



## Network Flows-Traffic Visibility

This chapter discusses how Secure Workload captures flows between workloads and proxy servers, providing insights into proxied traffic. The feature focuses on flow observations, which are per-minute data aggregations of unique network flows. The flow observations help identify malicious traffic and enforce security policies, additionally, allows users to filter and examine flow data. It also introduces the visibility and enforcement of defined malicious IPv4 addresses. This feature lets users identify and block traffic to these addresses using predefined filters, updated every 24 hours to help identify threats.

The chapter explains components like the Corpus Selector, Columns and Filters, and Top N Charts. These tools help users select and visualize specific datasets for detailed network flow analysis. The Corpus Selector can handle up to 2 billion flow observations. Client-Server Classification is vital for policy discovery and enforcement. It relies on accurately identifying the client and server in a flow. This process is improved with deep visibility or enforcement agents, which enhance detection accuracy. This document also covers Conversation mode, which is a simpler alternative to Detailed Mode. In Conversation mode, only conversations are reported rather than individual flows, therefore reducing computational demands. This mode is useful for segmentation tasks.



**Attention** 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.

**Table 1: Feature Information**

Feature Name	Release	Feature Description	Where to Find
Visibility and Enforcement of Well-known IPv4 Malicious Traffic	Secure Workload 3.9	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 20</a>

- [Visibility Into Network Traffic Flows, on page 2](#)
- [Network Traffic Flows, on page 2](#)
- [Corpus Selector, on page 3](#)
- [Columns and Filters, on page 3](#)
- [Filtered Time series, on page 9](#)
- [Top N Charts, on page 10](#)
- [Observations List, on page 11](#)
- [Explore Observations, on page 13](#)
- [Client-Server Classification, on page 15](#)
- [Conversation Mode, on page 17](#)
- [Visibility in Proxied Flows, on page 19](#)
- [Visibility of Well-Known Malicious IPv4 Addresses, on page 20](#)

## Visibility Into Network Traffic Flows

On the Secure Workload UI, from the navigation pane, choose **Investigate** > **Traffic** that takes you to the **Flow Search** page. The **Flow Search** page provides the means for quickly filtering and drilling down into the flows corpus. The basic unit of flow search is **Flow Observation**, which is a per-minute aggregation of each unique flow.

## Network Traffic Flows

There are two sides of a traffic flow - **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.



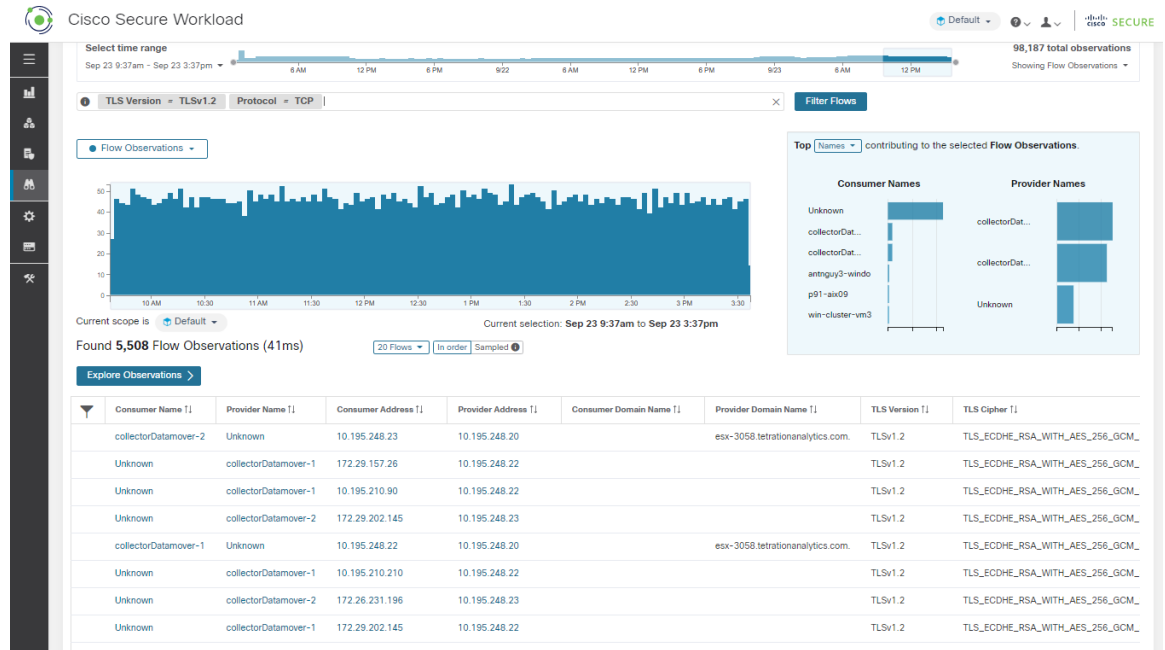
---

**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 1: Flows Overview



# Corpus Selector

Figure 2: 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.

