Illumio Adaptive Security Platform Advanced Command-Line Interface Tool 1.1.0 Guide

12/12/2018

Table of Contents

Product Version

Illumio® Adaptive Security Platform® Advanced Command-Line Interface Tool version 1.1.0 is compatible with Illumio Adaptive Security Platform (ASP), versions 17.1 and later.

The Illumio ASP PCE Advanced Command-Line Interface (CLI) tool version numbering is independent from the release and version numbering of Illumio ASP PCE and VEN. The CLI tool works with multiple versions of the PCE and the VEN and does not necessarily need software changes in parallel with releases of the PCE or the VEN.

About Illumio

Illumio products and services are built on Illumio's patented technologies. For more information, see Illumio Patents.

Illumio Professional Services for Deployment

To ensure optimal deployment of the Illumio Adaptive Security Platform, contact your Illumio Professional Services representative.

Preview Features Only for Evaluation Before General Availability

Any preview features in this release of Illumio Adaptive Security Platform are for your evaluation only.

> ⚠ **Do not deploy preview features in a production environment**
> Be sure to install these preview features only on non-production systems. To avoid inadvertently impacting your current operations, do *not* install the preview features on production systems.
> The purpose of preview features is to make them more useful for your needs before general availability.

Illumio welcomes your comments and suggestions for improving preview features and documentation. For more information and to send feedback, contact Illumio Customer Support.

Illumio Adaptive Security Platform Training

Illumio offers a wide yet focused training curriculum for Illumio Adaptive Security Platform, from beginning to advanced topics.

To see available courses, log into your Illumio support account and select the **Training** tab.

Search Knowledge Base and Documentation

For useful short articles about Illumio Adaptive Security Platform, log into your Illumio support account and select the **Knowledge Base** or **Documentation** tabs.

Illumio Adaptive Security Platform Support

If you cannot find what you are looking for in this document or in support Knowledge Base and Documentation, contact us at:

- support@illumio.com
- +1-888-631-6354
- +1-408-831-6354

Recommended Skills

Illumio recommends that you be familiar with the following:

- Your organization's security goals.
- Solid understanding of Illumio ASP.
- General computer use, Linux shell (bash), Windows PowerShell, or both.

Related Documentation

Illumio® Adaptive Security Platform® documentation is available from the Support portal.

- *Policy Compute Engine (PCE) Web Console Guide*: working with Illumination®, designing security policy, and provisioning and administering VENs.
- *Policy Compute Engine (PCE) Deployment Guide*: planning and installing the PCE.
- *Policy Compute Engine (PCE) Operations Guide:* common management tasks of the PCE.
- *Advanced Command-line Tool Interface Guide*: common PCE-related tasks to use on your local computer.
- *Policy Compute Engine (PCE) Supercluster Deployment and Usage Guide*: designing, deploying, and managing the PCE Supercluster of multiple, distributed standard PCE clusters.
- *Policy Compute Engine (PCE) REST API Guide*: web-programming Illumio Adaptive Security Platform.
- *Virtual Enforcement Node (VEN) Deployment Guide:* installing and activating the VEN, including PCE-based distribution of the VEN and on-worload installation and management
- *Virtual Enforcement Node (VEN) Operations Guide*: common management tasks of the VEN.

- *Auditable Events and SIEM Integration Guide*: analyzing significant events on the PCE and VEN and securely transferring event records to analytics or Security Information and Event (SIEM) systems.

Notational Conventions

- Newly introduced terminology is *italicized*. Example: *activation code* (also known as *pairing key*).
- `Command-line` examples are `monospace`. Example: `illumio-ven-ctl --activate`.
- Arguments on command lines are *`monospace italics`*. Example: `illumio-ven-ctl --activate` *`activation_code`*.
- In some examples, the output might be shown across several lines but is actually on one single line.
- Command input or output lines not essential to an example are sometimes omitted, as indicated by three periods in a row:
  `...`
  some command or command output
  `...`
- References to section titles in this guide are in double quotation marks. Example: See "Basic Theory of Operation".
- Reference to other guides in the Illumio library are *italicized*. Example: See the *PCE Web Console User Guide*.

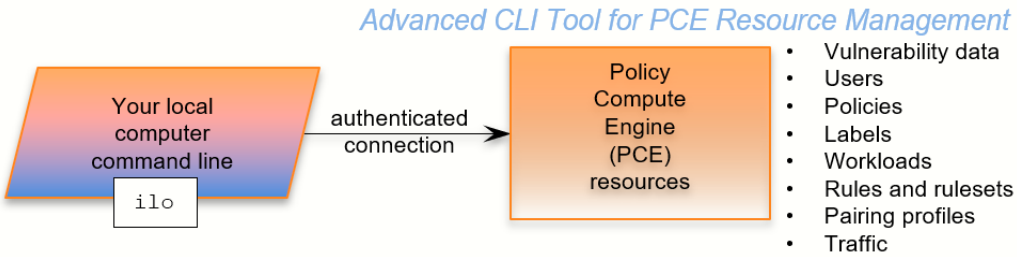
How to Use This Guide

This guide includes several major sections:

- Overview to the CLI Tool.
- Installation.
- Formal syntax of the `ilo` command.
- Tutorials for various operations.
- Uploading vulnerability data.
- Security policy import and export.

Overview to the Command-Line Interface Tool

With the Illumio ASP CLI tool, you can manage many of your PCE's resources directly from your local computer.



Some purposes of the CLI tool include the following:

- Import vulnerability data for analysis with Illumination®.
- Help with tasks such as directly importing workload information to create workloads in bulk.
- Create, view, and manage your organization's security policy rules, rulesets, labels, and other resources.

> **Exercise caution**
> The CLI tool is a powerful way to work with your PCE resources. Exercise caution to make sure your use of the tool does not adversely affect your system. If possible, test your CLI tool commands against a non-production system before using them on your production PCEs.

The CLI tool is named `ilo`. It is a wrapper around the Illumio Adaptive Security Platform REST API. No knowledge of the REST API is required.

Prerequisites and Installation

This section details prerequisites and the installation of the CLI tool. A checklist is below.

- License for vulnerability data upload.
- Vulnerability data for upload.
- Functional PCE.
- Supported operating systems.
- TLS/SSL certificate for authenticating to the PCE.
- The CLI tool installer program.

License Required for Upload of Vulnerability Data

An Illumio ASP Vulnerability Maps license is required to import vulnerability data into the Illumio PCE. For information about obtaining license, contact Illumio Customer Support. For details on activating the license, see "Adding the License for Vulnerability Data Upload".

Vulnerability Data to Upload

If you plan on using the CLI tool to upload vulnerability data, make sure you have the data to upload in advance. See "Supported vulnerability data sources".

Functional PCE Installed

Because the CLI tool is for managing resources on your PCE, you need to have already installed a fully functional PCE.

Supported Operating Systems on Your Local Computer

The CLI tool is supported on the following operating systems.

