



Illumio Core PCE CLI Tool Guide

1.4.2

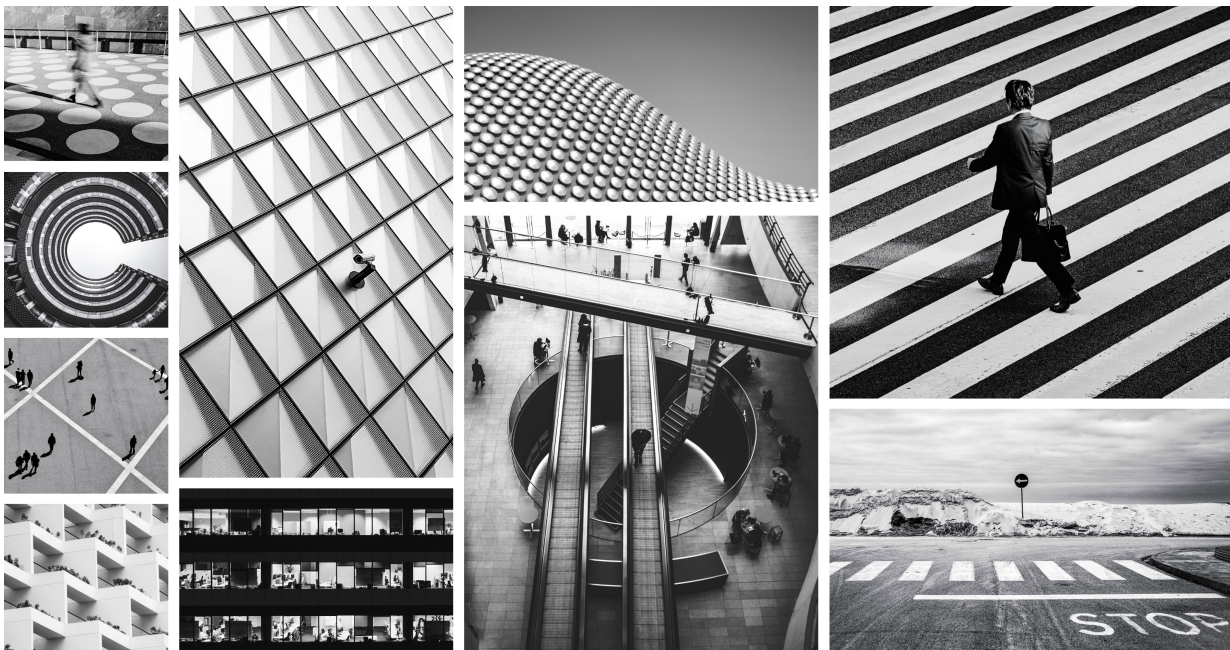


Table of Contents

Legal Notice	4
Overview of the CLI Tool	5
About This Guide	5
CLI Tool Versioning	5
How to Use This Guide	6
Before Reading This Guide	6
Notational Conventions in This Guide	6
CLI Tool and PCE Resource Management	6
The <code>ilo</code> Command	7
Formal Syntax	7
CLI Tool Help	8
HTTP Response Codes and Error Messages	8
REST API HTTP Response Codes	8
Error Messages	8
Environment Variables	9
Installation and Authentication	11
Installation Prerequisites	11
Prerequisite Checklist	11
TLS/SSL Certificate for Access to the PCE	12
Set the Illumio Core REST API Version	12
Install, Upgrade, and Uninstall the CLI Tool	12
Download the Installation Package	12
Install Linux CLI Tool	13
Upgrade Linux CLI Tool	13
Uninstall Linux CLI Tool	13
Install Windows CLI Tool	14
Upgrade Windows CLI Tool	14
Uninstall Windows CLI Tool	14
Authenticate with the PCE	14
Authenticate with an API Key	15
Create an API Key	15
Log Into the PCE	15
Log Out of the PCE	16
CLI Tool Commands for Resources	18
View Workload Rules	18
View Report of Workload Services or Processes	19
View Host and System Inventory	19
Use the <code>list</code> Option for Resources	20
Default List of All Fields	20
List Only Specific Fields	20
Nested Resource Fields and Wildcards	20
Linux: Save Fields for Reuse	21
List of All Workloads	22
About the Workload UUID	22
View Individual Workload	23
List Draft or Active Version of Rulesets	24
Import and Export Security Policy	25
Export and Import Policy Objects	25
About Exporting Rules	25
Workflow for Security Policy Export/Import	26
Output Options, Format, and Contents	26
Effects of Policy Import	27
Upload Vulnerability Data	27