# Columns and Filters

Figure 3: Filter Input

 Filter Flows

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 4: Filter Input Supports for Consumer Address

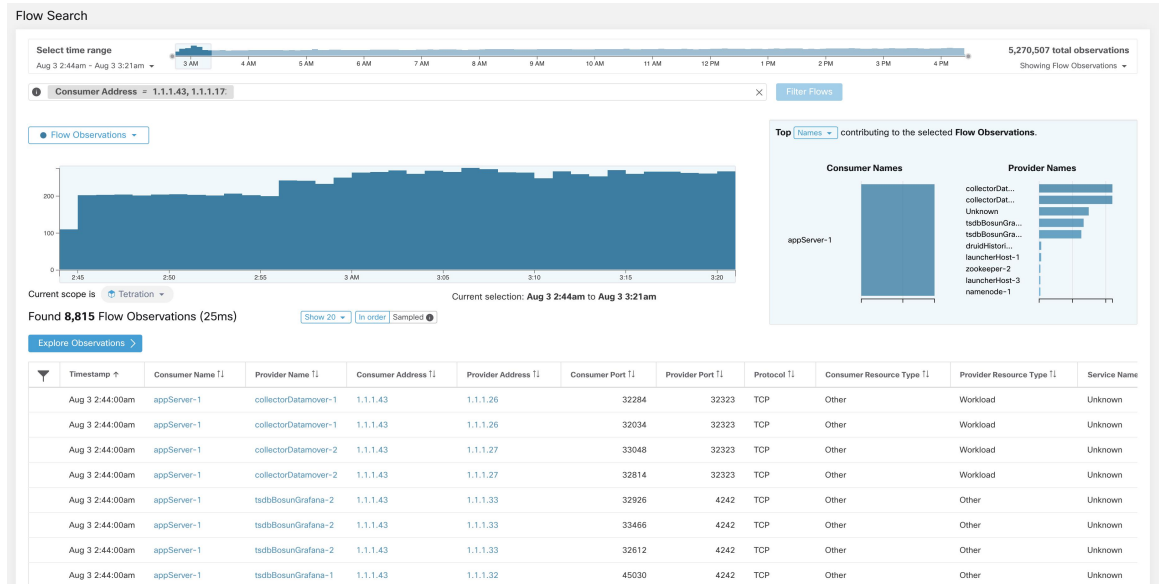


Figure 5: Filter Input Supports Range Query for Consumer Address

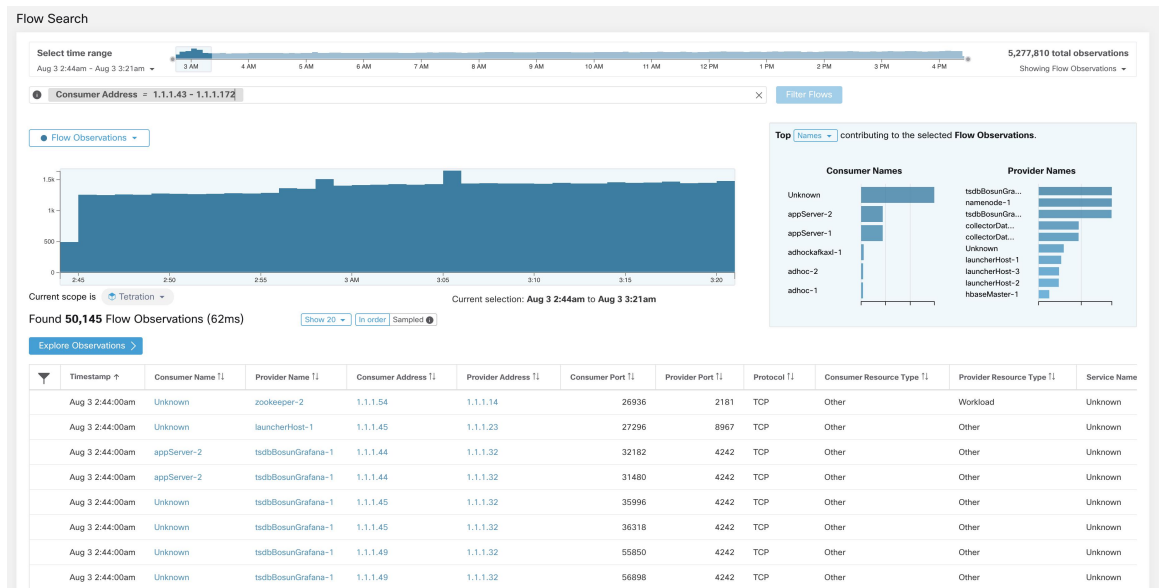


Table 2: Available Columns and Filters

Columns (Names exposed in API)	Description	Source
<b>Consumer Address</b> ( <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
<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

