields

The below tables describe searchable fields on the Forensics Analysis page search bar.

## Miscellaneous Fields

| Field              | Description                                  |
|--------------------|----------------------------------------------|
| Forensic Rule Name | Events labeled by a particular forensic rule |
| Hostname           | Events from a particular hostname            |
| Sensor ID          | Events from a particular Sensor              |
| Severity           | Events of a particular severity              |

## Search Terms in Forensic Analysis

### Common Fields

These fields are common to various event types. They have the prefix “Event name - Event”, for example, “Binary Changed - Binary Attribute - CTime (epoch nanoseconds)”

| Field                                        | Description                                                 |
|----------------------------------------------|-------------------------------------------------------------|
| Binary Attribute - CTime (epoch nanoseconds) | Changed time in linux/ Create time in windows of the binary |
| Binary Attribute - Hash                      | Sha256 hash of the binary                                   |
| Binary Attribute - MTime (epoch nanoseconds) | Modified time of the binary                                 |
| Binary Attribute - Filename                  | Name of the binary on the file system                       |
| Binary Attribute - Size (bytes)              | Size of the binary on the file system                       |
| Event Binary Path                            | Full path of the binary                                     |
| Command Line                                 | Full command line of the process that gets executed         |

### Binary Changed

There are no other search terms other than the ones described in “Common Fields” table.

### File Access

File Access search terms have the prefix “File Access - “, for example, “File Access - Filename”

| Field                  | Description                                 |
|------------------------|---------------------------------------------|
| Filename               | Full path of the file that was accessed     |
| Is = Permission - Read | Indicates that the file had Read permission |

| Field                       | Description                                            |
|-----------------------------|--------------------------------------------------------|
| Is = Permission - ReadWrite | Indicates that the file had Read and Write permissions |
| Is = Permission - Write     | Indicates that the file had Write permission           |

## Follow Process

Follow Process search terms have the prefix “Follow Process - “, for example, “Follow Process - Parent Command Line”

| Field                                                     | Description                                                                                |
|-----------------------------------------------------------|--------------------------------------------------------------------------------------------|
| Parent Command Line                                       | Full command line of the parent process                                                    |
| Parent Exec Path                                          | Binary path of the parent process                                                          |
| Parent Uptime (microseconds)                              | Time since the parent process was executed                                                 |
| Parent Username                                           | Username of the user that executed the parent process                                      |
| Process Start Time Since Last File Changed (microseconds) | Time that is elapsed between process start and the most recent (corresponding) file change |
| Username                                                  | Username that are associated with the process being followed                               |

## Follow User Logon

Follow User Logon search terms have the prefix "Follow User Logon - ", for example, "Follow User Logon - Username"

| Field    | Description                                |
|----------|--------------------------------------------|
| Username | Username that is associated with a process |

## Ldap

Ldap search terms have the prefix “Ldap - “, for example, “Ldap - Department”

| Field       | Description                                                                   |
|-------------|-------------------------------------------------------------------------------|
| Department  | AMS Ldap user department associated with the process username (if available)  |
| Description | AMS Ldap user description associated with the process username (if available) |
| Username    | AMS Ldap username associated with the process (if available)                  |

## Library Changed

Library Changed search terms have the prefix “Library Changed - “, for example, “Library Changed - Department”

| Field        | Description                                   |
|--------------|-----------------------------------------------|
| Lib Filename | The full path of the Library that was changed |

## Privilege Escalation

Privilege Escalation search terms have the prefix “Privilege Escalation - “, for example, “Privilege Escalation - Parent Com- mand Line”

| Field                        | Description                                       |
|------------------------------|---------------------------------------------------|
| Parent Command Line          | Full command line of the parent of the process    |
| Parent Exec Path             | Full path of the parent of the process            |
| Parent Uptime (microseconds) | Time since the parent of the process was executed |
| Parent Username              | User that executed the parent of the process      |
| Type - Suid Binary           | Indicates whether the binary has the suid bit set |

## Process Info

Process Info search terms have the prefix “Process Info - “, for example, “Process Info - Binary Hash”

| Field                    | Description                                             |
|--------------------------|---------------------------------------------------------|
| Binary Hash              | Hash of the binary associated with the process          |
| Command String Tokenized | Tokenized command line of the process                   |
| Command String           | Full command line of the process                        |
| Exec Path                | Full path of the binary that corresponds to the process |

## Raw Socket

Raw Socket search terms have the prefix “Raw Socket - ”, for example, “Raw Socket - Exec Path”

| Field     | Description                                          |
|-----------|------------------------------------------------------|
| Exec Path | Full path of the process that created the raw socket |

## Shellcode

Shellcode search terms have the prefix “Shellcode - ”, for example, “Shellcode - Source - Not From Login”

| Field                   | Description                                                                         |
|-------------------------|-------------------------------------------------------------------------------------|
| Source - Not From Login | Indicates that a shell process has no tty that is associated with it                |
| Source - Powershell     | Indicates that the process has powershell dll loaded (System.Management.Automation) |

## Side Channel

Side Channel search terms have the prefix “Shellcode - ”, for example, “Shellcode - Source - Meltdown”

| Field             | Description                           |
|-------------------|---------------------------------------|
| Source - Meltdown | Indicates the use of Meltdown exploit |

## Unseen Command

Unseen Command search terms have the prefix “Unseen Command - ”, for example, “Unseen Command - Anomaly - Similarity - High”

| Field                         | Description                                                                                                                             |
|-------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------|
| Anomaly - Score               | Score (0 to 1.0) indicating how frequently the command line was seen previously, lower score implies that the command is more anomalous |
| Anomaly - Similarity - High   | True if the anomaly score is larger than 0.8 and is smaller than 1                                                                      |
| Anomaly - Similarity - Medium | True if the anomaly score is larger than 0.6 and is smaller than or equal to 0.8                                                        |
| Anomaly - Similarity - Low    | True if the anomaly score is larger than 0 and is smaller than or equal to 0.6                                                          |
| Anomaly - Similarity - Seen   | True if the anomaly score is 1, i.e. the same command has been seen before                                                              |
| Anomaly - Similarity - Unique | True if the anomaly score is 0, i.e. the command has never been seen before                                                             |
| Parent Cmdline                | Full command line of the parent process                                                                                                 |
| Parent Exepath                | Binary path of the parent process                                                                                                       |
| Parent Uptime                 | Time since the parent process was executed                                                                                              |



| Field                             | Description                                                                        |
|-----------------------------------|------------------------------------------------------------------------------------|
| Parent Username                   | Username of the user that executed the parent process                              |
| Sensor Uptime                     | Uptime of the sensor                                                               |
| Anomaly - Latest Similar Commands | 5 latest previously observed command which are similar to the command of the event |

## Unseen Library

Unseen Library search terms have the prefix “Unseen Library - ”, for example, “Unseen Library - Lib Filename”

| Field        | Description                                                                         |
|--------------|-------------------------------------------------------------------------------------|
| Lib Filename | The full path of the library file that was previously not associated to the process |

## User Account

User Account search terms have the prefix “User Account - ”, for example, “User Account - Account Name”

| Field                   | Description                            |
|-------------------------|----------------------------------------|
| Account Name            | Username of the user that was created  |
| Operation - Add Account | Indicates that a new account was added |

## User Logon

User Logon search terms have the prefix “User Logon - ”, for example, “User Logon - Auth Type - Password”

| Field                                  | Description                                                         |
|----------------------------------------|---------------------------------------------------------------------|
| Auth Type - Password                   | Indicates password authentication                                   |
| Auth type - Pubkey                     | Indicates key based authentication                                  |
| Login Type - Login Via SSH             | Indicates that a user logged in via ssh                             |
| Login Type - Windows Login Batch       | Indicates windows batch login (Type 4, eg schtasks)                 |
| Login Type - Windows Login Cached      | Indicates logon via cached credentials (Type 11, CachedInttractive) |
| Login Type - Windows Login Interactive | Indicates interactive logon (Type 2, eg RDP)                        |
| Login Type - Windows Network Cleartext | Indicates logon via ssh (Type 8)                                    |
| Login Type - Windows Network           | Indicates network login (Type 3, eg Psexec)                         |

| Field                                         | Description                                                       |
|-----------------------------------------------|-------------------------------------------------------------------|
| Login Type - Windows Login New Credential     | Indicates the usage of new credentials (Type 9, eg Runas command) |
| Login Type - Windows Login Remote Interactive | Indicates remote logon (Type 10, eg RDP)                          |
| Login Type - Windows Login Service            | Indicates that a service was started by SCM (Type 5)              |
| Login Type - Windows Login Unlock             | Indicates that the workstation was unlocked (Type 7)              |
| Source IP                                     | The source IP from which the login event was generated            |
| Source Port                                   | The source port from which the login event was generated          |
| Username                                      | Username associated with the log in event                         |

## User Logon Failed

User Logon Failed search terms have the prefix “User Logon Failed - “, e.g., “User Logon Failed - Auth Type - Password”

| Field                                         | Description                                                         |
|-----------------------------------------------|---------------------------------------------------------------------|
| Auth Type - Password                          | Indicates password authentication                                   |
| Auth type - Pubkey                            | Indicates key based authentication                                  |
| Login Type - Login Via SSH                    | Indicates that a user logged in via ssh                             |
| Login Type - Windows Login Batch              | Indicates windows batch login (Type 4, eg schtasks)                 |
| Login Type - Windows Login Cached             | Indicates logon via cached credentials (Type 11, CachedInttractive) |
| Login Type - Windows Login Interactive        | Indicates interactive logon (Type 2, eg RDP)                        |
| Login Type - Windows Network Cleartext        | Indicates logon via ssh (Type 8)                                    |
| Login Type - Windows Network                  | Indicates network login (Type 3, eg Psexec)                         |
| Login Type - Windows Login New Credential     | Indicates the usage of new credentials (Type 9, eg Runas command)   |
| Login Type - Windows Login Remote Interactive | Indicates remote logon (Type 10, eg RDP)                            |
| Login Type - Windows Login Service            | Indicates that a service was started by SCM (Type 5)                |
| Login Type - Windows Login Unlock             | Indicates that the workstation was unlocked (Type 7)                |
| Source IP                                     | The source IP from which the login event was generated              |

| Field       | Description                                              |
|-------------|----------------------------------------------------------|
| Source Port | The source port from which the login event was generated |
| Username    | Username associated with the log in event                |

## Forensics alerts

Forensic events can be found in the Secure Workload Alert System if their matching rules contain an **Alert** action.

## Accessing Forensic Alerts

This section explains how to access forensic alerts.

### Before You Begin

- Login into the system as a **Site Admin**, **Customer Support** or **Scope Owner**.
- Turn on alerts for **Forensics** alert source.

### Procedure

---

- Step 1** From the navigation pane, choose **Configure Alerts**.
- Step 2** Alert page appears.
- 

## Checking Alert Details

Before You Begin:

You must log in as **Site Admin**, **Customer Support** or **Scope Owner** in the system.

### Procedure

---

- Step 1** From the alert page, click on the alert to be checked.
- Step 2** Click on profile/rule to see the details of the matching forensic profile/rule. If the matching profile/rule is updated after alerts are raised, there will be a warning indicator.



```
}
```

The value in `alert_details` is itself an escaped JSON string whose content for the above alert can be seen below:

```
{
 "Sensor Id": "d89f926cddc7577553eb8954e492528433b2d08e",
 "Hostname": "collectorDatamover-1",
 "Process Id": 20196,
 "scope_id": "5efcfd5497d4f474f1707c2",
 "forensic": {
 "Unseen Command": "true",
 "Unseen Command - Sensor Uptime (microseconds)": "34441125356",
 "Unseen Command - Parent Uptime (microseconds)": "35968418683",
 "Unseen Command - Parent Username": "root",
 "Unseen Command - Parent Command Line": "svlogd -tt /local/logs/tetration/efe/ ",
 "Unseen Command - Parent Exec Path": "/sbin/svlogd",
 "Unseen Command - Anomaly - Score": "0",
 "Unseen Command - Anomaly - Similarity - Unique": "true",
 "Process Info - Command String": "gzip ",
 "Process Info - Exec Path": "/bin/gzip"
 },
 "profile": {
 "id": "5efcfd6497d4f474f1707e4",
 "name": "Tetration Profile",
 "created_at": 1593638390,
 "updated_at": 1593638390,
 "root_app_scope_id": "5efcfd5497d4f474f1707c2"
 },
 "rule": {
 "id": "5efcfd6497d4f474f1707d6",
 "name": "Tetration - Anomalous Unseen Command",
 "clause_chips":
 "[{"type": "filter", "facet": {"field": "event_type", "title": "Event
type", "type": "STRING"}, "operator": {"label": "=", "type": "eq"}, "displayValue": "Unseen
Command", "value": "Unseen
Command"}, {"type": "filter", "facet": {"field": "forensic_event_and_not_seen_data_and_line_anomaly_info_score", "title": "Unseen
Command - Anomaly -
Score", "type": "NUMBER"}, "operator": {"label": "<", "type": "lt"}, "displayValue": "0.6", "value": "0.6"}]",
 "created_at": 1593638390,
 "updated_at": 1595539498,
 "root_app_scope_id": "5efcfd5497d4f474f1707c2"
 }
}
```

The details of the forensic events are included in the field `forensic`. For the list of attributes of the forensic events, see [Fields Displayed in Forensic Events](#). These attributes are also shown in the alert details in the UI.

## Forensics Score

### Where to See Forensic Score

Security Dashboard:

Figure 342: Forensics Score Section in Security Dashboard

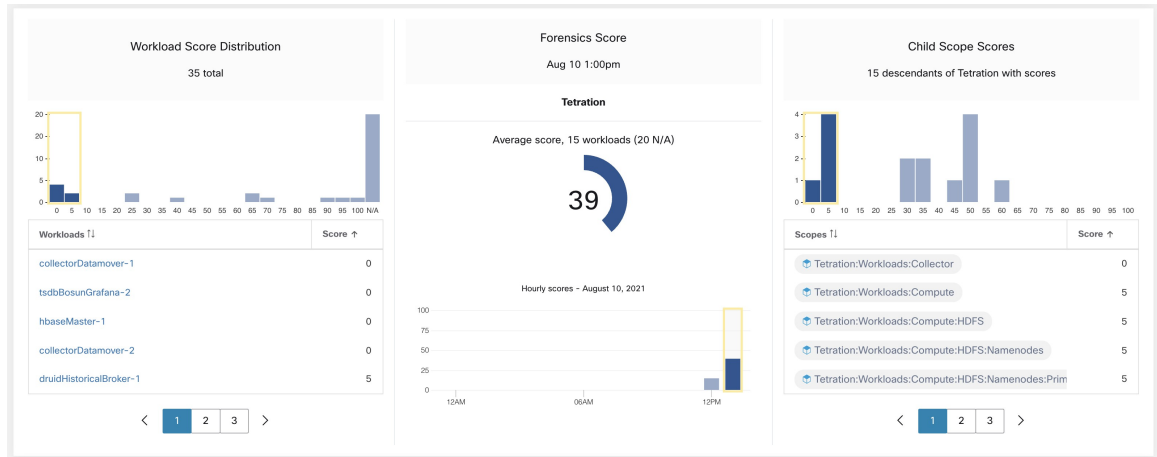


Figure 343: Forensics Score Details Section in Security Dashboard



9 Forensic Events

| Timestamp        | Rule                             | Command                            | Hostname    | Event Type     | Severity |
|------------------|----------------------------------|------------------------------------|-------------|----------------|----------|
| Aug 10 1:00:00pm | Tetration - Unseen Command       | /bin/sh (ps                        | zookeeper-1 | Unseen Command | LOW      |
| Aug 10 1:00:00pm | Tetration - Unseen Command       | /bin/bash /usr/bin/atopd           | zookeeper-1 | Unseen Command | LOW      |
| Aug 10 1:00:00pm | Tetration - Unseen Command       | /bin/sh (/usr/sbin/ntpq            | zookeeper-1 | Unseen Command | LOW      |
| Aug 10 1:00:00pm | Tetration - Unseen Command       | /bin/bash /etc/rc.d/init.d/atop    | zookeeper-1 | Unseen Command | LOW      |
| Aug 10 1:00:00pm | Tetration - Unseen Command       | /bin/bash ulimit                   | zookeeper-1 | Unseen Command | LOW      |
| Aug 10 1:01:00pm | Tetration - Unseen Command       | /bin/bash /etc/cron.hourly/0anacro | zookeeper-1 | Unseen Command | LOW      |
| Aug 10 1:01:00pm | Tetration - Unseen Command       | /bin/bash /usr/bin/run-parts       | zookeeper-1 | Unseen Command | LOW      |
| Aug 10 1:18:00pm | Tetration - Anomalous Unseen Con | bash /usr/hdp/current/zookeeper-c  | zookeeper-1 | Unseen Command | HIGH     |
| Aug 10 1:22:00pm | Tetration - Anomalous Unseen Con | pickup                             | zookeeper-1 | Unseen Command | HIGH     |

## How the Forensic Score is Calculated

For each Workload, we compute a Forensics Score. A Workload’s Forensics Score is derived from the Forensic Events observed on that Workload based on the profiles that are enabled for this scope. A score of 100 means that no Forensic Events were observed via configured rules in enabled profiles, and a score of 0 means that there is a Forensic Event detected that requires immediate action. The Forensics Score for a Scope is the average Workload score within that Scope. Forensics Score for a given hour is a minimum of all scores within that hour.

- A Forensic Event with the severity **REQUIRES IMMEDIATE ACTION** reduces the Score for the entire Scope to zero.
- A Forensic Event with the severity **CRITICAL** reduces workload’s score with the weight of 10.

- A Forensic Event with the severity HIGH reduces workload's score with the weight of 5.
- A Forensic Event with the severity MEDIUM reduces workload's score with the weight of 3.
- A Forensic Event with the severity LOW doesn't contribute to the Forensics Score. This is recommended for new rules where the quality of the signal is still being tuned and is likely to be noisy.

For example, a workload has 3 forensic events that match 2 rules with *CRITICAL* severity, 1 rule with *HIGH* severity, 1 rule with *LOW*, respectively. The forensic score for that workload is:  $100 - 1 * 10 - 1 * 5 - 1 * 0 = 85$ .

The Forensics Scores are N/A for workloads in which the Forensics feature is not enabled.

## How to Improve Forensic Score

Tuning your Forensics Score can be done by adjusting the Forensic Rules enabled. Creating rules that are less noisy will give you a more accurate score. Acting upon and preventing legitimate Forensic Events (events that are evidence of an intrusion or other bad activity) is another good way to improve your Forensics Score.

## Caveats

- Forensics Score details show all forensic events within that hour. That means Forensic Score details may show forensic events other than the ones used for computing forensic score.
- Forensics Score is currently available for Deep Visibility and Enforcement sensors.

## PCR-Based Network Anomaly Detection

Network Anomaly feature detects abnormally large amounts of data flowing into or out of the workloads based on the concept of Producer Consumer Ratio (PCR). The PCR is defined as:

$$\text{PCR} = \frac{\text{Egress app byte count} - \text{Ingress app byte count}}{\text{Egress app byte count} + \text{Ingress app byte count}}$$

The value of PCR is in the [-1.0, 1.0] range where:

- PCR = 1.0 means the workload purely sends data out.
- PCR = -1.0 means the workload purely receives data.
- PCR = 0.0 means the workload has balanced amounts of data in and data out.

Similar to other Forensics features, you can use the intent-based configuration to configure the Network Anomaly events you want to record and/or alert. Detected Network Anomaly events from workloads are exported every 5 minutes and are matched against configured rules 5 minutes later. As a result, new Network Anomaly events are only observed on the UI every 5 minutes with delay of up to 10 minutes from the time of the event.




---

**Note** In 3.2 and 3.1 versions of Secure Workload software, Network Anomaly detection was known as Data Leak detection.

---

## Forensic Rules for Network Anomaly Events

Refer to [Forensic Configuration](#) on how to add forensic rules.

### Rule Attributes

This section explains the details of the attributes to define a Network Anomaly related rule. The simplest Network Anomaly rule is:

```
Event Type = Network Anomaly
```

Other attributes in the Network Anomaly event to refine the rules for your data centers:

**Table 31: Rule Attributes in Network Anomaly event**

| Attribute                      | Description                                                                                                                                                                                                     |
|--------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Host Name                      | The host name of the workload emitting this event.                                                                                                                                                              |
| Timestamp (epoch milliseconds) | The timestamp (in milliseconds) of the event.                                                                                                                                                                   |
| PCR Deviation                  | The deviation of PCR from the mean at the event time as a multiple of historical standard deviation.                                                                                                            |
| Non-seasonal Deviation         | This is the PCR deviation after removing the seasonality pattern (for example, by cron-jobs). The value of Non-seasonal Deviation is always larger than or equal to 6.0.                                        |
| PCR                            | The Producer-Consumer Ratio.                                                                                                                                                                                    |
| EIR                            | The Egress Ingress Ratio, which is the ratio between the total Egress App Byte Count and the Ingress App Byte Count.                                                                                            |
| Egress App Byte Count          | The egress application byte count, which is the total byte count of packet contents (excluding headers) flowing out of the work-load.                                                                           |
| Ingress App Byte Count         | The ingress application byte count, which is the total byte count of packet contents (excluding headers) flowing into the work-load.                                                                            |
| Protocol                       | The protocol for which the PCR time series is calculated. Currently, the supported protocols are TCP, UDP, and Aggregate. Aggregate PCR is calculated based on the total sum of TCP, UDP, and ICMP byte counts. |



| Attribute                     | Description                                                                                                                                                                                                                                                                                                                                                                 |
|-------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| User Logon Count              | The number of user login events on the workload within approximately the last 15 minutes. This is the count of the User Logon events regardless of whether there are matched rules. To know the details of the User Logon events, you must define rules to record the events for workloads of interests and view them on the Forensics Analysis page.                       |
| User Logon Failed Count       | The number of users logon failed events on the workload within approximately the last 15 minutes. This is the count of the User Logon failed events regardless of whether there are matched rules. To know the details of the User Logon Failed events, you must define rules to record the events for workloads of interests and view them on the Forensics Analysis page. |
| Unseen Command Count          | The number of unseen command events on the workload within approximately the last 15 minutes. This is the count of the Unseen Command events regardless of whether there are matched rules. To know the details of the Unseen Command events, you must define rules to record the events for workloads of interests and view them on the Forensics Analysis page.           |
| Date Time (UTC) - Year        | The year of the event time.                                                                                                                                                                                                                                                                                                                                                 |
| Date Time (UTC) - Month       | The month of the event time (1, 2, . . .).                                                                                                                                                                                                                                                                                                                                  |
| Date Time (UTC) - Day         | The day of a month of the event time (1, 2, . . .).                                                                                                                                                                                                                                                                                                                         |
| Date Time (UTC) - Hour        | The hour of a day of the event time (1, 2, . . . , 24).                                                                                                                                                                                                                                                                                                                     |
| Date Time (UTC) - Minute      | The minute of an hour of the event time (1, 2, . . . , 60).                                                                                                                                                                                                                                                                                                                 |
| Date Time (UTC) - Second      | The second of minute of the event time (1, 2, . . . , 60).                                                                                                                                                                                                                                                                                                                  |
| Date Time (UTC) - Day of Week | The day of a week of the event time (0-7, for Monday to Sunday).                                                                                                                                                                                                                                                                                                            |

Figure 344: Defining Forensic Rules for Network Anomaly Events

Create Rule

Rule Name  
Network Anomaly with Failed Logins

Ownership Scope  
Tetration

Actions  
ALERT, RECORD

Severity  
HIGH

Clause

Network Anomaly - User Logon Count > 0 AND Event type = Network Anomaly AND Network Anomaly - Non-seasonal deviation > 5.5

Save Cancel

Below are some sample rules:

Listing 7.10.1.1.1: Detects network anomalies for UDP only.

```
Event Type = Network Anomaly AND Network Anomaly Is = Protocol - UDP
```

Listing 7.10.1.1.2: Detects large deviation after removing seasonal pattern (if detected), with a threshold on the egress app byte count for a subset of workloads whose names contain *sensitiveDataServer*.

```
Event Type = Network Anomaly AND Network Anomaly - Non-seasonal Deviation > 10.0)
AND Network Anomaly - Egress App Byte Count > 1000000
AND Network Anomaly - Host Name CONTAINS sensitiveDataServer
```

Listing 7.10.1.1.3: Detects Network Anomaly events on workloads with unseen command events except the Network Anomaly events happen from 7.30AM UTC to 7.35AM UTC everyday.

```
Event Type = Network Anomaly AND Network Anomaly - Unseen Command Count > 0
AND (Network Anomaly - Date Time (UTC) - Hour != 7
OR Network Anomaly - Date Time (UTC) - Minute < 30 OR Network Anomaly - Date Time (UTC)
- Minute > 35)
```

## Rule Actions

| Action | Description                                                                                                                                           |
|--------|-------------------------------------------------------------------------------------------------------------------------------------------------------|
| RECORD | The matched events contribute to the Network Anomaly Score and can be found via the Security Dashboard or the <a href="#">Network Anomalies Tab</a> . |
| ALERT  | The matched events shows up in the <a href="#">Current Alerts</a> and the chosen <a href="#">Choose Alert Publishers</a> .                            |

The next section describes in more detail where to find detected Network Anomaly events in the UI.

# Where to See Network Anomaly Events



**Note** Network Anomaly events are *not* currently shown on Forensics Analysis page. You can find Network Anomaly events on the following pages.

- **Security Dashboard:** Network Anomaly events that match rules with **RECORD** action can be found in the Network Anomaly score section in the Security Dashboard. If there are workloads with nonbest (less than 100) scores, clicking on the workload name, you are able to view the PCR time series and the Network Anomaly events on that workload. On the right side of each row of the Network Anomaly event table, you can see action links that can help you search for flows and other forensic events around the time of the corresponding Network Anomaly event. See [Network Anomaly Latency](#) for known delay in Network Anomaly score reporting.

**Figure 345: Network Anomaly Score in Security Dashboard**

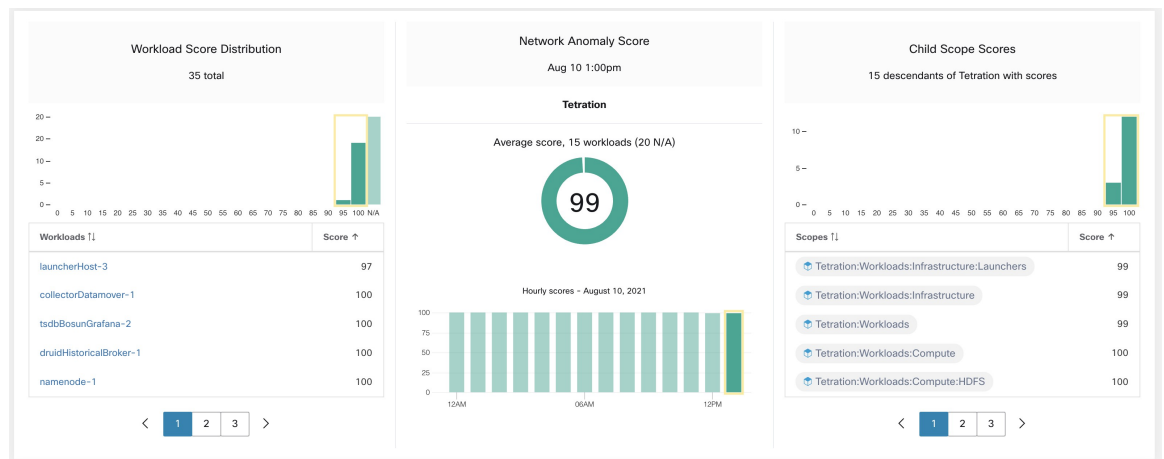
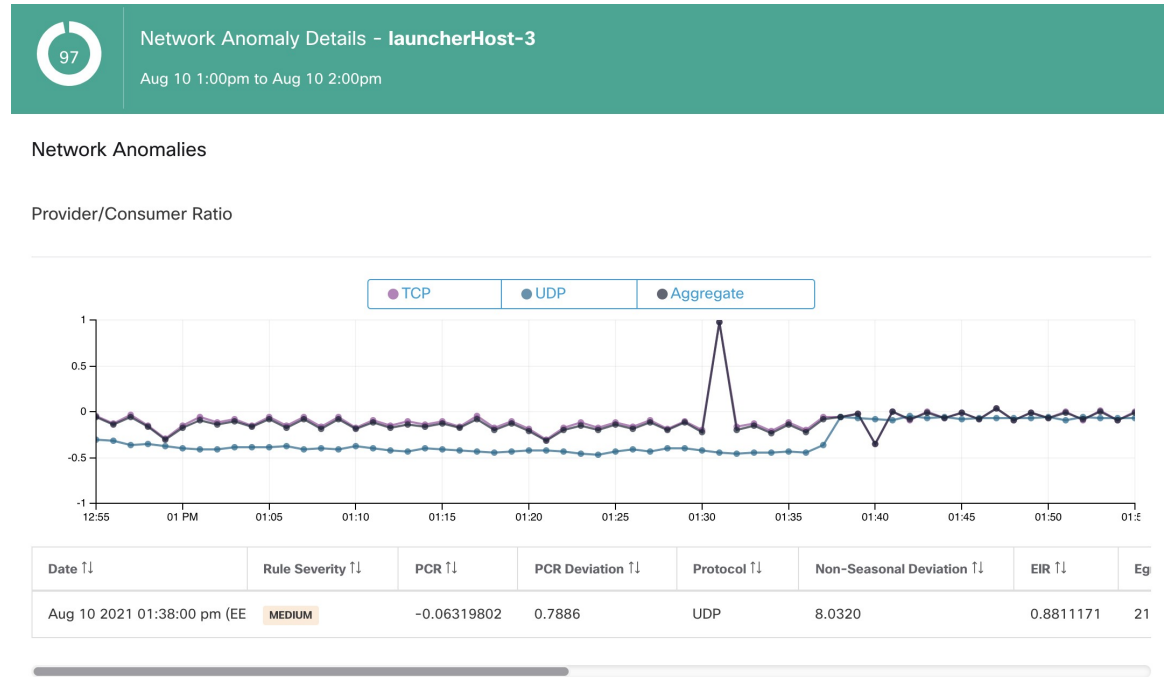


Figure 346: Network Anomaly score in Security Dashboard Drilled-Down by Workload



- **Network Anomalies Tab:** on this page, you can see the PCR time series graph and the Network Anomaly events that match rules with **RECORD** action. What you can see that on this page is similar to what you find by clicking on the workload name in the security dashboard.

Figure 347: Network Anomaly Tab on Workload Profile Page



- **Alerts:** If the Network Anomaly rule is configured with **ALERT** action, the matched events is displayed on the **Current Alerts** and are also available via Alert Publisher.

Figure 348: Network Anomaly Alert

| Event Time | Status | Alert Text                                                              | Severity | Type      | Actions |
|------------|--------|-------------------------------------------------------------------------|----------|-----------|---------|
| 2:38 PM    | ACTIVE | Tetration - Network Anomaly with Unseen Command on launcherHost-2 (UDP) | MEDIUM   | FORENSICS |         |

Details

**Profile** Tetration Profile

**Rule** Tetration - Network Anomaly with Unseen Command

**Alert Trigger** Event type = Network Anomaly    Network Anomaly - Unseen Command Count > 3  
 Network Anomaly - Non-seasonal deviation > 0

**Forensic Event** Host Name = launcherHost-2  
 Network Anomaly = true  
 Network Anomaly - Date Time (UTC) - Day = 10  
 Network Anomaly - Date Time (UTC) - Day of Week = 2  
 Network Anomaly - Date Time (UTC) - Hour = 11  
 Network Anomaly - Date Time (UTC) - Minute = 38  
 Network Anomaly - Date Time (UTC) - Month = 8  
 Network Anomaly - Date Time (UTC) - Second = 0

## Rule Severities and Network Anomaly Scores

The Network Anomaly Score is computed similarly to the Forensics Score. For each Workload we compute a Network Anomaly Score. The Network Anomaly Score of a Workload is derived from the Network Anomaly Events observed on that Workload based on the profiles that are enabled for this scope. A score of 100 means no Network Anomaly Events were observed via configured rules in enabled profiles. A score of 0 means there is a Network Anomaly Event detected that requires immediate action.

- A Network Anomaly Event with the severity **REQUIRES IMMEDIATE ACTION** reduces the Score for the entire Scope to 0.
- A Network Anomaly Event with the severity **CRITICAL** reduces workload's score with the impact of 10.
- A Network Anomaly Event with the severity **HIGH** reduces workload's score with the impact of 5.
- A Network Anomaly Event with the severity **MEDIUM** reduces workload's score with the impact of 3.
- A Network Anomaly Event with the severity **LOW** doesn't contribute to the Network Anomaly Score. This is recommended for new rules where the quality of the signal is still being tuned and is likely to be noisy.

For each workload, the total impact score is aggregated every 5 minutes to compute the score of that workload within those 5 minutes.

For workloads without Network Anomaly enabled sensor types, the Network Anomaly scores are N/A.

## PCR Data and Network Anomaly Events Retention

PCR data and Network Anomaly events are kept for 7 days.

## Network Anomaly Latency

Network Anomaly scores reported in the security dashboard have 5-minute delays. For instance, the score of a workload for the hour 10:00 a.m-10:59 a.m is based on Network Anomaly events happen from 9:55 a.m to 10:54 a.m.

## Caveats

- Old Data Leak events remain as Data Leak events instead of Network Anomaly events.
- Network Anomaly detection per protocol is a new feature in 3.3 and the protocol is not set in old Data Leak events.

## Process Hash Anomaly Detection

As the name suggested, this feature detects process hash anomalies by assessing the consistency of process binary hashes across the system. The motivation of this feature is as follows. Imagine that you have a farm of Apache web servers that are cloned from the same setup configuration (for example, those servers are deployed from the same automation scripts). You can expect that the hashes of [httpd](#) binaries on all servers are the same. If there is a mismatch, it is an anomaly and might worth a further investigation.

Formally, we define a *process group* as the set of processes across workloads in the same rootscope that have the same combination of executable binary path, OS version, and package info (if applicable)1.




---

**Note** Package information is included since 3.4 release; in the previous releases, the process group is defined based on the combination of executable binary path and only the OS version.

---

In the above example, if all Apache web servers are running httpd 2.4.43 on CentOS 7.7 and in the same rootscope, then the corresponding process group is the set of processes (across all servers) that have the same combination: binary path of `/usr/sbin/httpd` & OS version of `CentOS-7.7` & package version of `httpd-2.4.43`. It is expected that the hashes of all binaries in the same process group are the same, and an anomaly will appear if any mismatch is detected.

Besides detecting anomalous process hashes, this feature also detects process hashes that appear in a [Flagged list File Hashes](#) that are uploaded. The motivation is that you may have a list of known malware hashes, and want to know if a process associated with any of these hashes is run.

To reduce false alarms, we use the [National Software Reference Library's Reference Data Set \(RDS\)](#) provided by NIST, also called NIST RDS dataset as a Benign list; a benign hash is considered “safe” (see the Analyze Threat Intelligence Reports section on how to enable NIST RDS dataset). Also, you can see the File Hashes section to upload from your own hash Benign list.

In addition to the NIST RDS dataset, we also curate Secure Workload **Hash Verdict** service. When you enable the service, if any known malware hash shows up, it is detected as a malicious hash. However, if the hash is known and legit, then it is also marked as benign in the anomaly analysis. Due to the large dataset and fast updates that cover all known and legit process hashes that can be used to either approve or red flag processes running on a workload, Secure Workload **Hash Verdict** is only available through Secure Workload Cloud. To ensure that Secure Workload **Hash Verdict** service is accessible from your appliance, see Automatic Threat Intelligence Updates.

Output of this feature is a security score known as **process hash score**. This score is calculated and output hourly. Like all other security scores, a higher process hash score is better. In particular, for a process hash:

- Hash score of 0 means that the hash is flagged or malicious.
- Hash score of 100 means that the hash is either benign, or consistent across workloads (no mismatch)

- Hash score from 1-99 means that the hash is considered anomalous (that is, there is some mismatch)

The process hash score of a workload is the minimum process hash score of all hashes observed in that workload, with 0 meaning there is a flagged or malicious process hash in the system, and 100 meaning there is no hash anomaly observed in the system.

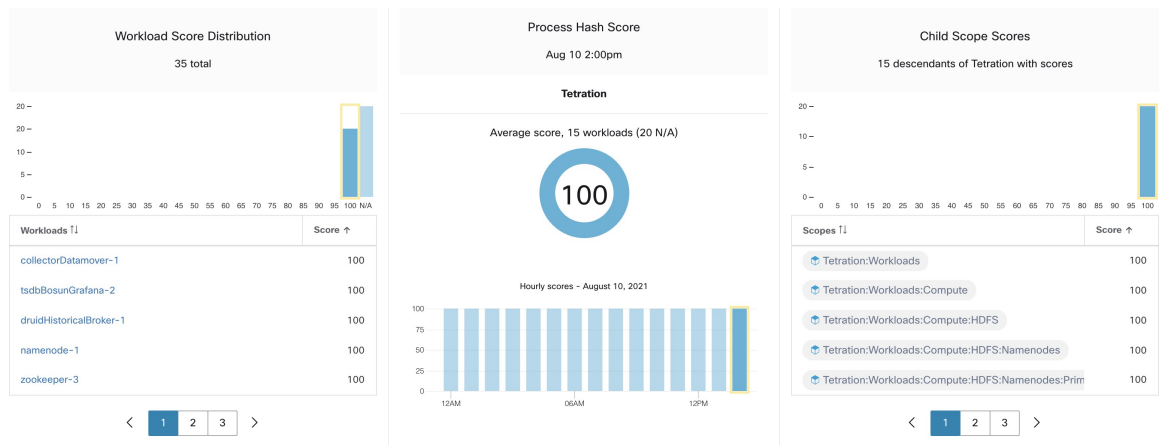
## How to Enable Process Hash Feature

Process hash feature is enabled by default on deep visibility agents and enforcement agents; no forensic config is needed. If there are such agents in your system, you should begin to see scores within 2 hours after the system starts.

## Where to See Process Hash Score

- **Security Dashboard:**

Figure 349: Process Hash Score Section in Security Dashboard



Process Hash Score section in [View Security Dashboard](#)

- **Workload Profile Page / File Hashes Tab:**

Figure 350: File Hashes Tab on Workload Profile Page

Observed in the last hour

File Hashes

| Benign                   | SHA1 Hash | SHA256 Hash | File Path                                                                                                  | Anomaly Score | Reason              | Links                            |
|--------------------------|-----------|-------------|------------------------------------------------------------------------------------------------------------|---------------|---------------------|----------------------------------|
| <input type="checkbox"/> | d9a44b4   | 7eedeeb     | /opt/tetration/e2e/test_framework/src/e2e/misc_tests/deadpool_tests/go_tools/fakemw/bin/fakemw_linux_amd64 | 0.00          | Flagged / Malicious | <a href="#">Inventory Search</a> |
| <input type="checkbox"/> | 36f9ca4   | 8b2e701     | /usr/bin/sigcheck                                                                                          | 0.00          | Flagged / Malicious | <a href="#">Inventory Search</a> |
| <input type="checkbox"/> | 07b6dd0   | 087b38b     | /local/tmp/legit_linux_amd64                                                                               | 58.33         | Anomalous           | <a href="#">Inventory Search</a> |

File Hashes tab in [Workload Profile](#)

## How the Process Hash Score is Calculated

For each process hash, we compute a score as follows:

1. If the hash is flagged or malicious,  $score = 0$
2. Else, if hash is benign,  $score = 100$
3. Else, if hash is an anomaly,  $score$  is in the range of  $[1, 99]$ , the higher the better.
4. Else,  $score = 100$

The logic for calculating score in (3) is that we first calculate the minority score of the hash (which is one minus the population ratio of that hash in workload population under the same rootscope), then map it to range  $[0.0, 1.0]$  using an information function  $-\log_2(x)$  if the minority score of the hash is above 0.5, then map the score again to a range  $[1.0, 99.0]$ . Let us take the above example of the Apache web server farm and consider the hash of `httpd`. Below are some scenarios:

- Suppose that `httpd` has two hash values ( $h_1$  and  $h_2$ ) across 1000 servers in the farm:  $h_1$  in 1 server,  $h_2$  in the rest 999 servers. In this case:
  - $population\_ratio(h_1) = 0.001$ ,  $population\_ratio(h_2) = 0.999$ . Then:
  - $minority\_score(h_1) = 0.999$ ,  $minority\_score(h_2) = 0.001$ . Then:
  - $score(h_1) = -\log_2(0.999) * 98 + 1 = 1.14$ ;
  - Since  $minority\_score(h_2) < 0.5$ ,  $h_2$  is not considered an anomaly, hence  $score(h_2) = 100$ .
- Suppose that `httpd` has two hash values ( $h_1$  and  $h_2$ ) across 10 servers in the farm:  $h_1$  in 1 server,  $h_2$  in the rest 9 servers. In this case:
  - $population\_ratio(h_1) = 0.1$ ,  $population\_ratio(h_2) = 0.9$ . Then:
  - $minority\_score(h_1) = 0.9$ ,  $minority\_score(h_2) = 0.1$ . Then:
  - $score(h_1) = -\log_2(0.9) * 98 + 1 = 15.90$ ;
  - Since  $minority\_score(h_2) < 0.5$ ,  $h_2$  is not considered an anomaly, hence  $score(h_2) = 100$ .
- Suppose that `httpd` has two hash values ( $h_1$  and  $h_2$ ) across 2 servers in the farm:  $h_1$  in one server,  $h_2$  in the other. In this case:
  - $population\_ratio(h_1) = population\_ratio(h_2) = 0.5$ . Then:
  - $minority\_score(h_1) = minority\_score(h_2) = 0.5$ . Then:
  - $score(h_1) = score(h_2) = -\log_2(0.5) * 98 + 1 = 99.0$ . This is the highest score for any hash that is considered an anomaly.
- Suppose that `httpd` has only one hash value ( $h_1$ ) across all servers. In this case,  $minority\_score(h_1) = 0.0 < 0.5$ ; hence it is not considered an anomaly, and  $score(h_1) = 100$ .

Finally, the process hash score of a workload is the minimum process hash score of all that hashes observed in that workload.

Additional information about the  $-\log_2(x)$  information function is found [here](#).



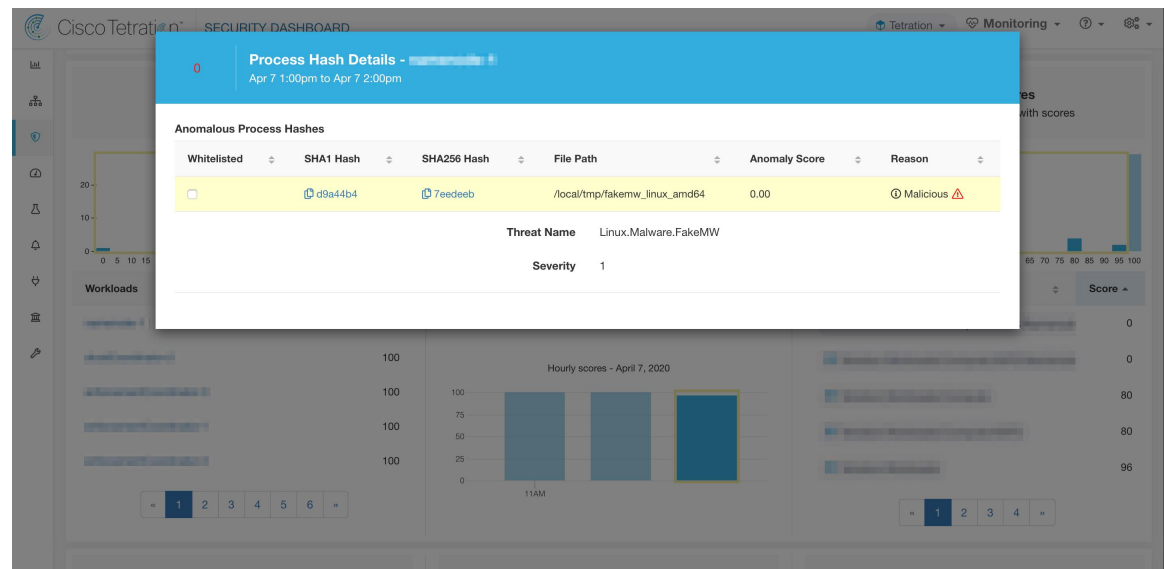
## How to Improve Process Hash Score

The process hash score of 0 on a workload means that a flagged or malicious process hash has shown up in that workload; preventing such process to run again improves the score. A positive process hash score less than 100 means that there is a process hash anomaly across your system; it is not malicious but worth a further investigation. After a careful investigation, if the hash is concluded to be safe, adding it to your Benign list will also improve the score. User can mark anomalous hashes as 'benign' by clicking on the Benign checkbox in the File Hashes / Process Hash Details page or by [User Uploaded Filehashes](#).

## Threat Info Details

As mentioned earlier, if Secure Workload the Hash Verdict service is enabled, any known malware hash when showing up would be flagged as malicious. In that case, more threat information of the malicious hash (gathered via our threat intelligence platform) will be provided. Currently the additional threat data include *threat name* and *severity*. Threat name is the name of the threat, while severity is a value from 1-5 to indicate how severe the threat is, where 1 means the least and 5 means the most severe.

**Figure 351: User Can Click on the Row of Malicious Hash to View Its Threat Info Details**



## Caveats

- Process hash analysis task is run once per hour, but it may take up to 2 hours for the expected scores/results to show in the security dashboard depending on the action. For examples:
  - If you upload your hash Flagged list and a process hash in that list shows up, it may take up to 1 hour for the score to be reflected in the security dashboard.
  - If you remove a hash from your Flagged list, it may take up to 2 hours for it to be cleared and the score is reflected in the security dashboard.
- Retention:
  - Detailed results from process hash analysis are kept for at least 7 days.

- File Hashes tab in Workload Profile page only shows process hash details analyzed in the last hour.
- Previous versions of deep visibility and enforcement agents, and AnyConnect endpoints only report SHA256 hash values. Thus, matching against SHA1 hash Flagged/Benign list is not supported for those agents.
- Process hash score is calculated regarding a particular rootscope. If a workload belongs to multiple rootscopes, the process hash score of that workload is the minimum score across all rootscopes that it belongs to.
- To further reduce the false alarms in process hash anomaly analysis, we also mark all Secure Workload agent binaries as benign according to their file paths. This mechanism happens only when these hashes do not appear in any user-defined hash list, or are not flagged by Secure Workload Hash Verdict service.



## CHAPTER 8

# Network Flows - Traffic Visibility

---

On the Secure Workload UI, from the navigation pane, choose **Investigate** > **Traffic** that takes you to the Flow Search page. This page provides the means for quickly filtering and drilling down into the flows corpus. The basic unit is **Flow Observation**, which is a per-minute aggregation of each unique flow. The two sides of the flow are called **Consumer** and **Provider**, the consumer initiates the flow, and the provider responds to the consumer (for example **Client** and **Server** respectively). Each observation tracks the number of packets, bytes, and other metrics in each direction for that flow for that minute interval. In addition to quickly filtering, the flows can be explored visually with **Explore Observations**. The resulting list of flows observations can be clicked to view details of that flow, including latency, packets, and bytes over the lifetime of that flow.



---

**Warning** For hosts instrumented with Deep Visibility Agents or Enforcement Agents, Secure Workload is able to correlate flow data against the process that provides or consumes the flow. As a result, full command-line arguments, which may include **sensitive information such as database or API credentials**, used to launch the process are available for analysis and display.

---

Figure 352: Flows Overview

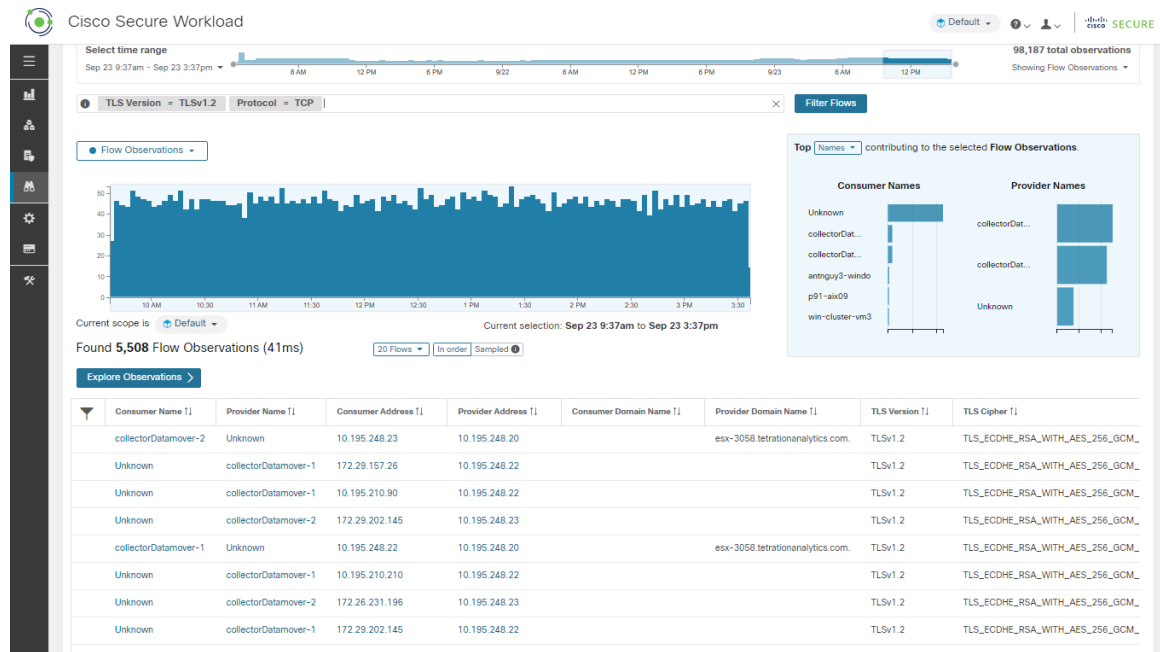


Table 32: Feature Information

| Feature Name                                                    | Release     | Feature Description                                                                                                                                                                                                                                              | Where to Find                                                                  |
|-----------------------------------------------------------------|-------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------|
| Visibility and Enforcement of Well-known IPv4 Malicious Traffic | 3.9 Patch 2 | You can now identify any traffic to and from the workloads to well-known malicious IPv4 addresses. You can also create policies to block any traffic to these malicious IPs using a pre-defined read-only inventory filter titled <b>Malicious inventories</b> . | <a href="#">Visibility of Well-Known Malicious IPv4 Addresses, on page 645</a> |

- [Corpus Selector, on page 627](#)
- [Columns and Filters, on page 627](#)
- [Filtered Time series, on page 633](#)
- [Top N Charts, on page 634](#)
- [Observations List, on page 635](#)
- [Explore Observations, on page 637](#)
- [Client-Server Classification, on page 639](#)
- [Conversation Mode, on page 642](#)
- [Visibility in Proxied Flows, on page 643](#)
- [Visibility of Well-Known Malicious IPv4 Addresses, on page 645](#)

# Corpus Selector

**Figure 353: Corpus Selector**



This is the unfiltered summary timeseries data for the current **Scope** for the entire corpus. The purpose of this component is to allow you to know what date range is being viewed, and easily change that date range by dragging within the component. The data in the chart is there in case it's useful for deciding which time range to select. You can select different metrics to be shown, by default the count of **flow observations** is shown.

The Corpus Selector can currently support selecting up to *approximately 2 billion flow observations*.

# Columns and Filters

**Figure 354: Filter Input**

This is where you define filters to narrow-down the search results. Click the (?) icon next to the word **Filters** for all possible dimensions. For any User Labels data, those columns will also be available for the appropriate intervals. This input also supports **and**, **or**, **not**, and **parenthesis** keywords, use these to express more complex filters. For example, a direction-agnostic filter between IP *1.1.1.1* and *2.2.2.2* can be written:

*Consumer Address = 1.1.1.1 and Provider Address = 2.2.2.2 or Consumer Address = 2.2.2.2 and Provider Address = 1.1.1.1*

And to additionally filter on Protocol = TCP:

*(Consumer Address = 1.1.1.1 and Provider Address = 2.2.2.2 or Consumer Address = 2.2.2.2 and Provider Address = 1.1.1.1) and Protocol = TCP*

The filter input also supports “,” and “-” for Port, Consumer Address and Provider Address, by translating “-” into range queries. The following are examples of a valid filter:

Figure 355: Filter Input Supports for Consumer Address

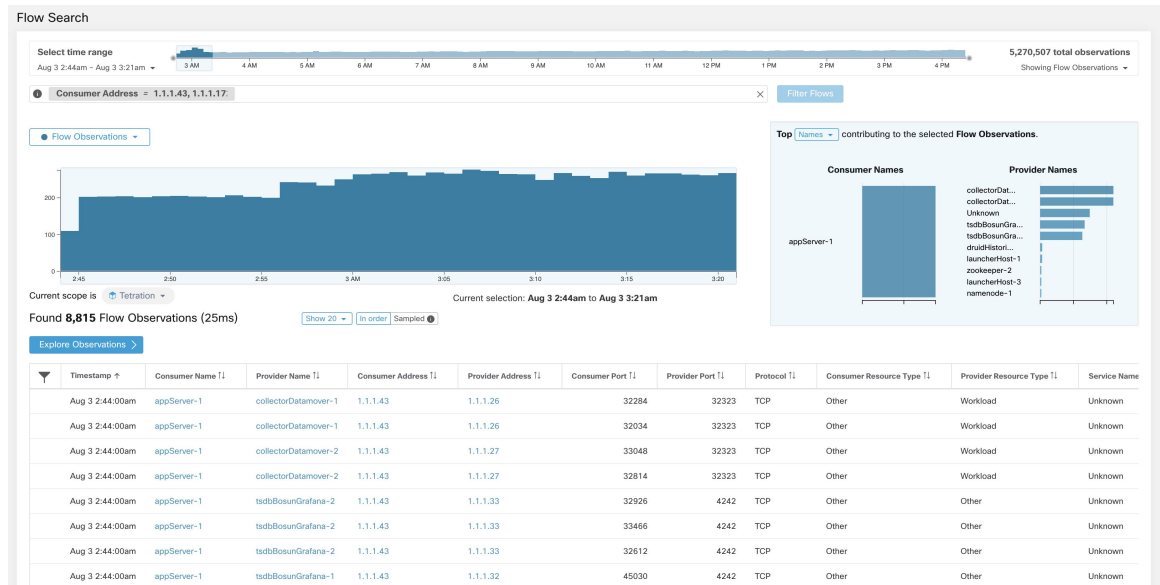


Figure 356: Filter Input Supports Range Query for Consumer Address

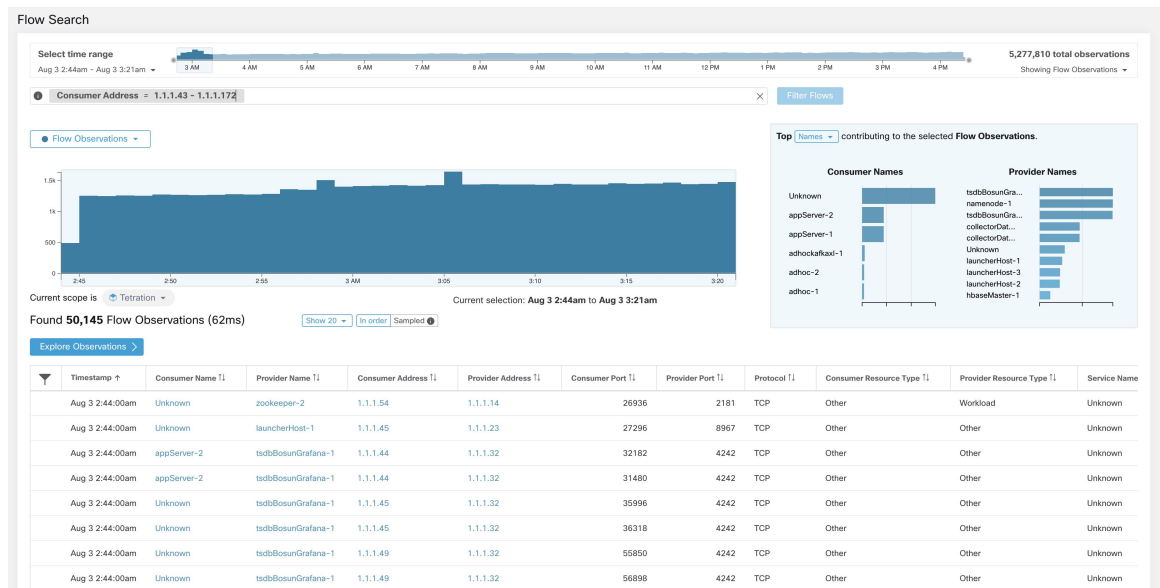


Table 33: Available Columns and Filters

| Columns (Names exposed in API)          | Description                                                                                                                                                                      | Source                                |
|-----------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------|
| Consumer Address ( <i>src_address</i> ) | Enter a subnet or IP Address using CIDR notation (for example, 10.11.12.0/24). Matches flow observations whose consumer address overlaps with the provided IP Address or subnet. | Software Agents and Ingest Appliances |

| Columns (Names exposed in API)                   | Description                                                                                                                                                                     | Source                                   |
|--------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------|
| <b>Provider Address</b> ( <i>dst_address</i> )   | Enter a subnet or IP Address using CIDR notation (for example, 10.11.12.0/24) Matches flow observations whose provider address overlaps with the provided IP address or subnet. | Software Agents and Ingest Appliances    |
| <b>Consumer Name</b>                             | Matches flow observations whose consumer workload name overlaps with the entered consumer workload name.                                                                        | Software Agents and AnyConnect Connector |
| <b>Provider Name</b>                             | Matches flow observations whose provider workload name overlaps with the entered provider workload name.                                                                        | Software Agents and AnyConnect Connector |
| <b>Consumer User</b>                             | Matches flow observations whose consumer name overlaps with the entered consumer name who generated the flow.                                                                   | Software Agents and AnyConnect Connector |
| <b>Provider User</b>                             | Matches flow observations whose provider name overlaps with the entered provider name who handled the flow.                                                                     | Software Agents and AnyConnect Connector |
| <b>Consumer Domain Name</b>                      | Matches flow observations whose consumer domain name (associated with the consumer IP address or subnet) overlaps with the entered consumer domain name.                        | Software Agents and AnyConnect Connector |
| <b>Provider Domain Name</b>                      | Matches flow observations whose provider domain name (associated with the provider IP address/subnet) overlaps with the entered provider domain name.                           | Software Agents and AnyConnect Connector |
| <b>Consumer Hostname</b> ( <i>src_hostname</i> ) | Matches flows whose consumer hostname overlaps with the provided hostname.                                                                                                      | Software Agents and AnyConnect Connector |
| <b>Provider Hostname</b> ( <i>dst_hostname</i> ) | Matches flows whose provider hostname overlaps with the provided hostname.                                                                                                      | Software Agents and AnyConnect Connector |
| <b>Consumer Malicious</b>                        | If the value is <b>true</b> , the IP address of the consumer is known to be malicious.                                                                                          | Internal                                 |
| <b>Provider Malicious</b>                        | If the value is <b>true</b> , the IP address of the provider is known to be malicious.                                                                                          | Internal                                 |

| Columns (Names exposed in API)                                                                  | Description                                                                                                                                       | Source                               |
|-------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------|
| <b>Consumer Enforcement Group</b><br>( <i>src_enforcement_epg_name</i> )                        | The Consumer Enforcement Group is the name of the filter (Scope, Inventory Filter or Cluster) in the enforced policies that matches the consumer. | Internal                             |
| <b>Provider Enforcement Group</b><br>( <i>dst_enforcement_epg_name</i> )                        | The Provider Enforcement Group is the name of the filter (Scope, Inventory Filter or Cluster) in the enforced policies that matches the provider. | Internal                             |
| <b>Consumer Analysis Group</b>                                                                  | The Consumer Analysis Group is the name of the filter (Scope, Inventory Filter, or Cluster) in the analyzed policies that matches the consumer.   | Internal                             |
| <b>Provider Analysis Group</b>                                                                  | The Provider Analysis Group is the name of the filter (Scope, Inventory Filter or Cluster) in the analyzed policies that matches the provider.    | Internal                             |
| <b>Consumer Scope</b> ( <i>src_scope_name</i> )                                                 | Matches flows whose consumer belongs to the specified Scope.                                                                                      | Internal                             |
| <b>Provider Scope</b> ( <i>dst_scope_name</i> )                                                 | Matches flows whose provider belongs to the specified Scope.                                                                                      | Internal                             |
| <b>Consumer Port</b> ( <i>src_port</i> )                                                        | Matches flows whose Consumer port overlaps with the provided port.                                                                                | Software Agents, ERSPAN, and NetFlow |
| <b>Provider Port</b> ( <i>dst_port</i> )                                                        | Matches flows whose Provider port overlaps with the provided port.                                                                                | Software Agents, ERSPAN, and NetFlow |
| <b>Consumer Country</b> ( <i>src_country</i> )                                                  | Matches flows whose Consumer country overlaps with the provided country.                                                                          | Internal                             |
| <b>Provider Country</b> ( <i>dst_country</i> )                                                  | Matches flows whose Provider country overlaps with the provided country.                                                                          | Internal                             |
| <b>Consumer Subdivision</b><br>( <i>src_subdivision</i> )                                       | Matches flows whose Consumer subdivision overlaps with the provided subdivision (state).                                                          | Internal                             |
| <b>Provider Subdivision</b><br>( <i>dst_subdivision</i> )                                       | Matches flows whose Provider subdivision overlaps with the provided subdivision (state).                                                          | Internal                             |
| <b>Consumer Autonomous System Organization</b><br>( <i>src_autonomous_system_organization</i> ) | Matches flows whose Consumer autonomous system organization overlaps with provided autonomous system organization (ASO).                          | Internal                             |
| <b>Provider Autonomous System Organization</b><br>( <i>dst_autonomous_system_organization</i> ) | Matches flows whose Provider autonomous system organization overlaps with provided autonomous system organization (ASO).                          | Internal                             |



| Columns (Names exposed in API)                         | Description                                                                                                                                                 | Source                                |
|--------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------|
| <b>Protocol</b> ( <i>proto</i> )                       | Filter flow observations by Protocol type (TCP, UDP, ICMP).                                                                                                 | Software Agents and Ingest Appliances |
| <b>Address Type</b> ( <i>key_type</i> )                | Filter flow observations by Address type (IPv4, IPv6, DHCPv4).                                                                                              | Software Agents and Ingest Appliances |
| <b>Fwd TCP Flags</b>                                   | Filter flow observations by flags (SYN, ACK, ECHO).                                                                                                         | Software Agents, ERSPAN, and NetFlow  |
| <b>Rev TCP Flags</b>                                   | Filter flow observations by flags (SYN, ACK, ECHO).                                                                                                         | Software Agents, ERSPAN, and NetFlow  |
| <b>Fwd Process UID</b><br>( <i>fwd_process_owner</i> ) | Filter flow observations by process owner UID (root, admin, yarn, mapred).                                                                                  | Software Agents                       |
| <b>Rev Process UID</b><br>( <i>rev_process_owner</i> ) | Filter flow observations by process owner UID (root, admin, yarn, mapred).                                                                                  | Software Agents                       |
| <b>Fwd Process</b> ( <i>fwd_process_string</i> )       | Filter flow observations by process (java, hadoop, nginx). See <a href="#">Process String Visibility Warning</a>                                            | Software Agents                       |
| <b>Rev Process</b> ( <i>rev_process_string</i> )       | Filter flow observations by process (java, hadoop, nginx). See <a href="#">Process String Visibility Warning</a>                                            | Software Agents                       |
| <b>Consumer In Collection Rules?</b>                   | Match only internal Consumers.                                                                                                                              | Internal                              |
| <b>Provider In Collection Rules?</b>                   | Match only internal Providers.                                                                                                                              | Internal                              |
| <b>SRTT Available</b>                                  | Matches flows which have SRTT measurements available using the values 'true' or 'false'. (This is equivalent to SRTT > 0).                                  | Internal                              |
| <b>Bytes</b>                                           | Filter flow observations by Byte traffic bucket. Matches flows which Byte traffic bucket values are =, <, > (bucketed by powers of 2 (0, 2, 64, 1024)).     | Software Agent and Ingest Appliances  |
| <b>Packets</b>                                         | Filter flow observations by Packet traffic bucket. Matches flows which Packet traffic bucket values are =, <, > (bucketed by powers of 2 (0, 2, 64, 1024)). | Software Agent and Ingest Appliances  |

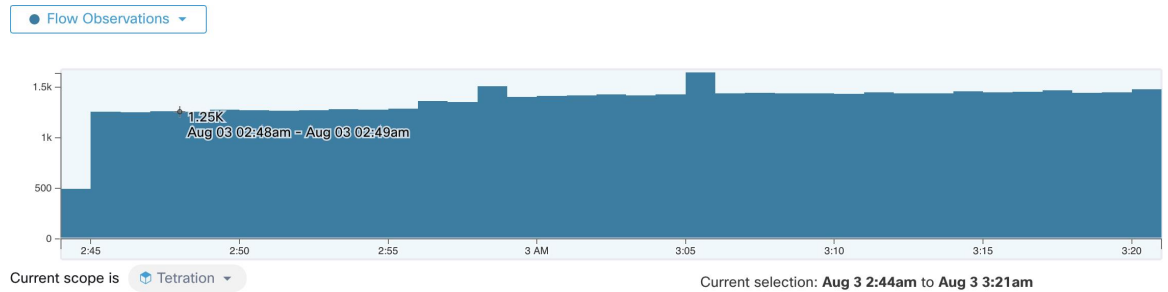
| Columns (Names exposed in API)                                             | Description                                                                                                                                                                 | Source         |
|----------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------|
| <b>Flow Duration (µs)</b>                                                  | Filter flow observations by Flow Duration bucket. Matches flows which Flow Duration bucket values are =, <, > (bucketed by powers of 2 (0, 2, 64, 1024)).                   | Internal       |
| <b>Data Duration (µs)</b>                                                  | Filter flow observations by Data Duration bucket. Matches flows which Data Duration bucket values are =, <, > (bucketed by powers of 2 (0, 2, 64, 1024)).                   | Internal       |
| <b>SRTT (µs)</b> ( <i>srtt_dim_usec</i> )                                  | Filter flow observations by SRTT bucket. Matches flows which SRTT bucket values are =, <, > (bucketed by powers of 2 (0, 2, 64, 1024)).                                     | Software Agent |
| <b>Fwd Packet Retransmissions</b><br>( <i>fwd_tcp_pkts_retransmitted</i> ) | Filter flow observations by Packet Retransmissions bucket. Matches flows which Packet Retransmissions bucket values are =, <, > (bucketed by powers of 2 (0, 2, 64, 1024)). | Software Agent |
| <b>Rev Packet Retransmissions</b><br>( <i>rev_tcp_pkts_retransmitted</i> ) | Filter flow observations by Packet Retransmissions bucket. Matches flows which Packet Retransmissions bucket values are =, <, > (bucketed by powers of 2 (0, 2, 64, 1024)). | Software Agent |
| <b>User Labels</b> (* or <i>user_</i> prefix)                              | User-defined data that is associated to the manually uploaded custom labels that are prefixed with * in the UI and <i>user_</i> in OpenAPI.                                 | CMDB           |
| <b>TLS Version</b>                                                         | SSL protocol version used in the flow.                                                                                                                                      | Software Agent |
| <b>TLS Cipher</b>                                                          | Algorithm type used by the SSL protocol in the flow.                                                                                                                        | Software Agent |
| <b>Consumer Agent Type</b>                                                 | Specify the consumer agent type.                                                                                                                                            | Internal       |
| <b>Provider Agent Type</b>                                                 | Specify the provider agent type.                                                                                                                                            | Internal       |
| <b>Consumer Resource Type</b>                                              | Represents the flow of resources from a source to a consumer. It can be either workload, pods, services, or others                                                          | Internal       |
| <b>Provider Resource Type</b>                                              | Represents the flow of resources from a provider to a consumer. . It can be either workload, pods, services, or others.                                                     | Internal       |



**Note** Because flow data is labeled with User Labels only at ingestion time, User Labels will not appear immediately after enabling them. It may take a few minutes before the labels start appearing in Flow Search. Also, the available User Labels will be different depending on which part of the **Corpus Selector** you have selected, since the enabled Labels might have been changed at various times.

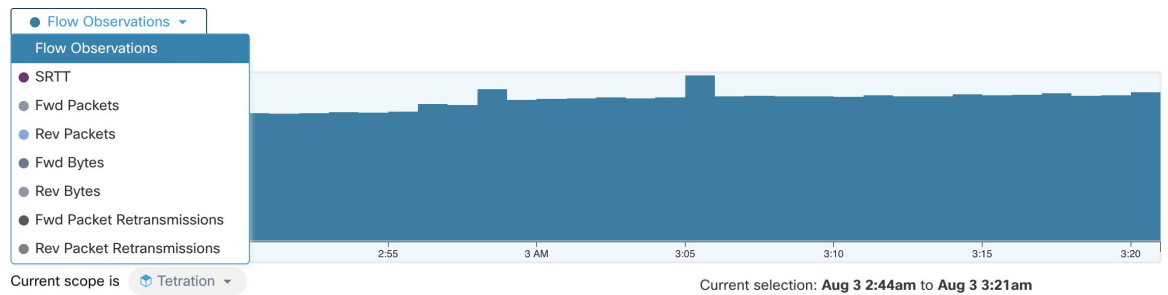
# Filtered Time series

**Figure 357: Filtered Time series**



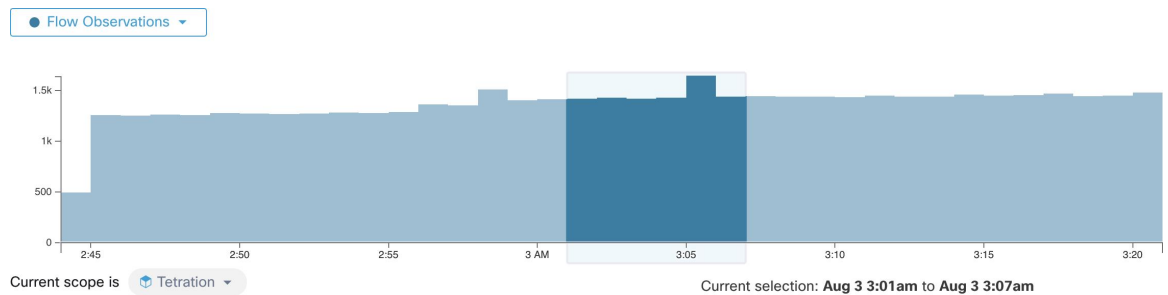
This component displays the aggregated totals of various metrics for the interval selected (the selection made in the above [Corpus Selector](#), on page 627). Use the dropdown to change which metric is displayed.

**Figure 358: Time series dropdown**



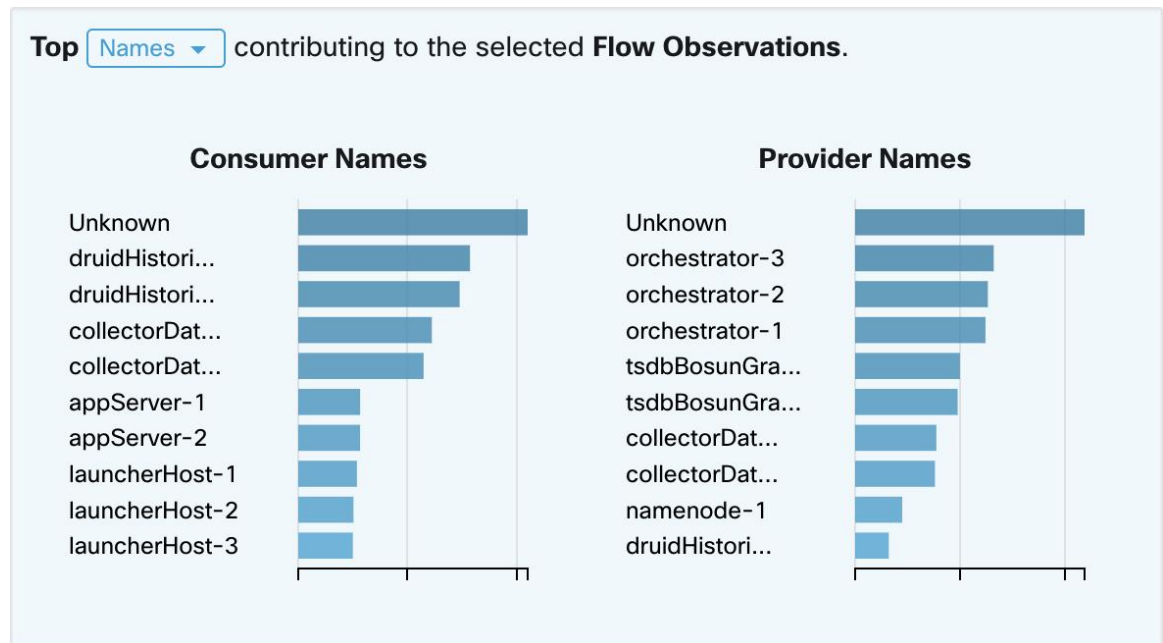
Further-narrowing of the selected interval can also be done in this component. Click the area of the chart that you'd like to focus on, and the Top N Charts and the data below will all be updated to include only data from that selected interval.

**Figure 359: Time series with selection**



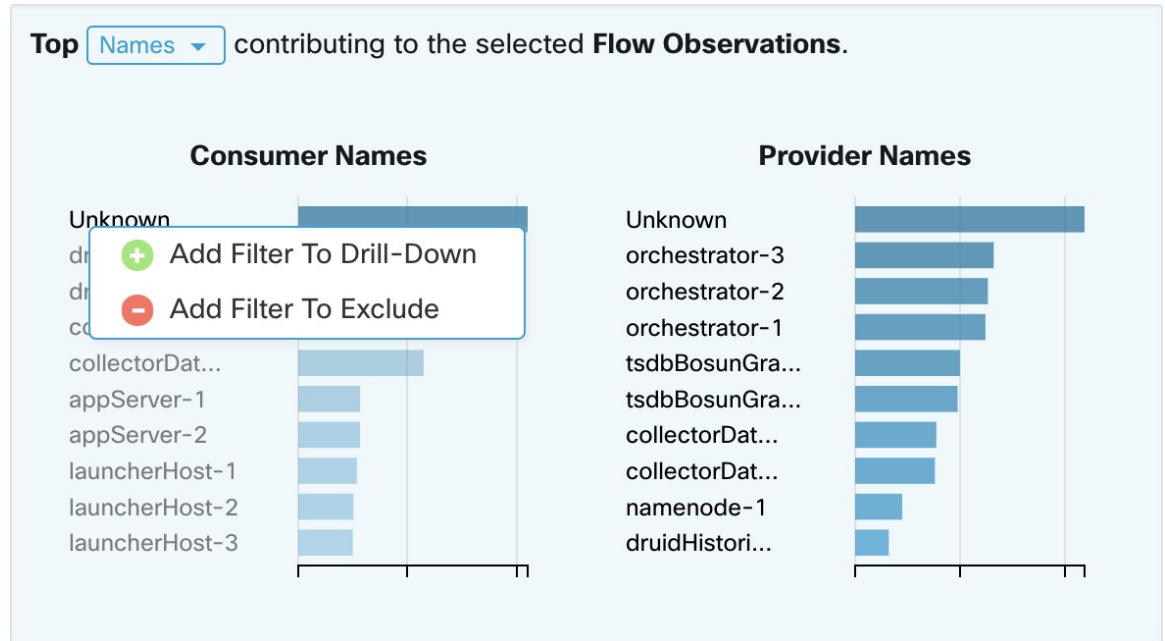
# Top N Charts

Figure 360: Top N Charts



The charts display the Top N values that contribute to the selection in the Filtered Time series chart to the left. Selecting a peak in Flow Observations in the time series chart, and hostnames in the Top N charts, displays the list of hostnames (Consumer and Provider) that contribute the most to those flow observations. Also, if the time series chart is set to display SRTT, then the Top Hostnames display those that contribute most to that selected SRTT.

Figure 361: Drill-down/Exclude



Click on any of the items in the Top N charts to display a menu that allows you to either **Drill-Down** or **Exclude** that value.

- Click **Drill-Down** to add a filter that confines the results to just that value.
- Click **Exclude** to add a filter that excludes that value from the results.



**Note** After clicking **Drill-Down** or **Exclude**, the **Filter** icon must be pressed for the filter to take effect. This is so that multiple **Exclude** actions can be taken quickly without having the page repeatedly update in the middle.

## Observations List

Found 5,917 Flow Observations (19ms) Show 20 In order Sampled

[Explore Observations](#)

| Timestamp       | Consumer Name        | Provider Name | Consumer Address | Provider Address | Consumer Port | Provider Port | Protocol | Consumer Resource Type | Provider Resource Type | Service Name |
|-----------------|----------------------|---------------|------------------|------------------|---------------|---------------|----------|------------------------|------------------------|--------------|
| Aug 3 9:12:00am | collectorDatamover-2 | Unknown       | 172.21.156.183   | 172.21.156.129   | 0             | 0             | ICMP     | Workload               | Other                  | Unknown      |
| Aug 3 9:12:00am | collectorDatamover-2 | appServer-2   | 172.21.156.183   | 172.21.156.180   | 60674         | 443           | TCP      | Workload               | Workload               | HTTPS        |
| Aug 3 9:12:00am | collectorDatamover-1 | appServer-2   | 172.21.156.182   | 172.21.156.180   | 38290         | 443           | TCP      | Workload               | Workload               | HTTPS        |
| Aug 3 9:12:00am | collectorDatamover-1 | Unknown       | 172.21.156.182   | 172.21.156.129   | 0             | 0             | ICMP     | Workload               | Other                  | Unknown      |
| Aug 3 9:12:00am | collectorDatamover-1 | appServer-2   | 172.21.156.182   | 172.21.156.180   | 39048         | 443           | TCP      | Workload               | Workload               | HTTPS        |
| Aug 3 9:12:00am | collectorDatamover-2 | appServer-2   | 172.21.156.183   | 172.21.156.180   | 60678         | 443           | TCP      | Workload               | Workload               | HTTPS        |

This is the list of actual **Flow Observations** that matches the filters and selections in the page above. By default, 20 will be loaded starting from the beginning of the interval. It's possible to increase the number that are loaded by using the dropdown. It's also possible to load a random set of flow observations from the selected interval by using **Sampled** rather than **In order**. The **Sampled** setting is useful for getting a more representative

set of flow observations from the selected interval rather than loading them sequentially from the beginning of the interval.

Figure 362: Sampled

Found 5,917 Flow Observations (95ms) Show 20 In order Sampled

[Explore Observations](#)

| Timestamp        | Consumer Name        | Provider Name | Consumer Address | Provider Address | Consumer Port | Provider Port | Protocol | Consumer Resource Type | Provider Resource Type | Service Name |
|------------------|----------------------|---------------|------------------|------------------|---------------|---------------|----------|------------------------|------------------------|--------------|
| Aug 3 9:22:00am  | collectorDatamover-2 | Unknown       | 172.21.156.183   | 172.21.106.115   | 56800         | 53            | UDP      | Workload               | Other                  | DNS          |
| Aug 3 10:04:00am | collectorDatamover-2 | appServer-2   | 172.21.156.183   | 172.21.156.180   | 43882         | 443           | TCP      | Workload               | Workload               | HTTPS        |
| Aug 3 10:12:00am | collectorDatamover-1 | Unknown       | 172.21.156.182   | 171.68.38.66     | 123           | 123           | UDP      | Workload               | Other                  | NTP          |
| Aug 3 10:16:00am | collectorDatamover-2 | Unknown       | 172.21.156.183   | 172.21.156.129   | 0             | 0             | ICMP     | Workload               | Other                  | Unknown      |
| Aug 3 10:25:00am | collectorDatamover-2 | appServer-2   | 172.21.156.183   | 172.21.156.180   | 53512         | 443           | TCP      | Workload               | Workload               | HTTPS        |
| Aug 3 10:40:00am | collectorDatamover-2 | Unknown       | 172.21.156.183   | 172.21.106.115   | 14212         | 53            | UDP      | Workload               | Other                  | DNS          |

## Flow Details

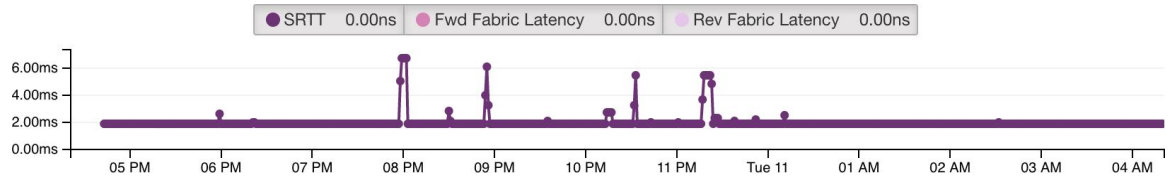
Click on any of the rows to expand the **Flow Details** section. This displays a summary of the flow and charts for various metrics for the lifetime of that flow. For long-lived flows, a summary chart is displayed at the bottom that allows you to choose different intervals for which to view time series data.