Add the License for Vulnerability Data Upload	27
Vulnerability Data Upload Process	28
Vulnerability Data Upload CLI Tool Syntax	31
Vulnerability Data Examples	33
CLI Tool Tutorials	37
How to Import Traffic Flow Summaries	37
Goal	37
Setup	37
Steps	37
Results	38
How to Create Kerberos-Authenticated Workloads	38
Goals	38
Setup	38
Steps	38
Results	39
How to Work with Large Datasets	39
Goal	39
Steps	39
How to Upload Vulnerability Data	40
Goal	40
Steps	41
Results	41

Legal Notice

Copyright © 2024 Illumio 920 De Guigne Drive, Sunnyvale, CA 94085. All rights reserved.

The content in this documentation is provided for informational purposes only and is provided "as is," without warranty of any kind, expressed or implied of Illumio. The content in this documentation is subject to change without notice.

Resources

- [Legal information](#)
- [Trademarks statements](#)
- [Patent statements](#)
- [License statements](#)

Contact Information

- [Contact Illumio](#)
- [Contact Illumio Legal](#)
- [Contact Illumio Documentation](#)

Overview of the CLI Tool

This topic provides an overview of the CLI Tool, describes the general syntax of the CLI Tool command, and lists the environment variables you can use to customize the CLI Tool.



IMPORTANT

See the *Illumio Core CLI Tool 1.4.0 Release Notes* and *Illumio Core CLI Tool 1.4.1 Release Notes* and *Illumio CORE CLI Tool 1.4.2 Release Notes* in your respective Illumio Core Technical Documentation portal for the updates to the CLI Tool for these releases.

About This Guide

The following sections provide useful information to help you get the most out of this guide.

CLI Tool Versioning

Illumio Core CLI Tool version 1.4.2 is compatible with Illumio Core PCE versions:

PCE 19.3.6-H2 (LTS)

PCE 21.2.4 (LTS)

PCE 21.5.20 (LTS)

PCE 22.1.1 (Standard)

PCE 22.2.0 (Standard)

The CLI Tool version numbering is independent from the release and version numbering of Illumio Core 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.



IMPORTANT

See the *Illumio Core CLI Tool 1.4.0 Release Notes*, *Illumio Core CLI Tool 1.4.1 Release Notes* and *Illumio Core CLI Tool 1.4.2 Release Notes* in your respective Illumio Core Technical Documentation portal for the updates to the CLI Tool for these releases.

How to Use This Guide

This guide includes several major sections:

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

Before Reading This Guide

Before performing the procedures in this guide, be familiar with the following information:

- The CLI Tool interacts with the PCE; therefore, be familiar with PCE concepts such as core and data nodes, workloads, and traffic. See the PCE Administration Guide.
- The CLI Tool is often used to upload vulnerability data; therefore, understand how vulnerability data is used in the PCE web console. See the "Vulnerability Maps" topic in Visualization Guide.
- The CLI Tool can be used with workload data; therefore, you must understand what workloads are. See the "VEN Architecture and Components" topic in the VEN Administration Guide.
- The CLI Tool can be used with security policy rules, rulesets, labels, and similar resources; therefore, be familiar with these concepts. See "The Illumio Policy Model" in the Security Policy Guide.