Linux

- RHEL/CentOS, versions 6 and 7
- Ubuntu, versions 14.04 and 16.04

Microsoft Windows

> **Windows 64-bit CPU Architecture required**
> The CLI tool is not supported on Windows 32-bit CPU architecture. Ensure that you run it on Windows 64-bit CPU architecture.

- Microsoft Windows version 2008R2, 64-bit
- Microsoft Windows version 2012, 64-bit
- Microsoft Windows 10, 64-bit

TLS/SSL Certificate for Access to the PCE

You need a TLS/SSL certificate to securely connect to the PCE. Requirements for this certificate are detailed in the *PCE Deployment Guide*.

Alternative Trusted Certificate Store

To secure the connection to the PCE, by default, the CLI relies on your computer's trusted certificate store to verify the PCE's TLS certificate. You can specify a different trusted store. If you have installed a self-signed certificate on the PCE, the alternative trusted store might be necessary.

Example: Set envar for alternative trusted certificate store

```
export ILO_CA_FILE=~/self-signed-cert.pem
```

CLI Tool Install, Upgrade and Uninstall

Download the Installation Package

Download the CLI tool installation package from the Tools Catalog page to a convenient location on your local computer.

Linux Install the CLI Tool

The CLI installer for Linux is delivered as an RPM for RedHat/CentOS and DEB for Debian/Ubuntu .

The CLI tool is installed in the local binaries directory `/usr/local/bin`.

Log into your local Linux computer as a normal user and then use `sudo` to run one of the following commands.

Operating System	Commands to Install the CLI Tool
RedHat/CentOS	`$ sudo rpm -ivh /path_to/nameOfCliRpmFile.rpm`
Debian/Ubuntu	`$ sudo dpkg -i / path_to / nameOfCliDebFile .deb`

Linux Upgrade the CLI Tool

Log into your local Linux computer as a normal user and then use `sudo` to run one of the following commands.

Operating System	Commands to Install the CLI Tool
RedHat/CentOS	`$ sudo rpm -Uvh /path_to/nameOfCliRpmFile.rpm`
Debian/Ubuntu	`$ sudo dpkg -i / path_to / nameOfCliDebFile .deb` Note: The same option `-i` is used for installation or upgrade.

Linux Uninstall the CLI Tool

Log into your local Linux computer as a normal user and then use `sudo` to run one of the following commands.

Operating System	Commands to Install the CLI Tool
RedHat/CentOS	`$ sudo rpm -e nameOfCliRpmFile`
Debian/Ubuntu	`$ sudo dpkg -r nameOfCliDebFile`

Windows Install the CLI Tool

The CLI installer for Windows is delivered as an `.exe` file.

Log into your local Windows computer as administrator and start the installation program in any of the following ways.

- In the Windows GUI, double-click the `.exe` file.
- In a `cmd` window, run the `.exe`.
- In a PowerShell window, run the `.exe`.

After starting the installation program, follow the leading prompts.

A successful installation ends with the message **Installation Successfully Completed** and the help text for the CLI tool is displayed.

Windows Upgrade the CLI Tool

CLI 1.1 cannot be directly upgraded from an existing CLI 1.0 installation.

If you have already installed CLI 1.0, first, manually uninstall it with the Windows Control Panel's **Add/Remove Programs**.

After uninstalling CLI 1.0, install CLI 1.1 as described in "Install the CLI Tool on Windows".

Windows Uninstall the CLI Tool

Log into your local Windows computer as administrator, and from the Windows Control Panel, launch **Add/Remove Programs**.

Select **Illumio CLI** from the list and click **Uninstall** button.

The ilo Command

The CLI tool is named `ilo`.

Formal Syntax

The formal syntax for the `ilo` command is as follows:

```
ilo  resource_or_specialCommand  argument options_for_resourceOrSpecialCommand
```

where:

- *`resource_or_specialCommand`* represents either a resource managed by the PCE or a non-resource.
  - A *resource* is an object that the PCE manages, such as a workload, labels, and pairing profiles.

Example resource command on Linux: create a workload

```
ilo workload create --name FriendlyWorkloadName --hostname myWorkload.BigCo.com
```

  - A *`specialCommand`* is *not* a resource, such as `user`, `login`, `use_api_key`, and `node_available`.

Example special command on Windows: logout of PCE

```
ilo user logout --id 6
```

- The *`argument`* represents an operation on the resource or special command.
- The *`options_for_argument`* are allowed option for the *`resource_or_specialCommand`*. That is, the specific option depends on the type of resource or special command.

CLI Tool Help

The `ilo` command without options displays the high-level syntax of special commands, resources, and their allowable options. This high-level help is shown below.

For details about a resource's or special command's arguments, specify the name of the resource followed by the argument followed by the `--help` option.

```
ilo workload create --help
```

REST API HTTP Response Code Per Command

At the end of its output, the `ilo` command displays the REST API HTTP response code from the command. For example, a successful operation shows the following:

```
...
200, OK
```

Authentication to PCE with API Key or Explicit Login

There are two ways to authenticate to your PCE:

- With an API key and key secret. This is the easiest way. Before you create the API key and secret, you need to log in to authenticate to the PCE. After creating and using the key, you do not have to specify your username and password again.
- With the explicit command to login, which always requires a username and password. This method also requires you to log out with a user ID displayed at login. The explicit login times out after ten minutes of inactivity, after which you must login again.

For both authentication mechanisms, on the command line, you always need to specify the FQDN and port of your PCE. The default port for the PCE is 8443. However, your system administrator can change this default. Check with your system administrator to verify the port you need.

Authenticate with an API Key

To authenticate to the PCE with an API key, you must first explicitly login to the PCE, create the API key, and then use the key to authenticate.

1. Authenticate via explicit login:

   ```
   ilo login --server yourPCEfqdn:itsPort
   ```

2. Create the API key:

   ```
   ilo api_key create --name someLabel
   ```

   someLabel is an identifier for the key.

3. Use the API key to authenticate:

   ```
   ilo use_api_key --server yourOwnPCEandPort --key_id yourOwnKeyId --key-
   secret yourOwnKeySecret
   ```

Create an API Key and Store it in a File

On Linux, for later ease of use, with the `api_key --create-env-output` option, you can store the API key, API secret, and the PCE server name and port as environment variables in a file that you source in future Linux sessions.

Linux Example. This example creates the API key and secret and stores them as environment variables in a file named `ilo_key_MY_SESSION_KEY`.