Columns (Names exposed in API)	Description	Source
<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
<b>Consumer Enforcement Group</b> ( <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> ( <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> ( <i>src_subdivision</i> )	Matches flows whose Consumer subdivision overlaps with the provided subdivision (state).	Internal
<b>Provider Subdivision</b> ( <i>dst_subdivision</i> )	Matches flows whose Provider subdivision overlaps with the provided subdivision (state).	Internal

Columns (Names exposed in API)	Description	Source
<b>Consumer Autonomous System Organization</b> ( <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> ( <i>dst_autonomous_system_organization</i> )	Matches flows whose Provider autonomous system organization overlaps with provided autonomous system organization (ASO).	Internal
<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> ( <i>fwd_process_owner</i> )	Filter flow observations by process owner UID (root, admin, yarn, mapred).	Software Agents
<b>Rev Process UID</b> ( <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

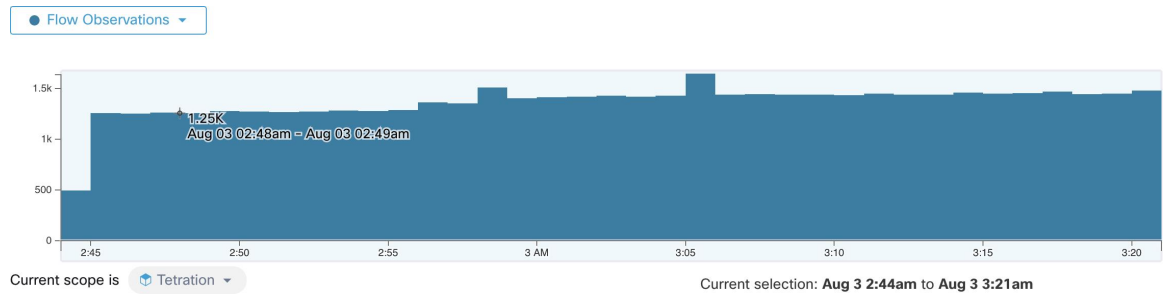
Columns (Names exposed in API)	Description	Source
<b>Packets</b>	Filter flow observations by Packet traffic bucket. Matches flows which Packet traffic bucket values are =, <, > (bucketed by powers of 2 (0, 2, 64, 1024)).	Software Agent and Ingest Appliances
<b>Flow Duration (µs)</b>	Filter flow observations 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) (<i>srtt_dim_usec</i>)</b>	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> ( <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> ( <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.