Figure 363: Flow details



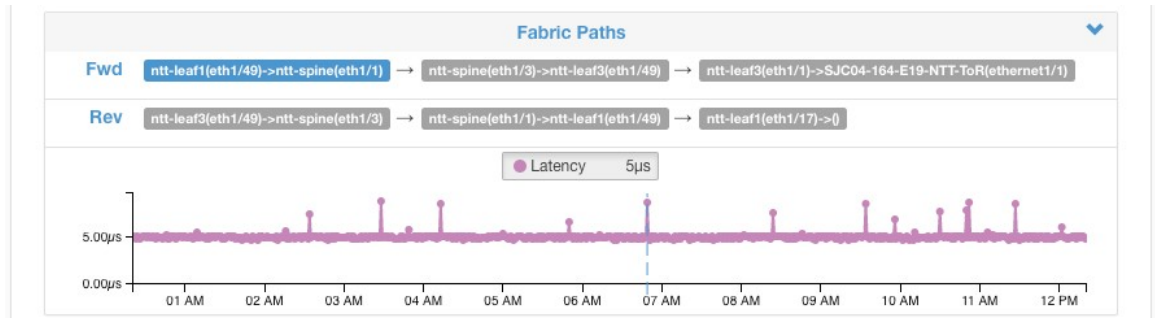
For flows labeled with Fabric Path information, **Fwd/Rev Fabric Latency** and **SRTT** are available. Time series charts for other metrics such as **Fwd/Rev Burst Indicators** and **Fwd/Rev Burst+drop Indicators** may be displayed if available. See [Compatibility](#).

Figure 364: Latency



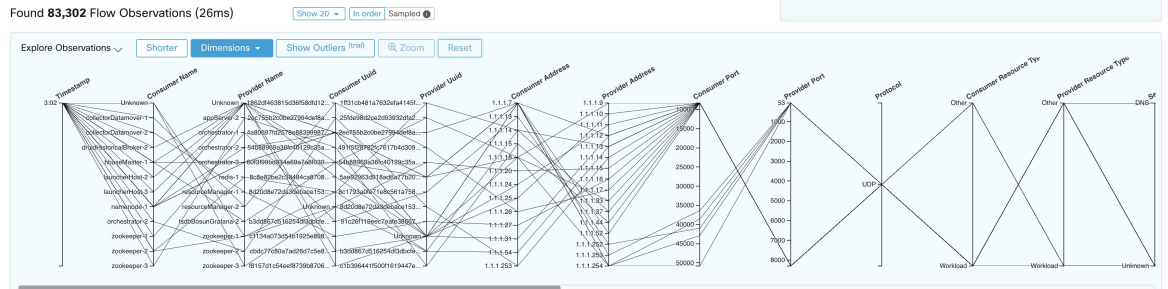
In addition, details about the **Fwd/Rev Fabric Path** are available. Each link is clickable, toggling **Latency** and **Drop Indicators** time series charts (when none-zero). Click **Fwd** or **Rev** to navigate to the Fabric Path Overlay page drill-down for the flow.

Figure 365: Fabric paths



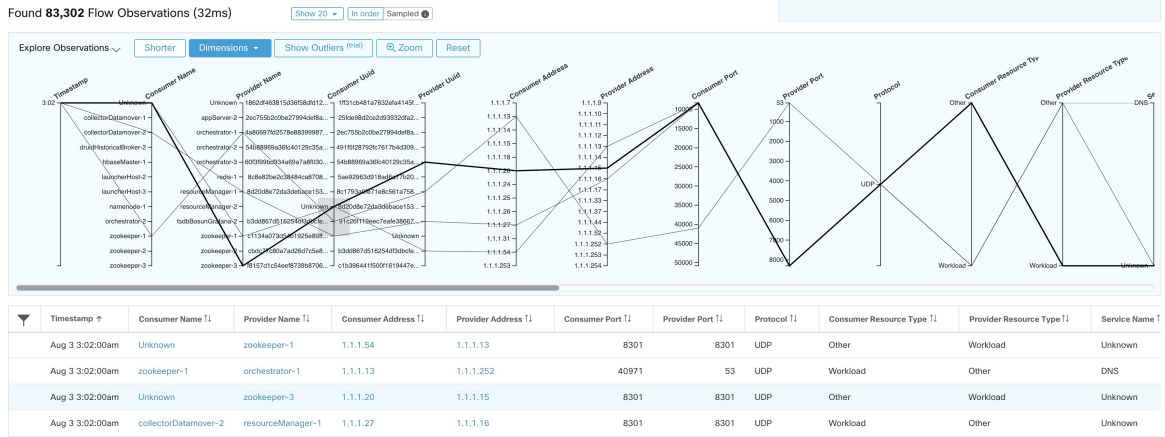
# Explore Observations

Figure 366: Explore Observations



Click **Explore Observations** to enable a chart view that allows quick exploration of the high-dimensional data (**Parallel Coordinates** chart). A bit overwhelming at first, this chart is useful when enabling only the dimensions you're interested in (by unchecking items in the **Dimensions** dropdown), and when rearranging the order of the dimensions. A single line in this chart represents a single observation, and where that line intersects with the various axes indicates the value of that observation for that dimension. This can become clearer when hovering over the list of observations below the chart to see the highlighted line representing that observation in the chart:

Figure 367: Flow Observation hovered



Due to the high-dimensional nature of the flow data, this chart is wide by default, and requires scrolling to the right to see the entire chart. For this reason, it's useful to disable all but the dimensions you're interested in.

### Sampling vs. In-Order

It's recommended that Explore Observations be done with **sampling** enabled, and with a larger number of flows. This allows you to see more of the variety of flows that comprise the selected interval. So, if you've selected 2 million flow observations in the time series chart above, loading a sample of 1000 will take uniformly from throughout the interval, whereas loading flows **In-order** will load the first 1000 flow observations from the very beginning of the interval:

Figure 368: 1000 In-order

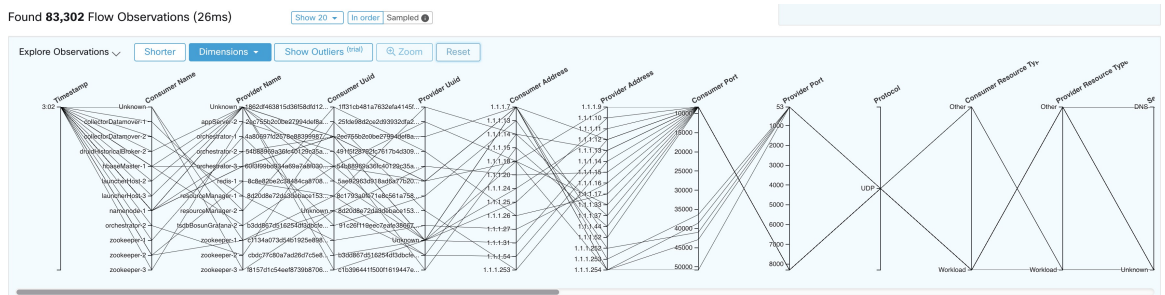
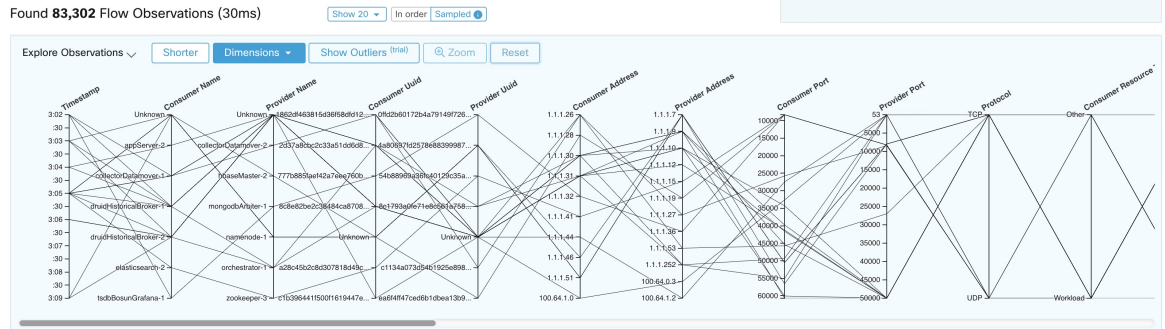




Figure 369: vs. 1000 sampled

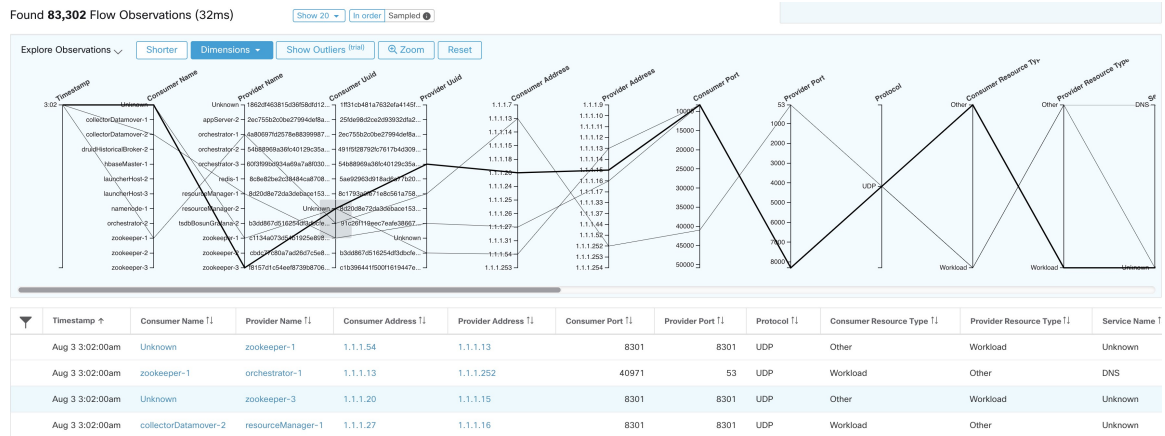


Notice how the **Timestamp** for all of the in-order observations is from 9:09 and how the observations are evenly distributed through the selected interval in the sampled version.

**Filtering**

Dragging the cursor along any of the axes create a selection that shows only observations that match that selection. Click again on the axis to remove the selection at any time. Selections can be made on any number of axes at a time. The list of observations updates to show only the selected observations:

Figure 370: Explore with selection



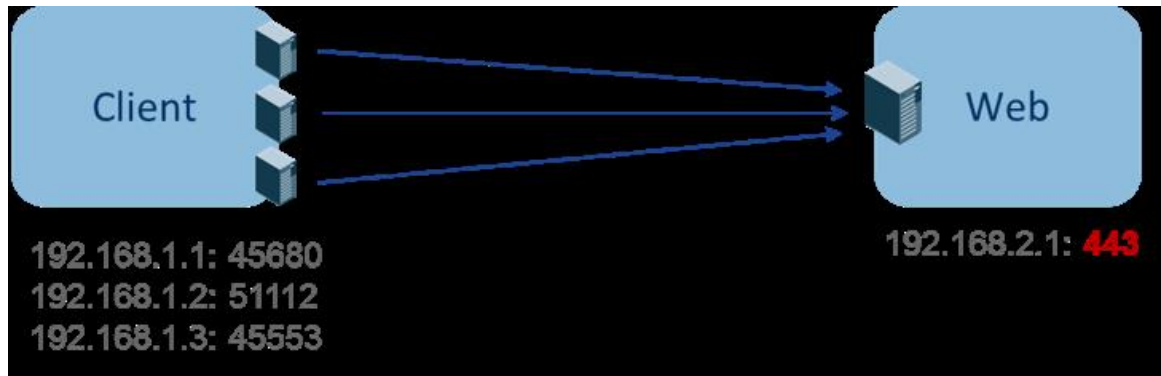
Download table data as JSON

# Client-Server Classification

Flow direction (client/server or provider/consumer classification) is important for visibility, automatic policy discovery, and enforcement. Every unicast flow has a client and a server classification.

For example, if there are clients (192.168.1.1-192.168.1.3) accessing a web server (192.168.2.1) using https, typically the source port is an ephemeral port in the range 1025-65535 and the destination port is 443.

Figure 371: Client-Server Classification



The accurate client-server direction is:

- Client: 192.168.1.1-3
- Server: 192.168.2.1
- Services: TCP port 443

Policies generated by automatic policy discovery are shown in the figure (with left endpoints grouped):

Figure 372: Policies Generated



Now, if the client - server direction decision is reversed (an inaccurate classification), that is:

- Client: 192.168.2.1
- Server: 192.168.1.1-3
- Services: the list of ephemeral ports (45680, 51112, 45553)

Then, in the above inaccurate classification, the policies generated may be as shown in the figure:

Figure 373: Inaccurate classification



This consumes more resources in terms of policy enforcement. In addition, depending on how you enforce the policy, even though 192.168.1.1-3 uses these ephemeral ports, they can't access 192.168.2.1. For example, if you use Secure Workload software sensor enforcement, the enforcement policy for Client to Web above (ESTAB) doesn't match with traffic generated by Client destined to Web (NEW, ESTAB).

Timestamps and TCP flags are used in Secure Workload to determine the client-server direction. If there are no TCP flags information (SYN, SYN/ACK) because, for example, the packets could be UDP/ICMP or an HW sensor is used that doesn't support direction signals, then user-defined override rules, timestamps, and other heuristics are used to infer the flow direction. Heuristics by definition don't guarantee 100% accuracy. Client-server accuracy is a function of the type of sensor used and the conditions in which sensors are used. You can use Secure Workload's REST-API (OpenAPI) to insert client-server override rules to identify the server ports for those flow types that Secure Workload gets the direction wrong. Then allow Secure Workload to process new flow data captured with those rules in place, and then generate the policies over the time duration when the flow direction were fixed. For more details on the API to specify override rules, refer to: [Client Server configuration, on page 854](#). You can also manually define the policies and examine/remove the undesired policies. See [Policies, on page 756](#).

## Sensor Type Recommendation

Deep visibility or enforcement software agents provide the best signals to Secure Workload client-server classification algorithms. It is encouraged to consider deploying deep visibility or enforcement agents. These agents get all the necessary signals to drive the correct client-server classification. If deployment of deep visibility or enforcement agents is not possible for few workloads, it is recommended to use ERSPAN sensors and stopping there for automatic policy discovery. Secure Workload assists as best as it can and we're continuously improving our heuristics algorithms based on feedback.

When the correct client-server direction information isn't available, Secure Workload uses user-defined overrides or heuristics to infer what the direction may have been. Heuristics by definition don't guarantee 100% accuracy. The accuracy drops with the type of sensor that is used and the condition in which it was used.

The following is the recommended order for client-server decision for policy generation use cases:

- **Deep visibility or enforcement agents:** For best results, use Software Sensors (Deep visibility or Enforcement agents). Traffic flows started before the sensor was started would be processed by heuristics that are discussed below.
- **ADC Sensors like F5/Citrix/. . . agents:** These agents gather the client-server state from the ADC devices and stream that source of truth to Secure Workload.
- **ERSPAN sensors :** With an ERSPAN sensor, user needs to take care of providing full visibility of the traffic to and from the workload in question, and make sure the ERSPAN sensor sees all the spanned traffic. The ERSPAN sensor must also not be over subscribed, so that its visibility is not impaired of the network communication of the workload. Furthermore, user must ensure that packet drops for ERSPAN sensors are kept to the minimum. The operator will not see process information with the network flow information for automatic policy discovery.

While using Netflow sensor listed below, user has to sign up for lot more manual work on policy analysis and generate exception rules. Secure Workload uses extensive use of heuristics, which by definition, aren't 100% accurate.

- **Netflow Sensor :** NetFlow provides sampled and aggregated flow data. The aggregation and sampling processes lose client-server direction information. This impacts automatic policy discovery and policy generation results and makes the problem harder. NetFlow data is excellent for high-level visibility.

Secure Workload has to fall back to heuristics, which sometimes, if incorrect, requires more manual work on behalf of the operator – like defining exception rules for Secure Workload. NetFlow data also misses some of the short flows and the signal quality depends on the device producing NetFlow data. We recommend using NetFlow with Secure Workload for specialized use cases like stitching flows through L3/L4 NAT devices like Application Delivery Controllers (or Server Load Balancers) to provide Secure Workload visibility into which flow is related to which other flow.

More details of the client- server direction analysis follow.

## Identifying Producers (aka Servers) and Consumers (aka Clients) for a flow

There are multiple ways (often heuristics) to detect servers:

- If a sensor sees the SYN handshake, it can figure out who the server is.
- Based on time - the initiator of a connection is deemed client.
- Degree model - a server typically has many clients talking to it. In contrast, the degree for client port is expected to be far less.

The priority order is SYN\_ANALYSIS/NETSTAT > USER\_CONFIG > DEGREE\_MODEL.

The thinking behind giving SYN\_ANALYSIS higher priority over user config is that config can get stale, and that sensor has the best vantage point to establish ground truth. DEGREE\_MODEL is where learning/heuristics come into play, and the accuracy can't be 100% guaranteed.

It's possible that our heuristics for client-server detection can go wrong, despite our best intentions and continuous algorithmic refinements that we make in this area. For those scenarios, the OpenAPI interface can be used to punch well-known server ports. These configs aren't applied to past flows, and only affect markings on flows from that point on (that is, going forward). It's intended as a last resort fallback, rather than the normal modus operandi.

We also recommend not to keep flipping the client- server marking for the full duration of a given flow (even if we get it wrong, and when our internal models have changed - which they do over time, as more flow patterns are observed/analyzed). Higher/equal priority updates are allowed to override lower priority ones (we will flip client server for the existing flows as well). In other words, the stickiness of marking “for the lifetime of a flow” only applies to degree model based marking.

## Conversation Mode

Secure Workload supports the following flow analysis fidelity modes:

- **Detailed Mode:** Historically, the **Detailed Mode** was the only mode available, where every observed flow was reported by the agent along with detailed stats about the observed flow. Stats captured are packet and byte counts, TCP flags, connection stats, network latency, SRTT, and others. While this kind of reporting is desirable in numerous cases, it is computationally intensive to report and process, also, it may not be strictly required when the primary use case is segmentation only.
- **Conversations Mode:** The **Conversations Mode** offers a more lightweight alternative to the traditional detailed mode. Agents in conversations mode aim to report conversations as opposed to flows whenever possible (i.e, whenever they are able to make the client-server classification accurately). This is applicable to TCP, UDP, and ICMP flows.

In detailed mode, for TCP/UDP flows, we report 5-tuple flows {source and destination IP, source and destination port, and protocol}. While for conversations mode, the agent omits the source port as they are ephemeral ports {changes on every new connection}, making it a 4-tuple flow.



---

**Note** Detecting a flow as 4-tuple also depends on client-server detection algorithms, which relies on the server/destination port being a well-known port (0–1023).

---

Thus, if you're using a custom application which doesn't use well-known server/destination ports, the OpenAPI interface can be used to punch well-known server ports. These configs are not applied to past flows, and only affect markings on flows from that point on (that is, going forward). To optimize server ports, see [Client Server Configuration](#).

Agent reports in conversations mode contain trimmed down information, full list of omitted fields includes:

- TCP/UDP source port (ephemeral ports)
- Fwd/Rev TCP bottleneck
- TCP handshake bucket
- SRTT(μs)
- Fwd/Rev Packet retransmissions
- SRTT Available
- Fwd/Rev Congestion Window Reduced
- Fwd/Rev MSS Changed
- Fwd/Rev TCP Rev Window Zero? Fwd/Rev Burst Indicator
- Fwd/Rev Max Burst Size (KB)

To enable conversations mode, see the Flow Visibility config section in: [Software Agent Config](#)



---

**Note** The exact benefit gained by changing agents to report in conversation mode may vary due to multiple factors, including, but not limited to percentage of TCP flows, number of services listening on well-known service ports, and memory limitations at the agent.

---



---

**Note** After turning on conversations mode for some agents, there may be a mixture of conversations and flows in the observations on the flow search page.

---

## Visibility in Proxied Flows

A proxy acts as a server positioned between client machines and the internet, controlling and restricting direct client access to the internet. When a client wants to access internet services, it directs the proxy server to initiate a TCP connection with web servers on its behalf. After successfully establishing the connection, the

proxy sends an HTTP response with a status to the client. Later, the client interacts over the established TCP connection, appearing to communicate directly with the web service. The proxy serves as a bridge, facilitating the transmission of data across the two TCP connections.

The workload, hosting an application with the CSW agent installed, initiates a request for internet services. Initially, it instructs the proxy to create a communication channel on its behalf. The interaction with the internet service takes place over the established connection to the proxy. The CSW agent captures solely the flow between the workload and the proxy server. The actual destination of this flow remains unknown with the current CSW agent configuration.

The agent employs the current PCAP filter to analyze all the outgoing TCP packets, scanning for the "CONNECT" HTTP verb within the payload. This process enables the agent to capture the proxy request within the flow. Upon exporting the flows to Collectors, the agent generates an **Effective Flow** for each identified proxy flow. It establishes a connection between the proxy and proxied flows using the **related\_key** field, incorporating the 5-tuple information.




---

**Note** The visibility in proxied flows is on by default. To disable the feature, add `enable_proxy_flows_visibility: 0` to the sensor config file.

---

#### Prerequisite

Set **Flow Analysis Fidelity** to **Detailed** mode.




---

**Note**

- Operates solely with HTTP/ HTTPS proxy.
- Captures CONNECT requests only. Currently, there is no support for GET requests.
- By default, the flow analysis fidelity mode in agents is **Conversations**.

---

#### Procedure

1. From the navigation menu, choose **Investigate** > **Traffic**.

The **Traffic** page facilitates swift filtering and in-depth exploration of the flow corpus.

2. Click the **Expand** icon to view the Flow Details.

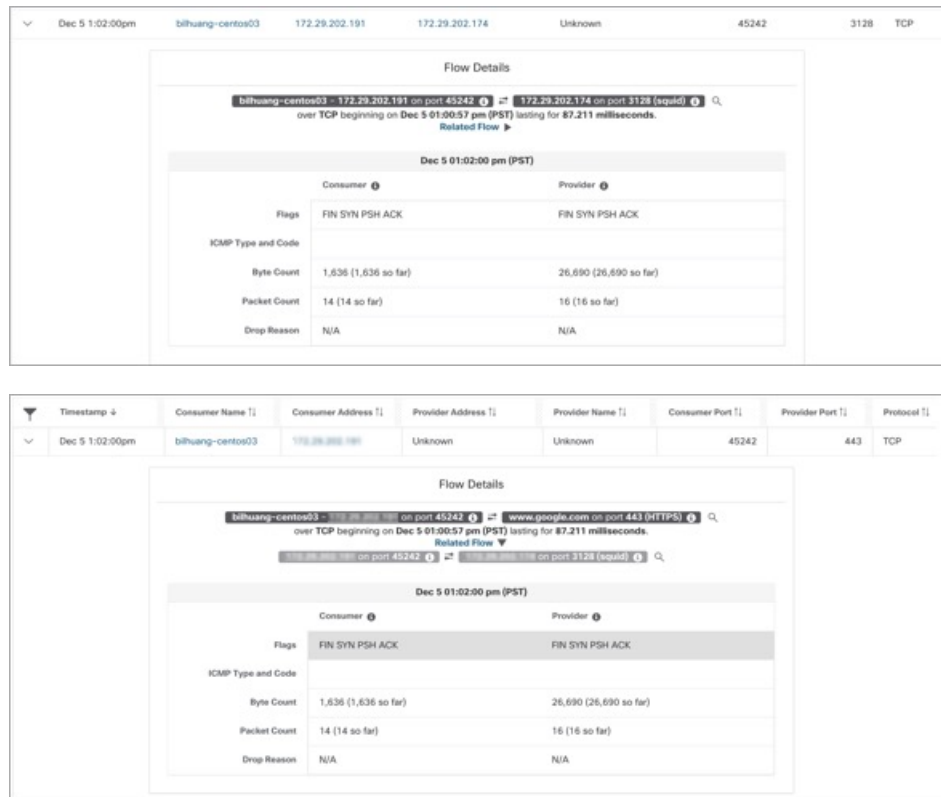
Agents of Version 3.9 and later can capture the destination of proxied flows. On the **Investigate** > **Traffic** page, you can observe two distinct flows:

- **Proxy Flow**: Originates from the workload to the proxy.
- **Proxied Flow**: Represents an effective and tunneled flow from the workload to the remote Fully Qualified Domain Name (FQDN) or IP Address.

These flows are interconnected and designated as **Related**. Specific considerations include:

- If the request to the proxy is directed at a remote FQDN, the **Provider Address** of the effective flow is marked as **Unknown**, but the **Provider Domain Name** is set to the FQDN.
- If the request to the proxy is directed at a remote IP address, the **Provider Address** is that specific address, while the **Provider Domain Name** is left empty.

Figure 374: Flow Details



## Visibility of Well-Known Malicious IPv4 Addresses

The Secure Workload threat intelligence data packs are updated with well-known malicious IPv4 addresses every 24 hours. The traffic originating from a consumer or a provider is analyzed against these malicious IPv4 addresses. This analysis helps in identifying the workloads connecting to these malicious IPv4 addresses, on the **Flow Search** page. To filter the flows connecting to well-known malicious IPv4 addresses, filter with **Malicious? = true**, **Provider Malicious? = true**, or **Consumer Malicious? = true** queries.



**Note** By default, the feature to identify well-known malicious IP addresses is disabled. You will not be able to view the malicious consumer or provider IP addresses and the read-only inventory filter for malicious inventory until you enable the feature.

To enable the feature to identify well-known malicious IP addresses, contact [Cisco Technical Assistance Center](#).







## CHAPTER 9

# Configure Alerts

---

Alerts in Secure Workload help you monitor workload security and respond to potential threats. The various components of alerts work together to provide visibility, alert sources and configuration, and the ability to send alerts from publishers. You can configure alerts, view alerts trigger rules, and choose publishers to send alerts. Alerts that are displayed on the configuration page vary depending on the user's role. Alert publishers can be either Alerts or Notifiers.

Table 34: Feature Information

| Feature Name       | Release | Feature Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   | Where to Find                                          |
|--------------------|---------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------|
| Alert Enhancements | 3.9     | <p>On the <b>Investigate &gt; Alerts</b> page. Other enhancements include:</p> <ul style="list-style-type: none"> <li>• Introduction of Alert Name field: A new field, <b>Alert Name</b>, is introduced to facilitate a structured approach to alert management, allowing you to assign unique names to alerts.</li> <li>• Introduction of drop-down for <b>Alert Type</b>: A new drop-down, <b>Alert Type</b>, is introduced in the <b>Alerts – Configs</b> page to facilitate quick filtering.</li> <li>• Icon Update: The configuration of alerts can now be initiated by selecting the new + icon on the <b>Alerts – Configs</b> page.</li> </ul> | <a href="#">Alert Configuration Modal, on page 652</a> |
|                    |         | Multisearch capabilities with enhanced filtering options for alerts.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  | <a href="#">Current Alerts, on page 660</a>            |



**Note** From the Secure Workload 3.0 release, the Secure WorkloadApp Store does not support alerts and compliance apps. You can configure alerts and the compliance alerts on this page without creating an Alert Application instance or Compliance Application instance.

- [Alert Types and Publishers, on page 649](#)
- [Create Alerts, on page 650](#)
- [Alert Configuration Modal, on page 652](#)
- [Generate Test Alerts, on page 658](#)
- [Current Alerts, on page 660](#)

- [Alert Details, on page 662](#)

## Alert Types and Publishers

Alerts in Secure Workload consist of the following components:

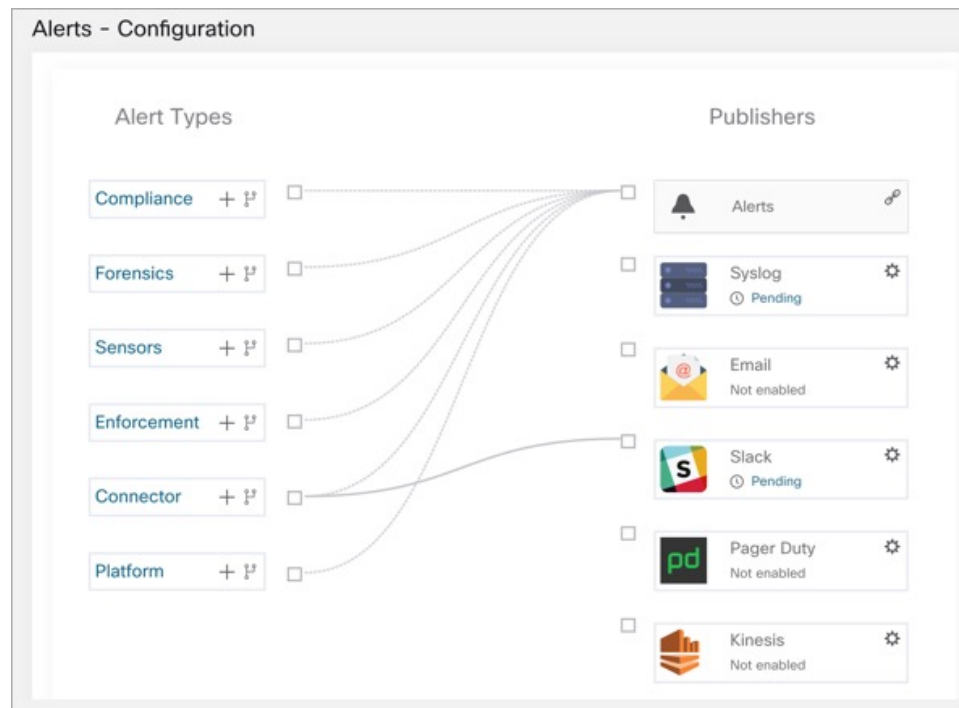
- **Alert Visibility**
  - **Current Alerts:** From the navigation pane, choose **Investigate > Alerts**. Preview of alerts is sent to a Data Tap.

**Figure 375: Current Alerts**

| Event Time     | Alert Name                          | Status | Alert Text                                          | Severity | Type      | Actions          |
|----------------|-------------------------------------|--------|-----------------------------------------------------|----------|-----------|------------------|
| Nov 9, 4:55 PM | ISE-Connector-Alert                 | ACTIVE | Missing ISE heartbeats, it might be down            | HIGH     | CONNECTOR | 2 <sup>2</sup> 🚩 |
| Nov 9, 4:55 PM | Syslog-Connector-Alert              | ACTIVE | Missing Syslog heartbeats, it might be down         | HIGH     | CONNECTOR | 2 <sup>2</sup> 🚩 |
| Nov 9, 4:55 PM | Slack-Connector-Alert               | ACTIVE | Missing Slack heartbeats, it might be down          | HIGH     | CONNECTOR | 2 <sup>2</sup> 🚩 |
| Nov 9, 4:55 PM | ServiceNow-Connector-Alert          | ACTIVE | Missing ServiceNow heartbeats, it might be down     | HIGH     | CONNECTOR | 2 <sup>2</sup> 🚩 |
| Nov 9, 4:55 PM | Edge Appliance-Appliance-Down-Alert | ACTIVE | Missing Edge Appliance heartbeats, it might be down | HIGH     | CONNECTOR | 2 <sup>2</sup> 🚩 |

- **Alert Sources and Configuration:**
  - **Alerts - Configuration:** Choose **Manage > Alerts Configs**. Both alert configurations that are configured using the common modal and alert publisher, and notifier settings are displayed.

Figure 376: Alerts - Configuration



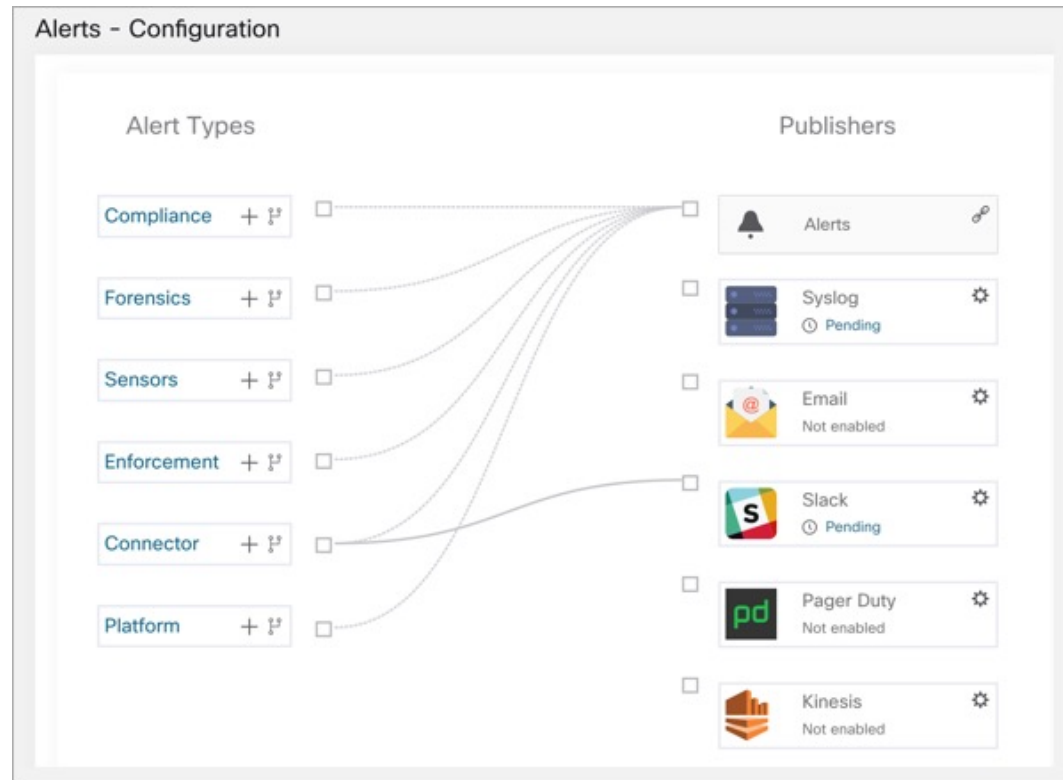
- **Send Alerts:**

- **Alerts App:** An implicit Secure Workload app that sends generated alerts to a configured Data Tap. The Alerts App handles features such as **Snooze** and **Mute**.
- **Alerts Publisher:** Limits the number of alerts that are displayed and pushes alerts to Kafka (MDT or DataTap) for external consumption.
- **Edge Appliance:** Pushes alerts to other systems such as Slack, PagerDuty, Email, and so on.

## Create Alerts

From the navigation pane, choose **Alerts > Configuration** to configure the following alert types:

Figure 377: Create an Alert



- **Enforcement Alerts**

- Agent Reachability
- Workload Firewall
- Workload Policy

- **Sensor Alerts**

- Agent Upgrade
- Agent Flow Export
- Agent Check In
- Agent Memory Usage
- Agent CPU Quota
- Amount Of Flow Observations
- New Agent Registered
- Pcap Status
- Agent Uninstalled
- Not Recommended Cipher

- Deprecated TLS Version
- Agent Auto Removal
- **Compliance Alerts**
  - Enforcement Policy
  - Live Analysis Policy

**Note**

- Alert trigger rules are enforced on the currently selected root scope for the Enforcement and Sensors alert types.
- You must have an enforced capability on the currently selected scope to create an alert trigger rule for the Compliance alert type.

The following Alert Types do not have a configuration modal:

- Forensics
- Connectors
- Traffic

**Traffic Alerts**

You can create **Traffic** alerts to be notified when workloads communicate with known malicious IPv4 addresses. By default, the option to detect malicious addresses are disabled. To enable, see [Visibility of Well-Known Malicious IPv4 Addresses, on page 645](#). The available alert conditions are:

- `Malicious flows are Observed`: Communication to the known malicious IPv4 addresses is observed.
- `Malicious flows are Permitted`: After the policy analysis and enforcement, this condition notifies about the malicious flows which are permitted.
- `Malicious flows are Rejected`: After the policy analysis and enforcement, this condition notifies about the malicious flows which are rejected.

## Alert Configuration Modal

The Alert configuration modal consists of the following sections:

- The **Alert Name**
- The **Alert Types**
- The *subject* of an alert. The subject depends on the app, and may be prepopulated when the alert modal is contextual.
- The **Alert Condition** on which an alert is triggered. Hover over the **info** icon to view a list of available conditions.

- If there are several alerts that are generated, alerts with higher *Severity* are displayed preferentially over alerts with lower severity.
- Click **Show Advanced Settings** for more configuration options.

**Note**

- Upon completion of the upgrade, all existing alert configuration rules in the current tenants are assigned with an **Alert Name** based on the predefined format. In instances where the Alert Name is absent, the format to be employed is *Alert\_SubType\_{DatabaseID}*. For example, *Workload\_Firewall\_64bf9b8493dfc94ca0095718*.
- Following deployment or upgrade, all default alert configuration rules—those created during the inception of a new tenant—are assigned with an **Alert Name** in the predefined format: *Alert\_SubType*. For example, *Upgrade\_Status*.

**Figure 378: Configure Alerts**

## Summary Alerts

Summary Alerts are allowed for some applications and configuration options depend on the application.

- **Individual Alerts** refers to alerts that are generated over non-aggregated (or minimally aggregated) information and are likely to have a time range of one minute. Note that this does not necessarily mean the alerts are actually generated and sent at a minute interval; the individual alerts can still be generated at the *App Frequency* interval.
- **Summary Alerts** refers to alerts generated over metrics produced over an hour or to the summarization of less frequent alerts.

| App         | App Frequency1 | Individual Alerts     | Hourly Alerts         | Daily Alerts          |
|-------------|----------------|-----------------------|-----------------------|-----------------------|
| Compliance  | Minute         | Yes: at app frequency | Summary of Individual | Summary of Individual |
| Enforcement | Minute         | Yes: at app frequency | Summary of Individual | Summary of Individual |
| Sensors     | Minute         | Yes: at app frequency | Summary of Individual | Summary of Individual |



**Note** The Event Time of summary alerts represents the first occurrence of the same type alert over the past hour or a specified interval window.

## Summarization Versus Snoozing

Summarization applies to the entire set of alerts generated according the alert configuration, while snoozing applies to a specific alert. This distinction is minor when the alert configuration is very specific, but is notable when the alert configuration is broad.

- For example, Compliance configuration is quite broad: an application workspace, and on which type of violation an alert should be generated. Thus, summarization would apply to all alerts triggered by a ‘escaped’ condition, while snoozing would apply to a very specific consumer scope, provider scope, provider port, protocol, and the escaped condition.
- On the opposite end, a platform alert configured to alert on a path between source scope and destination scope with a hop count less than some amount, will generate a very specific alert.

Other distinctions:

- Snoozing only results in an alert being sent when a new alert is generated after the snooze interval has passed. There is no indication of how many suppressed alerts might have occurred during the snooze interval.
- A summary alert is generated at the specified frequency, as many as alerts were generated within that interval. Summary alerts provide a count of the number of alerts triggered within the window, along with aggregated or range metrics.

## Secure Workload Alerts Notifier (TAN)



**Note** Starting Secure Workload Release 3.3.1.x, TAN is moving to **Secure Workload Edge Appliance**.

Alert Notifiers provide capabilities to send alerts through various tools such as Amazon Kinesis, Email, Syslog, and Slack in the currently selected scope. As a Scope Owner or Site Admin, each notifier can be configured with required credentials and other information specific to the notifier application.



## Configure Notifiers

To configure notifiers, you must configure the alert-related connectors. The connectors can only be configured after a Secure Workload Edge Appliance is deployed. For more information on deploying Secure Workload Edge appliance, see [Virtual Appliances for Connectors](#).

After the Secure Workload Edge appliance is set up, you can configure each notifier with its specific required input. After the Secure Workload Edge appliance is set up, you will be able to see dashed lines connecting Alert Types to Alerts publisher. This is because the notifier is built on the Alerts publisher.

After the Secure Workload Edge appliance is set up, you can configure each notifier with the required input. After the Secure Workload Edge appliance is set up, you are able to view the dashed lines connecting Alert Types to Publisher. This is due the fact that notifier is built on the Publisher.

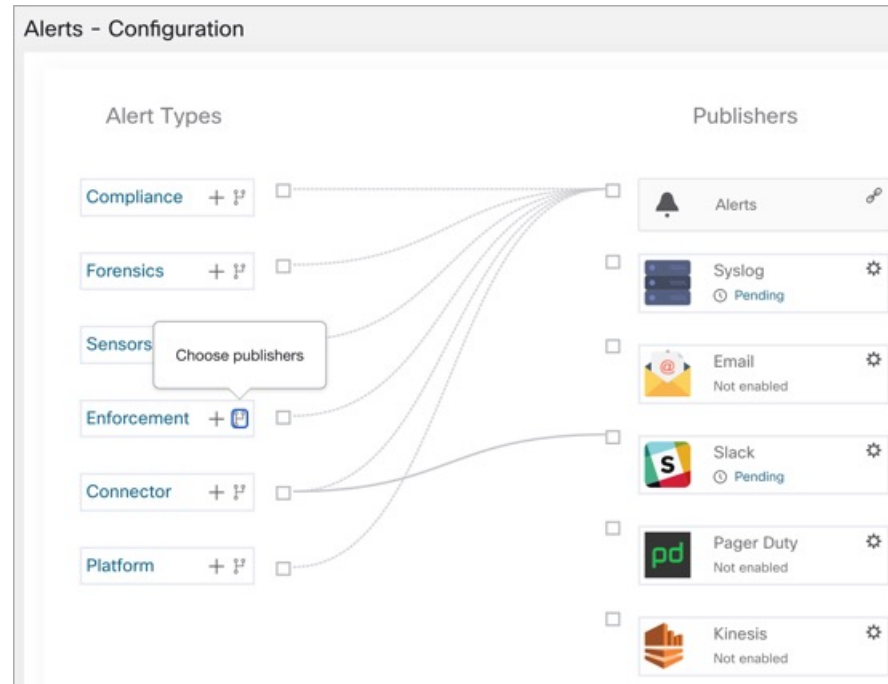
App Frequency is approximately how often the application runs and generates alerts. For example, Compliance has a flexible run frequency, and may actually compute alerts over a couple minutes together.

## Choose Alert Publishers

**Scope Owners** and **Site Admins** can choose **Publishers** to **Send** alerts. **Publishers** include Kafka (Data Tap) and **Notifiers**.

All the available Publishers are displayed in the **Alerts - Configuration** window, including the **Alerts** and **Active Notifiers**. You can toggle the **Send** icon to choose the **Publishers** for the alert type. **Minimum Alert Severity** refers to the severity level an alert must reach to be sent through the **Publishers**.

*Figure 379: Choose Alert Publishers*





**Note** TaaS clusters have a maximum number of alerts that can be processed of up to 14000 alerts per minute batch. This could also be reduced by choosing external data taps.

## External Syslog Tunneling Moves to TAN



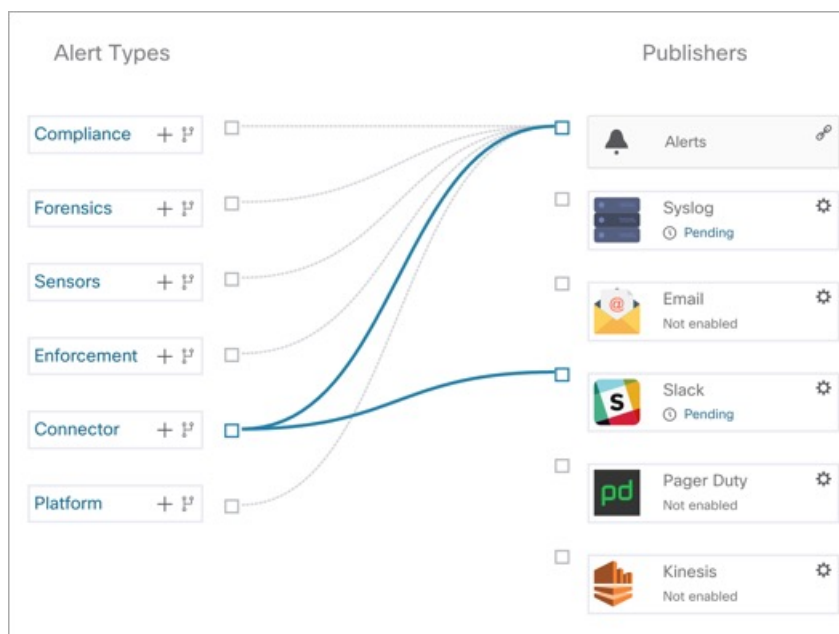
**Note** Starting the 3.1.1.x release, the syslog tunneling feature moves to TAN. To configure syslog for getting platform level syslog events, you must configure TAN on the Secure Workload Edge appliance on default rootscope. When the Secure Workload Edge appliance is configured on the default rootscope, you can set up the syslog server. To enable platform alerts, enable syslog notifications for Platform. This can be done by enabling Platform Syslog connection.

For details about how to configure syslog, see [Syslog Connector](#).

## Connection Chart

The connection chart displays the connections between **Alert Types** and **Publishers**. After you choose a publisher for an alert type, a blue line is established between the alert type and publisher. Note that the line pointing to the Internal Kafka (Data Tap) is always a line created using dashes as it represents an internal mechanism of how alert notifications are built upon.

**Figure 380: Connection Chart**



## View Alerts Trigger Rules

You can view a list of all the configured Alerts Trigger Rules on the **Alerts - Configuration** page. You can also perform the following tasks:

- You can filter the rules by **Alert Type** and other properties.
- In the **Actions** column, click the **pencil** icon to modify the details such as **Alert Name**, **Alert Types**, **Alert Condition**, **Severity** and so on.
- Click **See All Configured [alert type] Alerts** to view all the alerts of the selected **Alert Type** in a new tab.

**Figure 381: View Alerts Trigger Rules**

| Alert Type ↑↓ | Alert Name ↑↓                                        | Configuration ↑↓                                                                                                                                         | Actions ↑↓ |
|---------------|------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------|------------|
| ENFORCEMENT   | Agent_Not_Reachable_6537dc4a5da30b497a94de63         | Scope : Default when<br>Agent not Reachable (seconds) > 300                                                                                              |            |
| ENFORCEMENT   | Workload_Firewall_6537dc4a5da30b497a94de64           | Scope : Default when<br>Firewall = Off                                                                                                                   |            |
| ENFORCEMENT   | Workload_Policy_Deviation_s_6537dc4a5da30b497a94de65 | Scope : Default when<br>Policy = Deviated                                                                                                                |            |
| SENSORS       | Upgrade_Status_6537dc4a5da30b497a94de66              | Scope : Default when<br>Agent Upgrade Status = Failed                                                                                                    |            |
| SENSORS       | Iface_Flow_Export_Status_6537dc4a5da30b497a94de67    | Scope : Default when<br>Agent Flow Export Status = Stopped                                                                                               |            |
| SENSORS       | Upgrade_Srv_Check_In_6537dc4a5da30b497a94de68        | Scope : Default when<br>Agent Check-In Service = Inactive                                                                                                |            |
| SENSORS       | Agent_Mem_Usage_6537dc4a5da30b497a94de69             | Scope : Default when<br>Deep Visibility Memory Usage (MB) > 512<br>and<br>Enforcement Memory Usage (MB) > 512<br>and<br>Forensic Memory Usage (MB) > 256 |            |
| SENSORS       | Agent_Cpu_Quota_6537dc4a5da30b497a94de6a             | Scope : Default when<br>Deep Visibility CPU Quota (%) > 3 and<br>Enforcement CPU Quota (%) > 3 and<br>Forensic CPU Quota (%) > 3                         |            |
| SENSORS       | Amt_Of_Flow_Obs_6537dc4a5da30b497a94de6b             | Scope : Default when<br>Amount of Flow Observations > 500000                                                                                             |            |

The Alerts Trigger Rules window is used to filter alerts trigger rules by Alert Type and trigger condition.



**Note** Alert trigger condition is an exact match condition.

## Alerts Trigger Rules Details

Click a row in the **Alerts Trigger Rules** section to view the configuration details.

1. **Alert Type:** Type of the alert.
2. **Alert Name:** Name of the alert.

**3. Configuration:** The condition when an alert is triggered in a particular scope.

You can also view other details such as **Severity**, **Individual Alerts**, and **Summary Alert Frequency**.

**Figure 382: Alert Configuration Details**

The screenshot displays the 'Alerts Trigger Rules' configuration page. At the top, there is a search bar with 'All' selected and a 'Filter Alerts' button. Below this is a table with the following columns: Alert Type, Alert Name, Configuration, and Actions. Two rows are visible:

| Alert Type                                                                                         | Alert Name              | Configuration                                               | Actions        |
|----------------------------------------------------------------------------------------------------|-------------------------|-------------------------------------------------------------|----------------|
| ENFORCEMENT                                                                                        | Agent_Not_Reachable_    | Scope : Default when<br>Agent not Reachable (seconds) > 300 | [Trash] [Edit] |
| <p>Details</p> <p>Severity: Medium<br/>Individual Alerts: Enable<br/>Summary Alert Freq.: None</p> |                         |                                                             |                |
| SENSORS                                                                                            | Upgrade_Status_6537dc4a | Scope : Default when<br>Agent Upgrade Status = Failed       | [Trash] [Edit] |
| <p>Details</p> <p>Severity: Medium<br/>Individual Alerts: Enable<br/>Summary Alert Freq.: None</p> |                         |                                                             |                |

## Generate Test Alerts

The primary usage of generating a test alert is to verify the connectivity with the publisher. You can configure a test alert to send alerts based on the alert type and linked publisher in the alert configuration.



- Note**
- Generating test alerts is not from the actual sources and is generated for test purpose only.
  - Test alerts can be generated for alert types which are linked to at least one publisher.

To generate a test alert, follow the steps below:

### Procedure

- Step 1** From the navigation pane, choose **Manage > Workloads > Alerts Config**.
- Step 2** To configure a test alert, click **Test Alert**.

Figure 383: Test Alert Configuration

**Step 3** Under the **Keys** tab, enter the value for Alert Key and choose the values for Event Time, Alert Time, Alert Severity and Alert Type.

**Step 4** Under the **Scope** tab, the values of Scope ID and Tenant ID are autogenerated based on the current scope.

**Note** If the Tenant ID is the same as Tenant ID VRF, then the system automatically checks the Tenant ID VRF check box.

**Step 5** Under the **Details** tab, enter the values for Alert Text, Event Notes, Alert Details, and Alert Configuration ID.

**Note** Alert Details can be string or data in JSON format.

Options for JSON content are:

- a. Containing fields expected by that type of alert.
- b. Any sample JSON data, if that alert type does not expect default json fields.

Sample JSON:

```
{"alert_name ":"sample","alert_category":{"severity": "dummy"}}
```

**Step 6** Under the **Configuration** tab, choose the value for Individual Alert, Alert Frequency, and Summary Alert Frequency.

For individual alerts, choose *ENABLE* or *DISABLE* from the drop-down.

Alert frequency is autoselected with frequency as *INDIVIDUAL*.

**Note** It supports only individual alerts and does not consider summarization.

Summary alert is autoselected to *NONE*.

**Step 7** To generate the test alert, click **TEST**.

**Note** A test alert is generated and sent to the configured publisher.

## Current Alerts

Navigate to the **Investigate > Alerts** page to view the list of all active alerts. You can filter the alerts by **Status**, **Type**, **Severity**, and Time Range.

Only alerts with severity set to IMMEDIATE\_ACTION, CRITICAL, HIGH, MEDIUM, or LOW are displayed on the **Current Alerts** page. All alerts irrespective to the severity values are sent to the configured Kafka broker.

**Figure 384: Current Alerts**

| Event Time     | Alert Name                          | Status | Alert Text                                          | Severity | Type      | Actions |
|----------------|-------------------------------------|--------|-----------------------------------------------------|----------|-----------|---------|
| Nov 9, 4:55 PM | ISE-Connector-Alert                 | ACTIVE | Missing ISE heartbeats, it might be down            | HIGH     | CONNECTOR |         |
| Nov 9, 4:55 PM | Syslog-Connector-Alert              | ACTIVE | Missing Syslog heartbeats, it might be down         | HIGH     | CONNECTOR |         |
| Nov 9, 4:55 PM | Slack-Connector-Alert               | ACTIVE | Missing Slack heartbeats, it might be down          | HIGH     | CONNECTOR |         |
| Nov 9, 4:55 PM | ServiceNow-Connector-Alert          | ACTIVE | Missing ServiceNow heartbeats, it might be down     | HIGH     | CONNECTOR |         |
| Nov 9, 4:55 PM | Edge Appliance-Appliance-Down-Alert | ACTIVE | Missing Edge Appliance heartbeats, it might be down | HIGH     | CONNECTOR |         |

### Filter Alerts by Time Range

1. Choose a range from the drop-down list. The default value is 1 month.
2. Click **Custom** and fill in the **From** and **To** dates to configure a custom range. Click **Apply**. Note that when a custom time range is selected, the **Refresh** button is disabled.

### Advanced Filtering

1. Click **Switch to Advanced**.
2. Enter the attributes to filter. Hover over the **info** icon to view the properties to filter.

The alert filters are not retained when you switch back to the basic options.

### View Additional Alert Details

You can view more details by clicking an alert.

Figure 385: Alert Details

| Details         |                             |
|-----------------|-----------------------------|
| Name            | SLACK                       |
| Type            | SLACK                       |
| Appliance ID    | 653a32375da30b0faaa111ef    |
| Connector ID    | 653a34111af9610a1686ef48    |
| Connector IP    | 172.21.196.125/24           |
| Last Checkin At | Nov 07 2023 10.03.51 AM UTC |

- Only 60 alerts per minute per root scope are displayed. A higher volume of alerts result in an alert type called Summary Alerts, with a count of alerts that are not displayed .
- There is a maximum number of alerts that are displayed at any point in time; older alerts are dropped as new alerts come in.

For more information, see [Configuration Limits in Secure Workload](#).

## Snooze Alerts

The Alerts App allows alerts of the same type to be snoozed for a chosen amount of time. The type of the alert is defined differently depending on the workspace that the alert has currently been configured for. For example, the Compliance alert type is defined as the four tuples: consumer scope, provider scope, protocol, and provider port.



**Note** Currently, you cannot snooze or mute the user app-created alerts.

### Snooze or Mute an Alert

#### Snooze Alerts:

1. Under **Actions**, click the **Snooze** icon.
2. Choose an interval from the drop-down.
3. Click **Snooze**.

Figure 386: Snooze an Alert

| Event Time      | Alert Name                          | Status | Alert Text                                          | Severity | Type      | Actions           |
|-----------------|-------------------------------------|--------|-----------------------------------------------------|----------|-----------|-------------------|
| Nov 10, 4:59 PM | Stack-Connector-Alert               | ACTIVE | Missing Stack heartbeats, it might be down          | HIGH     | CONNECTOR | [Snooze an alert] |
| Nov 10, 4:59 PM | Edge Appliance-Appliance Down-Alert | ACTIVE | Missing Edge Appliance heartbeats, it might be down | HIGH     | CONNECTOR | [Snooze an alert] |
| Nov 10, 4:59 PM | Spring-Connector-Alert              | ACTIVE | Missing Spring heartbeats, it might be down         | HIGH     | CONNECTOR | [Snooze an alert] |
| Nov 10, 4:59 PM | ServiceNow-Connector-Alert          | ACTIVE | Missing ServiceNow heartbeats, it might be down     | HIGH     | CONNECTOR | [Snooze an alert] |
| Nov 10, 4:59 PM | ISE-Connector-Alert                 | ACTIVE | Missing ISE heartbeats, it might be down            | HIGH     | CONNECTOR | [Snooze an alert] |

**Mute Alert:**

Use the mute option to stop receiving alerts.

1. Under **Actions**, click the **Mute** icon.
2. To confirm, click **Yes**.

To unmute, remove the alert from the muted list. Use the **Status** filter drop-down to view all **MUTED** alerts and unmute the required alert.




---

**Note** You can view up to 5000 muted or snoozed alerts in a scope.

---

**Admiral Alerts**

Admiral is an integrated alerting system, which replaces Bosun from earlier releases. For more information, see the Admiral Alerts section.

## Alert Details

### Common Alert Structure

All alerts follow an overall common structure. The structure corresponds to the json message structure available through Kafka DataTaps.

| Field         | Format | About                                                                                                                                                |
|---------------|--------|------------------------------------------------------------------------------------------------------------------------------------------------------|
| root_scope_id | string | Scope Id corresponding to top scope in scope hierarchy.                                                                                              |
| key_id        | string | id field used for determining 'similar' alerts. Identical key_id's can be snoozed.                                                                   |
| type          | string | Type of the alert. Fixed set of string values: COMPLIANCE, USERAPP, FORENSICS, ENFORCEMENT, SENSOR, PLATFORM, FEDERATION, CONNECTOR                  |
| event_time    | long   | timestamp of when the event triggered (or if event spanned a range, then the beginning of the range). This timestamp is in epoch milliseconds (UTC). |



| Field                 | Format | About                                                                                                                                                                 |
|-----------------------|--------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| alert_time            | long   | Timestamp of when the alert was first attempted to be sent. This will be after the timerange of the event. This timestamp is in epoch milliseconds (UTC).             |
| alert_text            | string | Title of the alert.                                                                                                                                                   |
| alert_text_with_names | string | Same content as alert_text but with any id fields replaced by corresponding name. This field may not exist for all alerts.                                            |
| severity              | string | Fixed set of string values: LOW, MEDIUM, HIGH, CRITICAL, IMMEDIATE_ACTION. This is the severity of the alert. For some types of alerts these values are configurable. |
| alert_notes           | string | Usually not set. May exist in some special cases for passing additional information through Kafka DataTap.                                                            |
| alert_conf_id         | string | id of the alert configuration that triggered this alert. May not exist for all alerts.                                                                                |
| alert_details         | string | Structured data. Stringified json. See feature details for specific alert type, since the exact structure of this field varies based on the type of alert.            |
| alert_details_json    | json   | Same content of alert_details, but not stringified. Only present for compliance alerts, and only available through Kafka.                                             |
| tenant_id             | string | May contain vrf corresponding to root_scope_id. Or may contain 0 as the default value. Or may not be present at all.                                                  |
| alert_id              | string | Internal generated temporary id. Best ignored.                                                                                                                        |
| alert_name            | string | Name of the alert.                                                                                                                                                    |

- Compliance: lab-compliance-alert-details

- Forensics: [External Integration](#) and [Fields Displayed in Forensic Events](#)
- Sensor: [Sensor Alert Details](#)
- Enforcement: [Enforcement Alert Details](#)
- Connector: Alert Details
- Federation: federation-alert-details
- Platform: Alert Details

## General Alert Format by Notifier

The following are the examples of how alerts display across various notifier types.



**Note** Starting Secure Workload 3.9, notifier details include **Alert Name**.

### Kafka (DataTaps)

Kafka (DataTap) messages are in JSON format. Example below; see above alert\_details for some additional examples.

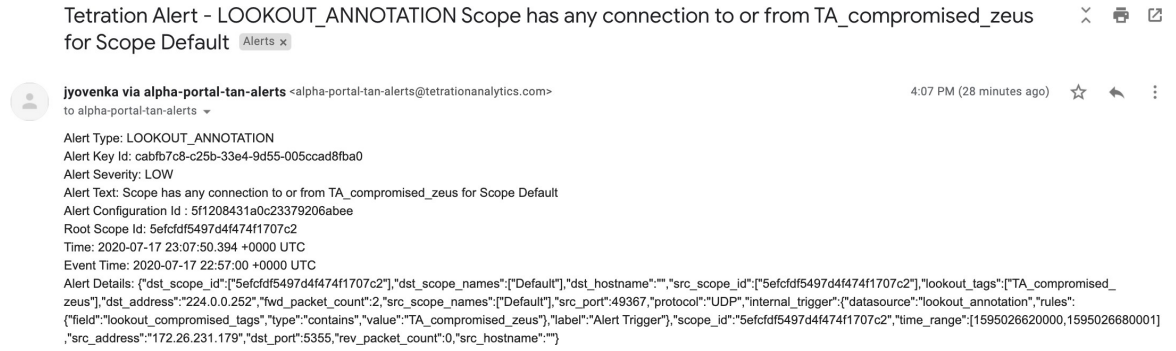
```
{
 "severity": "LOW",
 "tenant_id": 0,
 "alert_time": 1595207103337,
 "alert_text": "Lookout Annotated Flows contains TA_zeus for
<scope_id:5efcfd5497d4f474f1707c2>",
 "key_id": "0a4a4208-f721-398c-b61c-c07af3be9413",
 "alert_id": "/Alerts/5efcfd5497d4f474f1707c2/DataSource{location_type='TETRATION_PARQUET',
location_name='lookout_annotation', location_grain='HOURLY',
root_scope_id='5efcfd5497d4f474f1707c2'}/bd33f37af32a5ce71e888f95ccfe845305e61a12a7829ca5f2d72bf96237d403",

 "alert_text_with_names": "Lookout Annotated Flows contains TA_zeus for Scope Default",
 "root_scope_id": "5efcfd5497d4f474f1707c2",
 "alert_conf_id": "5f10c7141a0c236b78148da1",
 "type": "LOOKOUT_ANNOTATION",
 "event_time": 1595204760000,
 "alert_details":
 [{"scope_id": "5efcfd5497d4f474f1707c2", "time_range": [1595204760000, 159520480000], "src_address": "172.26.20.124", "dst_port": 437, "keypad_conf": 0, "src_hostname": ""}
]
}
```

### Email

Information about configuring Email alerts: [Email Connector](#)

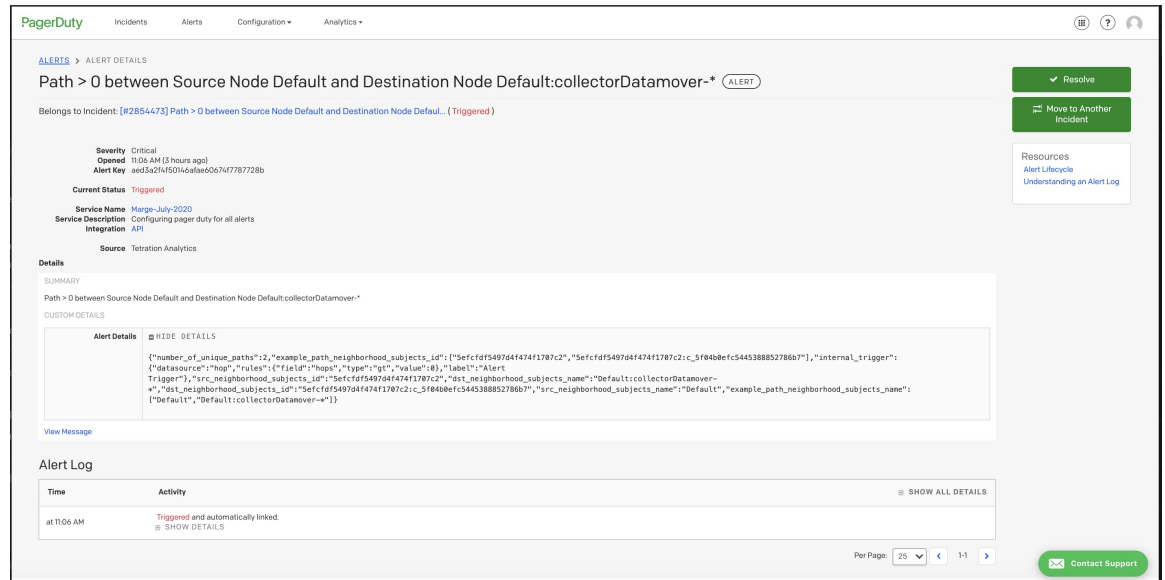
Figure 387: Example of a Cisco Secure Workload Alert



# PagerDuty

Information about configuring PagerDuty alerts: [PagerDuty Connector](#)

Figure 388: Example of a Secure Workload Alert in PagerDuty



Alerts sent to PagerDuty is a re-trigger of the same alert based on the key\_id.

Severity is mapped to PagerDuty severity as follows:

| Secure Workload Severity | PagerDuty Severity |
|--------------------------|--------------------|
| IMMEDIATE_ACTION         | critical           |
| CRITICAL                 | critical           |
| HIGH                     | error              |
| MEDIUM                   | warning            |
| LOW                      | info               |

## Syslog

Information about configuring Syslog alerts, and adjusting severity mapping: [Syslog Connector](#)

Figure 389: Example of several Secure Workload alerts sent to syslog

```
Aug 2 18:46:21 tan-5f035bae1a8c231d5880d7f8-tac-demo-data-ingest Tetratation Alert[26841]: [DEBUG] {"keyId":"3ee0d8b7-bc81-3427-9e84-6b9f8fedb98c","eventTime":"1596393720000","alertTime":"1596393968822","alertText":"Enforcement Annotated Flows contains escaped for \u003capplication_id:5f04b0b9755f024d4e36a279\u003e","severity":"LOW","tenantId":"","type":"COMPLIANCE","alertDetails":{"consumer_scope_ids":["5efcfd5497d4f474f1707c2"],"consumer_scope_names":["Default"],"provider_scope_names":["Default"],"provider_port":53,"application_id":"5f04b0b9755f024d4e36a279","constituent_flows":{"consumer_port":37367,"protocol":"UDP","consumer_address":"172.31.163.139","provider_address":"171.70.168.183","provider_port":53},"consumer_port":39652,"protocol":"UDP","consumer_address":"172.31.163.137","provider_address":"171.70.168.183","provider_port":53},"consumer_port":63811,"protocol":"UDP","consumer_address":"172.31.163.138","provider_address":"171.70.168.183","provider_port":53},"consumer_port":57448,"protocol":"UDP","consumer_address":"172.31.163.138","provider_address":"173.36.131.10","provider_port":53},"consumer_port":12599,"protocol":"UDP","consumer_address":"172.31.163.141","provider_address":"173.36.131.10","provider_port":53},"consumer_port":7385,"protocol":"UDP","consumer_address":"172.31.163.140","provider_address":"173.36.131.10","provider_port":53},"escaped_count":6,"provider_scope_ids":["5efcfd5497d4f474f1707c2"],"policy_type":"ENFORCED_POLICY","protocol":"UDP","internal_trigger":{"datasource":"compliance","rules":{"field":"policy_violations","type":"contains"},"value":"escaped"},"label":"Alert Trigger"},"time_range":["1596393720000,1596393779999"],"policy_category":["ESCAPED"]},"rootScopeId":"5efcfd5497d4f474f1707c2","alertConfId":"5f15cca71a0c231ebd66ca3b","alertTextWithNames":"Enforcement Annotated Flows contains escaped for Enforced Application j1"}
Aug 2 18:45:21 tan-5f035bae1a8c231d5880d7f8-tac-demo-data-ingest Tetratation Alert[26841]: [DEBUG] {"keyId":"8f0cfc5b-f8c1-3130-a069-3721b7d50159","eventTime":"1596393720000","alertTime":"1596393968822","alertText":"Enforcement Annotated Flows contains escaped for \u003capplication_id:5f04b0b9755f024d4e36a279\u003e","severity":"LOW","tenantId":"","type":"COMPLIANCE","alertDetails":{"consumer_scope_ids":["5efcfd5497d4f474f1707c2"],"consumer_scope_names":["Default"],"provider_scope_names":["Default"],"provider_port":5668,"application_id":"5f04b0b9755f024d4e36a279","constituent_flows":{"consumer_port":17131,"protocol":"TCP","consumer_address":"172.26.231.193","provider_address":"172.31.163.140","provider_port":5668},"escaped_count":1,"provider_scope_ids":["5efcfd5497d4f474f1707c2"],"policy_type":"ENFORCED_POLICY","protocol":"TCP","internal_trigger":{"datasource":"compliance","rules":{"field":"policy_violations","type":"contains"},"value":"escaped"},"label":"Alert Trigger"},"time_range":["1596393720000,1596393779999"],"policy_category":["ESCAPED"]},"rootScopeId":"5efcfd5497d4f474f1707c2","alertConfId":"5f15cca71a0c231ebd66ca3b","alertTextWithNames":"Enforcement Annotated Flows contains escaped for Enforced Application j1"}
Aug 2 18:45:21 tan-5f035bae1a8c231d5880d7f8-tac-demo-data-ingest Tetratation Alert[26841]: [DEBUG] {"keyId":"1ef4a974-be89-31de-abe9-d071cb0170ad","eventTime":"1596393720000","alertTime":"1596393968822","alertText":"Enforcement Annotated Flows contains escaped for \u003capplication_id:5f04b0b9755f024d4e36a279\u003e","severity":"LOW","tenantId":"","type":"COMPLIANCE","alertDetails":{"consumer_scope_ids":["5efcfd5497d4f474f1707c2"],"consumer_scope_names":["Default"],"provider_scope_names":["Default"],"provider_port":443,"application_id":"5f04b0b9755f024d4e36a279","constituent_flows":{"consumer_port":17792,"protocol":"TCP","consumer_address":"172.26.231.193","provider_address":"172.31.163.133","provider_port":443},"escaped_count":1,"provider_scope_ids":["5efcfd5497d4f474f1707c2"],"policy_type":"ENFORCED_POLICY","protocol":"TCP","internal_trigger":{"datasource":"compliance","rules":{"field":"policy_violations","type":"contains"},"value":"escaped"},"label":"Alert Trigger"},"time_range":["1596393720000,1596393779999"],"policy_category":["ESCAPED"]},"rootScopeId":"5efcfd5497d4f474f1707c2","alertConfId":"5f15cca71a0c231ebd66ca3b","alertTextWithNames":"Enforcement Annotated Flows contains escaped for Enforced Application j1"}
```

## Slack

Information about configuring Slack alerts: [Slack Connector](#)

Figure 390: Example of a Secure Workload alert sent to slack channel

10:37 Tetratation Alert Wednesday, July 29th

be200f5c2dbc linux-amd64 AgentInactive

|                                   |                               |
|-----------------------------------|-------------------------------|
| Severity                          | Type                          |
| MEDIUM                            | SENSOR                        |
| Alert Time                        | Event Time                    |
| 2020-07-29 17:37:49.519 +0000 UTC | 2020-07-29 17:37:01 +0000 UTC |

Root Scope Id  
5efcfd5497d4f474f1707c2

Details

```
{
 "agent_uuid": "6a968f8a8ddf2a4ec4534955d247bcb5ce484046",
 "details": {
 "AgentType": "NETSCALER",
 "Bios": "53C9551F-F149-4BC7-FAE4-BAF211FDF910",
 "CurrentVersion": "3.5.2.69722.stshanta.mrpm.build-netscaler",
 "DesiredVersion": "3.5.2.70759.dashboard.selfpmr.mrpm.build",
 "HostName": "be200f5c2dbc",
 "IP": "10.24.28.80",
 "LastConfigFetchAt": "2020-07-02 01:28:59 +0000 UTC",
 "Platform": "linux-amd64"
 },
 "scope_id": "5efcfd5497d4f474f1707c2",
 "scope_name": "Default",
 "vrf_id": 1
}
```

[Show less](#) ↓ Latest messages

## Kinesis

Information about configuring Kinesis alerts: [Kinesis Connector](#)

Kinesis alerts are similar to Kafka alerts, as these are both message queues.





## CHAPTER 10

# Monitor Configurations in Secure Workload

---

The **Monitoring** options available to you vary depending on your role.

- [Agent Monitoring](#), on page 669
- [Agent Monitoring Type](#), on page 669
- [Agent Status and Statistics](#), on page 671
- [Enforcement Status](#), on page 673
- [Enforcement Status for Cloud Connectors](#), on page 674
- [Pause Policy Updates](#), on page 675

## Agent Monitoring

The page displays the count of all monitored agents in a cluster based on the currently selected root scope.



---

**Note** Total Inventory count is the summation of all inventory observed on the network after applying collection rules.

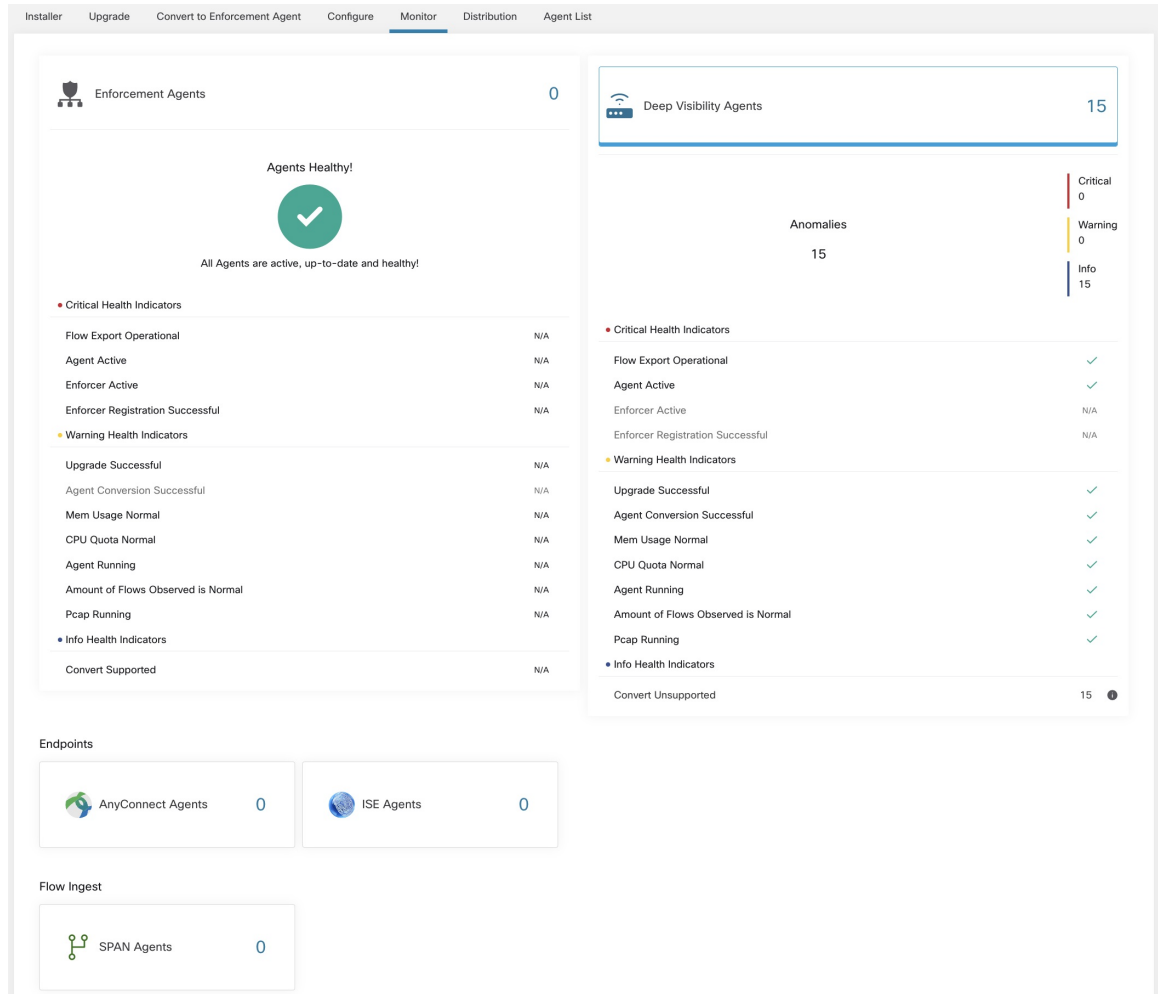
---

## Agent Monitoring Type

To monitor agents, click **Manage > Agents** in the left navigation bar, then click the **Monitor** tab.

This page is only available for users that have **Site Admin** and **Customer Support** roles. **Scope owners** can see Inventory, Deep Visibility Agents, and Enforcement Agents.

Figure 391: Total Number of Installed Agents



The following table shows the differences between each agent type.

| Agent Type             | Description                                                                                                                                                                           |
|------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Deep Visibility</b> | Provides highest fidelity in terms of time series flow data, processes running on a host. Most Linux and Windows platforms are supported. See <code>sw_agents_deployment-label</code> |
| <b>Enforcement</b>     | Provides all capabilities available in Deep Visibility Agents. In addition, Enforcement agents are capable of setting firewall rules on the installed host.                           |



|                                                                                                             |                                                                                                                                                                                                                                                                                                                                                |
|-------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>AnyConnect</b>                                                                                           | Provides time series flow data on endpoints running AnyConnect Secure Mobility Agent with Network Visibility Module (NVM) without requiring any Cisco Secure Workload agent installation. IPFIX records generated by NVM are sent to Secure Workload AnyConnect Proxy connector. Windows, Mac, and certain smartphone platforms are supported. |
| <b>ISE</b>                                                                                                  | Provides metadata about endpoints registered with Cisco ISE. Through ISE pxGrid, ISE connector collects the metadata, registers the ISE endpoints on Secure Workload as ISE agents pushes labels based on the attributes fetched from ISE appliance and LDAP attributes for the users logged in to the endpoints.                              |
| The following table provides a brief summary of various appliance agents provided by Cisco Secure Workload. |                                                                                                                                                                                                                                                                                                                                                |
| Appliance Agents                                                                                            | Description                                                                                                                                                                                                                                                                                                                                    |
| <b>SPAN</b>                                                                                                 | Provides the flow analysis without requiring any per-host agent installation. It runs in the Secure Workload ERSPAN VM appliance. It consumes ERSPAN packets sourced by any Cisco switch.                                                                                                                                                      |



**Note** Appliance agents such as NetFlow, NetScaler, F5, AWS, and AnyConnect Proxy are now supported as connectors. For more information on connectors, see [What are Connectors](#).

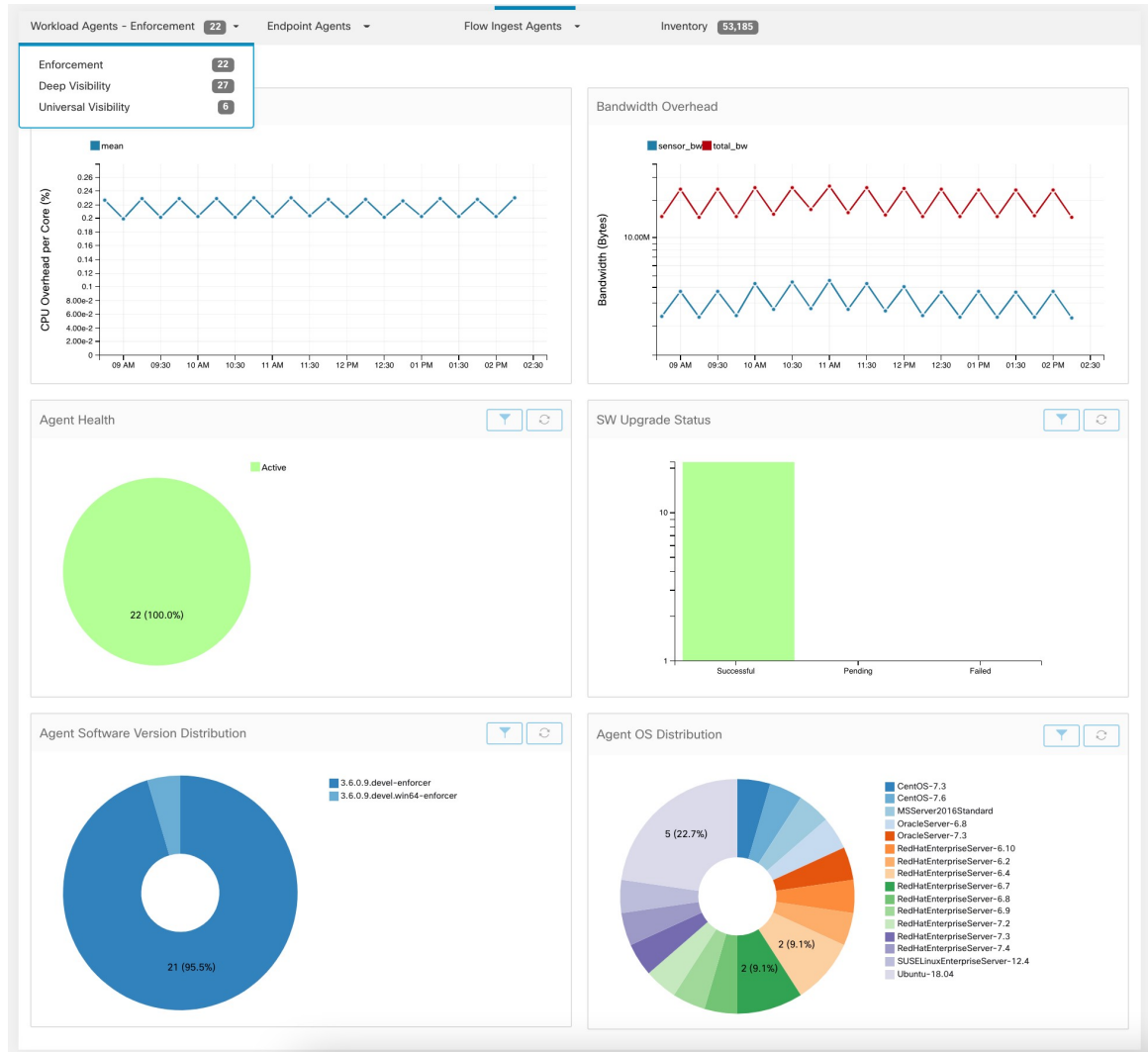
Any non-zero agent type button allows further drill-down into the distribution of each agent type.

## Agent Status and Statistics

To view the charts described in this topic, choose **Manage > Agents**, then click the **Distribution** tab.

The following charts are available for both Deep Visibility and Enforcement Agent types.

Figure 392: Agents Distribution



For each agent type, this page provides an overview and the health of registered agents including overall CPU overhead, bandwidth overhead, missed packets, OS/version distribution and agent upgrade status.

**CPU Overhead Chart**

The CPU Overhead chart provides an aggregated view of CPU overhead per core from all agents. Per-agent CPU Overhead is provided as part of the [Workload Profile](#). This chart is only available for Deep Visibility and Enforcement Agent Types.

**Bandwidth Overhead Chart**

The Bandwidth Overhead chart provides aggregated stats of total bandwidth and bandwidth used by agents. Per-agent bandwidth overhead is provided as part of the [Workload Profile](#). This chart is only available for Deep Visibility and Enforcement Agent Types.

**Agent Health Chart**

The Agent Health chart provides the number of active or inactive agents. Active agents are the ones checking in with config server for upgrade on regular intervals. The checking interval is 30 minutes. If we see that an agent has missed more than two check-in periods from an agent, it would be declared as an inactive agent.

