Query for records through the REST API

Published: 2020-06-08

The ExtraHop REST API enables you to query for records stored on an ExtraHop explore appliance. By querying records with a REST API script, you can import records into a third party application, such as Microsoft Excel. Also, if your query matches more than the maximum number of records returned by the REST API, you can configure the script to recursively query for the remaining records. In this topic, we show methods for querying records through both the ExtraHop REST API Explorer and a Python script.

Before you begin

- You must log in to the ExtraHop system with an account that has full write 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 the ExtraHop REST API Guide to learn how to navigate the ExtraHop REST API Explorer.

Query records 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 **Record Log** and then click **POST /records/search**.
5. Click **Try it out**.
   The JSON schema is automatically added to the body parameter text box.
6. In the body text box, specify fields for your record query.
   For example, the following fields retrieve records from the last 30 minutes that include an IP address, domain name, or URI that has been identified as suspicious according to threat intelligence found in Reveal(x):

```
{
    "from": "-30m",
    "filter": {
        "field": "ex.isSuspicious",
        "operator": "=",
        "operand": {
            "type": "boolean",
            "value": "true"
        }
    }
}
```

   For a complete list of valid fields, see the Body Parameters section under **POST /records/search** in the REST API Explorer.

Python script examples

The following Python scripts query for records that involve an IP address, domain name, or URI that has been identified as suspicious according to threat intelligence found in Reveal(x). The scripts then write specified record fields to a CSV file that can be viewed in a spreadsheet program.

Note: For more information about threat intelligence with ExtraHop, see Threat intelligence and Upload STIX files through the REST API.

Script example with an Explore appliance

The following Python script retrieves records from an Explore appliance.

Important: If the query matches more than the maximum number of records that can be retrieved at once, the script retrieves the remaining records by sending a cursor to the ExtraHop system with the POST /records/cursor operation. This operation is only valid with Explore appliances. If you have configured a third-party recordstore, see Script example with a third-party recordstore.

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. Note that this hostname is not the hostname of the connected Explore appliance that the records are stored on.
- **APIKEY:** The API key.
- **FILENAME:** The file that output is written to.
- **TIME_LIMIT:** If the record query matches more than 100 records, the amount of time after the initial query that the remaining records can be retrieved from the appliance.
- **QUERY:** The record query parameters.
- **COLUMNS:** The record fields that are written to the CSV output file.

```
#!/usr/bin/python3

import json
import requests
import unicodecsv as csv

HOST = 'extrahop.example.com'
API_KEY = '123456789abcdefghijklmnop'
FILENAME = "records.csv"
TIME_LIMIT = '1m'
QUERY = {
    "context_ttl": TIME_LIMIT,
    "from": "-30m",
    "filter": {
        "field": "ex.isSuspicious",
        "operator": "=",
        "operand": {
            "type": "boolean",
            "value": "true"
        }
    }
}
COLUMNS =
 ['timestamp','sender','senderAddr','senderPort','receiver','receiverAddr','receiverPort

# Method that performs an initial record query on an ExtraHop system
def recordQuery(query):
    url = HOST + '/api/v1/records/search'
```

```
    headers = {'Authorization': 'ExtraHop apikey=%s' % API_KEY}
    r = requests.post(url, headers=headers, data=json.dumps(query))
    try:
        return json.loads(r.text)
    except:
        print('Record query failed')
        print(r.text)
        print(r.status_code)

# Method that retrieves remaining records from a record query
def continueQuery(cursor):
    url = HOST + '/api/v1/records/cursor'
    headers = {'Authorization': 'ExtraHop apikey=%s' % API_KEY}
    query = {'cursor': cursor}
    r = requests.post(url, headers=headers, data=json.dumps(query))
    try:
        return json.loads(r.text)
    except:
        print ('Record query failed')
        print (r.text)
        print (r.status_code)

# Query records from appliance
response = recordQuery(QUERY)
records = response['records']
if 'cursor' in response:
    response_cursor = response['cursor']
    retrieved = len(records)
    while retrieved > 0:
        print('Retrieved  ' + str(len(records)) + ' of ' +
 str(response['total']) + ' total records')
        response = continueQuery(response_cursor)
        newRecords = response['records']
        retrieved = len(newRecords)
        records = records + newRecords

print('Total records retrieved = ' + str(len(records)))

# Simplify and format records for CSV
table = []
for record in records:
    row = {}
    fields = record['_source']
    for column in COLUMNS:
        try:
            value = fields[column]
            # Retrieve isSuspicious field from ex object
            if column == 'ex':
                try:
                    row['isSuspicious'] = value['isSuspicious']
                except:
                    row[column] = value
            # Concatenate values returned as lists
            elif type(value) is list:
                row[column] = ' '.join(value)
            # Retrieve values from dict objects
            elif type(value) is dict:
                try:
                    # If value is a list, concatenate list
                    if type(value['value']) is list:
                        row[column] = ' '.join(value['value'])
                    else:
                        row[column] = value['value']
                except:
```

```
                    row[column] = value
            else:
                row[column] = value
        except:
            row[column] = ''
    table.append(row)


# Write records to csv
with open(FILENAME, 'wb') as csvfile:
    csvwriter = csv.writer(csvfile, encoding='utf-8')
    csvwriter.writerow(list(table[0].keys()))
    for row in table:
        csvwriter.writerow(list(row.values()))
```