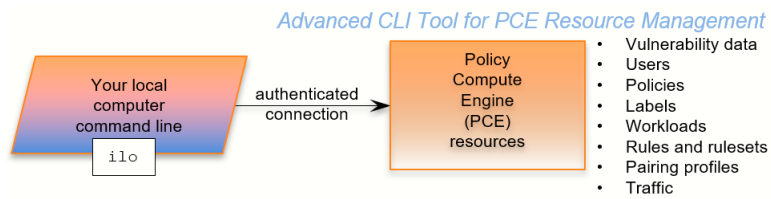
Notational Conventions in This Guide

- 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. Example:

```
...  
some command or command output  
...
```

CLI Tool and PCE Resource Management

With the Illumio 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 capabilities:

- 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.



CAUTION

The CLI Tool is a powerful way to work with your PCE resources. Exercise caution to make sure that 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 Core REST API. No knowledge of the REST API is required.

The `ilo` Command

This section describes the general syntax of the CLI Tool command, `ilo`, and tells how to use command-line help to get more specific syntax information.

Formal Syntax

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

```
ilo resource_or_specialCommand argument options
```

Where:

- `resource_or_specialCommand` represents either a resource managed by the PCE or a command that is not related to a particular resource.
A resource is an object that the PCE manages, such as a workload, label, or pairing profile.
Example resource command on Linux (create a workload):

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

A special command is a command that is not related to a specific resource. Special commands include `user`, `login`, `use_api_key`, and `node_available`.

Example special command on Windows (log out of PCE):

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

- The `argument` represents an operation on the resource or special command.
- The `options` are allowed options for the `resource_or_specialCommand`. The specific option depends on the type of resource or special command.

CLI Tool Help

To get a complete list of all the available CLI Tool commands, use the `ilo` command without options. This command displays the high-level syntax of special commands, resources, and their allowable options.

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. For example:

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

HTTP Response Codes and Error Messages

This section describes the response codes and error messages that can be returned when you use CLI Tool commands.

REST API HTTP Response Codes

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 output:

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

Error Messages

For many syntactical or other types of errors, the CLI Tool displays a 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`

Environment Variables

Illumio provides Linux environment variables to allow users to customize operation of the CLI tool.

Environment Variable	Purpose
<code>ILO_API_KEY_ID</code>	API key for non-password based authentication and cookie less session with PCE. See "Authenticate with an API Key".
<code>ILO_API_KEY_SECRET</code>	API key secret for non-password based authentication and cookie less session with PCE. See "Authenticate with an API Key".
<code>ILO_API_VERSION</code>	API version to be used to execute CLI commands. Set this if you want to override the default API version. See "Set the Illumio ASP REST API Version." Default: v2. Example: <code>\$ export ILO_API_VERSION=v1</code>
<code>ILO_CA_DIR</code>	Directory that contains certificates. See "TLS/SSL Certificate for Access to the PCE".
<code>ILO_CA_FILE</code>	Absolute path to certificate file. See "TLS/SSL Certificate for Access to the PCE".
<code>ILO_DISPLAY_CONFIG</code>	Absolute path to the display configuration file that is to be used with the list command. See "Linux Save Specific Fields to File For Reuse".
<code>ILO_INSECURE_PASSWORD</code>	Provide a password for login. If this variable is set, the login password prompt does not appear, and this password is used instead. Do not use in a production system when authentication security is desired. Example: <code>\$ export ILO_INSECURE_PASSWORD=myInsecurePassword</code>
<code>ILO_KERBEROS_SPN</code>	Kerberos service principal name (SPN). Specify this variable when using Kerberos authentication.
<code>ILO_LOGIN_SERVER</code>	PCE login server FQDN. Use this variable when the login server FQDN is not the same as the PCE FQDN. See "Explicit Log into the PCE".
<code>ILO_ORG_ID</code>	Organization identifier for certificate-authenticated session with PCE. Value is always 1. Does not need to be explicitly set The environment variable is set by the system and should not be explicitly set. See "Authentication to PCE with API Key or Explicit Login".
<code>ILO_PCE_VERSION</code>	PCE version for the CLI to use. Default: 19.1.0 Example: <code>\$ export ILO_PCE_VERSION=18.2.5</code>
<code>ILO_PREVIEW</code>	Enable any preview features that are included in this release. To disable preview features, remove this variable from the environment.