#### Software Agent Updates to Latest Revision Chart

Every time an agent checks in with the config server, the agent would also provide its current RPM version. If an agent is configured to a specific version and is not able to update after 2 check-in periods, the agent would be declared as not able to upgrade to the latest version.

#### Agent Packet Missed Chart

In rare occasions when the traffic volume traversing a host is greater than the rate at which the agent is able to inspect, some packets are skipped from being analyzed. The number of missed packets and the corresponding agent name are displayed in this chart.

#### Agent Software Version and OS Distribution Charts

These charts show the agent version distribution and parent OS platform of all agents registered with the Secure Workload cluster.

## Enforcement Status

To view enforcement status, click **Defend > Enforcement Status** in the navigation bar at the left side of the window.

This page is available for site admin/customer support users and scope owners to get an overview of the current status of all the enforcement agents, including the cloud connectors that are enforcing a policy.

If any of the charts shows red or orange, see the applicable topic:

**Table 35: Enforcement Status Charts**

| Chart                     | Result         | Take Action                                                                                                                                                                                                                                                                            |
|---------------------------|----------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Agent Enforcement Enabled | Not Enabled    | Make sure enforcement is enabled in the agent configuration. See <a href="#">Create an Agent Configuration Profile, on page 67</a> .                                                                                                                                                   |
| Agent Policy Config       | Stale Policies | This situation is generally temporary and typically doesn't require any action. It occurs because a Secure Workload deployment based on labels updates inventory and policies dynamically.<br><br>However, if this situation persists for any individual workloads, contact Cisco TAC. |
| Agent Concrete Policies   | Skipped        | This indicates that policies weren't pushed to some agents.                                                                                                                                                                                                                            |



- Tip**
- To view status for individual scopes or for the entire tenant, use the **Filter by Scope** option at the top-left side of the page.
  - If the charts indicate a problem, identify which workloads have the problem by clicking the relevant part of a chart.  
The table displays the affected workloads.  
Alternatively, to see filtering options, click the (i) button in the **Filter** box below the charts.
  - To view a wealth of additional details, click the IP address link in the filtered list of workloads to display the Workload Profile page.

The following table describes the fields shown in the enforcement status table.

| Field                     | Description                                                                                                                                                                              |
|---------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Host Name                 | Host name of the workload.                                                                                                                                                               |
| Address                   | IP addresses of all the interfaces on the workload.                                                                                                                                      |
| Enforcement Enabled       | Indicates whether enforcement is enabled or not on the agent.                                                                                                                            |
| Concrete Policies in Sync | This indicates whether the desired version of concrete policies are currently enforced on the agent.                                                                                     |
| Concrete Policies         | If this value shows <b>Skipped</b> for any host, this means the limit on policies is reached for the agent on that host. (See <a href="#">Limits Related to Policies, on page 954.</a> ) |
| Policy Count              | The number of concrete policies on the agent.                                                                                                                                            |
| Status                    | The status of the latest policy config enforcement. If the status is <b>CONFIG_SUCCESS</b> , it indicates that current version is enforced without any issue.                            |

## Enforcement Status for Cloud Connectors

If you have set up AWS or Azure cloud connectors:

All interfaces enforcement status are displayed on the enforcement status page. If the policies are applied successfully, the policies are in sync else the corresponding error messages are displayed.

Policy count in the enforcement status page is Secure Workload accounting but not AWS or Azure rule accounting.

(AWS only) The hostname field on this page is derived from public DNS. If the public DNS is not enabled on the given VPC, the hostname field is empty.

# Pause Policy Updates



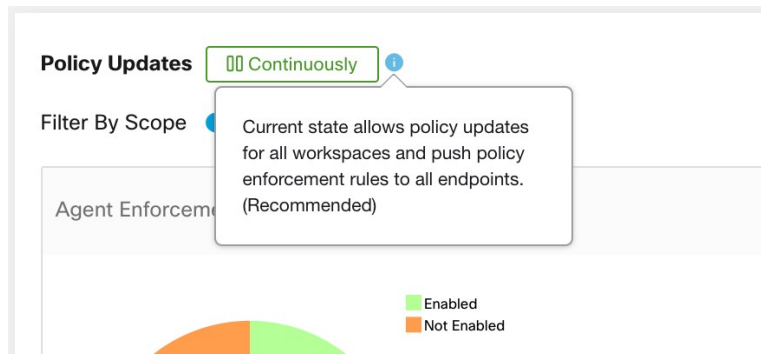
**Caution** This option pauses policy updates for ALL workloads in ALL scopes.

This feature requires site admin or customer support privileges.

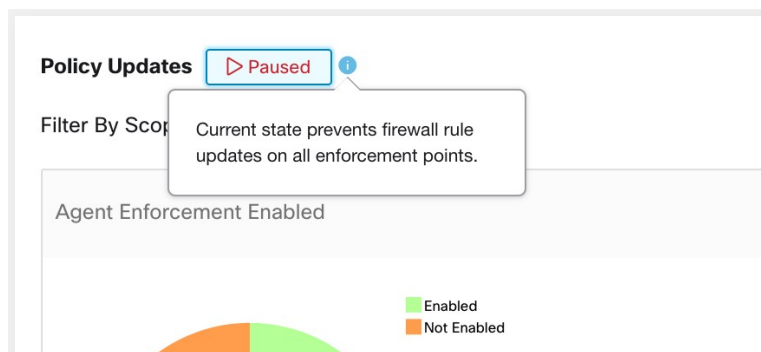
To pause rule updates for all enforcement endpoints in all scopes:

1. From the navigation pane, choose **Defend > Enforcement**.
2. Click the status beside **Policy Updates**.
3. Read and accept the caution.

**Figure 393: Firewall Rules are being Updated Continuously**



**Figure 394: Firewall Rule Updates are Paused**







## CHAPTER 11

# View Security Dashboard

---

This chapter provides information on the Security score, Security score categories, and Scope-Level score details that are presented under the Security dashboard.

Security Dashboard presents actionable security scores by bringing together multiple signals available in Secure Workload, which helps in understanding the current security position and improving it. Security Dashboard acts as a springboard to many richer drill-downs within Secure Workload, such as flow search, inventory search, automatic policy discovery, and forensics.

- [View the Security Dashboard, on page 677](#)
- [Security Score, on page 678](#)
- [Security Score Categories, on page 678](#)
- [High-Level View, on page 678](#)
- [Scope Level Score Details, on page 678](#)
- [Score Details, on page 681](#)

## View the Security Dashboard

To view the Security Dashboard, from the navigation pane, choose **Overview**.

# Security Score

Security Score is a number from 0 through 100, indicating the security position in a category. A score of 100 is the best score and a score of 0 is the worst. Scores closer to 100 are better.

The Security Score computation considers vulnerabilities in installed software packages, consistency of process hashes, open ports on different interfaces, forensic and network anomaly events, and compliance or noncompliance to policies.

## Security Score Categories

There are six different score categories. Most security aspects of a workload are taken into account to come up with these categories.

- **Vulnerability Score:** Vulnerabilities in the installed packages on a workload are used for scoring.
- **Process Hash Score:** Process hash consistency (and anomaly) along with Benign and Flagged process hashes is used for scoring.
- **Attack Surface Score:** Process may have one or more ports open on multiple interfaces to make services available. Unused open ports are used for scoring.
- **Forensics Score:** Severity of forensic events on a workload is used for scoring.
- **Network Anomaly Score:** Severity of network anomaly events on a workload is used for scoring.
- **Segmentation Compliance Score:** Compliance (permitted) and violations (escaped) to automatically discovered policies is used for scoring.

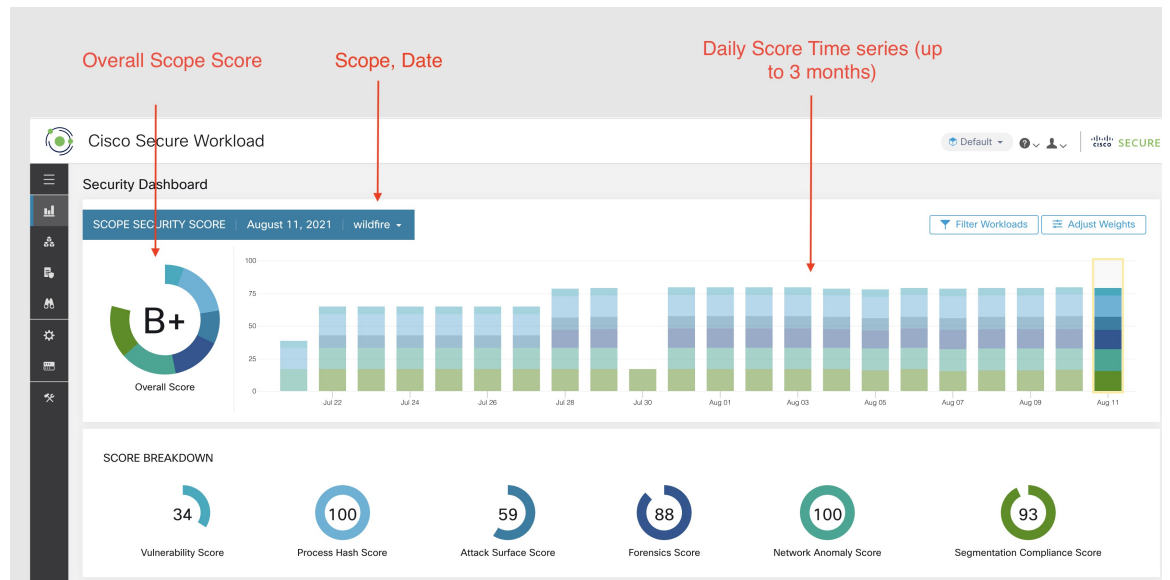
## High-Level View

Security dashboard has scope level scores for the selected scope. There is an overall score with time series and score breakdown. Score details for the six score categories for the selected scope are displayed.

## Scope Level Score Details

Scope Level Score details are on top of the dashboard.



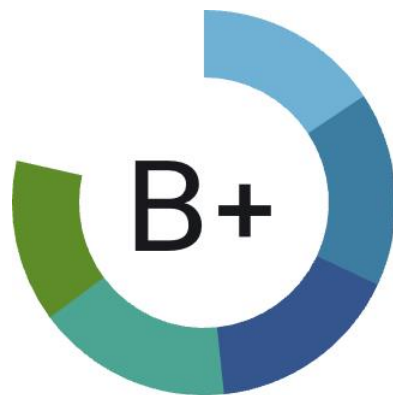


The following details are displayed:

- **Overall Scope Score:** Overall score for the selected scope.
- **Daily Score Time series:** Stacked time series that can go up to 3 months.
- **Score Breakdown:** Breakdown of category scores for the selected day on time series.

## Overall Score

The overall score is represented as a letter from **A+**, **A**, ..., **F**, with **A+** considered as the best score and **F** to be the worst. It is displayed as a donut chart with each slice (color-coded) representing a score category.

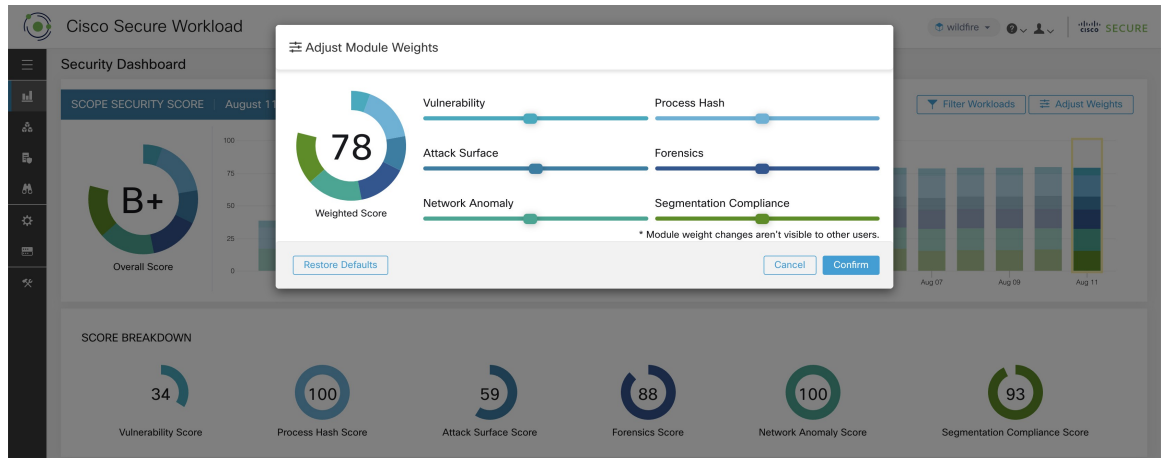


Overall Score

Overall score is the weighted average of the six score categories. By default, all weights are equal. If a score is **N/A**, it is considered as 0 in the overall score calculation.

$$\text{Overall score} = \frac{\sum W_{\text{category}} \times \text{Score}_{\text{category}}}{\sum W_{\text{category}}}$$

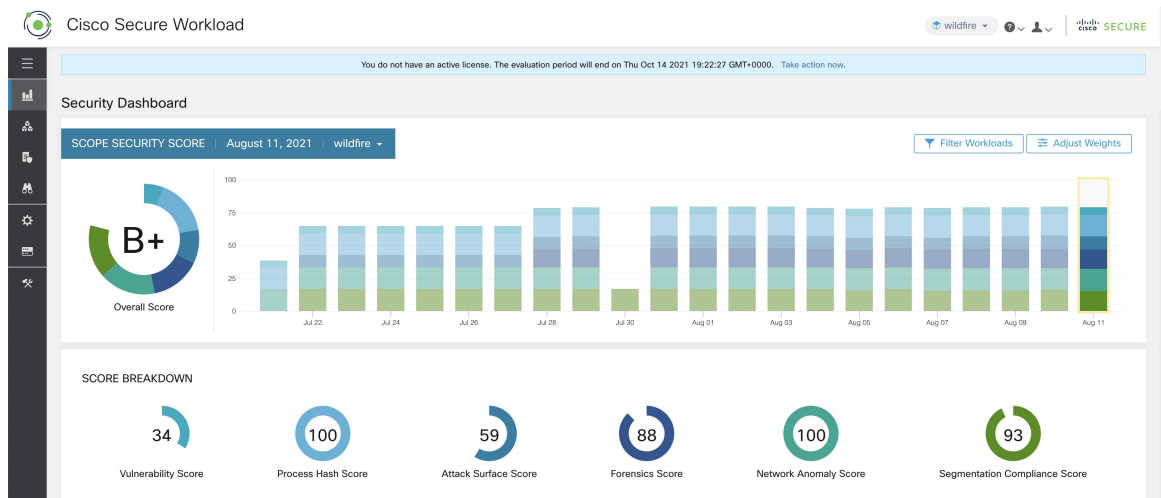
Weights can be adjusted using slides in the **Adjust Weights** module. Each user can set their own weight adjustments, which help in aligning scores with your priorities.



**Important:** If a score is N/A, it is considered as 0 in the overall score calculation.

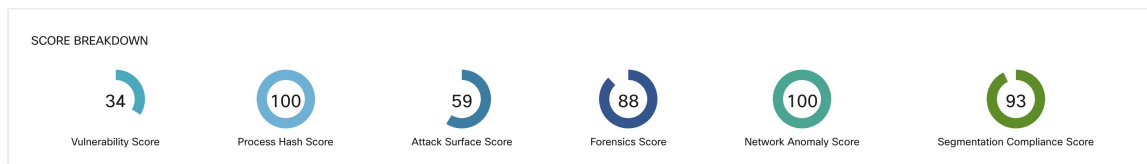
## Daily Time Series

Stacked time series that can go up to three months. It helps in tracking security position over a long period. Each stack represents an overall score for a day. Each segment in the stack is a category that is represented by different color. You can click on day to get the score breakdown for the day.



## Score Breakdown

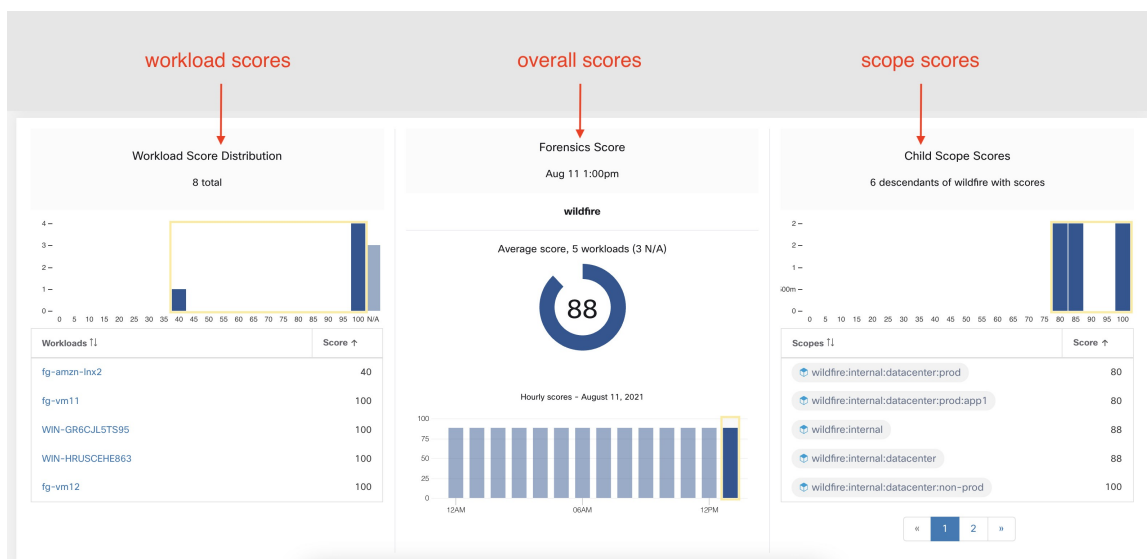
The Score Breakdown displays the score for all the six categories for the day that is selected on the time series. Score N/A indicates that score is not available. It will be counted as 0 for the overall score calculation.



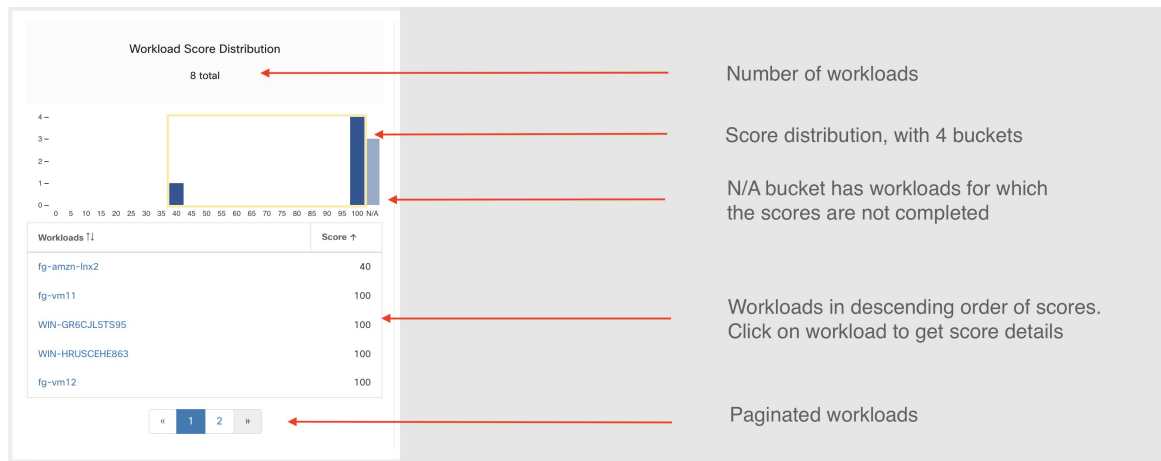
**Important** If a score is N/A, it is considered as 0 for overall score calculation.

## Score Details

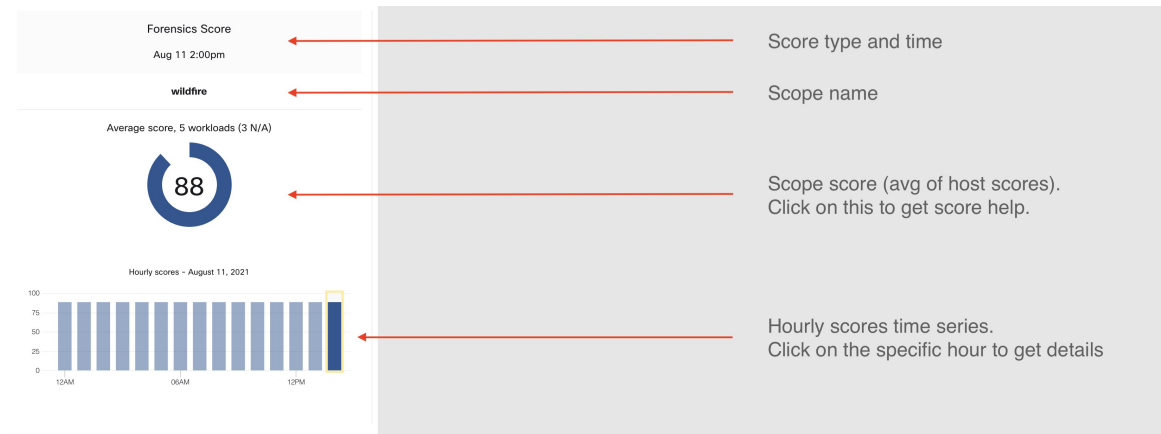
Each of the six categories follows the following template. It has workload score distribution, hourly time series, and child scope score distribution.



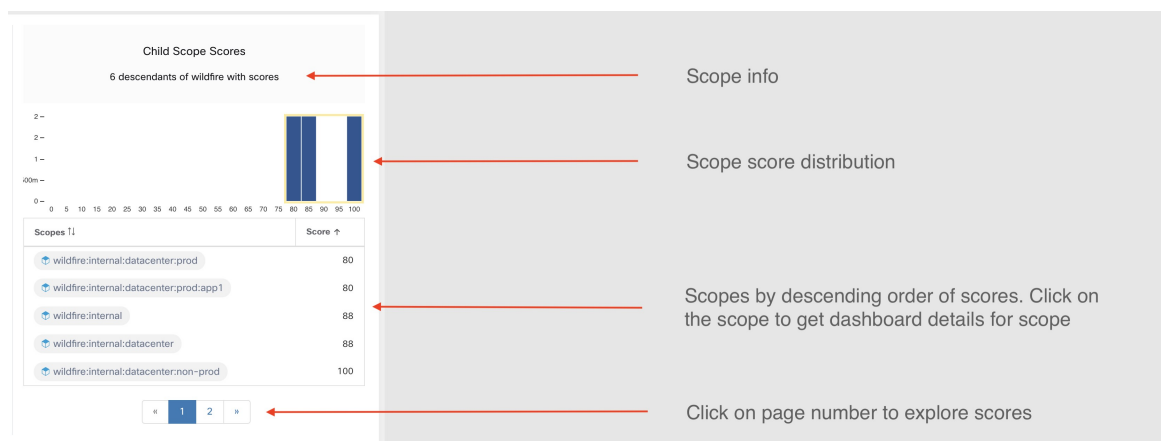
Workload score distribution provides insight into score contribution from workloads under the selected scope. It helps to bubble up lowest-scoring workloads to expedite corrective actions.



Hourly time series helps in getting the hourly score over the course of a selected day. Selecting an hour in the hourly time series updates the workload score distribution and descendent scope distribution to show the selected hour.



Descendent scope distribution provides insight into the score contribution of child scopes of the selected scope.

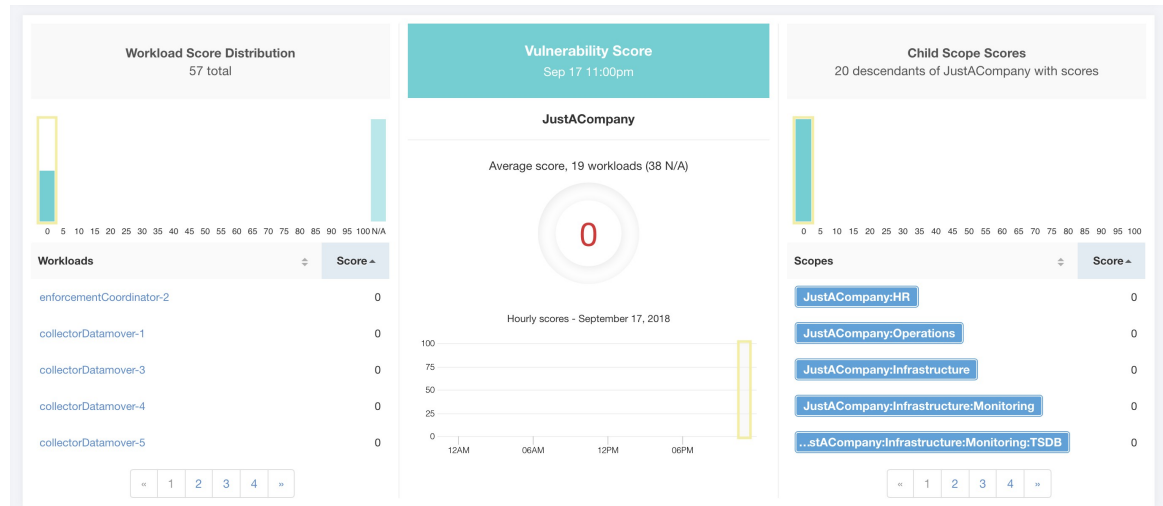


Details of each score category are explained in this section.

## Vulnerability Security Score

Vulnerabilities in software packages installed on workloads are used for computing Vulnerability Security Score.

**Figure 395: Vulnerability Security Score Details**



Lower score indicates:

- One or more installed software packages have serious vulnerabilities.
- Apply patch or upgrade to reduce the chances of exposures or exploits

Software packages on workloads could potentially be associated with known vulnerabilities ([CVE](#)). [CVSS \(Common Vulnerability Scoring System\)](#) is used for assessing the impact of a [CVE](#). CVSS score range is 0–10, with 10 being the most severe.

[CVE](#) can have CVSS v2 and CVSS v3 score. To compute Vulnerability score, CVSS v3 is considered if available, else CVSS v2 is considered.

Vulnerability score for a workload is derived from scores of vulnerable software that is detected on that workload. The Workload Vulnerability Score is calculated based on the CVSS scores, the vendor data, and may be adjusted by our security research team when data is missing or inaccurate (common for new vulnerabilities). This data is updated every 24 hours when the threat feed is configured. Higher the severity of the most severe vulnerability, lower is the score.

Scope score is average of workload scores in the scope. Improve the score by identifying workload or scopes with vulnerable software packages, and patch or upgrade with safer packages.

Figure 396: Help for Vulnerability Security Score

?

## Vulnerability Score Help

**Supported Agent Types** 19 supported workloads

|                             |                        |                   |
|-----------------------------|------------------------|-------------------|
| ✘ Universal Visibility (38) | ✔ Deep Visibility (19) | ✔ Enforcement (0) |
| ✘ AnyConnect (0)            | ✘ Hardware Switch (0)  |                   |

**What is a Vulnerability Score?**

A Vulnerability Score is an indicator of security posture in your deployment as it relates to software package vulnerabilities. We use standard [Common Vulnerability Scoring System](#) (CVSS score) to assess the impact of a vulnerability. The Vulnerability Score is calculated based on CVSS scores of vulnerabilities detected on a workload. Like all other Security Scores, a higher score is better, with 0 meaning there is a workload that requires immediate action, and 100 meaning there are no vulnerable packages observed within this Scope.

**How is the Vulnerability Score calculated?**

A Workload's Vulnerability Score is derived from the scores of vulnerable software detected on that workload. We use the vulnerable package's CVSS score to assess the impact of a vulnerability. Vulnerability score of a workload depends on the most severe vulnerability present in the system; higher the severity of most severe vulnerability, lower is the workload's score. The Vulnerability Score for a Scope is the average Vulnerability score of all workloads within that Scope.

**How do I improve my score?**

Updating software packages on the most vulnerable workloads to versions without (or with less severe) vulnerabilities is the best way to improve the score.

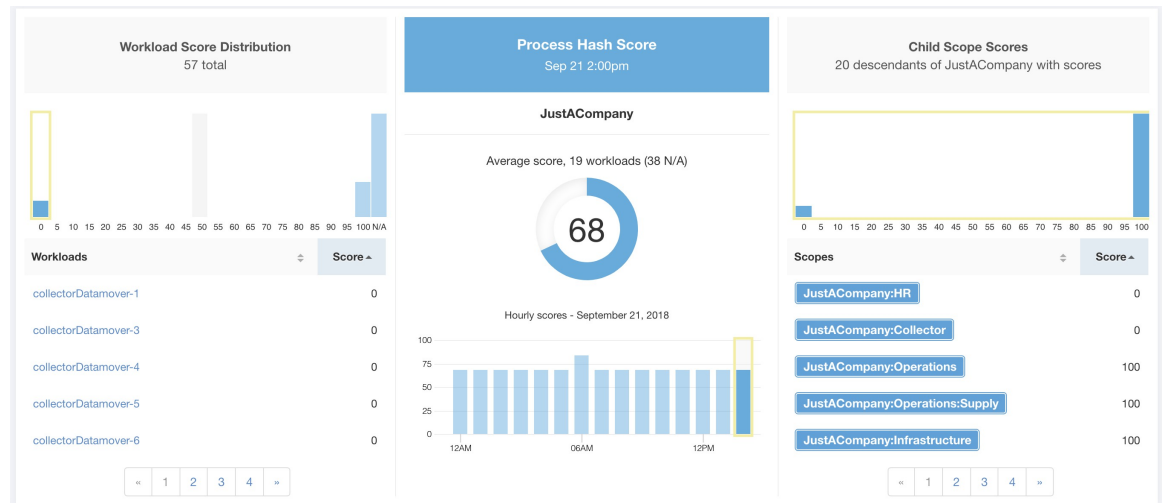
**How do I increase the number of workloads with scores?**

Vulnerability Scores can only be calculated when Deep Visibility Sensors are present. Install Deep Visibility Sensors on more workloads to improve your score coverage.

## Process Hash Score

Process hash score is an assessment of process binary hash (file hash) consistency across workloads. For example: A web server farm running Apache that is cloned from the same setup config is expected to have same hash for [httpd](#) binaries on all servers. A mismatch is an anomaly.

Figure 397: Process Hash Score Details



Lower score indicates, at least one or both of:

- One or more process hashes are flagged.
- One or more process hashes are anomalous.

Refer to [Process Hash Anomaly Detection](#) for more details.

Figure 398: Help for Process Hash Score

? Process Hash Score Help

**Supported Agent Types** 19 supported workloads

|                                                                                 |                                                                              |                                                                         |
|---------------------------------------------------------------------------------|------------------------------------------------------------------------------|-------------------------------------------------------------------------|
| <span style="color: red; font-weight: bold;">✘</span> Universal Visibility (38) | <span style="color: green; font-weight: bold;">✔</span> Deep Visibility (19) | <span style="color: green; font-weight: bold;">✔</span> Enforcement (0) |
| <span style="color: green; font-weight: bold;">✔</span> AnyConnect (0)          | <span style="color: red; font-weight: bold;">✘</span> Hardware Switch (0)    |                                                                         |

**What is a Process Hash Score?**

A Process Hash Score gives an assessment of the consistency of a process binary hash across the system. For example, if you have a farm of web servers running Apache that are cloned from the same configured setup, you would expect that the hashes of [httpd](#) binaries on all servers are the same. If there is a mismatch, it is an anomaly and worth a further investigation. To reduce false alarms, we use the [NIST RDS hash dataset](#) as a whitelist. A whitelisted hash is considered "safe." You can also upload your own hash whitelist and blacklist. A blacklisted hash, if detected, will require immediate action.

Like all Security Scores, a higher score is better, with 0 meaning there is a blacklisted process hash in the system, and 100 meaning there is no hash anomaly observed in the system.

**How is the Process Hash Score calculated?**

For each process hash we compute a score as follows:

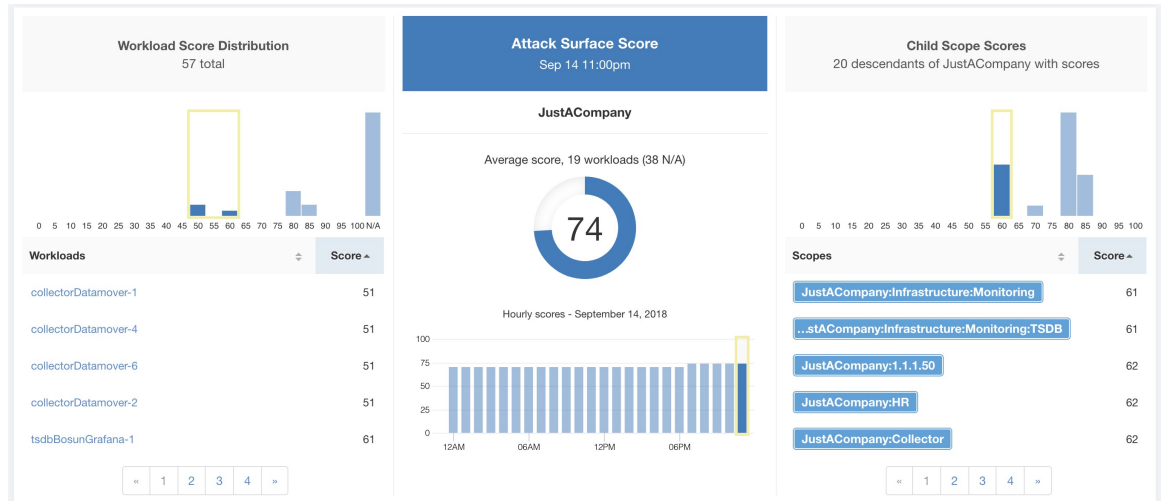
1. If hash is blacklisted: score = 0
2. Else, if hash is whitelisted: score = 100
3. Else, if hash is an anomaly: score is in the range of [1, 99], the higher the better
4. Else: score = 100

## Attack Surface Score

Attack Surface Score highlights potential attack surface in a workload. Open unused ports (open ports without traffic) contribute to lowering this score.



Figure 399: Attack Surface Score Details



A lower score indicates:

- Many open ports without any traffic in the last 2 weeks
- Well-known attack ports may be open and unused in last 2 weeks.
- One or more open ports are attached with packages that have serious vulnerabilities.

The attack surface score is a function of unused open ports relative to total ports, with a smoothing factor. Open ports without any traffic over the past 2 weeks are considered “unused open ports”. An extra penalty is applied to unused open ports which are well-known ports that are used in attacks (for example, 21, 22, 8080 and so on).

Figure 400: Attack Surface Score Formula


$$\text{Attack surface score} = \frac{\alpha + \sum \text{used open ports}}{\alpha + \sum \text{open ports} + (\rho * \sum \text{unused common attack ports}) + f_v(\text{vulnerability pkgs})}$$

$$f_v = \max \left( \left\{ \text{cve}_{\text{score}} = \begin{cases} \text{CVSS}_{v3}, & v3 \text{ exist} \\ \text{CVSS}_{v2}, & v3 \text{ not exist} \end{cases} \right\} \right)$$

Laplace smoothing is used with a penalty factor based on heuristic data. Score is computed daily with the past 2 weeks of data.

Tenant score is average of workload scores in the scope. Improve the score by identifying workload or scopes with unused open ports, and closing the unused ports.

When a workload link is clicked an attack surface modal is opened with details on all available ports and interfaces within the context of that workload.



33

**Attack Surface Details -** [redacted]

Jun 19 12:00pm to Jun 19 1:00pm

**22 Total Ports (12 unused ports on this workload)** Unused Ports Only

These are open ports and interfaces that haven't had traffic in the last 15 days (see help for specifics). Consider closing them to reduce your attack surface (and increase your Attack Surface Score) if they aren't needed.

| Port             | Package Name   | Total Permitted | CVE Max Score | Process Hash | Interfaces | Package Publisher  | Package Version |
|------------------|----------------|-----------------|---------------|--------------|------------|--------------------|-----------------|
| 22 (SSH)         | openssh-server | 16226           | None          | ...cec50428  | 2          | CentOS BuildSystem | 5.3p1           |
| 25 (SMTP)        | None           | 16254           | None          | ...6ed2d10f  | 2          | N/A                | None            |
| 53 (DNS)         | dnsmasq        | 36540           | 9.8           | ...5d28e929  | 2          | CentOS BuildSystem | 2.48            |
| 68               | dhclient       | N/A             | None          | ...69235c25  | 1          | CentOS BuildSystem | 4.1.1           |
| 123 (NTP)        | ntp            | 100425          | 7.5           | ...7c8791b1  | 6          | CentOS BuildSystem | 4.2.6p5         |
| 631              | cups           | N/A             | 7.5           | ...d417c9ea  | 1          | CentOS BuildSystem | 1.4.2           |
| 3128             | squid          | N/A             | 8.6           | ...7dc4807b  | 1          | CentOS BuildSystem | 3.1.23          |
| 5111             | collector      | 15998           | None          | ...a506dd9f  | 1          | (none)             | 3.4.2.4f        |
| 5222             | None           | 7999            | None          | ...524a83d7  | 1          | N/A                | None            |
| 5640 (Tetration) | collector      | N/A             | None          | ...a506dd9f  | 1          | (none)             | 3.4.2.4f        |

« 1 2 3 »


#### Features:

- **Unused Ports Only:** checkbox that when toggled filters out the ports that are used and only shows you the unused ports that are associated with the workload.
- **Columns:** Approved, port, package name, total permitted, CVE Max Score, Process Hash, Interfaces, Package Publisher, Package Version, Total Escaped, Total Rejected, Commonly Hacked Port, Links.
- **Interfaces:** If you click on any one of the line items in the Attack Surface table you can view the interfaces that are associated with each port inside a modal.
- **Approved:** checkbox that when toggled, allows you to intentionally set an “unused port” as “approved” on any one of the scopes on the scope chain that that workload has access to. Note: if a port is approved on a scope and that port is not explicitly approved on any of the children (if that scope has children), then the scope checkboxes are disabled as it is implied that any child scope that the parent scope has access to already is approved in that chain.

#### Approval Modal:

### Edit Approval of port 22

Make sure to be as specific as you can while approving higher up the scope chain as you will be approving this port in all of its children.

Tetration : Collector  
 Tetration   
 Default

Interfaces Modal:

### Interfaces for port: 4242

| Interface | Permitted * | CVE Score | PID   | Escaped | Rejected | Links |
|-----------|-------------|-----------|-------|---------|----------|-------|
| 0.0.0.0   | 8518443     | None      | 25642 | N/A     | N/A      | None  |
| 0.0.0.0   | 8518443     | None      | 21680 | N/A     | N/A      | None  |

\* Based on Host Firewall

Figure 401: Help for Attack Surface Score

?
Attack Surface Score Help

**Supported Agent Types** 19 supported workloads

|                             |                        |                   |
|-----------------------------|------------------------|-------------------|
| ✘ Universal Visibility (38) | ✔ Deep Visibility (19) | ✔ Enforcement (0) |
| ✘ AnyConnect (0)            | ✘ Hardware Switch (0)  |                   |

**What is an Attack Surface Score?**

An Attack Surface Score is an indicator of security posture in your deployment as it relates to unused open ports on the workloads. Intuitively, the more open ports available to an attacker, the larger the attack surface. Unused ports are ones that can be easily remedied by blocking those ports if they aren't needed.

Ports are considered unused if no traffic is observed on them over the previous 2 weeks. When this feature is initially enabled - either in a new deployment (or upgrade to 3.1) or a new Deep Visibility sensor is installed on a workload - the score will gradually improve over the course of those two weeks as the system stabilizes and learns what ports are in fact unused. Scores are computed daily; newly added sensors will not have scores immediately.

Like all Security Scores, a higher score is better, with 0 meaning there is an open port on a host that needs to be immediately closed, and 100 meaning there are no unused open ports observed in the system.

**How is the Attack Surface Score calculated?**

The Attack Surface Score is based on the ratio of unused ports to total opened ports, with an additive smoothing to adjust the score so smaller numbers of unused ports will give better scores. E.g. 1 unused port and 2 total ports should give a better score than 100 unused ports and 200 total ports even though the ratio in both cases is 1/2.

The most well-known ports that are commonly hacked are penalized with a much greater weight since they often expose many more vectors of attack. Examples of those ports are 21-FTP, 22-SSH, 23-Telnet, and 8080, 8088, 8888, etc (which are often used for web servers).

**How do I improve my score?**

Currently, the only way to improve your Attack Surface Score is by closing unused interfaces and/or ports. We will be incorporating more sophisticated approaches in the future, including combining open ports with known vulnerabilities, and allowing unused ports to be present if there are policies that apply to that port.

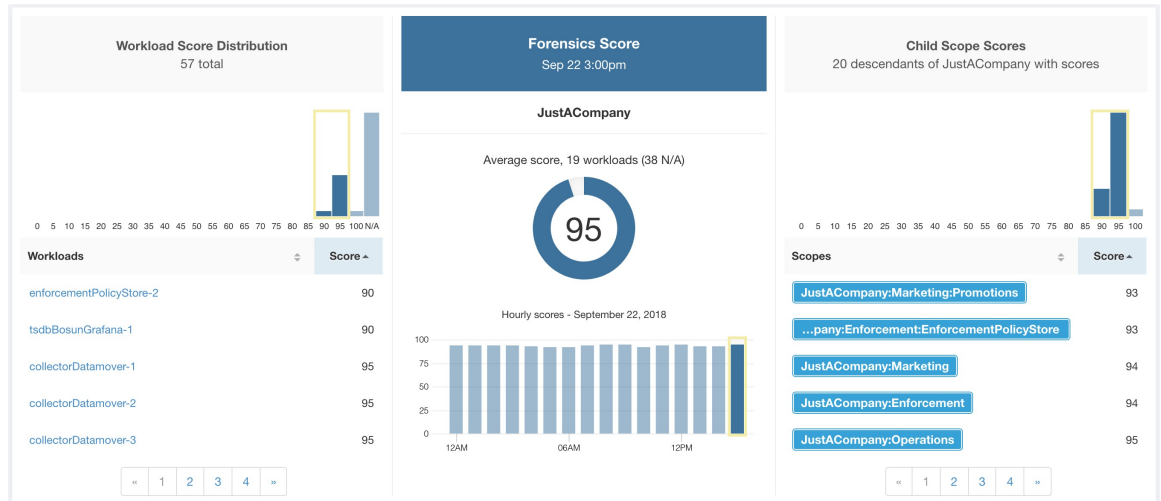
**How do I increase the number of workloads with scores?**

Attack Surface Scores can only be calculated when Deep Visibility, Enforcement, or AnyConnect Sensors are present. Install more of these sensors to increase your Attack Surface Score coverage.

## Forensics Score

Severity of forensics events on workloads is used for computing the scores.

Figure 402: Forensics Score Details



Lower score indicates:

- One or more forensics events were observed on the workload.
- Or one/more forensics rules are noisy and/or incorrect.

To improve the score:

- Fix the issue if any to reduce the chances of exposures/exploits.
- Tweak forensics rules to reduce noise and false alarms.

Forensics score for a workload is inverse function of total impact score of forensics events. Higher is the total impact score of forensics events, lower is the forensics score.

| Severity         | Impact Score |
|------------------|--------------|
| IMMEDIATE_ACTION | 100          |
| CRITICAL         | 10           |
| HIGH             | 5            |
| CRITICAL         | 3            |

Figure 403: Forensics Score Formula

$$\text{forensics score} = \max(0, (100 - \sum \text{forensics event impact score}))$$

Refer to [Configure and Monitor Forensic Events](#) for more details.

Figure 404: Help for Forensics Score

? Forensics Score Help

**Supported Agent Types** 19 supported workloads

|                                                              |                                                                  |                                                             |
|--------------------------------------------------------------|------------------------------------------------------------------|-------------------------------------------------------------|
| <span style="color: red;">✘</span> Universal Visibility (38) | <span style="color: green;">✔</span> <b>Deep Visibility (19)</b> | <span style="color: green;">✔</span> <b>Enforcement (0)</b> |
| <span style="color: red;">✘</span> AnyConnect (0)            | <span style="color: red;">✘</span> Hardware Switch (0)           |                                                             |

**What is a Forensics Score?**

A Forensics Score is one of the Security Scores that when combined will give a simple assessment of your overall security posture. Like all other Security Scores, a higher score is better, with 0 meaning there is a workload that requires immediate action, and 100 meaning there are no Forensic Events observed within this Scope.

**How is the Forensics Score calculated?**

For each Workload we compute a Forensics Score. A Workload's Forensics Score is derived from the Forensic Events observed on that Workload based on the [profiles enabled for this scope](#). A score of 100 means no Forensic Events were observed, and a score of 0 means there is a Forensic Event detected that requires immediate action. The Forensic Score for a Scope is the average Workload score within that Scope.

- A Forensic Event with the severity **CRITICAL** reduces a workload's score with the weight of **10**.
- A Forensic Event with the severity **HIGH** reduces a workload's score with the weight of **5**.
- A Forensic Event with the severity **MEDIUM** reduces a workload's score with the weight of **3**.
- A Forensic Event with the severity **LOW** doesn't contribute to the Forensics Score. This is recommended for new rules where the quality of the signal is still being tuned and is likely to be noisy.
- A Forensic Event with the severity **REQUIRES IMMEDIATE ACTION** will reduce the Score for the entire Scope to zero.

**How do I improve my score?**

Tuning your Forensics Score can be done by adjusting the Forensic Rules [enabled for this Scope](#). Creating rules that are less noisy will give you a more accurate score. Acting upon and preventing legitimate Forensic Events (events that are evidence of an intrusion or other bad activity) is another good way to improve your Forensic Score.

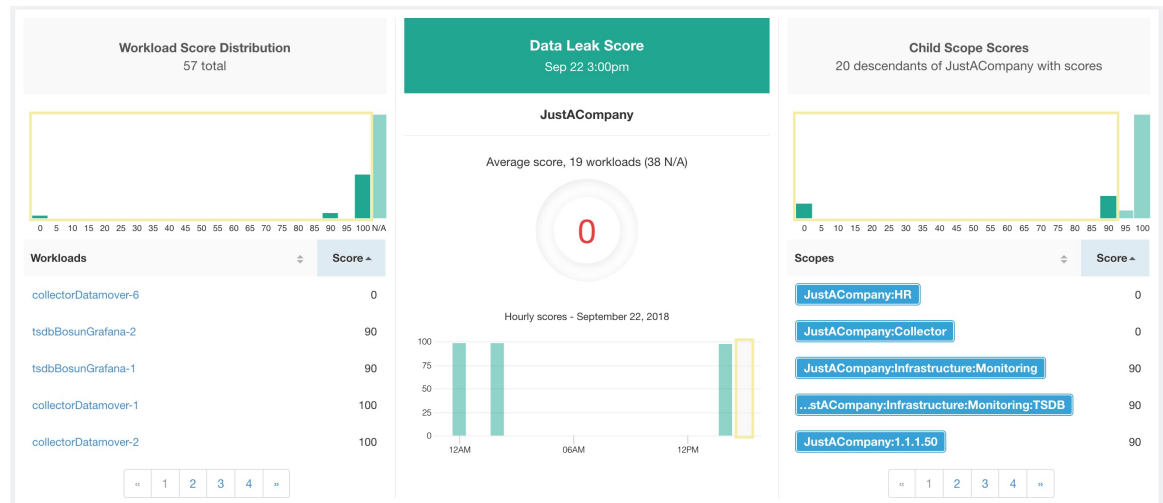
**How do I increase the number of workloads with scores?**

See the compatibility chart above for which sensor types are compatible. Installing the supported sensor types on more Workloads will increase your Forensic coverage.

## Network Anomaly Score

Severity of Network Anomaly events on workloads is used for computing the scores.

Figure 405: Data Leak Score Details



Lower score indicates:

- Unusually high amount of data is being transferred out of workloads.
- Or Network Anomaly forensic rule is incorrect or noisy.

To improve the score:

- Fix the issue if any to reduce the chances of data exfiltration.
- Adjust Network Anomaly rules to reduce noise and false alarms.

Network Anomaly score for a workload is inverse function of total severity score of Network Anomaly events. Higher is the total severity score, lower is the Network Anomaly score.

| Severity         | Score |
|------------------|-------|
| IMMEDIATE_ACTION | 100   |
| CRITICAL         | 10    |
| HIGH             | 5     |
| CRITICAL         | 3     |

Figure 406: Data Leak Score Formula

$$\text{data leak score} = \max(0, (100 - \sum \text{data leak event severity score}))$$

Refer to [PCR-Based Network Anomaly Detection](#) for more details.

Figure 407: Help for Data Leak Score

? Data Leak Score Help

**Supported Agent Types** 19 supported workloads

|                                                                                                                                                                      |                                                                                                                                                                      |                                                                                |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------|
| <p><span style="color: red; font-weight: bold;">✘</span> Universal Visibility (38)</p> <p><span style="color: green; font-weight: bold;">✔</span> AnyConnect (0)</p> | <p><span style="color: green; font-weight: bold;">✔</span> Deep Visibility (19)</p> <p><span style="color: red; font-weight: bold;">✘</span> Hardware Switch (0)</p> | <p><span style="color: green; font-weight: bold;">✔</span> Enforcement (0)</p> |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------|

**What is a Data Leak Score?**

A Data Leak Score gives you an assessment of whether there are any symptoms of unusually significant amounts of data being transmitted out of your workloads. Like all Security Scores, a higher score is better, with 0 meaning there is a workload that requires immediate action, and 100 meaning there are no Data Leak Events observed within this Scope.

**How is the Data Leak Score calculated?**

The Data Leak Score is also computed similarly to the Forensics Score. For each Workload we compute a Data Leak Score. A Workload's Data Leak Score is derived from the Data Leak Events observed on that Workload based on the profiles enabled for this scope. A score of 100 means no Data Leak Events were observed, and a score of 0 means there is a Data Leak Event detected that requires immediate action. The Data Leak Score for a Scope is the average Workload score within that Scope.

- A Data Leak Event with the severity CRITICAL reduces a workload's score with the weight of 10.
- A Data Leak Event with the severity HIGH reduces a workload's score with the weight of 5.
- A Data Leak Event with the severity MEDIUM reduces a workload's score with the weight of 3.
- A Data Leak Event with the severity LOW doesn't contribute to the Data Leak Score. This is recommended for new rules where the quality of the signal is still being tuned and is likely to be noisy.
- A Data Leak Event with the severity REQUIRES IMMEDIATE ACTION will reduce the Score for the entire Scope to zero.

**How do I improve my score?**

Tuning your Data Leak Score can be done by adjusting the Forensic Rules for Data Leak Events enabled for this Scope. Creating rules that are less noisy will give you a more accurate score. Acting upon and preventing legitimate Data Leak Events (events that are evidence of anomalous exfiltration activities) is another good way to improve your Data Leak Score.

**How do I increase the number of workloads with scores?**

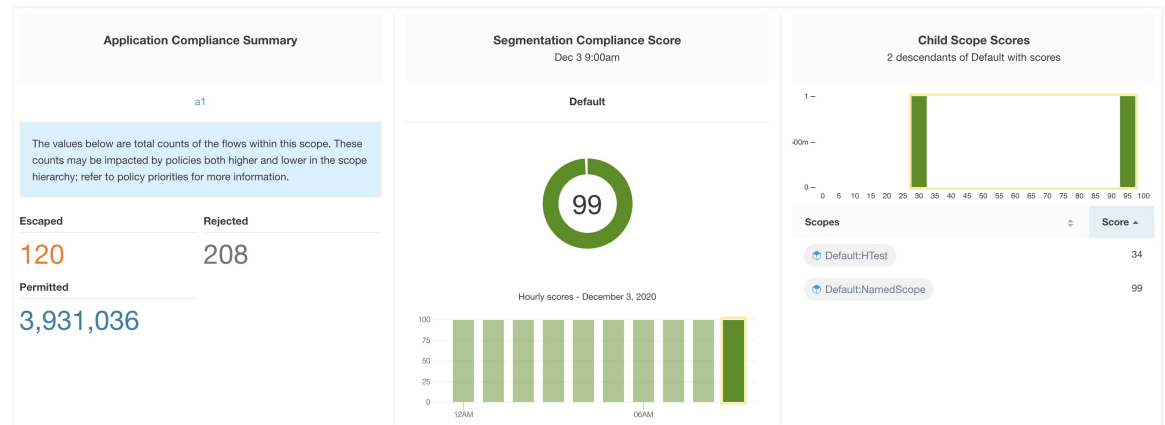
Data Leak Scores can only be calculated when Deep Visibility Sensors are present. Install Deep Visibility Sensors on more workloads to improve your score coverage.

## Segmentation Compliance Score

Segmentation Compliance Score presents a top-level view of policy violations and emphasizes which scopes and workspaces have the most violations.



Figure 408: Segmentation Compliance Score Details



**Note** Escaped/Rejected/Permitted count that is displayed on security dashboard for root scope does not add up to all the counts respectively displayed for all child scopes. Escaped/Rejected/Permitted count is an evaluation on the policy and not just on source or destination.

Lower score indicates:

- Significant number of escaped flows (policy violations) relative to permitted
- Score is 0 when more escaped flows than permitted.

Segmentation Compliance Score is computed for scopes with an enforced primary workspace. For scopes without enforced workspaces, the score will be computed as the average of descendant scope scores with enforced policies.

Score is computed by using the ratio between escaped and permitted.

Figure 409: Segmentation Compliance Score Formula

$$\text{compliance score} = \left[ 100 - \frac{100 \times \text{escaped}}{\text{permitted}} \right]$$

Improve score by reducing number of policy violations

- Verify policies correctly cover desired behavior.
- Verify that policies are correctly being enforced.

Figure 410: Help for Segmentation Compliance Score Details

**? Segmentation Compliance Score Help**

Supported Agent Types 5,059 supported workloads

|                                                                     |                                                                 |                                                             |
|---------------------------------------------------------------------|-----------------------------------------------------------------|-------------------------------------------------------------|
| <input checked="" type="checkbox"/> <b>Universal Visibility (8)</b> | <input checked="" type="checkbox"/> <b>Deep Visibility (23)</b> | <input checked="" type="checkbox"/> <b>Enforcement (25)</b> |
| <input checked="" type="checkbox"/> <b>AnyConnect (5,002)</b>       | <input checked="" type="checkbox"/> <b>Hardware Switch (1)</b>  |                                                             |

**What is a Segmentation Compliance Score?**  
A Segmentation Compliance Score is an indication of how effectively enforced Applications are based on observed Rejected and Escaped flows. Rejected and Escaped flows are a sign that enforcement isn't reliable and should be investigated. This score is only applicable if you have Applications with policies that are enforced.

**How is the Segmentation Compliance Score calculated?**  
Segmentation Compliance differs from the other modules in that the score applies only to Scopes and not to specific workloads. If the Scope has an enforced Application, the score is derived from the number of Rejected and Escaped flows relative to the total number of flows observed. The counts are displayed in the left pane, clicking them will take you to the enforced application view. For Scopes that don't have an enforced application, the score is the average of the child scope scores.

**How do I improve my score?**  
Investigating and reducing the number of Rejected and Escaped flows will improve and increase your Segmentation Compliance Score.

**How do I increase the number of Scopes with scores?**  
Create more Enforced Applications will increase your Segmentation Compliance coverage.



# CHAPTER 12

## View Vulnerability Dashboard

Cisco Secure Workload identifies and displays a list of the known Common Vulnerabilities and Exposures (CVE) across your workloads on the **Vulnerabilities** page. Using the displayed scores and the severity of the CVEs, you can focus your efforts on the most critical vulnerabilities and workloads that need most attention. Select a scoring system and the scope to view the CVEs according to the severity and other attribute details.

The different scoring systems used in Secure Workload are:

- **Common Vulnerability Scoring System (CVSS):** CVSS is a qualitative measurement of severity of the CVEs, from low to critical. The scores help you to prioritize responses for the most critical severities. CVSS V3 is the most recent version of the CVSS scoring mechanism.
- **Cisco Security Risk Score:** The Cisco Security Risk Score provides accurate risk assessments of the CVEs in your workloads. The risk scores help you to comprehend your organization's risk profile and help your security team prioritize remediation strategies.

**Table 36: Scoring Systems and Corresponding Attributes**

| Scoring System            | Attributes                                                                                                                                                                                                                                                             |
|---------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Cisco Security Risk Score | <ul style="list-style-type: none"><li>• Cisco Security Risk Score with Severity</li><li>• Active Internet Breach</li><li>• Easily Exploitable</li><li>• Fix Available</li><li>• Malware Exploitable</li><li>• Popular Target</li><li>• Predicted Exploitable</li></ul> |

| Scoring System | Attributes                                                                                                                                                                                                                                                                                                                          |
|----------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| CVSS V3        | <ul style="list-style-type: none"> <li>• CVE Score with Severity</li> <li>• Attack Complexity</li> <li>• Attack Vector</li> <li>• Availability Impact</li> <li>• Base Severity</li> <li>• Confidentiality Impact</li> <li>• Integrity Impact</li> <li>• Privileges Required</li> <li>• Scope</li> <li>• User Interaction</li> </ul> |
| CVSS V2        | <ul style="list-style-type: none"> <li>• CVE Score with Severity</li> <li>• Access Complexity</li> <li>• Access Vector</li> <li>• Authentication</li> <li>• Availability Impact</li> <li>• Confidentiality Impact</li> <li>• Integrity Impact</li> <li>• Severity</li> </ul>                                                        |

The dashboard highlights the distribution of vulnerabilities in the chosen scope and displays vulnerabilities by different attributes, for example, complexity of exploits, can the vulnerabilities be exploited over the network or does attacker need local access to the workload. Furthermore, the statistics can filter out vulnerabilities that are remotely exploitable and have lowest complexity to exploit.

The CVE threat databases in Secure Workload are updated every 24 hours by retrieving the latest CVE details from popular sources such as NIST, Microsoft, Oracle, and Cisco Vulnerability Management. If the Secure Workload cluster is in an air-gapped environment, the CVE threat data packs must be downloaded from <https://updates.tetrationcloud.com> and uploaded in Secure Workload.

The CVE threat databases in Secure Workload are updated every 24 hours by retrieving the latest CVE details from popular sources such as NIST, Microsoft, and Oracle. If the Secure Workload cluster is in an air-gapped environment, the CVE threat data packs must be downloaded from <https://updates.tetrationcloud.com> and uploaded in Secure Workload.

By using the scores and the required attributes of the known CVEs in your workloads, you can:

- Create inventory filters. See [Inventory Filters, on page 406](#).
- Configure microsegmentation policies to block the external communication from the impacted workloads and publish virtual patching rules to Cisco Secure Firewall Management Center.

Table 37: Feature Information

| Feature Name                                                                                                 | Release     | Feature Description                                                                                                                                                                                                                         | Where to Find                                                                                                               |
|--------------------------------------------------------------------------------------------------------------|-------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------|
| Integration of Cisco Vulnerability Management for Deep CVE Insights with Cisco Risk Score for Prioritization | 3.9 Patch 2 | You can use the Cisco Security Risk Scores of the CVEs to create inventory filters, microsegmentation policies to block communication from the impacted workloads, and virtual patching rules to publish the CVEs to Cisco Secure Firewall. | <a href="#">Vulnerability Dashboard, on page 699</a><br><a href="#">Cisco Security Risk Score-Based Filter, on page 410</a> |

- [Vulnerability Dashboard, on page 699](#)
- [CVEs Tab, on page 701](#)
- [Packages Tab, on page 702](#)
- [Workloads Tab, on page 703](#)
- [Pods Tab, on page 705](#)

## Vulnerability Dashboard

To view the Vulnerabilities page, from the navigation pane, choose **Investigate > Vulnerabilities**. The vulnerabilities identified using the different scoring system are displayed. The graphs and widgets display the number of vulnerabilities with the associated risk level and attributes depending on the scoring systems to identify workloads which requires immediate attention and the packages which needs to be patched immediately to reduce the risks.

Figure 411: Vulnerabilities Page

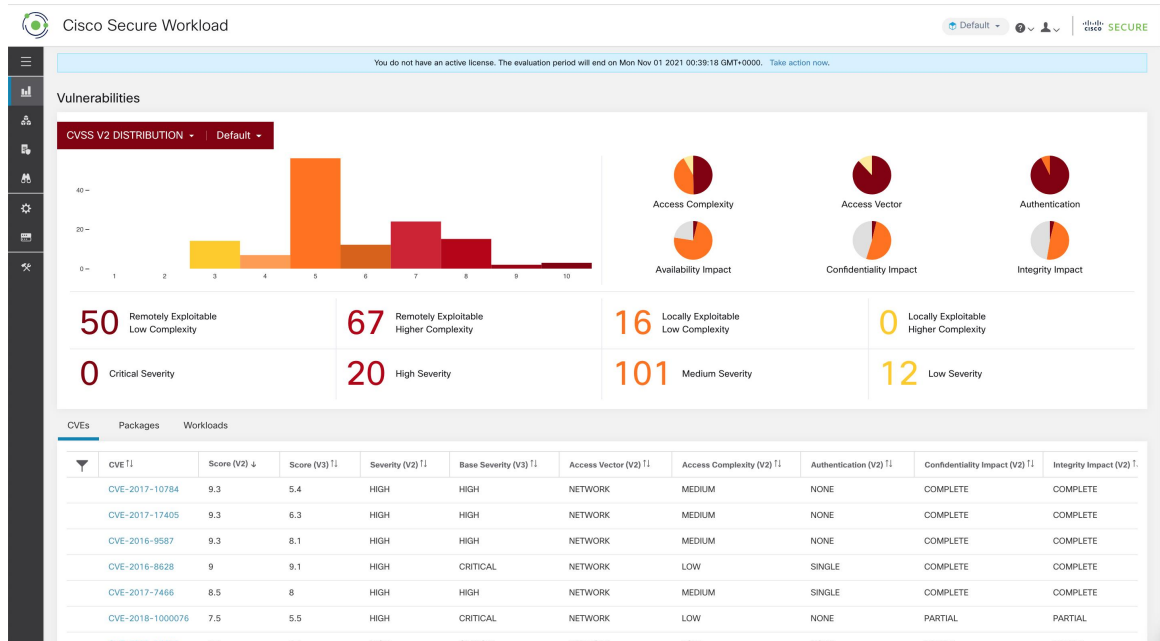
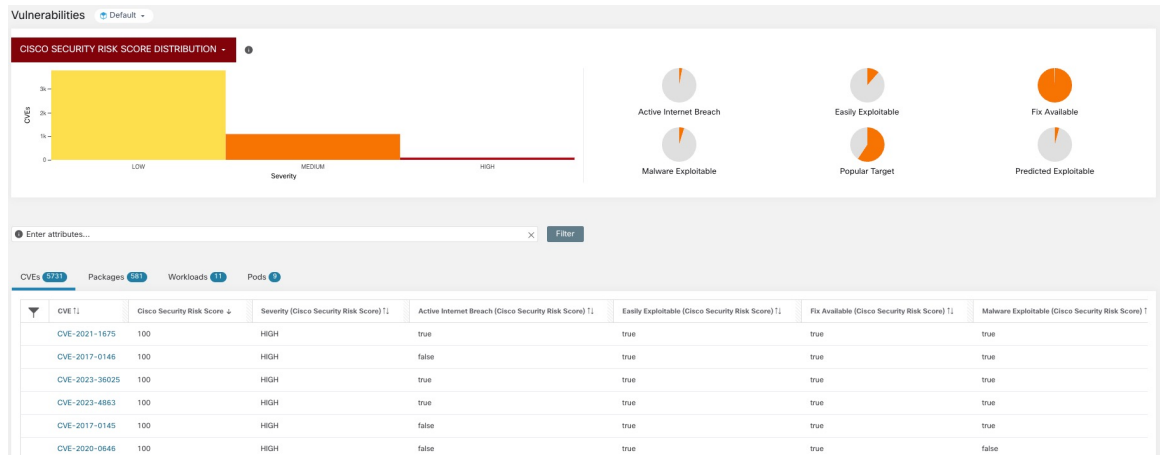


Figure 412: Vulnerabilities Page



The following tabs are filtered based on the selected portion of the graphs or widgets:

- The **CVEs** tab highlight the vulnerabilities that requires attention in the selected scope.
- The **Packages** tab lists the packages that must be patched.
- The **Workloads** tab lists the impacted workloads in the selected scope.
- The **Pods** tabs lists the impacted Kubernetes pods in the selected scope.

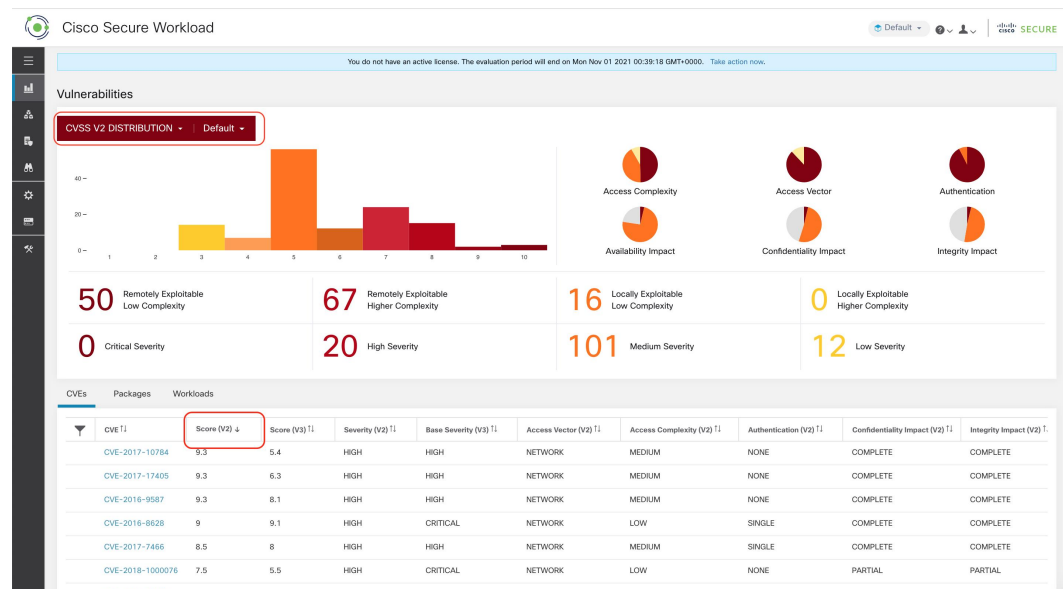
For details, click the required row in the tabs. For example, click a row in the Packages tab to view the workloads where the package or version is installed and the associated vulnerabilities for the package. The displayed lists can be downloaded as a JSON or CSV file using the download links.

# CVEs Tab

Based on the scoring system and selected scope, the CVEs tab lists the vulnerabilities identified on the workloads. For each CVE, besides basic impact metrics, exploit information based on Secure Workload's threat intelligence is displayed:

- **Exploit Count:** Number of times the CVE was seen exploited in the organizations in the previous year.
- **Last Exploited:** Last time the CVE was seen exploited in the organizations by Secure Workload's threat intelligence.

**Figure 413: CVEs Tab Listing Vulnerabilities in Specified Scope**



The graphs and pie chart can be used to filter the CVEs based on the severity or the required attributes of the scoring system. For example, if you click the Critical severity bar in any of the scoring system, the table will display only the workloads, packages, and pods containing the critical CVEs.

Click the required row under the CVEs tab to get more details on that vulnerability and the impacted workloads.

**Figure 414: Details for a CVE**

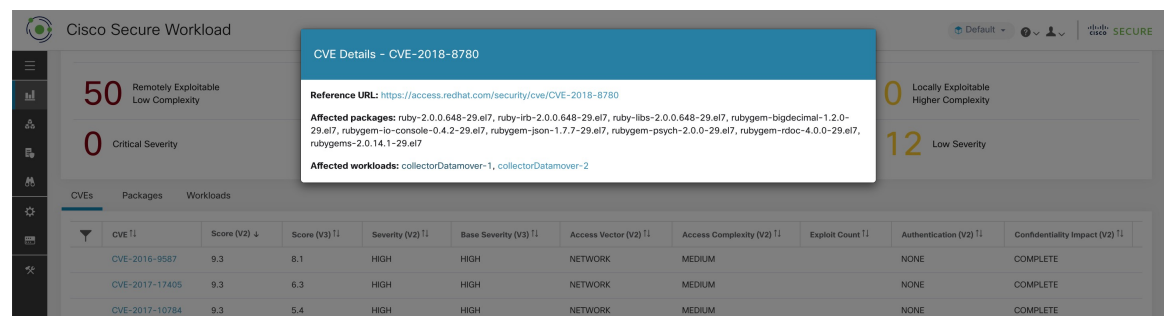
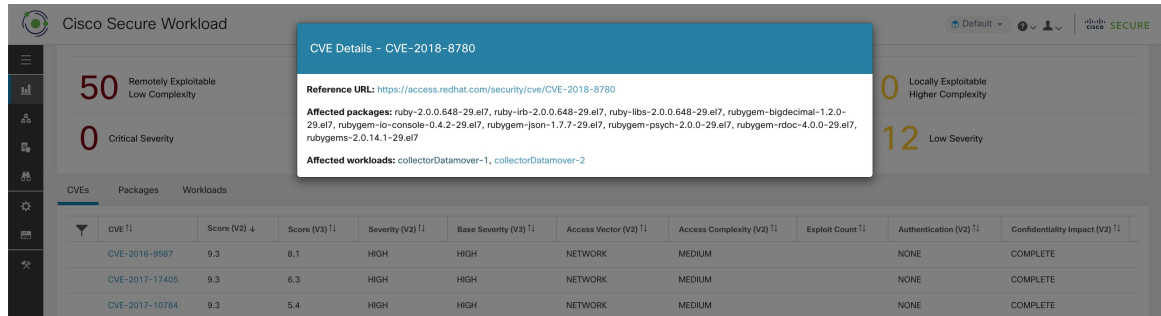


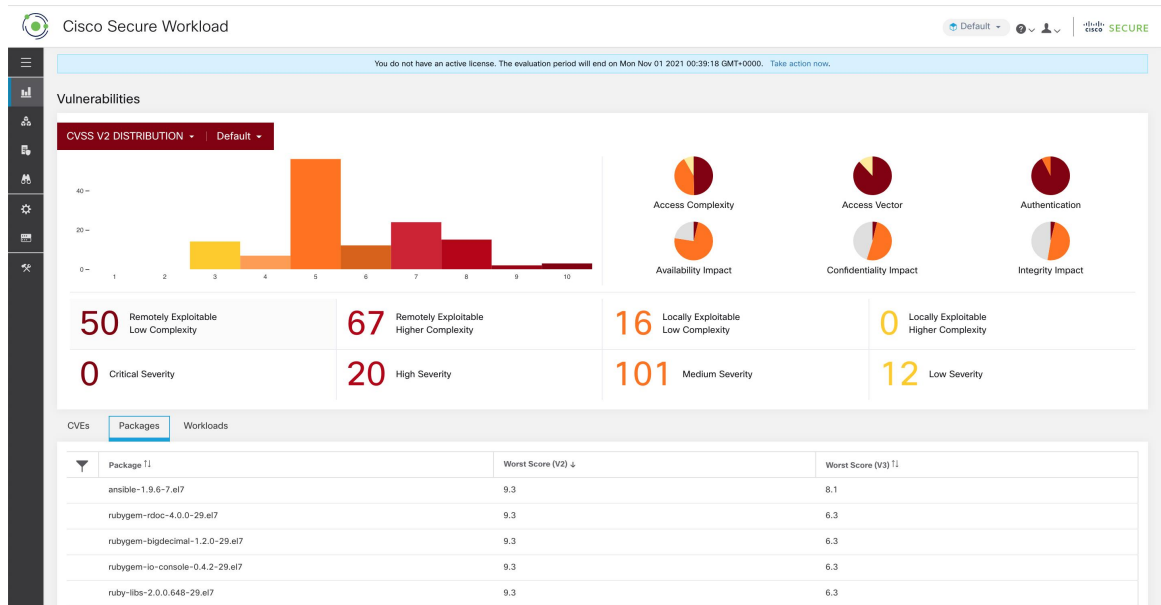
Figure 415: Details for a CVE



## Packages Tab

The Packages tab lists the impacted software packages that must be upgraded to reduce their attack surface.

Figure 416: Packages Tab Listing Vulnerable Software in Specified Scope



Click the required row under the Packages tab to get more details on impacted packages, the workloads with the packages, and the identified CVEs in the packages.



Figure 417: Details of Vulnerabilities and Affected Workloads for a Package



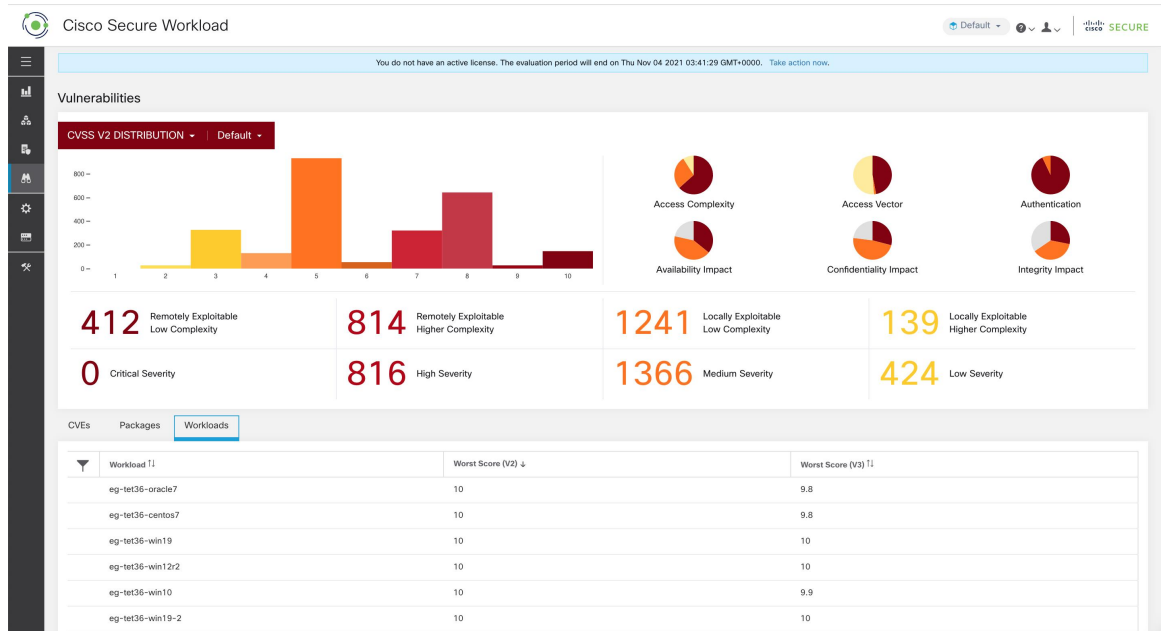
Figure 418: Details of Vulnerabilities and Affected Workloads for a Package



## Workloads Tab

The Workloads tab lists the workloads that require immediate attention in terms of software updates or patches.

Figure 419: Workloads Tab Listing Vulnerable Workloads in Specified Scope



Click the required row under the Workloads tab to get more details on vulnerable packages present in the selected workload. To view the workload profile, click the workload name next to the title of the dialog box.

Figure 420: Vulnerabilities Details in an Impacted Workload

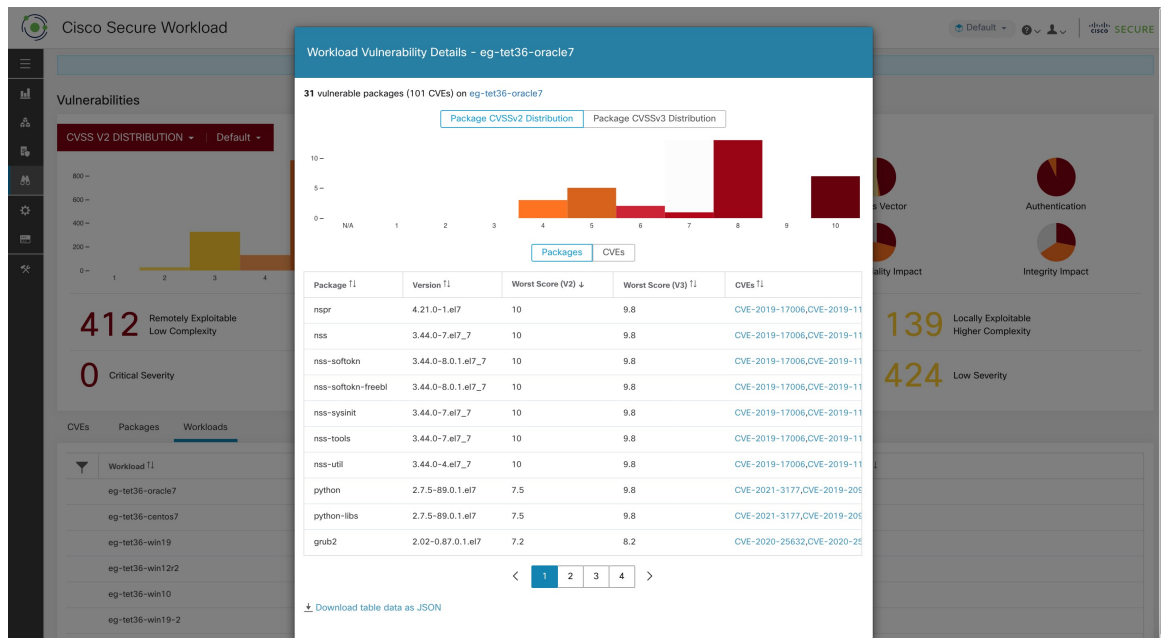
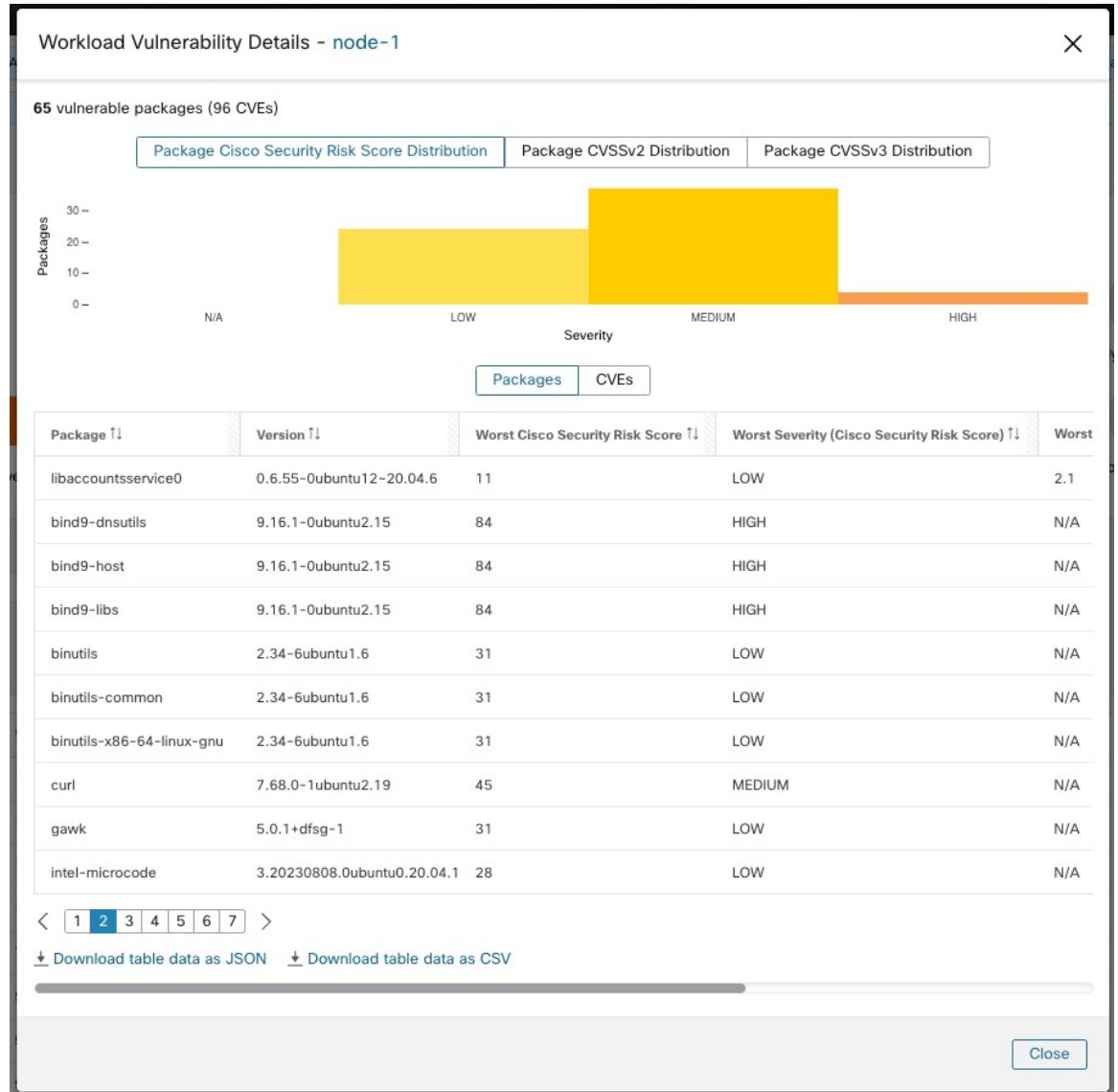


Figure 421: Vulnerabilities Details in an Impacted Workload

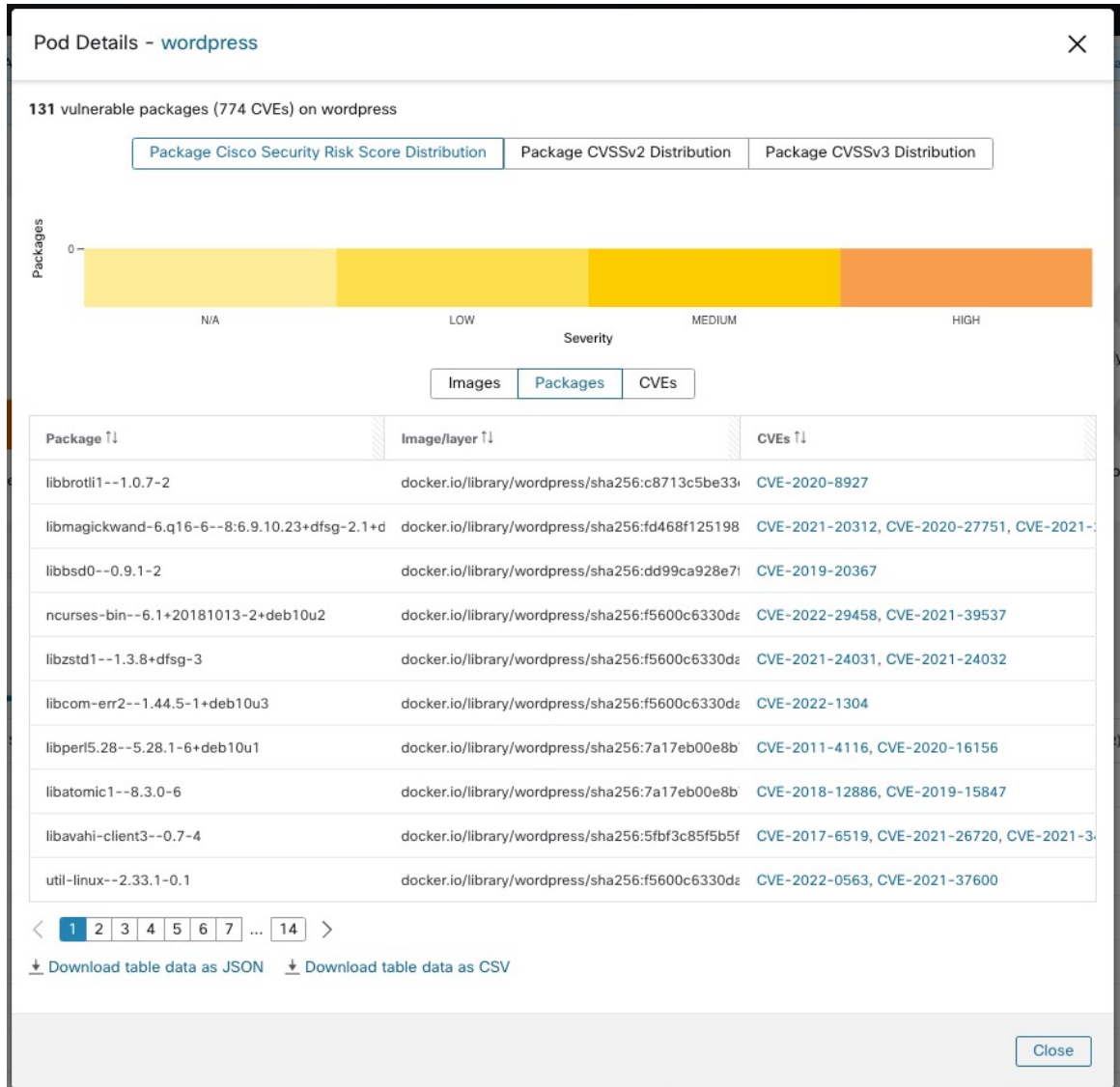


## Pods Tab

The Pods tab lists the Kubernetes pods that require immediate attention in terms of software updates or patches.

