External Orchestrators

External orchestrators can be 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.

Currently supported external orchestrators:

Table 1: Currently supported 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.

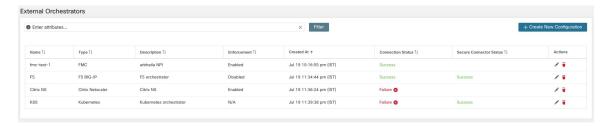| Type	Description/ When to use
Infoblox	To import networks, hosts, and A/AAAA records with extensible attributes from an Infoblox appliance with IPAM/DNS enabled. The imported extensible attributes can be used as labels in Secure Workload scopes and policies.
F5 BIG-IP	To read virtual server configurations 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.
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.

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 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:
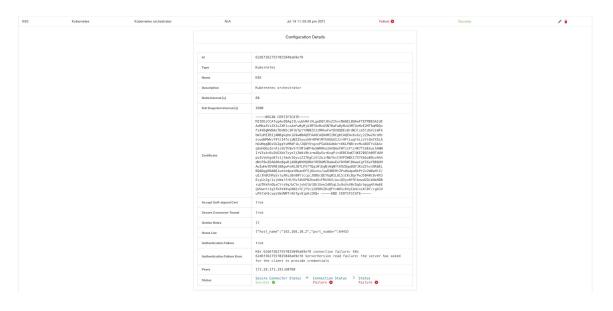
Figure 1: External orchestrator's main page

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.

The Secure Connector tunnel can be enabled 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**.
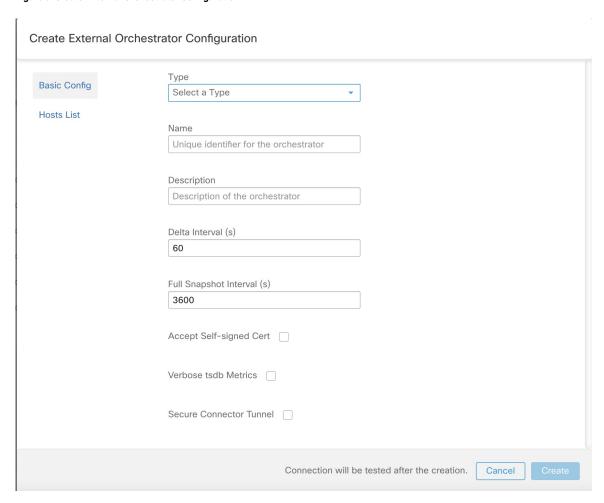
Figure 2: Example External Orchestrator Authentication Failure

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 3: Create External Orchestrator Configuration

Create External Orchestrator Configuration

Basic Config	**Type** Select a Type ▾
Hosts List	**Name** Unique identifier for the orchestrator
	Description Description of the orchestrator
	Delta Interval (s) 60
	Full Snapshot Interval (s) 3600
	Accept Self-signed Cert ☐
	Verbose tsdb Metrics ☐
	Secure Connector Tunnel ☐

Connection will be tested after the creation.   Cancel   Create

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.

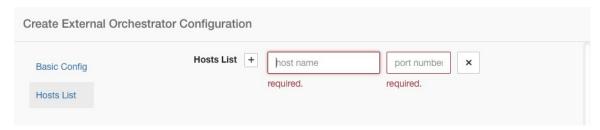| Common Field	Required	Description
Description	No	Description of the external orchestrator.
Full Snapshot Interval(s)	Yes	Interval in seconds the external orchestrator will try to import the full snapshot of configuration from the selected *Type*.
Accept Self-signed Cert	No	Check this option to accept self-signed server certificates for the HTTPS connection used by Secure Workload to retrieve configuration data from the selected *Type*. 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 4: 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.

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 5: 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 6: 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 7: Delete External Orchestrator

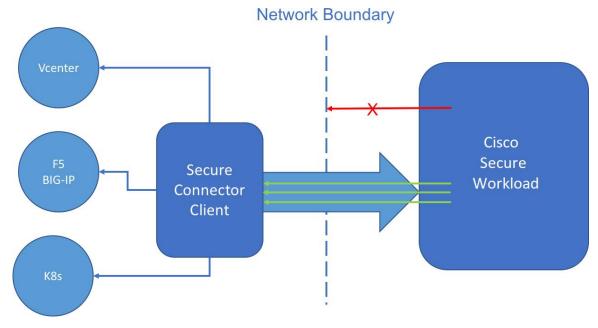| Name ↑↓	Type ↓	Description ↑↓	Enforcement ↑↓	Created At ↑↓	Connection Status ↑↓	Delete
mock_config	Vcenter	mockdata	N/A	Oct 14 03:41:19 am (EEST)	Success	🖉 🗑

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/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), 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 8: 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/private key pair and signs its public key certificate remotely by the server. A cryptograhic 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 tunneled 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

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

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 tetration-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 tetration-secure-connector`

[Optional] Deploy Specific Version of Secure Connector Client

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 Tokenendpoint). 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 .

