Importieren Sie externe Daten in Ihr ExtraHop-System

Veröffentlicht: 2025-02-04

Die ExtraHop Open Data Context API ermöglicht es Ihnen, Daten von einem externen Host in die Sitzungstabelle auf Ihrem ExtraHop zu importieren. Sensor. Auf diese Daten kann dann zugegriffen werden, um benutzerdefinierte Messwerte zu erstellen , die Sie zu ExtraHop-Diagrammen hinzufügen, in Datensätzen in einem Recordstore speichern oder in ein externes Analysetool exportieren können.

Nachdem Sie die Open Data Context API auf Ihrem aktiviert haben Sensor, können Sie Daten importieren, indem Sie ein Python-Skript von einem Memcache-Client auf einem externen Host ausführen. Diese externen Daten werden in Schlüssel-Wert-Paaren gespeichert und können durch Schreiben eines Auslöser abgerufen werden.

Sie könnten beispielsweise ein Memcached-Client-Skript auf einem externen Host ausführen, um CPU-Lastdaten in die Sitzungstabelle auf Ihrem Sensor. Anschließend können Sie einen Auslöser schreiben, der auf die Sitzungstabelle zugreift und die Daten als benutzerdefinierte Metriken festschreibt.

Warnung: Die Verbindung zwischen dem externen Host und dem ExtraHop-System ist nicht verschlüsselt und sollte keine vertraulichen Informationen übertragen.

Aktivieren Sie die Open Data Context API

Sie müssen die Open Data Context API auf Ihrem aktivieren Sensor bevor es Daten von einem externen Host empfangen kann.

Bevor Sie beginnen

- Sie müssen eingerichtet haben oder System- und Zugriffsadministrationsrechte um auf die Administrationsseite Ihres ExtraHop-Systems zuzugreifen.
- Wenn Sie über eine Firewall verfügen, müssen Ihre Firewallregeln externen Hosts den Zugriff auf die angegebenen TCP- und UDP-Ports ermöglichen. Die Standard-Portnummer ist 11211.

1. Loggen Sie sich in die Administrationseinstellungen des ExtraHop-Systems ein über `https://<extrahop-hostname-or-IP-address>/admin`.
2. In der Konfiguration des Systems Abschnitt, klicken **Erfassen**.
3. klicken **Datenkontext-API öffnen**.
4. klicken **Open Data Context API aktivieren**.
5. Konfigurieren Sie jedes Protokoll, über das Sie externe Datenübertragungen zulassen möchten:

Option	Description
TCP	1. Wählen Sie die **TCP-Port aktiviert** Ankreuzfeld. 2. In der **TCP-Anschluss** In diesem Feld geben Sie die Portnummer ein, die externe Daten empfangen soll.
UDP	1. Wählen Sie die **UDP-Port aktiviert** Ankreuzfeld. 2. In der **UDP-Anschluss** In diesem Feld geben Sie die Portnummer ein, die externe Daten empfangen soll.

6. klicken **Speichern Sie die Aufnahme und starten Sie sie neu**.

Wichtig: Der Sensor erfasst keine Messwerte, während er neu gestartet wird.

7. Klicken Sie **Erledigt**.

Schreiben Sie ein Python-Skript, um externe Daten zu importieren

Bevor Sie externe Daten in die Sitzungstabelle auf Ihrem importieren können Sensor, Sie müssen ein Python-Skript schreiben, das Ihre identifiziert Sensor und enthält die Daten, die Sie in die Sitzungstabelle importieren möchten. Das Skript wird dann von einem Memcached-Client auf dem externen Host ausgeführt.

Dieses Thema enthält Hinweise zur Syntax und bewährte Methoden für das Schreiben des Python-Skripts. EIN vollständiges Skriptbeispiel ist am Ende dieses Handbuchs verfügbar.

Bevor Sie beginnen

Stellen Sie sicher, dass Sie einen Memcached-Client auf dem externen Host-Computer haben. Sie können jede Standard-Memcached-Clientbibliothek installieren, z. B. http://libmemcached.org/ oder https://pypi.python.org/pypi/pymemcache. Der Sensor fungiert als Memcached-Server der Version 1.4.