Click the required row under the Pods tab to get more details on impacted Kubernetes pod, packages, images, and the identified CVEs.

Figure 422: CVEs and Affected Packages in a Kubernetes Pod





# CHAPTER 13

## View Reporting Dashboard

Reporting dashboard, which is designed for Executives, Network Administrators, and Security Analysts, provides visual representations of critical workflow status, troubleshooting capabilities, and report creation functionalities. From the navigation pane, choose **Reporting > Reporting Dashboard** to access the dashboard.

**Table 38: Feature Information**

| Feature Name                                                                                                 | Release     | Feature Description                                                                                                                                                                                                                         | Where to Find                           |
|--------------------------------------------------------------------------------------------------------------|-------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------|
| Integration of Cisco Vulnerability Management for Deep CVE Insights with Cisco Risk Score for Prioritization | 3.9 Patch 2 | You can use the Cisco Security Risk Scores of the CVEs to create inventory filters, microsegmentation policies to block communication from the impacted workloads, and virtual patching rules to publish the CVEs to Cisco Secure Firewall. | <a href="#">Compliance, on page 716</a> |

- [Reporting Dashboard, on page 707](#)

## Reporting Dashboard

The sections below provide an overview of the reports and how to schedule and email reports.

### Schedule Email Reports

To generate a report, choose from either of the options:

- **Download:** After you generate a report, you can download and save a copy of the report for future reference.
- **Email:** If you choose the option to email reports, an email will be triggered to recipients with the attached report.
- **Schedule:** You have two options to choose from to schedule a report.

- Daily
- Weekly

To schedule a report, enter the schedule details to trigger the report. Select Weekly or Daily, enter the day and time, and the email addresses of the recipients. Click **Create Scheduled PDF** to save the details.




---

**Note** If the report scheduling fails, check the schedule for any incorrect email address or incorrectly entered date and time.

---

To access the report schedules that were generated earlier, choose **Generated Reports > Schedules**. If the report scheduling fails, check the schedule for any incorrect email address or incorrect date, time.




---

**Note** The maximum number of schedules that you can store in the scheduling dashboard is five.

---

## Overview

The overview section provides real-time insights into the network flow information, security policies, system performances, and security threats. It enables security analysts and network administrators to make informed decisions and take measures to protect their data resources.

### Segmentation Summary

Workspaces are the building blocks to discover, apply, and manage policies and enforcement within the cluster. You can define segmentation memberships by selecting the appropriate scope.

Segmentation summary captures the configuration details for every workspace, all policy-related activities, such as defining, analyzing, and enforcing policies for a particular scope in the workspace or workspaces that are associated with that scope.

The graph displays a summary of the various policies that are associated with the workspaces.

**Figure 423: Segmentation Summary**

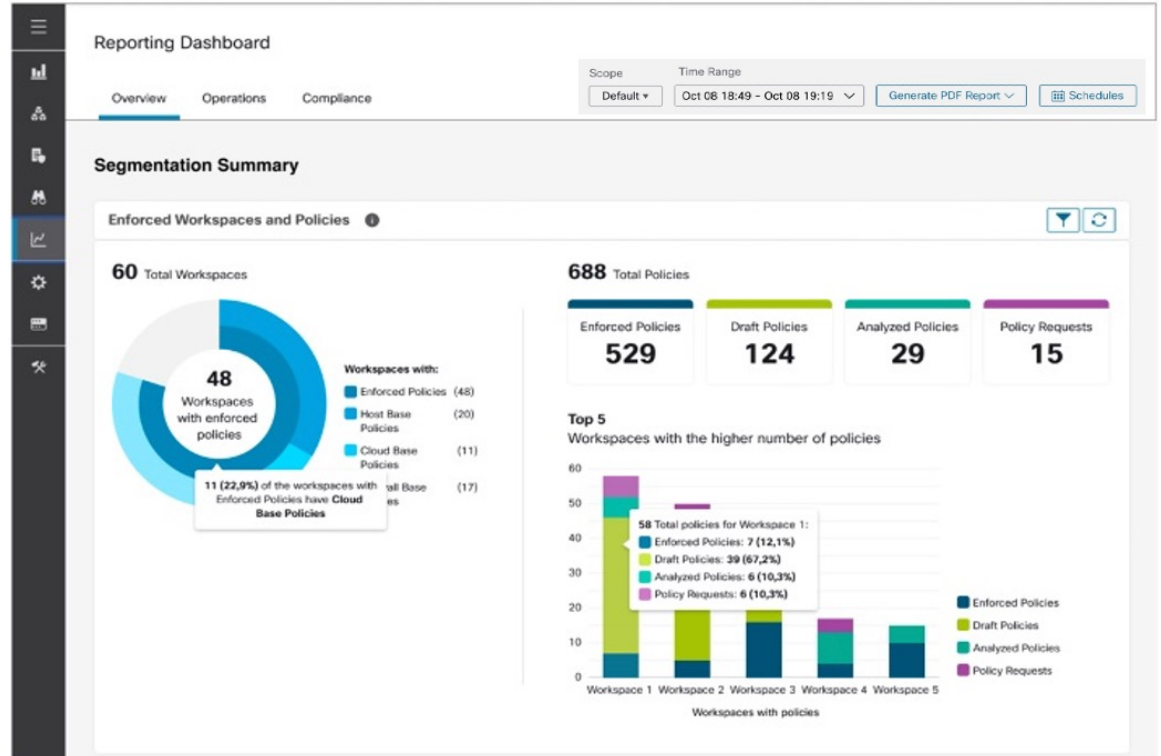
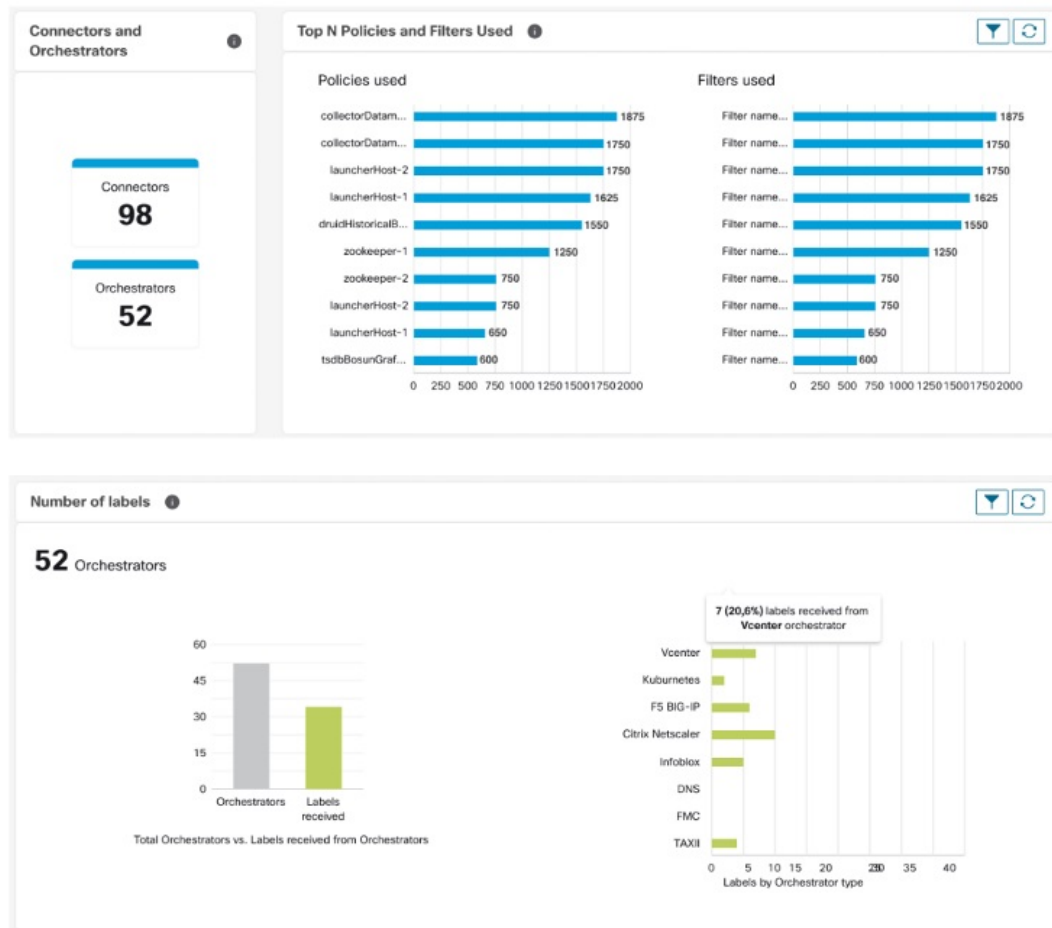


Figure 424: Connectors and Orchestrators



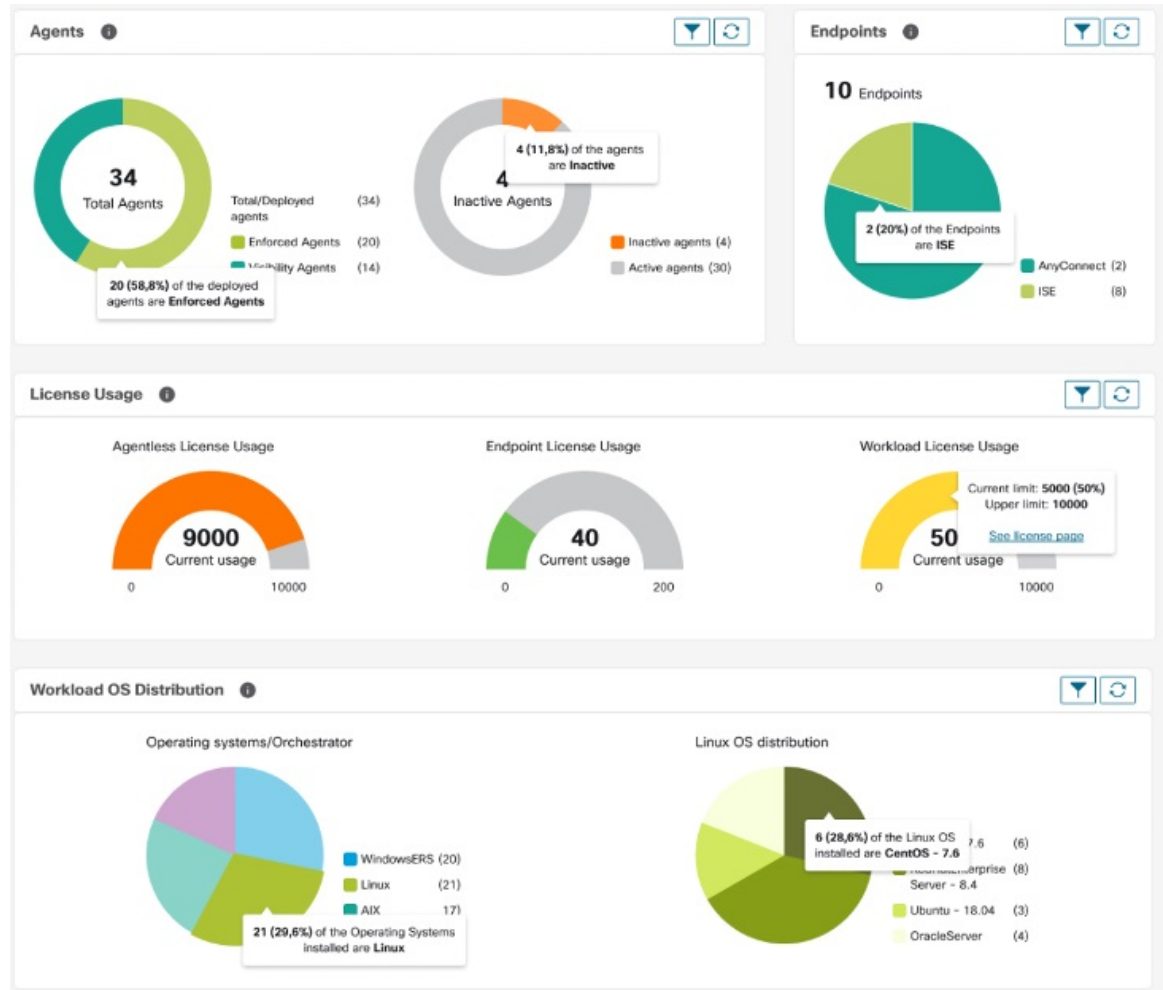
## Workload Summary

The Workload summary provides the following details about the agents that are deployed on one or more servers and endpoints in the infrastructure:

- Agents monitor and collect network flow information.
- Agents enforce security policies with firewall rules on the installed hosts.
- Agents communicate the status of the workload.
- Agents receive updates on the security policies.



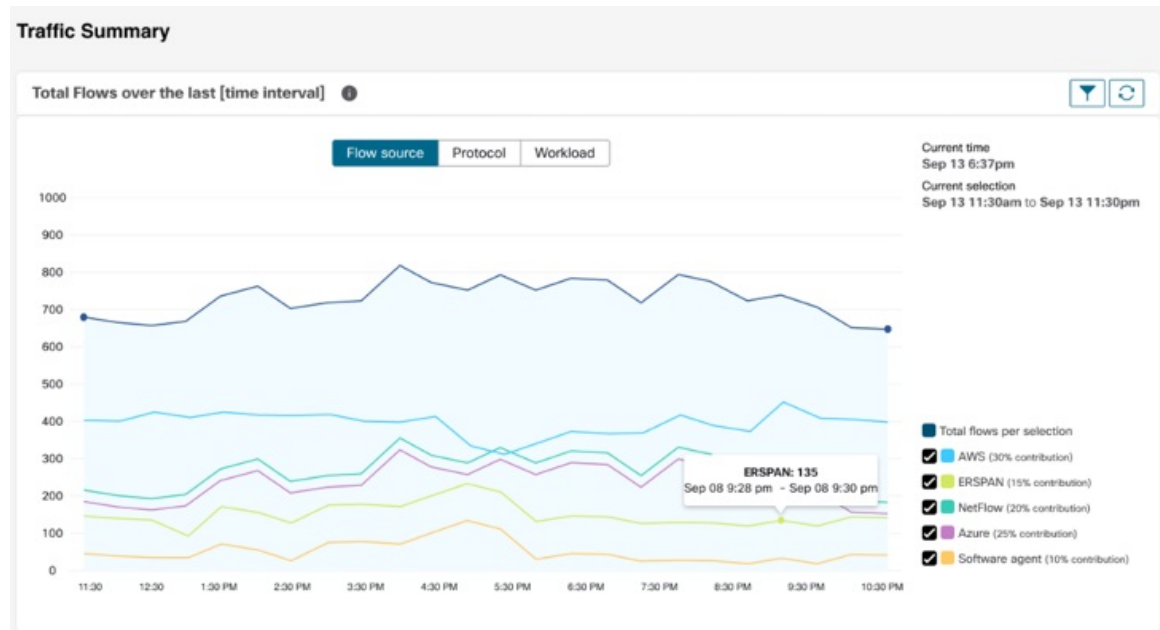
Figure 425: Workload Summary



## Traffic Summary

Traffic summary captures the flow observations of each flow. Each observation in the flow source tracks the number of packets, bytes, and other metrics for the flows.

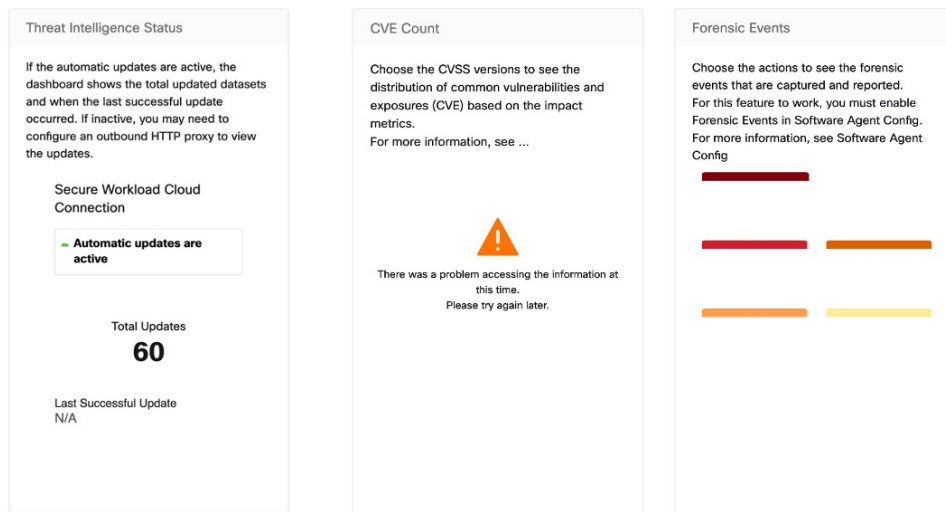
Figure 426: Traffic Summary



## Security Summary

Security Summary provides Threat Intelligence Status (last time when the threat intelligence status updates were received are shown), count of CVEs, and distribution of Forensic events.

Figure 427: Security Summary



# Operation

## Workload Summary

Workload summary provides a view of the total agents that are deployed on one or more servers and endpoints in the network. The agents monitor and collect network flow information, enforce security policies with firewall rules on the installed hosts, communicate the status of the workload, and receive updates on the security policies.

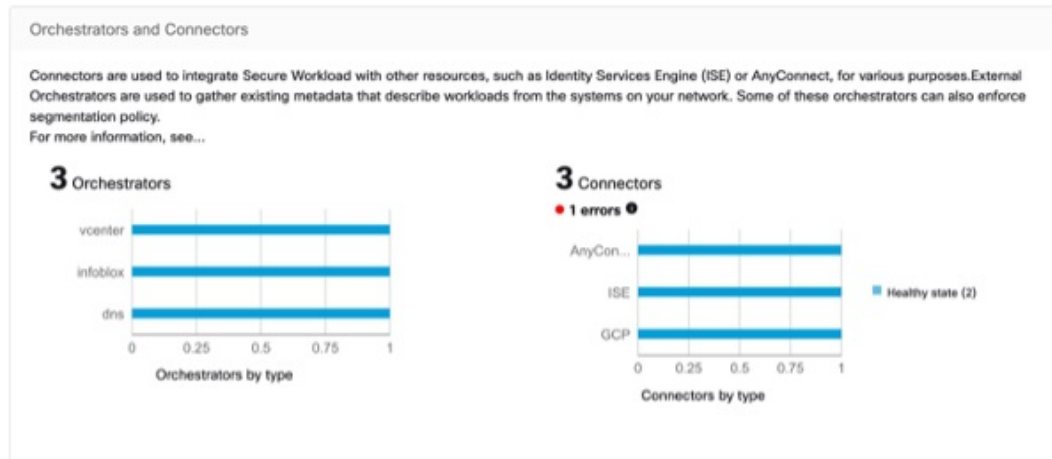
**Figure 428: Workload Summary**



## Telemetry Summary

Many connectors that are deployed on the virtual appliance collect telemetry from various points in the network, these connectors must listen on specific ports on the appliance. Connectors can ingest flow logs if you have setup flow logs for your specific security groups. You can also use the telemetry data for visualization and segmentation policy generation.

Figure 429: Telemetry Summary



### Cluster Summary

Site admins can access the cluster status page, but the actions can be carried out only by Customer Support users. It shows the status of all the physical servers in Cisco Secure Workload rack.

The processing and retention time for clusters refer to the duration for which data is stored and processed within a cluster. The specific processing and retention times depend on the requirements of the workload and the policies of the organization.

It is important to consider the processing time requirements when configuring the cluster, as this can impact the storage capacity and processing power that is needed to meet the workload's needs.

Retention time refers to the length of time that data is retained within a cluster. For some workloads, data may need to be retained for regulatory or compliance purposes, while for others it may be deleted once it has been processed. It is important to establish retention policies for the workload to ensure that data is retained for the appropriate length of time and then deleted securely to prevent unauthorized access.

Figure 430: Cluster Summary



### Segmentation Summary

Segmentation or Application Workspaces are the building blocks for discovering, applying, and managing policy and enforcement within the cluster. The segmentation summary captures the configuration details for each of the Application Workspaces implemented, the no. of workspaces with and without enforcement, policies that have been enabled or disabled, workspaces that have up-to-date policies or out of sync, and with or without draft policies.

Figure 431: Segmentation Summary

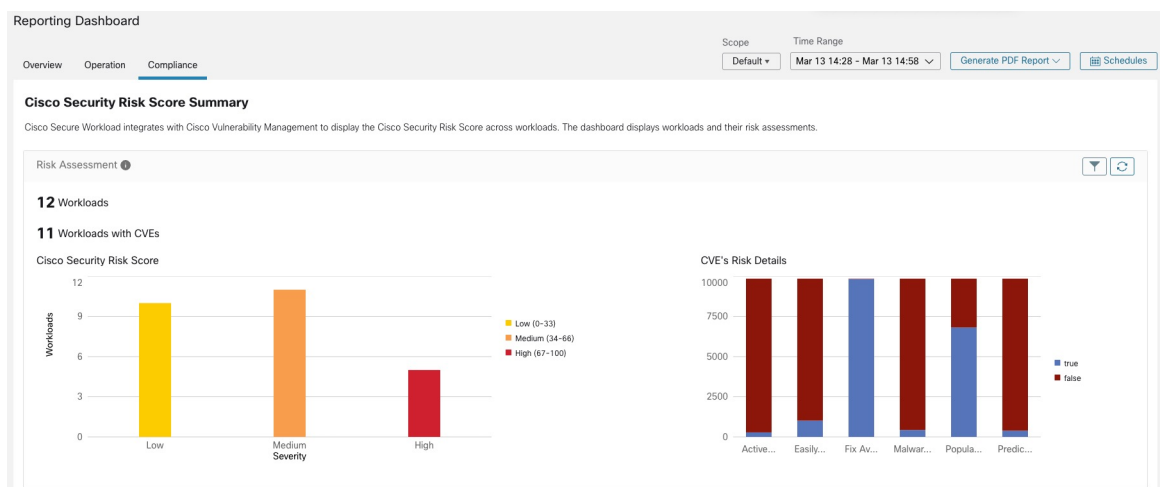


## Compliance

### Cisco Security Risk Score Summary

Cisco Security Risk Score Summary provides the risk assessment of your workloads based on the Cisco Security Risk scores. The **Cisco Security Risk Score** graph displays the number of workloads with high, medium, and low severity CVEs. The **CVE's Risk Details** graph displays the number of CVEs for each of the supported attributes of Cisco Vulnerability Management.

Figure 432: Cisco Security Risk Score Summary

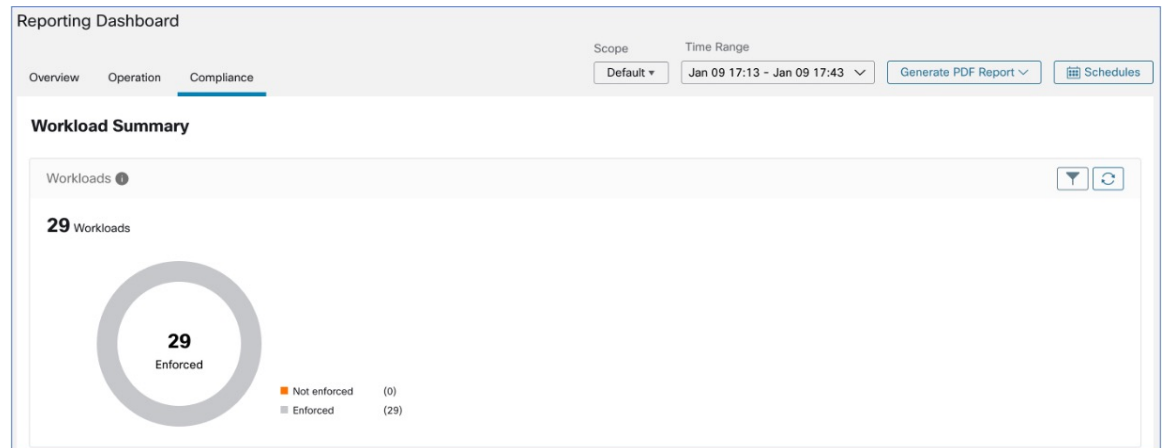


The downloaded compliance report lists top 10 workloads with CVEs of high severity status based on the Cisco Security Risk Score.

### Workload Summary

Workload Summary provides a view of the total agents that are deployed on one or more servers and endpoints in the infrastructure. The agents monitor and collect network flow information, enforce security policies with firewall rules on the installed hosts, communicate the status of the workload and receive updates on the security policies.

**Figure 433: Workload Summary**



### Security Summary

Configure your forensic events; once configured, all tactics are displayed without any rules under them, with a count of 0. Select one or more forensic rules to make the selection at the tactic level. Selecting a tactic selects all the rules under it. Default MITRE ATT&CK rules are provided to alert techniques from the MITRE ATT&CK Framework.

**Figure 434: Security Summary**



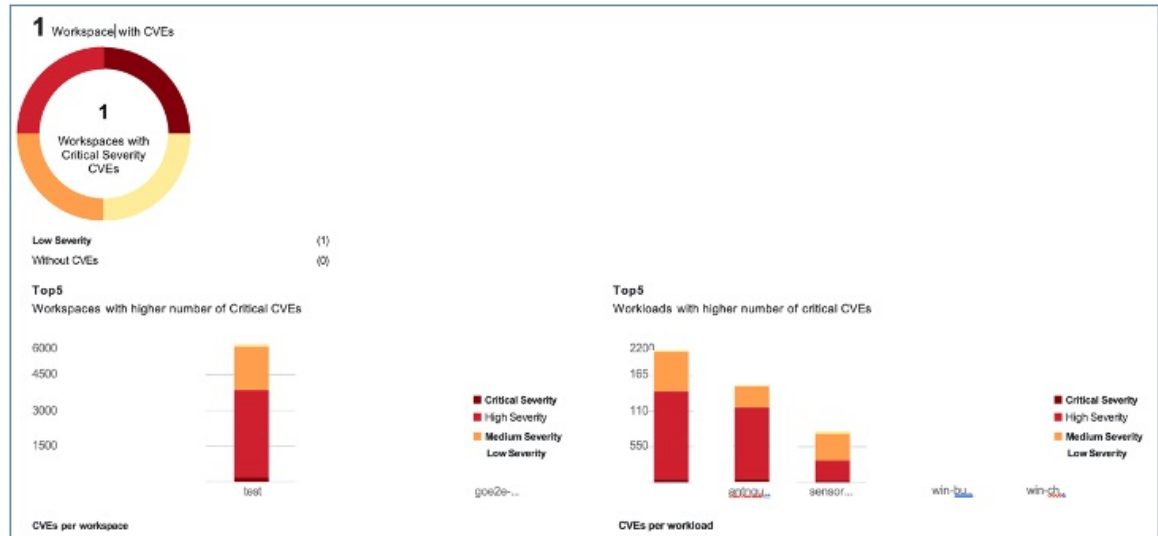
### Workspaces with CVEs

Based on the scope that is selected and the scoring system (v2 or v3), the common vulnerabilities and exposures (CVE) count highlights the vulnerabilities (sorted by the scores) on workloads in the selected scopes. See the distribution of workspaces and the workloads with the highest number of critical CVEs.

Software packages on a workload could potentially be associated with known vulnerabilities (CVE). Common Vulnerability Scoring System (CVSS) is used for assessing the impact of a CVE. CVE can have CVSS v2 and CVSS v3 score. To compute the vulnerability score, consider CVSS v3 if it is available, else CVSS v2 is considered.

Vulnerability score for a workload is derived from scores of vulnerable software that is detected on that workload. The Workload Vulnerability Score is calculated based on the CVSS scores, the vendor data, and the security research team may adjust when the data is missing or inaccurate. Higher the severity of the most severe vulnerability, lower is the score.

**Figure 435: Workspaces with CVEs**







## CHAPTER 14

# Set up System Configurations in Secure Workload

---

System-level settings are available to you depending on your role. For example, only users with **Site Administrator** and **Customer Support user** role, can view the **Users** option.

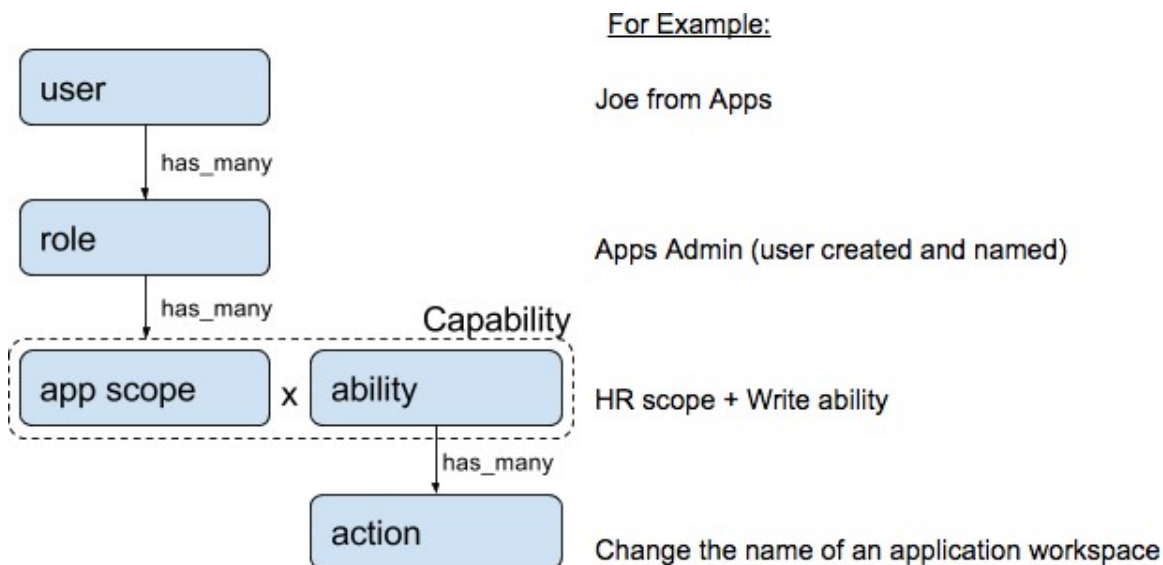
- [Roles, on page 719](#)
- [Change Log, on page 726](#)
- [Collection Rules, on page 727](#)
- [Session Configuration, on page 728](#)
- [Idle Session, on page 728](#)
- [Preferences, on page 729](#)
- [Scopes, on page 732](#)
- [Users, on page 732](#)

## Roles

You can restrict access to features and data using role-based access control (RBAC) model.

- User - someone with login access to Cisco Secure Workload.
- Role - user created set of capabilities that is assigned to a user.
- Capability - scope + ability pair
- Ability - collections of actions
- Action - low-level user action such as “change workspace name”

Figure 436: Role Model



A user can have any number of roles. Roles can have any number of capabilities. For example, the “HR Search Engineer” role could have two capabilities: “Read on the HR Scope” to give visibility and context and “Execute on “HR:Search” capability to allow the engineers assigned this role to make specific changes that are related to their applications.

Use the **Users** page to assign users to the different roles. Roles have several capabilities and you can assign users to any number of roles.

System roles are defined to allow users to get started more quickly. They define different levels of access to **all Scopes**, that is, all data on the system. These system roles are defined below.

| Role            | Description                                                                                                                                                        |
|-----------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Agent Installer | Provide the ability to manage agents life cycle including install, monitor, upgrade, and convert, but <b>cannot</b> delete agents and access agent config profile. |

## Abilities and Capabilities

Roles are made up of capabilities which include a scope and an ability. These define the allowed actions and the set of data that they apply to. For example, the (HR, Read) capability should be read and interpreted as “Read ability on the HR scope”. This capability would allow access to the HR scope and all its children.

| Ability   | Description                                                        |
|-----------|--------------------------------------------------------------------|
| Installer | Install, monitor, and upgrade software agents.                     |
| Audit     | Global appliance data read support and access to change logs.      |
| Read      | Read all data including flows, application, and inventory filters. |
| Write     | Make changes to applications and inventory filters.                |

| Ability | Description                                                                                           |
|---------|-------------------------------------------------------------------------------------------------------|
| Execute | Perform Automatically discover policies run and publish policies for analysis.                        |
| Enforce | Enforce policies that are defined in application workspaces that are associated with the given scope. |



**Important** Abilities are inherited, for example, the Execute ability allows all the Read, Write, and Execute actions.



**Important** Abilities apply to the scope and all the scope's children.

## Menu Access by Role

The menu items that you see and use on the navigation pane depend on the assigned role:

**Table 39: Overview Menu**

| Menu     | Option   | Tenant Owner | Agent Installer |
|----------|----------|--------------|-----------------|
| Overview | Overview | Yes          | No              |

**Table 40: Organize Menu**

| Menu     | Option               | Tenant Owner | Agent Installer |
|----------|----------------------|--------------|-----------------|
| Organize | Scopes and Inventory | Yes          | No              |
| Organize | Use Uploaded Labels  | Yes          | No              |
| Organize | Inventory Filters    | Yes          | No              |

**Table 41: Defend Menu**

| Menu   | Option             | Tenant Owner | Agent Installer |
|--------|--------------------|--------------|-----------------|
| Defend | Segmentation       | Yes          | No              |
| Defend | Enforcement Status | Yes          | No              |

| Menu   | Option           | Tenant Owner | Agent Installer |
|--------|------------------|--------------|-----------------|
| Defend | Policy Templates | Yes          | No              |
| Defend | Forensic Rules   | Yes          | No              |

Table 42: Investigate Menu

| Menu        | Option          | Tenant Owner | Agent Installer |
|-------------|-----------------|--------------|-----------------|
| Investigate | Traffic         | Yes          | No              |
|             | Alerts          | Yes          | No              |
|             | Vulnerabilities | Yes          | No              |
|             | Forensics       | Yes          | No              |

Table 43: Reporting Menu

| Menu      | Option              | Tenant Owner | Agent Installer |
|-----------|---------------------|--------------|-----------------|
| Reporting | Reporting Dashboard | Yes          | No              |

Table 44: Manage Menu

| Menu   | Option                 | Tenant Owner | Agent Installer |
|--------|------------------------|--------------|-----------------|
| Manage | Agents                 | Yes          | Yes             |
| Manage | Alerts Configs         | Yes          | No              |
| Manage | Change Logs            | Yes          | No              |
| Manage | Connectors             | Yes          | No              |
| Manage | External Orchestrators | Yes          | No              |
| Manage | Secure Connector       | Yes          | No              |
| Manage | Virtual Appliances     | Yes          | No              |

| Menu   | Option                | Tenant Owner | Agent Installer |
|--------|-----------------------|--------------|-----------------|
| Manage | Users                 | Yes          | No              |
| Manage | Roles                 | Yes          | No              |
| Manage | Collection Rules      | Yes          | No              |
| Manage | Session Configuration | Yes          | No              |
| Manage | Usage Analytics       | Yes          | No              |
| Manage | Data Tap Admin        | Yes          | No              |

## Create a Role

### Before you begin

You must already have a **Site Admin** or **Customer Support** user role.

1. In the navigation bar on the left, click **Manage > User Access > Roles**.
2. Click **Create New Role**. The **Roles** panel appears.

Creating a role using the Create Role Wizard is three-step process.

### Procedure

- Step 1**
- a) Enter the appropriate values in the following fields:

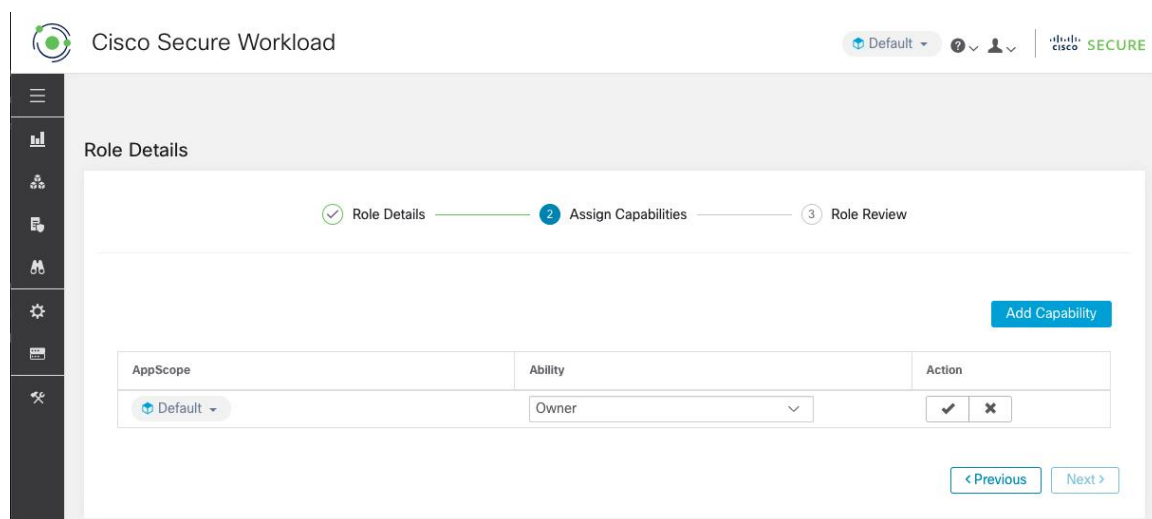
| Field              | Description                                        |
|--------------------|----------------------------------------------------|
| <b>Name</b>        | The name to identify the role.                     |
| <b>Description</b> | A short description to add context about the role. |

b) Click the **Next** button to move to the next step or **Back to Roles Page** to go back to Roles Page.

### Step 2

- Click the **Add Capability** button to show the creation form in the top row.
- Select scope and ability.
- Click the **Checkmark** button to create a new capability or **Cancel** button to cancel.
- Click **Next** to review role details or **Previous** to go back and edit.

*Figure 437: Capability Assignment*



### Step 3

- Review the role details and capabilities.
- Click **Create** to create role.

Figure 438: Role Review

Cisco Secure Workload

Default ?

SECURE

Role Details

Role Details — Assign Capabilities — 3 Role Review

Role Details

|             |                               |
|-------------|-------------------------------|
| Name        | Site Engineer                 |
| Description | Secure Workload Site Engineer |
| Show All?   | <input type="radio"/> No      |

Capabilities

|         |         |
|---------|---------|
| Scope   | Ability |
| Default | Owner   |

< Previous Create

## Edit a Role

This section explains how **Site Admins** and **Customer Support users** can edit roles.

### Before you begin

You must be Site Admin or Customer Support User.

1. In the navigation bar on the left, click **Manage > User Access > Roles**.
2. In the row of the role to edit, click the **Edit** button in the right-hand column. The **Roles** panel appears.

Editing a role using the Edit Role Wizard is three-step process.

### Procedure

- Step 1**
- a) Update the name or description if desired.
  - b) Click the **Next** button to move to the next step or **Back to Roles Page** to go back to Roles Page.
- Step 2**
- a) Remove any capability as needed. In the row of the capability to delete, click the **Delete** icon in the right-hand column.
  - b) To add, click the **Add Capability** button to show the creation form in the top row.
  - c) Select scope and ability.
  - d) Click **Next** to review role details or **Previous** to go back and edit.

- Step 3**
- Review the role details and capabilities.
  - Click **Update** to create the role or **Previous** to go back and edit. Changes to role details and capability assignment are saved after **Update**.

**Note** Capabilities cannot be edited, they must be deleted and recreated.

## Change Log

**Site Admins** can access the **Change Log** page under the **Manage** menu in the navigation bar at the left side of the window. This page displays the most recent changes that are made within Cisco Secure Workload.



**Note** **Change Log Retention Period:** Secure Workload manages change logs for a duration of up to one year on both SaaS and On-premises clusters. An hourly job deletes change logs that exceed a one-year timeframe.

**Figure 439: Change Log Page**

| Change At                     | Type       | Action  | Details | Change By |
|-------------------------------|------------|---------|---------|-----------|
| May 24 2019 03:05:06 pm (PDT) | Capability | create  |         | N/A ⓘ     |
| May 24 2019 03:05:06 pm (PDT) | Capability | destroy |         | N/A ⓘ     |
| May 24 2019 03:05:06 pm (PDT) | Capability | create  |         | N/A ⓘ     |
| May 24 2019 03:05:06 pm (PDT) | Capability | destroy |         | N/A ⓘ     |
| May 24 2019 03:05:06 pm (PDT) | Capability | create  |         | N/A ⓘ     |
| May 24 2019 03:05:06 pm (PDT) | Capability | destroy |         | N/A ⓘ     |

The details of each change log entry can be viewed by clicking on the link in the **Change At** column. This page includes a **Before** and **After** snapshot of the fields changed. The fields may include technical names that require some interpretation to understand how they are surfaced elsewhere throughout Secure Workload.

**Figure 440: Change Log Details Page**

| Change Log Details for Capability (60f1dc0e497d4f4854625b69) |                                                                                                                | Full log for this Capability » |
|--------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------|--------------------------------|
| Version                                                      | 1                                                                                                              |                                |
| Change At                                                    | Jul 16 2021 10:20:46 pm (EEST)                                                                                 |                                |
| Change By                                                    | N/A ⓘ                                                                                                          |                                |
| Action                                                       | create                                                                                                         |                                |
| Before                                                       |                                                                                                                |                                |
| After                                                        | <pre>app_scope_id: 60f1dc0e497d4f4854625b65 ability: "AGENT_INSTALLER" role_id: 60f1dc0e497d4f4854625b67</pre> |                                |



The complete list of changes for an entity can be viewed by clicking the button in the upper-right corner, titled **Full log for this <entity type>**. This page displays the details of each change. It also includes the **Current State** of the entity, when available.

**Figure 441: Full Change Log for Entity**

| Change Log for Capability (60f1dc0e497d4f4854625b69)                                                                                                           |                                                                                                                |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------|
| Current State                                                                                                                                                  |                                                                                                                |
| <pre>id: "60f1dc0e497d4f4854625b69" app_scope_id: 60f1dc0e497d4f4854625b65 role_id: 60f1dc0e497d4f4854625b67 ability: "AGENT_INSTALLER" inherited: false</pre> |                                                                                                                |
| Version                                                                                                                                                        | 1                                                                                                              |
| Change At                                                                                                                                                      | Jul 16 2021 10:20:46 pm (EEST)                                                                                 |
| Change By                                                                                                                                                      | N/A                                                                                                            |
| Action                                                                                                                                                         | create                                                                                                         |
| Before                                                                                                                                                         |                                                                                                                |
| After                                                                                                                                                          | <pre>app_scope_id: 60f1dc0e497d4f4854625b65 ability: "AGENT_INSTALLER" role_id: 60f1dc0e497d4f4854625b67</pre> |

## Collection Rules

**Site Admins** and **Customer Support users** can access the **Collection Rules** page under the **Manage > Service Settings** menu in the navigation bar at the left side of the window. This page displays the hardware collection rules by VRF that is used by switches running the Cisco Secure Workload agent. There is a row in the table for each VRF.

## Rules

Click the **Edit** button on a VRF to modify its collection rules. By default, every VRF is configured with two default catch-all rules, one for IPv4 (`0.0.0.0/0 INCLUDE`) and one for IPv6 (`::/0 INCLUDE`). *These default rules can be removed, but do so with caution.*

Extra include and exclude rules can be added. Enter a valid subnet, select include or exclude, and click **Add Rule**. The priority of these rules can be adjusted via drag-and-drop. Click-and-hold on a rule in the list and drag it to adjust the order.

Changes may take several minutes to propagate to your switches. Click the **Back** button in the upper-right corner to return to the VRF list.

## Priority

Collection Rules are ordered in decreasing order priority. No longest prefix match is done to determine the priority. The rule appearing first has higher priority over all the subsequent rules. Example:

1. 1.1.0.0/16 INCLUDE
2. 1.0.0.0/8 EXCLUDE

### 3. 0.0.0.0/0 INCLUDE

In the earlier example, all addresses belonging to 1.0.0.0/8 subnet are excluded except subnet 1.1.0.0/16 which is included.

Another Example with changed order:

1. 1.0.0.0/8 EXCLUDE
2. 1.1.0.0/16 INCLUDE
3. 0.0.0.0/0 INCLUDE

In the above example, all addresses belonging to 1.0.0.0/8 subnet are excluded. Rule number-2 does not get exercised here because of a higher-order rule already defined for its subnet.

## Session Configuration

UI User Authentication idle session timeout can be configured here. This config applies to all the users of the appliance. The default idle session duration is 1 hour. The idle session duration can be set within the range of 5 minutes to 24 hours. The session timeout takes effect on a user's authenticated session when this value is saved.

**Site Admins** and **Customer Support users** can access this setting. In the left navigation pane, click **Manage > Service Settings > Session Configuration**.

## Idle Session

For those who are authenticating using a local database, this section explains how failed login attempts may lock the user account:

### Procedure

- 
- Step 1** Five failed login attempts using email and password result in locking the account.
- Note** As a security measure against probing, no specific message indicating the lock will be provided in the login interface when trying to sign in a locked account.
- Step 2** Lock out interval is set at 30 minutes. After the account is unlocked, use the correct password to log in or initiate password recovery by clicking *Forgot password?*
- Note** Once a user is successfully signed in, one hour of inactivity logs out the user. This timeout is configured from **Manage > Service Settings > Session Configuration**.
-

# Preferences

The **Preferences** page displays your account details and enables you to update your display preferences, change your landing page, change your password, and configure two-factor authentication.

## Change Your Landing Page Preference

To change the page you see when you sign in:

### Procedure

---

- Step 1** On the top-right corner of the window, click the user icon and choose **User Preferences**.
- Step 2** Choose a landing page from the drop-down menu. Your preference is saved as the default or home page when you log in. To see the change, click the Secure Workload logo at the top-left corner of the page.
- 

## Changing a Password

### Procedure

---

- Step 1** Click on the user icon in the top-right corner.
- Step 2** Select **User Preferences**.
- Step 3** In the **Change Password** pane, enter your current password in the **Old Password** field.
- Step 4** Enter your new password in the **Password** field.
- Step 5** Re-enter your new password in the **Confirm Password** field.
- Step 6** Click **Change Password** to submit the change.

**Note** Password must be 8–128 characters and contain at least one of the each following:

- Lower case letters ( a b c d . . . )
  - Upper case letters ( A B C D . . . )
  - Numbers ( 0 1 2 3 4 5 6 7 8 9 )
  - Special characters ( ! " # \$ % & ' ( ) \* + , - . / : ; < = > ? @ [ \ ] ^ \_ { | } ~ ), space included
- 

## Recovering Passwords

This section explains how to recover your password.

### Before you begin

To reset a password, you must first have an account. A new account can be added by **Site Admins** and **Customer Support users**.

### Procedure

---

**Step 1** Point your browser to the Cisco Secure Workload URL and click the **Forgot Password** link. The **Forgot your password?** dialog box is displayed.

**Step 2** Enter your email address in the **Email** field.

**Step 3** Click **Reset Password**.

Password reset instructions are sent to your email.

**Note** The password recovery procedure for two-factor authentication requires contacting Secure Workload Customer Support because the email-based password recovery cannot contain the one-time password.

---

## Enabling Two-Factor Authentication

This section explains how to enable two-factor authentication.

### Procedure

---

**Step 1** Click on the user icon in the top-right corner.

**Step 2** Select **User Preferences**.

**Step 3** In the **Two-Factor Authentication** pane, click the **Enable** button. A new **Two-Factor Authentication** pane appears.

**Step 4** Enter your password.

**Step 5** Scan the QR code that is displayed under the **Current Password** field using any time-based one-time password (TOTP) app, such as Google Authenticator (for Android or iOS) or Authenticator (for Windows Phone).

**Step 6** Enter the validation code that is shown by your chosen TOTP app.

**Step 7** Click **Enable**.

Figure 442: Two-Factor Authentication Pane

## Two-Factor Authentication



Two-factor authentication is disabled.

Current Password:

Scan QR Code:



Scan this code using any Time-based One-Time Password (TOTP) app, such as:

- Google Authenticator for [Android](#)  and [iOS](#) 
- Authenticator for [Windows Phone](#) 

Verify:

The next time that you log into the system, you must select the **Use two-factor authentication** check box and enter the verification code that is shown in your TOTP app to sign in.

**Note** The password recovery procedure for two-factor authentication requires contacting Secure Workload Customer Support because the email-based password recovery cannot contain the one-time password.

---

## Disabling Two-Factor Authentication

This section explains how to disable two-factor authentication.

### Procedure

---

- Step 1** Click on the user icon in the top-right corner.
- Step 2** Select **User Preferences**.
- Step 3** Under two-factor authentication, click the **Disable** button. The **Two-Factor Authentication** pane appears.
- Step 4** Enter your password.
- Step 5** Click the **Disable** button again.
- You will no longer be required to enter a two-factor verification code during login.
- 

## Scopes



**Note** The **Scopes** page is merged with **Inventory Search**. For more information, see the [Scopes and Inventory](#) page.

---

## Users

Site Admins and Root Scope Owners can access the **Users** page under the **Manage > User Access** menu from the navigation pane.

This page shows all Service Provider users and users associated with the scope on the page header.

### Multitenancy

To support multitenancy, assign users to a root scope. Users with the **Owner** ability on the root scope manage these users and assign roles that are associated with the same scope.

Service Providers users are without a scope; users are assigned to roles that allow them to perform actions across root scopes.

## Add a User

### Before you begin

- You must be a **Site Admin** or **Scope Owner** user to add users in Secure Workload.
- If a user is assigned a scope for multitenancy, only roles that are assigned to the same scope may be selected.



**Note** This page is filtered by the scope preference that is selected on the page header.

---

## Procedure

- Step 1** If applicable, select the appropriate root scope from the page header.
- Step 2** From the navigation pane, choose **Manage > User Access > Users**.
- Step 3** Click **Create New User**.  
The **User Details** page is displayed.
- Step 4** Update the following fields under **User Details**.

**Table 45: User Details Field Descriptions**

| Field             | Description                                                                                                                   |
|-------------------|-------------------------------------------------------------------------------------------------------------------------------|
| <b>Email</b>      | Enter the email ID of the user. It is non case-sensitive. We use the lower case version of your email if it contains letters. |
| <b>First Name</b> | Enter the user's first name.                                                                                                  |
| <b>Last Name</b>  | Enter the user's last name.                                                                                                   |
| <b>Scope</b>      | Root scope that is assigned to the user for multitenancy. (Available to site admins)                                          |

- Step 5** Click **Next**.
- Step 6** Under **Assign Roles**, add or remove assigned roles to the user.
- Click **Add Roles** to assign new roles, and then click the **Add** check box.

**Figure 443: Assigned User Roles**

The screenshot shows the 'User Details' page in Cisco Secure Workload. The 'Assign Roles' step is active, showing a list of available roles. The 'Customer Support' role is selected, indicated by a checked checkbox.

| Add                                 | Name ↑↓                                               | Tenant ↑↓        | Capability                  | Users |
|-------------------------------------|-------------------------------------------------------|------------------|-----------------------------|-------|
| <input type="checkbox"/>            | Agent Installer - Install, Monitor and Upgrade Agents | Unknown          | AGENT_INSTALLER   Unknown   | 0     |
| <input type="checkbox"/>            | Agent Installer - Install, Monitor and Upgrade Agents | Default          | AGENT_INSTALLER   Default   | 3     |
| <input type="checkbox"/>            | Agent Installer - Install, Monitor and Upgrade Agents | Tetration        | AGENT_INSTALLER   Tetration | 0     |
| <input type="checkbox"/>            | Agent Installer - Install, Monitor and Upgrade Agents | Tenant           | AGENT_INSTALLER   Tenant    | 0     |
| <input checked="" type="checkbox"/> | Customer Support - Technical Support or Advanced Ser  | Service Provider | OWNER   All Scopes          | 8     |

- Select the assigned roles, click **Edit Assigned Roles**, and then click the **Remove** icon.
- You can filter the user roles using **Name** or **Tenant**.

Figure 444: Filter User Roles

The screenshot shows the 'User Details' page in Cisco Secure Workload. The page has a navigation sidebar on the left and a main content area. At the top, there is a notification: 'You do not have an active license. The evaluation period will end on Mon Nov 01 2021 00:39:18 GMT+0000. Take action now.' Below this, the 'User Details' section is active, showing a progress bar with three steps: 'User Details' (checked), 'Assign Roles' (current step), and 'User Review'. Under 'Available Roles', there is a search filter: 'Name contains Customer'. A table lists the available roles:

| Add                                 | Name ↑↓                                              | Tenant ↑↓        | Capability | Users      |   |
|-------------------------------------|------------------------------------------------------|------------------|------------|------------|---|
| <input checked="" type="checkbox"/> | Customer Support - Technical Support or Advanced Ser | Service Provider | OWNER      | All Scopes | 8 |

Buttons for '< Previous' and 'Next >' are located at the bottom right of the table.

**Step 7** Click **Next**.

**Step 8** Under **User Review**, review the user details and the assigned roles. Click **Create**.

If external authentication is enabled, the authentication details are displayed.

**Note** After the user is added in Secure Workload, an activation email is sent to the registered email ID to set up the password.

## Edit User Details or Roles

### Before you begin

You must be a **Site Admin** or **Root Scope Owner** user to edit users in Secure Workload.



**Note** This page is filtered by the scope preference that is selected on the page header.



## Procedure

- Step 1** If applicable, select the appropriate root scope from the page header.
- Step 2** From the navigation pane, choose **Manage > User Access > Users**.
- Step 3** For the required user account, under **Actions**, click **Edit**.  
The **User Details** page is displayed.
- Step 4** Edit the following details.
- a) Update the following fields under **User Details**.

*Table 46: User Details Field Descriptions*

| Field                          | Description                                                                                                                                                          |
|--------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Email</b>                   | Update the email ID of the user.                                                                                                                                     |
| <b>First Name</b>              | Update the user's first name.                                                                                                                                        |
| <b>Last Name</b>               | Update the user's last name.                                                                                                                                         |
| <b>Scope</b>                   | Root scope that is assigned to the user for multitenancy. (Available to site admins)                                                                                 |
| <b>Reset MFA</b>               | If the user has lost their multifactor authentication (MFA) device or is locked out of MFA, then click <b>Reset MFA</b> . MFA of the user is reset in a few minutes. |
| <b>Resend Activation Email</b> | If a user has not received an activation email or the activation email link has expired, then click <b>Resend Activation Email</b> .                                 |

- b) Click **Next**.
- c) Under **Assign Roles**, add or remove assigned roles to the user.
- Click **Add Roles** to assign new roles, and then click the **Add** check box.
  - Select the assigned roles, click **Edit Assigned Roles**, and then click the **Remove** icon.
- d) Click **Next**.
- e) Under **User Review**, review the user details and the assigned roles. Click **Update** to update the user account.

If external authentication is enabled, the authentication details are displayed.

## Deactivating a User Account



---

**Note** To maintain consistency of change log audits, users can only be deactivated, they are not deleted from database.

---

### Before you begin

You must be a **Site Admin** or **Root Scope Owner** user.



---

**Note** This page is filtered by the scope preference that is selected on the page header.

---

### Procedure

- 
- Step 1** In the navigation bar on the left, click **Manage > User Access > Users**.
  - Step 2** If applicable, select the appropriate root scope from the top right of the page.
  - Step 3** In the row of the account you want to deactivate, click **Deactivate** button in the right-hand column.  
To view deactivated users, toggle **Hide Deleted Users** button.
- 

## Reactivating a User Account

If a user has been deactivated, you can reactivate the user.

### Before you begin

You must be a **Site Admin** or **Root Scope Owner** user.



---

**Note** This page is filtered by the scope preference that is selected on the page header.

---

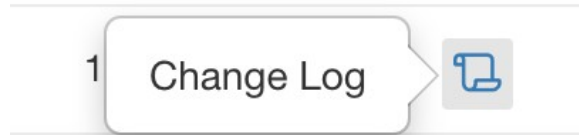
### Procedure

- 
- Step 1** In the navigation bar on the left, click **Manage > User Access > Users**.
  - Step 2** If applicable, select the appropriate root scope from the top right of the page.
  - Step 3** Toggle **Hide Deleted Users** to display all users, including deactivated users.
  - Step 4** For the required deactivated account, click **Restore** in the right-hand column to reactivate the account.
-

## Change Log – Users

**Site Admins** and users with the `SCOPE_OWNER` ability on the root scope can view the change logs for each user by clicking on the icon in the **Actions** column as shown in the following figure.

*Figure 445: Change Log*



For more information on the **Change Log**, see [Change Log](#). Root scope owners are restricted to viewing change log entries for entities belonging to their scope.





## CHAPTER 15

# Secure Workload OpenAPIs

---

OpenAPI provides a REST API for Secure Workload features.

- [OpenAPI Authentication](#), on page 740
- [Workspaces and Security Policies](#), on page 742
- [Scopes](#), on page 796
- [Configure Alerts](#), on page 801
- [Roles](#), on page 804
- [Users](#), on page 808
- [Inventory filters](#), on page 813
- [Flow Search](#), on page 816
- [Inventory](#), on page 823
- [Workload](#), on page 828
- [Default Policy Generation Config](#), on page 837
- [Forensics Intent](#), on page 839
- [Forensics Intent Orders](#), on page 841
- [Forensics Profiles](#), on page 842
- [Forensics Rules](#), on page 844
- [Enforcement](#), on page 847
- [Client Server configuration](#), on page 854
- [Software Agents](#), on page 858
- [Secure Workload software download](#), on page 865
- [Secure Workload Agents Upgrade](#), on page 867
- [User Uploaded Filehashes](#), on page 868
- [User-Defined Labels](#), on page 870
- [Virtual Routing and Forwarding](#), on page 882
- [Orchestrators](#), on page 885
- [Orchestrator Golden Rules](#), on page 891
- [FMC Orchestrator Domains](#), on page 892
- [RBAC \(Role-Based Access Control\) Considerations](#), on page 894
- [High Availability and Failover Considerations](#), on page 894
- [Kubernetes RBAC Resource Considerations](#), on page 894
- [Service Health](#), on page 896
- [Secure Connector](#), on page 896
- [Kubernetes Vulnerability Scanning](#), on page 897

- [Policy Enforcement Status for External Orchestrators, on page 902](#)
- [Download Certificates for Managed Data Taps and Datasinks, on page 903](#)
- [Change Logs, on page 904](#)
- [Non-Routable Endpoints, on page 906](#)
- [Config and Command Schemas for External Appliances and Connectors, on page 909](#)

## OpenAPI Authentication

OpenAPI uses a digest-based authentication scheme. The workflow is as follows:

1. Log in to the Secure Workload UI Dashboard.
2. Generate an API key and an API secret with the desired capabilities.
3. Use Secure Workload API SDK to send REST requests in JSON format.
4. To use the Python SDK, you install the SDK using `pip install tetpyclient`.
5. After the Python SDK is installed, here is some boilerplate code for instantiating the RestClient:

```
from tetpyclient import RestClient

API_ENDPOINT="https://<UI_VIP_OR_DNS_FOR_TETRATION_DASHBOARD>"

``verify`` is an optional param to disable SSL server authentication.
By default, cluster dashboard IP uses self signed cert after
deployment. Hence, ``verify=False`` might be used to disable server
authentication in SSL for API clients. If users upload their own
certificate to cluster (from ``Platform > SSL Certificate``)
which is signed by their enterprise CA, then server side authentication
should be enabled; in such scenarios, in the code below, verify=False
should be replaced with verify="path-to-CA-file"
credentials.json looks like:
{
"api_key": "<hex string>",
"api_secret": "<hex string>"
}

restclient = RestClient(API_ENDPOINT,
 credentials_file='<path_to_credentials_file>/credentials.json',
 verify=False)

followed by API calls, for example API to retrieve list of agents.
API can be passed /openapi/v1/sensors or just /sensors.
resp = restclient.get('/sensors')
```

## Generate API Key and Secret

### Procedure

- Step 1** In the upper right corner of Secure Workload UI, click the logged in account and choose **API Keys**.
- Step 2** Click **Create API Key**.
- Step 3** (Optional) Enter a description for the API key.

**Step 4** Select the required capabilities for the key and secret.

Select the limited set of capabilities that are intended for using the API Key+Secret pair.

**Note** The availability of the API capabilities varies based on the user role.

**Table 47: API Capabilities**

| Capability                 | Description                                                                                              |
|----------------------------|----------------------------------------------------------------------------------------------------------|
| sensor_management          | To configure and monitor status of software agents                                                       |
| software_download          | To download software packages for agents or virtual appliances                                           |
| flow_inventory_query       | To query flows and inventory items in Secure Workload cluster                                            |
| user_role_scope_management | To read, add, modify, or remove users, roles, and scopes                                                 |
| user_data_upload           | To allow users to upload data for annotating flows and inventory items or upload good or bad file hashes |
| app_policy_management      | To manage workspaces (applications) and enforce policies                                                 |
| external_integration       | To allow integration with external systems such as vCenter and kubernetes                                |

**Table 48: API Capabilities for Site Administrators**

| Capability           | Description                                                                  |
|----------------------|------------------------------------------------------------------------------|
| appliance_management | To manage Secure Workload appliance                                          |
| appliance_monitoring | To monitor Secure Workload appliance settings and configurations (read-only) |

**Step 5** Click **Create**.

API key and secret are generated and must be copied to a file, and saved in a safe location. Alternatively, you can download the JSON file with the key and secret.



**Note** If External Auth with LDAP and LDAP Authorization are enabled, access to OpenAPI using API Keys stop because Secure Workload roles that are derived from LDAP MemberOf groups are reassessed after the user session terminates. Therefore, to ensure uninterrupted OpenAPI access, it is recommended that any user with API keys have the **Use Local Authentication** option that is enabled in the Edit User Details flow for the user.

# Workspaces and Security Policies

The following pages describe the OpenAPI endpoints to manage [Manage Policies Lifecycle in Secure Workload](#).

## Workspaces

Workspaces (formerly known as “application workspaces” or “Applications”) are the containers for defining, analyzing and enforcing policies for the workloads in a particular scope. For more information about how they work see the [Use Workspaces to Manage Policies](#) documentation. This set of APIs requires the `app_policy_management` capability associated with the API key.

## Workspace Object

The workspace (“application”) JSON object is returned as a single object or an array of objects depending on the API endpoint. The object’s attributes are described below:

| Attribute            | Type    | Description                                                                                                                                                                                  |
|----------------------|---------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| id                   | string  | A unique identifier for the workspace.                                                                                                                                                       |
| name                 | string  | User specified name of the workspace.                                                                                                                                                        |
| description          | string  | User specified description of the workspace.                                                                                                                                                 |
| app_scope_id         | string  | ID of the scope with which the workspace is associated.                                                                                                                                      |
| author               | string  | First and last name of the user who created the workspace.                                                                                                                                   |
| primary              | boolean | Indicates if the workspace is primary for its scope.                                                                                                                                         |
| alternate_query_mode | boolean | Indicates if ‘dynamic mode’ is used for the workspace. In the dynamic mode, an automatic policy discovery run creates one or more candidate queries for each cluster. Default value is true. |
| created_at           | integer | Unix timestamp of when the workspace was created.                                                                                                                                            |
| latest_adm_version   | integer | The latest adm (v*) version of the workspace.                                                                                                                                                |
| analysis_enabled     | boolean | Indicates if analysis is enabled on the workspace.                                                                                                                                           |
| analyzed_version     | integer | The analyzed p* version of the workspace.                                                                                                                                                    |
| enforcement_enabled  | boolean | Indicates if enforcement is enabled for the workspace.                                                                                                                                       |
| enforced_version     | integer | The enforced p* version of the workspace.                                                                                                                                                    |



## List Applications

This endpoint will return an array of workspaces (“applications”).

```
GET /openapi/v1/applications
```

**Table 49: Parameters**

| Name         | Type   | Description                                            |
|--------------|--------|--------------------------------------------------------|
| app_scope_id | string | Match workspaces associated with a specific app scope. |
| exact_name   | string | Match workspaces with exactly the value provided.      |

Response object: Returns an array of workspace objects.

### Sample python code

```
restclient.get('/applications')
```

## Retrieve a Single Workspace

This endpoint will return the requested workspace (“application”) as a single JSON object.

```
GET /openapi/v1/applications/{application_id}
```

Parameters: The request URL contains the following parameters

| Name           | Type   | Description                              |
|----------------|--------|------------------------------------------|
| application_id | string | The unique identifier for the workspace. |

Response object: Returns the workspace object for the specified ID.

### Sample python code

```
application_id = '5d02b493755f0237a3d6e078'
restclient.get('/applications/%s' % application_id)
```

## Create a Workspace

This endpoint creates a workspace (“application”). It is possible to define policies by posting a JSON body containing the cluster and policy definitions.



**Note** If a primary workspace exists for the same scope and new policies are provided, the policies will be added as a new version to the existing workspace.

```
POST /openapi/v1/applications
```

Parameters: The JSON query body contains the following keys

| Name         | Type   | Description                              |
|--------------|--------|------------------------------------------|
| app_scope_id | string | The scope ID to assign to the workspace. |

| Name                 | Type    | Description                                                                                                                                                                                             |
|----------------------|---------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| name                 | string  | (optional) A name for the workspace.                                                                                                                                                                    |
| description          | string  | (optional) A description for the workspace.                                                                                                                                                             |
| alternate_query_mode | boolean | (optional) Indicates if 'dynamic mode' is used for the workspace. In the dynamic mode, an automatic policy discovery run creates one or more candidate queries for each cluster. Default value is true. |
| strict_validation    | boolean | (optional) Will return an error if there are unknown keys or attributes in the uploaded data. Useful for catching misspelled keys. Default value is false.                                              |
| primary              | string  | (optional) Set to 'true' to if this workspace should be primary for the associated scope. <b>Default is true</b>                                                                                        |

Extra optional parameters may be included describing policies to be created within the workspace.



**Note** The scheme corresponds to that returned during export from the UI and the **Details** endpoint.

| Name              | Type                       | Description                                            |
|-------------------|----------------------------|--------------------------------------------------------|
| clusters          | array of clusters          | Groups of nodes to be used to define policies.         |
| inventory_filters | array of inventory filters | Filters on datacenter assets.                          |
| absolute_policies | array of policies          | Ordered policies to be created with the absolute rank. |
| default_policies  | array of policies          | Ordered policies to be created with the default rank.  |
| catch_all_action  | string                     | "ALLOW" or "DENY"                                      |

Cluster object attributes:

| Name | Type   | Description                                 |
|------|--------|---------------------------------------------|
| id   | string | Unique identifier to be used with policies. |

| Name            | Type           | Description                                                                                                                                                    |
|-----------------|----------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------|
| name            | string         | Displayed name of the cluster.                                                                                                                                 |
| description     | string         | Description of the cluster.                                                                                                                                    |
| nodes           | array of nodes | Nodes or endpoints that are part of the cluster.                                                                                                               |
| consistent_uuid | string         | Must be unique to a given workspace. After an automatic policy discovery run, the similar/same clusters in the next version will maintain the consistent_uuid. |

Node object attributes:

| Name | Type   | Description                                                 |
|------|--------|-------------------------------------------------------------|
| ip   | string | IP or subnet of the node. For example 10.0.0.0/8 or 1.2.3.4 |
| name | string | Displayed name of the node.                                 |

Inventory Filter object attributes:

| Name  | Type   | Description                                              |
|-------|--------|----------------------------------------------------------|
| id    | string | Unique identifier to be used with policies.              |
| name  | string | Displayed name of the cluster.                           |
| query | object | JSON object representation of an inventory filter query. |

Policy object attributes:

| Name               | Type              | Description                                          |
|--------------------|-------------------|------------------------------------------------------|
| consumer_filter_id | string            | ID of a cluster, user inventory filter or app scope. |
| provider_filter_id | string            | ID of a cluster, user inventory filter or app scope. |
| action             | string            | “ALLOW” or “DENY”                                    |
| l4_params          | array of l4params | List of allowed ports and protocols.                 |

L4Params object attributes:

| Name  | Type    | Description                                        |
|-------|---------|----------------------------------------------------|
| proto | integer | Protocol Integer value (NULL means all protocols). |

| Name     | Type    | Description                                                       |
|----------|---------|-------------------------------------------------------------------|
| port     | array   | Inclusive range of ports. For example, [80, 80] or [5000, 6000].  |
| approved | boolean | (optional) Indicates if the policy is approved. Default is False. |

Response object: Returns the newly created workspace object.

### Sample python code

```

name = 'test'
scope_id = '5ce480cc497d4f1b4b9a9e8d'
filter_id = '5ce480cd497d4f1b4b9a9ea4'
application = {
 'app_scope_id': scope_id,
 'name': name,
 'absolute_policies': [
 {
 # consumer/provider filter IDs can be ID of a cluster identified during automatic
 # user inventory filter or app scope.
 'provider_filter_id': filter_id,
 'consumer_filter_id': filter_id,
 'action': 'ALLOW',
 # ALLOW policy for TCP on port 80.
 'l4_params': [
 {
 'proto': 6, # TCP
 'port': [80, 80], # port range
 }
],
 }
],
 'catch_all_action': 'ALLOW'
}
restclient.post('/applications', json_body=json.dumps(application))

```

## Import a New Version

Imports policies and creates a new v\* version for the workspace (“application”).

POST /openapi/v1/applications/{application\_id}/import

The parameters are the same as the create workspace endpoint.

Response object: Returns the workspace object.

## Validate a Set of Policies

Validates a set of policies without creating a new version.

POST /openapi/v1/applications/validate\_policies

An *app\_scope\_id* is required. The rest of the parameters are the same as the create workspace endpoint.

Response object:

| Attribute | Type    | Description                          |
|-----------|---------|--------------------------------------|
| valid     | boolean | Indicates if the policies are valid  |
| errors    | array   | If invalid, details about the errors |

## Delete a Workspace

Removes a workspace (“application”).

```
DELETE /openapi/v1/applications/{application_id}
```

Enforcement must be disabled on the workspace before it can be deleted.

If the workspace, or its clusters, are used on by other Applications (via a Provided Service relationship) this endpoint will return 422 `Unprocessable Entity`. The returned Error object will contain a `details` attribute with the count of dependent objects along with the ids of the first 10 of each type. This information can be used to locate and remove the blocking dependencies.

Parameters: The request URL contains the following parameters

| Name           | Type   | Description                              |
|----------------|--------|------------------------------------------|
| application_id | string | The unique identifier for the workspace. |

Response object: None

### Sample python code

```
application_id = '5d02b493755f0237a3d6e078'
restclient.delete('/applications/%s' % application_id)
```

## Update a Workspace

This end point updates an existing workspace (“application”).

```
PUT /openapi/v1/applications/{application_id}
```

Parameters: The request URL contains the following parameters

| Name           | Type   | Description                              |
|----------------|--------|------------------------------------------|
| application_id | string | The unique identifier for the workspace. |

The JSON query body contains the following keys

| Name    | Type   | Description                                           |
|---------|--------|-------------------------------------------------------|
| name    | string | (optional) The updated name for the workspace.        |
| descrip | string | (optional) The updated description for the workspace. |

| Name    | Type   | Description                                                                                             |
|---------|--------|---------------------------------------------------------------------------------------------------------|
| primary | string | (optional) Set to 'true' to make the workspace primary. Set to 'false' to make the workspace secondary. |

Response object: The updated workspace object for the specified ID.

#### Sample python code

```
application_id = '5d02b493755f0237a3d6e078'
req_payload = {
 'name': 'Updated Name',
 'description': 'Updated Description',
 'primary': 'true'
}
resp = restclient.put('/applications/%s' % application_id,
 json_body=json.dumps(req_payload))
```

## Retrieve Workspace Details

This endpoint returns a full export JSON file for the workspace. This will include policy and cluster definitions.

GET /openapi/v1/applications/{application\_id}/details

Parameters: The request URL contains the following parameters

| Name           | Type   | Description                                                               |
|----------------|--------|---------------------------------------------------------------------------|
| application_id | string | The unique identifier for the workspace.                                  |
| version        | string | (optional) A version in the form of 'v10' or 'p10', defaults to 'latest'. |

Response object: Returns the clusters and policies for the given workspace version.

#### Sample python code

```
application_id = '5d02b493755f0237a3d6e078'
For v* version v10 and for p* version p10
version = 'v10'
resp = restclient.get('/applications/%s/details?version=%s' % (application_id, version))
```

## List Workspace Versions

This endpoint will return a list of all the versions for a given workspace.

GET /openapi/v1/applications/{application\_id}/versions

Parameters: The request URL contains the following parameters

| Name           | Type   | Description                              |
|----------------|--------|------------------------------------------|
| application_id | string | The unique identifier for the workspace. |

| Name           | Type    | Description                                                                                |
|----------------|---------|--------------------------------------------------------------------------------------------|
| created_before | integer | (optional) For pagination, set to 'created_at' of the last version from previous response. |
| limit          | integer | (optional) Max results to return, default is 50.                                           |

Response object: An array of objects with the following attributes:

| Attribute   | Type    | Description                                       |
|-------------|---------|---------------------------------------------------|
| version     | string  | A version in the form of 'v10' or 'p10'.          |
| created_at  | integer | Unix timestamp of when the workspace was created. |
| description | string  | User provided description.                        |
| name        | string  | Displayed name.                                   |

#### Sample python code

```
application_id = '5d02b493755f0237a3d6e078'
created_before = 1612325705
limit = 10
resp = restclient.get('/applications/%s/versions?created_before=%s&limit=%s' %
 (application_id, created_before, limit))
```

## Delete Workspace Version

This endpoint will remove the given version including clusters and policies. Enforced or Analyzed versions can not be deleted. If members are referenced by another workspace, through an external policy, the response will return error with a list of the references.

```
DELETE /openapi/v1/applications/{application_id}/versions/{version}
```

Parameters: The request URL contains the following parameters

| Name           | Type   | Description                              |
|----------------|--------|------------------------------------------|
| application_id | string | The unique identifier for the workspace. |
| version        | string | A version in the form of 'v10' or 'p10'. |

Response object: None

#### Sample python code

```
application_id = '5d02b493755f0237a3d6e078'
version = 'v10'
resp = restclient.delete('/applications/%s/versions/%s' %
 (application_id, version))
```

## Compare Workspace versions

This endpoint computes the difference between the provided workspace versions. It will return the added, removed and optionally the unchanged policies. Cluster changes are included if the cluster is present in both versions, defined by a matching consistent\_uuid, and the query has changed.

GET /openapi/v1/applications/{application\_id}/version\_diff

Parameters: The request URL contains the following parameters

| Name              | Type    | Description                                                   |
|-------------------|---------|---------------------------------------------------------------|
| application_id    | string  | The unique identifier for the workspace.                      |
| base_version      | string  | Full version, for example 'v10' or 'p10'.                     |
| draft_version     | string  | Full version, for example 'v10' or 'p10'.                     |
| include_unchanged | boolean | Default is false. Returns unchanged policies in the response. |

Response object: Returns an object with the following attributes:

| Attribute | Type  | Description                                          |
|-----------|-------|------------------------------------------------------|
| clusters  | array | The clusters that have changed between the versions. |
| policies  | array | The policies that have changed between the versions. |

## Analyze latest policies

Enable analysis on the latest set of policies in the workspace.

POST /openapi/v1/applications/{application\_id}/enable\_analysis

Parameters: The request URL contains the following parameters

| Name           | Type   | Description                              |
|----------------|--------|------------------------------------------|
| application_id | string | The unique identifier for the workspace. |

Parameters: The optional JSON query body contains the following keys

| Name        | Type   | Description                                        |
|-------------|--------|----------------------------------------------------|
| action_note | string | (optional) Reason for the publish policies action. |
| name        | string | (optional) Name for the published policy version.  |



| Name        | Type   | Description                                              |
|-------------|--------|----------------------------------------------------------|
| description | string | (optional) description for the published policy version. |

Response object: Returns an object with the following attributes:

| Attribute               | Type    | Description                                 |
|-------------------------|---------|---------------------------------------------|
| data_set                | object  | JSON object representation of the data set. |
| analyzed_policy_version | integer | The analyzed p* version of the workspace.   |

### Sample python code

```
application_id = '5d02b493755f0237a3d6e078'
req_payload = {
 'action_note': 'Policy analysis',
 'name': 'Test run 1',
 'description': 'New workloads added.'
}
resp = restclient.post('/applications/%s/enable_analysis' % application_id,
 json_body=json.dumps(req_payload))
```

## Disable policy analysis on a single workspace

Disable policy analysis on the workspace.

POST /openapi/v1/applications/{application\_id}/disable\_analysis

Parameters: The request URL contains the following parameters

| Name           | Type   | Description                              |
|----------------|--------|------------------------------------------|
| application_id | string | The unique identifier for the workspace. |

Response object: Returns an object with the following attributes:

| Attribute               | Type    | Description                                 |
|-------------------------|---------|---------------------------------------------|
| data_set                | object  | JSON object representation of the data set. |
| analyzed_policy_version | integer | Last analyzed p* version of the workspace.  |

### Sample python code

```
application_id = '5d02b493755f0237a3d6e078'
resp = restclient.post('/applications/%s/disable_analysis' % application_id)
```

## Enforce a single workspace

Enable enforcement on the latest set of policies in the workspace.

```
POST /openapi/v1/applications/{application_id}/enable_enforce
```



**Warning** New host firewall rules will be inserted and any existing rules will be deleted on the relevant hosts.

Parameters: The request URL contains the following parameters

| Name           | Type   | Description                               |
|----------------|--------|-------------------------------------------|
| application_id | string | The unique identifier for the workspace.  |
| version        | string | (optional) The policy version to enforce. |

If a version is not provided the latest policies of the workspace will be enforced. versions is preferred to be of the form 'p\*', if just an integer is provided the corresponding 'p\*' version will be enforced.

Response object: Returns an object with the following attributes:

| Name  | Type   | Description                                           |
|-------|--------|-------------------------------------------------------|
| epoch | string | Unique identifier for the latest enforcement profile. |

### Sample python code

```
application_id = '5d02b493755f0237a3d6e078'
req_payload = {
 'version': 'p10'
}
resp = restclient.post('/applications/%s/enable_enforce' % application_id,
 json_body=json.dumps(req_payload))
```

## Disable enforcement for a single workspace

Disable enforcement on the workspace.

```
POST /openapi/v1/applications/{application_id}/disable_enforce
```



**Warning** New host firewall rules will be inserted and any existing rules will be deleted on the relevant hosts.

Parameters: The request URL contains the following parameters

| Name           | Type   | Description                              |
|----------------|--------|------------------------------------------|
| application_id | string | The unique identifier for the workspace. |

Response object: Returns an object with the following attributes:

| Name  | Type   | Description                                           |
|-------|--------|-------------------------------------------------------|
| epoch | string | Unique identifier for the latest enforcement profile. |

#### Sample python code

```
application_id = '5d02b493755f0237a3d6e078'
resp = restclient.post('/applications/%s/disable_enforce' %
 application_id)
```

## Initiate Automatic Policy Discovery

Automatically discover policies for the workspace. (Formerly known as “submitting an ADM run”).