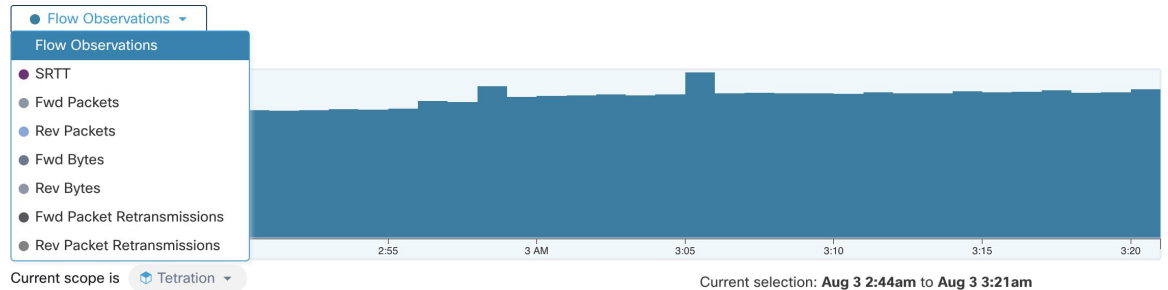
## Filtered Time series

**Figure 6: Filtered Time series**



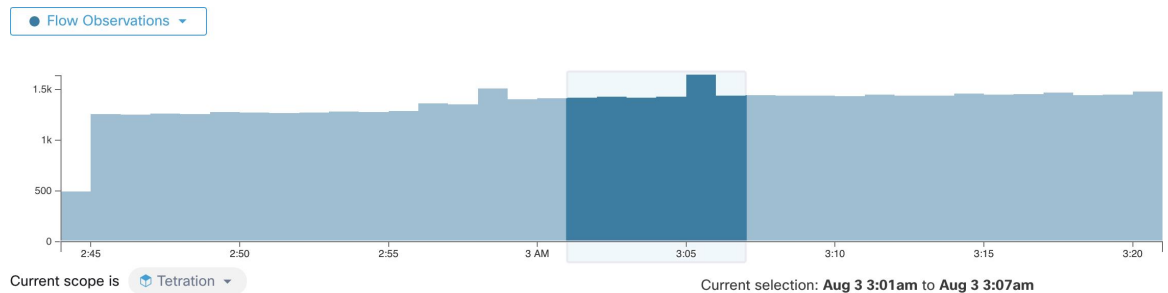
This component displays the aggregated totals of various metrics for the interval selected (the selection made in the above [Corpus Selector, on page 3](#)). Use the dropdown to change which metric is displayed.

**Figure 7: Time series dropdown**



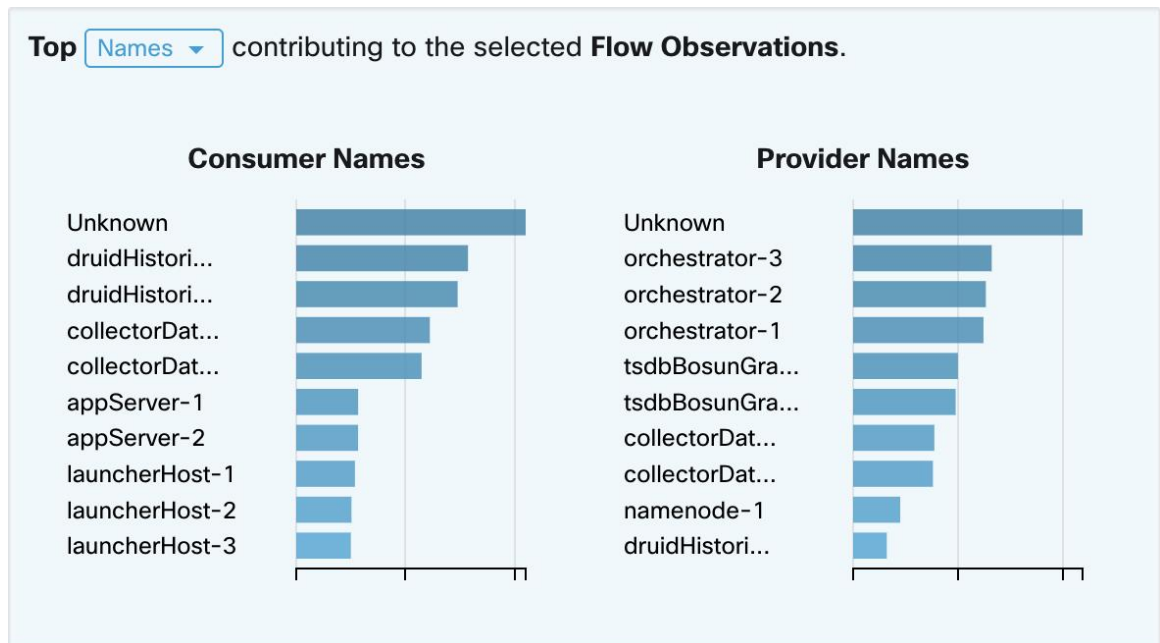
Further-narrowing of the selected interval can also be done in this component. Click the area of the chart that you'd like to focus on, and the Top N Charts and the data below will all be updated to include only data from that selected interval.