Example: Store API key and secret in file

```
# ilo api_key create --name MY_SESSION_KEY --create-env-output

# Created file ilo_key_MY_SESSION_KEY with the following contents:

export ILO_API_KEY_ID=14ea453b6f8b4d509
export ILO_API_KEY_SECRET=e1fa1262461ca2859fcf9d91a0546478d10a1bcc4c579d888a4e1cace71f9787
export ILO_SERVER=myPCE.BigCo.com:8443
export ILO_ORG_ID=1

# To export these variables:
# $ source ilo_key_MY_SESSION_KEY
```

Explicit Log into the PCE

Without an API key, you must explicitly log into the PCE. The command syntax is shown below:

For standalone on-premises PCE	The FQDN and port of the PCE: `ilo login --server` *`yourPCEfqdn`*`:`*`itsPort`* ⚠ **Do not specify a URL** For *`yourPCEfqdn`*`:`*`itsPort`*, do not specify a URL instead of the PCE's FQDN and port. If you do, an error message is displayed.
For the Illumio Central (SaaS)	`ilo login --server` *`URL_or_bare_PCEfqdn`*`:`*`itsPort`* `--login-server login.illum.io:443` See above for explanation of the argument to the `--server` option.

Notes about logging into PCE

- After login, the output of the command shows a user ID value. Make a note of this value because you need it when you log out.
- The session with the PCE remains in effect as long as you keep using the CLI tool. After ten minutes of inactivity, the session times out, and you must login again.

Example. In this example, the user ID is 6.

```
C:\Users\marie.curie> ilo login --server myPCE.BigCo.com:8443
Enter User Name: albert.einstein@BigCo.com
Enter Password: Welcome Albert!
User ID = 6
Last Login Time 2018-08-10T-09:58:07.000Z from someIPaddress
Access to Orgs:
Albert: (2)
Roles: [3]
Capabilities: {"basic"=>["read", "write"], "org_user_roles"=>["read", "write"]}
User Time Zone: America/Los_Angeles
Server Time: 2018-08-12T17:58:07.522Z
Product Version: 16.09.0-1635
Internal Version: 48.0.0-255d6983962db54dc7ca627534b9f24b94429bd5
Fri Aug 6 16:11:50 2018 -0800
Done
```

Log Out of the PCE

To end a session with the PCE, use the following command:

```
ilo user logout --id valueOfUserIdFromLogin
```

where:

- *valueOfUserIdFromLogin* is the user ID from your login. See "Log In to the PCE".

Example. In this example, the user ID is 6.

Example command-line logout with user ID 6

```
ilo user logout --id 6
```

Example CLI Tool Commands for Various Resources

This section illustrates the use of the CLI tool with various PCE resources.

Drilldown on the list argument

Many resources take the `list` option. This section details some of its uses.

Default List of All Fields

The default `list` command displays all fields associated with the resource:

```
ilo resource list
```

List Only Specific Fields

With the `--field` option, specify the fields to display.

```
ilo resource list --field CSV_list_of_fieldnames
```

For example, to display a list of labels with only the `href`, `key`, and `value` fields, use the `--field` option with those fields as comma-separated arguments.

Example list with selected fields

```
ilo label list --fields href,key,value
+---------------------+------+-----------------+
| Href                | Key  | Value           |
+---------------------+------+-----------------+
| /api/v1/2/labels/1  | role | Web             |
| /api/v1/2/labels/2  | role | Database        |
...
| /api/v1/2/labels/48 | loc  | Asia            |
+---------------------+------+-----------------+
```

Nested Resource Fields and Wildcards

Some resources have hierarchical, nested fields. For example, the `workload` resource includes the following hierarchy for the `agent` field:

```
agent/config/log_traffic
```

- A field named `agent`
    - that has a field named `config`
        - that has a field named `log_traffic`

To list nested fields, separate the hierarchy of the field names with a slash to the depth of the desired field.

To see all nested fields of one of a resource's fields, use the `*` wildcard.

Example. The following example displays all fields under the `agent/config` field.

Example of all nested fields with * wildcard

```
ilo workload list --field agent/config/*
+-------------+------------------+-------------+
| Log Traffic | Visibility Level | Mode        |
+-------------+------------------+-------------+
| false       | flow_summary     | illuminated |
| false       | flow_summary     | idle        |
+-------------+------------------+-------------+
```

You can combine individual field names, nested field names, and the `*` wildcard.

Example combination of individual fields, nested fields, and wildcard

```
ilo workload list --fields href,hostname,agent/config/*,agent/status/uid,agent/status/status
+------------------------------------------------------------+-----------------------------+-------------+-------
| Href                                                       | Hostname                    | Log Traffic |
Visibility Level | Mode        | Uid                                  | Status |
+------------------------------------------------------------+-----------------------------+-------------+-------
| /api/v1/1/workloads/527b8aca-97aa-43b9-82e1-29b17a947cdd | hrm-web.webscaleone.info    | false       |
flow_summary     | illuminated | 0ffd2290-e26a-4ec6-b241-9e2205c0b730 | active |
| /api/v1/1/workloads/4a8743a4-14ee-40d0-9ed2-990fe3f0ffb1 | hrm-db.webscaleone.info     | false       |
flow_summary     | illuminated | 145a3cc8-01a8-4a52-97b8-74264ad690e4 | active |
+------------------------------------------------------------+-----------------------------+-------------+----
...
```

Linux Save Specific Fields to File for Reuse

On Linux, for ease of reuse of specific fields, create a *display configuration file* in YAML format and set the environment variable `ILO_DISPLAY_CONFIG` to point to that file. Thereafter, you no longer need to specify specific fields on the `list` command line.

Example. Configure the `workloads list` command to display only the `href`, `hostname`, all `agent` configuration fields, and agent version:

Example command to save to list configuration file

```
ilo workload list --fields href,hostname,agent/config/*,agent/status/agent_version
```

Add the field names to a display configuration file in the following YAML format:

Example YAML layout of display configuration file

```
workload:
  fields:
    - href
    - hostname
  agent:
    config:
      fields:
        - '*'
    status:
      fields:
        - agent_version
```

Set the Linux environment variable `ILO_DISPLAY_CONFIG` to the path to the YAML file.

Example envar ILO_DISPLAY_CONFIG

```
$ export ILO_DISPLAY_CONFIG=~/ilo_display/display_config.yaml
```

List of All Workloads

To view all details for all workloads, use the following command:

```
ilo workload list
```

About the Workload UUID

To view an individual workload, you need the workload's identifier, called the UUID, or Universal Unique Identifier.

The UUID is shown in the list of all workloads described in "View List of All Workloads". The UUID is the last word of the value of the workload's `href` field, as shown in bold in the following example:

```
/api/v1/orgs/28/workloads/2ca0715a-b7e3-40e3-ade0-79f2c7adced0
```

View Individual Workload

To see the details about an individual workload, use the following command:

```
ilo workload read -workload-id UUID
```

where:

- *UUID* is the workload's UUID. See "About the Workload UUID".