Environment Variable	Purpose
ILO_SERVER	FQDN of PCE for login and authentication with PCE. See "Authentication to PCE with API Key or Explicit Login".
TSC_ACCESS_KEY	These two ENV variables have been added in the release 1.4.2 to set up the Tenable SC API keys, which are used for authentication.
TSC_SECRET_KEY	
TSC_HOST	Variable that specifies the target host for Tenable
QAP_HOST	Variable that specifies the target host for Qualys

Installation and Authentication

This section describes how to install the CLI Tool. It also describes how to set up authentication, how to upgrade the tool, and how to uninstall it.

Installation Prerequisites

This section details prerequisites and the installation of the CLI Tool. Be sure you meet the prerequisites in the checklist.

Prerequisite Checklist

- License for vulnerability data upload
- Vulnerability data for upload
- Functional PCE
- Supported operating systems
- TLS/SSL certificate for authenticating to the PCE
- API version set in configuration
- The CLI Tool installation program

License for Vulnerability Data

The Illumio Core Vulnerability Maps license is required to import vulnerability data into the Illumio PCE. For information about obtaining a license, contact Illumio Customer Support. For information on activating the license, see [Add the License for Vulnerability Data Upload \[27\]](#).

Upload Vulnerability Data

When 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 \[30\]](#) for information.

Install Functional PCE

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

Supported Computer Operating Systems

The CLI Tool is supported on the following operating systems.

Linux

- Ubuntu 18.04
- Ubuntu 20.04
- Centos/RHEL 7.9
- Centos/RHEL 8.4

Microsoft Windows

**NOTE**

The CLI Tool is not supported on Windows 32-bit CPU architecture. Ensure that you run it on Windows 64-bit CPU architecture.

- Windows 2012 64 bit
- Windows 2016 64 bit
- Windows 10 64 bit

**NOTE**

CLI 1.4.2 is no longer supported on Windows 2008 R2 (EOL). The CLI Tool should work and can be used at your own risk.

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 provided in the PCE Installation and Upgrade Guide.

Alternative Trusted Certificate Store

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

Example: Set envvar for alternative trusted certificate store z

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

Set the Illumio Core REST API Version

The CLI Tool uses v2 of the Illumio Core REST API by default.

Install, Upgrade, and Uninstall the CLI Tool

This section explains how to install, upgrade, or uninstall the CLI Tool on Linux or Windows.

Download the Installation Package

Download the CLI Tool installation package from the [Tools Catalog](#) page (login required) to a convenient location on your local computer.

Install Linux CLI Tool

The CLI Tool 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.

RedHat/CentOS:

```
$ sudo rpm -ivh /path_to/nameOfCliRpmFile.rpm
```

Debian/Ubuntu:

```
$ sudo dpkg -i / path_to / nameOfCliDebFile .deb
```

Upgrade Linux CLI Tool

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

RedHat/CentOS:

```
$ sudo rpm -Uvh /path_to/nameOfCliRpmFile.rpm
```

Debian/Ubuntu:

```
$ sudo dpkg -i / path_to / nameOfCliDebFile .deb
```

The same option, `-i`, is used for installation or upgrade.

Uninstall Linux CLI Tool

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

RedHat/CentOS:

```
$ sudo rpm -e nameOfCliRpmFile
```

Debian/Ubuntu:

```
$ sudo dpkg -r nameOfCliDebFile
```

Install Windows CLI Tool

The CLI Tool 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 "Installation Successfully Completed" message and the help text for the CLI Tool is displayed.