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)
```

Script example with a third-party recordstore

The following Python script retrieves records from a third-party recordstore.

Note: If the query matches more than the maximum number of records that can be retrieved at once, the script retrieves the remaining records by sending additional requests with the `offset` parameter. The offset parameter skips a specified number of records in a query.

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. Note that this hostname is not the hostname of the connected Explore appliance that the records are stored on.
- **APIKEY:** The API key.
- **FILENAME:** The file that output is written to.
- **LIMIT:** The maximum number of records to retrieve at a time.
- **QUERY:** The record query parameters.
- **COLUMNS:** The record fields that are written to the CSV output file.

```
#!/usr/bin/python3

import json
import requests
import unicodecsv as csv

HOST = 'extrahop.example.com'
API_KEY = '123456789abcdefghijklmnop'
FILENAME = "records.csv"
LIMIT = 1000

QUERY = {
    "from": 1586273860000,
    "until": 1586273860500,
    "limit": LIMIT,
    "filter": {
        "field": "ex.isSuspicious",
```

```
        "operator": "=",
        "operand": {
            "type": "boolean",
            "value": "true"
        }
    },
    "sort": [
      {
        "direction": "asc",
        "field": "ipaddr"
      }
    ]
}

COLUMNS =
 ['timestamp','sender','senderAddr','senderPort','receiver','receiverAddr','receiverPort

# Method that queries records from the ExtraHop system
def recordQuery(query):
    url = HOST + '/api/v1/records/search'
    headers = {'Authorization': 'ExtraHop apikey=%s' % API_KEY}
    r = requests.post(url, headers=headers, data=json.dumps(query))
    try:
        return json.loads(r.text)
    except:
        print('Record query failed')
        print(r.text)
        print(r.status_code)

# Query records from appliance
response = recordQuery(QUERY)
total = response['total']
records = response['records']
offset = LIMIT
print('Retrieved ' + str(len(records)) + ' out of ' + str(total) + '
 records')
while total > offset:
    QUERY['offset'] = offset
    response = recordQuery(QUERY)
    new_records = response['records']
    records = records + new_records
    offset = offset + LIMIT
    print('Retrieved ' + str(len(records)) + ' out of ' + str(total) + '
 records')

# Simplify and format records for CSV
table = []
for record in records:
    row = {}
    fields = record['_source']
    for column in COLUMNS:
        try:
            value = fields[column]
            # Retrieve isSuspicious field from ex object
            if column == 'ex':
                try:
                    row['isSuspicious'] = value['isSuspicious']
                except:
                    row[column] = value
            # Concatenate values returned as lists
            elif type(value) is list:
                row[column] = ' '.join(value)
            # Retrieve values from dict objects
            elif type(value) is dict:
```

```
                try:
                    # If value is a list, concatenate list
                    if type(value['value']) is list:
                        row[column] = ' '.join(value['value'])
                    else:
                        row[column] = value['value']
                except:
                    row[column] = value
            else:
                row[column] = value
        except:
            row[column] = ''
    table.append(row)

# Write records to CSV file
if len(table) > 0:
    with open(FILENAME, 'wb') as csvfile:
        csvwriter = csv.writer(csvfile, encoding='utf-8')
        csvwriter.writerow(list(table[0].keys()))
        for row in table:
            csvwriter.writerow(list(row.values()))
```

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)
```