ThreatQuotient

FireEye CMS Connector Guide

Version 3.4.1

June 28, 2021

ThreatQuotient
11400 Commerce Park Dr., Suite 200
Reston, VA 20191

Support
Email: support@threatq.com
Web: support.threatq.com
Phone: 703.574.9893

Warning and Disclaimer

ThreatQuotient, Inc. provides this document “as is”, without representation or warranty of any kind, express or implied, including without limitation any warranty concerning the accuracy, adequacy, or completeness of such information contained herein. ThreatQuotient, Inc. does not assume responsibility for the use or inability to use the software product as a result of providing this information.

Contents

Versioning

- Current integration version: `3.4.1`
- Supported on ThreatQ versions >= `4.34.0`
- FireEye Central Management version: `8.7.1`
- Support Python versions: `2.7`, `3.5`, `3.6`

There are two versions of this integration:

- Python 2 version
- Python 3 version

Introduction

This is a custom connector for a FireEye CMS instance. This connector is meant to attach to a single FireEye CMS instance.

The connector has several purposes:

- Pull alerts from FireEye CMS and upload the data as indicators and events to a ThreatQ instance. The events are tagged as "Malware" type events.
- Upload indicators from ThreatQ to the FireEye CMS instance by inputting one or multiple Data Collections in the UI.

The `-n` flag may be used for multiple instances, but it is not a tested use case.

Installation

The connector can be installed from the ThreatQuotient repository with YUM credentials or offline via a .whl file.

> **Upgrading Users** - Review the Change Log for updates to configuration parameters before updating. If there are changes to the configuration file (new/removed parameters), you must first delete the previous version's configuration file before proceeding with the install steps listed below. Failure to delete the previous configuration file will result in the connector failing.

1. Install the connector using one of the following methods:

   ThreatQ Repository

   a. Run the following command:

   ```
   <> pip install tq_conn_fireeye_cms
   ```

   Offline via .whl file

   To install this connector from a wheel file, the wheel file (.whl) will need to be copied via SCP into your ThreatQ instance.

   a. Download the connector whl file with its dependencies:

   ```
   <> mkdir /tmp/tq_conn_fireeye_cms

      pip download tq_conn_fireeye_cms -d

      /tmp/tq_conn_fireeye_cms/
   ```

   b. Archive the folder with the .whl files:

   ```
   <> tar -czvf tq_conn_fireeye_cms.tgz /tmp/
      tq_conn_fireeye_cms/
   ```

   c. Transfer all the whl files, the connector and all the dependencies, to the ThreatQ instance.

   d. Open the archive on ThreatQ:

   ```
   <> tar -xvf tq_conn_fireeye_cms.tgz
   ```

e. Install the connector on the ThreatQ instance.

> The example assumes that all the whl files are copied to `/tmp/conn` on the ThreatQ instance.

```
<> pip install /tmp/conn/tq_conn_fireeye_cms-<version>-
   <python version>-none-any.whl --no-index --find-links /
   tmp/conn/
```

> pip install /tmp/conn/tq_conn_fireeye_cms-3.4.0-py2-none-any.whl --no-index --find-links /tmp/conn/

> Two executables will be installed: `tq-conn-fireeye-cms` for ingesting alerts in ThreatQ and `tq-conn-fireeye-ioc-sync` for exporting indicators of compromise from ThreatQ to FireEye CMS.

2. Once the application has been installed, a directory structure must be created for all configuration, logs and files, using the `mkdir -p` command. Use the commands below to create the required directories:

```
<> mkdir -p /etc/tq_labs/
   mkdir -p /var/log/tq_labs
```

3. Perform an initial run using the following command:

```
<> tq-conn-fireeye-cms -c /etc/tq_labs/ -ll /var/log/tq_labs/ -v3
```

4. Enter the following parameters when prompted:

PARAMETER	DESCRIPTION
ThreatQ Host	This is the host of the ThreatQ instance, either the IP Address or Hostname as resolvable by ThreatQ.
Client ID	This is the OAuth id that can be found at Settings Gear → User Management → API details within the user's details.
Email Address	This is the User in the ThreatQ System for integrations.
Password	The password for the above ThreatQ account.

PARAMETER	DESCRIPTION
Status	This is the default status for objects that are created by this Integration. It is common to set this to "Active", but Organization SOPs should be respected when setting this.

Example Output

```
tq-conn-fireeye-cms -c /etc/tq_labs/ -ll /var/log/tq_labs/ -v3
ThreatQ Host: <ThreatQ Host IP or Hostname>
Client ID: <ClientID>
E-Mail Address: <EMAIL ADDRESS>
Password: <PASSWORD>
Status: Active
Connector configured. Set information in UI
```

You will still need to configure and then enable the connector.

Configuration

> ThreatQuotient does not issue API keys for third-party vendors. Contact the specific vendor to obtain API keys and other integration-related credentials.

To configure the integration:

1. Navigate to your integrations management page in ThreatQ.
2. Select the **Labs** option from the *Category* dropdown (optional).