Upgrade Windows CLI Tool

The CLI Tool cannot be directly upgraded from an existing CLI Tool installation.

If you have already installed a previous version of the CLI Tool, manually uninstall it with the Windows Control Panel's Add/Remove Programs.

After uninstalling the previous version of the CLI Tool, install the new version of the CLI Tool as described in [Install Windows CLI Tool \[14\]](#).

Uninstall Windows CLI Tool

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

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

Authenticate with the PCE

When using the CLI Tool, you can authenticate to your PCE in the following ways:

- **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 log in:**

This 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 log in 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 log into 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 --org-id --key-secret yourOwnKeySecret
```

Create an API Key

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`.

```
# 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=elfa1262461ca2859fcf9d91a0546478d10a1bcc4c579d888a4e1cace71f9787
export ILO_SERVER=myPCE.BigCo.com:8443
export ILO_ORG_ID=1

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

Log Into the PCE

Without an API key, you must explicitly log into the PCE.

For on-premises PCE deployments, the login syntax is the FQDN and port of the PCE:

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

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 Secure Cloud customers, the login syntax is:

```
ilo login --server URL_or_bare_PCEfqdn:itsPort --login-server
login.illum.io:443
```

See the explanation above about the argument to the `--server` option.

- After login, the output of the command shows a user ID value. Make a note of this value. 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 10 minutes of inactivity, the session times out, and you must log in 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 Into the PCE \[15\]](#) for information.

Example

In this example, the user ID is 6.

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

CLI Tool Commands for Resources

This section describes how to use the CLI Tool with various PCE resources.

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 \[22\]](#) for information.

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
+-----+
| Name | Href |
+-----+
+-----+
| [[-1, -1, nil]] | [{"href"=>"/api/v2/orgs/28/sec_policy/8/rule_sets/1909", "name"=>"Default", "secure_connect"=>false, "peers"=>[{"type"=>"ip_list", "href"=>"/api/v2/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/v2/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 \[22\]](#).

In the example, 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

View Host and System Inventory

You can use the following commands to get a quick source of information for troubleshooting or when working with Illumio Customer Support. Using these commands is a quicker and less detailed alternative to running a PCE support report.

To show host inventory for the "local" node:

```
$ illumio-pce-env show host-inventory
```

To show system inventory for the PCE:

```
$ illumio-pce-env show system-inventory
```

To show host inventory for all PCE nodes and also the PCE system inventory:

```
$ illumio-pce-env show inventory
```

Use the `list` Option for Resources

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/v2/2/labels/1 | role | Web             |
| /api/v2/2/labels/2 | role | Database        |
| ...            |     |                 |
| /api/v2/2/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 asterisk (*) wildcard.

Examples

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	Status
/api/v2/1/workloads/527b8aca-97aa-43b9-82e1-29b17a947cdd	hrm-web.webscaleone.info	false	flow_summary	active
/api/v2/1/workloads/4a8743a4-14ee-40d0-9ed2-990fe3f0ffb1	hrm-db.webscaleone.info	false	flow_summary	active

...

Linux: Save Fields 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.

Examples

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 ILO_DISPLAY_CONFIG environment variable

```
$ 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 [List of All Workloads \[22\]](#). The UUID is the last word of the value of the workload's href field, as shown in bold in the following example:

```
/api/v2/orgs/28/workloads/2ca0715a-b7e3-40e3-ade0-79f2c7adced0
```



```

| 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 states 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                                     | Name | Description | Enabled |
+-----+-----+-----+-----+
| /api/v2/orgs/28/sec_policy/draft/rule_sets/2387 | fool |             | true    |
| {"href"=>"/api/v2/users/74"} |      |             |         |
| /api/v2/orgs/28/sec_policy/draft/rule_sets/1909 | Default |           | true    | ...
200, OK
```

The state of the policy is stored in the agent/status/status field. See [Nested Resource Fields and Wildcards \[20\]](#) for information.

Import and Export Security Policy

Using the CLI Tool, you can export and import security policy to and from the PCE. Importing and 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.

Export and Import 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

You can export rules for workloads, virtual services, or virtual servers.

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. When 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 for 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 [Authenticate with the PCE \[14\]](#) for information.
- Export the policy to a file. Syntax summary:

