

# Create a custom detection

Published: 2022-09-27

Custom detections enable you to specify criteria that generates detections on the ExtraHop system. Machine-learning and rules-based detections capture unusual behaviors and common threats, however by creating a custom detection, you can hone in on the devices and behaviors that are critical for your network.

When creating a custom detection, you must create a trigger that identifies the system event and conditions that the system should watch for, and then you can assign the trigger to the specific devices or device groups you want to monitor. When the event occurs, a detection is generated.

In this guide, we provide the steps and an example script that generates a custom detection when suspicious connections are made to specified websites through Windows PowerShell.

## Before you begin

- You must have some familiarity with ExtraHop [Triggers](#). In particular, consider [these best practices](#) when writing your script and assigning triggers.
- You must have a user account with the [privileges](#) required to create triggers.
- If you have a console, create a trigger on the console and the trigger will run on all connected sensors.

## Create a trigger to generate custom detections

Triggers generate custom detections by calling the `commitDetection` function in the trigger script.


In the following example, the trigger generates a custom detection when a PowerShell client reaches out to a website that is known as a staging site for exfiltrated data.

The trigger identifies PowerShell connections by looking for SSL client JA3 hashes that belong to known PowerShell clients.

If the SSL connection occurs from a PowerShell client to a suspicious host, the trigger generates a detection. The detection includes the version of PowerShell that initiated the connection, the server IP address, and the client IP address.



**Note:** For more information about the `commitDetection` function, see the [Trigger API Reference](#).

1. Click the System Settings icon  and then click **Triggers**.
2. Click **Create**.
3. Specify the following trigger configuration settings:

### Name

Type a name for your trigger. This name identifies your trigger, not the detection.

In our example, we will enter the name: `Custom Detection: PowerShell Connection to Suspicious Site`.

### Description

(Optional) Type the trigger description. This description is for the trigger, not the detection.

In our example, we will enter the description: `Creates a detection every time a PowerShell client connects to pastebin, raw.githubusercontent.com, or githubhack. PowerShell clients are identified by JA3 hashes.`

### Events

Select the event on which the trigger runs.

In our example, we will select the SSL\_OPEN event. This event occurs when an SSL connection is first established.

### Assignments

Select the device or device group that you want to monitor. Initially, assign your trigger to a single device for testing. After you have confirmed that the custom detection works properly, assign the trigger to a device group that contains all of the devices you want to monitor.

Because PowerShell is a Windows command-line tool, select a Microsoft server to test the trigger. After you confirm that the custom detection is working correctly, change the assignment to a device group that contains all your critical Microsoft servers. For more information about creating device groups, see [Create a device group](#).

- In the right pane, type the code that determines when your custom detection is generated. In our example, the following trigger code identifies when a client initiates connection to pastebin, githubusercontent, or githack:

```
if(SSL.host.match(/pastebin/i) || SSL.host.match(/raw.githubusercontent.com/i) || SSL.host.match(/githack/i)) {
}
```

- Next, type the code that commits your custom detection. The `commitDetection` function must be written in the following format:

```
commitDetection('<detection type ID>', {
  title: '<title>',
  description: '<detection description>',
  categories: ['<category>'],
  riskScore: <risk score>,
  participants: [{
    object:<offender participant>,
    role: 'offender'
  }, {
    object: <victim participant>,
    role: 'victim'
  }],
  identityKey: '<identity key>',
  identityTtl: '<time period>',
});
```

Enter values for each of the following parameters in your script.

Value	Description
detection type ID	A unique string that identifies your custom detection. This string can only contain letters, numbers, and underscores.
title	Text that appears at the top of the detection card. Type a descriptive title that is easy to scan.
detection description	Text that appears below the title and category on a detection card. Type information about the event that generates the detection.  This field supports markdown. We recommend that you include interpolation variables to display specific information about your detection.  For example, the variables <code>\$(Flow.client.ipaddr)</code> and