   > If you are installing the connector for the first time, it will be located under the **Disabled** tab.

3. Click on the integration to open its details page.
4. Enter the following parameter under the **Configuration** tab:

PARAMETER	DESCRIPTION
FireEye CMS Hostname or IP Address	The hostname or the IP Address of the FireEye CMS instance.
Username	The FireEye CMS user with access to the API and alert information.
Password	The password for authenticating with FireEye CMS.
Alert Duration	The interval in which to search for alerts, in hours. Options include (hours): 1, 2, 6, 12, 24, and 48.
Select the severity level of the alerts to ingest from FireEye CMS	Select the severity level of the alerts to reduce the level of noise ingested from FireEye CMS.

PARAMETER	DESCRIPTION
Exclude alerts with the following action	Select the alerts that need to be excluded from ingestion in ThreatQ. Valid values are **Blocked** and **Notified**.
List of Data Collections (Optional)	Enter the names of the data collection(s) with indicators to send to FireEye. Multiple data collection names should be comma-delimited. This field is optional.

5. Review the **Settings** configuration, make any changes if needed, and click on **Save**.
6. Click on the toggle switch, located above the *Additional Information* section, to enable it.

Usage

The following section contains steps, commands, and arguments required to run the connector.

FireEye CMS Driver (ingest alerts)

Use the following command to execute the FireEye CMS driver:

```
<> tq-conn-fireeye-cms -c /etc/tq_labs/ -ll /var/log/tq_labs/ -ds -v3
```

By default, the FireEye CM instance uses a self-signed certificate. To disable SSL verification use the `-ds` flag.

If you want to parse data from a JSON file (instead of the API), append the `-f /path/to/your/file` flag to the end of your command. This will parse the file, but will not change the internal time record for when this connector last connected to FireEye CM. Ingesting duplicate data will not cause issues, as the data will be recorded the same way.

FireEye IoC Driver (sync data collections)

The integration offers the capability to synchronize indicators from ThreatQ data collections with custom lists in FireEye.

To execute the sync, use the command:

```
<> tq -conn-fireeye-ioc-sync -c /etc/tq_labs/ -ll /var/log/tq_labs/
   -v3
```

Command Line Arguments

> All location-based options default to the current working directory if they are not provided. To find additional options and option descriptions, invoke the program with `-h`.

This connector supports the following custom command line arguments:

ARGUMENT	DESCRIPTION
`-h`, `--help`	Shows this help message and exits.
`-n`, `--name`	Sets the name for this connector. In some cases, it is useful to have multiple connectors of the same type executing against a single TQ instance. For example, the Syslog Exporter can be run against multiple target and multiple exports, each with their own name and configuration
`-ll LOGLOCATION`, `--loglocation LOGLOCATION`	Sets the logging location for the connector. The location should exist and be writable by the current. A special value of 'stdout' means to log to the console (this happens by default).
`-c CONFIG`, `--config CONFIG`	This is the location of the configuration file for the connector. This location must be readable and writable by the current user. If no config file path is given, the current directory will be used. This file is also where some information from each run of the connector may be put (last run time, private oauth, etc.)
`-v {1,2,3}`, `--verbosity {1,2,3}`	This is the logging verbosity level where **3** means everything. A special value of `stdout` means to log to the console. This is by default.
`-f, --file`	Allows you to specify parsing JSON of a file instead of using the API endpoint of FireEye CMS.

ARGUMENT	DESCRIPTION
`-ds, --disable_ssl`	This allows you to disable SSL verification to all requests to the FireEye CMS API.
`-ep, -external-proxy`	This allows you to use the proxy that is specified in the ThreatQ UI.

CRON

Automatic CRON configuration has been removed from this script. To run this script on a recurring basis, use CRON or some other jobs scheduler. The argument in the CRON script must specify the config and log locations.

Add an entry to your Linux crontab to execute the connector at a recurring interval. Depending on how quickly you need updates, this can be run multiple times a day (no more than once an hour) or a few times a week.

In the example below, the command will execute the connector every two hours.

1. Log into your ThreatQ host via a CLI terminal session.
2. Enter the following command:

   ```
   <> crontab -e
   ```

   This will enable the editing of the crontab, using vi. Depending on how often you wish the cronjob to run, you will need to adjust the time to suit the environment.

3. Enter the commands below:

   Every 2 Hours Example

   ```
   <> 0 */2 * * * tq-conn-fireeye-cms -c /etc/tq_labs/ -ll /var/log/
      tq_labs/ -v3
   ```

4. Save and exit CRON.

Connector Endpoints

Login

	DESCRIPTION
Purpose	Connects to the FireEye CMS instance and logs in, getting the required header.
Endpoint	`POST /wsapis/v2.0.0/auth/login`
API Response	Empty body of the response. The header contains the X-FeApi-Token which is used for further API requests.