The details of an individual workload are grouped under major headings:

- Workload -> Interfaces
- Workload -> Labels
- Workload -> Services
- Services -> Open Service Ports
- Agent -> Status

Example list of individual workload

```
ilo workload read --workload-id 2ca0715a-b7e3-40e3-ade0-79f2c7adced0
+--------------------------+-------------------------------------------------------------------------------------
| Attribute                | Value
+--------------------------+-------------------------------------------------------------------------------------
|  href                    |  /orgs/1/workloads/2ca0715a-b7e3-40e3-ade0-79f2c7adced0
|  deleted                 |  false
...
Workload -> Interfaces
+------+-----------------+------------+-------------------------+------------+------------+--------------------
| Name | Address         | Cidr Block | Default Gateway Address | Link State | Network Id | Network Detection
Mode
+------+-----------------+------------+-------------------------+------------+------------+--------------------
| eth0 | 10.0.0.16       | 8          | 10.0.0.1                | up         | 1          | single_private_brn
...
Workload -> Labels
+--------------------+
| Href               |
+--------------------+
| /orgs/1/labels/37 |
...
Workload -> Services
+-----------------+---------------------------+
| Attribute       | Value                     |
+-----------------+---------------------------+
|  uptime_seconds |  69016553                 |
...
Services -> Open Service Ports
+----------+---------+------+--------------+------+---------+------------------+
| Protocol | Address | Port | Process Name | User | Package | Win Service Name |
+----------+---------+------+--------------+------+---------+------------------+
| 17       | 0.0.0.0 | 123  | ntpd         | root |         |                  |
...
Workload -> Agent
+-----------+-------------------------------------------------------------------------------------+
| Attribute | Value                                                                               |
+-----------+-------------------------------------------------------------------------------------+
|  config   |  {"log_traffic"=>true, "visibility_level"=>"flow_summary", "mode"=>"enforced"} |
|  href     |  /orgs/1/agents/16                                                                  |
...
Agent -> Status
+---------------------------+---------------------------------------+
| Attribute                 | Value                                 |
+---------------------------+---------------------------------------+
|  uid                      |  db482b06-41c6-4297-a60c-396de13576ad |
|  last_heartbeat_on        |  2016-12-07T04:07:03.756Z             |
...
200, OK
```

List Draft or Active Version of Rulesets

A security policy item consists of ruleset, IP lists, label groups, services, and security settings. Before changes to these items take effect, the policy must be provisioned on the managed workload by setting its state to active with the CLI tool or provisioning it with the PCE web console.

To view a ruleset and provisioning state use the following command:

```
ilo rule_set list --pversion state
```

where *state* is one of the following values:

- `draft`: Any policy item that has not yet been provisioned.
- `active`: All policy items that have been provisioned and are enabled on workloads.

The provisioning state are listed in the **`Enabled`** column:

- `true`: the policy is provisioned.
- empty: the policy is a draft.

Example draft versions of rulesets

```
ilo rule_set list --pversion draft
+-------------------------------------------------+------------------------------+---------+-------------+-----
| Href                                            | Created By                   | Name    | Description |
Enabled |
+-------------------------------------------------+------------------------------+---------+-------------+-----
| /api/v1/orgs/28/sec_policy/draft/rule_sets/2387 | {"href"=>"/api/v1/users/74"} | foo1    |             | true
| /api/v1/orgs/28/sec_policy/draft/rule_sets/1909 | {"href"=>"/api/v1/users/0"}  | Default |             |
true    ...
200, OK
```

The state of the policy is stored in the `agent/status/status` field. See "Nested Resource Fields and Wildcards".

View Workload Rules

You can view a specific workload's rules with the following command:

```
ilo workload rule_view --workload-id UUID
```

where:

- *UUID* is the workload's UUID. See "About the Workload UUID".

In the example below, the workload's UUID is as follows.

```
2ca0715a-b7e3-40e3-ade0-79f2c7adced0
```

Example view workload rules

```
ilo workload rule_view --workload-id 2ca0715a-b7e3-40e3-ade0-79f2c7adced0
+------------+-------+
| Attribute  | Value |
+------------+-------+
|  providing |  []   |
+------------+-------+
Using
+---------------------+-------------------------------------------------------------------------------------------------
| Ports And Protocols | Rulesets
| Href                                        | Name         |
+---------------------+-------------------------------------------------------------------------------------------------
| [[-1, -1, nil]]     | [{"href"=>"/api/v1/orgs/28/sec_policy/8/rule_sets/1909", "name"=>"Default",
"secure_connect"=>false, "peers"=>[{"type"=>"ip_list", "href"=>"/api/v1/orgs/28/sec_policy/8/ip_lists/188",
"name"=>"Any (0.0.0.0/0)", "ip_ranges"=>[{"from_ip"=>"0.0.0.0/0"}]}]}] | /api/v1/orgs/28/sec_policy/8/services/
1153 | All Services |
+---------------------+-------------------------------------------------------------------------------------------------
200, OK
```

View Report of Workload Services or Processes

The following command lists all running services or processes on a workload:

```
ilo workload service_reports_latest  --workload-id UUID
```

where:

- *UUID* is the workload's UUID. See "About the Workload UUID".

In the example below, the workload's UUID is as follows.

```
2ca0715a-b7e3-40e3-ade0-79f2c7adced0
```

Example workload service report

```
ilo workload service_reports_latest --workload-id 2ca0715a-b7e3-40e3-ade0-79f2c7adced0
+-----------------+---------------------------+
| Attribute       | Value                     |
+-----------------+---------------------------+
|  uptime_seconds |  1491                     |
|  created_at     |  2015-10-20T15:13:00.681Z |
+-----------------+---------------------------+
Open Service Ports
+----------+---------+-------+--------------+-----------------+---------+------------------+
| Protocol | Address | Port  | Process Name | User            | Package | Win Service Name |
+----------+---------+-------+--------------+-----------------+---------+------------------+
| udp      | 0.0.0.0 | 5355  | svchost.exe  | NETWORK SERVICE |         | Dnscache         |
...
| tcp      | 0.0.0.0 | 135   | svchost.exe  | NETWORK SERVICE |         | RpcSs            |
+----------+---------+-------+--------------+-----------------+---------+------------------+
200, OK
```

Brief CLI Tool Tutorials

These tutorials walk you step-by-step to accomplish certain goals with `ilo`.

Tutorial – Upload Vulnerability Data

This example tutorial shows how to upload vulnerability data to the PCE. For background, see "Upload Vulnerability Data". The source of the vulnerability data in this example comes from Qualys®.

> ⚠ **Setup for vulnerability data**
> The tutorial assumes that you already have a license to upload vulnerability data, have activated the license, and have the vulnerability data to upload. See "Upload Vulnerability Data".

Goal

Upload authoritative vulnerability data for analysis in Illumination.

Steps

1. Do a non-authoritative upload of vulnerability data for examination:

   ```
   ilo upload_vulnerability_report --input-file C:\Users\albert-einstein0.xml --source-
   scanner qualys --format xml
   ```

2. Examine a single uploaded vulnerability record identified by its vulnerability identifier, `qualys-38173`. See "Vulnerability Identifier".

