Extract the device list through the REST API

Published: 2024-02-23

The ExtraHop REST API enables you to extract the list of devices discovered by the sensor or console. By extracting the list with a REST API script, you can export the list in a format that can be read by third-party applications, such as a configuration management database (CMDB). In this topic, we show methods for extracting a list through both the cURL command and a Python script.

Before you begin

- For sensors and ECA VMs, you must have a valid API key to make changes through the REST API and complete the procedures below. (See Generate an API key.)
- For Reveal(x) 360, you must have valid REST API credentials to make changes through the REST API and complete the procedures below. (See Create REST API credentials.)

Retrieve the device list with the cURL command

The device list includes all device metadata, such as MAC addresses and device IDs. However, you can filter the list of devices with a JSON parser to extract the specific information you want to export. In this example, the device list is retrieved and then filtered with the jq parser to only extract the display name of each device.

Note: The following procedure is not compatible with the Reveal(x) 360 REST API. To retrieve the device list from Reveal(x) 360, see Retrieve the device list from Reveal(x) 360 with the cURL command.

Before you begin

- The cURL tool must be installed on your machine.
- The jq parser must be installed on your machine. For more information, see https://stedolan.github.io/jq/.

Open a terminal application and run the following command, where `YOUR_KEY` is the API for your user account, `HOSTNAME` is the hostname of your sensor or console, and `MAX_DEVICES` is a number large enough to be more than the total number of devices discovered by your system:

```
curl -s -X POST "https://HOSTNAME/api/v1/devices/search" --header
 "accept: application/json" --header "Authorization: ExtraHop
 apikey=YOUR_KEY" --header "Content-Type: application/json" -d
 "{ \"active_from\": 1, \"active_until\": 0, \"limit\": MAX_DEVICES}" |
 jq -r '.[] | .display_name'
```

Note: If the command returns no results, make sure that a trusted certificate has been added to your ExtraHop system. Alternatively, you can add the `--insecure` option to retrieve the device list from an ExtraHop system without a trusted certificate; however, this method is not secure and not recommended.

Tip: You can append the `select(.analysis == "LEVEL")` option to filter results by analysis level. For example, the following command limits the results to include only devices that are selected for advanced analysis:

```
curl -s -X POST "https://HOSTNAME/api/v1/devices/search" --
header "accept: application/json" --header "Authorization:
 ExtraHop apikey=YOUR_KEY" --header "Content-Type: application/
json" -d "{ \"active_from\": 1, \"active_until\": 0, \"limit\":
 1000000000}" | jq -r '.[] | select(.analysis == "advanced")
 | .display_name'
```

Tip: You can append the `select(.critical == BOOLEAN)` option to filter results by the critical field. For example, the following command limits the results to include only devices that are identified as critical by the ExtraHop system:

```
curl -s -X POST "https://HOSTNAME/api/v1/devices/search" --
header "accept: application/json" --header "Authorization:
 ExtraHop apikey=YOUR_KEY" --header "Content-Type: application/
json" -d "{ \"active_from\": 1, \"active_until\": 0, \"limit
\": 1000000000}" | jq -r '.[] | select(.critical == true)
 | .display_name'
```

Tip: You can append the `select(.cloud_instance_name != null)` option to filter results by the cloud instance name field. For example, the following command limits the results to include only devices with a cloud instance name:

```
curl -s -X POST "https://HOSTNAME/api/v1/devices/search" --
header "accept: application/json" --header "Authorization:
 ExtraHop apikey=YOUR_KEY" --header "Content-Type: application/
json" -d "{ \"active_from\": 1, \"active_until\": 0, \"limit
\": 1000000000}" | jq -r '.[] | select(.cloud_instance_name !=
 null) | .cloud_instance_name'
```

Retrieve the device list from Reveal(x) 360 with the cURL command

The device list includes all device metadata, such as MAC addresses and device IDs. However, you can filter the list of devices with a JSON parser to extract the specific information you want to export. In this example, the device list is retrieved and then filtered with the jq parser to only extract the display name of each device.

Note: The following procedure is only compatible with the Reveal(x) 360 REST API. To retrieve the device list from sensors and ECA VMs, see Retrieve the device list with the cURL command.

Before you begin

- The cURL tool must be installed on your machine.
- The jq parser must be installed on your machine. For more information, see https://stedolan.github.io/jq/.