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 9: 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:

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: • 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.

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 9.

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 - (*aws* in this case)
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 *region* 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.

Attribute	Description
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 *AWS Region* 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/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>*

Key	Value
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.
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 cluster-wide metrics are maintained.
Secureconnector Tunnel	Tunnel connections to this orchestrator's hosts through the Secure Connector tunnel

Field	Description
Use for policy discovery clustering	Include this Kubernetes cluster's label metadata when creating clusters during automatic policy discovery. This also turns on collection of additional resources such as replicasets and deployments to produce more accurate clusters. This option is currently available only for vanilla Kubernetes and OpenShift clusters. Cloud based Kubernetes services such as GKE, AKS and EKS are not yet supported.
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 only)
AWS Access ID	ACCESS KEY associated with the account for which orchestrator config is being created (Pre-existing EKS only)
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 only)
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 *region* values. :ref: https://docs.aws.amazon.com/general/latest/gr/rande.html. (Pre-existing EKS only)
AWS Assume Role ARN	Amazon resource number of the roleto assume while connecting to the orchestrator. :ref: https://docs.aws.amazon.com/STS/latest/APIReference/API_AssumeRole.html (Pre-existing EKS only)
Azure Tenant ID	Tenant ID associated with Azure subscription. (Pre-existing AKS only)
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
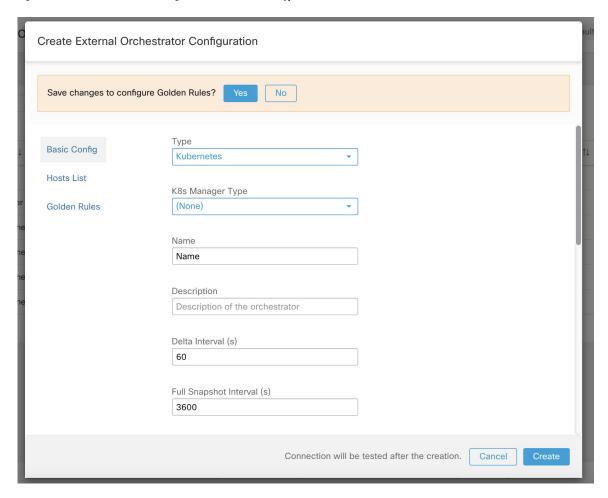
- Description: A string that describes the service

- 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 10: Create Golden Rules Configuration for Kubernetes Type

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: ZXlKaGJHY2lPaUpUVlX....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/certificate Credentials parsing/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/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/password, authtoken, client key/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 workaround 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/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 2: The following labels are instance specific.

Key	Value
orchestrator_system/workload_type	vm
orchestrator_system/machine_id	*BIOS UUID of bare metal/VM*
orchestrator_system/machine_name	*Hostname of the bare metal/VM*
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 succesful 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*:

Figure 11: Example Infoblox labels

```
1. orchestrator_Department = AES789
2. orchestrator_system/cluster_id = ▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓
3. orchestrator_system/cluster_name = scale13-ib
4. orchestrator_system/machine_id =
   record:host/▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓:client8/%20
5. orchestrator_system/machine_name = client8
6. orchestrator_system/orch_type = infoblox
```

Prerequisites

- Infoblox REST API endpoint supporting WAPI version 2.6, 2.6.1, 2.7, 2.7.1 (recommended)

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	The hosts list denotes one Infoblox grid, ie. more than one grid members with REST API access can be added, and the external orchestrator will switch over to the next one in the list in case of connection errors. If you want to import labels from another Infoblox grid, create a new external orchestrator for it.

Note For Infoblox external orchestrator, IPv4 and IPv6 (dual stack mode) addresses are supported. However, do note that dual stack support is a BETA feature.

Workflow

- First, the user must verify that the Infoblox REST API endpoint is reachable from the Secure Workload cluster.

- For TaaS or in cases, where the Infoblox server is not directly reachable, the user must configure a Secure Connector tunnel to provide connectivity.

- Create an external orchestrator with type *Infoblox*. Depending on the volume of Infoblox data, ie. the number of subnets, hosts and A/AAAA records it can take up to one hour for the first full snapshot is available in Secure Workload.

- 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 *policy enforcement* 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

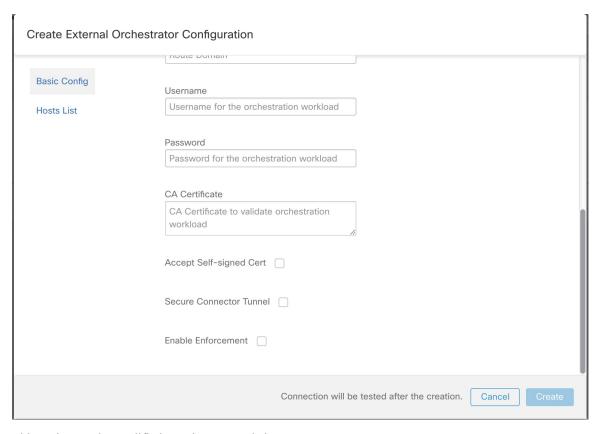For each virtual server the external orchestrator will generate the following labels:

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 12: Configuration Option "Enable Enforcement"

Create External Orchestrator Configuration

Basic Config

Hosts List

Route Domain

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.   Cancel   Create
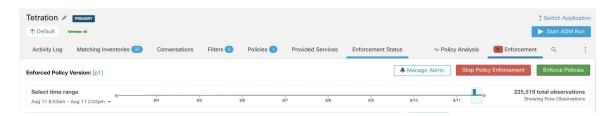
This option can be modified any time as needed.