   ```
   ilo vulnerability read --xorg-id=1 --reference-id=qualys-38173
   ```

3. Do another non-authoritative upload of vulnerability data.

   ```
   ilo upload_vulnerability_report --input-file C:\Users\albert-einstein99.xml  --source-
   scanner qualys --format xml
   ```

4. Do an authoritative upload of vulnerability data, overwriting any previously uploaded records and adding any new vulnerability records.

   ```
   ilo upload_vulnerability_report --input-file C:\Users\albert.einstein_FINAL.xml --
   authoritative  --source-scanner qualys --format xml
   ```

Results

The authoritative vulnerability data has been uploaded and is ready for use in Illumination.

Tutorial – Import Traffic Flow Summary Data for Static Illumination™

Static Illumination provides "moment-in-time" visibility of inter-workload traffic. This visibility is useful to model policies, to look for specious traffic flows, and to ensure that metadata for labels is accurate.

Goal

Load workload and traffic data needed for analysis with static Illumination.

Setup

This tutorial relies on the following data to import.

- 1,000 workloads defined in the file `bulkworkloads-1000.csv`, which has the following columns:

Sample workload data to import

```
hostname,ips,os_type
10.14.59.8.netstat,10.14.59.8,linux
10.4.78.178.netstat,10.4.78.178,linux
10.37.134.179.netstat,10.37.134.179,linux
...
```

- 1,000,000 traffic flows defined in the CSV file `traffic.clean-1m.csv`, which has the following columns:

Sample traffic flow data to import

```
src_ip,dst_ip,dst_port,proto
10.40.113.86,10.14.59.8,10050,6
10.14.59.8,10.8.251.138,8080,6
10.40.113.124,10.14.59.8,22,6
...
```

Steps

The workflow is authenticate to the PCE and run two `ilo bulk_upload_csv` commands.

1. Authenticate to the PCE via API key or explicit login. See "Authentication to PCE with API Key or Explicit Login".

2. Load the workload data:
   `ilo workload bulk_upload_csv --file bulkworkloads-1000.csv`

3. Load the traffic flow data:
   `ilo traffic bulk_upload_csv --file traffic.clean-1m.csv`

Results

The data from the CSV files are uploaded.

Tutorial – Create Kerberos-authenticated Workloads

Notes on Kerberos tutorial

- This tutorial assumes that you already have your Kerberos implementation in place.
- As required by Kerberos, the Kerberos realm name is shown in all capital letters as `MYREALM`.
- VEN environment variables must be set *before* VEN installation. Environment variables for Linux are detailed in the *VEN Deployment Guide*.

Goals

- Create two workloads on Linux that are authenticated by Kerberos.
- Set the workloads' modes to `idle` and `illuminated` .
- Run the `kinit` command to get Kerberos tickets for the workloads.

Setup

The key data for using the `ilo` command to create these workloads are the name of the Kerberos realm and the Service Principle Name (SPN).

Steps

The workflow is authenticate, run two `workload create` commands that set the workloads' `mode`s, set the VEN environment variables, install the VEN, and run two Kerberos `kinit` commands to get Kerberos tickets for the workloads.

1. Authenticate to the PCE via API key or explicit login. See "Authentication to PCE with API Key or Explicit Login".
2. Create Kerberos-authenticated `myWorkload1` and set its `mode` to `idle`. The mode is a nested field, as described in "Resource Nested Fields and Wildcards":

   ```
   ilo workload create --hostname myPCE.BigCo.com --name myWorkload1 --service-principal-name host/myKerberosTicketGrantingServer@MYREALM --agent/config/mode idle
   ```

3. Create Kerberos-authenticated `myWorkload2` and set its `mode` to `illuminated`. The mode is a nested field, as described in "Resource Nested Fields and Wildcards":

   ```
    ilo workload create --hostname myPCE.BigCo.com --name myWorkload2 --service-principal-name host/ m yKerberosTicketGrantingServer @MYREALM --agent/config/mode illuminated
   ```

4. Before installation, set VEN environment variables:

   ```
   # Activate on installation
   VEN_INSTALL_ACTION=activate
   # FQDN and port PCE to pair with
   VEN_MANAGEMENT_SERVER=myPCE.BigCo.com:8443
   # Kerberos Service Principal Name
   ```

```
VEN_KERBEROS_MANAGEMENT_SERVER_SPN=host/myKerberosTicketGrantingServer
# Path to Kerberos shared object library
VEN_KERBEROS_LIBRARY_PATH=/usr/lib/libgssapi_krb5.so
```

5. Install the Linux VEN:

```
rpm -ivh illumio-ven*.rpm
```

6. Run `kinit` to get a Kerberos ticket for `myWorkload1`:

```
kinit -k -t /etc/krb5.keytab host/myWorkload1.BigCo.com@MYREALM
```

7. Run `kinit` to get a Kerberos ticket for `myWorkload2`:

```
kinit -k -t /etc/krb5.keytab host/ myWorkload2.BigCo .com @MYREALM
```

Results

The Kerberos-authenticated workloads are created, set in the desired modes, and given a Kerberos ticket.

Tutorial – Use --async option for asynchronous work with large datasets

The `--async` option is for working with large sets of data without having to wait for the results. The option works like "batch job".

The option can be used with any resource. The workflow is as follows:

- You issue the desired `ilo` command with the `--async` option, which displays a job ID.
- You take note of the job ID.
- Your session is freed up while the job runs.
- The job creates a data file, which you then view with `datafile --read --job-id` *`jobID`* .

Goal

Get a report of a large workload data set.

Steps

1. Issue the `--async` request for a workload list. Take note of job ID which is the final word of the href displayed on the `Location` line.

```
[kurt.goedel~]$ ilo workload list --async
```

```
 Using /home/kurt.goedel/.rvm/gems/ruby-2.2.1
Location: /orgs/1/jobs/fe8a1c2b-1674-4b83-8967-eb56c4ffa1e3
202, Accepted
```

2. Check to see if the job completed. Use the job ID from the `Location` output in previous command:

```
 [sigmund.freud~]$ ilo job read --job-id fe8a1c2b-1674-4b83-8967-eb56c4ffa1e
 Using /home/sigmund.freud/.rvm/gems/ruby-2.2.1
```

3. Download the resulting data file, specifying the job ID with `-uuid` *jobID* :

```
[bill.gates ~]$ ilo datafile read --uuid 1e1c1540-8a01-0136-ec14-02f4d6c1190c
Using /home/ bill.gates /.rvm/gems/ruby-2.2.1
+-----------------------------------------------+--------+-----+--
```

... Many lines not shown

```
+-----------------------------+----------------------+-----------------------------
+----------------------+
| Href                                                | Deleted | Name |
Description | Hostname
| Service Principal Name | Public Ip             | Distinguished Name | External Data
Set | External Data Reference
| Interfaces | Ignored Interface Names | Service Provider | Data Center
| Data Center Zone | Os Id | Os Detail | Online | Labels  | Services | Agent
 | Created At                  | Created By           | Updated At                  |
Updated By
+-----------------------------------------------------------+---------+------
+-------------+----------------
```

… More lines not shown

```
-----------------------------------------------------------+
| /orgs/1/workloads/50ce441e-75ac-4be8-9201-96169545019c | false   |      |
             | 10.14.59.8.netstat
   ...
```

... Many lines not shown

```
  ...
```

Upload Vulnerability Data

The focus of this section is to show how to use the `ilo` commands to upload vulnerability data to the PCE with for analysis in Illumination® .