1. Open a terminal application and run the following command, where `REVEAL_X_360_REST_API` is the hostname of the Reveal(x) 360 API. This hostname is displayed in Reveal(x) 360 on the API Access page under API Endpoint. The hostname does not include the `/oauth2/token`:

   ```
   HOST="https://REVEAL_X_360_REST_API"
   ```

2. Run the following command, where `YOUR_ID` is the ID of the REST API credentials:

   ```
   ID="YOUR_ID"
   ```

3. Run the following command, where `YOUR_SECRET` is the secret of the REST API credentials:

   ```
   SECRET="YOUR_SECRET"
   ```

4. Run the following command:

   ```
   AUTH=$(printf "$ID:$SECRET" | base64 --wrap=0)
   ```

5. Run the following command:

   ```
   ACCESS_TOKEN=$(curl -s \
   ```

```
    -H "Authorization: Basic ${AUTH}" \
    -H "Content-Type: application/x-www-form-urlencoded" \
    --request POST \
     ${HOST}/oauth2/token \
    -d "grant_type=client_credentials" \
    | jq -r '.access_token')
```

6. Run the following command, where `MAX_DEVICES` is a number large enough to be more than the total number of devices discovered by your system:

```
curl -s -X GET -H "Authorization: Bearer ${ACCESS_TOKEN}"  "$HOST/api/
v1/devices?active_from=1&active_until=0&limit=MAX_DEVICES" | jq -r '.[]
 | .display_name'
```

Tip: You can append the `select(.analysis == "LEVEL")` option to filter results by analysis level. For example, the following command limits the results to include only devices that are selected for advanced analysis:

```
curl -s -X GET -H "Authorization: Bearer
 ${ACCESS_TOKEN}"  "$HOST/api/v1/devices?
active_from=1&active_until=0&limit=10000000000" | jq -r '.[] |
 select(.analysis == "advanced") | .display_name'
```

Tip: You can append the `select(.critical == BOOLEAN)` option to filter results by the critical field. For example, the following command limits the results to include only devices that are identified as critical by the ExtraHop system:

```
curl -s -X GET -H "Authorization: Bearer
 ${ACCESS_TOKEN}"  "$HOST/api/v1/devices?
active_from=1&active_until=0&limit=10000000000"  | jq -r '.[] |
 select(.critical == true) | .display_name'
```

Tip: You can append the `select(.cloud_instance_name != null)` option to filter results by the cloud instance name field. For example, the following command limits the results to include only devices with a cloud instance name:

```
curl -s -X GET -H "Authorization: Bearer
 ${ACCESS_TOKEN}"  "$HOST/api/v1/devices?
active_from=1&active_until=0&limit=10000000000"  | jq -r '.[] |
 select(.cloud_instance_name != null) | .cloud_instance_name'
```

Retrieve and run the example Python script

The ExtraHop GitHub repository contains an example Python script that extracts the device list, including all device metadata, and writes the list to a CSV file in the same directory as the script.

1. Go to the ExtraHop code-examples GitHub repository and download the `extract_device_list/extract_device_list.py` file to your local machine.
2. In a text editor, open the `extract_device_list.py` file and replace the configuration variables with information from your environment.
    - For sensors and ECA VMs, specify the following configuration variables:
        - **HOST:** The IP address or hostname of the sensor or ECA VM.
        - **API_KEY:** The API key.
        - **CSV_FILE:** The file that contains the list of device groups.
        - **FILENAME:** The file that output will be written to

  - **LIMIT:** The maximum number of devices to retrieve with each GET request
  - **SAVEL2:** Retrieves L2 parent devices. This variable is valid only if you have enabled the ExtraHop system to discover devices by IP address.
  - **ADVANCED_ONLY:** Retrieves only devices that are currently under advanced analysis
  - **HIGH_VALUE_ONLY:** Retrieves only devices that are considered high value
- For Reveal(x) 360, specify the following configuration variables:
  - **HOST:** The hostname of the Reveal(x) 360 API. This hostname is displayed in the Reveal(x) 360 API Access page under API Endpoint. The hostname does not include the `/oauth2/token`.
  - **ID:** The ID of the Reveal(x) 360 REST API credentials.
  - **SECRET:** The secret of the Reveal(x) 360 REST API credentials.
  - **CSV_FILE:** The file that contains the list of device groups.
  - **FILENAME:** The file that output will be written to
  - **LIMIT:** The maximum number of devices to retrieve with each GET request
  - **SAVEL2:** Retrieves L2 parent devices. This variable is valid only if you have enabled the ExtraHop system to discover devices by IP address.
  - **ADVANCED_ONLY:** Retrieves only devices that are currently under advanced analysis
  - **HIGH_VALUE_ONLY:** Retrieves only devices that are considered high value

3. Run the following command:

```
python3 extract_device_list.py
```

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 sensor or console. Alternatively, you can add the `verify=False` option to bypass certificate verification. However, this method is not secure and not recommended. The following code sends an HTTP GET request without certificate verification:

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