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 13: 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 will enforce 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 policy using F5 ingress controller.

Step 1 Create an external orchestrator for *F5 BIG-IP* load balancer as described above.

Step 2 Create an external orchestrator for Kubernetes/OpenShift as described here.

Step 3 Create an ingress object in the Kubernetes cluster. A snapshot of the yaml file used to create 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 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 reate a backend service which will be accessed by the consumers outside the cluster. In the example provided below we have created a *nginx* service.
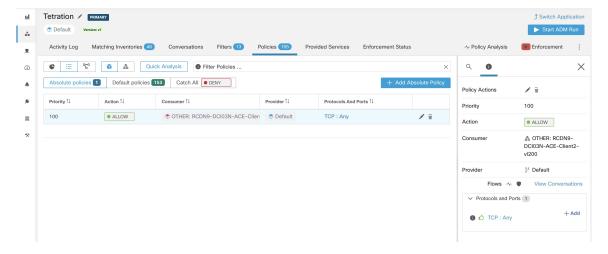
```
→  ~
→  ~ k8s get deploy
NAME    DESIRED  CURRENT  UP-TO-DATE  AVAILABLE  AGE
nginx   1        1        1           0          5s
→  ~
```
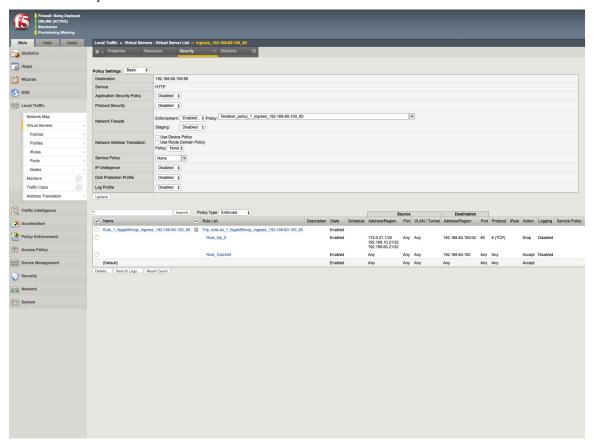
Step 6 Create a policy between external consumer and backend service. Enforce the policy using *Policy Enforcement* tab.

Policy enforcement for F5 ingress controller

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 the SNIP [in case SNAT pool is enabled] or F5 IP [auto map enabled] and destination will be backend pod IP.

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 *policy enforcement* 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

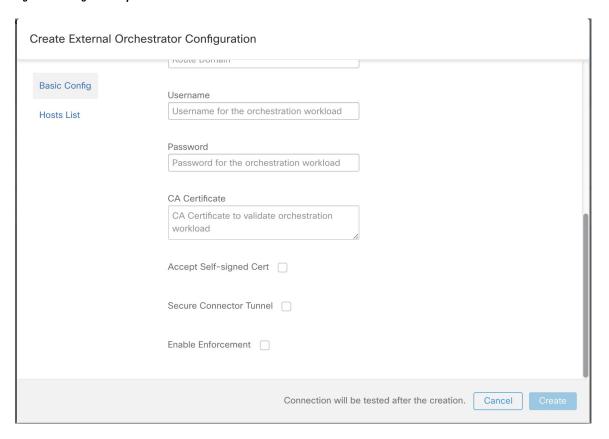For each load balancing virtual server the external orchestrator will generate the following labels:

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:
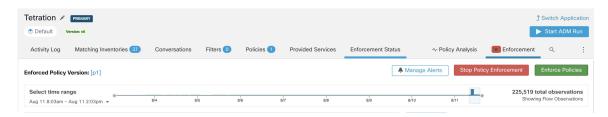
Figure 14: Configuration Option "Enable Enforcement"

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 15: 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 verion: 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 cer- tificate
Secureconnector Tunnel		Tunnel connections to this orchestrator's hosts through the Secure Con- nector 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	*TAXII*
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.

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 Intrusion Prevention System (IPS) protection to workloads where software agents are installed. You can use this method to protect your application from malicious traffic entering the system. When you configure virtual patching on Secure Workload, it publishes the Common Vulnerabilities and Exposures (CVEs) to the Cisco Secure Firewall to consider 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.

Orchestrator generated labels

None - The Secure Firewall Management Center external orchestrator does not generate any user annotations.