```
ilo sec_policy export --file someExportFilename
```

- Authenticate to the target PCE. See [Authenticate with the PCE \[14\]](#) for information.
- 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 [About Exporting Rules \[25\]](#) for information.

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 \[24\]](#) for information.

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 [About Exporting Rules \[25\]](#) for information.

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.

Upload Vulnerability Data

This section describes how to use the `ilo` commands to upload vulnerability data to the PCE for analysis in Illumination.

After uploading the data, you can use Vulnerability Maps in the PCE web console to gain insights into the exposure of vulnerabilities and attack paths across your applications running in data centers and clouds. See the "Vulnerability Maps" topic in the Visualization Guide for information.

Add the License for Vulnerability Data Upload

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

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



NOTE

Before adding the license, you must first authenticate to the PCE. See [Authenticate with the PCE \[14\]](#) for information.

To add the license, you must be the organization owner or a be a user who 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"	Yes	The quoted path to the <code>license.json</code> file from Illumio Example: <code>"~/secretDir/license.json"</code>
"feature_name"	Yes	The quoted string <code>"vulnerability_maps"</code> , which specifies the feature name the license enables
debug	No	Enable debugging
v verbose	No	For verbose logging
trace	No	Enable API trace

Vulnerability Data Upload Process

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

Using API to Download Vulnerability Data

In release CLI 1.3, Tenable IO and tenable SC have been supporting both manual and API download of vulnerability data while Qualys tool was only available for manual download.



IMPORTANT

Starting from the release CLI 1.4, Qualys supports also API download, with some minor differences in options.

For the release CLI 1.4.1, it is suggested that users use an API key instead of a login session while using Qualys API download.

For the release CLI 1.4.2 for Tenable, the most reliable way to provide authentication is through API keys instead of username/password. If customers observe any authentication issues while using Tenable SC API upload, they are advised to use API keys for authentication.

There are 2 ENV variables to set up the Tenable SC API keys which are used for authentication:

`TSC_ACCESS_KEY`

`TSC_SECRET_KEY`

The API connects directly to the cloud instance of Tenable or Qualys and the vulnerability tool then scans new vulnerabilities and downloads them into the PCE.

Users can also set up cron jobs that run in the desired intervals and check the state of the vulnerability scanner.

Qualys and Tenable scanners work in a similar way, using the username and password and similar options.

Automating Vulnerability Imports from Tenable-SC

Users of Illumio vulnerability maps can automate the import of vulnerabilities from tenable-sc using a script.

Illumio CLI supports the API username and password as environment variables or a cmd line switch (such as `--api-password`).

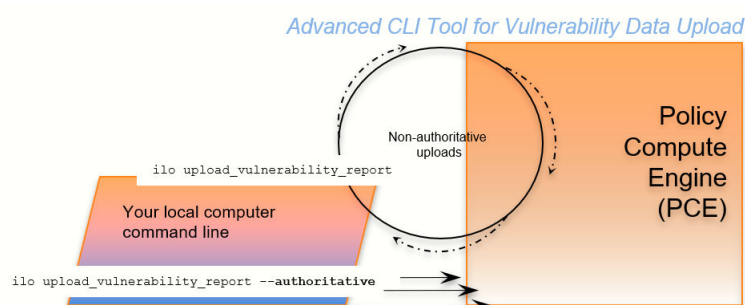
The ILO-CLI tool was updated to add a switch for `--api-user`.

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 “Vulnerability Maps” in the Visualization Guide for information.

Supported Vulnerability Data Sources

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

- Nessus Professional™
- Qualys®
- Tenable Security Center
- Tenable.io
- Rapid7®



NOTE

Before uploading Rapid7 data to the PCE, export the data from Rapid7 to Qualys format with Qualys XML Export.

Vulnerability Data Formats

In the CLI 1.4.0, 1.4.1 and 1.4.2 releases, Illumio supports the following report formats:

- For `tenable-io`: API, CSV
- For `tenable-sc`: API, CSV
- For `nessus-pro`: XML
- For `qualys`: API, XML

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 Single Uploaded Vulnerability \[35\]](#) for information.

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 [Add the License for Vulnerability Data Upload \[27\]](#) for information.
- XML-formatted vulnerability data files from one of the supported sources.
- Authenticated CLI-tool access to the target PCE. See [Authenticate with the PCE \[14\]](#) for information.
- Authenticated access and necessary permissions in the PCE web console for working with vulnerability maps. See [Authenticate with the PCE \[14\]](#) for information.

Vulnerability Data Upload CLI Tool Syntax

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

Where:

What	Required	Description
