Add observations through the REST API

Published: 2025-02-06

Observations enable you to associate two or more IP addresses. For example, you can add an observation that tracks the activity of a VPN user by reading VPN logs and then associating the IP address of the VPN client on your network with the external IP address assigned to the user on the internet. This guide provides instructions for adding an observation through the ExtraHop REST API Explorer and through a Python script.

Before you begin

- Familiarize yourself with the ExtraHop REST API Guide to learn how to navigate the ExtraHop REST API Explorer.
- For sensors and the ExtraHop console, 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 RevealX 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.)

Add observations 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 sensor or console, followed by `/api/v1/explore/`. For example, if your hostname is seattle-eda, the URL is `https://seattle-eda/api/v1/explore/`.
2. Enter your REST API credentials.
   - For sensors and the ExtraHop console, click **Enter API Key** and then paste or type your API key into the **API Key** field.
   - For RevealX 360, click **Enter API Credentials** and then paste or type the ID and secret of your API credentials into the **ID** and **Secret** fields.
3. Click **Authorize** and then click **Close**.
4. Click **Observations** and then click **POST /observations/associatedipaddrs**.
5. Click **Try it out**.

   The JSON schema is automatically added to the body parameter text box.
6. In the body text box, specify the observations you want to add.

   For example, the following fields associate 10.8.0.0 with 108.162.0.0:

```
{
    "observations": [
        {
            "associated_ipaddr": "108.162.0.0",
            "ipaddr": "10.8.0.0",
            "timestamp": 1257935231
        }
    ],
    "source": "OpenVPN"
}
```

7. Click **Send Request**.

Retrieve and run the example Python script

The ExtraHop GitHub repository contains an example Python script that creates associations on the ExtraHop system based on a CSV log file from OpenVPN. You can configure the script to read other CSV files by modifying the `IPADDR`, `ASSOCIATED_IPADDR`, and `TIMESTAMP` variables, which specify the names of the CSV columns that the script reads.

1. Go to the ExtraHop code-examples GitHub repository and download the `add_observations/add_observations.py` file to your local machine.
2. In a text editor, open the `add_observations.py` file and replace the following configuration variables with information from your environment:

   • **HOST:** The IP address or hostname of the sensor.

   • **API_KEY:** The API key.

   • **CSV_FILE:** The name of the CSV log file.

   • **SOURCE:** The source of the observations.

   • **IPADDR:** The name of the column in the CSV file that specifies the IP addresses of the VPN clients on your internal network.

   • **ASSOCIATED_IPADDR:** The name of the column in the CSV file that specifies the external IP addresses assigned to the users on the public internet.

   • **TIMESTAMP:** The name of the column in the CSV file that specifies the time that the observation was created by the source. By default, the timestamp must be in the format: `Month/Day/Year Hour:Minute:Second`. However, you can change the format by modifying the `pattern` variable in the `translateTime()` function.

   Tip: If the log file distributes timestamp values over multiple columns, you can modify the `timestamp` field in the `readCSV()` function to concatenate the values. For example, assume that the first four columns of the CSV file are organized as shown in the following table:

01	01	01	10:10:10
Month	**Day**	**Year**	**Time**

   The following code reads those first four columns into the default `translateTime()` function:

   ```
   'timestamp': translateTime(row[0] + '/' + row[1] + '/' + row[2] +
    ' ' + row[3])
   ```

3. Run the following command:

   ```
   python3 add_observations.py
   ```

   Note: If the script returns an error message that the TLS 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 is not recommended. The following code sends an HTTP GET request without certificate verification:

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