POST /openapi/v1/applications/{application\_id}/submit\_run

Parameters: The request URL contains the following parameters

| Name           | Type   | Description                              |
|----------------|--------|------------------------------------------|
| application_id | string | The unique identifier for the workspace. |

Parameters: The JSON query body contains the following keys

| Name                   | Type   | Description                                                                                                                                                                                                                   |
|------------------------|--------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| start_time             | string | Start time of the input time interval for an automatic policy discovery run.                                                                                                                                                  |
| end_time               | string | End time of the input time interval for an automatic policy discovery run.                                                                                                                                                    |
| clustering_granularity | string | (optional) <a href="#">Cluster Granularity</a> allows the user to have a control on the size of the clusters generated by automatic policy discovery. Expected values: VERY_FINE, FINE, MEDIUM, COARSE, or VERY_COARSE        |
| port_generalization    | string | (optional) <a href="#">Port Generalization</a> controls the level of statistical significance required when performing port generalization. Expected values: DISABLED, CONSERVATIVE, MODERATE, AGGRESSIVE, or VERY_AGGRESSIVE |

| Name                            | Type    | Description                                                                                                                                                                                                                                                                                                                                                                                          |
|---------------------------------|---------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| policy_compression              | string  | (optional) <a href="#">Policy Compression</a> when enabled, policies that are sufficiently frequent, i.e. they use the same provider port, among the generated clusters inside a workspace may be ‘factored out’ to the parent, that is, replaced with one or more policies applicable to the entire parent scope. Expected values: DISABLED, CONSERVATIVE, MODERATE, AGGRESSIVE, or VERY_AGGRESSIVE |
| auto_accept_policy_connectors   | boolean | (optional) <a href="#">Auto Accept Policy Connectors</a> any outgoing policy requests created during the automatic policy discovery will be auto accepted.                                                                                                                                                                                                                                           |
| enable_exclusion_filter         | boolean | (optional) Enable exclusion filter option provides the flexibility to ignore all conversations matching any of the user-defined exclusion filters (if any). For more information, see <a href="#">Exclusion Filters</a> .                                                                                                                                                                            |
| enable_default_exclusion_filter | boolean | (optional) Enable default exclusion filter option provides the flexibility to ignore all conversations matching any of the default exclusion filters (if any). For more information, see <a href="#">Default Exclusion Filters</a> for more info.                                                                                                                                                    |
| enable_service_discovery        | boolean | (optional) When <a href="#">Enable service discovery on agent</a> is set, ephemeral port-range information about services present on the agent node are reported. Policies are then generated based on the reported port-range information.                                                                                                                                                          |
| carry_over_policies             | boolean | (optional) When <a href="#">Carry over Approved Policies</a> is set, all the policies that are marked as approved by the user via UI or OpenAPI will be preserved.                                                                                                                                                                                                                                   |

| Name                   | Type    | Description                                                                                                                                                                                                                                             |
|------------------------|---------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| skip_clustering        | boolean | (optional) When <a href="#">Skip clustering and only generate policies</a> is set, no new clusters are generated, and policies are generated from any existing approved clusters or inventory filters and otherwise involve all workloads in the scope. |
| deep_policy_generation | boolean | (Optional) You can generate policies for a branch of the scope tree rather than for a single scope. For more information, see <a href="#">Discover Policies for One Scope or for a Branch of the Scope Tree, on page 439</a> and subtopic.              |
| use_default_config     | boolean | (optional) When this option is set, automatic policy discovery will use the Default Policy Discovery Config instead of the previous run config. For more information, see <a href="#">Default Policy Discovery Config</a> .                             |



**Note** Unspecified optional parameter default values will be taken from the previous automatic policy discovery run config if automatic policy discovery was performed earlier in the workspace or else the default values will be taken from the Default Policy Discovery Config.

Response object: Returns an object with the following attributes:

| Name    | Type   | Description                                                         |
|---------|--------|---------------------------------------------------------------------|
| message | string | Message about success or failure of automatic policy discovery run. |

#### Sample python code

```
application_id = '5d02b493755f0237a3d6e078'
req_payload = {
 'start_time': '2020-09-17T10:00:00-0700',
 'end_time': '2020-09-17T11:00:00-0700',
 # Optional Parameters.
 'clustering_granularity': 'FINE',
 'port_generalization': 'AGGRESSIVE',
 'policy_compression': 'AGGRESSIVE',
 'auto_accept_policy_connectors': False,
 'enable_exclusion_filter': True,
 'enable_default_exclusion_filter': True,
 'enable_service_discovery': True,
 'carry_over_policies': True,
 'skip_clustering': False,
```

```

 'deep_policy_generation': True,
 'use_default_config': False
 }
 resp = restclient.post('/applications/%s/submit_run' % application_id,
 json_body=json.dumps(req_payload))

```

## Get Status of a Policy Discovery Run

Query the status of an automatic policy discovery run in the workspace.

GET /openapi/v1/applications/{application\_id}/adm\_run\_status

Parameters: The request URL contains the following parameters

| Name           | Type   | Description                              |
|----------------|--------|------------------------------------------|
| application_id | string | The unique identifier for the workspace. |

Response object: Returns an object with the following attributes:

| Name   | Type   | Description                                                                        |
|--------|--------|------------------------------------------------------------------------------------|
| status | string | Status of the automatic policy discovery run. Values: PENDING, COMPLETE, or FAILED |

### Sample python code

```

application_id = '5d02b493755f0237a3d6e078'
resp = restclient.get('/applications/%s/adm_run_status' % application_id)

```

## Policies

This set of APIs can be used to manage add, edit or delete Policies. `version` parameter is required for create and update catch all actions. They require the `user_role_scope_management` capability associated with the API key.

### Policy object

The policy object attributes are described below:

| Attribute          | Type   | Description                                                                                                           |
|--------------------|--------|-----------------------------------------------------------------------------------------------------------------------|
| id                 | string | Unique identifier for the policy.                                                                                     |
| application_id     | string | The id for the workspace to which the policy belongs.                                                                 |
| consumer_filter_id | string | ID of a defined filter. Currently, any cluster, user defined filter or scope can be used as the consumer of a policy. |

| Attribute          | Type              | Description                                                                                                                                                         |
|--------------------|-------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| provider_filter_id | string            | ID of a defined filter. Currently, any cluster, user defined filter or scope can be used as the provider of a policy.                                               |
| version            | string            | Indicates the version of the workspace to which the policy belongs.                                                                                                 |
| rank               | string            | Policy rank, possible values: DEFAULT, ABSOLUTE or CATCHALL.                                                                                                        |
| policy_action      | string            | Possible values can be ALLOW or DENY. Indicates whether traffic should be allowed or dropped for the given service port/protocol between the consumer and provider. |
| priority           | integer           | Used to sort policy.                                                                                                                                                |
| l4_params          | array of l4params | List of allowed ports and protocols.                                                                                                                                |

L4Params object attributes:

| Name        | Type    | Description                                            |
|-------------|---------|--------------------------------------------------------|
| proto       | integer | Protocol Integer value (NULL means all protocols).     |
| port        | array   | Inclusive range of ports. eg [80, 80] or [5000, 6000]. |
| description | string  | Short string about this proto and port.                |
| approved    | boolean | If the policy has been approved by the user.           |

## Get Policies

This endpoint returns a list of policies in a particular workspace. This API is available to API keys with `app_policy_management` capability.

GET /openapi/v1/applications/{application\_id}/policies

Parameters: The request URL contains the following parameters

| Name               | Type   | Description                                                            |
|--------------------|--------|------------------------------------------------------------------------|
| version            | string | Indicates the version of the workspace from which to get the policies. |
| consumer_filter_id | string | (optional) Filters the output by the consumer filter id.               |
| provider_filter_id | string | (optional) Filters the output by the consumer filter id.               |

Policy IDs can change from version to version. To obtain the list of policies from a published version, the version number should be prefixed with a 'p'. For example, to fetch all the policies in published version 3 we can perform a request such as:

```
GET /openapi/v1/applications/{application_id}/policies?version=p3
```

Returns an object of all policies in this particular workspace as shown below

```
{
 absolute_policies: [...],
 default_policies: [...],
 catch_all_action:
}
```

### Sample Python code

```
application_id = '5f88c996755f023f3bafef163'
restclient.get('/applications/%s/policies' % application_id, params={'version': '1'})
```

### Get Default Policies

This endpoint returns a list of Default policies for a given workspace. This API is available to API keys with `app_policy_management` capability.

```
GET /openapi/v1/applications/{application_id}/default_policies
```

Parameters:

| Name    | Type    | Description                                                                                                     |
|---------|---------|-----------------------------------------------------------------------------------------------------------------|
| id      | string  | Unique identifier for the policy.                                                                               |
| version | string  | Indicates the version of the workspace for which to get the policies.                                           |
| limit   | integer | Limits the number of policies per request.                                                                      |
| offset  | integer | (optional) Offset number received from previous response, should always be used along with <code>limit</code> . |



| Name               | Type   | Description                                              |
|--------------------|--------|----------------------------------------------------------|
| consumer_filter_id | string | (optional) Filters the output by the consumer filter id. |
| provider_filter_id | string | (optional) Filters the output by the provider filter id. |

Returns a list of default policies for the provided version of this workspace. The response contains the requested number of policies and an `offset`, to get the next set policies use this `offset` in the subsequent requests. Absence of an `offset` in the response indicates that all the policies are already retrieved.

### Sample Python code

```
application_id = '5f88c996755f023f3bafef163'
restclient.get('/applications/%s/default_policies' % application_id, params={'version':
'1', 'limit': 3, 'offset': 3})
```

### Sample response

```
{
 "results": [
 PolicyObject4,
 PolicyObject5,
 PolicyObject6
],
 "offset": 6
}
```

### Get Absolute Policies

This endpoint returns a list of Absolute policies in a given workspace. This API is available to API keys with `app_policy_management` capability.

```
GET /openapi/v1/applications/{application_id}/absolute_policies
```

Parameters:

| Name               | Type    | Description                                                                                                     |
|--------------------|---------|-----------------------------------------------------------------------------------------------------------------|
| version            | string  | Indicates the version of the workspace from which to get the policies.                                          |
| limit              | integer | Limits the number of policies per request.                                                                      |
| offset             | integer | (optional) Offset number received from previous response, should always be used along with <code>limit</code> . |
| consumer_filter_id | string  | (optional) Filters the output by the consumer filter id.                                                        |
| provider_filter_id | string  | (optional) Filters the output by the provider filter id.                                                        |

Returns a list of absolute policies in the provided version of this workspace. The response contains the requested number of polices and an `offset`, to get the next set policies use this `offset` in the subsequent requests. Absence of an `offset` in the response indicates that all the policies are already retrieved.

### Sample Python code

```
application_id = '5f88c996755f023f3bafe163'
restclient.get('/applications/%s/absolute_policies' % application_id, params={'version':
'1', 'limit': 3})
```

### Sample response

```
{
 "results": [
 PolicyObject1,
 PolicyObject2,
 PolicyObject3
],
 "offset": 3
}
```

### Get Catch All Policies

This endpoint returns a Catch All policy for a given workspace. This API is available to API keys with `app_policy_management` capability.

```
GET /openapi/v1/applications/{application_id}/catch_all
```

Parameters:

| Name    | Type   | Description                                                            |
|---------|--------|------------------------------------------------------------------------|
| version | string | Indicates the version of the workspace from which to get the policies. |

Returns a single policy object representing the catch all policy of the given version of the workspace.

### Sample Python code

```
application_id = '5f88c996755f023f3bafe163'
restclient.get('/applications/%s/catch_all' % application_id, params={'version': '1'})
```

## Get Specific Policy

This endpoint returns an instance of a policy.

```
GET /openapi/v1/policies/{policy_id}
```

Returns the policy object associated with the specified ID.

### Sample Python code

```
policy_id = '5f88ca1e755f0222f85ce85c'
restclient.get('/policies/%s' % policy_id)
```

## Search for a Specific Policy With Policy Identifier

This endpoint searches for the specified policy using Policy Identifier Parameters as a composite key.

POST /openapi/v1/policies/search

The query body consists of a JSON body with the following schema:

| Name              | Type   | Description                                           |
|-------------------|--------|-------------------------------------------------------|
| application_id    | string | The ID of the application workspace.                  |
| policy_identifier | object | Fields that make up the consistent policy identifier. |

The policy identifier fields are made up using the following schema:

| Name                     | Type    | Description                                                                                                                                                     |
|--------------------------|---------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------|
| version                  | string  | (optional) Indicates the version of the application for which to get the policies; defaults to the latest 'v' version of the application when left unspecified. |
| consumer_consistent_uuid | string  | Consistent UUID of the consumer or source.                                                                                                                      |
| provider_consistent_uuid | string  | Consistent UUID of the provider or destination.                                                                                                                 |
| rank                     | string  | Policy rank has to be one of "DEFAULT" or "ABSOLUTE".                                                                                                           |
| action                   | string  | Policy action has to be one of "ALLOW" or "DENY".                                                                                                               |
| priority                 | integer | Priority value for the policy.                                                                                                                                  |
| protocol                 | integer | IP protocol number (0-255) for the .                                                                                                                            |
| start_port               | integer | (optional) Start of port range (0-65535); defaults to 0 when unspecified.                                                                                       |
| end_port                 | integer | (optional) End of port range (0-65535); defaults to 65535 if start_port is 0 or else to the start_prot.                                                         |

### Sample Python code

```
application_id = '5f88cale755f0222f85ce85c'
consumer_id = '5f88cale755f0222f85ce85d'
provider_id = '5f88cale755f0222f85ce85d'
rank = 'DEFAULT'
action = 'ALLOW'
priority = 100
protocol = 6
start_port = 80
version = 'p3'
```

```

req_body = f'''
{{
 "application_id": "{application_id}",
 "policy_identifier": {{
 "consumer_consistent_uuid": "{consumer_id}",
 "provider_consistent_uuid": "{provider_id}",
 "rank": "{rank}",
 "action": "{action}",
 "priority": {priority},
 "protocol": "{protocol}",
 "start_port": "{start_port}",
 "version": "{version}"
 }}
}}'''
restclient.post('/policies/search', json_body=req_body)

```

## Create a Policy

This endpoint is used to create new policies.

POST /openapi/v1/applications/{application\_id}/policies

Parameters:

| Attribute          | Type    | Description                                                                                                                             |
|--------------------|---------|-----------------------------------------------------------------------------------------------------------------------------------------|
| consumer_filter_id | string  | ID of a defined filter.                                                                                                                 |
| provider_filter_id | string  | ID of a defined filter.                                                                                                                 |
| version            | string  | Indicates the version of the workspace in which to update the policies.                                                                 |
| rank               | string  | values can be DEFAULT, ABSOLUTE or CATCHALL for ranking                                                                                 |
| policy_action      | string  | values can be ALLOW or DENY: means whether we should allow or drop traffic from consumer to provider on the given service port/protocol |
| priority           | integer | Used to sort policy.                                                                                                                    |

### Sample Python code

```

req_payload = {
 "version": "v1",
 "rank" : "DEFAULT",
 "policy_action" : "ALLOW",
 "priority" : 100,
 "consumer_filter_id" : "123456789",
 "provider_filter_id" : "987654321",
}
resp = restclient.post('/openapi/v1/applications/{application_id}/policies',
json_body=json.dumps(req_payload))

```

### Create a Default Policy

This endpoint is used to create new default policies. This endpoint creates a default policy similar to the create a policy endpoint.

```
POST /openapi/v1/applications/{application_id}/default_policies
```

### Create an Absolute Policy

This endpoint is used to create new absolute policies. This endpoint creates an absolute policy similar to the create a policy endpoint.

```
POST /openapi/v1/applications/{application_id}/absolute_policies
```

## Update a Policy

This endpoint updates a policy.

```
PUT /openapi/v1/policies/{policy_id}
```

Parameters:

| Attribute          | Type    | Description                                                                                                                                                         |
|--------------------|---------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| consumer_filter_id | string  | ID of a defined filter.                                                                                                                                             |
| provider_filter_id | string  | ID of a defined filter.                                                                                                                                             |
| policy_action      | string  | Possible values can be ALLOW or DENY. Indicates whether traffic should be allowed or dropped for the given service port/protocol between the consumer and provider. |
| priority           | integer | Used to sort policy priorities                                                                                                                                      |

Returns the modified policy object associated with specified ID.

### Update a Catch All

This endpoint updates Catch All for a particular workspace.

```
PUT /openapi/v1/applications/{application_id}/catch_all
```

Parameters:

| Attribute | Type   | Description                                                             |
|-----------|--------|-------------------------------------------------------------------------|
| version   | string | Indicates the version of the workspace in which to update the policies. |

| Attribute     | Type   | Description                                                                                                                                    |
|---------------|--------|------------------------------------------------------------------------------------------------------------------------------------------------|
| policy_action | string | Possible values can be ALLOW or DENY. Indicates whether traffic not matching any of the policies in this workspace will be allowed or dropped. |

## Adding Service Ports to a Policy

This endpoint is used to create service ports for a specific policy.

```
POST /openapi/v1/policies/{policy_id}/l4_params
```

Parameters:

| Attribute   | Type    | Description                                                            |
|-------------|---------|------------------------------------------------------------------------|
| version     | string  | Indicates the version of the workspace from which to get the policies. |
| start_port  | integer | Start port of the range.                                               |
| end_port    | integer | End port of the range.                                                 |
| proto       | integer | Protocol Integer value (NULL means all protocols).                     |
| description | string  | (optional) Short string about this proto and port.                     |

## Updating Service Ports of a Policy

This endpoint updates the specified service port of a Policy.

```
PUT /openapi/v1/policies/{policy_id}/l4_params/{l4_params_id}
```

Parameters:

| Attribute | Type | Description                   |
|-----------|------|-------------------------------|
| approved  | bool | Marks the policy as approved. |

## Deleting Service Ports of a Policy

This endpoint deletes the specified service port of a policy. (Optional) See [Exclusion Filters](#) for more details.

```
DELETE /openapi/v1/policies/{policy_id}/l4_params/{l4_params_id}
```

Parameters:

| Attribute               | Type | Description                                                                                                                                                                                                           |
|-------------------------|------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| create_exclusion_filter | bool | (Optional) If true, creates an exclusion filter matching the policy. Flows matching this filter will be excluded from future automatic policy discovery runs. See <a href="#">Exclusion Filters</a> for more details. |

## Deleting a Policy

This endpoint deletes the specified Policy. No exclusion filters are created.

```
DELETE /openapi/v1/policies/{policy_id}
```

## Deleting a Policy with Identifier

This endpoint deletes the specified policy using Policy Identifier Parameters. No exclusion filters are created.

```
DELETE /openapi/v1/policies/destroy_with_identifier
```

The query body consists of a JSON body with the following schema:

| Name              | Type   | Description                                           |
|-------------------|--------|-------------------------------------------------------|
| application_id    | string | The ID of the application workspace.                  |
| policy_identifier | object | Fields that make up the consistent policy identifier. |

The policy identifier fields are made up using the following schema:

| Name                     | Type    | Description                                                                                                                                                         |
|--------------------------|---------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| version                  | string  | (optional) 'v' version of the application workspace in which to perform the delete operation; defaults to the latest 'v' version of the workspace when unspecified. |
| consumer_consistent_uuid | string  | Consistent UUID of the consumer or source                                                                                                                           |
| provider_consistent_uuid | string  | Consistent UUID of the provider or destination                                                                                                                      |
| rank                     | string  | Policy rank has to be one of "DEFAULT" or "ABSOLUTE"                                                                                                                |
| action                   | string  | Policy action has to be one of "ALLOW" or "DENY"                                                                                                                    |
| priority                 | integer | Priority value for the policy                                                                                                                                       |

| Name       | Type    | Description                                                                                            |
|------------|---------|--------------------------------------------------------------------------------------------------------|
| protocol   | integer | IP protocol number (0-255) for the policy                                                              |
| start_port | integer | (optional) Start of port range (0-65535); defaults to 0 when unspecified                               |
| end_port   | integer | (optional) End of port range (0-65535); defaults to 65535 if start_port is 0 or else to the start_prot |

### Sample Python code

```

application_id = '5f88ca1e755f0222f85ce85c'
consumer_id = '5f88ca1e755f0222f85ce85d'
provider_id = '5f88ca1e755f0222f85ce85d'
action = 'ALLOW'
rank = 'DEFAULT'
protocol = 6
start_port = 80
priority = 100
version = '5'

req_body = f'''
{{
 "application_id": "{application_id}",
 "policy_identifier": {{
 "consumer_consistent_uuid": "{consumer_id}",
 "provider_consistent_uuid": "{provider_id}",
 "rank": "{rank}",
 "priority": {priority},
 "action": "{action}",
 "protocol": "{protocol}",
 "start_port": "{start_port}",
 "version": "{version}"
 }}
}}
'''
restclient.delete('/policies/destroy_with_identifier', json_body=req_body)

```

## Policy Quick Analysis

This endpoint can be used to find matching set of policies for any hypothetical flow against the analyzed/enforced policies in a root scope. For more details refer [Quick Analysis](#)

This API is only available to users with a minimum read access to root scope and requires `app_policy_management` capability associated with the API key.

POST /openapi/v1/policies/{rootScopeID}/quick\_analysis

The query body consists of a JSON body with the following schema:

| Name        | Type   | Description                          |
|-------------|--------|--------------------------------------|
| consumer_ip | string | IP Address of the client / consumer. |



| Name           | Type    | Description                                                                                                                                                                                                                                                                                                                                                                                |
|----------------|---------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| provider_ip    | string  | IP Address of the server / provider.                                                                                                                                                                                                                                                                                                                                                       |
| provider_port  | integer | (optional) Provider Port, only relevant for TCP or UDP flows.                                                                                                                                                                                                                                                                                                                              |
| protocol       | string  | Protocol of the flow, e.g. TCP.                                                                                                                                                                                                                                                                                                                                                            |
| analysis_type  | string  | Analysis type can be either <b>analyzed</b> or <b>enforced</b> . Analysis type “analyzed” makes the flow decision by matching the flow against all the analyzed polices in the root scope. Analysis type “enforced” makes the flow decision by matching the flow against all enforced policies in the root scope.                                                                          |
| application_id | string  | (optional) The ID of the primary workspace, always accompanied by the workspace ‘v’ version, if specified, makes the flow decision by using the policies from the specified version along with analyzed/enforced policies from other workspaces in the root scope. If this field is skipped, the flow decision is made by considering all the analyzed/enforced polices in the root scope. |
| version        | integer | (optional) The ‘v’ version of the workspace mentioned above. This must be specified if the application_id is specified and must be skipped otherwise.                                                                                                                                                                                                                                      |

### Sample request

The body of the request should be a JSON formatted query.

An example of a query body where the flow decision is based on all analyzed polices:

```
req_payload = {
 "consumer_ip": "4.4.1.1",
 "provider_ip": "4.4.2.1",
 "provider_port": 9081,
 "protocol": "TCP",
 "analysis_type": "analyzed"
}
resp = restclient.post('/openapi/v1/policies/{rootScopeID}/quick_analysis',
json_body=json.dumps(req_payload))
```

An example of a query body where the flow decision is based on the policies from the workspace's 'v' version along with the analyzed polices from all other workspaces in the root scope:

```
req_payload = {
 "consumer_ip": "4.4.1.1",
 "provider_ip": "4.4.2.1",
 "provider_port": 9081,
 "protocol": "TCP",
 "analysis_type": "analyzed",
 "application_id": "5e7e5f56497d4f0bc26c7bb3",
 "version": 1
}
resp = restclient.post('/openapi/v1/policies/{rootScopeID}/quick_analysis',
json_body=json.dumps(req_payload))
```

### Sample response

The response is a JSON object in the body with the following properties:

| Keys            | Values                                                                 |
|-----------------|------------------------------------------------------------------------|
| policy_decision | The decision of the hypothetical flow whether is allowed or denied.    |
| outbound_policy | The policy on the consumer thats allowing/denying the outgoing traffic |
| inbound_policy  | The policy on the provider thats allowing/denying the incoming traffic |

```
{
 "policy_decision": "ALLOW",
 "outbound_policy": {
 "policy_rank": "DEFAULT",
 "start_port": 9082,
 "l4_detail_id": "5e7e600f497d4f7341f4f6d0",
 "src_filter_id": "5e7e600e497d4f7341f4f459",
 "end_port": 9082,
 "cluster_edge_id": "5e7e600f497d4f7341f4f6d1",
 "dst_filter_id": "5e7d0efc497d4f44b6b09351",
 "action": "ALLOW",
 "protocol": "TCP",
 "app_scope_id": "5e7e5f3a497d4f0bc26c7bb0"
 },
 "inbound_policy": {
 "policy_rank": "DEFAULT",
 "start_port": 9082,
 "l4_detail_id": "5e7e600f497d4f7341f4f6d0",
 "src_filter_id": "5e7e600e497d4f7341f4f459",
 "end_port": 9082,
 "cluster_edge_id": "5e7e600f497d4f7341f4f6d1",
 "dst_filter_id": "5e7d0efc497d4f44b6b09351",
 "action": "ALLOW",
 "protocol": "TCP",
 "app_scope_id": "5e7e5f3a497d4f0bc26c7bb0"
 }
}
```

## Policy Statistics

This endpoint returns the number of packets, bytes, and conversations observed for a policy over a time interval. A conversation can be broadly described as a flow observation matching a policy that is aggregated with a granularity of one hour. The number of conversations that are measured for a given policy within one hour represents the number of distinct pairs of consumer and provider inventory items that have communicated over the network during that one hour.

Although this endpoint accepts Policy Identifier Parameters as input, we recommend you to use policy and L4 parameter IDs from a published version of the workspace.



**Note** After a new version of the application workspace is published, it can take up to 6 hours before results become available. All the timestamp resolutions will also have a minimum granularity of 6 hours.

To get the policy statistics for a policy across enforced versions of an application workspace the URL path is:

```
POST /openapi/v1/policies/stats/enforced
```

To get the policy statistics for a policy across analyzed versions of an application workspace the URL path is:

```
POST /openapi/v1/policies/stats/analyzed
```

The query body consists of a JSON body with the following schema:

**Table 50:**

| Name              | Type   | Description                                                                                                |
|-------------------|--------|------------------------------------------------------------------------------------------------------------|
| application_id    | string | The ID of the application workspace.                                                                       |
| t0                | string | The beginning of the time interval in RFC-3339 format.                                                     |
| t1                | string | (optional) The end of the time interval in RFC-3339 format; defaults to current time if left unspecified.  |
| policy_id         | string | The ID of the policy; not required if the policy identifier is present.                                    |
| l4_param_id       | string | The ID of the l4 parameter; not required if the policy identifier is present, or for "CATCH_ALL" policies. |
| policy_identifier | object | Fields that make up the consistent policy identifier.                                                      |

The policy identifier fields are made up using the following schema:

| Name                     | Type    | Description                                                                                             |
|--------------------------|---------|---------------------------------------------------------------------------------------------------------|
| consumer_consistent_uuid | string  | Consistent UUID of the consumer or source.                                                              |
| provider_consistent_uuid | string  | Consistent UUID of the provider or destination.                                                         |
| rank                     | string  | Policy rank has to be one of “DEFAULT” or “ABSOLUTE”.                                                   |
| action                   | string  | Policy action has to be one of “ALLOW” or “DENY”.                                                       |
| priority                 | integer | Priority value for the policy.                                                                          |
| protocol                 | integer | IP protocol number (0-255) for the policy.                                                              |
| start_port               | integer | (optional) Start of port range (0-65535); defaults to 0 when unspecified                                |
| end_port                 | integer | (optional) End of port range (0-65535); defaults to 65535 if start_port is 0 or else to the start_prot. |

### Sample Python code

```

application_id = '5f88ca1e755f0222f85ce85c'
consumer_id = '5f88ca1e755f0222f85ce85d'
provider_id = '5f88ca1e755f0222f85ce85d'
action = 'ALLOW'
rank = 'DEFAULT'
protocol = 6
start_port = 80
priority = 100

req_body = f'''
{{
 "application_id": "{application_id}",
 "t0": "2022-07-06T00:00:00Z",
 "t1": "2022-07-28T19:00:00Z",
 "policy_identifier": {{
 "consumer_consistent_uuid": "{consumer_id}",
 "provider_consistent_uuid": "{provider_id}",
 "rank": "{rank}",
 "priority": {priority},
 "action": "{action}",
 "protocol": "{protocol}",
 "start_port": "{start_port}"
 }}
}}'''
restclient.post('/policies/stats/analyzed', json_body=req_body)

For CATCH_ALL policies:
root_app_scope_id = '6f88ca1e755f0222f85ce85e'

```

```

rank = 'CATCH_ALL'
action = 'DENY'
req_body = f'''
{{
 "application_id": "{application_id}",
 "t0": "2022-07-06T00:00:00Z",
 "t1": "2022-07-28T19:00:00Z",
 "policy_identifier": {{
 "consumer_consistent_uuid": "{root_app_scope_id}",
 "provider_consistent_uuid": "{root_app_scope_id}",
 "rank": "{rank}",
 "action": "{action}"
 }}
}}'''

restclient.post('/policies/stats/analyzed', json_body=req_body)

```

### Sample response

The response is a JSON object in the body with the following properties.

**Table 51:**

| Keys               | Values                                                                               |
|--------------------|--------------------------------------------------------------------------------------|
| conversation_count | The number of conversations that are observed for the specified duration and policy. |
| packet_count       | The number of packets that are observed for the specified duration and policy.       |
| byte_count         | The number of bytes observed for the specified duration and policy.                  |
| first_seen_at      | The timestamp (in RFC-3339 format) when we first observed flows for this policy.     |
| last_seen_at       | The timestamp (in RFC-3339 format) when we last observed flows for this policy.      |
| agg_start_version  | The earliest published version of this policy on record from time t0 onwards.        |
| agg_start_time     | The timestamp the agg_start_version was published.                                   |

```

{
 "conversation_count": 72,
 "packet_count": 800,
 "byte_count": 1960,
 "first_seen_at": "2022-09-09T11:00:00.000Z",
 "last_seen_at": "2022-09-09T11:00:00.000Z",
 "agg_start_version": 4,
 "agg_start_time": "2022-08-10T23:00:00.000Z"
}

```

## Unused Policies

This endpoint returns the policy identifiers in a published workspace for which no conversations are observed over a specified time interval.

### Policy Identifier

All policies and ADM-generated clusters can change their IDs across application workspace versions even if the underlying filter queries or policy port and protocol do not change. In order to keep track of flow hit counts for a particular policy across workspace versions we use consistent UUIDs for the filters that do not change across versions and a composite key called the *policy identifier* comprising the provider and consumer consistent UUIDs along with rank, action, priority, port and protocol.

Thus, policy identifiers serve as a composite key that can both identify and describe the important aspects of a policy across application workspace versions, whereas policy IDs (such as those used in the regular CRUD endpoints) can change across versions of the workspace.




---

**Note** Provider/Consumer consistent UUIDs and policy identifiers do not uniquely identify a filter or policy as they are shared across different application workspace versions.

---

To perform CRUD operations on a particular cluster or policy it is recommended to resolve the identifier to a concrete policy for a specific application workspace version the search endpoint.

Regular CRUD operations can be performed using policy IDs whereas only Policy Statistics and Destroy With Identifier accept the policy identifier as input. This is mainly for convenience to avoid the intermediate call to search, and instead directly validate and destroy all unused policies in a workspace.

It is strongly recommended that policy and filter IDs are used wherever possible and to not manually generate policy identifiers for the Policy Statistics or Destroy With Identifier API endpoints. However, the following example illustrates one way of generating policy identifiers from the policy object:

```
resp = restclient.get(f'/policies/631b0590497d4f09b537b973')
policy = resp.json() # policy object
policy_identifier = {
 'consumer_consistent_uuid': policy['consumer_filter']['consistent_uuid'],
 'provider_consistent_uuid': policy['provider_filter']['consistent_uuid'],
 'rank': policy['rank'],
 'action': policy['action'],
 'priority': policy['priority'],
 'protocol': policy['l4_params'][0]['proto'],
 'start_port': policy['l4_params'][0]['port'][0],
 'end_port': policy['l4_params'][0]['port'][1]
}
```




---

**Note** After a new version of the application workspace is published it can take up to 6 hours before results become available. All the timestamp resolutions will also have a minimum granularity of 6 hours.

---

To get the unused policies across enforced versions of an application workspace the URL path is:

```
POST /openapi/v1/unused_policies/{application_id}/enforced
```

To get the unused policies across analyzed versions of an application workspace the URL path is:

```
POST /openapi/v1/unused_policies/{application_id}/analyzed
```

The query body consists of a JSON body with the following schema:

| Name   | Type    | Description                                                                                               |
|--------|---------|-----------------------------------------------------------------------------------------------------------|
| t0     | string  | The beginning of the time interval in RFC-3339 format.                                                    |
| t1     | string  | (Optional) The end of the time interval in RFC-3339 format; defaults to current time if left unspecified. |
| limit  | integer | (Optional) Limits the number of policies per request.                                                     |
| offset | string  | (Optional) Offset received from previous response – useful for pagination.                                |

```
application_id = '62e1915e755f026f2bccdd805'
resp = restclient.post(f'/unused_policies/{application_id}/analyzed', json_body=f'''
{{
 "t0": "2022-07-06T00:00:00Z",
 "t1": "2022-07-28T19:00:00Z"
}}''')
```

### Sample response

The response is a JSON object in the body with the following properties.

| Keys               | Values                                                     |
|--------------------|------------------------------------------------------------|
| application_id     | The ID of the application workspace.                       |
| policy_identifiers | A list of policy identifiers of the unused policies.       |
| offset             | Response offset to be passed for the next page of results. |

To generate the next page of results, take the object received by the response in offset and pass it as the value for the offset of the next query.

```
{
 "application_id": "63054a97497d4f2dc113a9c4",
 "policy_identifiers": [
 {
 "consumer_consistent_uuid": "62fff45c497d4f5064973c4d",
 "provider_consistent_uuid": "62fff45c497d4f5064973c4d",
 "version": "p1",
 "rank": "DEFAULT",
 "policy_action": "ALLOW",
 "priority": 10,
 "proto": 6,
 "start_port": 10000,
 "end_port": 10000,
 "agg_start_version": 1,
 "agg_start_time": "2022-08-10T23:00:00.000Z"
 }
]
}
```

```

 },
 {
 "consumer_consistent_uuid": "62fff45c497d4f5064973c4d",
 "provider_consistent_uuid": "62fff45c497d4f5064973c4d",
 "version": "p1",
 "rank": "DEFAULT",
 "policy_action": "ALLOW",
 "priority": 10,
 "protocol": 6,
 "start_port": 10001,
 "end_port": 10001,
 "agg_start_version": 1,
 "agg_start_time": "2022-08-10T23:00:00.000Z"
 }
],
 "offset": "eyJvZmZzZXQiOjZ9"
}

```

## Policy Templates

This set of APIs can be used to add, edit or delete Policy Templates and require the `app_policy_management` capability associated with the API key.

### Get Policy Templates

This endpoint returns a list of policy templates for a particular root scope. This API is available to API keys with `app_policy_management` capability.

```
GET /openapi/v1/application_templates?root_app_scope_id={root_app_scope_id}
```

Parameters: The request URL contains the following parameters

| Name              | Type   | Description                              |
|-------------------|--------|------------------------------------------|
| root_app_scope_id | string | The unique identifier of the root scope. |

Response object: Returns a list of policy template objects for the specified root scope.

#### Sample python code

```

root_app_scope_id = '<root-app-scope-id>'
restclient.get('/application_templates?root_app_scope_id=%s' % root_app_scope_id)

```

### Get Specific Policy Template

This endpoint returns an instance of policy templates.

```
GET /openapi/v1/application_templates/{template_id}
```

Parameters: The request URL contains the following parameters

| Name        | Type   | Description                                    |
|-------------|--------|------------------------------------------------|
| template_id | string | The unique identifier for the policy template. |



Response object: Returns the policy template object with the specified ID.

### Sample python code

```
template_id = '<template-id>'
restclient.get('/application_templates/%s' % template_id)
```

## Create a Policy Template

This endpoint is used to create a new policy template.

POST /openapi/v1/application\_templates

The JSON request body contains the following keys

| Attribute         | Type                    | Description                                                        |
|-------------------|-------------------------|--------------------------------------------------------------------|
| name              | string                  | Used as the name of the template during import.                    |
| description       | string                  | (optional) Template description displayed during the apply process |
| parameters        | parameters object       | Template parameters, see below.                                    |
| absolute_policies | array of policy objects | (optional) Array of absolute policies.                             |
| default_policies  | array of policy objects | (required) Array of default policies, can be empty.                |

Response object: Returns the created policy template object.

### Sample python code

```
root_app_scope_id = '<root-app-scope-id>'
payload = {'root_app_scope_id': root_app_scope_id,
 'name': "policy_name",
 'default_policies': [
 {
 'action': 'ALLOW',
 'priority': 100,
 'l4_params': [
 {
 'proto': 17,
 'port': [80, 90]
 }
]
 }
]
 }
restclient.post('/application_templates',
 json_body=json.dumps(payload))
```

## Update a Policy Template

This endpoint updates a policy template.

```
PUT /openapi/v1/application_templates/{template_id}
```

Parameters: The request URL contains the following parameters

| Name        | Type   | Description                                    |
|-------------|--------|------------------------------------------------|
| template_id | string | The unique identifier for the policy template. |

The JSON request body contains the following keys

| Attribute   | Type   | Description                                                        |
|-------------|--------|--------------------------------------------------------------------|
| name        | string | (optional) Used as the name of the template during import.         |
| description | string | (optional) Template description displayed during the apply process |

Response object: Returns the modified policy template object with the specified ID.

#### Sample python code

```
new_name = <new-name>
payload = {'name': new_name}
template_id = '<template-id>'
restclient.post('/application_templates/%s' % template_id,
 json_body=json.dumps(payload))
```

## Deleting a Policy Template

This endpoint deletes the specified policy template.

```
DELETE /openapi/v1/application_templates/{template_id}
```

Parameters: The request URL contains the following parameters

| Name        | Type   | Description                                    |
|-------------|--------|------------------------------------------------|
| template_id | string | The unique identifier for the policy template. |

Response object: None

#### Sample python code

```
template_id = '<template-id>'
restclient.delete('/application_templates/%s' % template_id)
```

## Download a Policy Template

This endpoint downloads a policy template.

```
GET /openapi/v1/application_templates/{template_id}/download
```

Parameters: The request URL contains the following parameters

| Name        | Type   | Description                                    |
|-------------|--------|------------------------------------------------|
| template_id | string | The unique identifier for the policy template. |

Response object: Returns the full policy template definition with the specified ID.

#### Sample python code

```
template_id = '<template-id>'
restclient.get('/application_templates/%s/download' % template_id)
```

## Clusters

This set of APIs can be used to add, edit or delete Clusters, which are members of workspaces (“applications”). They require the `user_role_scope_management` capability associated with the API key.

### Cluster object

The cluster object attributes are described below:

| Attribute         | Type             | Description                                                                                                 |
|-------------------|------------------|-------------------------------------------------------------------------------------------------------------|
| id                | string           | Unique identifier for the cluster.                                                                          |
| consistent_uuid   | string           | An id that is consistent across automatic policy discovery runs.                                            |
| application_id    | string           | The id for the workspace to which the cluster belongs.                                                      |
| version           | string           | The version of the workspace to which the cluster belongs                                                   |
| name              | string           | The name of the cluster.                                                                                    |
| description       | string           | The description of the cluster.                                                                             |
| approved          | boolean          | If the cluster has been ‘approved’ by the user.                                                             |
| query             | JSON             | Filter (or match criteria) associated with the filter in conjunction with the filters of the parent scopes. |
| short_query       | JSON             | Filter (or match criteria) associated with the filter.                                                      |
| alternate_queries | array of queries | Alternate suggested queries generated by an automatic policy discovery run in dynamic mode.                 |

| Attribute | Type               | Description                                                                                    |
|-----------|--------------------|------------------------------------------------------------------------------------------------|
| inventory | array of inventory | If requested, returns member inventory of the cluster including IP, hostname, vrf_id and uuid. |

## Get Clusters

This endpoint returns a list of clusters for a particular workspace (“application”). This API is available to API keys with `app_policy_management` capability.

```
GET /openapi/v1/applications/{application_id}/clusters
```

Parameters: The request URL contains the following parameters

| Name              | Type    | Description                                                             |
|-------------------|---------|-------------------------------------------------------------------------|
| application_id    | string  | The id for the workspace to which the cluster belongs.                  |
| version           | string  | Indications the version of the workspace for which to get the clusters. |
| include_inventory | boolean | Include the inventory of the clusters.                                  |

Response object: Returns an array of all clusters for this particular workspace and version.

### Sample python code

```
application_id = '5d02b493755f0237a3d6e078'
restclient.get('/applications/%s/clusters' % application_id)
```

## Get Specific Cluster

This endpoint returns an instance of a cluster.

```
GET /openapi/v1/clusters/{cluster_id}
```

Parameters: The request URL contains the following parameters

| Name              | Type    | Description                            |
|-------------------|---------|----------------------------------------|
| cluster_id        | string  | Unique identifier for the cluster.     |
| include_inventory | boolean | Include the inventory of the clusters. |

Response object: Returns the cluster object associated for the specified ID.

### Sample python code

```
cluster_id = '5d02d021497d4f0949ba74e4'
```

```
restclient.get('/clusters/%s' % cluster_id)
```

## Create a Cluster

This endpoint is used to create a new cluster.

```
POST /openapi/v1/applications/{application_id}/clusters
```

Parameters: The request URL contains the following parameters

| Name           | Type   | Description                                            |
|----------------|--------|--------------------------------------------------------|
| application_id | string | The id for the workspace to which the cluster belongs. |

The JSON query body contains the following keys

| Attribute   | Type    | Description                                                                                                                                                 |
|-------------|---------|-------------------------------------------------------------------------------------------------------------------------------------------------------------|
| name        | string  | The name of the cluster.                                                                                                                                    |
| version     | string  | Indicates the version of the workspace the cluster will be added to.                                                                                        |
| description | string  | (optional) The description of the cluster.                                                                                                                  |
| approved    | boolean | (optional) An approved cluster will not be updated during an automatic policy discovery run. Default false.                                                 |
| query       | JSON    | Filter (or match criteria) associated with the filter. Alternate Query Mode (also called Dynamic Mode) must be enabled on the workspace, otherwise ignored. |
| query       | JSON    | Filter (or match criteria) associated with the filter. Alternate Query Mode (also called Dynamic Mode) must be enabled on the workspace, otherwise ignored. |
| nodes       | Array   | List of ip addresses or endpoints. Will be used to create the query matching these ips unless a query is provided and the workspace is in Dynamic Mode.     |

Nodes object attributes:

| Name       | Type    | Description                      |
|------------|---------|----------------------------------|
| ip         | string  | IP address                       |
| name       | string  | (optional) The name of the node. |
| prefix_len | integer | (optional) Subnet mask.          |



**Note** The nodes will be used to create a query unless a query is provided and the workspace is in Dynamic Mode.

Response object: Returns the newly created cluster object.

#### Sample python code

```
application_id = '5d02b493755f0237a3d6e078'
payload = {
 'name': 'test_cluster',
 'version': 'v2',
 'description': 'basic granularity',
 'approved': False,
 'query': {
 'type': 'eq',
 'field': 'host_name',
 'value': 'centos6001'
 }
}
restclient.post('/applications/%s/clusters' % application_id)
```

## Update a Cluster

This endpoint updates a cluster.

PUT /openapi/v1/clusters/{cluster\_id}

Parameters: The request URL contains the following parameters

| Name       | Type   | Description                        |
|------------|--------|------------------------------------|
| cluster_id | string | Unique identifier for the cluster. |

The JSON query body contains the following keys

| Attribute   | Type    | Description                                                                       |
|-------------|---------|-----------------------------------------------------------------------------------|
| name        | string  | The name of the cluster.                                                          |
| description | string  | (optional) The description of the cluster.                                        |
| approved    | boolean | An approved cluster will not be updated during an automatic policy discovery run. |

| Attribute | Type | Description                                                                                                                                                 |
|-----------|------|-------------------------------------------------------------------------------------------------------------------------------------------------------------|
| query     | JSON | Filter (or match criteria) associated with the filter. Alternate Query Mode (also called Dynamic Mode) must be enabled on the workspace, otherwise ignored. |

Response object: Returns the modified cluster object associated with specified ID.

#### Sample python code

```
cluster_id = '5d02d2a4497d4f5194f104ef'
payload = {
 'name': 'new_test_cluster',
}
restclient.put('/clusters/%s' % cluster_id, json_body=json.dumps(payload))
```

## Deleting a Cluster

This endpoint deletes the specified Cluster. If the cluster is used by any policies the cluster will not be deleted and a list of dependents will be returned.

```
DELETE /openapi/v1/clusters/{cluster_id}
```

Parameters: The request URL contains the following parameters

| Name       | Type   | Description                        |
|------------|--------|------------------------------------|
| cluster_id | string | Unique identifier for the cluster. |

Response object: None

#### Sample python code

```
cluster_id = '5d02d2a4497d4f5194f104ef'
restclient.delete('/clusters/%s' % cluster_id)
```

## Conversations

Conversations are aggregated flows in the time range of an automatic policy discovery run where the consumer port is removed. More detailed description about the conversations can be found in [Conversations](#).

This API enables you to search the conversations generated during an automatic policy discovery run for a given workspace. It requires `app_policy_management` capability associated with the API key to invoke this API.

### Search Conversations in a Policy Discovery Run

This end point enables you to search the conversations in an automatic policy discovery run for a given workspace. You can also specify a subset of supported dimensions and metrics which you may want to see as part of the downloaded conversations. Optionally, you can query for a subset of conversations using filters on supported dimensions and metrics.

POST /openapi/v1/conversations/{application\_id}

The query consists of a JSON body with the following keys.

| Name       | Type    | Description                                                                                                                                                                                                                                                    |
|------------|---------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| version    | integer | Version of the automatic policy discovery run                                                                                                                                                                                                                  |
| filter     | JSON    | (optional) Query filter. If filter is empty (i.e. {}), then query matches all the conversations. More specific conversations can be downloaded using filters on supported dimensions and metrics. For the syntax on filters refer to <a href="#">Filters</a> . |
| dimensions | array   | (optional) List of dimensions to be returned for the downloaded conversations. The list of supported dimension can be found <a href="#">Supported Dimensions</a> .                                                                                             |
| metrics    | array   | (optional) List of metrics to be returned for the downloaded conversations. The list of supported metrics can be found <a href="#">Supported metrics</a> .                                                                                                     |
| limit      | integer | (optional) Number of conversations to be returned in a single API response.                                                                                                                                                                                    |
| offset     | string  | (optional) Offset received from previous response – useful for pagination.                                                                                                                                                                                     |

The body of the request should be a JSON formatted query. An example of a query body is shown below.

```
{
 "version": 1,
 "filter": {
 "type": "and",
 "filters": [
 {
 "type": "eq",
 "field": "excluded",
 "value": False
 },
 {
 "type": "eq",
 "field": "protocol",
 "value": "TCP"
 }
]
 },
 "dimensions": ["src_ip", "dst_ip", "port"],
}
```



```

 "metrics": ["byte_count", "packet_count"],
 "limit" : 2,
 "offset": <offset-object>
 }

```

## Response

The response is a JSON object in the body with the following properties.

| Keys    | Values                                                    |
|---------|-----------------------------------------------------------|
| offset  | Response offset to be passed for the next page of results |
| results | List of results                                           |

To generate the next page of results, take the object received by the response in `offset` and pass it as the value for the `offset` of the next query.

```

req_payload = {"version": 1,
 "limit": 10,
 "filter": {"type": "and",
 "filters": [
 {"type": "eq", "field": "excluded", "value": False},
 {"type": "eq", "field": "protocol", "value": "TCP"}
]
 }

resp = restclient.post('/conversations/{application_id}',
 json_body=json.dumps(req_payload))
print resp.status_code
if resp.status_code == 200:
 parsed_resp = json.loads(resp.content)
 print json.dumps(parsed_resp, indent=4, sort_keys=True)

```

## Top N Conversations in a Policy Discovery Run

This endpoint enables you to search the top conversations for an automatic policy discovery that is run for a given workspace based on a metric and grouped by a dimension. The current supported metrics are [Supported metrics](#) and the current supported group by dimensions are [Supported Dimensions](#) you can query for a subset of conversations using filters on supported dimensions and metrics. For example, you can search for the source IP address with the most byte traffic conversations using a query with the `src_ip` dimension with the `byte_count` metric.

```
POST /openapi/v1/conversations/{application_id}/topn
```

The query consists of a JSON body with the following keys.

| Name    | Type    | Description                                   |
|---------|---------|-----------------------------------------------|
| version | integer | Version of the automatic policy discovery run |

| Name      | Type    | Description                                                                                                                                                                                                                                                |
|-----------|---------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| dimension | string  | The dimension for the conversations to be grouped by for the top N query.<br><br>Supported dimensions: src_ip, dst_ip                                                                                                                                      |
| metric    | string  | The metric to be sorted by for the top N conversations. The list of supported metrics can be found <a href="#">Supported metrics</a> .                                                                                                                     |
| filter    | JSON    | (optional) Query filter. If filter is empty (i.e. {}), then query matches all the conversations. More specific conversations can be downloaded using filters on supported dimensions and metrics. For the syntax on filters, see <a href="#">Filters</a> . |
| threshold | integer | Number of top N results to be returned in a single API response.                                                                                                                                                                                           |

The body of the request should be a JSON-formatted query. An example of a query body is shown below.

```
{
 "version": 1,
 "dimension": "src_ip",
 "metric": "byte_count",
 "filter": {
 "type": "and",
 "filters": [
 {
 "type": "eq",
 "field": "excluded",
 "value": False
 },
 {
 "type": "eq",
 "field": "protocol",
 "value": "TCP"
 }
]
 },
 "threshold": 10
}
```

## Response

The response is a JSON object in the body with the following properties.

| Keys    | Values                                                                                                                                   |
|---------|------------------------------------------------------------------------------------------------------------------------------------------|
| results | List with one JSON object with a results key and a value of a list of results objects with keys matching the query dimension and metric. |

```
[{"result": [
 {
 "byte_count": 1795195565,
 "src_ip": "192.168.1.6"
 },
 {
 "byte_count": 1781002379,
 "src_ip": "192.168.1.28"
 },
 ...
] }]

req_payload = {"version": 1, "dimension": "src_ip", "metric": "byte_count",
 "filter": {"type": "and",
 "filters": [
 {"type": "eq", "field": "excluded", "value": False},
 {"type": "eq", "field": "protocol", "value": "TCP"},
 {"type": "eq", "field": "consumer_filter_id", "value": "16b12a5614c5af5b68afa7ce"},

 {"type": "subnet", "field": "src_ip", "value": "192.168.1.0/24"}
]
 },
 "threshold" : 10
}

resp = restclient.post('/conversations/{application_id}/topn',
json_body=json.dumps(req_payload))
print resp.status_code
if resp.status_code == 200:
 parsed_resp = json.loads(resp.content)
 print json.dumps(parsed_resp, indent=4, sort_keys=True)
```

## Supported Dimensions

| Name         | Type    | Description                                                    |
|--------------|---------|----------------------------------------------------------------|
| src_ip       | string  | IP address of the consumer                                     |
| dst_ip       | string  | IP address of the provider                                     |
| protocol     | string  | Protocol used in the communication. Ex: "TCP", "UDP" and so on |
| port         | integer | Port of the provider.                                          |
| address_type | string  | "IPv4" or "IPv6"                                               |

| Name               | Type    | Description                                                                                                                                        |
|--------------------|---------|----------------------------------------------------------------------------------------------------------------------------------------------------|
| consumer_filter_id | string  | Cluster ID of the cluster if the consumer IP belongs to a cluster, else the Scope ID the consumer IP belongs to.                                   |
| provider_filter_id | string  | Cluster ID of the cluster if the provider IP belongs to a cluster, else the Scope ID the provider IP belongs to.                                   |
| excluded           | boolean | Whether this conversation is excluded while generating policies.                                                                                   |
| confidence         | double  | The confidence level of consumer and provider classification. The value varies from 0.0 to 1.0 with 1.0 being more confident about classification. |

## Supported metrics

| Name         | Type    | Description                                 |
|--------------|---------|---------------------------------------------|
| byte_count   | integer | Total number of bytes in the conversation   |
| packet_count | integer | Total number of packets in the conversation |

## Exclusion Filters

This set of APIs can be used to add, edit or delete Exclusion Filters and require the `user_role_scope_management` capability associated with the API key.

Exclusion Filters exclude flows from the automatic policy discovery clustering algorithm. See [Exclusion Filters](#) for more information.

## Exclusion Filter object

The exclusion filter object attributes are described below:

| Attribute      | Type   | Description                                                         |
|----------------|--------|---------------------------------------------------------------------|
| id             | string | Unique identifier for the cluster.                                  |
| application_id | string | The id for the workspace to which the exclusion filter belongs.     |
| version        | string | The version of the workspace to which the exclusion filter belongs. |

| Attribute          | Type    | Description                                                                                                                                      |
|--------------------|---------|--------------------------------------------------------------------------------------------------------------------------------------------------|
| consumer_filter_id | string  | ID of a defined filter. Currently, any cluster belonging to the workspace, user defined filter or scope can be used as the consumer of a policy. |
| provider_filter_id | string  | ID of a defined filter. Currently, any cluster belonging to the workspace, user defined filter or scope can be used as the provider of a policy. |
| proto              | integer | Protocol Integer value (NULL means all protocols).                                                                                               |
| port               | array   | Inclusive range of ports. eg [80, 80] or [5000, 6000]. NULL means all ports.                                                                     |
| updated_at         | integer | Unix timestamp of when the exclusion filter was updated.                                                                                         |

## Get Exclusion Filters

This endpoint returns a list of exclusion filters for a particular workspace. This API is available to API keys with `app_policy_management` capability.

GET `/openapi/v1/applications/{application_id}/exclusion_filters`

Parameters: The request URL contains the following parameters

| Name           | Type   | Description                                                                    |
|----------------|--------|--------------------------------------------------------------------------------|
| application_id | string | The unique identifier for the workspace.                                       |
| version        | string | Indicates the version of the workspace for which to get the exclusion filters. |

Response object: Returns a list of exclusion filter objects for the specified workspace and version.

### Sample python code

```
application_id = '<application-id>'
params = {'version': 'v10'}
restclient.get('/applications/%s/exclusion_filters' % application_id,
 params=params)
```

## Get Specific Exclusion Filter

This endpoint returns an instance of an exclusion filters.

```
GET /openapi/v1/exclusion_filters/{exclusion_filter_id}
```

Parameters: The request URL contains the following parameters

| Name                | Type   | Description                                     |
|---------------------|--------|-------------------------------------------------|
| exclusion_filter_id | string | The unique identifier for the exclusion filter. |

Response object: Returns the exclusion filter object with the specified ID.

#### Sample python code

```
exclusion_filter_id = '<exclusion-filter-id>'
restclient.get('/exclusion_filters/%s' % exclusion_filter_id)
```

## Create an Exclusion Filter

This endpoint is used to create a new exclusion filter.

```
POST /openapi/v1/applications/{application_id}/exclusion_filters
```

Parameters: The request URL contains the following parameters

| Name           | Type   | Description                              |
|----------------|--------|------------------------------------------|
| application_id | string | The unique identifier for the workspace. |

The JSON request body contains the following keys

| Attribute          | Type    | Description                                                                                                                                                 |
|--------------------|---------|-------------------------------------------------------------------------------------------------------------------------------------------------------------|
| version            | string  | The version of the workspace to which the exclusion filter belongs.                                                                                         |
| consumer_filter_id | string  | (optional) ID of a defined filter. Currently, any cluster belonging to the workspace, user defined filter or scope can be used as the consumer of a policy. |
| provider_filter_id | string  | (optional) ID of a defined filter. Currently, any cluster belonging to the workspace, user defined filter or scope can be used as the provider of a policy. |
| proto              | integer | (optional) Protocol Integer value (NULL means all protocols).                                                                                               |
| start_port         | integer | (optional) Start port of the range.                                                                                                                         |
| end_port           | integer | (optional) End port of the range.                                                                                                                           |

Missing optional parameters will be considered as wildcards (match any).

Response object: Returns the created exclusion filter object.

#### Sample python code

```
provider_filter_id = '<provider-filter-id>'
consumer_filter_id = '<consumer-filter-id>'
payload = {'version': 'v0',
 'consumer_filter_id': consumer_filter_id,
```

```

 'provider_filter_id': provider_filter_id,
 'proto': 6,
 'start_port': 800,
 'end_port': 1000}
application_id = '<application-id>'
restclient.post('/applications/%s/exclusion_filters' % application_id,
 json_body=json.dumps(payload))

```

## Update an Exclusion Filter

This endpoint updates an exclusion filter.

```
PUT /openapi/v1/exclusion_filters/{exclusion_filter_id}
```

Parameters: The request URL contains the following parameters

| Name                | Type   | Description                                     |
|---------------------|--------|-------------------------------------------------|
| exclusion_filter_id | string | The unique identifier for the exclusion filter. |

The JSON request body contains the following keys

| Attribute          | Type    | Description                                                                                                                                                 |
|--------------------|---------|-------------------------------------------------------------------------------------------------------------------------------------------------------------|
| consumer_filter_id | string  | (optional) ID of a defined filter. Currently, any cluster belonging to the workspace, user defined filter or scope can be used as the consumer of a policy. |
| provider_filter_id | string  | (optional) ID of a defined filter. Currently, any cluster belonging to the workspace, user defined filter or scope can be used as the provider of a policy. |
| proto              | integer | Protocol Integer value (NULL means all protocols).                                                                                                          |
| start_port         | integer | (optional) Start port of the range.                                                                                                                         |
| end_port           | integer | (optional) End port of the range.                                                                                                                           |

Response object: Returns the modified exclusion filter object with the specified ID.

### Sample python code

```

payload = {'proto': 17}
exclusion_filter_id = '<exclusion-filter-id>'
restclient.post('/exclusion_filters/%s' % exclusion_filter_id,
 json_body=json.dumps(payload))

```

## Deleting an Exclusion Filter

This endpoint deletes the specified exclusion filter.

```
DELETE /openapi/v1/exclusion_filters/{exclusion_filter_id}
```

Parameters: The request URL contains the following parameters

| Name                | Type   | Description                                     |
|---------------------|--------|-------------------------------------------------|
| exclusion_filter_id | string | The unique identifier for the exclusion filter. |

Response object: None

#### Sample python code

```
exclusion_filter_id = '<exclusion-filter-id>'
restclient.delete('/exclusion_filters/%s' % exclusion_filter_id)
```

## Default Exclusion Filters

This set of APIs can be used to add, edit or delete Default Exclusion Filters and require the `app_policy_management` capability associated with the API key.

Exclusion Filters exclude flows from the automatic policy discovery clustering algorithm. See [Exclusion Filters](#) for more information.

### Default Exclusion Filter object

The exclusion filter object attributes are described below:

| Attribute          | Type    | Description                                                                                                                                      |
|--------------------|---------|--------------------------------------------------------------------------------------------------------------------------------------------------|
| id                 | string  | Unique identifier for the default exclusion filter.                                                                                              |
| consumer_filter_id | string  | ID of a defined filter. Currently, any cluster belonging to the workspace, user defined filter or scope can be used as the consumer of a policy. |
| provider_filter_id | string  | ID of a defined filter. Currently, any cluster belonging to the workspace, user defined filter or scope can be used as the provider of a policy. |
| proto              | integer | Protocol Integer value (NULL means all protocols).                                                                                               |
| port               | array   | Inclusive range of ports. eg [80, 80] or [5000, 6000]. NULL means all ports.                                                                     |
| updated_at         | integer | Unix timestamp of when the exclusion filter was updated.                                                                                         |



## Get Default Exclusion Filters

This endpoint returns a list of default exclusion filters. This API is available to API keys with `app_policy_management` capability.

```
GET /openapi/v1/default_exclusion_filters?root_app_scope_id={root_app_scope_id}
```

Parameters: The request URL contains the following parameters

| Name              | Type   | Description                              |
|-------------------|--------|------------------------------------------|
| root_app_scope_id | string | The unique identifier of the root scope. |

Response object: Returns a list of default exclusion filter objects for the root scope.

### Sample python code

```
root_app_scope_id = '<root-app-scope-id>'
restclient.get('/default_exclusion_filters?root_app_scope_id=%s' % root_app_scope_id)
```

## Get Specific Default Exclusion Filter

This endpoint returns an instance of a default exclusion filters.

```
default_exclusion_filter_id = '<default-exclusion-filter-id>'
```

```
restclient.get('/default_exclusion_filters/%s' % default_exclusion_filter_id)
```

Parameters: The request URL contains the following parameters

| Name                        | Type   | Description                                     |
|-----------------------------|--------|-------------------------------------------------|
| default_exclusion_filter_id | string | The unique identifier for the exclusion filter. |

Response object: Returns the default exclusion filter object with the specified ID.

### Sample python code

```
default_exclusion_filter_id = '<default-exclusion-filter-id>'
restclient.get('/default_exclusion_filters/%s' % default_exclusion_filter_id)
```

## Create a Default Exclusion Filter

This endpoint is used to create a new default exclusion filter.

```
POST /openapi/v1/default_exclusion_filters?root_app_scope_id={root_app_scope_id}
```

Parameters: The request URL contains the following parameters

| Name              | Type   | Description                              |
|-------------------|--------|------------------------------------------|
| root_app_scope_id | string | The unique identifier of the root scope. |

The JSON request body contains the following keys

| Attribute          | Type    | Description                                                   |
|--------------------|---------|---------------------------------------------------------------|
| consumer_filter_id | string  | (optional) ID of a defined scope or inventory filter.         |
| provider_filter_id | string  | (optional) ID of a defined scope or inventory filter.         |
| proto              | integer | (optional) Protocol Integer value (NULL means all protocols). |
| start_port         | integer | (optional) Start port of the range.                           |
| end_port           | integer | (optional) End port of the range.                             |

Response object: Returns the created default exclusion filter object.

#### Sample python code

```

provider_filter_id = '<provider-filter-id>'
consumer_filter_id = '<consumer-filter-id>'
payload = {'consumer_filter_id': consumer_filter_id,
 'provider_filter_id': provider_filter_id,
 'proto': 6,
 'start_port': 800,
 'end_port': 1000}
root_app_scope_id = '<root-app-scope-id>'
restclient.post('/default_exclusion_filters?root_app_scope_id=%s' % root_app_scope_id,
 json_body=json.dumps(payload))

```

## Update a Default Exclusion Filter

This endpoint updates a default exclusion filter.

```
PUT /openapi/v1/default_exclusion_filters/{default_exclusion_filter_id}
```

Parameters: The request URL contains the following parameters

| Name                        | Type   | Description                                             |
|-----------------------------|--------|---------------------------------------------------------|
| default_exclusion_filter_id | string | The unique identifier for the default exclusion filter. |

The JSON request body contains the following keys

| Attribute          | Type    | Description                                           |
|--------------------|---------|-------------------------------------------------------|
| consumer_filter_id | string  | (optional) ID of a defined scope or inventory filter. |
| provider_filter_id | string  | (optional) ID of a defined scope or inventory filter. |
| proto              | integer | Protocol Integer value (NULL means all protocols).    |
| start_port         | integer | (optional) Start port of the range.                   |

|          |         |                                   |
|----------|---------|-----------------------------------|
| end_port | integer | (optional) End port of the range. |
|----------|---------|-----------------------------------|

Response object: Returns the modified default exclusion filter object with the specified ID.

#### Sample python code

```
payload = {'proto': 17}
default_exclusion_filter_id = '<default-exclusion-filter-id>'
restclient.post('/default_exclusion_filters/%s' % default_exclusion_filter_id,
 json_body=json.dumps(payload))
```

## Deleting a Default Exclusion Filter

This endpoint deletes the specified default exclusion filter.

```
DELETE /openapi/v1/default_exclusion_filters/{default_exclusion_filter_id}
```

Parameters: The request URL contains the following parameters

| Name                        | Type   | Description                                     |
|-----------------------------|--------|-------------------------------------------------|
| default_exclusion_filter_id | string | The unique identifier for the exclusion filter. |

Response object: None

#### Sample python code

```
default_exclusion_filter_id = '<default-exclusion-filter-id>'
restclient.delete('/default_exclusion_filters/%s' % default_exclusion_filter_id)
```

## Live Analysis

Live analysis or Policy Analysis is an important aspect of generating security policies. It allows you to evaluate the impact of a set of policies – where generated by automatic policy discovery or manually added by users – before actually enforcing those policies on the workloads. Live analysis allows users to run what-if analysis on live traffic without disrupting any application traffic.

The set of APIs available in this section allow downloading flows and the effect of current set of published policies in a workspace on those flows. It requires `app_policy_management` capability associated with the API key to invoke these set of APIs.

Flows available via Live Analysis have some attributes (dimensions and metrics) and the download API allows user to filter flows by different criteria on dimensions.

### Flow dimensions available in Live Analysis

This endpoint is useful to know the columns on which search criteria (or *filters*) can be specified for downloading flows available via Live Analysis. Most common use case would be to download *permitted*, *escaped* or *rejected* flows this can be achieved by passing a search criteria on category dimension to the download API. When used with **type: eq** the flow's inbound and outbound category must match. When used with **type: contains** the flows inbound or outbound category must match

```
GET /openapi/v1/live_analysis/dimensions
```

## Flow metrics available in Live Analysis

This endpoint returns the list of metrics (e.g. byte count, packet count) associated with live analysis. One use case for this endpoint would be to project a subset of metrics in the download API, i.e. instead of downloading all the metrics, users can specify a small subset of metrics they are interested in.

```
GET /openapi/v1/live_analysis/metrics
```

## Download flows available via Live Analysis

This endpoint returns the list of flows matching the filter criteria. Each flow object in the result has attributes that are a union of live analysis dimensions (returned by the live analysis dimensions API above) as well as the live analysis metrics (returned by the live analysis metrics API above). Optionally, user can also specify a small subset of dimensions or metrics if they are not interested in the full set of available dimensions and metrics – this projection of a smaller subset of dimensions or metrics also have the side effect of making API calls fast.

```
POST /openapi/v1/live_analysis/{application_id}
```

The query body consists of a JSON body with the following keys.

| Name       | Type              | Description                                                                                                                                                        |
|------------|-------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| t0         | integer or string | Start of time interval (epoch or ISO 8601)                                                                                                                         |
| t1         | integer or string | End of time interval (epoch or ISO 8601)                                                                                                                           |
| filter     | JSON              | Query filter. If filter is empty (i.e. {}), then query matches all flows. Refer to section on <a href="#">Filters</a> in Flow Search regarding syntax of filters.  |
| dimensions | array             | (optional) List of flow dimensions to be returned for the downloaded flows available through Live Analysis. If unspecified, all available dimensions are returned. |
| metrics    | array             | (optional) List of flow metrics to be returned for the downloaded flows available through Live Analysis.                                                           |
| limit      | integer           | (optional) Number of flows to be returned in a single API response.                                                                                                |
| offset     | string            | (optional) Offset received from previous response – useful for pagination.                                                                                         |

The body of the request should be a JSON formatted query. An example of a query body is shown below.

```

{
 "t0": "2016-06-17T09:00:00-0700",
 "t1": "2016-06-17T17:00:00-0700",
 "filter": {
 "type": "and",
 "filters": [
 {
 "type": "contains",
 "field": "category",
 "value": "escaped"
 },
 {
 "type": "in",
 "field": "dst_port",
 "values": ["80", "443"]
 }
]
 },
 "limit": 100,
 "offset": <offset-object>
}

```

## Response

The response is a JSON object in the body with the following properties.

| Keys    | Values                                                    |
|---------|-----------------------------------------------------------|
| offset  | Response offset to be passed for the next page of results |
| results | List of results                                           |

To generate the next page of results, take the object received by the response in `offset` and pass it as the value for the `offset` of the next query.

## Sample python code

```

req_payload = {"t0": "2016-11-07T09:00:00-0700",
 "t1": "2016-11-07T19:00:00-0700",
 "limit": 10,
 "filter": {"type": "and",
 "filters": [
 {"type": "contains", "field": "category", "value": "escaped"},
 {"type": "regex", "field": "src_hostname", "value": "web*"}
]
 }

resp = restclient.post('/live_analysis/{application_id}',
 json_body=json.dumps(req_payload))
print resp.status_code
if resp.status_code == 200:
 parsed_resp = json.loads(resp.content)
 print json.dumps(parsed_resp, indent=4, sort_keys=True)

```

# Scopes

This set of APIs can be used to manage Scopes (or AppScopes) in Secure Workload cluster deployment. They require the `user_role_scope_management` capability associated with the API key. The API to get the list of scopes is also available to API keys with `app_policy_management` or `sensor_management` capability.

## Scope object

The scope object attributes are described below:

| Attribute           | Type    | Description                                                                                                                                         |
|---------------------|---------|-----------------------------------------------------------------------------------------------------------------------------------------------------|
| id                  | string  | Unique identifier for the scope.                                                                                                                    |
| short_name          | string  | User specified name of the scope.                                                                                                                   |
| name                | string  | Fully qualified name of the scope. This is a fully qualified name, i.e. it has name of parent scopes (if applicable) all the way to the root scope. |
| description         | string  | User specified description of the scope.                                                                                                            |
| short_query         | JSON    | Filter (or match criteria) associated with the scope.                                                                                               |
| query               | JSON    | Filter (or match criteria) associated with the scope in conjunction with the filters of the parent scopes (all the way to the root scope).          |
| vrf_id              | integer | ID of the VRF to which scope belongs to.                                                                                                            |
| parent_app_scope_id | string  | ID of the parent scope.                                                                                                                             |
| child_app_scope_ids | array   | An array of scope children's ids.                                                                                                                   |
| policy_priority     |         | Used to sort workspace priorities. See <a href="#">Policy Attributes</a> .                                                                          |
| dirty               | bool    | Indicates a child or parent query has been updated and that the changes need to be committed.                                                       |
| dirty_short_query   | JSON    | Non-null if the query for this scope has been updated but not yet committed.                                                                        |

## Get scopes

This endpoint returns a list of scopes known to Secure Workload appliance. This API is available to API keys with either `app_policy_management` or `user_role_scope_management` capability.

GET /openapi/v1/app\_scopes

Parameters:

| Name              | Type    | Description                                                   |
|-------------------|---------|---------------------------------------------------------------|
| vrf_id            | integer | Match app scopes by vrf_id.                                   |
| root_app_scope_id | string  | Match app scopes by root app scope id.                        |
| exact_name        | string  | Returns scope matching the exact name, case-sensitive.        |
| exact_short_name  | string  | Returns scopes matching the exact short_name, case-sensitive. |

Returns a list of scope objects.

## Create a scope

This endpoint is used to create new scopes.

POST /openapi/v1/app\_scopes

Parameters:

| Name                | Type    | Description                                                                                                                                |
|---------------------|---------|--------------------------------------------------------------------------------------------------------------------------------------------|
| short_name          | string  | User specified name of the scope.                                                                                                          |
| description         | string  | User specified description of the scope.                                                                                                   |
| short_query         | JSON    | Filter (or match criteria) associated with the scope.                                                                                      |
| parent_app_scope_id | string  | ID of the parent scope.                                                                                                                    |
| policy_priority     | integer | Default is 'last'. Used to sort workspace priorities. See Policy Ordering under <a href="#">Review Automatically Discovered Policies</a> . |

### Sample python code

```
req_payload = {
 "short_name": "App Scope Name",
 "short_query": {
 "type": "eq",
```

```

 "field": "ip",
 "value": <....>
 },
 "parent_app_scope_id": <parent_app_scope_id>
}
resp = restclient.post('/app_scopes', json_body=json.dumps(req_payload))

```

To create a scope based on subnet, use the following `short_query`:

```

"short_query":
{
 "type": "subnet",
 "field": "ip",
 "value": "1.0.0.0/8"
},

```

## Get specific scope

This endpoint returns an instance of a scope.

```
GET /openapi/v1/app_scopes/{app_scope_id}
```

Returns the scope object associated with the specified ID.

## Update a scope

This endpoint updates a scope. Changes to the `name` and `description` are applied immediately. Changes to the `short_query` mark the scope as ‘dirty’ and set the `dirty_short_query` attribute. Once all scope query changes, under a given root scope, are made, one needs to ping the [Commit scope query changes](#) endpoint to commit all the required updates.

```
PUT /openapi/v1/app_scopes/{app_scope_id}
```

Parameters:

| Name        | Type   | Description                                           |
|-------------|--------|-------------------------------------------------------|
| short_name  | string | User specified name of the scope.                     |
| description | string | User specified description of the scope.              |
| short_query | JSON   | Filter (or match criteria) associated with the scope. |

Returns the modified scope object associated with specified ID.

## Delete a specific scope

This endpoint deletes the specified scope.

```
DELETE /openapi/v1/app_scopes/{app_scope_id}
```

If the Scope is associated with a workspace, Policy, User Inventory Filter, etc. this endpoint will return `422 Unprocessable Entity`. The returned Error object will contain a `details` attribute with the count of de-



pendent objects along with the ids of the first 10 of each type. This information can be used to locate and remove the blocking dependencies.

## Get scopes in policy priority order

This endpoint lists the scopes in the order that their corresponding primary workspace will be enforced.

```
GET /openapi/v1/app_scopes/{root_app_scope_id}/policy_order
```

Returns an array of scope objects.

## Update the policy order

This endpoint will update the order at which policies are applied.



**Warning** This endpoint changes the order at which policies are applied. As a result new host firewall rules will be inserted and any existing rules will be deleted on the relevant hosts.

```
POST /openapi/v1/app_scopes/{root_app_scope_id}/policy_order
```

Parameters:

| Name              | Type   | Description                                                     |
|-------------------|--------|-----------------------------------------------------------------|
| root_app_scope_id | string | Root scope or which the order is being changed.                 |
| ids               | array  | array of scope id strings in the order they should be enforced. |

The `ids` array parameter must include all members of the root scope, including the root.

## Commit scope query changes

This endpoint triggers an asynchronous background job to update all ‘dirty’ children under a given root scope. This job updates scopes and workspaces, see [Scopes](#) for more details.

```
POST /openapi/v1/app_scopes/commit_dirty
```

Parameters:

| Name              | Type    | Description                                                 |
|-------------------|---------|-------------------------------------------------------------|
| root_app_scope_id | string  | ID for a root scope for which all children will be updated. |
| sync              | boolean | (optional) Indicate if the request should be synchronous.   |

Returns 202 to indicate the job has been enqueued. To check if the job has completed, poll the root scope’s ‘dirty’ attribute to see if it has been set to false.

Users may pass the `sync` parameter to have the job run immediately. The request will return when done with a 200 status code. This request may take some time if many updates need to be applied.

## Submit a group suggestion request

Submit a group suggestion request for a scope.

PUT /openapi/v1/app\_scopes/{app\_scope\_id}/suggest\_groups

Parameters: The request URL contains the following parameters

| Name         | Type   | Description                          |
|--------------|--------|--------------------------------------|
| app_scope_id | string | The unique identifier for the scope. |

Parameters: The JSON query body contains the following keys

| Name       | Type   | Description                                             |
|------------|--------|---------------------------------------------------------|
| start_time | string | Start time of the group suggestion input time interval. |
| end_time   | string | End time of the group suggestion input time interval.   |

Response object: Returns an object with the following attributes:

| Name    | Type   | Description                                                                  |
|---------|--------|------------------------------------------------------------------------------|
| message | string | Message regarding success/failure in submission of group suggestion request. |

### Sample python code

```
app_scope_id = '5d02b493755f0237a3d6e078'
req_payload = {
 'start_time': '2020-09-17T10:00:00-0700',
 'end_time': '2020-09-17T11:00:00-0700',
}
resp = restclient.put('/app_scopes/%s/suggest_groups' % app_scope_id,
 json_body=json.dumps(req_payload))
```

## Get group suggestion status

Query group suggestion status of the scope.

GET /openapi/v1/app\_scopes/{app\_scope\_id}/suggest\_groups\_status

Parameters: The request URL contains the following parameters

| Name         | Type   | Description                          |
|--------------|--------|--------------------------------------|
| app_scope_id | string | The unique identifier for the scope. |

Response object: Returns an object with the following attributes:

| Name | Type | Description |
|------|------|-------------|
|------|------|-------------|

|        |        |                                                                      |
|--------|--------|----------------------------------------------------------------------|
| status | string | Status of the group suggestion. Values: PENDING, COMPLETE, or FAILED |
|--------|--------|----------------------------------------------------------------------|

### Sample python code

```
app_scope_id = '5d02b493755f0237a3d6e078'
resp = restclient.get('/app_scopes/%s/suggest_groups_status' % app_scope_id)
```

## Configure Alerts

This set of APIs can be used to manage user alerts. They require the `user_alert_management` capability associated with the API key.

- [Alert Object, on page 801](#)
- [Get Alerts, on page 802](#)
- [Create an Alert, on page 802](#)
- [Get Specific Alert, on page 803](#)
- [Update an Alert, on page 803](#)
- [Delete Specific Alert, on page 804](#)

## Alert Object

Each alert configuration object contains the following fields:

| Attribute          | Type    | Description                                                                               |
|--------------------|---------|-------------------------------------------------------------------------------------------|
| app_name           | string  | Application name associated with the alert configuration.                                 |
| rules              | object  | Set of conditions that must be met for the alert configuration to trigger an alert.       |
| subjects           | object  | List of users who should receive the alert.                                               |
| severity           | string  | Indicates the level of severity associated with an alert configuration.                   |
| individual_alert   | boolean | Indicates whether individual alerts should be sent that triggers the alert configuration. |
| summary_alert_freq | string  | Frequency of summary alerts to be sent for a particular alert configuration.              |

| Attribute       | Type   | Description                                                                        |
|-----------------|--------|------------------------------------------------------------------------------------|
| alert_type      | string | Unique identifier of the alert configuration.                                      |
| app_instance_id | string | Unique identifier of a particular instance associated with the alert configuration |

## Get Alerts

This endpoint retrieves the list of alert configurations for a user. Alerts can be filtered to a given root scope. If no scope is provided, all alerts, for all scopes the user has access to, are returned. Service provider alerts will only be returned if the user is a site admin.

GET /openapi/v1/alert\_confs

Parameters:

| Name              | Type   | Description                                                                 |
|-------------------|--------|-----------------------------------------------------------------------------|
| root_app_scope_id | string | (optional) ID of a root scope to return alerts only assigned to that scope. |

Response object: Returns a list of user alert objects.

## Create an Alert

This endpoint is used to create a new alert.

POST openapi/v1/alert\_confs

Parameters:

| Attribute | Type   | Description                                                                         |
|-----------|--------|-------------------------------------------------------------------------------------|
| app_name  | string | Application name associated with the alert configuration.                           |
| rules     | object | Set of conditions that must be met for the alert configuration to trigger an alert. |
| subjects  | object | List of users who should receive the alert.                                         |
| severity  | string | Indicates the level of severity associated with an alert configuration.             |

| Attribute          | Type    | Description                                                                               |
|--------------------|---------|-------------------------------------------------------------------------------------------|
| individual_alert   | boolean | Indicates whether individual alerts should be sent that triggers the alert configuration. |
| summary_alert_freq | string  | Frequency of summary alerts to be sent for a particular alert configuration.              |
| alert_type         | string  | Unique identifier of the alert configuration.                                             |
| app_instance_id    | string  | Unique identifier of a particular instance associated with the alert configuration.       |

The requesting user must have access to the provided scope. An alert without a scope is a 'Service Provider Alert' and only a site admin may create them.

Response object: Returns the newly created alert object.

## Get Specific Alert

This endpoint returns a specific alert object.

GET /openapi/v1/alert\_confs/

Parameters: The request URL contains the following parameters

| Name     | Type   | Description                    |
|----------|--------|--------------------------------|
| alert_id | string | Uniquely identifies the alert. |

Response object: Returns an alert object associated with the specified ID.

## Update an Alert

This endpoint is used to update an existing alert.

PUT /openapi/v1/alert\_confs/

Parameters: The request URL contains the following parameters

| Name     | Type   | Description                                                     |
|----------|--------|-----------------------------------------------------------------|
| alert_id | string | To retrieve or modify the configuration settings for the alert. |

The JSON request body contains the following parameters

| Name | Type   | Description                        |
|------|--------|------------------------------------|
| name | string | User specified name for the alert. |

| Name        | Type   | Description                               |
|-------------|--------|-------------------------------------------|
| description | string | User specified description for the alert. |

The requesting user must have access to the provided scope. A alert without a scope is called a ‘Service Provider Alert’ and only site admin may update them.

Response object: The updated alert object with the specified ID.

## Delete Specific Alert

This endpoint deletes the specified alert.

```
DELETE /openapi/v1/alert_confs/{alert_id}
```

Parameters: The request URL contains the following parameters

| Name     | Type   | Description                    |
|----------|--------|--------------------------------|
| alert_id | string | Uniquely identifies the alert. |

Response object: None.

## Roles

This set of APIs can be used to manage user roles. They require the `user_role_scope_management` capability associated with the API key.




---

**Note** These APIs are only available to site admins and owners of root scopes.

---

## Role object

The role object attributes are described below:

| Attribute    | Type   | Description                                                                    |
|--------------|--------|--------------------------------------------------------------------------------|
| id           | string | Unique identifier for the role.                                                |
| app_scope_id | string | Scope to which the scope is defined, maybe empty for “Service Provider Roles”. |
| name         | string | User specified name for the role.                                              |
| description  | string | User specified description for the role.                                       |

## Get roles

This endpoint returns a list of roles accessible to the user. Roles can be filtered to a given root scope. If no scope is provided, all roles, for all scopes the user has access to, are returned. Service provider roles will only be returned if the user is a site admin.