After data are uploaded, much of your work with vulnerability maps in Illumination is detailed in the *PCE Web Console User Guide* .

Adding the License for Vulnerability Data Upload

An Illumio ASP Vulnerability Maps license is required to upload vulnerability data into the Illumio PCE. For information about obtaining the license, contact Illumio Customer Support.

You will be provided with a license file named `license.json`. After you have obtained your license key, store it in a secure location.

Authenticate to PCE before adding license

- Before adding the license, you must first authenticate to the PCE. See "Authentication to PCE with API Key or Explicit Login".
- To add the license you must be the organization Owner or be a user that has Owner privileges.

Use the following command to inform the PCE of your valid license:

```
 ilo license create --license-file "path_to_license_file/license.json"   --feature "
feature_name"   [debug [v | verbose] trace]
```

where:

What	Required?	Description
"*path_to_license_file*/license.json"	Required	The quoted path to the `license.json` file from Illumio. Example: `"~/secretDir/license.json"`
"*feature_name*"	Required	The quoted string `"vulnerability_maps"`, which specifies the feature name the license enables.
`debug`	Optional	Enable debugging.
v | verbose	Optional	For verbose logging.
`trace`	Optional	Enable API trace.

Process and Details for Vulnerability Data Upload

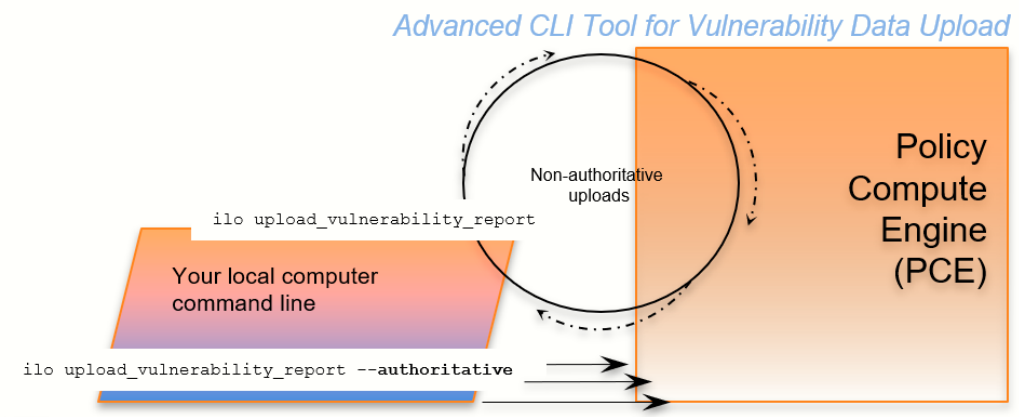
On upload, the CLI tool associates a workload's IP addresses with corresponding vulnerabilities identified for that workload.

Kinds of Vulnerability Data Uploads

There are two kinds of upload: non-authoritative and authoritative.

- **Non-authoritative.** This is the default. A non-authoritative upload:
  - Appends incoming data to any previously loaded records
  - Accumulates records for the same workloads without regard to duplicates.

  You can repeat the non-authoritative upload as many times as you like until you are satisfied with the results.



- **Authoritative.** You indicate authoritative data with the `-authoritative` option. An authoritative upload:
  - Overwrites any previously uploaded records for workloads matched to the incoming records.
  - Eliminates duplicate records.
  - Adds new records not previously written by other uploads.

  You can repeat the authoritative upload as many times as you like until you are satisfied with the results.

After either kind of upload, you can examine the uploaded data with the CLI tool or the PCE web console. See the *PCE Web Console User Guide*.

Supported vulnerability data sources

The CLI Tool works with vulnerability data from the following sources.

- Nessus Professional™.
- Qualys®.
- Rapid7©.

Rapid7 data exported in Qualys format
Before uploading to the PCE, Rapid7 vulnerability data must have been exported in Qualys format from Rapid7 with Qualys XML Export.

- Tenable Security Center

Common Vulnerabilities and Exposures (CVE)

Vulnerabilities are defined by Common Vulnerabilities and Exposures (CVE), with identifiers and descriptive names from the U.S. Department of Homeland Security National Cybersecurity Center.

Vulnerability Scores

Illumio computes a vulnerability score, which is a measure of the vulnerability of your entire organization. The score is displayed by the `ilo vulnerability list` command for all vulnerabilities or individual vulnerability via the vulnerability identifier.

Vulnerability Identifier

A uploaded vulnerability has an identifier as shown in the example below. The vulnerability identifier is tied to a specific CVE. You use this identifier with `--reference-id` option to examine specific uploaded vulnerabilities. See "Example – List a Single Uploaded Vulnerability".

The following are examples of vulnerability identifiers.

- Nessus Professional: `nessus-65432`
- Qualys: `qualys-23456`
- Rapid7: `qualys-98765`. Because Rapid7 data is first exported from Rapid7 in Qualys format, it is given a Qualys identifier when uploaded to the PCE.

Vulnerabilities for Unmanaged Workloads

You can upload vulnerabilities for unmanaged workloads. However, unmanaged workloads do not have any vulnerability score or associated CVE. If the unmanaged workload is later changed to managed, this information becomes available.

Prerequisites for Vulnerability Data Upload

Before uploading vulnerability data, ensure that you are ready with the following requirements.

- An Illumio Vulnerability Maps license is required to upload vulnerability data to the PCE. See "Adding the License for Vulnerability Data Upload".
- XML-formatted vulnerability data files from one of the supported sources.
- Authenticated CLI-tool access to the target PCE. See "Authentication to PCE with API Key or Explicit Login".
- Authenticated access and necessary permissions in the PCE web console for working with vulnerability maps. These details are described in the *PCE Web Console User Guide*.

Formal Syntax of CLI Tool Upload of Vulnerability Data

The key argument and option for uploading vulnerability data are as follows. For readability, this syntax is broken across several lines.

```
 ilo upload_vulnerability_report
--input-file path_to_datafile.xml
--source-scanner [nessus-pro|qualys|tenable-sc]
--format xml
[--authoritative]
[ --api-user ApiServerUserName --api-server SourceApiServer:port ]
```

where:

What	Required?	Description
`--input-file` *`path_to_datafile`*`.xml`	Required	Location of the data file to upload. The path to the datafile can be either an absolute path or a relative path.