Hier sind einige wichtige Überlegungen zur Open Data Context API:

- Die Open Data Context API unterstützt die meisten Memcached-Befehle wie `get`, `set`, und `increment`.
- Alle Daten müssen als Zeichenketten eingefügt werden, die lesbar sind für Sensor. Einige Memcached-Clients versuchen, Typinformationen in den Werten zu speichern. Beispielsweise speichert die Python-Memcache-Bibliothek Floats als gebeizte Werte, die beim Aufrufen zu ungültigen Ergebnissen führen `Session.lookup` in Auslösern. Die folgende Python-Syntax fügt einen Float korrekt als Zeichenfolge ein:

```
mc.set("my_float", str(1.5))
```

- Obwohl die Größe von Sitzungstabellenwerten nahezu unbegrenzt sein kann, kann das Festschreiben großer Werte in die Sitzungstabelle zu Leistungseinbußen führen. Darüber hinaus müssen in den Datenspeicher übertragene Metriken 4096 Byte oder weniger groß sein, und übergroße Tabellenwerte können zu verkürzten oder ungenauen Metriken führen.
- Einfache Statistikberichte werden unterstützt, detaillierte Statistikberichte nach Elementgröße oder Schlüsselpräfix werden jedoch nicht unterstützt.
- Das Festlegen des Ablaufs von Artikeln beim Hinzufügen oder Aktualisieren von Artikeln wird unterstützt, aber das Massenablaufen erfolgt über `flush` Befehl wird nicht unterstützt.
- Schlüssel laufen in Intervallen von 30 Sekunden ab. Wenn ein Schlüssel beispielsweise so eingestellt ist, dass er in 50 Sekunden abläuft, kann es zwischen 50 und 79 Sekunden dauern, bis er abläuft.
- Alle mit der Open Data Context API festgelegten Schlüssel werden über die verfügbar gemacht `SESSION_EXPIRE` lösen Ereignis aus, wenn sie ablaufen. Dieses Verhalten steht im Gegensatz zur Trigger-API, die ablaufende Schlüssel nicht über die `SESSION_EXPIRE` Ereignis.

1. Öffnen Sie in einem Python-Editor eine neue Datei.
2. Fügen Sie die IP-Adresse Ihres Sensor und die Portnummer, an die der Memcached-Client Daten sendet, ähnlich der folgenden Syntax:

```
client = memcache.Client(["eda_ip_address:eda_port"])
```

3. Fügen Sie die Daten, die Sie importieren möchten, über das Memcached zur Sitzungstabelle hinzu `set` Befehl, formatiert in Schlüssel-Wert-Paaren, ähnlich der folgenden Syntax:

```
client.set("some_key", "some_value")
```

4. Speichern Sie die Datei.

5. Führen Sie das Python-Skript vom Memcached-Client auf dem externen Host aus.

Schreiben Sie einen Auslöser für den Zugriff auf importierte Daten

Sie müssen einen Auslöser schreiben, bevor Sie auf die Daten in der Sitzungstabelle zugreifen können.

Bevor Sie beginnen
In diesem Thema wird Erfahrung mit dem Schreiben von Triggern vorausgesetzt. Wenn Sie mit Triggern nicht vertraut sind, schauen Sie sich die folgenden Themen an:

- Trigger
- Einen Auslöser erstellen
- Erfahren Sie, wie Sie einen Auslöser zum Sammeln benutzerdefinierter Metriken erstellen

1. Loggen Sie sich in das ExtraHop-System ein über `https://<extrahop-hostname-or-IP-address>`.
2. Klicken Sie auf das Symbol Systemeinstellungen und dann klicken **Auslöser**.
3. klicken **Neu**, und klicken Sie dann auf Konfiguration Registerkarte.
4. In der **Name** Feld, geben Sie einen eindeutigen Namen für den Auslöser ein.
5. In der **Ereignisse** Feld, beginnen Sie mit der Eingabe eines Veranstaltungsnamens und wählen Sie dann ein Ereignis aus der gefilterten Liste aus.
6. Klicken Sie auf **Herausgeber** Registerkarte.
7. In der Trigger-Skript Textfeld, schreiben Sie ein Triggerskript, das auf die Daten der Sitzungstabelle zugreift und diese anwendet. EIN vollständiges Skriptbeispiel ist am Ende dieses Handbuchs verfügbar.

   Das Skript muss das enthalten `Session.lookup` Methode, um einen bestimmten Schlüssel in der Sitzungstabelle zu finden und den entsprechenden Wert zurückzugeben.

   Der folgende Code sucht beispielsweise nach einem bestimmten Schlüssel in der Sitzungstabelle, um den entsprechenden Wert zurückzugeben, und überträgt den Wert dann als benutzerdefinierte Metrik an eine Anwendung:

   ```
   var key_lookup = Session.lookup("some_key");
                    Application("My
    App").metricAddDataset("my_custom_metric",
                    key_lookup);
   ```

   Hinweis: Sie können Schlüssel-Wert-Paare in der Sitzungstabelle auch mithilfe der Methoden hinzufügen, ändern oder löschen, die in der Session Klasse der ExtraHop Trigger API-Referenz .
8. klicken **Speichern und schließen**.