<code>--input-file</code> <code>path_to_datafile.xml</code> <code>[path_to_data-</code> <code>file.xml]...</code>	Yes	<p>Location of one or more data files to upload.</p> <p>The path to the data file can be either an absolute path or a relative path.</p> <p>If more than one data file is listed (bulk upload), separate the file names with space characters.</p>
<code>--debug</code>	No	Enable debugging
<code>--authoritative</code>	No	For uploading authoritative vulnerability data. The default command is without the <code>--authoritative</code> option. See Kinds of Vulnerability Data Uploads [29] for information.
<code>--workload-cache</code> <code>FILE</code>	No	DEBUGGING ONLY: Workload Cache file - use this if available
<code>--source-scanner</code> <code>[nessus-pro qualys </code> <code>tenable-sc]</code>	Yes	<p>Indicates the source of the scan. Note for rapid data:</p> <ul style="list-style-type: none"> Vulnerability data from Rapid must have been exported from Rapid in Qualys XML format. To load the Rapid data, use the 'qualys' argument
<code>--format</code>	Yes	Report format. Allowed values are:
<code>REPORT_FORMAT</code>		<p>xml</p> <ul style="list-style-type: none"> <code>--source-scanner nessus-pro</code> <code>--source-scanner qualys</code> <p>csv</p> <ul style="list-style-type: none"> <code>--source-scanner tenable-sc</code> <code>--source-scanner tenable-io</code> <p>api</p> <ul style="list-style-type: none"> <code>--source-scanner tenable-sc</code> <code>--source-scanner qualys</code> <code>--source-scanner nessus-pro</code> <p>See also <code>--api-server</code> and <code>--api-user</code>.</p>
<code>--api-server Sour-</code> <code>ceApiServer:port</code> <code>SERVER_FQDN</code>	<p>Yes for</p> <p>Tenable with</p> <p><code>--format</code></p> <p><code>api</code></p>	API server FQDN. Allowed formats are <code>HOST</code> or <code>HOST:PORT</code>
<code>--api-user ApiSer-</code> <code>verUserName</code> <code>USERNAME</code>	<p>Yes for</p> <p>source API</p> <p>server au-</p> <p>thentication</p>	<p>The user name for authenticating to the SourceApiServer.</p> <p>You are always prompted to enter your password.</p>
<code>--api-page-size</code> <code>PAGE_SIZE</code>	<p>Yes for</p> <p>Qualys and</p> <p>Tenable</p>	Appropriate page size if API supports pagination. The default page is 1000.

What	Required	Description
<code>--skip-cert-verification</code>	Yes for Qualys and Tenable	Disable certificate verification for API.
<code>--on-premise</code>	Yes only for Tenable io	Tenable IO deployment is on premise.
<code>--mitigated</code>	Yes only for Tenable sc	Tenable SC input is exported from the mitigated vulnerabilities analysis view.
<code>--scanned-after</code> <code>SCANNED_AFTER</code>	Yes for Qualys	Qualys users can select scan data to process after a certain date, in ISO 8601 format. When the optional <code>scanned-after</code> option is not provided, the system will pull all the historical vulnerability records from your Qualys account. If your account has historical records, it may take a very long time for the first time. With the <code>scanned-after</code> option, vulnerability data scanned after a certain date will be extracted and uploaded. It is recommended to include a certain scanned-after time if you use Qualys API upload option for the first time.
<code>--severities SEVERITIES</code>	No	Qualys API users can select vulnerabilities with defined severity levels to include in their report. Users can filter based on severity and avoid severity levels 1 and 2, which are often very informational and noisy. Example: <code>--only-include-severity=3,4,5</code> For Windows, be sure to include quotes around the severity levels: Example: <code>--only-include-severity="3,4,5"</code> NOTE: This option was added in Release 1.4.1
<code>-v, --verbose</code>	No	Verbose logging mode
<code>--trace</code>	No	Enable API trace mode

Using the ILO Command with Windows Systems

Windows systems take a maximum of four options with the ILO command for the vulnerability data upload. Users who choose to use more optional parameters need to set `api-server`, `username`, and `password` as the environmental variables to use other options in the command.

Work with Vulnerability Maps in Illumination

See "Vulnerability Maps" in the Visualization Guide for information.

Vulnerability Data Examples

Example - Upload Non-Authoritative Vulnerability Data

In this example, the `--source-scanner nessus-pro` option indicates that the data comes from Nessus Professional. On Windows, provide the absolute path to the data file. This Win-

dows 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. On Windows, you use the `--format qualys` option and the absolute path to the data file. 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, the prompt shows this is an authoritative upload.

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 overwrites 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 \[30\]](#) for information.