`--source-scanner [nessus-pro|qualys|tenable-sc]`	Required	Indicates the source of the scan. One of the following is required:
		
		• `nessus-pro`: Nessus Professional
		• `qualys`:
		  ◦ Qualys
		  ◦ Rapid7 data exported from Rapid7 in Qualys XML format.
		• `tenable-sc`: For vulnerability data from Tenable Security Center

<table>
<tr><th>What</th><th>Required?</th><th>Description</th></tr>
<tr><td><code>--format formatType</code></td><td>Required</td><td>Format of file to upload:<table><tr><th>Format</th><th>Valid --source-scanner Argument</th></tr><tr><td><code>xml</code></td><td><code>--source-scanner nessus-pro</code>
<code>--source-scanner qualys</code></td></tr><tr><td><code>csv</code></td><td><code>--source-scanner tenable-sc</code></td></tr><tr><td><code>api</code></td><td><code>--source-scanner tenable-sc.</code> See also <code>--api-server</code> and <code>--api-user</code></td></tr></table></td></tr>
<tr><td><code>--authoritative</code></td><td>Optional</td><td>For uploading authoritative vulnerability data. The default command is without the <code>--authoritative</code> option. See "Kinds of Vulnerability Data Uploads".</td></tr>
<tr><td><code>--api-server SourceApiServer:port</code></td><td>Required for Tenable with <code>--format api</code></td><td>SourceApiServer:port are the protocol, FQDN, and port of the source server.</td></tr>
<tr><td><code>--api-user ApiServerUserName</code></td><td>Required for source API server authentication</td><td>The user name for authenticating to the SourceApiServer.

You are always prompted to enter your password.</td></tr>
</table>

Examples of Working with Vulnerability Data

Example – Upload Non-Authoritative Vulnerability Data

In this example, the `--source-scanner nessus-pro` option indicates that the data comes from Nessus Professional. Note the absolute path to the data file on Windows. This Windows example is broken across several lines with the PowerShell line continuation character (`).

```
C:\Users\donald.knuth> ilo upload_vulnerability_report `
--input-file C:\Users\donald.knuth\Desktop\vuln_reports\nessus3.xml `
--source-scanner nessus-pro --format xml

Elapsed Time [0.05 (total : 0.05)] - Data parsing is done.
Elapsed Time [1.08 (total : 1.13)] - Got workloads. Workload count: 5.
Elapsed Time [0.0 (total : 1.13)] - Built workload interface mapping. Total
interfaces : 11.
 Elapsed Time [4.57 (total : 5.7)] - Imported Vulnerabilities..
 Elapsed Time [0.0 (total : 5.7)] - Detected Vulnerabilities are associated with
vulnerability and workload data..
Elapsed Time [0.83 (total : 6.53)] - Report Imported.

Summary:
Processed the report with the following details :
Report meta data =>
Name          : Generic
Report Type   : nessus
Authoritative : false
Scanned IPs   : ["10.1.0.74", "10.1.0.223", "10.1.0.232", "10.1.0.221", "10.1.0.11",
"10.1.0.82", "10.1.0.43", "10.1.0.91", "10.1.0.8", "10.1.1.250"]

Stats :
   Number of vulnerabilities           => 19
   Number of detected vulnerabilities  => 31

Done.
```

Example – Upload of Rapid7 Vulnerability Data

The syntax for uploading vulnerability data from Rapid7 is identical to the syntax for uploading vulnerability data from Qualys. Note the `--format` **`qualys`** option and the absolute path to the data file on Windows. This Windows example is broken across several lines with the PowerShell line continuation character (`).

> **Rapid7 data exported in Qualys format**
> Before uploading to the PCE, Rapid7 vulnerability data must have been exported in Qualys format from Rapid7 with Qualys XML Export.

```
C:\Users\edward.teller> ilo upload_vulnerability_report `
--input-file C:\Users\edward.teller\Desktop\vuln_reports\rapid7.xml `
```

```
--source-scanner qualys --format xml
...
Done.
```

Example – Upload Authoritative Vulnerability Data

In this example, note the prompt to be sure this is an authoritative upload.

> **All capital letters YES**
> If you want to proceed, you must enter the word **YES** in all capital letters.

```
C:\Users\jrobert.oppenheimer> ilo upload_vulnerability_report --input-file
dataDir/authoritativedata.xml --authoritative --source-scanner qualys --
format xml

Using /home/centos/.rvm/gems/ruby-2.4.1
Authoritative scan overwites the previous entries for all the ips within this scan.
There is no ROLLBACK
Are you sure this is an authoritative scan? (YES | NO)
YES
Elapsed Time [11.86 (total : 11.86] - Data parsing is done.
Elapsed Time [0.27 (total : 12.13] - Got workloads. Workload count: 3.
Elapsed Time [0.0 (total : 12.13] - Built workload interface mapping. Total
interfaces : 6.
Elapsed Time [3.02 (total : 15.15] - Imported Vulnerabilities..
Elapsed Time [0.0 (total : 15.15] - Detected Vulnerabilities are associated with
vulnerability and workload data..
Elapsed Time [0.84 (total : 16.0] - Report Imported.
Summary:
Processed the report with the following stats -
    Number of vulnerabilities           => 14
    Number of detected vulnerabilities  => 48
Done.
```

Example – List Single Uploaded Vulnerability

This example uses a single Qualys vulnerability identifier to show the associated vulnerability. The value passed to the `--reference-id` option is shown as `qualys-38173`. See "Vulnerability Identifier".

```
$ ilo vulnerability read --xorg-id=1 --reference-id=qualys-38173
 ...
```

```
| Attribute | Value |
+-------------+-------------------------------------------------------------------+
| href | /orgs/1/vulnerabilities/qualys-38173 |
| name | SSL Certificate - Signature Verification Failed Vulnerability
| score | 39 |
| cve_ids | [] |
| created_at | 2018-11-05T18:16:56.846Z |
...
```

Example – List All Uploaded Vulnerabilities

This example highlights the vulnerability identifier, the CVE identifiers, and the description of the CVE. See "Common Vulnerabilities and Exposures (CVE)" and "Vulnerability Identifier". The layout of the output is the same for all supported vulnerability data sources.

Nessus Professional

```
C:\Users\werner.heisenberg> ilo vulnerability list --xorg-id=1
...
| Href | Name | Score | Description | Cve Ids | Created At | Updated At | Created By |
Updated By |

---------------------+-------------------------+----------------------
+-----------------------+
| /orgs/1/vulnerabilities/nessus-18405 | Microsoft Windows Remote Desktop Protocol
Server Man-in-the-Middle Weakness | 51    |             |
["CVE-2005-1794"]                  | 2018-11-07T03:15:39.410Z |
2018-11-07T03:15:39.410Z | {"href"=>"/users/1"} | {"href"=>"/users/1"} |
...
```

Qualys

```
C:\Users\isaac.newton> ilo vulnerability list --xorg-id=1
...
| Href | Name | Score | Description | Cve Ids | Created At | Updated At | Created By
| Updated By |
---------------------+-------------------------+----------------------
+-----------------------+
| /orgs/1/vulnerabilities/qualys-38657 | Birthday attacks against TLS ciphers with
64bit block size vulnerability (Sweet32) | 69 | | ["CVE-2016-2183"] |
2018-07-27T18:16:57.166Z | 2018-08-08T22:30:32.421Z | {"href"=>"/users/1"} |
{"href"=>"/users/16"} |
...
```

Rapid7

Because Rapid7 vulnerability data must be in Qualys format before upload, the output is the same as for Qualys data, including the vulnerability identifier (qualys-38657 in the example above) and CVE. See "Common Vulnerabilities and Exposures (CVE)" and "Vulnerability Identifier".