**Figure 8: Time series with selection**



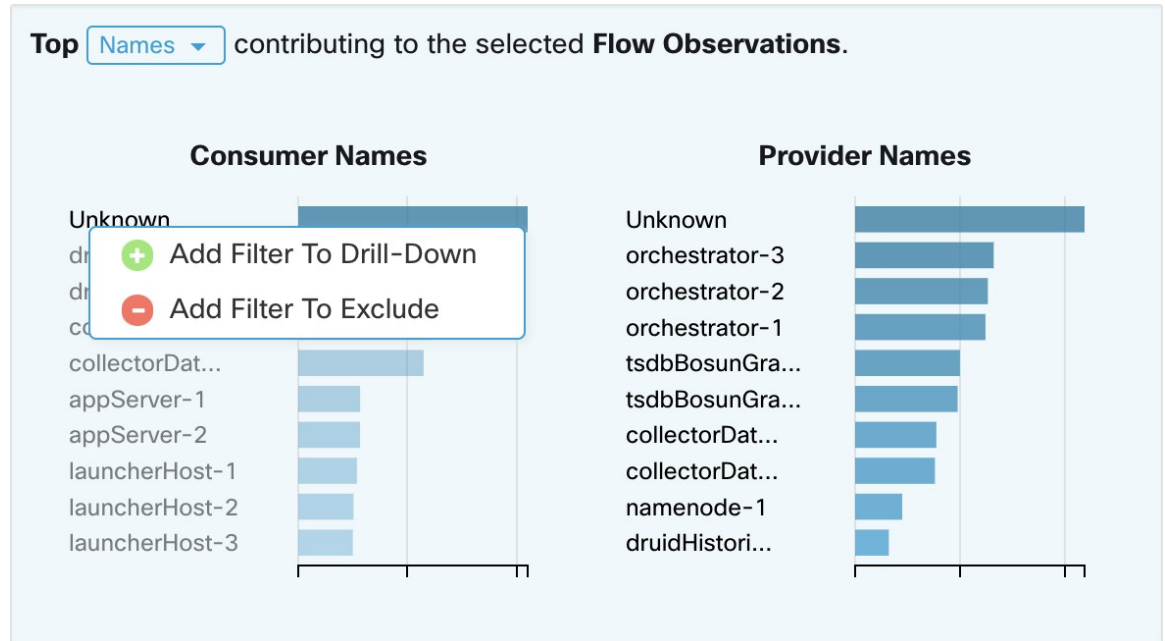
# Top N Charts

Figure 9: 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 10: 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 11: Sampled

Found 5,917 Flow Observations (95ms) Show 20 In order Sampled

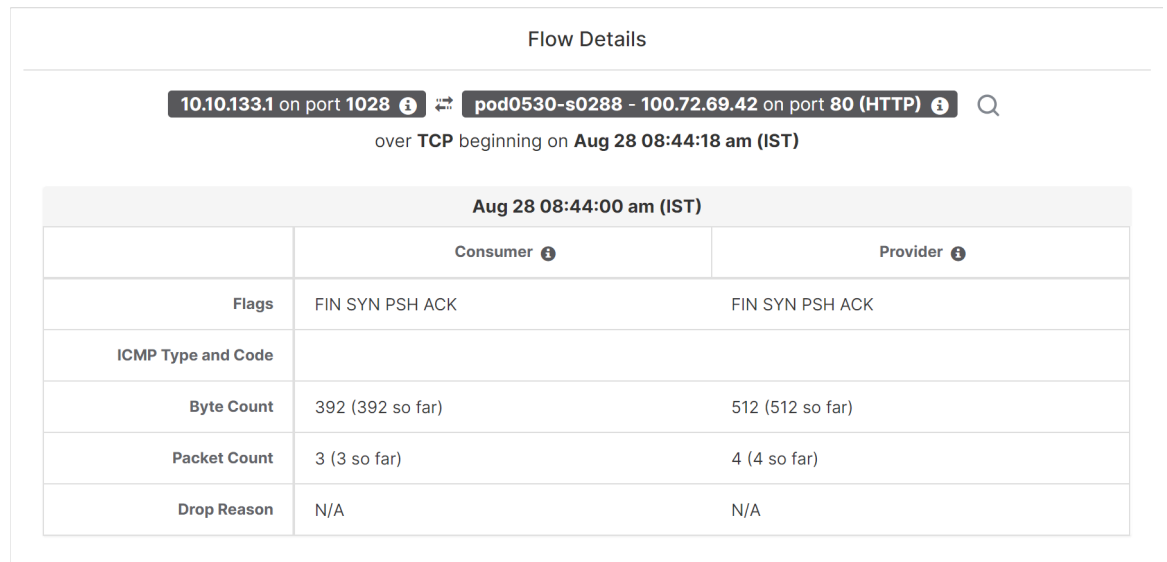
[Explore Observations](#)

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

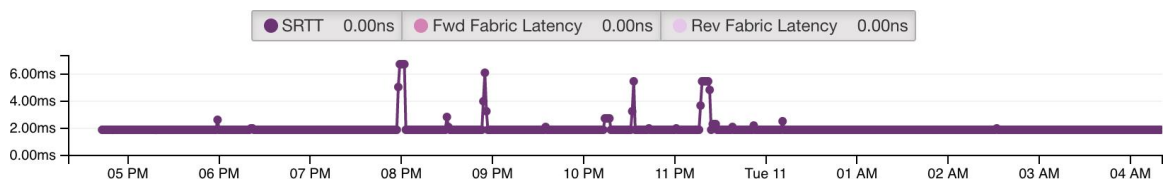
Click on any of the rows to expand the **Flow Details** section. This displays a summary of the flow and charts for various metrics for the lifetime of that flow. For long-lived flows, a summary chart is displayed at the bottom that allows you to choose different intervals for which to view time series data.