Nächste Schritte
Sie müssen den Auslöser einem Gerät oder einer Gerätegruppe zuweisen. Der Auslöser wird erst ausgeführt, wenn er zugewiesen wurde.

Beispiel für eine Open Data Context API

In diesem Beispiel erfahren Sie, wie Sie den Reputationswert und das potenzielle Risiko von Domains überprüfen, die mit Geräten in Ihrem Netzwerk kommunizieren. Zunächst zeigt Ihnen das Python-Beispielskript, wie Sie Domain-Reputationsdaten in die Sitzungstabelle auf Ihrem importieren Sensor. Anschließend zeigt Ihnen das Beispiel-Triggerskript, wie Sie IP-Adressen bei DNS-Ereignissen mit den importierten Domain-Reputationsdaten vergleichen und wie Sie aus den Ergebnissen eine benutzerdefinierte Metrik erstellen.

Beispiel für ein Python-Skript

Dieses Python-Skript enthält eine Liste von 20 beliebten Domainnamen und kann auf Domain-Reputationswerte verweisen, die aus einer Quelle wie Domain-Tools.

Dieses Skript ist eine REST-API, die eine POST-Operation akzeptiert, bei der der Hauptteil der Domänenname ist. Bei einem POST-Vorgang aktualisiert der Memcached-Client die Sitzungstabelle mit den Domäneninformationen.

```
#!/usr/bin/python
import flask
import flask_restful
import memcache
import sqlite3

top20 = { "google.com", "facebook.com", "youtube.com", "twitter.com",
          "microsoft.com", "wikipedia.org", "linkedin.com",
          "apple.com","adobe.com", "wordpress.org", "instagram.com",
          "wordpress.com", "vimeo.com", "blogspot.com", "youtu.be",
          "pinterest.com", "yahoo.com", "goo.gl", "amazon.com", "bit.ly}

dnsnames = {}

mc = memcache.Client(['10.0.0.115:11211'])

for dnsname in top20:
    dnsnames[dnsname] = 0.0

dbc = sqlite3.Connection('./dnsreputation.db')
cur = dbc.cursor()
cur.execute('select dnsname, score from dnsreputation;')
for row in cur:
    dnsnames[row[0]] = row[1]
dbc.close()

app = flask.Flask(__name__)
api = flask_restful.Api(app)

class DnsReputation(flask_restful.Resource):
    def post(self):
        dnsname = flask.request.get_data()
        #print dnsname
        mc.set(dnsname, str(dnsnames.get(dnsname, 50.0)), 120)
        return 'added to session table'

api.add_resource(DnsReputation, '/dnsreputation')

if __name__ == '__main__':
     app.run(debug=True,host='0.0.0.0')
```

Beispiel für ein Trigger-Skript

Dieses Beispiel-Triggerskript kanonisiert (oder konvertiert) IP-Adressen, die bei DNS-Ereignissen zurückgegeben werden, in Domänennamen und sucht dann in der Sitzungstabelle nach der Domain und ihrem Reputationswert. Wenn der Punktewert größer als 75 ist, fügt der Auslöser die Domain einem Anwendungscontainer mit dem Namen „DNSReputation“ als Detail-Metrik namens „Bad DNS reputation“ hinzu.

```
//Configure the following trigger settings:
//Name: DNSReputation
//Debugging: Enabled
//Events: DNS_REQUEST, DNS_RESPONSE
```

```
if (DNS.errorNum != 0 || DNS.qname == null
    || DNS.qname.endsWith("in-addr.arpa") || DNS.qname.endsWith("local")
    || DNS.qname.indexOf('.') == -1 ) {
    // error or null or reverse lookup, or lookup of local namereturn
    return;
}

//var canonicalname = DNS.qname.split('.').slice(-2).join('.');
var canonicalname = DNS.qname.substring(DNS.qname.lastIndexOf('.',
 DNS.qname.lastIndexOf('.')-1)+1)

//debug(canonicalname);


//Look for this DNS name in the session table
var score = Session.lookup(canonicalname)
if (score === null) {
    // Send to the service for lookup
    Remote.HTTP("dnsrep").post({path: "/dnsreputation", payload:
 canonicalname});
} else {
    debug(canonicalname + ':' +score);
    if (parseFloat(score) > 75) {
  //Create an application in the ExtraHop system and add custom metrics
        //Note: The application is not displayed in the ExtraHop system
 after the
  //initial request, but is displayed after subsequent requests.
        Application('DNSReputation').metricAddDetailCount('Bad DNS
 reputation', canonicalname + ':' + score, 1);
    }
}
```