GET /openapi/v1/roles

Parameters:

| Name         | Type   | Description                                                                |
|--------------|--------|----------------------------------------------------------------------------|
| app_scope_id | string | (optional) ID of a root scope to return roles only assigned to that scope. |

Response object: Returns a list of user role objects.

### Sample python code

```
resp = restclient.get('/roles')
```

## Create a role

This endpoint is used to create a new role.

POST /openapi/v1/roles

Parameters:

| Name         | Type   | Description                                                                                                                        |
|--------------|--------|------------------------------------------------------------------------------------------------------------------------------------|
| name         | string | User specified name for the role.                                                                                                  |
| description  | string | User specified description for the role.                                                                                           |
| app_scope_id | string | (optional) The scope ID under which the role is created. If no scope ID mentioned the role is considered as service provider role. |

The requesting user must have access to the provided scope. A role without a scope is called a 'Service Provider Role' and only site admin may create them.

Response object: Returns the newly created role object.

### Sample python code

```
app_scope_id = '<app-scope-id>'
req_payload = {
 'name': 'Role Name',
 'description': 'Role Description',
 'app_scope_id': app_scope_id
}
restclient.post('/roles', json_body=json.dumps(req_payload))
```

## Get specific role

This endpoint returns a specific role object.

```
GET /openapi/v1/roles/{role_id}
```

Parameters: The request URL contains the following parameters

| Name    | Type   | Description                   |
|---------|--------|-------------------------------|
| role_id | string | Uniquely identifies the role. |

Response object: Returns a role object associated with the specified ID.

### Sample python code

```
role_id = '<role-id>'
restclient.get('/roles/%s' % role_id)
```

## Update a role

This endpoint is used to update an existing role.

```
PUT /openapi/v1/roles/{role_id}
```

Parameters: The request URL contains the following parameters

| Name    | Type   | Description                   |
|---------|--------|-------------------------------|
| role_id | string | Uniquely identifies the role. |

The JSON request body contains the following parameters

| Name        | Type   | Description                              |
|-------------|--------|------------------------------------------|
| name        | string | User specified name for the role.        |
| description | string | User specified description for the role. |

The requesting user must have access to the provided scope. A role without a scope is called a 'Service Provider Role' and only site admin may update them.

Response object: The updated role object with the specified ID.

### Sample python code

```
role_id = '<role-id>'
req_payload = {
 'name': 'Role Name',
 'description': 'Role Description',
}
restclient.put('/roles/%s' % role_id, json_body=json.dumps(req_payload))
```

## Give a role access to scope

This endpoint gives a role the specified access level to a scope.



POST /openapi/v1/roles/{role\_id}/capabilities

Capabilities can only be added to the roles that the user has access to. If the roles is assigned to a scope, capabilities must correspond to that scope or its children. Service provider roles (those not assigned to a scope) can add capabilities for any scope.

Parameters: The request URL contains the following parameters

| Name    | Type   | Description                   |
|---------|--------|-------------------------------|
| role_id | string | Uniquely identifies the role. |

The JSON request body contains the following parameters

| Name         | Type   | Description                                                                           |
|--------------|--------|---------------------------------------------------------------------------------------|
| app_scope_id | string | ID of the scope to which access is provided.                                          |
| ability      | string | Possible values are SCOPE_READ, SCOPE_WRITE, EXECUTE, ENFORCE, SCOPE_OWNER, DEVELOPER |

For more description of abilities, refer to [Roles](#).

Response object:

| Name         | Type    | Description                                                                           |
|--------------|---------|---------------------------------------------------------------------------------------|
| app_scope_id | string  | ID of the scope to which access is provided.                                          |
| role_id      | string  | ID of the role.                                                                       |
| ability      | string  | Possible values are SCOPE_READ, SCOPE_WRITE, EXECUTE, ENFORCE, SCOPE_OWNER, DEVELOPER |
| inherited    | boolean |                                                                                       |

### Sample python code

```
role_id = '<role-id>'
req_payload = {
 'app_scope_id': '<app-scope-id>',
 'ability': 'SCOPE_READ'
}
restclient.post('/roles/%s/capabilities' % role_id,
 json_body=json.dumps(req_payload))
```

## Delete specific role

This endpoint deletes the specified role.

```
DELETE /openapi/v1/roles/{role_id}
```

Parameters: The request URL contains the following parameters

| Name    | Type   | Description                   |
|---------|--------|-------------------------------|
| role_id | string | Uniquely identifies the role. |

Response object: None.

### Sample python code

```
role_id = '<role-id>'
restclient.delete('/roles/%s' % role_id)
```

## Users

This set of APIs manages users. They require the `user_role_scope_management` capability associated with the API key.



**Note** These APIs are only available to site admins and owners of root scopes.

## User object

The user object attributes are described below:

| Attribute             | Type    | Description                                                                                    |
|-----------------------|---------|------------------------------------------------------------------------------------------------|
| id                    | string  | Unique identifier for the user role.                                                           |
| email                 | string  | Email associated with user account.                                                            |
| first_name            | string  | First name.                                                                                    |
| last_name             | string  | Last name.                                                                                     |
| app_scope_id          | string  | The scope to which the user is assigned. Maybe empty if the user is a “Service Provider User”. |
| role_ids              | list    | List of IDs of roles assigned to the user account.                                             |
| by-pass_external_auth | boolean | True for local users and false for external auth users (ldap or sso).                          |

| Attribute   | Type    | Description                                                                 |
|-------------|---------|-----------------------------------------------------------------------------|
| disabled_at | integer | Unix timestamp of when the user has been disabled. Zero or null, otherwise. |

## Get users

This endpoint returns a list of user objects known to the Secure Workload appliance.

GET /openapi/v1/users

Parameters: The request URL contains the following parameters

| Name             | Type    | Description                                                  |
|------------------|---------|--------------------------------------------------------------|
| include_disabled | boolean | (optional) To include disabled users, defaults to false.     |
| app_scope_id     | string  | (optional) Return only users assigned to the provided scope. |

Response object: Returns a list of user objects. Only site admins can see ‘Service provider users’, i.e. those not assigned to a scope.

### Sample python code

```
GET /openapi/v1/users
```

## Create a new user account

This endpoint is used to create a new user account.

POST /openapi/v1/users

Parameters: The JSON request body contains the following parameters

| Name         | Type   | Description                                                       |
|--------------|--------|-------------------------------------------------------------------|
| email        | string | Email associated with user account.                               |
| first_name   | string | First name.                                                       |
| last_name    | string | Last name.                                                        |
| app_scope_id | string | (optional) Root scope to which user belongs.                      |
| role_ids     | list   | (optional) The list of roles that should be assigned to the user. |

The `app_scope_id` is the ID of the root scope to which the user is to be assigned. If the `app_scope_id` is not present then the user is a ‘Service Provider user.’ Only site admins can create service provider users. The `role_ids` are the ids of the roles that were created under the specified app scope.

Response object: Returns the newly created user object.

### Sample python code

```
req_payload = {
 "first_name": "fname",
 "last_name": "lname",
 "email": "foo@bar.com"
 "app_scope_id": "root_appscope_id",
 "role_ids": ["roleid1", "roleid2"]
}
resp = restclient.post('/users', json_body=json.dumps(req_payload))
```

## Get specific user

This endpoint returns specific user object.

GET /openapi/v1/users/{user\_id}

Parameters: The request URL contains the following parameters

| Name    | Type   | Description            |
|---------|--------|------------------------|
| user_id | string | ID of the user object. |

Response object: Returns a user object associated with specified ID.

### Sample python code

```
user_id = '5ce480db497d4f1ca1fc2b2b'
resp = restclient.get('/users/%s' % user_id)
```

## Update a user

This endpoint updates an existing user.

PUT /openapi/v1/users/{user\_id}

Parameters: The request URL contains the following parameters

| Name    | Type   | Description                          |
|---------|--------|--------------------------------------|
| user_id | string | ID of the user object being updated. |

The JSON request body contains the following parameters

| Name         | Type   | Description                                      |
|--------------|--------|--------------------------------------------------|
| email        | string | Email associated with user account.              |
| first_name   | string | First name.                                      |
| last_name    | string | Last name.                                       |
| app_scope_id | string | Root App Scope ID (only allowed for site admins) |

Response object: Returns the newly updated user object.

#### Sample python code

```
req_payload = {
 "first_name": "fname",
 "last_name": "lname",
 "email": "foo@bar.com"
 "app_scope_id": "root_appscope_id",
}
restclient.put('/users', json_body=json.dumps(req_payload))
```

## Enable/reactivate a deactivated user

This endpoint is used to re-enable a deactivated user.

POST /openapi/v1/users/{user\_id}/enable

Parameters: The request URL contains the following parameters

| Name    | Type   | Description                          |
|---------|--------|--------------------------------------|
| user_id | string | ID of the user object being enabled. |

Response object: Returns the reactivated user object associated with the specified ID.

#### Sample python code

```
user_id = '5ce480db497d4f1ca1fc2b2b'
resp = restclient.post('/users/%s/enable' % user_id)
```

## Add role to the user account

This endpoint is used to add a role to a user account.

PUT /openapi/v1/users/{user\_id}/add\_role

Parameters: The request URL contains the following parameters

| Name    | Type   | Description                           |
|---------|--------|---------------------------------------|
| user_id | string | ID of the user object being modified. |

The JSON request body contains the following parameters

| Name    | Type   | Description                        |
|---------|--------|------------------------------------|
| role_id | string | ID of the role object to be added. |

Response object: Returns the modified user object associated with the specified ID.

#### Sample python code

```
user_id = '5ce480db497d4f1ca1fc2b2b'
req_payload = {
```

```

 "role_id": "5ce480d4497d4f1c155d0cef",
 }
 resp = restclient.put('/users/%s/add_role' % user_id,
 json_body=json.dumps(req_payload))

```

## Remove role from the user account

This endpoint is used to remove a role from a user account.

DELETE /openapi/v1/users/{user\_id}/remove\_role

Parameters: The request URL contains the following parameters

| Name    | Type   | Description                          |
|---------|--------|--------------------------------------|
| user_id | string | ID of the user object being deleted. |

The JSON request body contains the following parameters

| Name    | Type   | Description                          |
|---------|--------|--------------------------------------|
| role_id | string | ID of the role object to be removed. |

Response object: Returns the modified user object associated with the specified ID.

### Sample python code

```

user_id = '5ce480db497d4f1ca1fc2b2b'
req_payload = {
 "role_id": "5ce480d4497d4f1c155d0cef",
}
resp = restclient.delete('/users/%s/remove_role' % user_id,
 json_body=json.dumps(req_payload))

```

## Delete specific user

This endpoint deletes the specified user account.

DELETE /openapi/v1/users/{user\_id}

Parameters: The request URL contains the following parameters

| Name    | Type   | Description                          |
|---------|--------|--------------------------------------|
| user_id | string | ID of the user object being deleted. |

Response object: Returns the deleted user object associated with the specified ID.

### Sample python code

```

user_id = '5ce480db497d4f1ca1fc2b2b'
resp = restclient.delete('/users/%s' % user_id)

```

# Inventory filters

Inventory filters encode the match criteria for inventory search queries. This set of APIs provide functionality similar to what is described in [Inventory Filters](#) . They require either `sensor_management` or `app_policy_management` capability associated with the API key.

## Inventory Filter Object

The inventory filter JSON object is returned as a single object or an array of objects depending on the API endpoint. The object's attributes are described below:

| Attribute    | Type    | Description                                                                                                                                                                                                                                                       |
|--------------|---------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| id           | string  | Unique identifier for the inventory filter.                                                                                                                                                                                                                       |
| name         | string  | User specified name of the inventory filter.                                                                                                                                                                                                                      |
| app_scope_id | string  | ID of the scope associated with the filter.                                                                                                                                                                                                                       |
| short_query  | JSON    | Filter (or match criteria) associated with the filter.                                                                                                                                                                                                            |
| primary      | boolean | When 'true' the filter is restricted to the ownership scope.                                                                                                                                                                                                      |
| public       | boolean | When 'true' the filter provides a service for its scope. Must also be primary or scope restricted.                                                                                                                                                                |
| query        | JSON    | Filter (or match criteria) associated with the filter in conjunction with the filters of the parent scopes. These conjunctions take effect if 'restricted to ownership scope' checkbox is checked. If 'primary' field is false then query is same as short_query. |

## Get inventory filters

This endpoint returns a list of inventory filters visible to the user.

GET `/openapi/v1/filters/inventories`

Parameters:

| Name              | Type    | Description                                                            |
|-------------------|---------|------------------------------------------------------------------------|
| vrf_id            | integer | Match inventory filters by vrf id.                                     |
| root_app_scope_id | string  | Match inventory filters by root app scope id.                          |
| name              | string  | Returns inventory filters matching part of the name, case-insensitive. |
| exact_name        | string  | Returns inventory filters matching the exact name, case-sensitive.     |

## Create an inventory filter

This endpoint is used to create an inventory filter.

POST /openapi/v1/filters/inventories

Parameters:

| Name         | Type    | Description                                                                                     |
|--------------|---------|-------------------------------------------------------------------------------------------------|
| name         | string  | User specified name of the application scope.                                                   |
| query        | JSON    | Filter (or match criteria) associated with the filter.                                          |
| app_scope_id | string  | ID of the scope associated with the filter.                                                     |
| primary      | boolean | When 'true' the filter is restricted to the ownership scope.                                    |
| public       | boolean | When 'true' the filter provides a service for its scope. Must also be primary/scope restricted. |

### Sample python code

```
req_payload = {
 "app_scope_id": <app_scope_id>,
 "name": "sensor_config_inventory_filter",
 "query": {
 "type": "eq",
 "field": "ip",
 "value": <sensor_interface_ip>
 },
}
resp = restclient.post('/filters/inventories', json_body=json.dumps(req_payload))
```



## Validate an inventory filter query

This endpoint will validate a query's structure against the required schema.

POST /openapi/v1/filters/inventories/validate\_query

Parameters:

| Name  | Type | Description                                           |
|-------|------|-------------------------------------------------------|
| query | JSON | Filter (or match criteria) associated with the scope. |

Response object:

| Attribute | Type    | Description                          |
|-----------|---------|--------------------------------------|
| valid     | boolean | Indicates if the query is valid      |
| errors    | array   | If invalid, details about the errors |

## Get specific inventory filter

This endpoint returns an instance of an inventory filter.

GET /openapi/v1/filters/inventories/{inventory\_filter\_id}

Returns an inventory filter object associated with specified ID.

## Update specific inventory filter

This endpoint is used to update an inventory filter.

PUT /openapi/v1/filters/inventories/{inventory\_filter\_id}

Parameters:

| Name         | Type    | Description                                                                                                                 |
|--------------|---------|-----------------------------------------------------------------------------------------------------------------------------|
| name         | string  | User specified name of the scope.                                                                                           |
| query        | JSON    | Filter (or match criteria) associated with the scope.                                                                       |
| app_scope_id | string  | ID of the scope associated with the filter.                                                                                 |
| primary      | boolean | When 'true' the filter is restricted to the ownership scope.                                                                |
| public       | boolean | When 'true' the filter provides a service. May be used as part of policy generation. Must also be primary/scope restricted. |

| Name   | Type    | Description                                               |
|--------|---------|-----------------------------------------------------------|
| Usages | boolean | Collection of member policy and configuration statistics. |

## Delete a specific inventory filter

This endpoint deletes the specified inventory filter.

```
DELETE /openapi/v1/filters/inventories/{inventory_filter_id}
```

## Flow Search

The flow search feature provides similar functionality as described in [Network Flows - Traffic Visibility](#). These set of APIs require the `flow_inventory_query` capability associated with the API key.

### Query for Flow Dimensions

This endpoint returns the list of flow columns on which search criteria (or *filters*) can be specified for flow search queries (below). For more information on column descriptions, see [Columns and Filters](#).

```
GET /openapi/v1/flowsearch/dimensions
```

Parameters: None

Response object:

| Name       | Type            | Description                                        |
|------------|-----------------|----------------------------------------------------|
| dimensions | List of strings | List of user uploaded and orchestrator dimensions. |

#### Sample python code

```
restclient.get('/flowsearch/dimensions')
```

### Query for Flow Metrics

This endpoint returns the list of metrics, for example, byte count and packet count, associated with flow observations.

```
GET /openapi/v1/flowsearch/metrics
```

Parameters: None

Response object:

| Name    | Type             | Description                |
|---------|------------------|----------------------------|
| metrics | List of strings. | List of available metrics. |

#### Sample python code

```
restclient.get('/flowsearch/metrics')
```

## Query for Flows

This endpoint returns the list of flows matching the filter criteria. Each flow object in the result has attributes that are a union of flow dimensions (returned by the flow dimensions API above) as well as the flow metrics (returned by the flow metrics API above).

```
POST /openapi/v1/flowsearch
```

The list of columns that can be specified in the filter criteria can be obtained by `/openapi/v1/flowsearch/dimensions` API.

Parameters: The query body consists of a JSON body with the following keys.

| Name       | Type              | Description                                                                                                                                                                                                                                                                                                                 |
|------------|-------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| t0         | integer or string | Flow search start time (epoch or ISO 8601)                                                                                                                                                                                                                                                                                  |
| t1         | integer or string | Flow search end time (epoch or ISO 8601)                                                                                                                                                                                                                                                                                    |
| filter     | JSON              | Query filter. If filter is empty (i.e. {}), query matches all flows.                                                                                                                                                                                                                                                        |
| scopeName  | string            | Full name of the scope to which query is restricted.                                                                                                                                                                                                                                                                        |
| dimensions | array             | (Optional) List of dimension names to be returned in the result of flowsearch API. This is an optional parameter. If unspecified, flowsearch results return all the available dimensions. This option is useful to specify a subset of the available dimensions when caller does not care about the rest of the dimensions. |
| metrics    | array             | (Optional) List of metric names to be returned in the result of flowsearch API. This is an optional parameter. If unspecified, flowsearch results return all the available metrics. This option is useful to specify a subset of the available metrics when caller does not care about the rest of the metrics.             |
| limit      | integer           | (Optional) Number of response flows limit.                                                                                                                                                                                                                                                                                  |

| Name       | Type    | Description                                                                                                                                                                         |
|------------|---------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| offset     | string  | (Optional) Offset object received from previous response.                                                                                                                           |
| descending | boolean | (Optional) If this parameter is false or left unspecified, results are in ascending order of timestamps. If parameter value is true, results are in descending order of timestamps. |

The body of the request should be a JSON formatted query. An example of a query body is shown below.

```
{
 "t0": "2016-06-17T09:00:00-0700",
 "t1": "2016-06-17T17:00:00-0700",
 "filter": {
 "type": "and",
 "filters": [
 {
 "type": "contains",
 "field": "dst_hostname",
 "value": "prod"
 },
 {
 "type": "in",
 "field": "dst_port",
 "values": ["80", "443"]
 }
]
 },
 "scopeName": "Default:Production:Web",
 "limit": 100,
 "offset": <offset-object>
}
```

## Filters

The filter supports primitive filters and logical filters (“not”, “and”, “or”) comprised of one or more primitive filters. Format of primitive filter is as follows:

```
{"type": "<OPERATOR>", "field": "<COLUMN_NAME>", "value": "<COLUMN_VALUE>"}
```

For primitive filters, operator can be a comparison operator like `eq`, `ne`, `lt`, `lte`, `gt` or `gte`. Operator could also be `in`, `regex`, `subnet`, `contains` or `range`.

Some examples of primitive filters might include:

```
{"type": "eq", "field": "src_address", "value": "7.7.7.7"}
{"type": "regex", "field": "src_hostname", "value": "prod.*"}
{"type": "subnet", "field": "src_addr", "value": "1.1.11.0/24"}

Note, 'in' clause uses 'values' key instead of 'value'
{"type": "in", "field": "src_port", "values": [80, 443]}
```

You can also specify complex filters using boolean operations like not, and or or. Following are some examples of these type of filters:

```
"and" and "or" operators need to specify list of "filters"
{"type": "and",
 "filters": [
 {"type": "in", "field": "src_port", "values": [80, 443]},
 {"type": "regex", "field": "src_hostname", "value": "prod.*"}
]
}

"not" operator needs to specify a "filter"
{"type": "not",
 "filter": {"type": "subnet", "field": "src_addr", "value": "1.1.11.0/24"}
}
```

More formally, schema of `filter` in the flow search request is as follows:

| Keys    | Values                                                                                      |
|---------|---------------------------------------------------------------------------------------------|
| type    | Filter type                                                                                 |
| field   | Filter field column for primitive filters                                                   |
| filter  | Filter object (only used for <code>not</code> filter type)                                  |
| filters | List of filter objects (used for <code>and</code> and <code>or</code> filter types)         |
| value   | Value for primitive filters                                                                 |
| values  | List of values for primitive filters with filter type <code>in</code> or <code>range</code> |

## Primitive Filter Types

**eq, ne**—Searches flows for equality or inequality respectively in column specified by "field" with value specified by "value". Supports the following fields: `src_hostname`, `dst_hostname`, `src_address`, `dst_address`, `src_port`, `dst_port`, `src_scope_name`, `dst_scope_name`, `vrf_name`, `src_enforcement_epg_name`, `dst_enforcement_epg_name`, `proto`. These operators also work on user labelled columns.

**lt, lte, gt, gte**—Searches flows where values of column specified by "field" are less than, less than equal to, greater than or greater than equal to (as applicable) the value specified by "value". Supports the following fields: [`src_port`, `dst_port`].

**range**—Searches flows for values of column specified by "field" between range start and range end specified by "values" list (this list must be of size 2 for "range" filter type – first value is the range start and second is the range end). Supports the following fields: [`src_port`, `dst_port`].

**in**—Searches flows for membership in column specified by "field" with membership list specified by "values". Supports the following fields: `src_hostname`, `dst_hostname`, `src_address`, `dst_address`, `src_port`, `dst_port`, `src_scope_name`, `dst_scope_name`, `vrf_name`, `src_enforcement_epg_name`, `dst_enforcement_epg_name`, `proto`. This operator also works on user labelled columns.

**regex, contains**—Searches flows for regex matches or containment matches respectively in column specified by "field" with regex specified by "value". Supports the following fields: `src_hostname`, `dst_hostname`, `src_scope_name`, `dst_scope_name`, `vrf_name`, `src_enforcement_epg_name`, `dst_enforcement_epg_name`.

These operators also work on user labelled columns. Filters with `regex` type must use Java style regex patterns as `"value"`.

**subnet**—Searches flows for subnet membership specified by `"field"` as a string in CIDR notation. Supports the following fields: `["src_address", "dst_address"]`

## Logical Filter Types

- **not**—Logical "not" filter of object specified by `"filter"`.
- **and**—Logical "and" filter of list of filter objects specified by `"filters"`.
- **or**—Logical "or" filter of list of filter objects specified by `"filters"`.

Response object:

| Keys    | Values                                                    |
|---------|-----------------------------------------------------------|
| offset  | Response offset to be passed for the next page of results |
| results | List of results                                           |

To generate the next page of results, take the object received by the response in `offset` and pass it as the value for the `offset` of the next query.

### Sample python code

```
req_payload = {"t0": "2016-11-07T09:00:00-0700",
 "t1": "2016-11-07T19:00:00-0700",
 "scopeName": "Default:Prod:Web",
 "limit": 10,
 "filter": {"type": "and",
 "filters": [
 {"type": "subnet", "field": "src_address", "value": "1.1.11.0/24"},
 {"type": "regex", "field": "src_hostname", "value": "web*"}
]
 }
 }

resp = restclient.post('/flowsearch', json_body=json.dumps(req_payload))
print resp.status_code
if resp.status_code == 200:
 parsed_resp = json.loads(resp.content)
 print json.dumps(parsed_resp, indent=4, sort_keys=True)
```

## TopN Query for Flows

This endpoint returns a top N sorted list of values of specified dimension where rank in the list is determined by the aggregate of specified metric.

POST `/openapi/v1/flowsearch/topn`

Parameters:

The list of columns that can be specified in the filter criteria can be obtained by `/openapi/v1/flowsearch/dimensions` API. The body of the request should be a JSON formatted query. An

example of a query body is shown below. Parameters `t0` and `t1` in the request body can be in epoch format or in ISO 8601 format. TopN API only allows querying maximum time range of one day. The dimension on which the grouping has to be done should be specified through `dimension`. The metric by which top N results need to ranked should be specified in `metric` field in the JSON body. You should specify a `threshold` with a minimum value of 1 which signifies the ‘N’ in ‘TopN’. The maximum value of this `threshold` is 1000. Even if the user specify more than 1000 the API returns only a maximum of 1000 results. In addition, you must specify a parameter called `scopeName` which is the full name of the scope to which you want to restrict the search. The `filter` is same as that of filter of Flow Search [Filters, on page 818](#) . If the `filter` is not mentioned, the topN is applied on all the flow entries.

```
{
 "t0": "2016-06-17T09:00:00-0700", # t0 can also be 1466179200
 "t1": "2016-06-17T17:00:00-0700", # t1 can also be 1466208000
 "dimension": "src_address",
 "metric": "fwd_pkts",
 "filter": {"type": "eq", "field": "src_address", "value": "172.29.203.193"}, #optional
 "threshold": 5,
 "scopeName": "Default"
}
```

The query body consists of a JSON body with the following keys.

| Keys      | Values                                                                                                                    |
|-----------|---------------------------------------------------------------------------------------------------------------------------|
| t0        | Start time of the Flow (epoch or ISO 8601)                                                                                |
| t1        | End time of the Flow (epoch or ISO 8601)                                                                                  |
| filter    | Query filter. If filter is empty (i.e. {}), or filter is absent (optional) then topN query is applied on all flow entries |
| scopeName | Full name of the scope to which query is restricted to                                                                    |
| dimension | The dimension is a field on which we are grouping.                                                                        |
| metric    | The metric is the total count of values of the dimension.                                                                 |
| threshold | Threshold is N in the topN.                                                                                               |

Response object:

| Keys   | Values                     |
|--------|----------------------------|
| result | Array of the top N entries |

### Sample python code

```
req_payload = {
 "t0": "2017-06-07T08:20:00-07:00",
 "t1": "2017-06-07T14:20:00-07:00",
 "dimension": "src_address",
 "metric": "fwd_pkts",
```

```

 "filter": {"type": "ne", "field": "src_address", "value": "172.29.203.193"},
 "threshold": 5,
 "scopeName": "Default"
 }
 resp = rc.post('/flowsearch/topn',
 json_body=json.dumps(req_payload))
 print resp.status_code
 if resp.status_code == 200:
 parsed_resp = json.loads(resp.content)
 print json.dumps(parsed_resp)

```

### Sample response

```

[
 { "result": [
 {"src_address": "172.31.239.163", "fwd_pkts": 23104},
 {"src_address": "172.31.239.162", "fwd_pkts": 22410},
 {"src_address": "172.31.239.166", "fwd_pkts": 16185},
 {"src_address": "172.31.239.168", "fwd_pkts": 15197},
 {"src_address": "172.31.239.169", "fwd_pkts": 15116}
]
}
]

```

## Flow Count

This endpoint returns the number of flow observations matching the specified criteria.

POST /openapi/v1/flowsearch/count

### Parameters:

The body of the request should be a JSON formatted query. An example of a query body is shown below. Parameters `t0` and `t1` in the request body can be in epoch format or in ISO 8601 format. This API only allows querying maximum time range of one day. In addition, you need to specify the `scopeName` parameter which is the full name of the scope to which you want to restrict the search. If this parameter is not specified, flow observation count API request applies to all scopes to which you have read access. The `filter` is same as that of filter of Flow Search [Filters](#).

```

{
 "t0": "2016-06-17T09:00:00-0700", # t0 can also be 1466179200
 "t1": "2016-06-17T17:00:00-0700", # t1 can also be 1466208000
 "filter": {"type": "eq", "field": "src_address", "value": "172.29.203.193"},
 "scopeName": "Default"
}

```

The query body consists of a JSON body with the following keys.

| Keys      | Values                                                                   |
|-----------|--------------------------------------------------------------------------|
| t0        | Start time of the flow (epoch or ISO 8601)                               |
| t1        | End time of the flow (epoch or ISO 8601)                                 |
| filter    | Query filter. If filter is empty (i.e. {}) then query matches all flows. |
| scopeName | Full name of the scope to which query is restricted to                   |



Response object:

| Keys  | Values                                                             |
|-------|--------------------------------------------------------------------|
| count | The number of flow observations matching the flow search criteria. |

#### Sample python code

```
req_payload = {
 "t0": "2017-07-20T08:20:00-07:00",
 "t1": "2017-07-20T10:20:00-07:00",
 "scopeName": "Tetration",
 "filter": {
 "type": "eq",
 "field": "dst_port",
 "value": "5642"
 }
}
resp = rc.post('/flowsearch/count',
 json_body=json.dumps(req_payload))
print resp.status_code
if resp.status_code == 200:
 parsed_resp = json.loads(resp.content)
 print json.dumps(parsed_resp)
```

#### Sample response

```
{"count":508767}
```

## Inventory

The inventory search APIs provide similar functionality as described in inventory search. These set of APIs require the `flow_inventory_query` capability associated with the API key.

### Query for inventory dimensions

This endpoint returns the list of inventory columns on which search criteria (or *filters*) can be specified for inventory search queries.

```
GET /openapi/v1/inventory/search/dimensions
```

### Inventory search

This endpoint returns the list of inventory items matching the specified criteria.

```
POST /openapi/v1/inventory/search
```

The list of columns that can be specified in the filter criteria can be obtained with the `/openapi/v1/inventory/search/dimensions` API.

Parameters:

| Name   | Type | Description     |
|--------|------|-----------------|
| filter | JSON | A filter query. |

| Name      | Type    | Description                                                       |
|-----------|---------|-------------------------------------------------------------------|
| scopeName | string  | (optional) Name of the scope by which to limit results.           |
| limit     | integer | (optional) Max number of results to return.                       |
| offset    | integer | (optional) Offset from the previous request to get the next page. |

The body of the request must be a JSON formatted query. An example of a query body is shown below.

```
{
 "filter": {
 "type": "contains",
 "field": "hostname",
 "value": "collector"
 },
 "scopeName": "Default:Production:Web", // optional
 "limit": 100,
 "offset": "<offset-object>" // optional
}
```

To get the different types of filters supported refer to [Filters, on page 818](#)

The query body consists of a JSON body with the following keys.

| Keys              | Values                                                                                                                                                                                                                                                                                                      |
|-------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>filter</b>     | Query filter. If filter is empty (i.e. {}), then query matches all inventory items.                                                                                                                                                                                                                         |
| <b>scopeName</b>  | Full name of the scope to which query is restricted to (optional)                                                                                                                                                                                                                                           |
| <b>dimensions</b> | List of dimension names to be returned in the result of inventory search API. This is an optional parameter. If unspecified, results return all the available dimensions. This option is useful to specify a subset of the available dimensions when caller does not care about the rest of the dimensions. |
| <b>limit</b>      | Number of response items limit (optional)                                                                                                                                                                                                                                                                   |
| <b>offset</b>     | Offset object received from previous response (optional)                                                                                                                                                                                                                                                    |

### Response

The response is a JSON object in the body with the following properties.

| Name   | Type    | Description                                                |
|--------|---------|------------------------------------------------------------|
| offset | integer | Response offset to be passed for the next page of results. |

| Name    | Type             | Description      |
|---------|------------------|------------------|
| results | array of objects | List of results. |

The response may contain an `offset` field for paginated responses. Users will need to specify the same offset in the subsequent request to get the next set of results.

### Sample Python code

```
req_payload = {
 "scopeName": "Tetration", # optional
 "limit": 2,
 "filter": {"type": "and",
 "filters": [
 {"type": "eq", "field": "vrf_name", "value": "Tetration"},
 {"type": "subnet", "field": "ip", "value": "1.1.1.0/24"},
 {"type": "contains", "field": "hostname", "value": "collector"}
]
 }
}

resp = restclient.post('/inventory/search', json_body=json.dumps(req_payload))
print resp.status_code
if resp.status_code == 200:
 parsed_resp = json.loads(resp.content)
 print json.dumps(parsed_resp, indent=4, sort_keys=True)
```

## Inventory Statistics

This endpoint returns statistics for inventory items.

```
GET /openapi/v1/inventory/{id}/stats?t0=<t0>&t1=<t1>&td=<td>
```

**Table 52:**

| Path Parameter | Description                                            |
|----------------|--------------------------------------------------------|
| id             | Inventory item id as {ip}-{vrf_id} such as 1.1.1.1-123 |

| Query Parameter | Description                                                                                                                                |
|-----------------|--------------------------------------------------------------------------------------------------------------------------------------------|
| t0              | Start time for statistics in epoch time                                                                                                    |
| t1              | End time for statistics in epoch time                                                                                                      |
| td              | Granularity for statistic aggregations. An integer specifies number of seconds. Strings may be passed such as “minute”, “hour”, and “day”. |

### Sample Python code

```
resp = restclient.get('/inventory/1.1.1.1-123/stats?t0=1483228800&t1=1485907200&td=day')
```

## Inventory count

This endpoint returns the count of inventory items matching the specified criteria.

POST /openapi/v1/inventory/count

The list of columns that can be specified in the filter criteria can be obtained with the /openapi/v1/inventory/search/dimensions API.

Parameters:

| Name      | Type   | Description                                             |
|-----------|--------|---------------------------------------------------------|
| filter    | JSON   | A filter query.                                         |
| scopeName | string | (optional) Name of the scope by which to limit results. |

The body of the request must be a JSON formatted query. An example of a query body is shown below.

```
{
 "filter": {
 "type": "and",
 "filters": [
 {
 "type": "contains",
 "field": "hostname",
 "value": "prod"
 },
 {
 "type": "subnet",
 "field": "ip",
 "value": "6.6.6.0/24"
 }
]
 },
 "scopeName": "Default:Production:Web", # optional
}
```

### Response

The response is a JSON object in the body with the following properties.

**Table 53:**

| Keys  | Values                                                 |
|-------|--------------------------------------------------------|
| count | Number of inventory items matching the filter Criteria |

### Sample python code

```
req_payload = {
 "scopeName": "Tetration", # optional
 "filter": {"type": "and",
 "filters": [
 {"type": "eq", "field": "vrf_name", "value": "Tetration"},
 {"type": "subnet", "field": "ip", "value": "1.1.1.0/24"},
 {"type": "contains", "field": "hostname", "value": "collector"}
]
 }
}
```

```

 }
}

resp = restclient.post('/inventory/count', json_body=json.dumps(req_payload))
print resp.status_code
if resp.status_code == 200:
 parsed_resp = json.loads(resp.content)
 print json.dumps(parsed_resp, indent=4, sort_keys=True)

```

## Inventory vulnerability

This endpoint returns CVEs corresponding to IP addresses associate with vulnerable workloads.

This API is only available to users with a minimum read access to root scope.

POST /openapi/v1/inventory/cves/{rootScopeID}

Parameters:

| Name | Type            | Description                           |
|------|-----------------|---------------------------------------|
| ips  | list of strings | List of IPs to fetch CVE information. |

The body of the request must be a JSON formatted query. An example of a query body is shown below.

```

{
 "ips": [
 "10.18.187.72",
 "10.18.187.73"
]
}

```

### Response

The response is an array of JSON objects in the body with the following properties.

| Name    | Type            | Description                                           |
|---------|-----------------|-------------------------------------------------------|
| ip      | string          | IP address                                            |
| cve_ids | list of strings | List of CVE IDs on the inventory with the ip address. |

### Sample Python code

```

root_scope_id = "5fa0d242497d4f7d968c669b"
req_payload = {
 "ips": ["10.18.187.72", "10.18.187.73"]
}

resp = restclient.post('/inventory/cves/' + root_scope_id,
json_body=json.dumps(req_payload))
print resp.status_code
if resp.status_code == 200:
 parsed_resp = json.loads(resp.content)
 print json.dumps(parsed_resp, indent=4, sort_keys=True)

```

# Workload

The workload APIs provides programmatic access to the contents of the [Workload Profile](#) page. This set of APIs requires `sensor_management` or `flow_inventory_query` capability associated with the API key.

## Workload details

This endpoint returns the specific workload given agent UUID.

```
GET /openapi/v1/workload/{uuid}
```

| Path Parameter | Description |
|----------------|-------------|
| uuid           | Agent UUID  |

### Response

The response is a workload object associated with the specified UUID. The workload object's attributes schema is described below:

**Table 54:**

| Attribute                | Type    | Description                                                                |
|--------------------------|---------|----------------------------------------------------------------------------|
| agent_type               | integer | Agent type in enum                                                         |
| agent_type_str           | string  | Agent type in plain text                                                   |
| auto_upgrade_opt_out     | boolean | If true, agents do not get automatically upgraded on cluster upgrade       |
| cpu_quota_mode           | integer | CPU quota control                                                          |
| cpu_quota_us             | integer | CPU quota usage                                                            |
| current_sw_version       | string  | Version of agent software running on the workload                          |
| data_plane_disabled      | boolean | If true, flow telemetry data is not exported from the agent to the cluster |
| desired_sw_version       | string  | Version of agent software intended to be running on the workload           |
| enable_cache_sidechannel | boolean | If true, side channel attack detection is enabled                          |
| enable_forensics         | boolean | If true, forensics is enabled                                              |
| enable_meltdown          | boolean | If true, meltdown exploit detection is enabled                             |

| Attribute                 | Type    | Description                                                                                         |
|---------------------------|---------|-----------------------------------------------------------------------------------------------------|
| enable_pid_lookup         | boolean | If true, process lookup is enabled                                                                  |
| forensics_cpu_quota_mode  | integer | Forensics CPU quota control                                                                         |
| forensics_cpu_quota_us    | integer | Forensics quota usage                                                                               |
| forensics_mem_quota_bytes | integer | Forensics memory quota in bytes                                                                     |
| host_name                 | string  | Host name on the workload                                                                           |
| interfaces                | array   | Array of <a href="#">Interface</a> objects                                                          |
| kernel_version            | string  | Kernel version                                                                                      |
| last_config_fetch_at      | integer | Last config fetched at                                                                              |
| last_software_update_at   | integer | Last software is the timestamp at which agent reported its current version                          |
| max_rss_limit             | integer | Max memory limit                                                                                    |
| platform                  | string  | Platform of the workload                                                                            |
| uuid                      | string  | Unique ID of the agent                                                                              |
| windows_enforcement_mode  | string  | Type of Windows enforcement mode, WAF(Windows Advanced Firewall) or WFP(Windows Filtering Platform) |

#### Sample Python code

```
agent_uuid = 'aa28b304f5c79b2f22d87a5af936f4a8fa555894'
resp = restclient.get('/workload/%s' % (agent_uuid))
```

## Workload Statistics

This endpoint returns statistics for a workload.

```
GET /openapi/v1/workload/{uuid}/stats?t0=<t0>&t1=<t1>&td=<td>
```

| Path Parameter | Description |
|----------------|-------------|
| uuid           | Agent UUID  |

The query URL contains the following parameters

| Query Parameter | Description                             |
|-----------------|-----------------------------------------|
| t0              | Start time for statistics in epoch time |

| Query Parameter | Description                                                                                                                                |
|-----------------|--------------------------------------------------------------------------------------------------------------------------------------------|
| t1              | End time for statistics in epoch time. The end time cannot exceed the start time by more . than a day.                                     |
| td              | Granularity for statistic aggregations. An integer specifies number of seconds. Strings may be passed such as “minute”, “hour”, and “day”. |

### Response

The response is a JSON object in the body with the following properties.

| Name      | Type   | Description                                             |
|-----------|--------|---------------------------------------------------------|
| timestamp | string | Time at which metrics were gathered (epoch or ISO 8601) |
| results   | object | Metrics                                                 |

Metrics is a JSON object with the following properties

| Name            | Type    | Description                    |
|-----------------|---------|--------------------------------|
| flow_count      | integer | Number of flows.               |
| rx_byte_count   | integer | Number of received bytes.      |
| rx_packet_count | integer | Number of received packets.    |
| tx_byte_count   | integer | Number of transmitted bytes.   |
| tx_packet_count | integer | Number of transmitted packets. |

### Sample Python code

```
agent_uuid = 'aa28b304f5c79b2f22d87a5af936f4a8fa555894'
td = 15 * 60 # 15 minutes
resp = restclient.get('/workload/%s/stats?t0=1483228800&t1=1485907200&td=%d' % (agent_uuid,
td))

This code queries workload statistics for a week
t0 = 1483228800
for _ in range(7):
 t1 = t0 + 24 * 60 * 60
 resp = restclient.get('/workload/%s/stats?t0=%d&t1=%d&td=day' % (agent_uuid, t0, t1))
 t0 = t1
```

## Installed Software Packages

This endpoint returns list of packages installed on the workload.

```
GET /openapi/v1/workload/{uuid}/packages
```



| Path Parameter | Description |
|----------------|-------------|
| uuid           | Agent UUID  |

### Response

The response is an array of package JSON objects. The package object's schema is described below:

| Attribute    | Type   | Description                 |
|--------------|--------|-----------------------------|
| architecture | string | Architecture of the package |
| name         | string | Name of the package         |
| publisher    | string | Publisher of the package    |
| version      | string | Version of the package      |

### Sample Python code

```
agent_uuid = 'aa28b304f5c79b2f22d87a5af936f4a8fa555894'
resp = restclient.get('/workload/%s/packages' % (agent_uuid))
```

## Workload Vulnerabilities

This endpoint returns list of vulnerabilities observed on the workload.

```
GET /openapi/v1/workload/{uuid}/vulnerabilities
```

The vulnerabilities object consists of a JSON body with the following keys.

| Path Parameter | Description |
|----------------|-------------|
| uuid           | Agent UUID  |

### Response

The response is an array of vulnerability JSON objects. The vulnerability object's schema is described below:

| Attribute              | Type   | Description                                   |
|------------------------|--------|-----------------------------------------------|
| cve_id                 | string | Common Vulnerability Exposure ID              |
| package_infos          | array  | Array of <a href="#">Package Info</a> objects |
| v2_score               | float  | CVSS V2 Score                                 |
| v2_access_complexity   | string | CVSS V2 Access Complexity                     |
| v2_access_vector       | string | CVSS V2 Access Vector                         |
| v2_authentication      | string | CVSS V2 Authentication                        |
| v2_availability_impact | string | CVSS V2 Availability Impact                   |

| Attribute                  | Type   | Description                                      |
|----------------------------|--------|--------------------------------------------------|
| v2_confidentiality_impact  | string | CVSS V2 Confidentiality Impact                   |
| v2_integrity_impact        | string | CVSS V2 Intergrity Impact                        |
| v2_severity                | string | CVSS V2 Severity                                 |
| v3_score                   | float  | CVSS V3 Score                                    |
| v3_attack_complexity       | string | CVSS V3 Attack Compleixty                        |
| v3_attack_vector           | string | CVSS V3 Attack Vector                            |
| v3_availability_impact     | string | CVSS V3 Availability Impact                      |
| v3_base_severity           | string | CVSS V3 Base Severity                            |
| v3_confidentiality_impact  | string | CVSS V2 Confidentiality Impact                   |
| v3_integrity_impact        | string | CVSS V3 Intergrity Impact                        |
| v3_privileges_required     | string | CVSS V3 Privileges Required                      |
| v3_scope                   | string | CVSS V3 Scope                                    |
| v3_user_interaction        | string | CVSS V3 User Interaction                         |
| cvm_score                  | float  | Cisco Security Risk Score                        |
| cvm_severity               | string | Cisco Security Risk Score Severity               |
| cvm_easily_exploitable     | bool   | Cisco Security Risk Score Easily Exploitable     |
| cvm_malware_exploitable    | bool   | Cisco Security Risk Score Malware Exploitable    |
| cvm_active_internet_breach | bool   | Cisco Security Risk Score Active Internet Breach |
| cvm_popular_target         | bool   | Cisco Security Risk Score Popular Target         |
| cvm_predicted_exploitable  | bool   | Cisco Security Risk Score Predicted Exploitable  |
| cvm_fix_available          | bool   | Cisco Security Risk Score Fix Available          |

### Sample Python code

```
agent_uuid = 'aa28b304f5c79b2f22d87a5af936f4a8fa555894'
resp = restclient.get('/workload/%s/vulnerabilities' % (agent_uuid))
```

## Workload Long Running Processes

This endpoint returns list of long running processes on the workload. Long running processes are defined as processes that have at least 5 minutes uptime.

GET /openapi/v1/workload/{uuid}/process/list

| Path Parameter | Description |
|----------------|-------------|
| uuid           | Agent UUID  |

### Response

The response is a list of processes JSON objects.

| Attribute        | Type    | Description                                        |
|------------------|---------|----------------------------------------------------|
| cmd              | string  | Command string of the process                      |
| binary_hash      | string  | Sha256 of the process binary in hex                |
| ctime            | long    | ctime of the process binary in us                  |
| mtime            | long    | mtime of the process binary in us                  |
| exec_path        | string  | Process executable path                            |
| exit_usec        | long    | Time when the process exited in us                 |
| num_libs         | integer | Number of libs the process loads                   |
| pid              | integer | Process ID                                         |
| ppid             | integer | Parent process ID                                  |
| pkg_info_name    | string  | Name of the package associated with the process    |
| pkg_info_version | string  | Version of the package associated with the process |
| proc_state       | string  | Process state                                      |
| uptime           | long    | Uptime of the process in us                        |
| username         | string  | Username of the process                            |
| resource_usage   | array   | Array of <a href="#">Resource Usage</a> object     |

### Sample Python code

```
agent_uuid = 'aa28b304f5c79b2f22d87a5af936f4a8fa555894'
resp = restclient.get('/openapi/v1/workload/%s/process/list' % (agent_uuid))
```

## Workload Process Snapshot Summary

This endpoint returns process snapshot summary on this workload. A process snapshot contains all the processes that are captured by the workload at a given time. Currently one copy of the latest process snapshot is retained. The endpoint supports POST method with empty payload to enable easier future expansion.

POST /openapi/v1/workload/{uuid}/process/tree/ids

| Path Parameter | Description |
|----------------|-------------|
| uuid           | Agent UUID  |

### Response

The response is a list of process snapshot summary JSON objects.

| Attribute     | Type    | Description                                    |
|---------------|---------|------------------------------------------------|
| sensor_uuid   | string  | Agent UUID                                     |
| handle        | string  | Handle to the process snapshot to be retrieved |
| process_count | integer | Number of processes in the snapshot            |
| ts_usec       | integer | Timestamp when the snapshot is captured        |

### Sample Python code

```
agent_uuid = 'aa28b304f5c79b2f22d87a5af936f4a8fa555894'
payload = {
}
resp = restclient.post('/openapi/v1/workload/%s/process/tree/ids' %
 agent_uuid, json_body=json.dumps(payload))
```

## Workload Process Snapshot

This endpoint returns process snapshot on this workload. A process snapshot contains all the processes that are captured by the workload at a given time. Currently one copy of the latest process snapshot is retained. This endpoint needs to be used together with the workload process snapshot summary endpoint.

POST /openapi/v1/workload/{uuid}/process/tree/details

| Path Parameter | Description |
|----------------|-------------|
| uuid           | Agent UUID  |

| Payload Field | Type   | Description                                    |
|---------------|--------|------------------------------------------------|
| handle        | string | Handle to the process snapshot to be retrieved |

## Response

The response is a list of processes belonging to the snapshot in JSON.

| Attribute          | Type    | Description                                        |
|--------------------|---------|----------------------------------------------------|
| command_string     | string  | Tokenized command string                           |
| command_string_raw | string  | Raw command string                                 |
| binary_hash        | string  | Sha256 of the process binary in hex                |
| ctime              | long    | ctime of the process binary in us                  |
| mtime              | long    | mtime of the process binary in us                  |
| exec_path          | string  | Process executable path                            |
| process_id         | integer | Process ID                                         |
| parent_process_id  | integer | Parent process ID                                  |
| process_key        | integer | Unique key to the process                          |
| parent_process_key | integer | Unique key to the parent process                   |
| pkg_info_name      | string  | Name of the package associated with the process    |
| pkg_info_version   | string  | Version of the package associated with the process |
| proc_state         | string  | Process state                                      |
| uptime             | long    | Uptime of the process in us                        |
| username           | string  | Username of the process                            |
| cve_ids            | array   | Array of CVEID object                              |

## Sample Python code

```
agent_uuid = 'aa28b304f5c79b2f22d87a5af936f4a8fa555894'
payload = {
}
resp = restclient.post('/openapi/v1/workload/%s/process/tree/ids' %
 agent_uuid, json_body=json.dumps(payload))
handle = json.loads(resp.text)['process_summary'][0]['summary'][0]['handle']
payload = {
 "handle": handle,
}
resp = restclient.post('/openapi/v1/workload/%s/process/tree/details' %
 agent_uuid, json_body=json.dumps(payload))
```

## JSON Object Definitions

### Interface

| Attribute     | Type    | Description                                                 |
|---------------|---------|-------------------------------------------------------------|
| ip            | string  | IP Address of the interface                                 |
| mac           | string  | Mac Address of the interface                                |
| name          | string  | Name of the interface                                       |
| netmask       | string  | Netmask of the interface                                    |
| pcap_opened   | boolean | If false, packet captures are not enabled for the interface |
| tags_scope_id | array   | Scope IDs associated with the interface                     |
| vrf           | string  | VRF Name                                                    |
| vrf_id        | integer | VRF ID                                                      |

### Package Info

| Attribute | Type   | Description     |
|-----------|--------|-----------------|
| name      | string | Package name    |
| version   | string | Package version |

### Resource Usage

| Attribute       | Type    | Description                                         |
|-----------------|---------|-----------------------------------------------------|
| cpu_usage       | float   | CPU usage                                           |
| memory_usage_kb | integer | Memory usage                                        |
| ts_usec         | long    | Timestamp in us when the resource usage is captured |

### CVE ID

| Attribute                        | Type   | Description           |
|----------------------------------|--------|-----------------------|
| cve_id                           | string | cve ID                |
| impact_cvss_v2_access_complexity | string | CVE access complexity |
| impact_cvss_v2_access_vector     | string | CVE access vector     |

## Default Policy Generation Config

This set of APIs is used to read and update the default policy generation config for a root scope.

The APIs require the `app_policy_management` capability associated with the API key.



**Note** These APIs are only available to site admins and owners of root scopes.

- [Policy Generation Config object, on page 837](#)
- [Get the Default Policy Generation Config, on page 838](#)
- [Set the Default Policy Generation Config, on page 838](#)

## Policy Generation Config object

| Attribute                                    | Type    | Description                                                                                                                                                                      |
|----------------------------------------------|---------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>carry_over_policies</code>             | boolean | Any policy marked as approved will be maintained, if possible                                                                                                                    |
| <code>deep_policy_generation</code>          | boolean | creates policies for the whole scope tree under give scope, includes all the members in the give scope                                                                           |
| <code>skip_clustering</code>                 | boolean | set to true to skip clustering, will generate policies with existing approved clusters and scopes                                                                                |
| <code>auto_accept_policy_connectors</code>   | boolean | auto accepts all outgoing policy connectors                                                                                                                                      |
| <code>enable_exclusion_filter</code>         | boolean | apply exclusion filters to input flow data                                                                                                                                       |
| <code>enable_default_exclusion_filter</code> | boolean | apply default exclusion filters to input flow data                                                                                                                               |
| <code>remove_redundant_policies</code>       | boolean | remove redundant policies during deep policy generation                                                                                                                          |
| <code>enable_service_discovery</code>        | boolean | setting false skips policy generation based on ephemeral port range in adm pipeline reported by the sensor, currently used for generating policies for Windows Active Directory. |
| <code>externals</code>                       | array   | ordered list of external dependency objects                                                                                                                                      |
| <code>clustering_granularity</code>          | string  | one of VERY_COARSE, COARSE, MEDIUM, FINE, VERY_FINE                                                                                                                              |

| Attribute           | Type    | Description                                                          |
|---------------------|---------|----------------------------------------------------------------------|
| policy_compression  | string  | one of DISABLED, CONSERVATIVE, MODERATE, AGGRESSIVE, VERY_AGGRESSIVE |
| port_generalization | string  | one of DISABLED, CONSERVATIVE, MODERATE, AGGRESSIVE, VERY_AGGRESSIVE |
| sim_policy          | integer | 1 => flows, 2 => processes, 5 => both                                |

The External Dependency object

| Name        | Type   | Description                                                                                                                |
|-------------|--------|----------------------------------------------------------------------------------------------------------------------------|
| id          | string | id of the filter                                                                                                           |
| filter_type | string | AppScope or UserInventoryFilter                                                                                            |
| include     | array  | object with user_filters boolean to enable and user_filter_list for the ordered list of provided service inventory filters |

## Get the Default Policy Generation Config

This endpoint returns the current default policy generation config. Can return an empty object if none has been created.

```
GET /openapi/v1/app_scopes/default_adm_run_config
```

Parameters:

The request URL contains the following parameters

| Name              | Type   | Description                                                                  |
|-------------------|--------|------------------------------------------------------------------------------|
| root_app_scope_id | string | The unique identifier of the root scope to which this default config applies |

Response object: Returns the current default policy generation config or an empty object if none has been created

## Set the Default Policy Generation Config

This endpoint sets the default policy generation config.

```
PUT /openapi/v1/app_scopes/default_adm_run_config
```

Parameters in addition to the values of the policy generation config object list above.



| Name              | Type   | Description                                                                  |
|-------------------|--------|------------------------------------------------------------------------------|
| root_app_scope_id | string | The unique identifier of the root scope to which this default config applies |

Response object: Returns the default policy generation config. Response object: Returns the default policy generation config.

## Forensics Intent

The software agents APIs are associated with managing forensic intents.

Forensic intents link a forensic profile with the group of agents it applies to. The group of agents is defined using an inventory filter.

These set of APIs require the `sensor_management` capability associated with the API key.



**Note** These APIs are only available to site admins and owners of root scopes.

## Forensic intent object

| Attribute                  | Type    | Description                                           |
|----------------------------|---------|-------------------------------------------------------|
| id                         | string  | unique identifier of the intent                       |
| name                       | string  | name of the intent                                    |
| inventory_filter_id        | array   | id of the inventory filter associated with the intent |
| forensic_config_profile_id | integer | id of the profile associated with this intent         |
| created_at                 | integer | Unix timestamp of when the intent was created         |
| updated_at                 | integer | Unix timestamp of when the intent was last updated    |

## Listing a forensic intents

This endpoint lists all existing forensic profiles

GET `/openapi/v1/inventory_config/forensic_intents`

Parameters: None

This endpoint returns an array of forensic intent object summaries.

## Retrieving a Single Forensic Intent

```
GET /openapi/v1/inventory_config/forensic_intents/{intent_id}
```

Parameters:

| Name      | Type   | Description      |
|-----------|--------|------------------|
| intent_id | string | id of the intent |

Returns a detailed representation of the forensic intent object.

## Creating a Forensic Intent

```
POST /openapi/v1/inventory_config/forensic_intents
```

Parameters:

| Name                       | Type   | Description                                           |
|----------------------------|--------|-------------------------------------------------------|
| name                       | string | name of the intent                                    |
| inventory_filter_id        | string | id of the inventory filter associated with the intent |
| forensic_config_profile_id | string | id of the forensic profile associated with the intent |

Returns a forensic intent object.

## Update a Forensic Intent

```
PUT /openapi/v1/inventory_config/forensic_intents/{intent_id}
```

Parameters:

| Name                       | Type   | Description                                           |
|----------------------------|--------|-------------------------------------------------------|
| intent_id                  | string | id of the intent                                      |
| name                       | string | name of the intent                                    |
| inventory_filter_id        | string | id of the inventory filter associated with the intent |
| forensic_config_profile_id | string | id of the forensic profile associated with the intent |

Returns a forensic intent object.

## Delete a Forensic Intent

```
DELETE /openapi/v1/inventory_config/forensic_intents/{intent_id}
```

Parameters:

| Name      | Type   | Description      |
|-----------|--------|------------------|
| intent_id | string | id of the intent |

Returns a 200 on success.

## Forensics Intent Orders

The software agents APIs are associated with managing forensic intent orders.

Forensic profiles are applied to agents using intent. The intents use inventory filters to define the groups of agents. If the filters overlap we need know which to apply. We use the order to define intent priority.

These set of APIs require the 'sensor\_management' capability associated with the API key.



**Note** These APIs are only available to site admins and owners of root scopes.

- [Forensic Intent Order Object, on page 841](#)
- [Retrieve the Current Forensic Intent Order, on page 841](#)
- [Creating a Forensic Intent, on page 840](#)

## Forensic Intent Order Object

| Attribute  | Type   | Description                         |
|------------|--------|-------------------------------------|
| version    | string | version of the current ordering     |
| intents    | array  | name of the intent objects in order |
| intent_ids | array  | array of forensic intent ids        |

## Retrieve the Current Forensic Intent Order

This endpoint returns the current forensic intent order.

```
GET /openapi/v1/inventory_config/forensic_orders
```

Parameters: None

This endpoint returns the current forensic intent order object.

## Creating a Forensic Intent Order

```
POST /openapi/v1/inventory_config/forensic_orders
```

Parameters:

| Name       | Type   | Description                  |
|------------|--------|------------------------------|
| version    | string | must match the current order |
| intent_ids | array  | array of the intent ids      |

Returns a forensic intent order object.

## Forensics Profiles

The software agents APIs are associated with managing forensic profiles.

Forensic profiles are collections of rules that can be applied to groups of agents using forensic intents.

These set of APIs require the 'sensor\_management' capability associated with the API key.




---

**Note** These APIs are only available to site admins and owners of root scopes.

---

- [Forensic Profile Object, on page 842](#)
- [Listing Forensic Profiles, on page 843](#)
- [Retrieving a Single Forensic Profile, on page 843](#)
- [Creating a Forensic Profile, on page 843](#)
- [Update a Forensic Profile, on page 843](#)
- [Delete a Forensic Profile, on page 844](#)

## Forensic Profile Object

| Attribute         | Type    | Description                                         |
|-------------------|---------|-----------------------------------------------------|
| id                | string  | unique identifier of the profile                    |
| name              | string  | name of the profile                                 |
| forensic_rules    | array   | array of the rules associated with the profile      |
| created_at        | integer | Unix timestamp of when the profile was created      |
| updated_at        | integer | Unix timestamp of when the profile was last updated |
| is_readonly       | boolean | indicates if the profile is read only               |
| root_app_scope_id | string  | id of the root scope to which the profile belongs   |

## Listing Forensic Profiles

This endpoint lists all existing forensic profiles

GET /openapi/v1/inventory\_config/forensic\_profiles

Parameters: None

This endpoint returns an array of forensic profile object summaries.

## Retrieving a Single Forensic Profile

GET /openapi/v1/inventory\_config/forensic\_profiles/{profile\_id}

Parameters:

| Name       | Type   | Description       |
|------------|--------|-------------------|
| profile_id | string | id of the profile |

Returns a detailed representation of the forensic profile object.

## Creating a Forensic Profile

POST /openapi/v1/inventory\_config/forensic\_profiles

Parameters:

| Name              | Type   | Description                                                   |
|-------------------|--------|---------------------------------------------------------------|
| name              | string | name of the profile                                           |
| root_app_scope_id | string | id of the root scope to which the profile belongs             |
| forensic_rule_ids | array  | array of forensic rule ids to be associated with this profile |

Returns a forensic profile object.

## Update a Forensic Profile

PUT /openapi/v1/inventory\_config/forensic\_profiles/{id}

Parameters:

| Name              | Type   | Description                                                   |
|-------------------|--------|---------------------------------------------------------------|
| profile_id        | string | id of the profile                                             |
| name              | string | name of the profile                                           |
| forensic_rule_ids | array  | array of forensic rule ids to be associated with this profile |

Returns a forensic profile object.

## Delete a Forensic Profile

```
DELETE /openapi/v1/inventory_config/forensic_profiles/{profile_id}
```

Parameters:

| Name       | Type   | Description       |
|------------|--------|-------------------|
| profile_id | string | id of the profile |

Returns a 200 on success.

## Forensics Rules

The software agents APIs are associated with managing forensic rules.

Forensic rules are used in forensic profiles which are then applied to groups of agents.

These set of APIs require the 'sensor\_management' capability associated with the API key.




---

**Note** These APIs are only available to site admins and owners of root scopes.

---

- [Forensic Rule Object, on page 844](#)
- [Listing a Forensic Rules, on page 845](#)
- [Retrieving a Single Forensic Rule, on page 845](#)
- [Creating a Forensic Rule, on page 845](#)
- [Update a Forensic Rule, on page 846](#)
- [Delete a Forensic Rule, on page 846](#)

## Forensic Rule Object

| Attribute       | Type   | Description                   |
|-----------------|--------|-------------------------------|
| id              | string | unique identifier of the rule |
| name            | string | name of the rule              |
| description     | string | description of the rule       |
| type            | string | PREDEFINED or USER_DEFINED    |
| eval_group_type | string | AS_INDIVIDUAL or AS_GROUP     |

| Attribute  | Type    | Description                                          |
|------------|---------|------------------------------------------------------|
| severity   | string  | one of IMMEDIATE_ACTION, CRITICAL, HIGH, MEDIUM, LOW |
| actions    | array   | array of ALERT or REPORT strings                     |
| created_at | integer | Unix timestamp of when the rule was created          |
| updated_at | integer | Unix timestamp of when the rule was last updated     |

## Listing a Forensic Rules

This endpoint lists all existing forensic rules

```
GET /openapi/v1/inventory_config/forensic_rules
```

Parameters: None

This endpoint returns an array of forensic rule object summaries.

## Retrieving a Single Forensic Rule

```
GET /openapi/v1/inventory_config/forensic_rules/{rule_id}
```

Parameters:

| Name    | Type   | Description    |
|---------|--------|----------------|
| rule_id | string | id of the rule |

Returns a detailed representation of the forensic rule object.

## Creating a Forensic Rule

```
POST /openapi/v1/inventory_config/forensic_rules
```

Parameters:

| Name              | Type   | Description                                     |
|-------------------|--------|-------------------------------------------------|
| root_app_scope_id | string | id of the root scope to which this rule belongs |
| name              | string | name of the rule                                |
| description       | string | description of the rule                         |
| eval_group_type   | string | type of the rule                                |

| Name     | Type   | Description                      |
|----------|--------|----------------------------------|
| severity | string | severity of the rule             |
| actions  | array  | array or ALERT or REPORT strings |
| clause   | string | the query clause of the rule.    |

Returns a forensic rule object.

## Update a Forensic Rule

PUT /openapi/v1/inventory\_config/forensic\_rules/{rule\_id}

Parameters:

| Name            | Type   | Description                      |
|-----------------|--------|----------------------------------|
| rule_id         | string | id of the rule                   |
| name            | string | name of the rule                 |
| description     | string | description of the rule          |
| eval_group_type | string | type of the rule                 |
| severity        | string | severity of the rule             |
| actions         | array  | array or ALERT or REPORT strings |
| clause          | string | the query clause of the rule.    |

Returns a forensic rule object.

## Delete a Forensic Rule

DELETE /openapi/v1/inventory\_config/forensic\_rules/{rule\_id}

Parameters:

| Name    | Type   | Description    |
|---------|--------|----------------|
| rule_id | string | id of the rule |

Returns a 200 on success.



## Enforcement

Policy enforcement is the feature where generated policies are pushed to the assets in the scope associated with a workspace and new firewall rules are written. This set of APIs require the `app_policy_management` capability associated with the API key.

For more information, see [Enforce Policies](#).

## Agent Network Policy Config

This endpoint returns an [Agent](#) object according to the agent ID. It is useful for fetching the network policy, agent configuration, its version, etc.

```
GET /openapi/v1/enforcement/agents/{aid}/network_policy_config
```

Parameters:

The request URL contains the following parameters

| Name | Type   | Description                           |
|------|--------|---------------------------------------|
| aid  | string | Agent UUID for network policy config. |

The JSON query body contains the following keys

| Name                 | Type    | Description                                          |
|----------------------|---------|------------------------------------------------------|
| include_filter_names | boolean | Includes filter names and ID's in network policies.  |
| inject_versions      | boolean | Includes ADM workspace versions in network policies. |

### Response

The response of this endpoint is an [Agent](#) object.

## Concrete Policy Statistics

This endpoint returns statistics for concrete policies given the agent ID and the concrete policy ID. The endpoint returns an array of [Timeseries Concrete Policy Result](#) objects.

```
GET /openapi/v1/enforcement/agents/{aid}/concrete_policies/{cid}/stats?t0=<t0>&t1=<t1>
,→&td=<td>
```

Parameters:

The request URL contains the following parameters

| Name | Type   | Description                |
|------|--------|----------------------------|
| aid  | string | Agent UUID for statistics. |

| Name | Type   | Description                          |
|------|--------|--------------------------------------|
| cid  | string | Concrete Policy UUID for statistics. |

The JSON query body contains the following keys

**Table 55:**

| Name | Type              | Description                                                                                                                                |
|------|-------------------|--------------------------------------------------------------------------------------------------------------------------------------------|
| t0   | integer           | Start time for statistics in epoch time                                                                                                    |
| t1   | integer           | End time for statistics in epoch time                                                                                                      |
| td   | integer or string | Granularity for statistic aggregations. An integer specifies number of seconds. Strings may be passed such as “minute”, “hour”, and “day”. |

## JSON Object Definitions

### Agent

| Attribute                          | Type    | Description                                                                      |
|------------------------------------|---------|----------------------------------------------------------------------------------|
| agent_uuid                         | string  | Agent UUID.                                                                      |
| agent_config                       | object  | <a href="#">Agent Config</a>                                                     |
| agent_config_status                | object  | <a href="#">Agent Config Status</a>                                              |
| desired_network_policy_config      | object  | <a href="#">Network Policy Configuration</a>                                     |
| provisioned_network_policy_config  | object  | <a href="#">Provisioned Network Policy Config</a>                                |
| provisioned_state_update_timestamp | integer | epoch timestamp in seconds when agent acknowledged the above provisioned policy. |
| desired_policy_update_timestamp    | integer | epoch timestamp in seconds when desired_network_policy_config is generated.      |
| agent_info                         | object  | <a href="#">Agent Info</a>                                                       |
| skipped                            | boolean | true, when concrete policy generation is skipped.                                |

| Attribute | Type   | Description                                       |
|-----------|--------|---------------------------------------------------|
| message   | string | Reason why concrete policy generation is skipped. |

### Agent Config

| Attribute                  | Type    | Description                                        |
|----------------------------|---------|----------------------------------------------------|
| agent_uuid                 | string  | Agent UUID.                                        |
| enforcement_enabled        | boolean | Config stating if enforcement is enabled on Agent. |
| fail_mode                  | string  | Fail Mode.                                         |
| version                    | number  | Agent config version number.                       |
| control_tet_rules_only     | boolean | Control tet rules only config.                     |
| allow_broadcast            | boolean | Allow Broadcast config.                            |
| allow_multicast            | boolean | Allow Multicast config.                            |
| allow_link_local           | boolean | Allow Link Local config.                           |
| enforcement_cpu_quota_mode | string  | Enforcement Agent CPU quota mode.                  |
| enforcement_cpu_quota_us   | string  | Enforcement Agent CPU quota microseconds.          |
| enforcement_max_rss_limit  | number  | Enforcement Agent Max RSS limit.                   |

### Network Policy Configuration

| Attribute                | Type   | Description                                                      |
|--------------------------|--------|------------------------------------------------------------------|
| version                  | string | Version number.                                                  |
| network_policy           | array  | Array of <a href="#">Network Policy</a> objects.                 |
| address_sets             | array  | Array of <a href="#">Address Set</a> objects for IP set feature. |
| container_network_policy | array  | Array of <a href="#">ContainerNetworkPolicy</a> objects.         |

### Network Policy

| Attribute | Type   | Description                  |
|-----------|--------|------------------------------|
| priority  | string | Priority of concrete policy. |

| Attribute              | Type   | Description                                                                                                           |
|------------------------|--------|-----------------------------------------------------------------------------------------------------------------------|
| enforcement_intent_id  | string | Enforcement Intent ID.                                                                                                |
| concrete_policy_id     | string | Concrete Policy ID.                                                                                                   |
| match                  | object | <a href="#">Match</a> criteria for policy. This field is deprecated.                                                  |
| action                 | object | <a href="#">Action</a> for policy match.                                                                              |
| workspace_id           | string | ID for a workspace.                                                                                                   |
| adm_data_set_id        | string | Automatic policy discovery data set id of workspace.                                                                  |
| adm_data_set_version   | string | Automatic policy discovery data set version of the workspace. Set only when inject_versions=true is passed in params. |
| cluster_edge_id        | string | Cluster Edge ID.                                                                                                      |
| policy_intent_group_id | string | Policy intent group ID.                                                                                               |
| match_set              | object | <a href="#">Match Set</a> object for IP set support. Exactly one of match or match_set will be present.               |
| src_filter_id          | string | Source inventory filter ID. This will be set when include_filter_names=true passed as params.                         |
| src_filter_name        | string | Source inventory filter name. This will be set when include_filter_names=true passed as params.                       |
| dst_filter_id          | string | Destination inventory filter ID. This will be set when include_filter_names=true passed as params.                    |
| dst_filter_name        | string | Destination Inventory filter name. This will be set when include_filter_names=true passed as params.                  |

### ContainerNetworkPolicy

| Attribute | Type   | Description |
|-----------|--------|-------------|
| pod_id    | string | POD ID.     |

| Attribute        | Type   | Description                                      |
|------------------|--------|--------------------------------------------------|
| network_policy   | array  | Array of <a href="#">Network Policy</a> objects. |
| deployment       | string | Deployment Name.                                 |
| service_endpoint | array  | List of service endpoint names.                  |

### Match

| Attribute            | Type   | Description                                                   |
|----------------------|--------|---------------------------------------------------------------|
| src_addr             | object | <a href="#">Subnet</a> object for source address.             |
| dst_addr             | object | <a href="#">Subnet</a> object for destination address.        |
| src_port_range_start | int    | Source port range start.                                      |
| src_port_range_end   | int    | Source port range end.                                        |
| dst_port_range_start | int    | Destination port range start.                                 |
| dst_port_range_end   | int    | Destination port range end.                                   |
| ip_protocol          | string | IP Protocol.                                                  |
| address_family       | string | IPv4 or IPv6 address family.                                  |
| direction            | string | Direction of match, INGRESS or EGRESS.                        |
| src_addr_range       | object | <a href="#">Address Range</a> object for source address.      |
| dst_add_range        | object | <a href="#">Address Range</a> object for destination address. |

### Action

| Attribute | Type   | Description  |
|-----------|--------|--------------|
| type      | string | Action type. |

### Match Set

| Attribute  | Type   | Description                                                                                                                              |
|------------|--------|------------------------------------------------------------------------------------------------------------------------------------------|
| src_set_id | string | Source set ID of <a href="#">Address Set</a> object in the <a href="#">Network Policy Configuration</a> <code>address_sets</code> array. |

| Attribute      | Type   | Description                                                                                                                                   |
|----------------|--------|-----------------------------------------------------------------------------------------------------------------------------------------------|
| dst_set_id     | string | Destination set ID of <a href="#">Address Set</a> object in the <a href="#">Network Policy Configuration</a> <code>address_sets</code> array. |
| src_ports      | array  | Array of <a href="#">Port Range</a> objects for source ports.                                                                                 |
| dst_ports      | array  | Array of <a href="#">Port Range</a> objects for destination ports.                                                                            |
| ip_protocol    | string | IP Protocol.                                                                                                                                  |
| address_family | string | IPv4 or IPv6 address family.                                                                                                                  |
| direction      | string | Direction of match, INGRESS or EGRESS.                                                                                                        |

### Address Set

| Attribute   | Type   | Description                                     |
|-------------|--------|-------------------------------------------------|
| set_id      | string | Address set ID.                                 |
| addr_ranges | array  | Array of <a href="#">Address Range</a> objects. |
| subnets     | array  | Array of <a href="#">Subnet</a> objects.        |
| addr_family | string | IPv4 or IPv6 address family.                    |

### Subnet

| Attribute     | Type   | Description               |
|---------------|--------|---------------------------|
| ip_addr       | string | IP address.               |
| prefix_length | int    | Prefix length for subnet. |

### Address Range

| Attribute     | Type   | Description                 |
|---------------|--------|-----------------------------|
| start_ip_addr | string | Start IP address for range. |
| end_ip_addr   | string | End IP address for range.   |

**Port Range**

| Attribute  | Type | Description           |
|------------|------|-----------------------|
| start_port | int  | Start port for range. |
| end_port   | int  | End port for range.   |

**Agent Config Status**

| Attribute            | Type    | Description                                         |
|----------------------|---------|-----------------------------------------------------|
| disabled             | boolean | Config stating is enforcement is disabled on Agent. |
| current_version      | number  | Current Agent config version applied on Agent.      |
| highest_seen_version | number  | Highest version of agent config received by Agent.  |

**Provisioned Network Policy Config**

| Attribute            | Type    | Description                                                                |
|----------------------|---------|----------------------------------------------------------------------------|
| version              | string  | Network policy config version provisioned by Agent.                        |
| error_reason         | string  | CONFIG_SUCCESS when Agent successfully applied policies else error reason. |
| disabled             | boolean | Config stating is enforcement is disabled on Agent.                        |
| current_version      | number  | Current NPC version applied on Agent.                                      |
| highest_seen_version | number  | Highest version of NPC received by Agent.                                  |
| policy_status        | object  | Every network policy status.                                               |

**Agent Info**

| Attribute            | Type    | Description                                  |
|----------------------|---------|----------------------------------------------|
| agent_info_supported | boolean | Agent capability if agent_info is supported. |
| ipset_supported      | boolean | Agent capability if ipsets are supported.    |

**Concrete Policy Result**

| Attribute  | Type | Description                            |
|------------|------|----------------------------------------|
| byte_count | int  | Byte count for concrete policy hits.   |
| pkt_count  | int  | Packet count for concrete policy hits. |

**Timeseries Concrete Policy Result**

| Attribute | Type   | Description                                  |
|-----------|--------|----------------------------------------------|
| timestamp | string | Timestamp string for aggregation of results. |
| result    | object | <a href="#">Concrete Policy Result</a>       |

## Client Server configuration

Detecting client and server relationships is central to various features in Secure Workload which is why we recommend using the Software Agent whenever possible as it can report the ground truth. Any telemetry monitoring point in the network cannot guarantee to observe every packet for a given flow - due to a wide range of circumstances, for example: two unidirectional halves of a TCP flow may take unique paths through the network - therefore will always unavoidably be affected by a level of error.

Secure Workload attempts to detect and minimize these errors without any user interaction by applying machine learning algorithms to each flow, building a statistical model which provides a judgement when inconsistent telemetry is reported. For the majority of cases, users do not need to worry about this set of APIs. However, in some minority of cases the client server detection algorithm does not get the flow direction correct. Features which rely on flow direction, for example, automatic policy discovery may exhibit undesired behavior like opening unnecessary ports.

A set of APIs are provided that can be used to provide hints about known server ports to Secure Workload algorithms. This set of APIs is available to users with root scope ownership role and requires the `app_policy_management` capability associated with the API key for those users.

There are 2 options for Client Server configuration:

## Host Config

Configuration of known server ports that are applicable to a specific subset of IP addresses within a root scope

**Add server port configuration**

This API can be used to provide hints to Secure Workload algorithms about known server ports for a given root scope. You can provide a list of known TCP/UDP server ports for a set of IP addresses belonging to a root scope to aid Secure Workload algorithms with figuring out client server direction correct in flows.

```
POST /openapi/v1/adm/{root_scope_id}/server_ports
```

Parameters: The request URL contains the following parameters



| Name          | Type   | Description                           |
|---------------|--------|---------------------------------------|
| root_scope_id | string | Unique identifier for the root scope. |

Additionally, a text file provided as input to this API contains the endpoint server port configuration in the following format:

#### Endpoint server port configuration

| Attribute        | Type        | Description                                                        |
|------------------|-------------|--------------------------------------------------------------------|
| ip_address       | string      | IP Address (can be ipv4 or ipv6 address). Subnets are not allowed. |
| tcp_server_ports | List of int | List of known TCP server ports corresponding to the ip_address.    |
| udp_server_ports | List of int | List of known UDP server ports corresponding to the ip_address.    |

#### Bulk server port configuration

| Attribute   | Type                                                                | Description                                              |
|-------------|---------------------------------------------------------------------|----------------------------------------------------------|
| host_config | List of <a href="#">Endpoint server port configuration</a> objects. | List of IP addresses with associated known server ports. |

#### Sample python code

```
contents of below file:
{"host_config": [
{"ip_address": "1.1.1.1",
"tcp_server_ports": [100, 101, 102],
"udp_server_ports": [103]
},
{"ip_address": "1.1.1.2",
"tcp_server_ports": [200, 201, 202]
}
]
}

file_path = '<path_to_file>/server_ports.txt'
root_scope_id = '<root-scope-id>'
restclient.upload(file_path,
 '/adm/%s/server_ports' % root_scope_id,
 timeout=200) # seconds
```



**Note** Above API overwrites the full state of known server port configuration in the backend. If you need to modify anything, they need re-upload the full configuration after modifications.

**Get server port configuration**

This API returns list of known uploaded server ports for endpoints in a root scope.

```
GET /openapi/v1/adm/{root_scope_id}/server_ports
```

Parameters: The request URL contains the following parameters

| Name          | Type   | Description                           |
|---------------|--------|---------------------------------------|
| root_scope_id | string | Unique identifier for the root scope. |

Response object: A list of ref:*ServerPortConfig* objects.

**Sample python code**

```
root_scope_id = '<root-scope-id>'
restclient.get('/adm/%s/server_ports' % root_scope_id)
```

**Delete server port configuration**

This API deletes server port configuration for specified root scope.

```
DELETE /openapi/v1/adm/{root_scope_id}/server_ports
```

Parameters: The request URL contains the following parameters

| Name          | Type   | Description                           |
|---------------|--------|---------------------------------------|
| root_scope_id | string | Unique identifier for the root scope. |

Response object: None.

**Sample python code**

```
root_scope_id = '<root-scope-id>'
restclient.delete('/adm/%s/server_ports' % root_scope_id)
```

## Port Config

Configuration of known server ports that are applicable to all IP addresses that belong to a root scope

**Push server port configuration**

This API can be used to provide hints to Secure Workload algorithms about known server ports for a given root scope. Users can provide a list of known TCP/UDP server ports for a given root scope to aid Secure Workload algorithms with figuring out client server direction correct in flows. Users also have the option of specifying a service name associated with each server port.

There is also a default list of known services that are applicable to all root scopes(hereafter referred to as global services). This list can be overridden at any point by the user.

**Service configuration**

A service is defined to be a (port, name) pair.

| Attribute             | Type    | Description                                                |
|-----------------------|---------|------------------------------------------------------------|
| port                  | int     | TCP/UDP server port number                                 |
| name                  | string  | Service name associated with this port (optional)          |
| override_in_conflicts | boolean | Force host to be provider in case of a conflict (optional) |

### Bulk service configuration

| Attribute           | Sub-Attribute    | Type                                          | Description                |
|---------------------|------------------|-----------------------------------------------|----------------------------|
| server_ports_config | tcp_service_list | List of <i>Service configuration</i> objects. | List of known TCP services |
|                     | udp_service_list | List of <i>Service configuration</i> objects. | List of known UDP services |

Push services per root scope:

```
POST /openapi/v1/adm/{root_scope_id}/server_ports_config
```

### Sample python code

```
contents of below file:
#{"server_ports_config":
{
"tcp_service_list": [
{
"port": 80,
"name" : "http"
},
{
"port": 53,
"name" : "dns"
},
{
"port": 514,
"name" : "syslog",
"override_in_conflicts": true
}
],
"udp_service_list": [
{
"port": 161
},
{
"port": 53,
"name": "dns"
}
]
}
#}

file_path = '<path_to_file>/server_ports.json'
```

```
Updating service list for a given root scope
#restclient.upload(file_path,
'/openapi/v1/adm/{root_scope_id}/server_ports_config',
timeout=200) # seconds
```



**Note** Above API overwrites the full state of known server port configuration in the backend. If user needs to modify anything, they need re-upload the full configuration after modifications.

### Retrieve server port configuration

This API returns list of known server ports in a root scope uploaded by the user. Response is *Bulk service configuration*.

```
Retrieve configured services per root scope:
GET /openapi/v1/adm/{root_scope_id}/server_ports_config

Retrieve configured global services:
GET /openapi/v1/adm/server_ports_config
```

### Remove server port configuration

This API deletes server port configuration for specified root scope.

```
Remove configured services per root scope:
DELETE /openapi/v1/adm/{root_scope_id}/server_ports_config
```

## Software Agents

### Agent APIs

The software agents APIs are associated with managing Secure Workload software agents. These set of APIs require the `sensor_management` capability associated with the API key. *GET* APIs below are also available with `flow_inventory_query` capability associated with the API key.

#### Get software agents

This endpoint returns a list of software agents. Each software agent has two fields to describe its agent type, `agent_type_str` is in plain text while `agent_type` is an enum.

```
GET /openapi/v1/sensors
```

Parameters:

| Name  | Type    | Description                                      |
|-------|---------|--------------------------------------------------|
| limit | integer | Limits the number of results returned (optional) |

| Name   | Type   | Description                                                                                                                                                         |
|--------|--------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| offset | string | Offset is used for paginated requests. If response returns offset then subsequent request must use the same offset to get more results in the next page. (optional) |

### Get specific software agent

This endpoint returns attributes for the software agent whose UUID is part of the URI. Each software agent has two fields to describe its agent type, <agent\_type\_str> is in plain text while <agent\_type> is an enum.

```
GET /openapi/v1/sensors/{uuid}
```

### Deleting software agent

This endpoint is used to decommission a software agent given its UUID.