Value	Description
category	<p>A label by which you can filter similar detections. You can add multiple categories. For a full list of category values, see the <a href="#">Trigger API Reference</a>.</p>
risk score	<p>A number that measures the likelihood, complexity, and business impact of a security detection. The risk score icon appears at the top of the detection card and is color coded by severity as red (80-99), orange (31-79), or yellow (1-30). You can <a href="#">sort detections by risk</a>.</p>
offender participant victim participant	<p>An array of objects that identifies the participants in the detection. Define the role of the participant as either 'offender' or 'victim' and provide a reference to a device, IP address, or application object for that role.</p> <p>For example, the following array identifies the server as the offender and the client as the victim in a flow:</p> <pre data-bbox="878 999 1458 1224"> participants: [   { role: 'offender', object:     Flow.server.device},   { role: 'victim', object:     Flow.client.device } ] </pre> <p>For more information about device, IP address, and application objects, see the <a href="#">Trigger API Reference</a>.</p>
identity key	<p>A string that enables the identification of ongoing detections. If multiple detections with the same identity key and detection type are generated within the time period specified by the <code>identityTtl</code> parameter, the detections are consolidated into a single ongoing detection.</p> <p>Create a unique identity key string by combining characteristics of the detection.</p> <p>For example, the following identity key is created by combining the server IP address and the client IP address:</p> <pre data-bbox="878 1766 1458 1871"> identityKey: [Flow.server.ipaddr, Flow.client.ipaddr].join('!!!') </pre>

Value	Description
time period	<p>The amount of time after a detection is generated that duplicate detections are consolidated into an ongoing detection. The time period is reset, and the detection does not end until the time period expires.</p> <p>The following time periods are valid:</p> <ul style="list-style-type: none"> <li>• hour</li> <li>• day</li> <li>• week</li> </ul> <p>The default time period is <code>hour</code>.</p>

The following example shows the completed script section.

```
commitDetection('powershell_ja3', {
  title:
  'PowerShell / BitsAdmin Suspicious Connection',
  description:
  "This SSL client matched a variant of PowerShell." + "\n"+
  "Investigate other client behaviors on the victim host." + "\n"+
  "- ** PowerShell/BitsAdmin JA3 client match**" + "\n"+
  "- **Client IP:** " + Flow.client.ipaddr + "\n"+
  "- **JA3 Client Value:** " + ja3 + "\n"+
  "- **JA3 Client Match:** " + suspect_ja3_hashes[ja3],
  categories: ['sec.caution'],
    riskScore: 60,
    participants: [{
      object:Flow.client.device,
      role: 'offender'
    }],
    identityKey: [
Flow.server.ipaddr,
Flow.client.ipaddr,
hash
].join('!!!'),
    identityTtl: 'hour',
  });
```

These values appear in the detection card similar to the following figure:

6. Click **Save** and then click **Done**.  
See [Example Custom Detection Trigger](#) for a complete annotated script.

## Create a custom detection type

After you create a trigger to generate your custom detection, create the custom detection type to specify a display name and to add MITRE links, which enable your custom detection to appear in the matrix on the Group by MITRE Technique page.

1. Log in to the ExtraHop system through `https://<extrahop-hostname-or-IP-address>`.
2. Click the System Settings icon and then click **Detection Catalog**.
3. Click **Create**.
4. Complete the following fields:

### Display Name

Type a unique name that appears in place of the detection type ID.

### Detection Type ID

Type the value that you entered for the detection type ID in the trigger. For example, if you typed: `commitDetection('network_segmentation_breach')`, the detection type ID is "network\_segmentation\_breach".

### Author

Identify the author of the custom detection.

### MITRE Technique

From the drop-down list, select one or more MITRE techniques that you want to link to the detection.

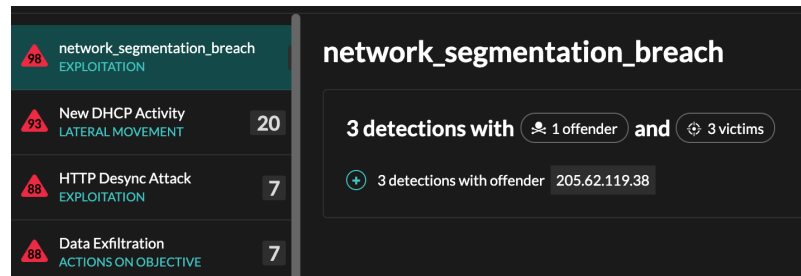
5. Click **Save**.

## View custom detections

When a custom detection is generated, you can view it on the Detections page.

From the top left of the page, select **Types**. Custom detections are grouped by the detection type ID or by the display name.

For example, if your detection type ID is `network_segmentation_breach`, it would appear in the detection list similar to the following figure:



From the top left of the page, select **Techniques**. The MITRE techniques that were linked to the custom detection are highlighted in the matrix.