Figure 12: Flow details



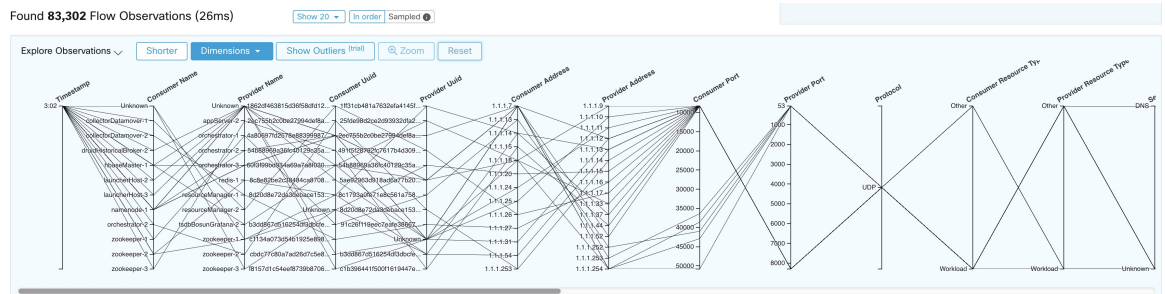
For flows labeled with Fabric Path information, **Fwd/Rev Fabric Latency** and **SRRT** are available. Time series charts for other metrics such as **Fwd/Rev Burst Indicators** and **Fwd/Rev Burst+drop Indicators** may be displayed if available. See [Visibility Warning](#).

Figure 13: Latency



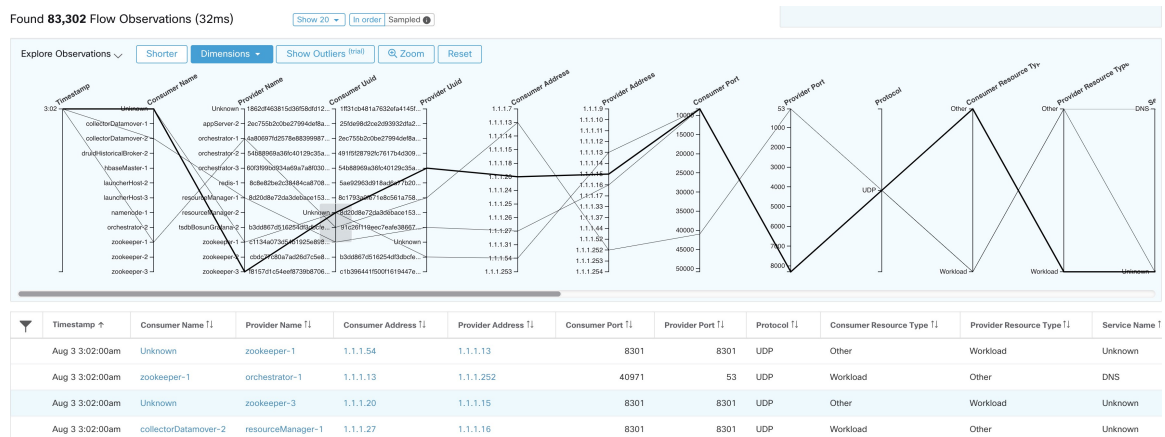
# Explore Observations

Figure 14: 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 15: Flow Observation hovered



[Download table data as JSON](#)