Use the API with caution; once an agent is deleted, it is no longer available in the Secure Workload dashboard and if the agent is active, flow exports from the agent are not allowed in Secure Workload.

```
DELETE /openapi/v1/sensors/{uuid}
```

## Software agent configuration using Intents

This API workflow uses few REST endpoints defined below.

### Creating an inventory filter

This endpoint is used to specify criteria that match agent hosts on which user wants to configure software agents.

```
POST /openapi/v1/filters/inventories
```

Parameters:

| Name         | Type   | Description                                     |
|--------------|--------|-------------------------------------------------|
| app_scope_id | string | The scope ID to assign to the inventory filter. |
| name         | string | A name for the inventory filter.                |
| query        | json   | Filter or match criteria for agent host.        |

### Sample python code

```
app_scope_id can be retrieved by /app_scopes API
req_payload = {
 "app_scope_id": <app_scope_id>,
 "name": "sensor_config_inventory_filter",
 "query": {
 "type": "eq",
```

```

 "field": "ip",
 "value": <sensor_interface_ip>
 }
}
resp = restclient.post('/filters/inventories',
 json_body=json.dumps(req_payload))
print resp.status_code
returned response will contain the created filter and it's ID.

```

### Creating a software agent configuration profile

This endpoint is used to specify the set of configuration options to apply to target set of software agents.

POST /openapi/v1/inventory\_config/profiles

Following configuration options can be specified as part of agent configuration profile:

- `allow_broadcast`: option to allow/disallow broadcast traffic (default value of this option is True).
- `allow_multicast`: option to allow/disallow multicast traffic (default value of this option is True).
- `allow_link_local`: option to allow/disallow link local traffic (default value of this option is True).
- `auto_upgrade_opt_out`: if true, agents are not auto-upgraded during upgrade of Secure Workload cluster.
- `data_plane_disabled`: if true, agent stops reporting flows to Secure Workload.
- `enable_forensics`: option to enable collection of forensic events on the workload (agent uses more CPU as a result).
- `enable_meltdown`: enables Meltdown Exploit detection on the workload (agent uses more CPU as a result).
- `enable_pid_lookup`: if true, agent tries to attach process information to flows. Note this config option uses more CPU on the end host.
- `enforcement_disabled`: can be used to disable enforcement on hosts running enforcement agents.
- `preserve_existing_rules`: option to specify whether to preserve existing iptable rules.
- `windows_enforcement_mode`: option to use WAF (Windows Advanced Firewall) or WFP (Windows Filtering Platform) (default option is WAF).

For more details about the configuration options, refer to [Software Agent Config](#)

### Sample python code

```

Define profile to disable data_plane on agent
req_payload = {
 "root_app_scope_id": <root_app_scope_id>,
 "data_plane_disabled": True,
 "name": "sensor_config_profile_1",
 "enable_pid_lookup": True,
 "enforcement_disabled": False
}
resp = restclient.post('/inventory_config/profiles',
 json_body=json.dumps(req_payload))
print resp.status_code
returned response will contain the created profile and it's ID.
parsed_resp = json.loads(resp.content)

```

### Get software agent configuration profiles

This endpoint returns a list of software agent configuration profiles visible to the user.

```
GET /openapi/v1/inventory_config/profiles
```

Parameters: None

### Get specific software agent configuration profile

This endpoint returns an instance of software agent configuration profile.

```
GET /openapi/v1/inventory_config/profiles/{profile_id}
```

Returns the software agent configuration profile object associated with the specified ID.

### Update a software agent configuration profile

This endpoint updates a software agent configuration profile.

```
PUT /openapi/v1/inventory_config/profiles/{profile_id}
```

Following configuration options can be specified as part of agent configuration profile:

- `allow_broadcast`: option to allow/disallow broadcast traffic (default value of this option is True).
- `allow_multicast`: option to allow/disallow multicast traffic (default value of this option is True).
- `allow_link_local`: option to allow/disallow link local traffic (default value of this option is True).
- `auto_upgrade_opt_out`: if true, agents are not auto-upgraded during upgrade of Secure Workload cluster.
- `data_plane_disabled`: if true, agent stops reporting flows to Secure Workload.
- `enable_forensics`: option to enable collection of forensic events on the workload (agent uses more CPU as a result).
- `enable_meltdown`: enables Meltdown Exploit detection on the workload (agent uses more CPU as a result).
- `enable_pid_lookup`: if true, agent tries to attach process information to flows. Note this config option uses more CPU on the end host.
- `enforcement_disabled`: can be used to disable enforcement on hosts running enforcement agents.
- `preserve_existing_rules`: option to specify whether to preserve existing iptable rules.
- `windows_enforcement_mode`: option to use WAF (Windows Advanced Firewall) or WFP (Windows Filtering Platform) (default option is WAF).

For more details about the configuration options, refer to [Software Agent Config](#)

Returns the modified software agent configuration profile object associate with the specified ID.

### Delete a software agent configuration profile

This endpoint deletes the specified software agent configuration profile.

```
DELETE /openapi/v1/inventory_config/profiles/{profile_id}
```

### Creating a software agent configuration intent

This endpoint is used to specify the intent to apply set of configuration options to specified set of software agents. This will create the intent and updates the intent order by adding the newly created intent to the order.

```
POST /openapi/v1/inventory_config/intents
```

#### Sample python code

```
req_payload = {
 "inventory_config_profile_id": <>,
 "inventory_filter_id": <>
}
resp = restclient.post('/inventory_config/intents',
 json_body=json.dumps(req_payload))
print resp.status_code
returned response will contain the created intent object and it's ID.
```

### Specifying order of intents

This endpoint is used to specify the ordering of various software agent configuration intents. For example, there could be two intents – one to enable process ID lookup on development machines and second one to disable process ID lookup on windows machines. If the first intent has higher priority, then development windows machines will have process ID lookup enabled. NOTE: By default, when intent is created, it is added to the beginning of intent orders list. This endpoint is only to be used if end user needs to modify the existing order of intents.

```
POST /openapi/v1/inventory_config/orders
```

#### Sample python code

```
Read the agent config intents ordered list
resp = restclient.get('/inventory_config/orders')
order_result_json = json.loads(resp.content)

Modify the list by prepending the new intent in the list
order_rslt_json['intent_ids'].insert(0,<intent_id>)

Post the new ordering back to the server
resp = restclient.post('/inventory_config/orders',
 json_body=json.dumps(order_rslt_json))
```

### Remove agent config intent

This endpoint is used to remove a specific agent configuration intent.

```
DELETE /openapi/v1/inventory_config/intents/{intent_id}
```

#### Sample python code

```
intent_id = '588a51dcb5b30d0ee6da084a'
resp = restclient.delete('/inventory_config/intents/%s' % intent_id)
```

## Interface Config Intents

The recommended way to assign VRFs to agents is using Remote VRF configuration settings. In rare cases, when agent hosts may have multiple interfaces that need to be assigned different VRFs, users can choose to assign them VRFs using Interface Config Intents. Go to **Manage > Agents** and click the **Configure** tab.



### Inventory Config Intent Object

The GET and POST methods return an array of inventory config intent JSON objects. The object's attributes are described below:

| Attribute           | Type    | Description                                                         |
|---------------------|---------|---------------------------------------------------------------------|
| vrf_id              | integer | VRF ID integer                                                      |
| vrf_name            | string  | VRF Name                                                            |
| inventory_filter_id | string  | Inventory Filter ID                                                 |
| inventory_filter    | JSON    | Inventory filter. See OpenAPI > Inventory Filters for more details. |

### Get Interface Config Intents

This endpoint returns a list of inventory config intents to the user.

GET /openapi/v1/inventory\_config/interface\_intents

Parameters: None

### Create or Update list of Interface Config Intents

This endpoint is used to create or modify list of interface config intents. The API takes an ordered list of intents. To remove an intent in this list, users would need to read the existing list of intents, modify the list and write the modified list back.

POST /openapi/v1/inventory\_config/interface\_intents

Parameters:

| Name                | Type    | Description                            |
|---------------------|---------|----------------------------------------|
| inventory_filter_id | string  | Inventory filter ID to match interface |
| vrf_id              | integer | VRF ID to assign interface             |

### Sample python code

```
req_payload = {
 "intents": [
 {"inventory_filter_id": <inventory_filter_id_1>, "vrf_id": <vrf_id_1>},
 {"inventory_filter_id": <inventory_filter_id_1>, "vrf_id": <vrf_id_2>}
]
}
resp = restclient.post('/inventory_config/interface_intents',
 json_body=json.dumps(req_payload))
```

## VRF configuration for agents behind NAT

Following set of APIs are useful to specify policies to assign VRFs to agents behind NAT boxes. These set of APIs require the `sensor_management` capability associated with the API key and are only available to site admin users.

### List VRF configuration rules for agents behind NAT

This endpoint returns a list of VRF configuration rules applicable to agents behind NAT.

```
GET /openapi/v1/agentnatconfig
```

### Create a new VRF configuration applicable to agents behind NAT

This endpoint is used to specify criteria for VRF labeling for hosts based on their source IP and source port as seen by Secure Workload appliance.

```
POST /openapi/v1/agentnatconfig
```

Parameters:

| Name                 | Type    | Description                                                                                                   |
|----------------------|---------|---------------------------------------------------------------------------------------------------------------|
| src_subnet           | string  | Subnet to which source IP can belong to (CIDR notation).                                                      |
| src_port_range_start | integer | Lower bound of source port range (0-65535).                                                                   |
| src_port_range_end   | integer | Upper bound of source port range (0-65535).                                                                   |
| vrf_id               | integer | VRF ID to use for labeling flows for agents whose source address and port falls in the above specified range. |

### Sample python code

```
req_payload = {
 src_subnet: 10.1.1.0/24, # src IP range for sensors
 src_port_range_start: 0,
 src_port_range_end: 65535,
 vrf_id: 676767 # VRF ID to assign
}

resp = rc.post('/agentnatconfig', json_body=json.dumps(req_payload))
print resp.status_code
```

### Delete existing VRF configuration

```
DELETE /openapi/v1/agentnatconfig/{nat_config_id}
```

# Secure Workload software download

The Secure Workload software download feature provides a way to download software packages for Secure Workload agents. These set of APIs require the `software_download` capability associated with the API key. This capability is only available to site admin users, root scope owners and users with agent installer roles.

## API to get supported platforms

This end point returns the list of supported platforms.

```
GET /openapi/v1/sw_assets/platforms
```

Parameters: None

Response object: Returns the list of supported platforms.

### Sample python code

The sample code below retrieves all the supported platforms.

```
resp = restclient.get('/sw_assets/platforms')
if resp.status_code == 200:
 parsed_resp = json.loads(resp.content)
 print json.dumps(parsed_resp)
```

### Sample response

```
{"results": [{"platform": "OracleServer-6.3", "agent_type": "enforcer", "arch": "x86_64"}, {"platform": "MSWindows8Enterprise", "agent_type": "legacy_sensor", "arch": "x86_64"}]}
```

## API to get supported software version

This endpoint returns the list of supported software version for specified “agent\_type”, “package\_type”, “platform” and “architecture”.

```
GET /openapi/v1/sw_assets/download?platform=<platform>&agent_type=<agent_type>&pkg_type=<pkg_type>&arch=<arch>&list_version=<list_version>&installation_id=<installation_id>
```

where <agent\_type>, <platform> and <arch> can be any one of the results retrieved from the **API to get supported platforms**, and <pkg\_type> can be either “sensor\_w\_cfg” or “sensor\_bin\_pkg”. Both <pkg\_type> and <agent\_type> are optional but at least one of them should be specified. <list\_version> must be “True” to enable this API.

Parameters: The request URL contains the following parameters.

| Name       | Type   | Description                                                                                      |
|------------|--------|--------------------------------------------------------------------------------------------------|
| platform   | string | Specify the platform.                                                                            |
| agent_type | string | (optional) Specify the agent type.                                                               |
| pkg_type   | string | (optional) Specify the package type, the value can be either “sensor_w_cfg” or “sensor_bin_pkg”. |
| arch       | string | Specify the architecture.                                                                        |

| Name         | Type   | Description                                      |
|--------------|--------|--------------------------------------------------|
| list_version | string | Set to “True” to enable software version search. |

Response object: Returns a list of supported software version.

### Sample python code

```
resp =
restclient.get('/sw_assets/download?platform=OracleServer-6.3&pkg_type=sensor_w_cfg&arch=x86_64&list_version=True')
if resp.status_code == 200:
 print resp.content

resp =
restclient.get('/sw_assets/download?platform=OracleServer-6.3&pkg_type=sensor_bin_pkg&arch=x86_64&list_version=True')
if resp.status_code == 200:
 print resp.content
```

### Sample response

```
3.3.1.30.devel
3.3.1.31.devel
```

## API to create installer ID

This endpoint creates an installer ID for API to download Secure Workload software.

```
GET /openapi/v1/sw_assets/installation_id
```

Response object: Returns an installer ID that can be used in API to download Secure Workload software.

### Sample python code

```
resp = restclient.get('/sw_assets/installation_id')
if resp.status_code == 200:
 parsed_resp = json.loads(resp.content)
 print json.dumps(parsed_resp)
```

### Sample response

```
"5af2085fcd27819d57f3254750f5ed186451867038a54905408134a9c588292dfad06085567722a719f0039d089ff6032e67846922ba5f874269ba6505e49460509"
```

## API to download Secure Workload software

This endpoint enables clients to download the software for specified “agent\_type”, “package\_type”, “platform”, “ar- chitecture” and “sensor\_version”.

```
GET
/openapi/v1/sw_assets/download?platform=<platform>&agent_type=<agent_type>&pkg_type=<pkg_type>&arch=<arch>&sensor_version=<sensor_version>&installation_id=<installer_id>
```

where <agent\_type>, <platform> and <arch> can be any one of the results retrieved from the **API to get supported platforms**, and <pkg\_type> can be either “sensor\_w\_cfg” or “sensor\_bin\_pkg”. Both <pkg\_type> and <agent\_type> are optional but at least one of them should be specified. <sensor\_version> can be any one of the results retrieved from the **API to get supported software version**. If “sensor\_version” is not specified, it will download the **latest** software.

Parameters: The request URL contains the following parameters.

| Name           | Type   | Description                                                                                      |
|----------------|--------|--------------------------------------------------------------------------------------------------|
| platform       | string | Specify the platform.                                                                            |
| agent_type     | string | (optional) Specify the agent type.                                                               |
| pkg_type       | string | (optional) Specify the package type, the value can be either “sensor_w_cfg” or “sensor_bin_pkg”. |
| arch           | string | Specify the architecture.                                                                        |
| sensor_version | string | (optional) Specify the software version, defaults to empty string.                               |

Response object: Returns the Secure Workload software for the given parameters.

#### Sample python code

```
resp =
if resp.status_code == 200:
 print 'file downloaded successfully'
```

## Secure Workload Agents Upgrade

The Secure Workload agents upgrade feature provides a way to upgrade installed Secure Workload agents to specific version. It only updates the metadata, actual upgrade will happen during next check-in. The API requires the `software_download` capability associated with the API key. This capability is only available to site admin users, root scope owners or users with agent installer roles.

### API to upgrade an agent to specific version

This end point triggers the agent given its “UUID” upgrade to specific “sensor\_version”, the latest version will be applied if “sensor\_version” is not provided. This API won’t proceed downgrade requests.

```
POST /openapi/v1/sensors/{UUID}/upgrade?sensor_version=<sensor_version>
```

where <sensor\_version> can be any one of the results retrieved from the [API to get supported software version](#)

Parameters: The request URL contains the following parameters

| Name           | Type   | Description                                                                           |
|----------------|--------|---------------------------------------------------------------------------------------|
| sensor_version | string | (optional) Specify the desired version, the latest version will be applied by default |

Returns the status for this upgrade request.

#### Sample python code

```

resp = restclient.post('/openapi/v1/sensors/{UUID}/upgrade?sensor_version=3.4.1.1.devel')

if resp.status_code == 200:
 print 'agent upgrade was triggered successfully and in progress'
elif resp.status_code == 304:
 print 'provided version is not newer than current version'
elif resp.status_code == 400:
 print 'provided version is invalid'
elif resp.status_code == 403:
 print 'user does not have required capability'
elif resp.status_code == 404:
 print 'agent with {UUID} does not exist'

```

## User Uploaded Filehashes

Users can upload a list of filehashes to Secure Workload and specify whether those hashes are benign or flagged. Secure Workload flags processes with the respective binary hashes accordingly.

This set of APIs can be used to upload or remove a list of filehashes to Secure Workload. To call these APIs, use an API key with the `user_data_upload` capability.



**Note** You can have up to 1 million file hashes per root scope. 500000 for both benign and flagged hashes each.

**The following APIs are available to scope owners and site admins and are used to upload/download/remove filehashes in a single root scope on the |product| appliance.**

## User Filehash Upload

This endpoint is used to upload a CSV file with filehash for a root scope on the Secure Workload appliance. The column headers `HashType` and `FileHash` must appear in the CSV file. `HashType` should be `SHA-1` or `SHA-256`, `FileHash` must not be empty and must be in the format of 40-hex SHA1 or 64-hex SHA256.

`FileName` and `Notes` headers are optional. Given file name should not exceed maximum length of 150 characters and given notes should not exceed maximum length of 1024 characters.

```
POST /openapi/v1/assets/user_filehash/upload/{rootAppScopeNameOrID}/{benignOrflagged}
```

Parameters: The request URL contains the following parameters

| Name                 | Type   | Description                                                 |
|----------------------|--------|-------------------------------------------------------------|
| rootAppScopeNameOrID | string | Root scope name or ID.                                      |
| benignOrflagged      | string | Can be one of <code>benign</code> or <code>flagged</code> . |

Response object: None

### Sample python code

```

Sample CSV File
HashType, FileHash, FileName, Notes
SHA-1, 1AF17E73721DBE0C40011B82ED4BB1A7DBE3CE29, application_1.exe, Sample Notes
#

```

```
SHA-256,8F434346648F6B96DF89DDA901C5176B10A6D83961DD3C1AC88B59B2DC327AA4,application_2.exe,Sample
Notes
```

```
file_path = '<path_to_file>/user_filehash.csv'
root_app_scope_name = 'Tetration'
restclient.upload(file_path, '/assets/user_filehash/upload/%s/benign' % root_app_scope_name)
```

## User Filehash Delete

This endpoint is used to upload a CSV file to delete filehashes from the root scope on the Secure Workload appliance. CSV file must have `FileHash` as a header.

```
POST /openapi/v1/assets/user_filehash/delete/{rootAppScopeNameOrID}/{benignOrflagged}
```

Parameters: The request URL contains the following parameters

| Name                 | Type   | Description                                                  |
|----------------------|--------|--------------------------------------------------------------|
| rootAppScopeNameOrID | string | Root scope name or ID.                                       |
| benignOrflagged      | string | Can be one of <code>benign</code> and <code>flagged</code> . |

Response object: None

### Sample python code

```
Sample CSV File
FileHash
1AF17E73721DBE0C40011B82ED4BB1A7DBE3CE29
8F434346648F6B96DF89DDA901C5176B10A6D83961DD3C1AC88B59B2DC327AA4

file_path = '<path_to_file>/user_filehash.csv'
root_app_scope_name = 'Tetration'
restclient.upload(file_path, '/assets/user_filehash/delete/' + root_app_scope_name +
'/benign')
```

## User Filehash Download

This endpoint returns the user file hash for the given root scope on the Secure Workload appliance as a CSV file. The CSV file has the headers `HashType`, `FileHash`, `FileName` and `Notes` in the respective order.

```
GET /openapi/v1/assets/user_filehash/download/{rootAppScopeNameOrID}/{benignOrflagged}
```

Parameters: The request URL contains the following parameters

| Name                 | Type   | Description                                                 |
|----------------------|--------|-------------------------------------------------------------|
| rootAppScopeNameOrID | string | Root scope name or ID.                                      |
| benignOrflagged      | string | Can be one of <code>benign</code> or <code>flagged</code> . |

Response object: None

### Sample python code

```
file_path = '<path_to_file>/output_user_filehash.csv'
```

```

root_app_scope_name = 'Tetration'
restclient.download(file_path, '/assets/user_filehash/download/%s/benign' %
root_app_scope_name)

```

## User-Defined Labels

These APIs are used to add or remove user-defined labels that label flows and inventory items on the Secure Workload appliance. To call these APIs, use an API key with the `user_data_upload` capability. See the [Label Key Schema](#) section of the UI user guide for guidelines governing keys and values that are used for labeling flows and inventory items.



**Note** Refer to [Importing Custom Labels](#) for instructions on accessing this functionality via the UI.



**Note** Refer to [Label Limits](#) for limits on the number of IPv4/IPv6 addresses or subnets that can be uploaded.

## Scope-Dependent APIs

The following APIs are used to get/set/delete labels in a root scope on the Secure Workload appliance. They are available to root **scope owners** and **site admins**. Also, the GET API calls are available to users with **read access** to the root scope.

### Get Inventory Label

This endpoint returns labels for an IPv4/IPv6 address or subnet in root scope on the Secure Workload appliance. The address/subnet used to query this endpoint must exactly match the one used for uploading labels.

```
GET /openapi/v1/inventory/tags/{rootAppScopeName}?ip={IPorSubnet}
```

Parameters: The request URL contains the following parameters

| Name             | Type   | Description                  |
|------------------|--------|------------------------------|
| rootAppScopeName | string | Root scope name.             |
| IPorSubnet       | string | IPv4/IPv6 address or subnet. |

Response object:

| Name       | Type | Description                                                   |
|------------|------|---------------------------------------------------------------|
| attributes | JSON | Key/value map for labeling matching flows and inventory items |

### Sample python code

```

root_app_scope_name = 'Tetration'
restclient.get('/inventory/tags/%s' % root_app_scope_name, params={'ip': '10.1.1.1/24'})

```



### Search Inventory Label

This endpoint allows for searching labels for an IPv4/IPv6 address or subnet in root scope on the Secure Workload appliance.

```
GET /openapi/v1/inventory/tags/{rootAppScopeName}/search?ip={IPorSubnet}
```

Parameters: The request URL contains the following parameters

| Name             | Type   | Description                  |
|------------------|--------|------------------------------|
| rootAppScopeName | string | Root scope name.             |
| IPorSubnet       | string | IPv4/IPv6 address or subnet. |

Response object: This API returns a list of objects of the following format

| Name      | Type    | Description                                     |
|-----------|---------|-------------------------------------------------|
| key       | string  | IPv4/IPv6 address or subnet.                    |
| updatedAt | integer | Unix timestamp of when the labels were updated. |
| value     | JSON    | Key/value map of attributes for the key.        |

### Sample python code

```
root_app_scope_name = 'Tetration Scope'
encoded_root_app_scope_name = urllib.quote(root_app_scope_name, safe='')
restclient.get('/inventory/tags/%s/search' % encoded_root_app_scope_name, params={'ip':
'10.1.1.1/24'})
```

### Set Inventory Label

This endpoint is used to set labels for labeling flows and inventory items in root scope on the Secure Workload appliance.

```
POST /openapi/v1/inventory/tags/{rootAppScopeName}
```

Parameters: The request URL contains the following parameters

| Name             | Type   | Description      |
|------------------|--------|------------------|
| rootAppScopeName | string | Root scope name. |

The JSON query body contains the following keys

| Name       | Type   | Description                                                   |
|------------|--------|---------------------------------------------------------------|
| ip         | string | IPv4/IPv6 address or subnet.                                  |
| attributes | JSON   | Key/value map for labeling matching flows and inventory items |

Response object:

| Name     | Type | Description                                                                  |
|----------|------|------------------------------------------------------------------------------|
| warnings | JSON | Key/value map containing warnings that are encountered while setting labels. |

### Sample python code

```
root_app_scope_name = 'Tetration'
req_payload = {'ip': '10.1.1.1/24', 'attributes': {'datacenter': 'SJC', 'location': 'CA'}}

restclient.post('/inventory/tags/%s' % root_app_scope_name,
json_body=json.dumps(req_payload))
```

### Delete Inventory Label

This endpoint deletes labels for an IPv4/IPv6 address or subnet in a root scope on the Secure Workload appliance.

```
DELETE /openapi/v1/inventory/tags/{rootAppScopeName}
```

Parameters: The request URL contains the following parameters

| Name             | Type   | Description      |
|------------------|--------|------------------|
| rootAppScopeName | string | Root scope name. |

The JSON query body contains the following keys

| Name | Type   | Description                 |
|------|--------|-----------------------------|
| ip   | string | IPv4/IPv6 address or subnet |

### Sample python code

```
root_app_scope_name = 'Tetration'
req_payload = {'ip': '10.1.1.1/24'}
restclient.delete('/inventory/tags/%s' % root_app_scope_name,
json_body=json.dumps(req_payload))
```

### Upload Labels

This endpoint is used to upload a CSV file with labels for labeling flows and inventory items in a root scope on the Secure Workload appliance. A column header with name `IP` must appear in the CSV file. Of the remaining column headers, up to 32 can be used to annotate flows and inventory items. To use non-English characters in labels, the uploaded csv file must be in UTF-8 format.

```
POST /openapi/v1/assets/cmdb/upload/{rootAppScopeName}
```

Parameters:

User must provide an operation type (`X-Tetration-Oper`) as a parameter to this API. `X-Tetration-Oper` can be one of the following:

- **add:** Appends labels to new and existing addresses/subnets. Resolves conflicts by selecting new labels over existing ones. For example, if labels for an address in the database are `{“foo”: “1”, “bar”: “2”}`, and the CSV file contains `{“z”: “1”, “bar”: “3”}`, *add* sets labels for this address to `{“foo”: “1”, “z”: “1”, “bar”: “3”}`.
- **overwrite:** inserts labels for new addresses/subnets and replaces labels for existing ones. For example, if labels for an address in the database are `{“foo”: “1”, “bar”: “2”}` and the CSV file contains `{“z”: “1”, “bar”: “3”}`, *overwrite* sets labels for this address to `{“z”: “1”, “bar”: “3”}`.
- **merge:** Merges labels to existing addresses/subnets. Resolves conflicts by selecting nonempty values over empty values. For example, if labels for an address in the database are `{“foo”: “1”, “bar”: “2”, “qux”: “”, “corge”: “4”}`, and the CSV file contains `{“z”: “1”, “bar”: “”, “qux”: “3”, “corge”: “4-updated”}`, *merge* sets that are labels for this address to `{“foo”: “1”, “z”: “1”, “bar”: “2”, “qux”: “3”, “corge”: “4-updated”}`.



**Note** Value of “bar” in not reset to “”(empty), instead existing value of “bar”=“2” is preserved.

- **delete:** removes labels for an address/subnet.

Response object:

| Name     | Type | Description                                                                  |
|----------|------|------------------------------------------------------------------------------|
| warnings | JSON | Key/value map containing warnings that are encountered while setting labels. |

#### Sample python code

```
file_path = '<path_to_file>/user_annotations.csv'
root_app_scope_name = 'Tetration'
req_payload = [tetpyclient.MultipartOption(key='X-Tetration-Oper', val='add')]
restclient.upload(file_path, '/assets/cmdb/upload/%s' % root_app_scope_name, req_payload)
```

#### Upload Labels Using JSON Format

This endpoint is used to upload labels for labeling flows and inventory items on the Secure Workload appliance using JSON format. Attributes in `ip_tags` must be a subset of headers, up to 32 headers can be used to annotate flows and inventory items. To use non-English characters in labels, the json payload must be in UTF-8 format.

POST `/openapi/v1/multi_inventory/tags/{rootAppScopeName}`

Parameters: The request URL contains the following parameter

| Name             | Type   | Description     |
|------------------|--------|-----------------|
| rootAppScopeName | string | Root scope name |

Table 56: Payload Parameters

| Name      | Type   | Description                                                                           |
|-----------|--------|---------------------------------------------------------------------------------------|
| headers   | array  | Array of strings that specifies the label names                                       |
| operation | string | Either <b>add</b> or <b>merge</b> , if not specified it is considered as <b>add</b> . |
| ip_tags   | array  | Array of ip_tags object                                                               |

Table 57: ip\_tags

| Name       | Type   | Description                                                            |
|------------|--------|------------------------------------------------------------------------|
| ip         | string | Valid IPV4/IPV6 Address or Subnet                                      |
| attributes | JSON   | JSON of label, value string pairs; labels must be a subset of headers. |



- Note**
1. If record labels are not a subset of given headers, a warning is generated to alert the user of a discrepancy between the given headers and the labels in the record.
  2. This endpoint will only allow user to upload a maximum of 95,000 entries and 36 attributes.

- For add request, *operation* is optional, if not specified, it is considered as **add**.
- For merge request, *operation* must be specified as **merge**.

3. You can upload a maximum of 95,000 entries and 36 attributes.

#### Sample python code for add request

```
{
 "headers" : ["key1", "key2"],
 "operation": "add"
 "ip_tags" : [
 {
 "ip": "10.10.10.11",
 "attributes": {
 "key1": "val1",
 "key2": "val2"
 }
 },
 {
 "ip": "10.10.10.12",
 "attributes": {
 "key1": "val111",
 "key2": "val2"
 }
 }
]
}
```

### Sample python code for merge request

```

{{
 "headers" : ["key1", "key2"],
 "operation": "merge"
 "ip_tags" : [
 {
 "ip": "10.10.10.11",
 "attributes": {
 "key1": "val1",
 "key2": "val2"
 }
 },
 {
 "ip": "10.10.10.12",
 "attributes": {
 "key1": "val1",
 "key2": "val2"
 }
 }
]
}
}
resp = restclient.post('/multi_inventory/tags/%s' % rootAppScopeName,
json_body=json.dumps(req_payload))

```

### Download User Labels

This endpoint returns user-uploaded labels for a root scope on the Secure Workload appliance as a CSV file.

GET /openapi/v1/assets/cmdb/download/{rootAppScopeName}

Parameters: The request URL contains the following parameters

| Name             | Type   | Description      |
|------------------|--------|------------------|
| rootAppScopeName | string | Root scope name. |

Response:

Content-Type: *text/csv*

CSV file containing user uploaded labels for the scope.

### Sample python code

```

file_path = '<path_to_file>/output.csv'
root_app_scope_name = 'Tetration'
restclient.download(file_path, '/assets/cmdb/download/%s' % root_app_scope_name)

```

### Get Column Headers

This endpoint returns a list of column headers for a root scope on the Secure Workload appliance.

GET /openapi/v1/assets/cmdb/attributenames/{rootAppScopeName}

Parameters: The request URL contains the following parameters

| Name             | Type   | Description      |
|------------------|--------|------------------|
| rootAppScopeName | string | Root scope name. |

Response object: An array of facets available for a label.

### Sample python code

```
root_app_scope_name = 'Tetration'
resp = restclient.get('/assets/cmdb/attributenames/%s' % root_app_scope_name)
```

### Delete Column Header

This endpoint deletes a column header in a root scope on the Secure Workload appliance. Deleting a column header drops it from the list of labeled facets and removes it from existing labels.

```
DELETE /openapi/v1/assets/cmdb/attributenames/{rootAppScopeName}/{attributeName}
```

Parameters: The request URL contains the following parameters

| Name             | Type   | Description              |
|------------------|--------|--------------------------|
| rootAppScopeName | string | Root scope name.         |
| attributeName    | string | Attribute being deleted. |

Response object: None

### Sample python code

```
root_app_scope_name = 'Tetration'
attribute_name = 'column1'
resp = restclient.delete('/assets/cmdb/attributenames/%s/%s' % (root_app_scope_name,
attribute_name))
```

### Delete Labels Using JSON Format

This endpoint deletes labels for multiple IPv4/IPv6 address or subnet using JSON format.

```
DELETE /openapi/v1/multi_inventory/tags/{rootAppScopeName}
```

Parameters: The request URL contains the following parameters

| Name             | Type   | Description     |
|------------------|--------|-----------------|
| rootAppScopeName | string | Root scope name |

**Table 58: Payload Parameters**

| Name    | Type  | Description                                                |
|---------|-------|------------------------------------------------------------|
| headers | array | (Optional) array of strings that specifies the label names |
| ip_tags | array | Array of ip_tags object                                    |

Table 59: ip\_tags

| Name       | Type   | Description                                                                       |
|------------|--------|-----------------------------------------------------------------------------------|
| ip         | string | Valid IPV4/IPV6 Address or Subnet                                                 |
| attributes | JSON   | (Optional) JSON of label, value string pairs; labels must be a subset of headers. |

### Sample python code

```
{
 "ip_tags" : [
 {
 "ip": "10.10.10.11",
 },
 {
 "ip": "10.10.10.12",
 "attributes": {
 "key1": "val1",
 "key2": "val2"
 }
 }
]
}
resp = restclient.delete('/multi_inventory/tags/%s' % rootAppScopeName,
json_body=json.dumps(req_payload))
```

### Get List of Labeled Facets

This endpoint returns a list of labeled facets for a root scope on the Secure Workload appliance. Labeled facets are a subset of column headers in the uploaded CSV file that is used for annotating flows and inventory items in that scope.

```
GET /openapi/v1/assets/cmdb/annotations/{rootAppScopeName}
```

Parameters: The request URL contains the following parameters

| Name             | Type   | Description      |
|------------------|--------|------------------|
| rootAppScopeName | string | Root scope name. |

Response object: An array of labeled facets for the root scope.

### Sample python code

```
root_app_scope_name = 'Tetration'
resp = restclient.get('/assets/cmdb/annotations/%s' % root_app_scope_name)
```

### Update List of Labeled Facets

This endpoint updates list of facets that are used for annotating flows and inventory items in a root scope on the Secure Workload appliance.

```
PUT /openapi/v1/assets/cmdb/annotations/{rootAppScopeName}
```

Parameters: The request URL contains the following parameters

| Name             | Type   | Description      |
|------------------|--------|------------------|
| rootAppScopeName | string | Root scope name. |

Response object: None

#### Sample python code

```
the following list is a subset of column headers in the
uploaded CSV file
req_payload = ['location', 'region', 'detail']
root_app_scope_name = 'Tetration'
restclient.put('/assets/cmdb/annotations/%s' % root_app_scope_name,
 json_body=json.dumps(req_payload))
```

#### Flush User Uploaded Labels

This endpoint flushes labels for flows and inventory items in root scope on the Secure Workload appliance. The changes affect new data; older labeled data remains unaltered.

POST /openapi/v1/assets/cmdb/flush/{rootAppScopeName}

Parameters: The request URL contains the following parameters

| Name             | Type   | Description      |
|------------------|--------|------------------|
| rootAppScopeName | string | Root scope name. |

Response object: None

#### Sample python code

```
root_app_scope_name = 'Tetration'
restclient.post('/assets/cmdb/flush/%s' % root_app_scope_name)
```

## Scope-Independent APIs

The following APIs can span multiple scopes on the Secure Workload appliance.



**Note** The number of scope independent and dependent annotated facets must not exceed 32 for any root scope.

#### Upload Labels

This endpoint is used to upload a CSV file with labels for labeling flows and inventory items on the Secure Workload appliance. Column headers with names `IP` and `VRF` must appear in the CSV file and `VRF` must match the root scope for a label. Of the remaining column headers, up to 32 can be used to annotate flows and inventory items.

POST /openapi/v1/assets/cmdb/upload

Parameters:



User must provide an operation type (`X-Tetration-Oper`) as a parameter to this API to specify the [operation](#) to be performed.

Response object:

| Name     | Type | Description                                                                     |
|----------|------|---------------------------------------------------------------------------------|
| warnings | JSON | Key or value map containing warnings that are encountered while setting labels. |

#### Sample python code

```
file_path = '<path_to_file>/user_annotations.csv'
req_payload = [tetpyclient.MultiPartOption(key='X-Tetration-Oper', val='add')]
restclient.upload(file_path, '/assets/cmdb/upload', req_payload)
```

#### Download User Labels

This endpoint returns the user-uploaded labels for all scopes on the Secure Workload appliance as a CSV file.

GET `/openapi/v1/assets/cmdb/download`

Parameters: None

Response:

Content-Type: `text/csv`

CSV file containing user uploaded labels for the scope.

#### Sample python code

```
file_path = '<path_to_file>/output.csv'
restclient.download(file_path, '/assets/cmdb/download')
```

## Scope-Independent Labels

These labels are not tied to a particular root scope and apply to all scopes on the appliance.

#### Get Inventory Label

These endpoints return scope independent labels for an IPv4/IPv6 address or subnet on the Secure Workload appliance. The address or subnet that is used to query this endpoint must exactly match the one used for uploading labels.

GET `/openapi/v1/si_inventory/tags?ip={IPorSubnet}`

Parameters: The request URL contains the following parameters.

| Name       | Type   | Description                  |
|------------|--------|------------------------------|
| IPorSubnet | string | IPv4/IPv6 address or subnet. |

Response object:

| Name       | Type | Description                                                      |
|------------|------|------------------------------------------------------------------|
| attributes | JSON | Key or value map for labeling matching flows and inventory items |

### Sample python code

```
restclient.get('/si_inventory/tags', params={'ip': '10.1.1.1/24'})
```

### Search Inventory Label

This endpoint allows for searching labels for an IPv4/IPv6 address or subnet on the Secure Workload appliance.

```
GET /openapi/v1/si_inventory/tags/search?ip={IPorSubnet}
```

Parameters: The request URL contains the following parameters

| Name       | Type   | Description                  |
|------------|--------|------------------------------|
| IPorSubnet | string | IPv4/IPv6 address or subnet. |

Response object: This API returns a list of objects of the following format

| Name      | Type    | Description                                     |
|-----------|---------|-------------------------------------------------|
| key       | string  | IPv4/IPv6 address or subnet.                    |
| updatedAt | integer | Unix timestamp of when the labels were updated. |
| value     | JSON    | Key or value map of attributes for the key.     |

### Sample python code

```
restclient.get('/si_inventory/tags/search', params={'ip': '10.1.1.1/24'})
```

### Set Inventory Label

This endpoint is used to set labels for labeling flows and inventory items on the Secure Workload appliance.

```
POST /openapi/v1/si_inventory/tags
```

Parameters: The JSON query body contains the following keys

| Name       | Type   | Description                                                       |
|------------|--------|-------------------------------------------------------------------|
| ip         | string | IPv4/IPv6 address or subnet.                                      |
| attributes | JSON   | Key or value map for labeling matching flows and inventory items. |

Response object:

| Name     | Type | Description                                                                     |
|----------|------|---------------------------------------------------------------------------------|
| warnings | JSON | Key or value map containing warnings that are encountered while setting labels. |

#### Sample python code

```
req_payload = {'ip': '10.1.1.1/24', 'attributes': {'datacenter': 'SJC', 'location': 'CA'}}
restclient.post('/si_inventory/tags', json_body=json.dumps(req_payload))
```

#### Delete Inventory Label

This endpoint deletes labels for an IPv4/IPv6 address or subnet on the Secure Workload appliance.

```
DELETE /openapi/v1/si_inventory/tags
```

Parameters: The JSON query body contains the following keys

| Name | Type   | Description                 |
|------|--------|-----------------------------|
| ip   | string | IPv4/IPv6 address or subnet |

#### Sample python code

```
req_payload = {'ip': '10.1.1.1/24'}
restclient.delete('/si_inventory/tags', json_body=json.dumps(req_payload))
```

#### Get List of Labeled Facets

This endpoint returns a list of scope-independent labeled facets on the Secure Workload appliance. Labeled facets are a subset of column headers that are used for annotating flows and inventory items across all scopes.




---

**Note** Exclude the scope name from the request url to view and update the list of annotated scope-independent facets.

---

```
GET /openapi/v1/assets/cmdb/annotations
```

Response object: An array of scope-independent labeled facets.

#### Sample python code

```
resp = restclient.get('/assets/cmdb/annotations')
```

#### Update List of Labeled Facets

This endpoint updates list of scope-independent facets that are used for annotating flows and inventory items on the Secure Workload appliance.

```
PUT /openapi/v1/assets/cmdb/annotations
```

Response object: None

#### Sample python code

```
the following list is a subset of column headers in the
uploaded CSV file
req_payload = ['location', 'region', 'detail']
restclient.put('/assets/cmdb/annotations',
 json_body=json.dumps(req_payload))
```

## Virtual Routing and Forwarding

This set of APIs manages Virtual Routing and Forwarding (VRF).



**Note** These APIs are only available to site admins.

## VRF Object

The VRF object attributes are described below:

| Attribute         | Type    | Description                                   |
|-------------------|---------|-----------------------------------------------|
| id                | int     | Unique identifier for the VRF.                |
| name              | string  | User specified name of the VRF.               |
| tenant_id         | int     | ID of parent tenant.                          |
| root_app_scope_id | string  | ID of associated root scope.                  |
| created_at        | integer | Unix timestamp when the VRF was created.      |
| updated_at        | integer | Unix timestamp when the VRF was last updated. |

## Get VRFs

This endpoint returns a list of VRFs. This API is available to API keys with `sensor_management` or `flow_inventory_query` capability.

```
GET /openapi/v1/vrfs
```

Parameters: None

Response object: Returns a list of VRF objects.

### Sample python code

```
resp = restclient.get('/vrfs')
```

## Create a VRF

This endpoint is used to create new VRFs. An associated root scope will automatically be created with a query matching the VRF ID. This API is available to API keys with `sensor_management` capability.

POST /openapi/v1/vrfs

Parameters:

| Name                   | Type    | Description                                                                                                                                                                                                                                           |
|------------------------|---------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| id                     | int     | (optional) Unique identifier for the VRF. If unspecified, Secure Workload cluster will generate a unique ID for the newly created VRF. Best practice is to let Secure Workload generate these IDs instead of caller explicitly specifying unique IDs. |
| tenant_id              | int     | (optional) ID of parent tenant.                                                                                                                                                                                                                       |
| name                   | string  | User specified name of the VRF.                                                                                                                                                                                                                       |
| apply_monitoring_rules | boolean | (optional) Whether collection rules should be applied for the VRF. Defaults to 'false'.                                                                                                                                                               |

The `tenant_id` is optional. If not provided, the VRF will be added to the tenant with the same id as the VRF, autocreating if necessary. If the `tenant_id` is provided, the tenant will not be automatically created, and an error will be returned if the tenant does not exist.

Response object: Returns the newly created VRF object.

### Sample python code

```
req_payload = {
 "tenant_id": <tenant_id>,
 "name": "Test",
 "apply_monitoring_rules": True
}
resp = restclient.post('/vrfs', json_body=json.dumps(req_payload))
```

## Get Specific VRF

This endpoint returns information for the specified VRF ID. This API is available to API keys with `sensor_management` or `flow_inventory_query` capability.

GET /openapi/v1/vrfs/{vrf\_id}

Parameters: The request URL contains the following parameters

| Name   | Type | Description                    |
|--------|------|--------------------------------|
| vrf_id | int  | Unique identifier for the VRF. |

Response object: Returns a VRF object that is associated with the specified ID.

### Sample python code

```
vrf_id = 676767
resp = restclient.get('/vrfs/%d'% vrf_id)
```

## Update a VRF

This endpoint updates a VRF. This API is available to API keys with `sensor_management` capability.

PUT /openapi/v1/vrfs/{vrf\_id}

Parameters: The request URL contains the following parameters

| Name   | Type | Description                    |
|--------|------|--------------------------------|
| vrf_id | int  | Unique identifier for the VRF. |

The JSON request body contains the following parameters

| Name                   | Type    | Description                                                       |
|------------------------|---------|-------------------------------------------------------------------|
| name                   | string  | User specified name of the VRF.                                   |
| apply_monitoring_rules | boolean | (optional) Whether collection rules should be applied to the VRF. |

Response object: Returns the modified VRF object that is associated with the specified ID.

### Sample python code

```
vrf_id = 676767
req_payload = {
 "name": "Test",
 "apply_monitoring_rules": True
}
resp = restclient.put('/vrfs/%d'% vrf_id,
 json_body=json.dumps(req_payload))
```

## Delete Specific VRF

This endpoint deletes a VRF. It fails if there is associated root scope. This API is available to API keys with `sensor_management` capability.

DELETE /openapi/v1/vrfs/{vrf\_id}

Parameters: The following parameter is part of the URL.

| Name   | Type | Description                    |
|--------|------|--------------------------------|
| vrf_id | int  | Unique identifier for the VRF. |

### Sample python code

```
vrf_id = 676767
resp = restclient.delete('/vrfs/%d'% vrf_id)
```

## Orchestrators

This set of APIs can be used to manage external orchestrator inventory learning in Secure Workload cluster deployment. They require the `external_integration` capability associated with the API key.

Currently supported Orchestrator types are 'vcenter' (vCenter 6.5 and later), 'kubernetes', 'dns', 'f5', 'netscaler', 'infoblox' and 'Cisco FMC'. Supported user interface located at [External Orchestrators in Secure Workload](#).

## Orchestrator Object

The orchestrator object attributes are described below - some of the fields are applicable only for specific orchestrator types; restrictions are mentioned in the table below.

| Attribute      | Type   | Description                                                                                                                                  |
|----------------|--------|----------------------------------------------------------------------------------------------------------------------------------------------|
| id             | string | Unique identifier for the orchestrator.                                                                                                      |
| name           | string | User specified name of the orchestrator.                                                                                                     |
| type           | string | Type of orchestrator - supported values ( <i>vcenter</i> , <i>kubernetes</i> , <i>f5</i> , <i>netscaler</i> , <i>infoblox</i> , <i>dns</i> ) |
| description    | string | User specified description of the orchestrator.                                                                                              |
| username       | string | Username for the orchestration endpoint. (unnecessary for <i>dns</i> )                                                                       |
| password       | string | Password for the orchestration endpoint. (unnecessary for <i>dns</i> )                                                                       |
| certificate    | string | Client certificate used for authentication (unnecessary for <i>dns</i> )                                                                     |
| key            | string | Key corresponding to client certificate (unnecessary for <i>dns</i> )                                                                        |
| ca_certificate | string | CA Certificate to validate orchestration endpoint (unnecessary for <i>dns</i> )                                                              |
| auth_token     | string | Opaque authentication token (bearer token) (applies only for <i>kubernetes</i> )                                                             |

| Attribute                  | Type    | Description                                                                                                                                                                                                                                                                                                                                                                                                                                 |
|----------------------------|---------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| insecure                   | boolean | Turn off strict SSL verification                                                                                                                                                                                                                                                                                                                                                                                                            |
| delta_interval             | integer | Delta polling interval in seconds. Secure Workload Inventory manager will perform polling for incremental changes every <code>delta_interval</code> seconds. Note this parameter is not applicable for Infoblox and Secure Firewall Management Center.                                                                                                                                                                                      |
| full_snapshot_interval     | integer | Full snapshot interval in seconds. Secure Workload Inventory manager will perform a full refresh poll from the orchestrator.                                                                                                                                                                                                                                                                                                                |
| verbose_tsdb_metrics       | boolean | Per-Endpoint TSDB metrics                                                                                                                                                                                                                                                                                                                                                                                                                   |
| hosts_list                 | Array   | Array of { "host_name", port_number } pairs that specify how Secure Workload must connect to the orchestrator                                                                                                                                                                                                                                                                                                                               |
| use_secureconnector_tunnel | boolean | Tunnel connections to this orchestrator's hosts through the Secure Connector tunnel                                                                                                                                                                                                                                                                                                                                                         |
| route_domain               | integer | Route Domain number to poll on F5 LoadBalancers (applies only for <i>f5</i> )                                                                                                                                                                                                                                                                                                                                                               |
| dns_zones                  | Array   | Array of strings containing the DNS zones to poll from the DNS server (only for <i>dns</i> ). Each DNS Zone entry MUST end with a .                                                                                                                                                                                                                                                                                                         |
| enable_enforcement         | boolean | Applicable only for external orchestrators with policy enforcement support such as firewalls and load balancers. Examples are <i>Secure Firewall Management Center</i> , <i>F5 BIGIP</i> and <i>Citrix Netscaler</i> . This flag is false (policy enforcement is disabled) by default. If true, the external orchestrator will deploy policies to the given load balancer appliance when policy enforcement is performed for the workspace. |
| ingress_controllers        | object  | Array of <a href="#">Ingress Controller</a> objects.                                                                                                                                                                                                                                                                                                                                                                                        |



| Attribute            | Type   | Description                                                                                                                                                                                                                                                                                                                                                                                   |
|----------------------|--------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| fmc_enforcement_mode | string | Applicable only for <i>Secure Firewall Management Center external orchestrator</i> and must be either <i>merge</i> (default) or <i>override</i> . The first instance instructs Secure Firewall Management Center policy enforcer to put all Secure Workload policy rules before any existing prefilter rules, while the latter instance will remove all prefilter rules created by the users. |
| infoblox_config      | object | Applicable only for <i>Infoblox external orchestrator</i> . <a href="#">Infoblox Config</a> record type selectors.                                                                                                                                                                                                                                                                            |

## Ingress Controller

| Attribute         | Type   | Description                       |
|-------------------|--------|-----------------------------------|
| pod_selector      | object | <a href="#">Pod Selector</a>      |
| controller_config | object | <a href="#">Controller Config</a> |

## Pod Selector

| Attribute | Type   | Description                                                                         |
|-----------|--------|-------------------------------------------------------------------------------------|
| namespace | string | Namespace where the Ingress controller pod is running.                              |
| labels    | Array  | Array of {"key", "value"} pairs that specify the labels of ingress controller pods. |

## Controller Config

| Attribute     | Type   | Description                                                                |
|---------------|--------|----------------------------------------------------------------------------|
| ingress_class | string | Name of the ingress class which ingress controller satisfies.              |
| namespace     | string | Namespace is the name of the namespace which ingress controller satisfies. |
| http_ports    | Array  | Array of http ports.                                                       |

| Attribute   | Type  | Description           |
|-------------|-------|-----------------------|
| https_ports | Array | Array of https ports. |

## Infoblox Config

|                       |      |                                                                           |
|-----------------------|------|---------------------------------------------------------------------------|
| enable_network_record | bool | Default value is True. If false <i>network</i> type records are disabled. |
| enable_host_record    | bool | Default value is True. If false <i>host</i> type records are disabled.    |
| enable_a_record       | bool | Default value is True. If false <i>a</i> type records are disabled.       |
| enable_aaaa_record    | bool | Default value is True. If false <i>aaaa</i> type records are disabled.    |

\*\* Read-only status fields in the Orchestrator object \*\*

| Attribute                    | Type   | Description                                                                                                                                                                                                                                                                                 |
|------------------------------|--------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| authentication_failure       | bool   | Status of the connection to the Secure Workload Orchestrator - <i>true</i> indicates a successful connection to the orchestrator. If this field is <i>false</i> , the <i>authentication_failure_error</i> field will provide a detailed error message explaining the reason for the failure |
| authentication_failure_error | string | Detailed error message to help debug connectivity or credential failures with orchestrators                                                                                                                                                                                                 |
| scope_id                     | string | Tenant Root scope id where the inventory will be published and visible                                                                                                                                                                                                                      |

## Get Orchestrators

This endpoint returns a list of orchestrators that are known to Secure Workload appliance. This API is available to API keys with the `external_integration` capability.

```
GET /openapi/v1/orchestrator/{scope}
```

Parameters: None

Returns a list of orchestrator objects for the provided root scope. The *scope* MUST be a root scope id.

## Create Orchestrators

This endpoint is used to create orchestrators.

POST /openapi/v1/orchestrator/{scope}

### Sample python code for vCenter orchestrators

```
req_payload = {
 "name": "VCenter Orchestrator"
 "type": "vcenter",
 "hosts_list": [{ "host_name": "8.8.8.8", "port_number": 443}],
 "username": "admin",
 "password": "admin"
}
resp = restclient.post('/orchestrator/Default', json_body=json.dumps(req_payload))
```

### Sample python code for DNS orchestrators

```
req_payload = {
 "name": "DNS Server"
 "type": "dns",
 "hosts_list": [{ "host_name": "8.8.8.8", "port_number": 53}],
 "dns_zones": ["lab.corp.com.", "dev.corp.com."]
}
resp = restclient.post('/orchestrator/Default', json_body=json.dumps(req_payload))
```

### Sample python code for Kubernetes orchestrators

```
req_payload = {
 "name": "k8s"
 "type": "kubernetes",
 "hosts_list": [{ "host_name": "8.8.8.8", "port_number": 53}],
 "certificate": "",
 "key": "",
 "ca_certificate": "",
}
resp = restclient.post('/orchestrator/Default', json_body=json.dumps(req_payload))
```

### Sample python code for Kubernetes orchestrators with Ingress Controller

See information about the Kubernetes/OpenShift external orchestrator for creating authentication details.

```
req_payload = {
 "name": "k8s",
 "type": "kubernetes",
 "hosts_list": [{ "host_name": "8.8.8.8", "port_number": 53}],
 "certificate": "",
 "key": "",
 "ca_certificate": "",
 "ingress_controllers": [
 {
 "pod_selector": {
 "namespace": "ingress-nginx",
 "labels": [{"key": "app", "value": "nginx-ingress"}],
 }
 }
]
}
resp = restclient.post('/orchestrator/Default', json_body=json.dumps(req_payload))
```

### Sample python code for Kubernetes orchestrators with Multiple Ingress Controllers

See information about the Kubernetes/OpenShift external orchestrator for creating authentication details.

```
req_payload = {
 "name": "k8s",
 "type": "kubernetes",
 "hosts_list": [{ "host_name": "8.8.8.8", "port_number": 53}],
 "certificate": "",
 "key": "",
 "ca_certificate": "",
 "ingress_controllers": [
 {
 "pod_selector": {
 "namespace": "ingress-nginx",
 "labels": [{ "key": "app", "value": "nginx-ingress"}],
 },
 "controller_config": {
 "ingress_class": "nginx-class",
 }
 },
 {
 "pod_selector": {
 "namespace": "ingress-haproxy",
 "labels": [{ "key": "app", "value": "haproxy-ingress"}],
 },
 "controller_config": {
 "ingress_class": "haproxy-class",
 "http_ports": [8080],
 "https_ports": [8443],
 "namespace": "haproxy-watching-namespace"
 }
 }
],
}
resp = restclient.post('/orchestrator/Default', json_body=json.dumps(req_payload))

** Type AWS and EKS are no longer supported in external orchestrators. They have been
ported to
connectors.
```

## Get Specific Orchestrator

This endpoint returns an orchestrator instance.

```
GET /openapi/v1/orchestrator/{scope}/{orchestrator_id}
```

Returns the orchestrator object that is associated with the specified ID.

## Update an Orchestrator

This endpoint updates an orchestrator.

```
PUT /openapi/v1/orchestrator/{scope}/{orchestrator_id}
```

Parameters:

Same as POST parameters

## Delete Specific Orchestrator

This endpoint deletes the specified orchestrator.

```
DELETE /openapi/v1/orchestrator/{scope}/{orchestrator_id}
```

## Orchestrator Golden Rules

This set of APIs can be used to manage Golden Rules for external Kubernetes Orchestrators. Golden Rules are necessary to ensure Kubernetes control plane connectivity in allow list enforcement mode. They require the `external_integration` capability associated with the API key.

Currently supported Orchestrator type for Golden Rules is 'kubernetes' only. Requests to this endpoint for non- Kubernetes orchestrators will fail.

## Orchestrator Golden Rules Object

The orchestrator object attributes are described below:

| Attribute    | Type    | Description                          |
|--------------|---------|--------------------------------------|
| kubelet_port | integer | Kubelet node-local API port          |
| services     | Array   | Array of Kubernetes Services objects |

## Get Orchestrator Golden Rules

This endpoint returns the golden rules that are associated with an orchestrator. This API is available to API keys with the `external_integration` capability.

```
GET /openapi/v1/orchestrator/{scope}/{id}/gr
```

Parameters: None

Returns a single Golden Rules object

## Create or Update Golden Rules

This endpoint is used to create or update golden rules for an existing orchestrator.

```
POST /openapi/v1/orchestrator/{scope}/{id}/gr
```

Parameters:

| Attribute    | Type    | Description                          |
|--------------|---------|--------------------------------------|
| kubelet_port | integer | Kubelet node-local API port          |
| services     | Array   | Array of Kubernetes Services objects |

### Sample python code

```
req_payload = {
 "kubelet_port":10255,
 "services": [
 {
 "description": "kube-dns",
 "addresses": ["10.0.1.1:53/TCP", "10.0.1.1:53/UDP"],
 "consumed_by": ["NODES", "PODS"],
 }
]
}
resp = restclient.post('/orchestrator/{scope_id}/{orchestrator_id}/gr',
json_body=json.dumps(req_payload))
```

## FMC Orchestrator Domains

This set of APIs can be used to manage domains for external FMC Orchestrators. FMC Domains are required to enable enforcement on a given FMC Domain. They require the `external_integration` capability associated with the API key.

Currently supported Orchestrator type for FMC Domains is 'fmc' only. Requests to this endpoint for non-FMC orchestrators will fail.

## Orchestrator FMC Domains Object

The orchestrator object attributes are described below:

| Attribute   | Type  | Description                 |
|-------------|-------|-----------------------------|
| fmc_domains | Array | Array of FMC Domain objects |

FMC Domain object attributes are described below:

| Attribute           | Type    | Description                                                                                                                                                                              |
|---------------------|---------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| name                | string  | Name of FMC Domain                                                                                                                                                                       |
| enforcement_enabled | boolean | This flag is set to false by default. If true, the external orchestrator will deploy policies to the domain matching with 'name' when policy enforcement is performed for the workspace. |

URL attributes are described below:

| Attribute | Type   | Description                                                                    |
|-----------|--------|--------------------------------------------------------------------------------|
| scope     | string | Tenant Root scope name or id where the inventory will be published and visible |

| Attribute       | Type   | Description                         |
|-----------------|--------|-------------------------------------|
| orchestrator_id | string | Orchestrator ID of FMC orchestrator |

## Get FMC Domains

This endpoint returns the FMC domains that are configured on the FMC that is associated with an FMC orchestrator. This API is available to API keys with the `external_integration` capability.

```
GET /openapi/v1/orchestrator/{scope}/{orchestrator_id}/fmcdomains
```

Parameters: None

Returns a JSON object with a list of FMC Domain object attributes.

## Update FMC Domain Configuration for FMC External Orchestrator

This endpoint updates the FMC domain attributes for an existing FMC external orchestrator.

```
PUT /openapi/v1/orchestrator/{scope}/{orchestrator_id}/fmcdomains
```

Parameters:

| Attribute   | Type  | Description                 |
|-------------|-------|-----------------------------|
| fmc_domains | Array | Array of FMC Domain objects |

FMC Domain object attributes are described below:

| Attribute           | Type    | Description                                                                                                                                                                          |
|---------------------|---------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| name                | string  | Name of FMC Domain                                                                                                                                                                   |
| enforcement_enabled | boolean | This flag is set to false by default. If true, the external orchestrator deploys policies to the domain matching with 'name' when policy enforcement is performed for the workspace. |

URL attributes are described below:

| Attribute       | Type   | Description                                                                    |
|-----------------|--------|--------------------------------------------------------------------------------|
| scope           | string | Tenant Root scope name or id where the inventory will be published and visible |
| orchestrator_id | string | Orchestrator ID of FMC orchestrator                                            |

Sample python code

```

req_payload = {
 "fmc_domains": [
 {
 "enforcement_enabled": False,
 "name": "Global/Eng"
 },
 {
 "enforcement_enabled": True,
 "name": "Global/Prod"
 }
]
}
resp = restclient.put('/orchestrator/{scope}/{orchestrator_id}/fmcdomains',
json_body=json.dumps(req_payload))

```

## RBAC (Role-Based Access Control) Considerations

Access to orchestrators under a root scope requires that the API Key that is used for the request has the requisite privileges. All orchestrator API calls are scoped and always require the root scope id as part of the URL. Orchestrators always reside at the root scope level and cannot be created under subscopes. Orchestrators created (and inventory that is learned by these orchestrators) under a specific tenant root scope are invisible to other tenants.

In case of F5 load balancers that may have multiple route domains (vrf) configured, the F5 Route Domain filtering logic scans all entities on the F5 across all partitions but discards entities (services, snat pools, pools, and backends) that do not evaluate to the route domain specified in the F5 orchestrator *route\_domain* field.

## High Availability and Failover Considerations

The *hosts\_list* parameter allows configuration of multiple server addresses for an orchestrator. Secure Workload server selection logic in the case of multiple server addresses varies for each orchestrator type.

For *vCenter*, *Kubernetes*, *DNS*, *F5*, *Netscaler*, *Infoblox*, the selection is on a first healthy endpoint basis. Connections are not persistent (except for *kubernetes*) and thus, every poll period, Secure Connector Orchestrator Manager will scan the hosts and poll the first healthy endpoint encountered in the *hosts\_list*. For *kubernetes*, a persistent event channel is maintained and upon connection failure, a scan of all hosts and subsequent full poll will be performed using the next healthy endpoint.

## Kubernetes RBAC Resource Considerations

The Kubernetes client attempts to GET/LIST/WATCH the following resources.

The provided Kubernetes authentication credentials should have a minimum set of privileges to the following resources:

| Resources   | Verbs            |
|-------------|------------------|
| daemonsets  | [get list watch] |
| deployments | [get list watch] |



| Resources                         | Verbs            |
|-----------------------------------|------------------|
| endpoints                         | [get list watch] |
| namespaces                        | [get list watch] |
| nodes                             | [get list watch] |
| Pods                              | [get list watch] |
| replicasets                       | [get list watch] |
| replicationcontrollers            | [get list watch] |
| services                          | [get list watch] |
| statefulsets                      | [get list watch] |
| daemonsets.apps                   | [get list watch] |
| deployments.apps                  | [get list watch] |
| endpoints.apps                    | [get list watch] |
| namespaces.apps                   | [get list watch] |
| nodes.apps                        | [get list watch] |
| Pods.apps                         | [get list watch] |
| replicasets.apps                  | [get list watch] |
| replicationcontrollers.apps       | [get list watch] |
| services.apps                     | [get list watch] |
| statefulsets.apps                 | [get list watch] |
| daemonsets.extensions             | [get list watch] |
| deployments.extensions            | [get list watch] |
| endpoints.extensions              | [get list watch] |
| namespaces.extensions             | [get list watch] |
| nodes.extensions                  | [get list watch] |
| Pods.extensions                   | [get list watch] |
| replicasets.extensions            | [get list watch] |
| replicationcontrollers.extensions | [get list watch] |
| services.extensions               | [get list watch] |
| statefulsets.extensions           | [get list watch] |

## Service Health

This API can be used to get the health of all services that are used in the Secure Workload cluster along with their dependencies.




---

**Note** This API is only available to site admin users.

---

## Get Service Health

This endpoint returns a JSON object with service health information.

GET /openapi/v1/service\_status

Parameters: None

Response object: JSON object with service health information

### Sample Python code

```
resp = restclient.get('/service_status')
```

## Secure Connector

OpenAPI exposes the endpoints to manage the functions of the Secure Connector. These endpoints require the `external_integration` capability to be associated with the API key.




---

**Note** The Secure Connector APIs cannot be used at site level. They can only be used at the root scope level.

---

## Get Status

This endpoint returns the current status of the Secure Connector Tunnel for the specified root scope.

GET /openapi/v1/secureconnector/name/{ROOT\_SCOPE\_NAME}/status

GET /openapi/v1/secureconnector/{ROOT\_SCOPE\_ID}/status

READ permission to the specified root scope is required.

The returned status is a json object with the following schema:

| Key    | Type    | Value                                                        |
|--------|---------|--------------------------------------------------------------|
| active | boolean | A Secure Connector tunnel is currently active                |
| peer   | string  | <ip>:<port> of the Secure Connector client end of the tunnel |

| Key            | Type | Value                                                               |
|----------------|------|---------------------------------------------------------------------|
| start_time     | int  | Timestamp at which the tunnel was started (epoch time in seconds)   |
| last_heartbeat | int  | Timestamp of last heartbeat from the client (epoch time in seconds) |

## Get Token

This endpoint returns a new single-use limited-time token to be used for bootstrapping a Secure Connector client for the specified root scope.

```
GET /openapi/v1/secureconnector/name/{ROOT_SCOPE_NAME}/token
```

```
GET /openapi/v1/secureconnector/{ROOT_SCOPE_ID}/token
```

OWNER permission to the specified root scope is required.

The returned token is a string which contains a cryptographically signed token that is valid for one hour. A valid token can be used only once to bootstrap a Secure Connector client.

## Rotate Certificates

This endpoint forces the creation of a new certificate for the specified root scope. The new certificate will be used by the Secure Connector server and will be used to sign the certificate signing requests from clients for this root scope.

```
POST /openapi/v1/secureconnector/name/{ROOT_SCOPE_NAME}/rotate_certs?invalidate_old=
.<→{true|false}
```

```
POST /openapi/v1/secureconnector/{ROOT_SCOPE_ID}/rotate_certs?invalidate_old=.<→{true|false}
```

OWNER permission to the specified root scope is required.

Once this endpoint is called, communication between the client and server for this root scope will immediately transition to using the new certificate.

If *invalidate\_old* is set to false, any existing clients will automatically create a new public/private key pair and use their existing certificates to sign a new certificate for the new public key.

If *invalidate\_old* is set to true, the existing certificate will be immediately invalidated. Any existing clients will not be able to connect to the server and will have to be bootstrapped once again using a new token. See Secure Connector Deployment for more information.

# Kubernetes Vulnerability Scanning

## Get Kubernetes Registries used for Pod Vulnerability Scanning

This endpoint returns a list of all Kubernetes registries displayed in the Secure Workload cluster for a given VRF.

```
GET /openapi/kubernetes/{root_scope_name_or_id}/vulnerability_scanning/registry
```

Parameters: The JSON query body contains the following keys

| Name                  | Type   | Description           |
|-----------------------|--------|-----------------------|
| root_scope_name_or_id | string | Root scope name or ID |

Response object: Returns an array of registry objects with the following attributes

| Attribute         | Type   | Description                                            |
|-------------------|--------|--------------------------------------------------------|
| id                | string | ID of the registry                                     |
| clusters          | array  | Array of Kubernetes cluster objects using the registry |
| connection_status | string | Defines registry connection status                     |
| credential_status | string | Status stating if the credentials are provided         |
| last_scanned      | Int64  | Last scanner at time in epoch                          |
| url               | string | URL of the registry                                    |

Kubernetes cluster object

| Attribute      | Type   | Description                                                |
|----------------|--------|------------------------------------------------------------|
| id             | string | ID of the Kubernetes cluster                               |
| connector_id   | string | ID of the connector used to onboard the Kubernetes cluster |
| connector_type | string | Type of connector used to onboard the Kubernetes cluster   |
| name           | string | Name of the Kubernetes cluster                             |

Sample python code

```
root_app_scope_name = 'Tetration'
restclient.get('/kubernetes/%s/vulnerability_scanning/registry' % root_app_scope_name)
```

## Add Credentials to Kubernetes Registry

This endpoint allows you to add credentials for the Kubernetes registry. The accepted credentials are based on the registry type.

For example:

Registry type: AWS; Accepted credential type: aws\_auth Credentials object

Registry type: OTHER; Accepted credential type: basic\_auth Credentials object

```
PUT /openapi/kubernetes/{root_scope_name_or_id}/vulnerability_scanning/registry/{registry_id}
```

Parameters: The JSON query body contains the following keys:

| Name                  | Type   | Description            |
|-----------------------|--------|------------------------|
| root_scope_name_or_id | string | Root scope name or ID  |
| registry_id           | string | Kubernetes registry ID |
| registry_credential   | object | Credential object      |

Credential object

| Attribute  | Type   | Description                            |
|------------|--------|----------------------------------------|
| basic_auth | object | Basic authentication credential object |
| aws_auth   | object | AWS authentication credential object   |
| azure_auth | object | Azure authentication credential object |
| gcp_auth   | object | GCP authentication credential object   |

basic\_auth object

| Attribute | Type   | Description |
|-----------|--------|-------------|
| username  | string | Username    |
| password  | string | Password    |

aws\_auth object

| Attribute             | Type   | Description                   |
|-----------------------|--------|-------------------------------|
| aws_access_key_id     | string | AWS credentials access key    |
| aws_secret_access_key | string | AWS credentials access secret |

azure\_auth object:

| Attribute           | Type   | Description         |
|---------------------|--------|---------------------|
| azure_tenant_id     | string | Azure tenant ID     |
| azure_client_id     | string | Azure client ID     |
| azure_client_secret | string | Azure client secret |

gcp\_auth object:

| Attribute           | Type   | Description         |
|---------------------|--------|---------------------|
| gcp_service_account | object | GCP service account |

### Sample python code

```

root_app_scope_name = 'Tetration'
registry_id = '64cdc7a7362f57192dcc1625'
pay_load = {
 "registry_credential": {
 "basic_auth": {
 "username": "username",
 "password": "password",
 }
 }
}
restclient.put('/kubernetes/%s/vulnerability_scanning/registry/%s' % root_app_scope_name,
registry_id, json_body=json.dumps(pay_load))

```

## Get Kubernetes Pod Scanners

This endpoint returns a list of all Kubernetes pod scanners displayed in the Secure Workload cluster for a given VRF.

GET /openapi/kubernetes/{root\_scope\_name\_or\_id}/vulnerability\_scanning/scanner

Parameters: The JSON query body contains the following keys

| Name                  | Type   | Description           |
|-----------------------|--------|-----------------------|
| root_scope_name_or_id | string | Root scope name or ID |

Response object: Returns an array of registry objects with the following attributes

| Attribute          | Type   | Description                                            |
|--------------------|--------|--------------------------------------------------------|
| id                 | string | ID of the scanner                                      |
| kubernetes_cluster | object | <a href="#">Add Credentials to Kubernetes Registry</a> |
| health_status      | string | Defines scanner health state                           |
| health_object      | string | Health status object                                   |
| scanner_action     | string | Notifies if the scanner is ENABLED or DISABLED         |
| name               | string | Name of the scanner                                    |

Health object

| Attribute    | Type   | Description                    |
|--------------|--------|--------------------------------|
| last_checkin | string | Last reported at time in epoch |

| Attribute           | Type   | Description                                          |
|---------------------|--------|------------------------------------------------------|
| scanner_sensor_name | string | Kubernetes node name on which the scanner is running |
| scanner_sensor_uuid | string | ID of the agent running on the Kubernetes node       |
| status              | string | Notifies if health is reported by the scanner        |

### Sample python code

```
root_app_scope_name = 'Tetration'
restclient.get('/kubernetes/%s/vulnerability_scanning/scanner % root_app_scope_name)
```

## Edit Scanner Filter Query and Action

This endpoint allows you to edit Kubernetes scanner filter query and action.

```
PUT /openapi/kubernetes/{root_scope_name_or_id}/vulnerability_scanning/scanner/{scanner_id}
```

Parameters: The JSON query body contains the following keys

| Name             | Type   | Description                                                                                   |
|------------------|--------|-----------------------------------------------------------------------------------------------|
| rootAppScopeName | string | Root scope name or ID                                                                         |
| scanner_id       | string | Kubernetes scanner ID                                                                         |
| scanner_action   | string | To enable or disable the scanner. Expected values are ENABLED or DISABLED                     |
| filter_query     | object | <a href="#">Validate an inventory filter query</a> to filter pods for vulnerability scanning. |

### Sample python code

```
root_app_scope_name = 'Tetration'
scanner_id = '64cdc7a7362f57192dcc1625'
payload = {
 "scanner_action": "ENABLED"
 "filter_query": {
 "type": "contains",
 "field": "user_orchestrator_system/pod_name",
 "value": "pod"
 }
}
restclient.put('/kubernetes/%s/vulnerability_scanning/scanner/%s' % root_app_scope_name,
scanner_id, json_body=json.dumps(payload))
```

# Policy Enforcement Status for External Orchestrators

This set of APIs is used to provide policy enforcement status for load balancer external orchestrators such as *F5 BIG-IP* or *Citrix Netscaler*.



**Note** To use these APIs, you must have access to the scope attached to the VRF.

## Get Policy Enforcement Status for All External Orchestrators

This endpoint returns policy enforcement status for all external orchestrators belonging to the given VRF. This API is available to API keys with `external_integration` capability.

GET /openapi/v1/tnp\_policy\_status/{vrfID}

Parameters: The request URL contains the following parameters

| Name  | Type    | Description                |
|-------|---------|----------------------------|
| vrfID | integer | VRF ID for the root scope. |

Response object: Returns a list of network policies with the Status as ENFORCED OR FAILED OR IGNORED.

### Sample python code

```
vrf_id = 676767
restclient.get('/tnp_policy_status/%d' % vrf_id)
```

## Get Policy Enforcement Status for an External Orchestrator

This endpoint returns policy enforcement status for an external orchestrator belonging to the given VRF. This API is available to API keys with `external_integration` capability.

GET /openapi/v1/tnp\_policy\_status/{vrfID}/{orchestratorID}

Parameters: The request URL contains the following parameters

| Name           | Type    | Description                |
|----------------|---------|----------------------------|
| vrfID          | integer | VRF ID for the root scope. |
| orchestratorID | string  | External orchestrator ID.  |

Response object: Returns a list of network policies with the Status as ENFORCED OR FAILED OR IGNORED.

### Sample python code

```
vrf_id = 676767
orchestrator_id = '5ee3c991497d4f3b00f1ee07'
restclient.get('/tnp_policy_status/%d/%s' % (vrf_id, orchestrator_id))
```



# Download Certificates for Managed Data Taps and Datasinks

This set of APIs is used to download the certificates for the Managed Data Taps and Datasinks.



**Note** To use these APIs, you must have access to the scope attached to the VRF.

## Get List of Managed Data Taps for a Given VRF ID.

This endpoint returns a list of Managed Data Taps in a given VRF. This API is available to API keys with `external_integration` capability.

GET `/openapi/v1/mdt/{vrfID}`

Parameters: None

Returns a list of Managed Data Taps with attributes like Managed Data Tap ID.

## Download Managed Data Tap Certificates for a Given MDT ID

This endpoint is used to download the certificates for a given Managed Data Tap ID. The MDT ID can be obtained by using `/openapi/v1/mdt/{vrfID}` endpoint as explained in the above documentation. This API is available to API keys with `external_integration` capability.

GET `/openapi/v1/mdt/{vrfID}/{mdtID}/certs`

Parameters:

| Name   | Type   | Description                                                                                        |
|--------|--------|----------------------------------------------------------------------------------------------------|
| format | string | The keystore and trust store format. Values: <code>jks</code> (default value) or <code>cert</code> |

Returns a tar.gz file which contains the following files:-

For `jks` format: **truststore.jks**, **topic.txt**, **passphrase.txt**, **keystone.jks**, **kafkaBrokerIps.txt**, **consumer\_name.txt**, **consumer\_group\_id.txt**.

For `cert` format: **KafkaConsumerCA.cert**, **KafkaConsumerPrivateKey.key**, **kafkaCA.cert**, **kafkaBrokerIps.txt**, **topic.txt**

**KafkaConsumerCA.cert** is the Public certificate file and **KafkaConsumerPrivateKey.key** file has the private key. **kafkaCA.cert** has the CA certificate and **kafkaBrokerIps.txt** has the list of the Kafka brokers IP Addresses and Ports. **topic.txt** file has the name of the topic which should be used to fetch data from MDT. **truststore.jks** and **keystone.jks** are Java keystore files.

## Get List of DataSinks for a Given VRF ID

This endpoint returns a list of DataSinks in a given VRF. This API is available to API keys with `external_integration` capability.

```
GET /openapi/v1/datasinks/{vrfID}
```

Parameters: None

Returns a list of DataSinks with attributes like DataSink ID.

## Download DataSink Certificates for a Given DataSink ID

This endpoint is used to download the certificates for a given DataSink ID. The DataSink ID can be obtained by using `/openapi/v1/datasinks/{vrfID}` endpoint as explained in the above documentation. This API is available to API keys with `external_integration` capability.

```
GET /openapi/v1/datasinks/{vrfID}/{dsID}/certs
```

Parameters: None

Returns a tar.gz file which contains the following files:- **userCA.cert**, **userPrivateKey.key**, **intermediateCA.cert**, **kafkaCA.cert**, **kafkaBrokerIps.txt**, **topic.txt**.

**userCA.cert** is the Public certificate file and **KafkaConsumerPrivateKey.key** file has the private key. **intermediateCA.cert** and **kafkaCA.cert** has the CA certificate for intermediate and root CA respectively. **kafkaBrokerIps.txt** has the list of the Kafka brokers IP Addresses and Ports. **topic.txt** file has the name of the topic which should be used to fetch data from datasink.

## Change Logs

This API provides read access to change log items. This API requires the `user_role_scope_management` capability associated with the API key.



**Note** This API is only available to site admins and owners of root scopes.

## Change Log Object

The descriptions of the change log object attributes are as follows:

| Attribute         | Type             | Description                                                |
|-------------------|------------------|------------------------------------------------------------|
| id                | string           | Unique identifier for the change log item.                 |
| association_chain | array of objects | List of names and ids associated with this change.         |
| scope             | string           | Scope of change (not the same as a Secure Workload scope). |
| action            | string           | Change action.                                             |
| details           | string           | Further action details, when available.                    |

| Attribute  | Type    | Description                                         |
|------------|---------|-----------------------------------------------------|
| created_at | integer | Unix timestamp of when change log item was created. |
| modifier   | object  | User responsible for change.                        |
| modified   | object  | Modified fields and values.                         |
| original   | object  | Fields and values before modification.              |
| version    | integer | Version identifier.                                 |

## Search

This endpoint returns the list of change log items matching the specified criteria.

GET /openapi/v1/change\_logs

Parameters: The request URL contains the following parameters

| Name              | Type    | Description                                                                                |
|-------------------|---------|--------------------------------------------------------------------------------------------|
| root_app_scope_id | string  | (optional) Required for root scope owners. Filter results by root scope.                   |
| association_name  | string  | (optional) Required for root scope owners. The item type to return. For example: "H4Users" |
| history_action    | string  | (optional) Change action. For example: "update"                                            |
| details           | string  | (optional) Action details. For example: "soft-delete"                                      |
| before_epoch      | integer | (optional) Include results created before this unix timestamp.                             |
| after_epoch       | integer | (optional) Include results created after this unix timestamp.                              |
| offset            | integer | (optional) Number of results to skip.                                                      |
| limit             | integer | (optional) Limit number of results, default 1000.                                          |

Response object: Returns a list of change log objects.

### Response

The response is a JSON object in the body with the following properties.

| Name        | Type             | Description                                                     |
|-------------|------------------|-----------------------------------------------------------------|
| total_count | integer          | Total number of items matching before applying offset or limit. |
| items       | array of objects | List of results.                                                |

### Sample python code

Fetch last 100 scope object changes within a given root scope within the last day.

```
root_app_scope_id = '5ce480db497d4f1ca1fc2b2b'
one_day_ago = int(time.time() - 24*60*60)
resp = restclient.get('/change_logs', params={'root_app_scope_id': root_app_scope_id,
 'association_name': 'AppScope',
 'after_epoch': one_day_ago,
 'limit': 100})
```

Fetch the second thousand scope object changes.

```
root_app_scope_id = '5ce480db497d4f1ca1fc2b2b'
resp = restclient.get('/change_logs', params={'root_app_scope_id': root_app_scope_id,
 'association_name': 'AppScope',
 'offset': 1000})
```

Further refine these results to only show new scope creations.

```
root_app_scope_id = '5ce480db497d4f1ca1fc2b2b'
one_day_ago = int(time.time() - 24 * 60 * 60)
resp = restclient.get('/change_logs', params={'root_app_scope_id': root_app_scope_id,
 'association_name': 'AppScope',
 'history_action': 'create',
 'after_epoch': one_day_ago,
 'limit': 100})
```

A site admin could use limit and offset to iteratively fetch all changes across all scopes.

```
resp = restclient.get('/change_logs', params={'offset': 100, 'limit': 100})
```

## Non-Routable Endpoints

The following APIs are used to manage nonroutable endpoints, to mark an IP or subnet as nonroutable or get a list of nonroutable endpoints that are marked by a user or to unmark an IP or subnet as nonroutable endpoint. The `user_data_upload` capability that is associated with the API key is required.

## Non-Routable Endpoint Object

The descriptions of the Non-Routable Endpoint Object attributes are as follows:

| Attribute | Type   | Description                                      |
|-----------|--------|--------------------------------------------------|
| id        | string | Unique identifier for the nonroutable endpoint.  |
| name      | string | User specified name of the nonroutable endpoint. |
| subnet    | string | IPv4/IPv6 subnet.                                |

| Attribute    | Type   | Description                                                 |
|--------------|--------|-------------------------------------------------------------|
| vrf_id       | long   | ID of the VRF to which the nonroutable endpoint belongs to. |
| address_type | string | IPV4/IPV6 based on subnet address type                      |
| host_uuid    | string | Unique ID of the agent                                      |
| description. | string | User specified description of the nonroutable endpoint.     |

## GET Non-Routable Endpoints

This endpoint returns a list of nonroutable endpoints in the given tenant.

```
GET /openapi/v1/non_routable_endpoints/{rootScopeName}
```

Parameters: None

## Create a Non-Routable Endpoint

This endpoint is used to create a nonroutable endpoint.

```
POST /openapi/v1/non_routable_endpoints/{rootScopeName}
```

Parameters:

| Attribute              | Type   | Description                                             |
|------------------------|--------|---------------------------------------------------------|
| name                   | string | User specified name of the nonroutable endpoint.        |
| subnet                 | string | IPv4 or IPv6 subnet.                                    |
| address_type(optional) | string | IPv4or IPv6 based on subnet address type                |
| host_uuid(optional)    | string | Unique ID of the agent                                  |
| description(optional)  | string | User specified description of the nonroutable endpoint. |

\*if optional fields are not specified, null values are populated.