### Next steps

Generate alerts for your custom detection by [configuring detection alert settings](#). For example, you can configure the ExtraHop system to email you when your custom detection occurs.

## Example Custom Detection Trigger

The following script is the complete PowerShell/JA3 example referred to throughout these instructions.

```
// If the server is internal, exit
if ( ! Flow.server.ipaddr.isExternal ) {
    return;
}
// If the SSL host name is not set, exit
if(SSL.host === null) { return; }

// Continue only if the SSL hostname belongs to one of the suspicious sites
if(SSL.host.match(/pastebin/i) || SSL.host.match(/raw.githubusercontent.com/i) || SSL.host.match(/githack/i)) {

    // List of common PowerShell JA3 hashes
    let suspect_ja3_hashes = cache('suspect_ja3_hashes', () => ({
        '13cc575f247730d3eeb8ff01e76b245f': 'PowerShell/BitsAdmin/PowerShell
4.0 Windows Server 2012RT',
        '5e12c14bda47ac941fc4e8e80d0e536f': 'PowerShell/BitsAdmin/PowerShell
4.0 Windows Server 2012RT',
        '2c14bfb3f8a2067fbc88d8345e9f97f3': 'PowerShell/BitsAdmin Windows
Server 2012RT',
        '613e01474d42ebe48ef52dff6a20f079': 'PowerShell/BitsAdmin Windows
Server 2012RT',
        '05af1f5calb87cc9cc9b25185115607d': 'BitsAdmin/PowerShell 5.0 Windows
7 64 bit enterprise',
        '8c4a22651d328568ec66382a84fc505f': 'BitsAdmin/PowerShell 5.0 Windows
7 64 bit enterprise',
        '235a856727c14dba889ddee0a38dd2f2': 'BitsAdmin/PowerShell 5.1 Server
2016',
        '17b69de9188f4c205a00fe5ae9c1151f': 'BitsAdmin/PowerShell 5.1 Server
2016',
        'd0ec4b50a944b182fc10ff51f883ccf7': 'PowerShell/BitsAdmin (Microsoft
BITS/7.8) Server 2016',
```

```

'294b2f1dc22c6e6c3231d2fe311d504b': 'PowerShell/BitsAdmin (Microsoft
BITS/7.8) Server 2016',
'54328bd36c14bd82ddaa0c04b25ed9ad': 'BitsAdmin/PowerShell 5.1 Windows
10',
'fc54e0d16d9764783542f0146a98b300': 'BitsAdmin/PowerShell 5.1 Windows
10',
'2863b3a96f1b530bc4f5e52f66c79285': 'BitsAdmin/PowerShell 6.0 Windows
Server 2012RT',
'40177d2da2d0f3a9014e7c83bdeee15a': 'BitsAdmin/PowerShell 6.0 Windows
Server 2012RT',
'36f7277af969a6947a61ae0b815907a1': 'PowerShell/BitsAdmin Windows 7
32 bit enterprise',
    }));
    // Store the client JA3 hash in a variable
    const hash = SSL.ja3Hash;

    // Iterate through each PowerShell JA3 hash
    for ( let ja3 in suspect_ja3_hashes ) {

        // If the client JA3 hash is from PowerShell,
        // commit the detection
        if ( hash.includes(ja3) ) {

            commitDetection('PowerShell_JA3', {
                categories: ['sec.caution'],
                title: "PowerShell / BitsAdmin Suspicious Connection",
                // Specify the offender as the device object of the client
                participants: [
                    { role: 'offender', object: Flow.client.device }
                ],
                description:
                    "This SSL client matched a variant of PowerShell." +
                    "\n" +
                    "Investigate other client behaviors on the victim host."
                    + "\n" +
                    "- ** PowerShell/BitsAdmin JA3 client match**" + "\n" +
                    "- **Client IP:** " + Flow.client.ipaddr + "\n" +
                    "- **Server IP:** " + Flow.server.ipaddr + "\n" +
                    "- **JA3 Client Value:** " + ja3 + "\n" +
                    "- **JA3 Client Match:** " + suspect_ja3_hashes[ja3],
                // Create the identity key by combining the server IP
                address, client IP address, and PowerShell JA3 hash
                identityKey: [
                    Flow.server.ipaddr,
                    Flow.client.ipaddr,
                    hash
                ].join('!!'),
                riskScore: 60,
                identityTtl: 'hour'
            });
        }
    }
}

```