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 16: 1000 In-order

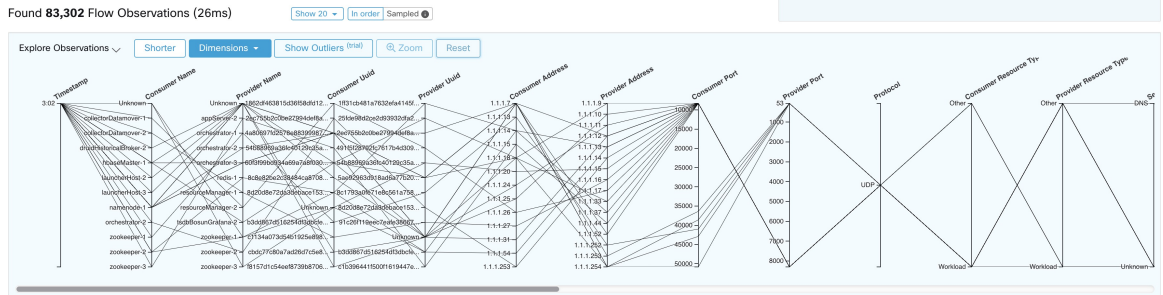
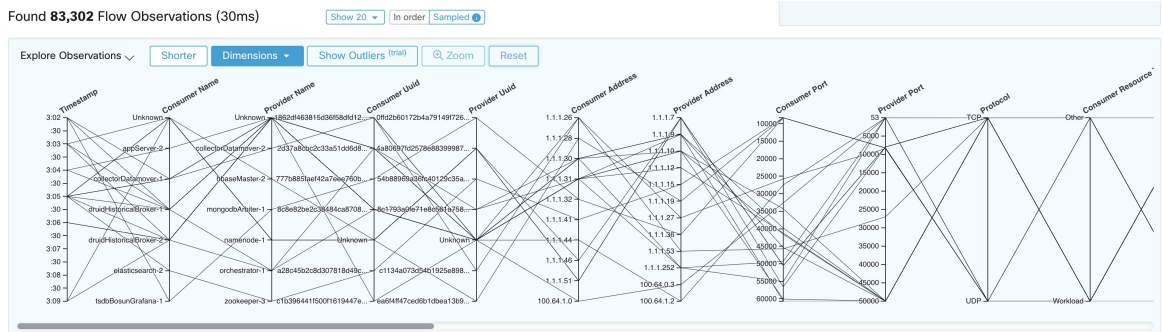


Figure 17: vs. 1000 sampled

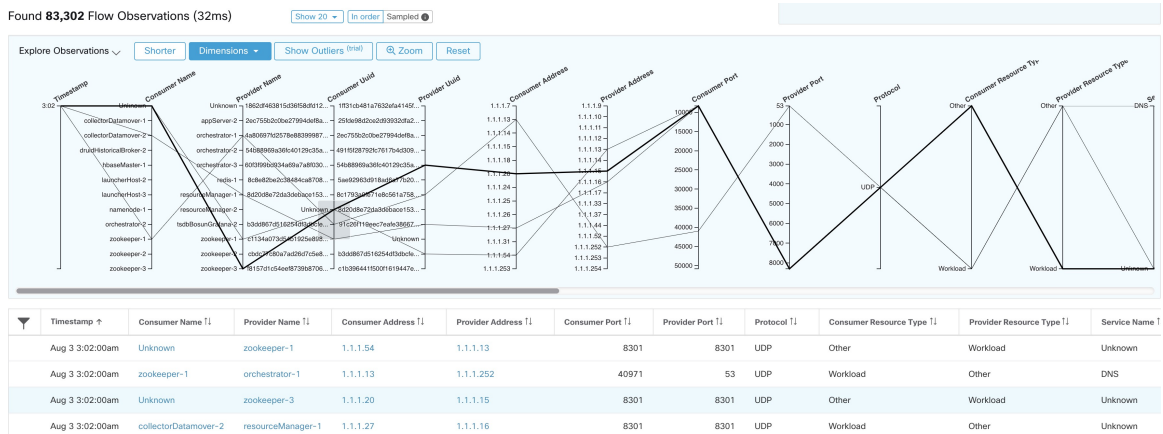


Notice how the **Timestamp** for all of the in-order observations is from 9:09 and how the observations are evenly distributed through the selected interval in the sampled version.

### 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 18: Explore with selection



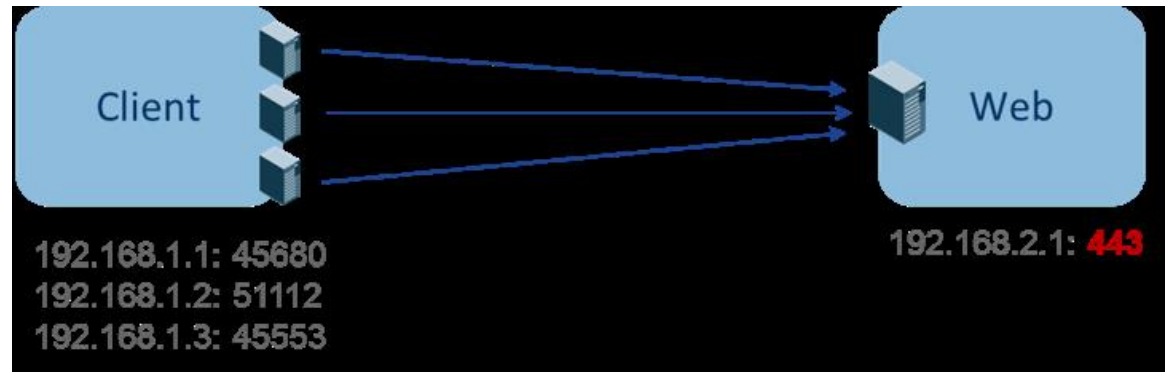
Download table data as JSON

## 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 19: 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 20: 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 21: 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, see [Client Server Configuration](#). You can also manually define policies, examine or remove the undesired policies. For more information on how to define or remove undesired policies, see [Policies](#).

