Create a device group through the REST API

Published: 2020-06-08

You can create a large number of complex device groups through the REST API by referencing a CSV file exported from a third-party application. In this topic, we show methods for creating a device group through both the ExtraHop REST API Explorer and a Python script.

Create a device group through the REST API Explorer

1. In a browser, navigate to the REST API Explorer.

   The URL is the hostname or IP address of your ExtraHop Discover or Command appliance, followed by `/api/v1/explore/`. For example, if your hostname is seattle-eda, the URL is `https://seattle-eda/api/v1/explore/`.
2. Click **Enter API Key** and then paste or type your API key into the **API Key** field.
3. Click **Authorize** and then click **Close**.
4. Click **Device Group** and then click **POST /devicegroups**.
5. Click **Try it out**.

   The JSON schema is automatically added to the body parameter text box.
6. In the body field, specify properties for the device group that you want to create.

   For example, the following body creates a device group that includes CIDR blocks `192.168.0.0/26`, `192.168.0.64/27`, and `192.168.0.96/30`:

```
{
    "name": "New group",
    "description": "A newly created group",
    "filter": {
        "rules": [
            {
                "field": "ipaddr",
                "operand": "192.168.0.0/26",
                "operator": "="
            },
            {
                "field": "ipaddr",
                "operand": "192.168.0.64/27",
                "operator": "="
            },
            {
                "field": "ipaddr",
                "operand": "192.168.0.96/30",
                "operator": "="
            }
        ],
        "operator": "or"
    }
}
```

7. Click **Send Request**.

Python script example

The following Python script creates device groups by reading criteria from a CSV file that meets the following specifications:

- The CSV file must not contain a header row.
- Each row of the CSV file must contain the following three columns in the specified order:

Device group name	Description	IP address or CIDR block

- Each column after the first required three columns must specify an IP address or CIDR block for the device group.

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

- **HOST:** The IP address or hostname of the Discover appliance
- **API_KEY:** The API key
- **CSV_FILE:** The file that contains the list of device groups

```
#!/usr/bin/python3

import json
import requests
import csv
import os.path

HOST = 'extrahop.example.com'
API_KEY = '123456789abcdefghijklmnop'
CSV_FILE = 'device_group_list.csv'

headers = {'Content-Type': 'application/json',
           'Accept': 'application/json',
           'Authorization': 'ExtraHop apikey=%s' % API_KEY}

def readCSV():
    devices = []
    with open(CSV_FILE, 'rt', encoding='ascii') as f:
        reader = csv.reader(f)
        for row in reader:
            device = {}
            device['name'] = row.pop(0)
            device['description'] = row.pop(0)
            rules = []
            for ip in row:
                rules.append({
                    'field': 'ipaddr',
                    'operand': ip,
                    'operator': '='
                })
            device['filter'] = {
                    'rules': rules,
                    'operator': 'or'
            }
            devices.append(device)
    return devices

def createDevice(device):
    url = HOST + '/api/v1/devicegroups'
    r = requests.post(url, headers=headers, data=json.dumps(device))
    if r.status_code != 201:
        print ("Could not create device: " + device['name'])
        print (r.status_code)
        print (r.json())
    else:
        print ("Created custom device: " + device['name'])
```

```
devices = readCSV()
for device in devices:
    createDevice(device)
```

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)
```