

Configure ticket tracking for detections

Published: 2019-02-15

ExtraHop detections identify when unusual behavior is discovered on your network. To ensure that each of these events is properly investigated, you can automatically create tickets in a third party ticketing system for each detection. You can then update detection details in the ExtraHop Web UI with information from your ticket tracking software, enabling you to sort detections by ticket status or assignee.

Any third-party ticketing system that can accept Open Data Stream (ODS) requests, such as Jira or Salesforce, can be linked to ExtraHop detections.

Ticket tracking requires a few steps before your detections are linked to your ticketing system.


1. You must enable ticket tracking.
2. Optionally, you can configure a URL template for your ticketing system that creates a clickable link in the detection card that opens the ticket in your ticketing system in a new browser window.
3. You must write a trigger and configure an ODS target to send detection information to your ticketing system.
4. You must write a REST API script that sends information from your ticketing system to the detection in the ExtraHop Web UI.

Before you begin:

- You must have access to an ExtraHop Discover appliance with a user account that has unlimited privileges.
- You must be familiar with writing ExtraHop Triggers. For more information, see [Triggers](#) and the procedures in [Build a trigger](#).
- You must create an ODS target for your ticket tracking server. For more information, see the following topics about configuring ODS targets: [HTTP](#), [Kafka](#), [MongoDB](#), [syslog](#), or [raw data](#).
- You must have a valid API key to make changes through the REST API and complete the procedures below. (See [Generate an API key](#).)

Enable ticket tracking and specify a URL template

You must enable ticket tracking before your scripts can update ticket information on your ExtraHop system through the REST API. Optionally, specify a URL template that adds an HTML link for the ticket to your detection card. Click the link to open the linked ticket in your ticketing system in a new browser tab.

 **Note:** If you have a connected Command appliance, you must complete these steps on all connected Discover and Command appliances that are licensed for detections.

1. Log into the Admin UI on the Discover or Command appliance.
2. In the System Configuration section, click **Ticket Tracking**.
3. Select the **Enable ticket tracking** checkbox.
4. Optional: In the URL field, specify the URL template for your ticketing system and add the `$ticket_id` variable at the appropriate location. For example, type a complete URL such as `https://jira.example.com/browse/$ticket_id`. The `$ticket_id` variable is replaced with the ticket ID associated with the detection.

Today 14:00
lasting an hour

83
RISK

LATERAL MOVEMENT

Status: **CLOSED**

Ticket ID: ✓ EX-4437

Assignee: hopuser

Suspicious CIFS Client File Share Access on AccountingLaptop

This device sent an excessive number of read requests over the Common Internet File System (CIFS) protocol. This anomaly indicates that the device might be compromised and is preparing files for data exfiltration.

Server linked to this anomaly:

- corpshare.example.com (192.168.6.179)

AccountingLaptop Activity Map

| CIFS Metric | 6-hour Snapshot | Peak Value | Expected Range | Deviation |
|-------------|-----------------|------------|----------------|-----------|
| Reads | | 1.13 K | 0-1 | 112,500% |

Write a trigger to create and update tickets about detections on your ticketing system

While there are many ways to write a trigger that interacts with ticket tracking software, this example shows you how to create a trigger that performs the following actions:

- Create a new ticket in the ticketing system every time a new detection appears on the ExtraHop system.
- Assign new tickets to a user named `escalations_team` in the ticketing system.
- Run every time a detection is updated on the ExtraHop system.
- Send detection updates over an HTTP Open Data Stream (ODS) to the ticketing system.

The complete example script is available at the end of this topic.

1. Log into the Web UI on the ExtraHop Discover or Command appliance.
2. Click the System Settings icon and then click **Triggers**.
3. Click **New**.
4. Click the Configuration tab, and specify a name and optional description for the trigger.
5. From the Events list, select **DETECTION_UPDATE**.
6. The `DETECTION_UPDATE` event runs every time that a detection is created or updated in the ExtraHop system.
7. Click the Editor tab.
8. In the Editor tab, specify `Detection class` parameters in a JavaScript object. These parameters determine the information that is sent to your ticketing system.

The following example code adds the detection ID, description, title, categories, and risk score to a JavaScript object called `payload`:

```
const summary = "ExtraHop Detection: " + Detection.id + ": " +
  Detection.title;
const description = "ExtraHop has detected the following event on your
  network: " + Detection.description
const payload = {
  "fields": {
    "summary": summary,
    "assignee": {
      "name": "escalations_team"
    },
  },
  "reporter": {
```

```

        "name": "ExtraHop"
      },
      "priority": {
        "id": Detection.riskScore
      },
      "labels": Detection.categories,
      "description": description
    }
  };

```

- Next, define the HTTP request parameters in a JavaScript object below the previous JavaScript object.

For example, the following code defines an HTTP request for the payload described in the previous example: defines a request with a JSON payload:

```

const req = {
  'path': '/rest/api/issue',
  'headers': {
    'Content-Type': 'application/json'
  },
  'payload': payload
};

```

For more information about ODS request objects, see [Open data stream classes](#).

- Finally, specify the HTTP POST request that sends the information to the ODS target. For example, the following code sends the HTTP request described in the previous example to an ODS target named ticket-server:

```

Remote.HTTP('ticket-server').post(req);

```

The complete trigger code should look similar to the following example:

```

const summary = "ExtraHop Detection: " + Detection.id + ": " +
  Detection.title;
const description = "ExtraHop has detected the following event on your
network: " + Detection.description
const payload = {
  "fields": {
    "summary": summary,
    "assignee": {
      "name": "escalations_team"
    },
    "reporter": {
      "name": "ExtraHop"
    },
    "priority": {
      "id": Detection.riskScore
    },
    "labels": Detection.categories,
    "description": description
  }
};

const req = {
  'path': '/rest/api/issue',
  'headers': {
    'Content-Type': 'application/json'
  },
  'payload': payload
};

```

```
Remote.HTTP('ticket-server').post(req);
```

Send ticket information to detections through the REST API

After you have configured a trigger to create tickets for detections in your ticket tracking system, you can update ticket information on your ExtraHop system through the REST API.

Ticket information appears in detections on the Detections page in the ExtraHop Web UI. For more information, see the [Detections](#) topic.

The following example Python script takes ticket information from a Python array and updates the associated detections on an ExtraHop system.

```
#!/usr/bin/env python2

import json
import requests
import csv

API_KEY = 'f6876657888a7c1f24ac77827'
HOST = 'https://example.extrahop.com/'

# Method that updates detections on an ExtraHop appliance
def updateDetection(detection):
    url = HOST + 'api/v1/detections/' + detection['detection_id']
    del detection['detection_id']
    data = json.dumps(detection)
    headers = {'Content-Type': 'application/json',
               'Accept': 'application/json',
               'Authorization': 'ExtraHop apikey=%s' % API_KEY}
    r = requests.patch(url, data=data, headers=headers)
    print r.status_code
    print r.text

# Array of detection information
detections = [
    {
        "detection_id": "1",
        "ticket_id": "TK-16982",
        "status": "new",
        "assignee": "sally",
        "resolution": None,
    },
    {
        "detection_id": "2",
        "ticket_id": "TK-2078",
        "status": None,
        "assignee": "jim",
        "resolution": None,
    },
    {
        "detection_id": "3",
        "ticket_id": "TK-3452",
        "status": None,
        "assignee": "alex",
        "resolution": None,
    }
]

for detection in detections:
    updateDetection(detection)
```



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)
```