Search for Alerts

	DESCRIPTION
Purpose	Collects alerts from FireEye CMS from a specific historical period.
Endpoint	`GET /wsapis/v2.0.0/alerts`

API Response:


```
{
  "alert": [
    {
      "explanation": {
        "malwareDetected": {
          "malware": [
            {
              "md5Sum": "fe4d8f227520e2468dd1019496ef0604",
              "sha256": "b71e3012e93f11a7b0b179ea54eeb0e787d02acc48705833e414a5e57a6a2032",
              "name": "Malware.Binary.FEC2",
              "originalInfectionId": 3181,
              "originalInfectionType": "MALWARE_OBJECT",
              "originalInfectionUrl": "https://10.11.113.143/botnets/events_for_bot?ma_id=3181"
            },
```

```
      ]
    },
    "osChanges": [
      {
        "application": {
          "app-name": "Windows Explorer"
        },
        "os": {...},
        "file_informational": [...],
        "uac": [...],
        "end-of-report": "",
        "process_informational": [...],
        "os_monitor": {
          "date": "Sep 19 2018",
          "build": 795854,
          "time": "12:59:48",
          "version": "17R1.7"
        },
        "malicious-alert": [
          {
            "classtype": "static_log",
            "display-msg": "Static Analysis"
          },
          {
            "classtype": "Static-Analysis",
            "display-msg": "Static Analysis"
          },
          {
            "classtype": "static_log",
            "display-msg": "Static Analysis"
          },
          {
            "classtype": "Static-Analysis",
            "display-msg": "Static Analysis"
          },
          {
            "classtype": "sa_only",
            "display-msg": "Heuristic"
          }
        ],
        "analysis": {
          "mode": "malware",
          "product": "MPS",
          "ftype": "exe",
          "version": 1.3977
        }
      },
    ]
  },
  "src": {
    "ip": "56.204.181.67",
    "mac": "00:20:18:11:ff:45",
    "port": 0
  },
  "alertUrl": "https://qa-cm7500-4-9-20/event_stream/events_for_bot?ma_id=12345",
  "action": "notified",
  "occurred": "2018-10-18 16:46:41 +0000",
  "dst": {
    "mac": "02:14:17:da:c9:2f",
    "port": 0,
```

```
      "ip": "249.207.161.251"
    },
    "applianceId": "000BABCD66F2",
    "id": 12345,
    "rootInfection": 3181,
    "sensorIp": "10.11.113.155",
    "name": "MALWARE_OBJECT",
    "severity": "MAJR",
    "uuid": "c9391258-1a79-4b54-be8e-144ddb5f118f",
    "ack": "yes",
    "product": "WEB_MPS",
    "sensor": "cms-nx2500-3",
    "vlan": 0,
    "malicious": "yes"
   }
 ],
 "appliance": "CMS",
 "version": "CMS (CMS) 8.4.0.805144",
 "msg": "normal",
 "alertsCount": 1
}
```


Upload Indicators from ThreatQ data collection to FireEye custom IOC file

	DESCRIPTION
Purpose	Upload indicators from ThreatQ data collection to FireEye custom IOC file.
Endpoint	`POST /wsapis/v2.0.0/customioc/feed/add`

Deletes a list of Custom IOCs in FireEye CM

	DESCRIPTION
Purpose	This action deleted a custom IOC list, if it exists, in FireEye.
Endpoint	`POST /wsapis/v2.0.0/customioc/feed/delete/<list name>`

Gets the contents of a custom IOC list from FireEye CM

	DESCRIPTION
Purpose	Gets the contents of a custom IOC list from FireEye CMS.
Endpoint	`GET /wsapis/v2.0.0/customioc/feed/download/<list name>`

Logout

	DESCRIPTION
Purpose	Logs out the session of the FireEye CMS instance.
Endpoint	`POST /wsapis/v2.0.0/auth/logout`
API Response	If the logout is successful, the API returns code 200 with an empty body.

Average Connector Runs

Object counts and connector runtimes are supplied as generalities only - objects returned by a provider can differ based on credential configurations and Feed runtime may vary based on system resources and load.

METRIC	RESULT
Run Time	3
Export Indicators	10,000
Add Indicator Attributes	160
Create ThreatQ Events	10
Create Indicators	50

Change Log

- **Version 3.4.1**
  - The **Exclude alerts with the following action** UI configuration is now optional.
- **Version 3.4.0**
  - Modified the UI configuration page for the connector.
  - Added the ability to filter alerts based on severity (Minor, Major, Critical, Unknown).
  - Added the ability to exclude alerts based on the action taken on the alert (Blocked or Notified).
- **Version 3.3.1**
  - Updated Parameter naming - **Saved Search** is now known as **Data Collection**.
- **Version 3.3.0**
  - Added Python 3 support.
  - Updated search system to use Data Collections / Saved Searches.
- **Version 3.2.1**
  - Added the ability to retrieve and upload data from FireEye CMS.
- **Version 2.0.0**
  - Minor documentation updates/enhancements.
- **Version 1.1.0**
  - Minor documentation updates/enhancements.
- **Version 1.0.0**
  - Initial Release