Example – View Vulnerability Report

The **Report Type** column identifies the source of the scan; in this example, Qualys.

```
C:\Users\gracemurry.hopper> ilo vulnerability_report list --xorg-id=1
...
| Href | Report Type | Name | Created At | Updated At | Num Vulnerabilities |
Created By | Updated By |
+-----------------------------------------------------+-------------
+----------------------+--------------------------+----------------------
| /orgs/1/vulnerability_reports/scan_1502310096_09344 | qualys  |
NewAuthoritativeScan | 2018-08-08T22:30:34.877Z | 2018-08-08T22:30:34.877Z | 62 |
{"href"=>"/users/16"} | {"href"=>"/users/16"} |
...
```

Working with Vulnerability Maps in Illumination

This information is detailed in the *PCE Web Console User Guide* .

Importing/Exporting Security Policy

Using the CLI tool, you can export and import security policy to and from the PCE. Importing/exporting security policy is particularly useful for moving policy from one PCE to another so you can avoid recreating policy from scratch on the target PCE. For example:

- You can test policy on a staging PCE and then move it to your production PCE.
- You can move policy from a proof-of-concept PCE deployment to your production PCE.

Exportable/Importable Policy Objects

You can use the CLI tool to export or import the following objects in the PCE:

- Labels: `labels`.

- Label Groups: `label_groups`.
- Pairing Profiles: `pairing_profiles`.
- IP Lists: `ip_lists`.
- Services: `services`.
- Rulesets and Rules: `rule_sets`.

About Exporting Rules for Workloads, Virtual Servers, and Virtual Services

For flexibility, Illumio recommends that you base your security policy rules on labels. Do not tie the rules to specific individual workloads, virtual services, or virtual servers.

Virtual servers and virtual services are not exported.

The CLI tool policy export does not include such references. If you have rules that are tied to individual workloads, virtual services, or virtual servers, a warning is displayed on export. Attempts to import such rules fail and display the reason for the failure.

Example failed attempt to export rules tied to individual workload

```
WARNING: rule /orgs/1/sec_policy/active/rule_sets/3/sec_rules/39 contains non-transferrable providers:
workload /orgs/1/workloads/a51ae67d-472a-44c3-984e-d518a8e95aee
Unable to proceed, please verify input
```

Workflow for Security Policy Export/Import

- Authenticate to the source PCE. See "Authentication to PCE with API Key or Explicit Login".
- Export the policy to a file. Syntax summary:
  `ilo sec_policy export --file` *`someExportFilename`*

- Authenticate to the target PCE. See "Authentication to PCE with API Key or Explicit Login".

- Import the saved policy. Syntax summary:
  `ilo sec_policy import --file` *`someImportFilename`*

Output Options, Format, and Contents

All exported policy is written to standard output. To write to a file, use the `--file` option.

Exported policy is in JSON format.

By default, all supported policy objects are exported. You can export a subset of policy by specifying one or more resource types with the '`-resource`'
option (`labels`, `label_groups`, `pairing_profiles`, `ip_lists`, `services`, or `rule_sets`).

When a subset of policy items is exported (such as only labels), all referenced resources are also exported.

See also "Exporting Rules for Workloads, Virtual Servers, and Bound Services".

Exported Rulesets

With the -- `rule_set` option, you can export multiple rulesets.

By default, only the most recently provisioned, active policy is exported. To export the current draft policy or a previous policy, use the `--pversion` *`state`* option. See "List Draft or Active Version of Rulesets".

For a single ruleset, make sure the `--pversion` *`state`* you specify matches the provisioned state of the ruleset. In the following example, the state is `draft`:

```
ilo sec_policy export --pversion draft --rule_set /orgs/1/sec_policy/draft/
rule_sets/1
```

Effects of Policy Import

All imported policy is read from standard input, unless you import from a file with the `--file` option.

You can import policy file multiple times. Each import affects only a single copy of a resource.

All imported policy is set to the `draft` provisioned state. After the import, you must explicitly provision the `active` state.

Non-transferrable policy rules (that is, rules tied to specific workloads, virtual servers,and bound services), the import aborts with a warning. See "Exporting Rules for Workloads, Virtual Servers, and Virtual Services".

Policy items already on the target PCE are updated by imported resources whose names match the already existing resources' names. Services do not have to have the same names. Services match if they have the same set of ports and protocols.

Resources are not deleted by an import. For example, if you export policy from PCE-1 to PCE-2, delete a resource "R" from PCE 1, and then export and import again, resource "R" is still present on PCE 2. You must explicitly delete resource "R" from PCE2.

General Error Message and Error Logging

For many syntactical or other types of errors, the CLI tool displays and general message encouraging you to verify your syntax with the CLI tool help:

```
The ilo command has encountered an error. Check your syntax with either of the
following commands:
```

- `ilo`
- `ilo <command> --help`

In addition, in some circumstances, the CLI tool writes a detailed log of errors:

```
For detailed error messages, see the file: location-of-local-temp-directory/illumio-
cli-error.log
```

where `location-of-local-temp-directory` is as follows:

- Linux: /tmp
- Windows: C:\Windows\Temp

Revision History

Illumio Adaptive Security Platform Advanced PCE Command-Line Tool Interface Guide

Document ID: 0008_20181026

Date	Description

2018-12-03	Updated for Command-line Tool (CLI) version 1.1:
	• All CLI installation packages are 64-bit.
	• Support for Windows 10.
	• Support for Nessus Professional vulnerability data.
	• Support for Rapid7 vulnerability data exported from Rapid7 in Qualys XML format.
	• Support for Tenable Security Center vulnerability data
	• Syntax change for uploading vulnerability data:
	  ◦ Formerly: `ilo upload_qualys_report`
	  ◦ Now: `ilo upload_vulnerability_report --source-scanner [nessus-pro|qualys|tenable-sc]`

2018-11-08	Corrected prefix of names of Linux environment variables from erroneous `ILLUMIO_*` to correct `ILO_*`.

Date	Description
2018-10-05	Updated for CLI tool 1.0.0 general availability (GA): • Support for Microsoft Windows.
2018-09-27	• CLI tool 1.0.0 (beta) first publication to all customers. • Start of revision history.