### Sample python code

```
req_payload = {
 "name": "nre-1",
 "subnet": "1.1.1.1/30",
 "address_type": IPV4,
 "description": "sample parameters test"
}
```

```
resp = restclient.post('/openapi/v1/non_routable_endpoints/Default',
json_body=json.dumps(req_payload))
```

## GET Specific Non-Routable Endpoints with Name

This endpoint returns a nonroutable endpoint for the specified name.

```
GET /openapi/v1/non_routable_endpoints/{rootScopeName}/name/{name}
```

Parameters: None

## GET Specific Non-Routable Endpoints with ID

This endpoint returns a nonroutable endpoint for the specified ID.

```
GET /openapi/v1/non_routable_endpoints/{rootScopeName}/id/{id}
```

Parameters: None

## Update Specific Non-Routable Endpoint Name

This endpoint is used to update a nonroutable endpoint. It uses either an ID or a name of the existing nonroutable endpoint to update its name.

```
PUT /openapi/v1/non_routable_endpoints/{rootScopeName}
```

Parameters:

| Attribute | Type   | Description                                      |
|-----------|--------|--------------------------------------------------|
| id        | string | Unique identifier for the nonroutable endpoint.  |
| name      | string | User specified name of the nonroutable endpoint. |
| new_name  | string | New name to update                               |

### Sample python code

```
req_payload = {
 "name": "nre-1",
 "new_name": "nre-updated",
}
resp = restclient.put('/openapi/v1/non_routable_endpoints/Default',
json_body=json.dumps(req_payload))
```

```
req_payload = {
 "id": "5f706964a5b5f16ed4b0aacb",
 "new_name": "nre-updated",
}
resp = restclient.put('/openapi/v1/non_routable_endpoints/Default',
json_body=json.dumps(req_payload))
```

## Delete Specific Non-Routable Endpoint with Name

This endpoint deletes the specific nonroutable endpoint.

```
DELETE /openapi/v1/non_routable_endpoints/{rootScopeName}/name/{name}
```

## Delete Specific Non-Routable Endpoint with ID

This endpoint deletes the specific nonroutable endpoint.

```
DELETE /openapi/v1/non_routable_endpoints/{rootScopeName}/id/{id}
```

# Config and Command Schemas for External Appliances and Connectors

## Config Groups APIs

The Config Groups APIs provides config schemas for the Appliances and Connectors APIs. These APIs require the `sensor_management` or `external_integration` capability that is associated with the API key.

### API to Get the Schema of Config

This endpoint returns a static config schema for selected type/groups of configs.

```
GET /openapi/v1/config_groups/schema/<type>
```

where <type> is the Appliance config type.

Parameters: The request URL contains the following parameters

| Name | Type   | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
|------|--------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| type | string | Specify the config type from "VM1" "VM3" "NTP" "LOG" "LDAP" "NETFLOW" "IPFIX" "NETSCALER" "F5" "AWS" "ENDPOINT" "SLACK_NOTIFIER" "GCP_CONNECTOR" "PAGERDUTY_NOTIFIER" "SYSLOG_NOTIFIER" "KINESIS_NOTIFIER" "EMAIL_NOTIFIER" "ISE" "MERAKI" "SLACK_NOTIFIER_OVERRIDE" "PAGERDUTY_NOTIFIER_OVERRIDE" "SYSLOG_NOTIFIER_OVERRIDE" "KINESIS_NOTIFIER_OVERRIDE" "AZURE_CONNECTOR" "EMAIL_NOTIFIER_OVERRIDE" "SYSLOG_SEVERITY_MAPPING" "SERVICENOW" "SYNC_INTERVAL" "ALERT" "VM3_ERSPAN" "AWS_CONNECTOR" "VM0" |

Response object: Returns the config schema for selected config type.

**Sample response**

```

resp = restclient.get('/config_groups/schema/LOG')
if resp.status_code == 200:
 parsed_resp = json.loads(resp.content)
 print json.dumps(parsed_resp)

```

**Sample response**

```

{
 "type": "LOG",
 "name": "Log",
 "mode": "TEST",
 "config": {
 "secured": {},
 "unsecured": {
 "log-level": "info",
 "max-log-size": 10,
 "max-log-age": 30,
 "max-log-backups": 20 }
 },
 "fill_ins": [
 {
 "field": "log-level",
 "label": "Logging Level",
 "placeholder": "info",
 "type": "user_fill_in",
 "input_type": "dropdown",
 "possible_values": [
 "trace",
 "debug",
 "info",
 "warn",
 "error"
]
 },
 {
 "field": "max-log-size",
 "label": "Max Log File Size (in MB)",
 "placeholder": 10,
 "type": "user_fill_in",
 "input_type": "number",
 "min": 1,
 "max": 10240
 },
 {
 "field": "max-log-age",
 "label": "Log Rotation (in days)",
 "placeholder": 30,
 "type": "user_fill_in",
 "input_type": "number",
 "min": 1,
 "max": 365
 },
 {
 "field": "max-log-backups",
 "label": "Log Rotation (in instances)",
 "placeholder": 20,
 "type": "user_fill_in",
 "input_type": "number",
 "min": 1,
 "max": 100
 }
]
}

```



## API to Get the Schema of Troubleshooting Commands

This endpoint returns a static troubleshooting command config schema for selected type of troubleshooting command.

GET /openapi/v1/config\_groups/command\_schema/<type>

In request payload, <type> is the troubleshooting command config type.

Parameters:

The request URL contains the following parameters

| Name | Type   | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
|------|--------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| type | string | Specify the command type from "SHOW LOG"<br>"SHOW_SERVICE_LOG"<br>"SHOW_RUNNING_CONF"<br>"SHOW_SERVICE_RUNNING_CONF"<br>"SHOW_SYS_COMMANDS"<br>"SHOW_DOCKER_COMMANDS"<br>"SHOW_DOCKER_INSTANCE_COMMANDS"<br>"OPER_DOCKER_INSTANCE_COMMANDS"<br>"SHOW_SUPERVISOR_COMMANDS"<br>"SHOW_SUPERVISOR_SERVICE_COMMANDS"<br>"OPER_SUPERVISOR_SERVICE_COMMANDS"<br>"NETWORK_CONNECTIVITY_COMMANDS"<br>"LIST_FILES" "LIST_SERVICE_FILES"<br>"PACKET_CAPTURE"<br>"SHOW_DATA_EXPORT_LGO"<br>"SHOW_DATAEXPORT_RUNNING_CONF"<br>"SHOW_DATA_EXPORT_SYS_COMMANDS"<br>"UPDATE_LISTENING_PORT"<br>"UPDATE_TAN_LOG_CONF"<br>"SNAPSHOT_APPLIANCE"<br>"SNAPSHOT_CONNECTOR"<br>"SHOW_AWS_DOWNLOADER_LOG"<br>"CONTROLLER_PROFILING"<br>"SERVICE_PROFILING"<br>"RESTART_CONNECTOR_CONTAINER"<br>"RESTART_CONNECTOR_SERVICE"<br>"CONNECTOR_ALERT_INTERVAL_APPLIANCE"<br>"CONNECTOR_ALERT_INTERVAL_CONNECTOR"<br>"EXEC_SCRIPT"<br>"SHOW_SEGMENTATION_POLICIES" |

Response object: Returns the config schema for selected config type.

### Sample response

```
resp = restclient.get('/config_groups/command_schema/SHOW_LOG')
if resp.status_code == 200:
 parsed_resp = json.loads(resp.content)
 print json.dumps(parsed_resp)
```

### Sample response

```

{
 "name": "Show logs",
 "desc": "Show the contents of a log file",
 "long_desc": "Show the contents of a log file and optionally grep the file for a specified
pattern. The output is tailed for the last 5000 lines.",
 "valid_appliances": [
 "TETRATION_DATA_INGEST",
 "TETRATION_EDGE",
 "TETRATION_EXPORT"
],
 "valid_connectors": [
 "netflow",
 "netscaler",
 "f5",
 "aws",
 "anyconnect",
 "slack",
 "kinesis",
 "syslog",
 "email",
 "pagerduty",
 "ise",
 "asa",
 "meraki",
 "servicenow",
 "wad"
],
 "arg_fillins": [
 {
 "field": "pattern",
 "label": "Grep Pattern",
 "input_type": "text",
 "optional": true
 }
],
 "output_type": "FILE",
 "output_ext": "LOG"
}

```

## External Appliances

### External Appliances APIs

The External Appliances APIs are associated with managing SecureWorkload external Appliances. These set of APIs require the `sensor_management` or `external_integration` capability associated with the API key.

#### API to Get List of Appliances

This endpoint returns the list of Appliances.

```
GET /openapi/v1/ext_appliances?root_scope_id=<root_scope_id>&type=<type>
```

where `<root_scope_id>` is the `root_scope_id` that can be obtained from the [Get scopes](#) API, `<type>` is a string to decide Appliance type.

Parameters: The request URL contains the following parameters

| Name                       | Type   | Description            |
|----------------------------|--------|------------------------|
| <code>root_scope_id</code> | string | Specify the root scope |

| Name | Type   | Description                                                                                                                                                   |
|------|--------|---------------------------------------------------------------------------------------------------------------------------------------------------------------|
| type | string | Specify the Appliance type, the value can be either “TETRATION_EDGE”, “TETRATION_DATA_INGEST”, “TETRATION_EXPORT”, “TETRATION_ERSPAN” or “TETRATION_INTERNAL” |

Response object: Returns the list of Appliance.

### Sample response

```
resp =
restclient.get('/ext_appliances?root_scope_id=63bf8d2f497d4f7287dbd335&type=TETRATION_INTERNAL')

if resp.status_code == 200:
 parsed_resp = json.loads(resp.content)
 print json.dumps(parsed_resp)
```

### Sample response

```
[
 {
 "root_scope_id": "63bf8d2f497d4f7287dbd335",
 "type": "tetration_internal",
 "status": {
 "state": "active",
 "controller_state": "up",
 "message": "",
 "display_state": "active"
 },
 "auto_upgrade": true,
 "created_at": 1673498141,
 "updated_at": 1673498141,
 "registered_at": 1673498141,
 "last_checkin_at": 0,
 "last_rpm_sent_at": 0,
 "upgrade_attempts": 0,
 "delete_attempts": 0,
 "last_delete_msg_sent_at": 0,
 "taas": false,
 "deleted": false,
 "deleted_at": 0,
 "connector_limit": 5000,
 "available_slots": 5000,
 "internal": true,
 "id": "63bf8e1d6419d06bef39bc85",
 "ha_peer_appliance_id": "",
 "display_type": "Tetration Internal"
 }
]
```

## API to Create an Appliance

This endpoint creates an Appliance.

POST /openapi/v1/ext\_appliances

In request payload, to obtain <config> schema, select one of the “valid\_config” from [API to Get Appliance Schema, on page 922](#) response, apply the “valid\_config” to [API to Get the Schema of Config, on page 909](#).

Parameters: The request URL contains the following parameters

| Name          | Type    | Description                                                                                                                                                   |
|---------------|---------|---------------------------------------------------------------------------------------------------------------------------------------------------------------|
| name          | string  | Specify the name                                                                                                                                              |
| root_scope_id | string  | Specify the root scope                                                                                                                                        |
| type          | string  | Specify the Appliance type, the value can be either “TETRATION_EDGE”, “TETRATION_DATA_INGEST”, “TETRATION_EXPORT”, “TETRATION_ERSPAN” or “TETRATION_INTERNAL” |
| config        | set     | Provide the filled config schema in JSON format                                                                                                               |
| taas          | boolean | Indicate if the Appliance is for TAAS env                                                                                                                     |
| version       | string  | Specify the version                                                                                                                                           |

Response object: Returns the created Appliance.

### Sample response

```
req_payload = {
 "name": "Data Ingest Appliance",
 "type": "tetration_data_ingest",
 "root_scope_id": "63c41ff2497d4f5f5be73662",
 "config": {
 "VM3": {
 "secured": {},
 "unsecured": {
 "cidr": [
 "172.26.231.141/23",
 "172.26.231.142/23",
 "172.26.231.143/23"
],
 "gateway": [
 "172.26.231.140",
 "172.26.231.140",
 "172.26.231.140"
],
 "cidr_v6": [],
 "gateway_v6": [],
 "dns": [
 "testserver"
],
 "search_domains": [],
 "hostname": "",
 "use_proxy_for_tetration": false,
 "https_proxy": "",
 "no_proxy": [],
 "docker_subnet": ""
 }
 }
 }
}
```

```

}
resp = restclient.post('/ext_appliances', json_body=json.dumps(req_payload))
if resp.status_code == 200:
 parsed_resp = json.loads(resp.content)
 print json.dumps(parsed_resp)

```

### Sample response

```

{
 "root_scope_id":"63c41ff2497d4f5f5be73662",
 "type":"tetration_data_ingest",
 "status":{
 "state":"pending_dio",
 "controller_state":"down",
 "message":"Setting up appliance",
 "display_state":"preparing"
 },
 "auto_upgrade":true,
 "created_at":1674183549,
 "updated_at":1674183549,
 "registered_at":0,
 "last_checkin_at":0,
 "last_rpm_sent_at":0,
 "upgrade_attempts":0,
 "delete_attempts":0,
 "last_delete_msg_sent_at":0,
 "name":"Data Ingest Appliance",
 "taas":false,
 "deleted":false,
 "deleted_at":0,
 "connector_limit":3,
 "available_slots":0,
 "internal":false,
 "id":"63ca037dbca44e263daeb5d0",
 "ha_peer_appliance_id":"",
 "display_type":"Tetration Data Ingest"
}

```

## API to Delete an Appliance

This endpoint deletes the given Appliance.

```
DELETE /openapi/v1/ext_appliances/<id>
```

where <id> is the appliance\_id that can be obtained from the [API to Get List of Appliances, on page 912](#).

Parameters: The request URL contains the following parameters

| Name | Type   | Description              |
|------|--------|--------------------------|
| id   | string | Specify the Appliance ID |

Response object: Returns the status of the deleted Appliance.

### Sample response

```

resp = restclient.delete('/ext_appliances/63be3b1ade36423c12bff6e1')
if resp.status_code == 200:
 parsed_resp = json.loads(resp.content)
 print json.dumps(parsed_resp)

```

### Sample response

```

{
 "status": 200,

```

```

 "code": 1000,
 "message": "deleted"
 }

```

## API to Get an Appliance by ID

This endpoint gets the Appliance with given ID.

```
GET /openapi/v1/ext_appliances/<id>
```

where <id> is the appliance\_id that can be obtained from the [API to Get List of Appliances, on page 912](#).

Parameters: The request URL contains the following parameters

| Name | Type   | Description              |
|------|--------|--------------------------|
| id   | string | Specify the Appliance ID |

Response object: Returns the Appliance with given ID.

### Sample response

```

resp = restclient.get('/ext_appliances/63bf8e1d6419d06bef39bc85')
if resp.status_code == 200:
 parsed_resp = json.loads(resp.content)
 print json.dumps(parsed_resp)

```

### Sample response

```

{
 "root_scope_id": "63bf8d2f497d4f7287dbd335",
 "type": "tetration_internal",
 "status": {
 "state": "active",
 "controller_state": "up",
 "message": "",
 "display_state": "active"
 },
 "auto_upgrade": true,
 "created_at": 1673498141,
 "updated_at": 1673498141,
 "registered_at": 1673498141,
 "last_checkin_at": 0,
 "last_rpm_sent_at": 0,
 "upgrade_attempts": 0,
 "delete_attempts": 0,
 "last_delete_msg_sent_at": 0,
 "taas": false,
 "deleted": false,
 "deleted_at": 0,
 "connector_limit": 5000,
 "available_slots": 5000,
 "internal": true,
 "id": "63bf8e1d6419d06bef39bc85",
 "ha_peer_appliance_id": "",
 "display_type": "Tetration Internal"
}

```

## API to Rename an Appliance

This endpoint renames the Appliance with given ID.

```
PUT /openapi/v1/ext_appliances/<id>
```

where <id> is the appliance\_id that can be obtained from the [API to Get List of Appliances, on page 912](#).

Parameters: The request URL contains the following parameters

| Name | Type   | Description                           |
|------|--------|---------------------------------------|
| id   | string | Specify the Appliance ID              |
| name | string | Specify the new name of the Appliance |

Response object: Returns the Appliance with given ID and the new name.

### Sample response

```
req_payload = {
 "name": "new_name",
}
resp = restclient.put('/ext_appliances/63bf8e1d6419d06bef39bc85',
json_body=json.dumps(req_payload))
if resp.status_code == 200:
 parsed_resp = json.loads(resp.content)
 print json.dumps(parsed_resp)
```

### Sample response

```
{
 "root_scope_id": "63bf8d2f497d4f7287dbd335",
 "type": "tetration_internal",
 "status": {
 "state": "active",
 "controller_state": "up",
 "message": "",
 "display_state": "active"
 },
 "auto_upgrade": true,
 "created_at": 1673498141,
 "updated_at": 1673498141,
 "registered_at": 1673498141,
 "last_checkin_at": 0,
 "last_rpm_sent_at": 0,
 "upgrade_attempts": 0,
 "delete_attempts": 0,
 "last_delete_msg_sent_at": 0,
 "name": "new_name",
 "taas": false,
 "deleted": false,
 "deleted_at": 0,
 "connector_limit": 5000,
 "available_slots": 5000,
 "internal": true,
 "id": "63bf8e1d6419d06bef39bc85",
 "ha_peer_appliance_id": "",
 "display_type": "Tetration Internal"
}
```

## API to Get the Configs on Config Type

This endpoint gets the configs of the Appliance with given ID and config type.

```
GET /openapi/v1/ext_appliances/<id>/config?config_type=<config_type>
```

where <id> is the appliance\_id that can be obtained from [API to Get List of Appliances, on page 912](#), <config\_type> is the "valid\_config" that can be obtained from [API to Get Appliance Schema, on page 922](#).

Parameters: The request URL contains the following parameters

| Name        | Type   | Description                                                                                                                                     |
|-------------|--------|-------------------------------------------------------------------------------------------------------------------------------------------------|
| id          | string | Specify the Appliance ID                                                                                                                        |
| config_type | string | Specify the config type. Refer to <a href="#">API to Get the Schema of Config, on page 909</a> for all possible values listed under Description |

Response object: Returns the configs with given Appliance ID and config type

### Sample response

```
resp =
restclient.get('/ext_appliances/63c1272039042a1c0ddd3e20/config?config_type=VM3_ERSPAN')
if resp.status_code == 200:
 parsed_resp = json.loads(resp.content)
 print json.dumps(parsed_resp)
```

### Sample response

```
[
 {
 "mode": "TEST_AND_APPLY",
 "name": "VM3_ERSPAN",
 "root_scope_id": "63bf8d2f497d4f7287dbd335",
 "vrf_id": 1,
 "appliance_id": "63c1272039042a1c0ddd3e20",
 "deleted": false,
 "type": "VM3_ERSPAN",
 "state": "COMMIT",
 "attempts": 0,
 "config": {
 "secured": {},
 "unsecured": {
 "cidr": [
 "172.26.231.141/23",
 "172.26.231.142/23",
 "172.26.231.143/23"
],
 "gateway": [
 "172.26.231.140",
 "172.26.231.140",
 "172.26.231.140"
],
 "cidr_v6": [],
 "gateway_v6": [],
 "dns": [
 "test"
],
 "search_domains": [],
 "hostname": "hjttest",
 "https_proxy": "",
 "no_proxy": []
 }
 },
 "push_to_dio_at": 0,
 "created_at": 1673602848,
```



```

 "updated_at": 1673602848,
 "discovery_completed_at": 0,
 "committed_at": 0,
 "delete_at": 0,
 "error_at": 0,
 "hidden": false,
 "discovery_phase": 0,
 "internal": false,
 "id": "63c1272039042a1c0ddd3e21"
 }
]

```

## API to Add a New Config to External Appliance

This endpoint adds a new config to the Appliance with given ID

POST /openapi/v1/ext\_appliances/<id>/config

where <id> is the appliance\_id that can be obtained from the [API to Get List of Appliances, on page 912](#). In request payload, <type> is the "valid\_config" that can be obtained from [API to Get Appliance Schema, on page 922](#). To obtain <config> schema, select one of the "valid\_config" from [API to Get Appliance Schema, on page 922](#) response, apply the "valid\_config" to [API to Get the Schema of Config, on page 909](#).

Parameters: The request URL contains the following parameters

| Name   | Type   | Description                                                                                                                                     |
|--------|--------|-------------------------------------------------------------------------------------------------------------------------------------------------|
| name   | string | Specify the config name                                                                                                                         |
| type   | string | Specify the config type. Refer to <a href="#">API to Get the Schema of Config, on page 909</a> for all possible values listed under Description |
| config | set    | Provide the filled config schema in JSON format                                                                                                 |

Response object: Returns the updated config.

### Sample response

```

req_payload = {
 "name": "new_config",
 "type": "VM3_ERSPAN",
 "config": {}
}
resp = restclient.post('/ext_appliances/63c1272039042a1c0ddd3e20/config',
json_body=json.dumps(req_payload))
if resp.status_code == 200:
 parsed_resp = json.loads(resp.content)
 print json.dumps(parsed_resp)

```

### Sample response

```

{
 "prev_id": "63c1272039042a1c0ddd3e21",
 "mode": "TEST_AND_APPLY",
 "name": "new_config",
 "root_scope_id": "63bf8d2f497d4f7287dbd335",
 "vrf_id": 1,
 "appliance_id": "63c1272039042a1c0ddd3e20",
 "deleted": false,

```

```

 "type": "VM3_ERSPAN",
 "state": "COMMIT",
 "attempts": 0,
 "config": {
 "secured": {},
 "unsecured": null
 },
 "push_to_dio_at": 0,
 "created_at": 1673661042,
 "updated_at": 1673661042,
 "discovery_completed_at": 0,
 "committed_at": 0,
 "delete_at": 0,
 "error_at": 0,
 "hidden": false,
 "discovery_phase": 0,
 "internal": false,
 "id": "63c20a7239042a0991b871b7"
 }
}

```

## API to Delete a Config

This endpoint deletes a config from given Appliance.

```
DELETE /openapi/v1/ext_appliances/<id>/config/<config_id>
```

where <id> is the appliance\_id that can be obtained from the [API to Get List of Appliances, on page 912](#), <config\_id> is the id that can be obtained from [API to Get the Configs on Config Type, on page 917](#).

Parameters: The request URL contains the following parameters

| Name      | Type   | Description              |
|-----------|--------|--------------------------|
| id        | string | Specify the Appliance ID |
| config_id | string | Specify the config ID    |

Response object: Returns the status of the config deleted for given Appliance.

### Sample response

```

resp =
restclient.delete('/ext_appliances/63c1272039042a1c0ddd3e20/config/63c1272039042a1c0ddd3e21')

if resp.status_code == 200:
 parsed_resp = json.loads(resp.content)
 print json.dumps(parsed_resp)

```

### Sample response

```

{
 "status": 200,
 "code": 1000,
 "message": "deleted"
}

```

## API to Get the Config

This endpoint gets the config with given ID

```
GET /openapi/v1/ext_appliances/<id>/config/<config_id>
```

where <id> is the appliance\_id that can be obtained from the [API to Get List of Appliances, on page 912](#), <config\_id> is the id that can be obtained from [API to Get the Configs on Config Type, on page 917](#).

Parameters: The request URL contains the following parameters

| Name      | Type   | Description              |
|-----------|--------|--------------------------|
| id        | string | Specify the Appliance ID |
| config_id | string | Specify the config ID    |

Response object: Returns the config with given ID.

### Sample response

```
resp =
restclient.get('/ext_appliances/63c1272039042a1c0ddd3e20/config/63c1272039042a1c0ddd3e21')
if resp.status_code == 200:
 parsed_resp = json.loads(resp.content)
 print json.dumps(parsed_resp)
b
```

### Sample response

```
{
 "mode": "TEST_AND_APPLY",
 "name": "VM3_ERSPAN",
 "root_scope_id": "63bf8d2f497d4f7287dbd335",
 "vrf_id": 1,
 "appliance_id": "63c1272039042a1c0ddd3e20",
 "deleted": false,
 "type": "VM3_ERSPAN",
 "state": "COMMIT",
 "attempts": 0,
 "config": {
 "secured": {},
 "unsecured": {
 "cidr": [
 "172.26.231.141/23",
 "172.26.231.142/23",
 "172.26.231.143/23"
],
 "gateway": [
 "172.26.231.140",
 "172.26.231.140",
 "172.26.231.140"
],
 "cidr_v6": [],
 "gateway_v6": [],
 "dns": [
 "test"
],
 "search_domains": [],
 "hostname": "hjtest",
 "https_proxy": "",
 "no_proxy": []
 }
 },
 "push_to_dio_at": 0,
 "created_at": 1673602848,
 "updated_at": 1673602848,
 "discovery_completed_at": 0,
 "committed_at": 0,
 "delete_at": 0,
```

```

 "error_at": 0,
 "hidden": false,
 "discovery_phase": 0,
 "internal": false,
 "id": "63c1272039042a1c0ddd3e21"
 }

```

## API to Get Appliance Schema

This endpoint gets the Appliance schema with given type

```
GET /openapi/v1/ext_appliances/schema/<type>
```

where <type> is a string to decide Appliance type.

Parameters: The request URL contains the following parameters

| Name | Type   | Description                                                                                                                                                   |
|------|--------|---------------------------------------------------------------------------------------------------------------------------------------------------------------|
| type | string | Specify the Appliance type, the value can be either "TETRATION_EDGE", "TETRATION_DATA_INGEST", "TETRATION_EXPORT", "TETRATION_ERSPAN" or "TETRATION_INTERNAL" |

Response object: Returns the config schema.

### Sample response

```

resp = restclient.get('/ext_appliances/schema/TETRATION_ERSPAN')
if resp.status_code == 200:
 parsed_resp = json.loads(resp.content)
 print json.dumps(parsed_resp)

```

### Sample response

```

{
 "name": "Data Ingest for ERSPAN",
 "type": "tetration_erspan",
 "desc": "Data Ingest Appliance for ERSPAN",
 "long_desc": "Data Ingest appliance for ERSPAN is a software appliance that can export flow data from ERSPAN appliance.",
 "valid_config": [
 "VM3_ERSPAN"
],
 "deploy_config": [
 "VM3_ERSPAN"
],
 "connectors": [
 "ERSPAN"
],
 "limit_connectors_per_appliance": 0,
 "limit_per_rootscope": 8,
 "limit_per_rootscope_taaS": 4,
 "limit_per_cluster": 150,
 "cco_url":
 "https://software.cisco.com/download/home/286309796/type/286309874/release/3.7.1.26.devel"
}

```

•

## API to List Troubleshooting Commands Available for an Appliance

This endpoint returns the list of troubleshooting commands available for given Appliance.

GET /openapi/v1/ext\_appliances/<id>/commands/available

where <id> is the appliance\_id that can be obtained from the [API to Get List of Appliances, on page 912](#).

Parameters: The request URL contains the following parameters

| Name | Type   | Description              |
|------|--------|--------------------------|
| id   | string | Specify the Appliance ID |

Response object: Returns the list of troubleshooting commands available for given Appliance.

### Sample response

```
resp = restclient.get('/ext_appliances/63c6ef42bca44e2b5e729191/commands/available')
if resp.status_code == 200:
 parsed_resp = json.loads(resp.content)
 print json.dumps(parsed_resp)
```

### Sample response

```
[
 {
 "type": "UPDATE_LISTENING_PORT",
 "name": "Update the listening port on a connector"
 },
 {
 "type": "SNAPSHOT_APPLIANCE",
 "name": "Collect Snapshot from Appliance"
 },
 {
 "type": "LIST_FILES",
 "name": "List a directory"
 },
 {
 "type": "CONTROLLER_PROFILING",
 "name": "Collect controller profile"
 },
 {
 "type": "SHOW_LOG",
 "name": "Show logs"
 },
 {
 "type": "SHOW_SUPERVISOR_COMMANDS",
 "name": "Execute supervisorctl command"
 },
 {
 "type": "PACKET_CAPTURE",
 "name": "Packet capture"
 },
 {
 "type": "NETWORK_CONNECTIVITY_COMMANDS",
 "name": "Test network connectivity"
 },
 {
 "type": "SHOW_DOCKER_COMMANDS",
 "name": "Execute docker command"
 },
 {
 "type": "CONNECTOR_ALERT_INTERVAL_APPLIANCE",
 "name": "Override connector alert interval for Appliance"
 }
]
```

```

 },
 {
 "type": "SHOW_RUNNING_CONF",
 "name": "Show running configuration"
 },
 {
 "type": "SHOW_SYS_COMMANDS",
 "name": "Execute system command"
 }
]

```

## API to List Troubleshooting Commands

This endpoint returns the list of troubleshooting commands that are enabled for given Appliance.

```
GET /openapi/v1/ext_appliances/<id>/commands
```

where <id> is the appliance\_id that can be obtained from the [API to Get List of Appliances, on page 912](#).

Parameters: The request URL contains the following parameters

| Name | Type   | Description              |
|------|--------|--------------------------|
| id   | string | Specify the Appliance ID |

Response object: Returns the list of troubleshooting commands that are enabled for given Appliance.

### Sample response

```

resp = restclient.get('/ext_appliances/63be3blade36423c12bff6e1/commands')
if resp.status_code == 200:
 parsed_resp = json.loads(resp.content)
 print json.dumps(parsed_resp)

```

### Sample response

```

[
 {
 "appliance_id": "63be3blade36423c12bff6e1",
 "state": "pending",
 "level": "APPLIANCE",
 "command": "SHOW_LOG",
 "arg_string": "",
 "args": {},
 "tailed": false,
 "rc": 0,
 "push_to_dio_at": 0,
 "attempts": 0,
 "deleted": false,
 "deleted_at": 0,
 "created_at": 1673595392,
 "total_chunk": 0,
 "id": "63c10a0039042a6ae1b008c"
 }
]

```

## API to Create a Troubleshooting Command

This endpoint creates a troubleshooting command available for given Appliance.

```
POST /openapi/v1/ext_appliances/<id>/commands
```

where <id> is the appliance\_id that can be obtained from the [API to Get List of Appliances, on page 912](#). In request payload, <command> is a troubleshooting command type that can be obtained from [API to Get the](#)

[Schema of Troubleshooting Commands, on page 911](#) response "valid\_appliances" field. <arguments> is a filled JSON object of the command schema, which can be obtained from [API to Get the Schema of Troubleshooting Commands, on page 911](#).

Parameters: The request URL contains the following parameters

| Name      | Type   | Description                                      |
|-----------|--------|--------------------------------------------------|
| id        | string | Specify the Appliance ID                         |
| command   | string | Specify the command type                         |
| arguments | set    | Provide the filled command schema in JSON format |

Response object: Returns the troubleshooting command created for given Appliance.

#### Sample response

```
req_payload = {
 "command": "SHOW_LOG",
 "arguments": {}
}
resp = restclient.post('/ext_appliances/63be3blade36423c12bff6e1/commands',
json_body=json.dumps(req_payload))
if resp.status_code == 200:
 parsed_resp = json.loads(resp.content)
 print json.dumps(parsed_resp)
```

#### Sample response

```
{
 "appliance_id": "63be3blade36423c12bff6e1",
 "state": "pending",
 "level": "APPLIANCE",
 "command": "SHOW_LOG",
 "args": {},
 "tailed": false,
 "rc": 0,
 "push_to_dio_at": 0,
 "attempts": 0,
 "deleted": false,
 "deleted_at": 0,
 "created_at": 1673595392,
 "total_chunk": 0,
 "id": "63c10a0039042a6aee1b008c"
}
```

.

## API to Delete a Troubleshooting Command

This endpoint deletes a troubleshooting command enabled for given Appliance.

```
DELETE /openapi/v1/ext_appliances/<id>/commands/<command_id>
```

where <id> is the appliance\_id that can be obtained from the [API to Get List of Appliances, on page 912](#), <command\_id> is the id that can be obtained from [API to List Troubleshooting Commands, on page 924](#).

Parameters: The request URL contains the following parameters

| Name       | Type   | Description              |
|------------|--------|--------------------------|
| id         | string | Specify the Appliance ID |
| command_id | string | Specify the command ID   |

Response object: Returns the status of the troubleshooting command deleted for given Appliance.

#### Sample response

```
resp =
restclient.delete('/ext_appliances/63be3blade36423c12bff6e1/commands/63c10a0039042a6aee1b008c')

if resp.status_code == 200:
 parsed_resp = json.loads(resp.content)
 print json.dumps(parsed_resp)
```

#### Sample response

```
{
 "status": 200,
 "code": 1000,
 "message": "deleted"
}
```

## API to Return a Troubleshooting Command

This endpoint returns the selected troubleshooting command for a given Appliance.

```
GET /openapi/v1/ext_appliances/<id>/commands/<command_id>
```

where <id> is the appliance\_id that is obtained from the [API to Get List of Appliances, on page 912](#), <command\_id> is the id that is obtained from [API to List Troubleshooting Commands, on page 924](#).

Parameters: The request URL contains the following parameters

| Name       | Type   | Description              |
|------------|--------|--------------------------|
| id         | string | Specify the Appliance ID |
| command_id | string | Specify the command ID   |

Response object: Returns the selected troubleshooting command for a given Appliance.

#### Sample response

```
resp =
restclient.get('/ext_appliances/63be3blade36423c12bff6e1/commands/63c10fd139042a1c0ddd3e1f')

if resp.status_code == 200:
 parsed_resp = json.loads(resp.content)
 print json.dumps(parsed_resp)
```

#### Sample response

```
{
 "appliance_id": "63be3blade36423c12bff6e1",
 "state": "pending",
 "level": "APPLIANCE",
 "command": "SHOW_LOG",
 "arg_string": "",
 "args": {},
 "tailed": false,
```



```

 "rc": 0,
 "push_to_dio_at": 0,
 "attempts": 0,
 "deleted": false,
 "deleted_at": 0,
 "created_at": 1673596881,
 "total_chunk": 0,
 "id": "63c10fd139042a1c0ddd3e1f"
 }

```

## API to Download the Output of the Appliance Command as a File

This endpoint downloads the output of the command as a file.

```
GET /openapi/v1/appliances/<id>/commands/{command_id}/download
```

where <id> is the appliance\_id that can be obtained from the [API to Get List of Appliances, on page 912](#), <command\_id> is the id that can be obtained from [API to List Troubleshooting Commands, on page 924](#). Not all commands have downloadable output, check the command schema provided by [API to Get the Schema of Troubleshooting Commands, on page 911](#), where "output\_type": "FILE" indicates it has downloadable content and "output\_ext" tells the file type.

Parameters: The request URL contains the following parameters

| Name       | Type   | Description              |
|------------|--------|--------------------------|
| id         | string | Specify the Appliance ID |
| command_id | string | Specify the command ID   |

Response object: Download the command output as a file.

### Sample response

```

resp = restclient.download('downloadFile',
'/appliances/63c6ef42bca44e2b5e729191/commands/63cace941a49bd4c0e0cf45a/download')

```

## Connectors

### Connectors APIs

The Connectors APIs are associated with managing Secure Workload Connectors. These set of APIs require the `sensor_management` or `external_integration` capability associated with the API key.

#### API to Get All Types of Connectors

This endpoint gets all types of Connectors for given root scope.

```
GET /openapi/v1/connectors/cards?root_scope_id=<root_scope_id>
```

where <root\_scope\_id> is the root\_scope\_id that can be obtained from the [Get scopes, on page 797](#) API.

Parameters: The request URL contains the following parameters

| Name          | Type   | Description            |
|---------------|--------|------------------------|
| root_scope_id | string | Specify the root scope |

Response object: Returns all types of Connector with given root scope ID.

**Sample response**

```
resp = restclient.get('/connectors/cards?root_scope_id=63bf8d2f497d4f7287dbd335')
if resp.status_code == 200:
 parsed_resp = json.loads(resp.content)
 print json.dumps(parsed_resp)
```

**Sample response**

```
[{
 "type": "NETFLOW",
 "name": "NetFlow",
 "desc": "Collect telemetry from network devices",
 "long_desc": "Collect NetFlow V9 and/or IPFIX telemetry from network devices such as routers and switches.",
 "group": "Flow Ingest",
 "index": 0,
 "appliance_type": "tetration_data_ingest",
 "state": "disabled",
 "limit_per_appliance": 3,
 "limit_per_rootscope": 10,
 "limit_per_cluster": 100,
 "config": [
 "LOG",
 "ALERT"
],
 "max_instances": 0,
 "noteworthy": false,
 "hidden": false,
 "capabilities": [
 "Flow Visibility"
],
 "connector_count": 0,
 "group_order": 0
},
{
 "type": "NETSCALER",
 "name": "NetScaler",
 "desc": "Collect telemetry from Citrix ADC",
 "long_desc": "Collect telemetry from Citrix ADC, stitch client and server side flows.",
 "group": "Flow Ingest",
 "index": 2,
 "appliance_type": "tetration_data_ingest",
 "state": "disabled",
 "limit_per_appliance": 3,
 "limit_per_rootscope": 10,
 "limit_per_cluster": 100,
 "config": [
 "LOG",
 "ALERT"
],
 "max_instances": 0,
 "noteworthy": false,
 "hidden": false,
 "capabilities": [
 "Flow Visibility",
 "Flow Stitching",
 "ADM"
],
 "connector_count": 0,
 "group_order": 0
},
{
 "type": "F5",
```

```

 "name": "F5",
 "desc": "Collect telemetry from F5 BIG-IP",
 "long_desc": "Collect telemetry from F5 BIG-IP, stitch client and server side flows,
 enrich client inventory with user attributes.",
 "group": "Flow Ingest",
 "index": 1,
 "appliance_type": "tetration_data_ingest",
 "state": "disabled",
 "limit_per_appliance": 3,
 "limit_per_rootscope": 10,
 "limit_per_cluster": 100,
 "config": [
 "LDAP",
 "LOG",
 "ALERT"
],
 "max_instances": 0,
 "noteworthy": false,
 "hidden": false,
 "capabilities": [
 "Flow Visibility",
 "User Insights",
 "Flow Stitching",
 "ADM"
],
 "connector_count": 0,
 "group_order": 0
 },
 {
 "type": "ANYCONNECT",
 "name": "AnyConnect",
 "desc": "Collect telemetry from AnyConnect NVM",
 "long_desc": "Collect telemetry data from Cisco AnyConnect Network Visibility Module
 (NVM) and enrich endpoint inventories with user attributes",
 "group": "Endpoints",
 "index": 0,
 "appliance_type": "tetration_data_ingest",
 "state": "disabled",
 "limit_per_appliance": 1,
 "limit_per_rootscope": 50,
 "limit_per_cluster": 500,
 "config": [
 "ENDPOINT",
 "LDAP",
 "LOG",
 "ALERT"
],
 "max_instances": 0,
 "noteworthy": false,
 "hidden": false,
 "capabilities": [
 "Flow Visibility",
 "Process Annotation",
 "User Insights",
 "Endpoint Insights",
 "Inventory Enrichment"
],
 "connector_count": 0,
 "group_order": 1
 },
 {
 "type": "ASA",
 "name": "Cisco Secure Firewall",
 "desc": "Collect telemetry from Cisco Secure Firewall",

```

```

 "long_desc": "Collect telemetry from Cisco Secure Firewall, stitch client and server
side flows. Recommended usage with ISE connector for flow visibility.",
 "group": "Flow Ingest",
 "index": 3,
 "appliance_type": "tetration_data_ingest",
 "state": "disabled",
 "limit_per_appliance": 1,
 "limit_per_rootscope": 10,
 "limit_per_cluster": 100,
 "config": [
 "LOG",
 "ALERT"
],
 "max_instances": 0,
 "noteworthy": false,
 "hidden": false,
 "capabilities": [
 "Flow Visibility",
 "Flow Stitching",
 "ADM"
],
 "connector_count": 0,
 "group_order": 0
},
{
 "type": "SLACK",
 "name": "Slack",
 "desc": "Send alerts to Slack",
 "long_desc": "Send Tetration Alerts to Slack.",
 "group": "Alerts",
 "index": 2,
 "appliance_type": "tetration_edge",
 "state": "disabled",
 "limit_per_appliance": 1,
 "limit_per_rootscope": 1,
 "limit_per_cluster": 150,
 "config": [
 "SLACK_NOTIFIER",
 "ALERT"
],
 "max_instances": 0,
 "noteworthy": false,
 "hidden": false,
 "capabilities": [
 "Alert Destination"
],
 "connector_count": 0,
 "group_order": 3
},
{
 "type": "KINESIS",
 "name": "Kinesis",
 "desc": "Send alerts to Kinesis",
 "long_desc": "Send Tetration Alerts to Kinesis.",
 "group": "Alerts",
 "index": 4,
 "appliance_type": "tetration_edge",
 "state": "disabled",
 "limit_per_appliance": 1,
 "limit_per_rootscope": 1,
 "limit_per_cluster": 150,
 "config": [
 "KINESIS_NOTIFIER",
 "ALERT"
]
}

```

```

],
 "max_instances": 0,
 "noteworthy": false,
 "hidden": false,
 "capabilities": [
 "Alert Destination"
],
 "connector_count": 0,
 "group_order": 3
 },
 {
 "type": "SYSLOG",
 "name": "Syslog",
 "desc": "Send alerts to Syslog server",
 "long_desc": "Send Tetration Alerts to Syslog server.",
 "group": "Alerts",
 "index": 0,
 "appliance_type": "tetration_edge",
 "state": "disabled",
 "limit_per_appliance": 1,
 "limit_per_rootscope": 1,
 "limit_per_cluster": 150,
 "config": [
 "SYSLOG_NOTIFIER",
 "SYSLOG_SEVERITY_MAPPING",
 "ALERT"
],
 "max_instances": 0,
 "noteworthy": false,
 "hidden": false,
 "capabilities": [
 "Alert Destination"
],
 "connector_count": 0,
 "group_order": 3
 },
 {
 "type": "EMAIL",
 "name": "Email",
 "desc": "Send alerts to Email",
 "long_desc": "Send Tetration Alerts to Email.",
 "group": "Alerts",
 "index": 1,
 "appliance_type": "tetration_edge",
 "state": "disabled",
 "limit_per_appliance": 1,
 "limit_per_rootscope": 1,
 "limit_per_cluster": 150,
 "config": [
 "EMAIL_NOTIFIER",
 "ALERT"
],
 "max_instances": 0,
 "noteworthy": false,
 "hidden": false,
 "capabilities": [
 "Alert Destination"
],
 "connector_count": 0,
 "group_order": 3
 },
 {
 "type": "PAGERDUTY",
 "name": "Pager Duty",

```

```

"desc": "Send alerts to Pager Duty",
"long_desc": "Send Tetration Alerts to Pager Duty",
"group": "Alerts",
"index": 3,
"appliance_type": "tetration_edge",
"state": "disabled",
"limit_per_appliance": 1,
"limit_per_rootscope": 1,
"limit_per_cluster": 150,
"config": [
 "PAGERDUTY_NOTIFIER",
 "ALERT"
],
"max_instances": 0,
"noteworthy": false,
"hidden": false,
"capabilities": [
 "Alert Destination"
],
"connector_count": 0,
"group_order": 3
},
{
 "type": "ISE",
 "name": "ISE",
 "desc": "Collect endpoints and inventories from Cisco ISE",
 "long_desc": "Collect information about endpoints and inventories managed by Cisco ISE appliances and enrich endpoint inventories with user attributes and secure group tags (SGT). Recommended usage with Cisco Secure Firewall connector for flow visibility.",
 "group": "Endpoints",
 "index": 1,
 "appliance_type": "tetration_edge",
 "state": "disabled",
 "limit_per_appliance": 1,
 "limit_per_rootscope": 1,
 "limit_per_cluster": 150,
 "config": [
 "LDAP",
 "LOG",
 "ALERT"
],
 "instance_spec": "ISE",
 "max_instances": 20,
 "noteworthy": false,
 "hidden": false,
 "capabilities": [
 "User Insights",
 "Inventory Enrichment",
 "Endpoint Insights",
 "Software Compliance Posture",
 "MDM Insights"
],
 "connector_count": 0,
 "group_order": 1
},
{
 "type": "MERAKEI",
 "name": "Meraki",
 "desc": "Collect telemetry from Meraki firewalls",
 "long_desc": "Collect telemetry data from Meraki firewalls.",
 "group": "Flow Ingest",
 "index": 5,
 "appliance_type": "tetration_data_ingest",
 "state": "disabled",

```

```

 "limit_per_appliance": 1,
 "limit_per_rootscope": 10,
 "limit_per_cluster": 100,
 "config": [
 "LOG",
 "ALERT"
],
 "max_instances": 0,
 "noteworthy": false,
 "hidden": false,
 "capabilities": [
 "Flow Visibility"
],
 "connector_count": 0,
 "group_order": 0
 },
 {
 "type": "SERVICENOW",
 "name": "ServiceNow",
 "desc": "Collect ServiceNow CMDB records for inventories",
 "long_desc": "Collect CMDB information and service records from ServiceNow instance and enriches endpoint inventories with cmdb attributes.",
 "group": "Inventory Enrichment",
 "index": 1,
 "appliance_type": "tetration_edge",
 "state": "disabled",
 "limit_per_appliance": 1,
 "limit_per_rootscope": 1,
 "limit_per_cluster": 150,
 "config": [
 "SERVICENOW",
 "SYNC_INTERVAL",
 "LOG",
 "ALERT"
],
 "instance_spec": "SERVICENOW",
 "max_instances": 10,
 "noteworthy": false,
 "hidden": false,
 "capabilities": [
 "User Insights",
 "Inventory Enrichment",
 "Endpoint Insights",
 "Software Compliance Posture"
],
 "connector_count": 0,
 "group_order": 2
 },
 {
 "type": "ERSPAN",
 "name": "ERSPAN",
 "desc": "Collect ERSPAN traffic",
 "long_desc": "",
 "group": "Flow Ingest",
 "index": 7,
 "appliance_type": "tetration_erspan",
 "state": "enabled",
 "limit_per_appliance": 3,
 "limit_per_rootscope": 24,
 "limit_per_cluster": 450,
 "config": [],
 "max_instances": 0,
 "noteworthy": false,
 "hidden": false,

```

```

 "capabilities": [],
 "connector_count": 3,
 "group_order": 0
 },
 {
 "type": "AWS_CONNECTOR",
 "name": "AWS",
 "desc": "AWS Connector",
 "long_desc": "",
 "group": "Cloud",
 "index": 0,
 "appliance_type": "tetration_internal",
 "state": "disabled",
 "limit_per_appliance": 5000,
 "limit_per_rootscope": 5000,
 "limit_per_cluster": 100000,
 "config": [
 "AWS_CONNECTOR"
],
 "max_instances": 0,
 "noteworthy": true,
 "pre_release_tag": "BETA",
 "hidden": false,
 "capabilities": [
 "Flow Visibility",
 "Segmentation",
 "Managed K8s",
 "Inventory Enrichment"
],
 "connector_count": 0,
 "group_order": 5
 },
 {
 "type": "AZURE_CONNECTOR",
 "name": "Azure",
 "desc": "Azure Connector",
 "long_desc": "",
 "group": "Cloud",
 "index": 1,
 "appliance_type": "tetration_internal",
 "state": "disabled",
 "limit_per_appliance": 5000,
 "limit_per_rootscope": 5000,
 "limit_per_cluster": 100000,
 "config": [
 "AZURE_CONNECTOR"
],
 "max_instances": 0,
 "noteworthy": true,
 "pre_release_tag": "BETA",
 "hidden": false,
 "capabilities": [
 "Flow Visibility",
 "Segmentation",
 "Managed K8s",
 "Inventory Enrichment"
],
 "connector_count": 0,
 "group_order": 5
 },
 {
 "type": "GCP_CONNECTOR",
 "name": "GCP",
 "desc": "GCP Connector",

```



```

 "long_desc": "",
 "group": "Cloud",
 "index": 2,
 "appliance_type": "tetration_internal",
 "state": "disabled",
 "limit_per_appliance": 5000,
 "limit_per_rootscope": 5000,
 "limit_per_cluster": 100000,
 "config": [
 "GCP_CONNECTOR"
],
 "max_instances": 0,
 "noteworthy": true,
 "pre_release_tag": "BETA",
 "hidden": false,
 "capabilities": [
 "Managed K8s"
],
 "connector_count": 0,
 "group_order": 5
]]

```

### API to Delete a Connector

This endpoint deletes the given Connector.

```
DELETE /openapi/v1/connectors/<id>
```

where <id> is the id that can be obtained from the [API to Get Connectors, on page 938](#).

Parameters: The request URL contains the following parameters

| Name | Type   | Description              |
|------|--------|--------------------------|
| id   | string | Specify the Connector ID |

Response object: Returns the status of the deleted Connector.

#### Sample response

```

resp = restclient.delete('/connectors/63c12e316419d0131767e21c')
if resp.status_code == 200:
 parsed_resp = json.loads(resp.content)
 print json.dumps(parsed_resp)

```

#### Sample response

```

{
 "status": 200,
 "code": 1000,
 "message": "deleted"
}

```

### API to Get a Connector by ID

This endpoint gets the Connector with given ID.

```
GET /openapi/v1/connectors/<id>
```

where <id> is the id that can be obtained from the [API to Get Connectors, on page 938](#).

Parameters: The request URL contains the following parameters

| Name | Type   | Description              |
|------|--------|--------------------------|
| id   | string | Specify the Connector ID |

Response object: Returns the Connector with given ID.

### Sample response

```
resp = restclient.get('/connectors/63c12e316419d0131767e21b')
if resp.status_code == 200:
 parsed_resp = json.loads(resp.content)
 print json.dumps(parsed_resp)
```

### Sample response

```
{
 "name": "ERSPAN",
 "updated_at": 0,
 "created_at": 1673604657,
 "appliance_id": "63c1272039042a1c0ddd3e20",
 "root_scope_id": "63bf8d2f497d4f7287dbd335",
 "vrf_id": 1,
 "type": "ERSPAN",
 "ip_bindings": [],
 "internal": false,
 "id": "63c12e316419d0131767e21b"
}
```

## API to Rename a Connector

This endpoint renames the Connector with given ID.

```
PUT /openapi/v1/connectors/<id>
```

where <id> is the id that can be obtained from the [API to Get Connectors, on page 938](#).

Parameters: The request URL contains the following parameters

| Name | Type   | Description                           |
|------|--------|---------------------------------------|
| id   | string | Specify the Connector ID              |
| name | string | Specify the new name of the Connector |

Response object: Returns the Connector with given ID and the new name.

### Sample response

```
req_payload = {
 "name": "ERSPAN2",
}
resp = restclient.put('/ext_appliances/63c12e316419d0131767e21b',
json_body=json.dumps(req_payload))
if resp.status_code == 200:
 parsed_resp = json.loads(resp.content)
 print json.dumps(parsed_resp)
```

### Sample response

```
{
 "name": "ERSPAN2",
 "updated_at": 0,
}
```

```

 "created_at": 1673604657,
 "appliance_id": "63c1272039042a1c0ddd3e20",
 "root_scope_id": "63bf8d2f497d4f7287dbd335",
 "vrf_id": 1,
 "type": "ERSPAN",
 "ip_bindings": [],
 "internal": false,
 "id": "63c12e316419d0131767e21b"
 }

```

## API to Get the Connector Info with Details

This endpoint gets the Connector info with details.

```
GET /openapi/v1/connectors/cards/<type>?root_scope_id=<root_scope_id>
```

where <root\_scope\_id> is the root\_scope\_id that can be obtained from the [Get scopes, on page 797](#) API. In request payload, <type> is a string to decide Connector type.

Parameters: The request URL contains the following parameters

| Name          | Type   | Description                                                                                                                                                                                                                                    |
|---------------|--------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| root_scope_id | string | Specify the root scope                                                                                                                                                                                                                         |
| type          | string | Specify the Connector type, the value can be either "NETFLOW", "NETSCALER" "F5" "AWS" "ANYCONNECT" "ASA" "SLACK" "KINESIS" "SYSLOG" "EMAIL" "MERAKE" "PAGERDUTY" "ISE" "SERVICENOW" "ERSPAN" "AWS_CONNECTOR" "AZURE_CONNECTOR" "GCP_CONNECTOR" |

Response object: Returns the Connectors under given scope.

### Sample response

```

resp = restclient.get('/connectors/cards/NETFLOW?root_scope_id=63bf8d2f497d4f7287dbd335')
if resp.status_code == 200:
 parsed_resp = json.loads(resp.content)
 print json.dumps(parsed_resp)

```

### Sample response

```

{
 "type": "NETFLOW",
 "name": "NetFlow",
 "desc": "Collect telemetry from network devices",
 "long_desc": "Collect NetFlow V9 and/or IPFIX telemetry from network devices such as routers and switches.",
 "group": "Flow Ingest",
 "index": 0,
 "appliance_type": "tetration_data_ingest",
 "state": "disabled",
 "limit_per_appliance": 3,
 "limit_per_rootscope": 10,
 "limit_per_cluster": 100,

```

```

"config": [
 "LOG",
 "ALERT"
],
"max_instances": 0,
"noteworthy": false,
"hidden": false,
"capabilities": [
 "Flow Visibility"
],
"connector_count": 0,
"info": {
 "help": "NetFlow connector collects telemetry data from various network devices (such
as routers, switches).
 It supports ingest of telemetry data in IPFIX and NetFlow V9
protocols. This connector can be used to discover inventory as it provides a network context.
The connector helps convert data from flow exports and send them securely as Tetration
Flow records into an instance of Tetration.

 Connector Alerts:
 When
Connector Alerts are enabled, you may receive the following alerts:
 1. NetFlow
Connector is down (due to missing heartbeats).
 2. Informational alert on high
CPU/Memory usage."
},
"group_order": 0
}

```

## API to Get Connectors

This endpoint returns all Connectors for a given Appliance.

GET

/openapi/v1/connectors?root\_scope\_id=<root\_scope\_id>&appliance\_id=<appliance\_id>&type=<type>&state=<state>

where <root\_scope\_id> is the root\_scope\_id that can be obtained from the [Get scopes, on page 797](#) API, <appliance\_id> is the appliance\_id that can be obtained from [API to Get List of Appliances, on page 912](#), <type> is a string to decide Connector type that can be obtained from [API to Get Appliance Schema, on page 922](#) "connectors" field, <state> is a filter for connectors state.

Parameters: The request URL contains the following parameters

| Name          | Type   | Description                                                                                                                                                                                                                                    |
|---------------|--------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| root_scope_id | string | Specify the root scope                                                                                                                                                                                                                         |
| appliance_id  | string | Specify the Appliance ID                                                                                                                                                                                                                       |
| type          | string | Specify the Connector type, the value can be either "NETFLOW", "NETSCALER" "F5" "AWS" "ANYCONNECT" "ASA" "SLACK" "KINESIS" "SYSLOG" "EMAIL" "MERAKE" "PAGERDUTY" "ISE" "SERVICENOW" "ERSPAN" "AWS_CONNECTOR" "AZURE_CONNECTOR" "GCP_CONNECTOR" |

| Name  | Type   | Description                                                                                |
|-------|--------|--------------------------------------------------------------------------------------------|
| state | string | Filter the Connectors state, the value can be either “ENABLED” “DISABLED” or “UNAVAILABLE” |

Response object: Returns all Connectors for given Appliance.

### Sample response

```
resp =
restclient.get('root_scope_id=63bf8d2f497d4f7287dbd335&appliance_id=63c1272039042a1c0ddd3e20&type=ERSPAN&state=ENABLED')

if resp.status_code == 200:
 parsed_resp = json.loads(resp.content)
 print json.dumps(parsed_resp)
```

### Sample response

```
[
 {
 "name": "ERSPAN",
 "updated_at": 0,
 "created_at": 1673604657,
 "appliance_id": "63c1272039042a1c0ddd3e20",
 "root_scope_id": "63bf8d2f497d4f7287dbd335",
 "vrf_id": 1,
 "type": "ERSPAN",
 "ip_bindings": [],
 "state": "enabled",
 "internal": false,
 "id": "63c12e316419d0131767e21b"
 },
 {
 "name": "ERSPAN",
 "updated_at": 0,
 "created_at": 1673604657,
 "appliance_id": "63c1272039042a1c0ddd3e20",
 "root_scope_id": "63bf8d2f497d4f7287dbd335",
 "vrf_id": 1,
 "type": "ERSPAN",
 "ip_bindings": [],
 "state": "enabled",
 "internal": false,
 "id": "63c12e316419d0131767e21c"
 },
 {
 "name": "ERSPAN",
 "updated_at": 0,
 "created_at": 1673604657,
 "appliance_id": "63c1272039042a1c0ddd3e20",
 "root_scope_id": "63bf8d2f497d4f7287dbd335",
 "vrf_id": 1,
 "type": "ERSPAN",
 "ip_bindings": [],
 "state": "enabled",
 "internal": false,
 "id": "63c12e316419d0131767e21d"
 }
]
```

## API to Create a Connector

This endpoint creates a Connector for given Appliance.

POST /openapi/v1/connectors

In request payload, <root\_scope\_id> is the root\_scope\_id that can be obtained from the [Get scopes, on page 797](#) API, <appliance\_id> is the appliance\_id that can be obtained from [API to Get List of Appliances, on page 912](#), <type> is a string to decide Connector type that can be obtained from [API to Get Appliance Schema, on page 922](#) "connectors" field.

Parameters: The request URL contains the following parameters

| Name          | Type   | Description                                                                                                                                                                                                                                    |
|---------------|--------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| name          | string | Specify the name                                                                                                                                                                                                                               |
| root_scope_id | string | Specify the root scope                                                                                                                                                                                                                         |
| appliance_id  | string | Specify the Appliance ID                                                                                                                                                                                                                       |
| type          | string | Specify the Connector type, the value can be either "NETFLOW", "NETSCALER" "F5" "AWS" "ANYCONNECT" "ASA" "SLACK" "KINESIS" "SYSLOG" "EMAIL" "MERAKI" "PAGERDUTY" "ISE" "SERVICENOW" "ERSPAN" "AWS_CONNECTOR" "AZURE_CONNECTOR" "GCP_CONNECTOR" |
| version       | string | Specify the version                                                                                                                                                                                                                            |

Response object: Returns the created connector.

### Sample response

```
req_payload = {
 "root_scope_id": "63c41ff2497d4f5f5be73662",
 "appliance_id": "63c6ef42bca44e2b5e729191",
 "type": "NETFLOW",
 "name": "netflowtest",
 "version": "1.1.1"
}
resp = restclient.post('/connectors', json_body=json.dumps(req_payload))
if resp.status_code == 200:
 parsed_resp = json.loads(resp.content)
 print json.dumps(parsed_resp)
```

### Sample response

```
{
 "name": "netflowtest",
 "updated_at": 0,
 "created_at": 1674187875,
 "appliance_id": "63c6ef42bca44e2b5e729191",
 "root_scope_id": "63c41ff2497d4f5f5be73662",
 "vrf_id": 1,
```

```

 "type": "NETFLOW",
 "ip_bindings": [],
 "sources": [],
 "internal": false,
 "id": "63ca14631a49bd4c0e0cefa2"
 }

```

## API to Get the Configs on Connector Config Type

This endpoint gets the configs of the Connector with given ID.

```
GET /openapi/v1/connectors/<id>/config?config_type=<config_type>
```

where <id> is the id that can be obtained from the [API to Get List of Appliances, on page 912](#). <config\_type> is the "valid\_config" that can be obtained from [API to Get Appliance Schema, on page 922](#).

Parameters: The request URL contains the following parameters

| Name        | Type   | Description                                                                                                                                     |
|-------------|--------|-------------------------------------------------------------------------------------------------------------------------------------------------|
| id          | string | Specify the Connector ID                                                                                                                        |
| config_type | string | Specify the config type. Refer to <a href="#">API to Get the Schema of Config, on page 909</a> for all possible values listed under Description |

Response object: Returns the configs with given Connector ID and config type.

### Sample response

```

resp = restclient.get('/connectors/63db5418e6ee1167a4c0986c/config?config_type=LOG')
if resp.status_code == 200:
 parsed_resp = json.loads(resp.content)
 print json.dumps(parsed_resp)

```

### Sample response

```

[{
 "mode": "TEST_AND_APPLY",
 "name": "Log instance 2/1/23 22:29",
 "root_scope_id": "63d98f45497d4f53005b24fa",
 "vrf_id": 1,
 "appliance_id": "63dad690e6ee1131f255e985",
 "connector_id": "63db5418e6ee1167a4c0986c",
 "service_id": "63db5418e6ee1167a4c0986d",
 "deleted": false,
 "type": "LOG",
 "state": "COMMIT",
 "error_code": "NO_ERROR",
 "error_text": "",
 "attempts": 1,
 "config": {
 "secured": {},
 "unsecured": {
 "log-level": "info",
 "max-log-size": 10,
 "max-log-age": 30,
 "max-log-backups": 20
 }
 }
},
 "push_to_dio_at": 1675319360,
 "created_at": 1675319360,

```

```

 "updated_at": 1675319364,
 "discovery_completed_at": 0,
 "committed_at": 1675319364,
 "delete_at": 0,
 "error_at": 0,
 "hidden": false,
 "discovery_phase": 0,
 "internal": false,
 "id": "63db5840f029813659f9fcf5"
 }
}

```

## API to Add a New Config to Connector

This endpoint adds a new config to the Connector with given ID

```
POST /openapi/v1/connectors/<id>/config
```

where <id> is the id that can be obtained from the [API to Get Connectors, on page 938](#). <config\_type> is the "valid\_config" that can be obtained from [API to Get Appliance Schema, on page 922](#). To obtain <config> schema, select one of the "config" from [API to Get All Types of Connectors, on page 927](#) response, apply the "config" to [API to Get the Schema of Config, on page 909](#).

Parameters: The request URL contains the following parameters

| Name   | Type   | Description                                                                                                                                     |
|--------|--------|-------------------------------------------------------------------------------------------------------------------------------------------------|
| name   | string | Specify the config name                                                                                                                         |
| type   | string | Specify the config type. Refer to <a href="#">API to Get the Schema of Config, on page 909</a> for all possible values listed under Description |
| config | set    | Provide the filled config schema in JSON format                                                                                                 |

Response object: Returns the updated config.

### Sample response

```

req_payload = {
 "name": "Log instance 2/1/23 22:29",
 "type": "LOG",
 "config": {
 "secured": {},
 "unsecured": {
 "log-level": "info",
 "max-log-size": 10,
 "max-log-age": 30,
 "max-log-backups": 20
 }
 }
}

resp = restclient.post('/connectors/63db5418e6ee1167a4c0986c/config',
 json_body=json.dumps(req_payload))
if resp.status_code == 200:
 parsed_resp = json.loads(resp.content)
 print json.dumps(parsed_resp)

```

### Sample response



```

{
 "mode": "TEST_AND_APPLY",
 "name": "Log instance 2/1/23 11:29",
 "root_scope_id": "63d98f45497d4f53005b24fa",
 "vrf_id": 1,
 "appliance_id": "63dad690e6ee1131f255e985",
 "connector_id": "63db5418e6ee1167a4c0986c",
 "deleted": false,
 "type": "LOG",
 "state": "PENDING",
 "attempts": 0,
 "config": {
 "secured": {},
 "unsecured": {
 "log-level": "info",
 "max-log-size": 10,
 "max-log-age": 30,
 "max-log-backups": 20
 }
 },
 "push_to_dio_at": 0,
 "created_at": 1675322272,
 "updated_at": 1675322272,
 "discovery_completed_at": 0,
 "committed_at": 0,
 "delete_at": 0,
 "error_at": 0,
 "hidden": false,
 "discovery_phase": 0,
 "internal": false,
 "id": "63db63a0f029813659f9fcf7"
}

```

## API to Delete a Config

This endpoint deletes a config from given Connector.

```
DELETE /openapi/v1/connectors/<id>/config/<config_id>
```

where <id> is the id that can be obtained from the [API to Get Connectors, on page 938](#), <config\_id> is the id that can be obtained from [API to Get the Configs on Connector Config Type, on page 941](#).

Parameters: The request URL contains the following parameters

| Name      | Type   | Description              |
|-----------|--------|--------------------------|
| id        | string | Specify the Connector ID |
| config_id | string | Specify the config ID    |

Response object: Returns the status of the config deleted for given Connector.

### Sample response

```

resp =
restclient.delete('/connectors/63c1272039042a1c0ddd3e20/config/63c1272039042a1c0ddd3e21')
if resp.status_code == 200:
 parsed_resp = json.loads(resp.content)
 print json.dumps(parsed_resp)

```

### Sample response

```

{
 "status": 200,

```

```

 "code": 1000,
 "message": "deleted"
 }
 •

```

## API to Get the Config

This endpoint gets the config with given ID

```
GET /openapi/v1/connectors/<id>/config/<config_id>
```

where <id> is the id that can be obtained from the [API to Get Connectors, on page 938](#), <config\_id> is the id that can be obtained from [API to Get the Configs on Connector Config Type, on page 941](#).

Parameters: The request URL contains the following parameters

| Name      | Type   | Description              |
|-----------|--------|--------------------------|
| id        | string | Specify the Appliance ID |
| config_id | string | Specify the config ID    |

Response object: Returns the config with given ID.

### Sample response

```
resp = restclient.get('/connectors/63db5418e6ee1167a4c0986c/config/63db5840f029813659f9fcf5')
```

```

if resp.status_code == 200:
 parsed_resp = json.loads(resp.content)
 print json.dumps(parsed_resp)

```

### Sample response

```

{
 "mode": "TEST_AND_APPLY",
 "name": "Log instance 2/1/23 22:29",
 "root_scope_id": "63d98f45497d4f53005b24fa",
 "vrf_id": 1,
 "appliance_id": "63dad690e6ee1131f255e985",
 "connector_id": "63db5418e6ee1167a4c0986c",
 "service_id": "63db5418e6ee1167a4c0986d",
 "deleted": false,
 "type": "LOG",
 "state": "COMMIT",
 "error_code": "NO_ERROR",
 "error_text": "",
 "attempts": 1,
 "config": {
 "secured": {},
 "unsecured": {
 "log-level": "info",
 "max-log-size": 10,
 "max-log-age": 30,
 "max-log-backups": 20
 }
 }
},
"push_to_dio_at": 1675319360,
"created_at": 1675319360,
"updated_at": 1675319364,
"discovery_completed_at": 0,
"committed_at": 1675319364,
"delete_at": 0,

```

```

 "error_at": 0,
 "hidden": false,
 "discovery_phase": 0,
 "internal": false,
 "id": "63db5840f029813659f9fcf5"
 }

```

## API to List Troubleshooting Commands Available for Connector

This endpoint returns the list of troubleshooting commands available for given Connector.

```
GET /openapi/v1/connectors/<id>/commands/available
```

where <id> is the id that can be obtained from the [API to Get Connectors, on page 938](#).

Parameters: The request URL contains the following parameters

| Name | Type   | Description              |
|------|--------|--------------------------|
| id   | string | Specify the Connector ID |

Response object: Returns the list of troubleshooting commands available for given Connector.

### Sample response

```

resp = restclient.get('/connectors/63c6f2701a49bd2bb0696cab/commands/available')
if resp.status_code == 200:
 parsed_resp = json.loads(resp.content)
 print json.dumps(parsed_resp)

```

### Sample response

```

[
 {
 "type": "LIST_SERVICE_FILES",
 "name": "List a directory in a service"
 },
 {
 "type": "CONTROLLER_PROFILING",
 "name": "Collect controller profile"
 },
 {
 "type": "SHOW_LOG",
 "name": "Show logs"
 },
 {
 "type": "SHOW_SERVICE_LOG",
 "name": "Show service logs"
 },
 {
 "type": "RESTART_CONNECTOR_CONTAINER",
 "name": "Restart the connector docker container"
 },
 {
 "type": "SHOW_SUPERVISOR_COMMANDS",
 "name": "Execute supervisorctl command"
 },
 {
 "type": "CONNECTOR_ALERT_INTERVAL_CONNECTOR",
 "name": "Override connector alert interval for Connector"
 },
 {
 "type": "SERVICE_PROFILING",
 "name": "Collect connector profile"
 }
]

```

```

 },
 {
 "type": "SNAPSHOT_CONNECTOR",
 "name": "Collect Snapshot from a connector"
 },
 {
 "type": "PACKET_CAPTURE",
 "name": "Packet capture"
 },
 {
 "type": "NETWORK_CONNECTIVITY_COMMANDS",
 "name": "Test network connectivity"
 },
 {
 "type": "SHOW_SERVICE_RUNNING_CONF",
 "name": "Show running configuration of a service"
 },
 {
 "type": "RESTART_CONNECTOR_SERVICE",
 "name": "Restart the connector service"
 },
 {
 "type": "SHOW_SYS_COMMANDS",
 "name": "Execute system command"
 }
]

```

## API to List Troubleshooting Commands

This endpoint returns the list of troubleshooting commands that are enabled for given Connector.

```
GET /openapi/v1/connectors/<id>/commands
```

where <id> is the id that can be obtained from the [API to Get Connectors, on page 938](#).

Parameters: The request URL contains the following parameters

| Name | Type   | Description              |
|------|--------|--------------------------|
| id   | string | Specify the Connector ID |

Response object: Returns the list of troubleshooting commands that are enabled for given Connector.

### Sample response

```

resp = restclient.get('/connectors/63db5418e6ee1167a4c0986c/commands')
if resp.status_code == 200:
 parsed_resp = json.loads(resp.content)
 print json.dumps(parsed_resp)

```

### Sample response

```

[
 {
 "appliance_id": "63dad690e6ee1131f255e985",
 "connector_id": "63db5418e6ee1167a4c0986c",
 "service_id": "63db5418e6ee1167a4c0986d",
 "state": "success",
 "level": "SERVICE",
 "command": "SHOW_LOG",
 "arg_string": "",
 "args": {
 "pattern": "info"
 }
 },

```

```

 "tailed": false,
 "rc": 0,
 "push_to_dio_at": 1675319615,
 "attempts": 1,
 "error_code": "NO_ERROR",
 "error_text": "",
 "deleted": false,
 "deleted_at": 0,
 "created_at": 1675319613,
 "total_chunk": 0,
 "id": "63db593df029813659f9fcf6"
 }
]

```

## API to Create a Troubleshooting Command

This endpoint creates a troubleshooting command available for given Connector.

POST /openapi/v1/connectors/<id>/commands

here <id> is the id that can be obtained from the [API to Get Connectors, on page 938](#). In request payload, <command> is a troubleshooting command type that can be obtained from [API to List Troubleshooting Commands Available for Connector, on page 945](#). <arguments> is a filled JSON object of the command schema, which can be obtained from [API to Get the Schema of Troubleshooting Commands, on page 911](#).

Parameters: The request URL contains the following parameters

| Name      | Type   | Description                                      |
|-----------|--------|--------------------------------------------------|
| id        | string | Specify the Connector ID                         |
| command   | string | Specify the command type                         |
| arguments | set    | Provide the filled command schema in JSON format |

Response object: Returns the troubleshooting command created for given Appliance.

### Sample response

```

req_payload = {
 "command": "SHOW_LOG",
 "arguments": {
 "pattern": "info"
 }
}
resp = restclient.post('/connectors/63db5418e6ee1167a4c0986c/commands',
json_body=json.dumps(req_payload))
if resp.status_code == 200:
 parsed_resp = json.loads(resp.content)
 print json.dumps(parsed_resp)

```

### Sample response

```

{
 "appliance_id": "63dad690e6ee1131f255e985",
 "connector_id": "63db5418e6ee1167a4c0986c",
 "service_id": "63db5418e6ee1167a4c0986d",
 "state": "pending",
 "level": "SERVICE",
 "command": "SHOW_LOG",
 "args": {
 "pattern": "info"
 }
}

```

## API to Delete a Troubleshooting Command

```

 },
 "tailed": false,
 "rc": 0,
 "push_to_dio_at": 0,
 "attempts": 0,
 "deleted": false,
 "deleted_at": 0,
 "created_at": 1675319613,
 "total_chunk": 0,
 "id": "63db593df029813659f9fcf6"
 }
}

```

## API to Delete a Troubleshooting Command

This endpoint deletes a troubleshooting command available for given Connector.

```
DELETE /openapi/v1/connectors/<id>/commands/<command_id>
```

where <id> is id that can be obtained from the [API to Get Connectors, on page 938](#), <command\_id> is the id that can be obtained from [API to List Troubleshooting Commands, on page 924](#).

Parameters: The request URL contains the following parameters

| Name       | Type   | Description              |
|------------|--------|--------------------------|
| id         | string | Specify the Connector ID |
| command_id | string | Specify the command ID   |

Response object: Returns the status of the troubleshooting command deleted for given Connector.

## Sample response

```

resp =
restclient.delete('/connectors/63c12e316419d0131767e21c/commands/63c10a0039042a6aee1b008c')

if resp.status_code == 200:
 parsed_resp = json.loads(resp.content)
 print json.dumps(parsed_resp)

```

## Sample response

```

{
 "status": 200,
 "code": 1000,
 "message": "deleted"
}

```

## API to Return a Troubleshooting Command

This endpoint returns the selected troubleshooting command for a given connector.

```
GET /openapi/v1/connectors/<id>/commands/<command_id>
```

where <id> is the id that can be obtained from the [API to Get Connectors, on page 938](#), <command\_id> is the id that can be obtained from [API to List Troubleshooting Commands, on page 924](#).

Parameters: The request URL contains the following parameters

| Name | Type   | Description              |
|------|--------|--------------------------|
| id   | string | Specify the Connector ID |

| Name       | Type   | Description            |
|------------|--------|------------------------|
| command_id | string | Specify the command ID |

Response object: Returns the selected troubleshooting command for a given connector.

### Sample response

```
resp =
restclient.get('/connectors/63db5418e6ee1167a4c0986c/commands/63db593df029813659f9fcf6')
 if resp.status_code == 200:
 parsed_resp = json.loads(resp.content)
 print json.dumps(parsed_resp)
```

### Sample response

```
{
 "appliance_id": "63dad690e6ee1131f255e985",
 "connector_id": "63db5418e6ee1167a4c0986c",
 "service_id": "63db5418e6ee1167a4c0986d",
 "state": "success",
 "level": "SERVICE",
 "command": "SHOW_LOG",
 "arg_string": "",
 "args": {
 "pattern": "info"
 },
 "tailed": false,
 "rc": 0,
 "push_to_dio_at": 1675319615,
 "attempts": 1,
 "error_code": "NO_ERROR",
 "error_text": "",
 "deleted": false,
 "deleted_at": 0,
 "created_at": 1675319613,
 "total_chunk": 0,
 "id": "63db593df029813659f9fcf6"
}
```

## API to Download the Output of the Connector Command as a File

This endpoint downloads the output of the command as a file.

```
GET /openapi/v1/connectors/<id>/commands/{command_id}/download
```

where <id> is the id that can be obtained from the [API to Get Connectors, on page 938](#), <command\_id> is the id that can be obtained from [API to List Troubleshooting Commands, on page 924](#). Not all commands have downloadable output, check the command schema that is provided by [API to Get the Schema of Troubleshooting Commands, on page 911](#), where "output\_type": "FILE" indicates it has downloadable content and "output\_ext" tells the file type.

Parameters: The request URL contains the following parameters

| Name       | Type   | Description              |
|------------|--------|--------------------------|
| id         | string | Specify the Connector ID |
| command_id | string | Specify the command ID   |

Response object: Returns the selected troubleshooting command for a given connector.

### Sample response

```
resp = restclient.download('downloadFile',
'/connectors/63c6ef42bca44e2b5e729191/commands/63cace941a49bd4c0e0cf45a/download')
```





# CHAPTER 16

## Configuration Limits in Secure Workload

The limits for various features in Cisco Secure Workload vary depending on the version and platform.

- [Cloud Connectors](#), on page 951
- [Connectors](#), on page 952
- [Label Limits](#), on page 953
- [Limits Related to Policies](#), on page 954
- [Additional Features](#), on page 955
- [Data-In or Data-Out](#), on page 956

### Cloud Connectors

| Cloud Connectors      | Metric                                            | Limit                  | Scale                         | Virtual Networks   | Kubernetes Clusters |
|-----------------------|---------------------------------------------------|------------------------|-------------------------------|--------------------|---------------------|
| AWS Connector         | Total number of flows exported by AWS connector   | 15000 flows per second | 5 accounts per connector      | 5 per account      | 5 per account       |
| Azure Connector       | Total number of flows exported by Azure connector | 15000 flows per second | 5 subscriptions per connector | 5 per subscription | 5 per subscription  |
| Google Cloud Platform | Total number of flows exported by GCP connector   | 15000 flows per second | 5 projects per connector      | 5 per project      | 5 per project       |



#### Note

- A maximum of 50 connectors, including cloud connectors, can be configured in a cluster across all tenants.
- The workloads managed by cloud connectors in Secure Workload require workload licenses, therefore, ensure that your total workloads are licensed and within the cluster limits.

# Connectors


**Note**

- A maximum of 50 connectors, including cloud connectors, can be configured in a cluster across all tenants.
- For limits applicable to individual connectors, see [What are Connectors](#).

| Connector            | Metric                                                                                  | Limit                                                                                                                                                                                |
|----------------------|-----------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| AnyConnect Connector | Total number of AnyConnect endpoints supported by one AnyConnect connector              | 5000 endpoints<br><b>Note</b> The number of AnyConnect endpoints across all AnyConnect Proxy sensors is limited by the number of sensors supported by the Secure Workload appliance. |
| AnyConnect Connector | Number of LDAP attributes that could be labelled on inventories of AnyConnect endpoints | 6 attributes                                                                                                                                                                         |
| AWS Connector        | Total number of flows exported by AWS connector                                         | 15000 flows per second                                                                                                                                                               |
| F5 Connector         | Total number of flows exported by F5 connector                                          | 15000 flows per second                                                                                                                                                               |
| NetFlow Connector    | Total number of flows exported by one NetFlow connector                                 | 15000 flows per second                                                                                                                                                               |
| NetScaler Connector  | Total number of flows exported by NetScaler connector                                   | 15000 flows per second                                                                                                                                                               |
| ERSPAN Connector     | Total number of flows exported by ERSPAN connector                                      | 15000 flows per second                                                                                                                                                               |

## Secure Workload Virtual Appliances for Connectors

| Appliance                        | Metric                                | Limit                 |
|----------------------------------|---------------------------------------|-----------------------|
| Secure Workload Ingest Appliance | Number of connectors on one appliance | 3                     |
|                                  | Number of appliances per root scope   | 100                   |
|                                  | Number of appliances per cluster      | 500                   |
| Secure Workload Edge Appliance   | Number of connectors on one appliance | 6                     |
|                                  | Number of appliances per root scope   | 1                     |
|                                  | Number of appliances per cluster      | Number of root scopes |

## Label Limits

Table 60: SaaS

| Feature      | Metric                                                                     | Limit                |
|--------------|----------------------------------------------------------------------------|----------------------|
| Label limits | Maximum number of IP Addresses that can be labeled per tenant (CMDB only). | 6,000 / 100 licenses |
|              | Maximum number of subnets that can be labeled per tenant (CMDB only).      | 120 / 100 licenses   |

## Limits Related to Policies

| Feature                                   | Metric                                                                                                               | Limit                                                    |
|-------------------------------------------|----------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------|
| Automatic policy discovery (formerly ADM) | Maximum number of member workloads (endpoints) allowed for automatic policy discovery run on a single scope.         | 10,000                                                   |
|                                           | Maximum number of conversations allowed for automatic policy discovery run on a single scope.                        | 10,000,000                                               |
|                                           | Maximum number of member workloads (endpoints) allowed for automatic policy discovery on a branch of the scope tree. | 37,500                                                   |
|                                           | Maximum number of conversations allowed for automatic policy discovery on a branch of the scope tree.                | 20,000,000                                               |
|                                           | Maximum number of total unique workloads (endpoints) allowed for automatic policy discovery run.                     | 15,000,000                                               |
|                                           | Maximum number of exclusion filters in Default Policy Discovery config.                                              | 100                                                      |
|                                           | Maximum number of exclusion filters allowed per workspace.                                                           | 100                                                      |
| Concrete policies                         | Aggregate size of policies on agents installed on non-Kubernetes workloads.                                          | 2.5 MB<br>(About 2000 policies, depending on complexity) |
|                                           | Aggregate size of policies on agents installed on Kubernetes nodes.                                                  | 7.5 MB<br>(About 6000 policies, depending on complexity) |

# Additional Features

| Feature        | Metric                                                                                                         | Limit                                                                                                                                                                                                                                                                                                                                                                                 |
|----------------|----------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Alerts         | Number of instances supported within a root scope                                                              | 256                                                                                                                                                                                                                                                                                                                                                                                   |
|                | Number of instances supported across root scopes                                                               | 1024                                                                                                                                                                                                                                                                                                                                                                                  |
|                | Number of latest alerts that are displayed per root scope (per status category- ACTIVE,SNOOZED, MUTED, CLOSED) | 5000                                                                                                                                                                                                                                                                                                                                                                                  |
|                | Maximum alert rate to preview in UI                                                                            | 60 per minute.<br><b>Note</b> If more than 60 alerts are sent per minute then UI will show a summary message indicating that alerts were sent to the DataTap but are suppressed in UI. Note that the 60 alerts per minute apply to the rate at which alerts are sent to datataps, and does not apply to the alert time nor event time and is unrelated to any specific batch of data. |
|                | Number of alerts configured per root scope (via modal)                                                         | 1000                                                                                                                                                                                                                                                                                                                                                                                  |
|                | Maximum number of alerts processed by Alerts App per minute batch                                              | 20000                                                                                                                                                                                                                                                                                                                                                                                 |
| Compliance App | Number of workspaces supported                                                                                 | 128                                                                                                                                                                                                                                                                                                                                                                                   |

# Data-In or Data-Out

| Feature   | Metric                                      | Limit | 8RU/39RU/SaaS/- |
|-----------|---------------------------------------------|-------|-----------------|
| Data Taps | Number of data taps supported per appliance | 10    | -               |