

ExtraHop 8.0

Upload STIX files through the REST API

Published: 2020-07-20

Published: 2020-07-20

Published: 2020-07-20

Threat collections enable your Reveal(x) system to identify suspicious IP addresses, hostnames, and URLs found in your network activity. While an ExtraHop-curated threat collection is available by default, you can also upload a custom threat collection from free or commercial sources.

Before you begin

- You must have unlimited [privileges](#) to generate an API key.
- You must have a valid API key to make changes through the REST API and complete the procedures below. (See [Generate an API key](#).)
- Familiarize yourself with [Threat intelligence](#).
- Familiarize yourself with the [ExtraHop REST API Guide](#) to learn how to navigate the ExtraHop REST API Explorer.

Threat collections must be added and updated to all connected Command and Discover appliances. And because these sources are often updated frequently, the REST API provides the opportunity to automate updates for threat collections to all appliances.

Custom threat collections must be formatted in Structured Threat Information Expression (STIX) as TAR files. Reveal(x) currently supports STIX versions 1.0 - 1.2.

Python script example

The following Python script shows you how to upload all STIX files in a given directory to a list of ExtraHop appliances. First, the script reads through a CSV file that contains the URLs and API keys for each appliance. For each appliance, the script gets a list of all threat collections that are already on the appliance. The script then processes each STIX file in the directory for each appliance.

If the name of the file matches the name of a threat collection on the appliance, the script overwrites the threat collection with the file contents. If there are no threat collection names that match the file name, the script uploads the file to create a new threat collection.

Each row of the CSV file must contain the following columns in the specified order:

HTTPS URL	API key
-----------	---------

 **Note:** The script does not accept a header row in the CSV file.

The script includes the following configuration variables that you must replace with information from your environment:

- **APPLIANCE_LIST:** The path of the CSV file with the HTTPS URLs and API keys of the appliances
- **STIX_DIR:** The path of the directory that contains the STIX files

```
#!/usr/bin/python3
import json
import os
```

```

import requests
import csv

APPLIANCE_LIST = 'appliances.csv'
STIX_DIR = 'stix_dir'

# Get appliance hostnames and Read through appliance URL and API key list
appliances = []
with open(APPLIANCE_LIST, 'rt', encoding='ascii') as f:
    reader = csv.reader(f)
    for row in reader:
        appliance = {
            'host': row[0],
            'api_key': row[1]
        }
        appliances.append(appliance)

# Get a json object of every threat collection on the appliance
def getCollections():
    url = host + 'api/v1/threatcollections'
    r = requests.get(url, headers=headers)
    print(r.status_code)
    return r.json()

# Check which STIX files have already been uploaded
def check_files(collections):
    update_list = []
    skip_list = []
    # Get a list of all stix files in the STIX_DIR directory
    names = []
    for dir, subdirs, files in os.walk(STIX_DIR):
        for file in files:
            if file.endswith(('.tar')):
                name = file.split('.')[0]
                names.append(name)
    # Check each threat collection for names that match the STIX file names
    for c in collections:
        c_name = c['name']
        if c_name in names:
            update_list.append(c)
            skip_list.append(c_name)
    return update_list, skip_list

# Process each file in the STIX_DIR directory
# If it is for a new threat collection, add the threat collection
# If there is already a threat collection for the file, update the threat
collection
def process_files(update_files, skip_list):
    for dir, subdirs, files in os.walk(STIX_DIR):
        for file in files:
            name = file.split('.')[0]
            if file.endswith(('.tar')) and name not in skip_list:
                upload_new(file, dir)
            else:
                for c in update_files:
                    if c['name'] == name:
                        update_old(file, dir, c)

# Upload a new threat collection
def upload_new(file, dir):
    print('Uploading ' + file + ' on ' + host)
    url = host + 'api/v1/threatcollections'
    file_path = os.path.join(dir, file)
    name = file.split('.')[0]

```

```
files = {'file': open(file_path, 'rb')}
values = {'name': name}
r = requests.post(url, data=values, files=files, headers=headers)
print(r.status_code)
print(r.text)

# Update an existing threat collection
def update_old(file, dir, c):
    print('Updating ' + file + ' on ' + host)
    url = host + 'api/v1/threatcollections/~' + str(c['user_key'])
    file_path = os.path.join(dir, file)
    files = {'file': open(file_path, 'rb')}
    r = requests.put(url, files=files, headers=headers, verify=False)
    print(r.status_code)
    print(r.text)

# Process STIX files for each appliance
for appliance in appliances:
    host = appliance['host']
    api_key = appliance['api_key']
    headers = {'Authorization': 'ExtraHop apikey=%s' % api_key}
    collections = getCollections()
    update_files, skip_list = check_files(collections)
    process_files(update_files, skip_list)
```



Note: If the script returns an error message that the SSL certificate verification failed, make sure that [a trusted certificate has been added to your appliance](#). Alternatively, you can add the `verify=False` option to bypass certificate verification. However, this method is not secure and is not recommended. The following code sends an HTTP GET request without certificate verification:

```
requests.get(url, headers=headers, verify=False)
```