```
$ 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 \(CVE\) \[30\]](#) and [Vulnerability Identifier \[30\]](#) for information. 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 \(CVE\) \[30\]](#) and [Vulnerability Identifier \[30\]](#) for information.

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"} |
...
```

Example - Upload a Qualys Report Using API

```
upload_vulnerability_report --source-scanner qualys --format api
--api-server qualysguard.qg3.apps.qualys.com --api-user um3sg
--scanned-after 2021-09-20
```

CLI Tool Tutorials

This section provides several hands-on exercises that demonstrate step-by-step how to perform common tasks using the CLI Tool.

How to Import Traffic Flow Summaries

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:

```
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:

```
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 [Authenticate with the PCE \[14\]](#) for information.
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.

How to Create Kerberos-Authenticated Workloads

This tutorial describes how to create workloads that use Kerberos for authentication. The tutorial makes the following assumptions:

- 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 Installation and Upgrade 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' modes, 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 [Authenticate with the PCE \[14\]](#) for information.
2. Create Kerberos-authenticated `myWorkload1` and set its mode to `idle`:

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

For information about how the mode is a nested field, see [Nested Resource Fields and Wildcards \[20\]](#).

3. Create Kerberos-authenticated `myWorkload2` and set its mode to `illuminated`:

```
ilo workload create --hostname myPCE.BigCo.com --name myWorkload2
--service-principal-name host/myKerberosTicketGrantingServer@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.

How to 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:

1. You issue the desired `ilo` command with the `--async` option, which displays a job ID.
2. You take note of the job ID.
3. Your session is freed up while the job runs.
4. 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-eb56c4ffale3
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-eb56c4ffale
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
...
```

How to Upload Vulnerability Data

This example tutorial shows how to upload vulnerability data to the PCE. For more information, see [Upload Vulnerability Data \[27\]](#). The source of the vulnerability data in this example comes from Qualys®.

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 \[30\]](#) for information.

```
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.