## 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:** The **Detailed Mode** captures every observed flow by the agent along with detailed statistics. Statistics captured are packet and byte counts, TCP flags, connection statistics, network latency, SRTT, and so on. While this type of reporting is desirable in numerous cases, it is computationally intensive to report and process the data. Additionally, it may not be strictly required when the primary use case is segmentation.
- **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 during the client-server classification. This is applicable to TCP, UDP, and ICMP flows.

In the detailed mode, for TCP/UDP flows, 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( $\mu$ s)
- 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.

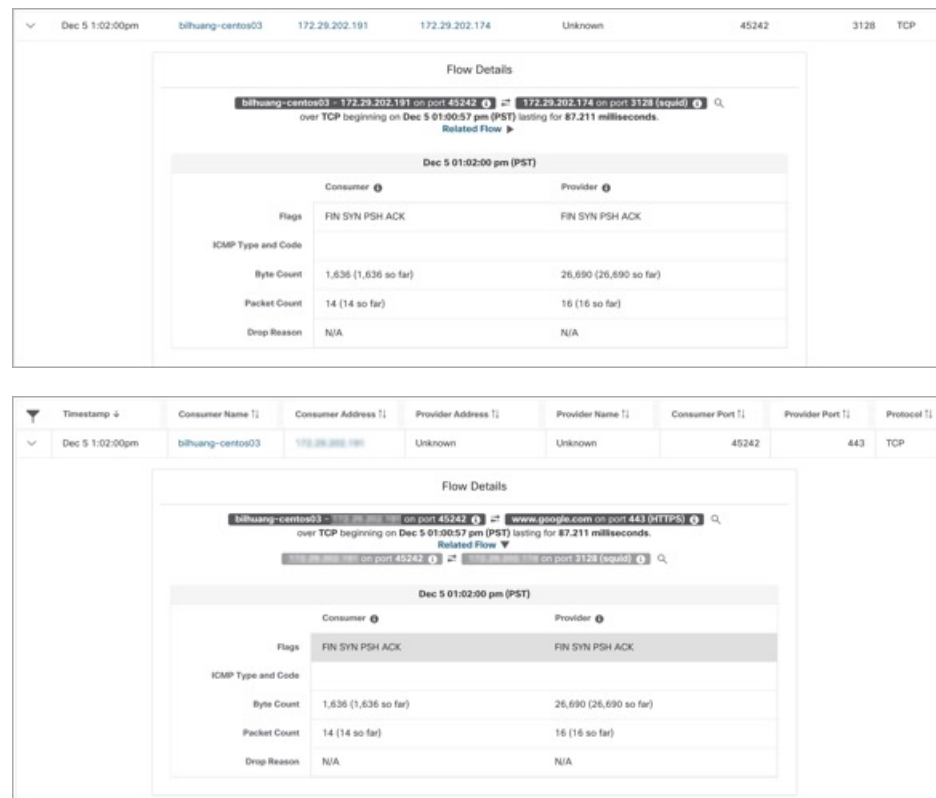
Agents of Version 3.9 and later can capture the destination of proxied flows. On the **Investigate** > **Traffic** page, you can observe two distinct flows:

- **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 22: Flow Details



## Visibility of Well-Known Malicious IPv4 Addresses

The Secure Workload threat intelligence data packs are updated every 24 hours with well-known malicious IPv4 addresses. Traffic originating from a consumer or provider is analyzed against these malicious IPv4 addresses. This analysis helps identify workloads connecting to these malicious IPv4 addresses, on the **Flow Search** page. To filter flows connecting to well-known malicious IPv4 addresses, use the following query:

**Malicious? = true, Provider Malicious? = true, or Consumer Malicious? = true**

The **Malicious inventories** filter is automatically created and populated from the datapack downloads. You can use this inventory filter to create, analyze, and enforce policies on your workloads to block traffic from malicious IPv4 addresses. For more information, see [Create and Discover Policies](#) and [Enforce Policies](#).



---

**Note** You cannot view the contents of the inventory filter because it is continuously updated with the most significant threats from the datapack.

---

