Importez des données externes dans votre système ExtraHop

Publié: 2024-08-09

L'API Open Data Context d'ExtraHop vous permet d'importer des données d'un hôte externe dans le tableau de session de votre ExtraHop sonde. Ces données sont ensuite accessibles pour créer des métriques personnalisées que vous pouvez ajouter aux graphiques ExtraHop, stocker dans des enregistrements sur un espace de stockage des enregistrements ou exporter vers un outil d'analyse externe.

Après avoir activé l'API Open Data Context sur votre sonde, vous pouvez importer des données en exécutant un script Python à partir d'un client Memcached sur un hôte externe. Ces données externes sont stockées dans des paires clé-valeur et sont accessibles en écrivant un déclencheur.

Par exemple, vous pouvez exécuter un script client memcached sur un hôte externe pour importer les données de charge du processeur dans la table de session de votre sonde. Vous pouvez ensuite écrire un déclencheur qui accède à la table de session et valide les données sous forme de métriques personnalisées.

Avertissement: La connexion entre l'hôte externe et le système ExtraHop n'est pas cryptée et ne doit pas transmettre d'informations sensibles.

Activer l'API Open Data Context

Vous devez activer l'API Open Data Context sur votre sonde avant de pouvoir recevoir des données d'un hôte externe.

Avant de commencer

- Vous devez avoir configuré ou privilèges d'administration du système et des accès pour accéder à la page d'administration de votre système ExtraHop.
- Si vous disposez d'un pare-feu, vos règles de pare-feu doivent autoriser les hôtes externes à accéder aux ports TCP et UDP spécifiés. Le numéro de port par défaut est 11211.

1. Connectez-vous aux paramètres d'administration du système ExtraHop via `https://<extrahop-hostname-or-IP-address>/admin`.
2. Dans le Configuration du système section, cliquez sur **Capturez**.
3. Cliquez **API Open Data Context**.
4. Cliquez **Activer l'API Open Data Context**.
5. Configurez chaque protocole pour lequel vous souhaitez autoriser les transmissions de données externes via :

Option	Description
TCP	1. Sélectionnez le **Port TCP activé** case à cocher. 2. Dans le **Port TCP** dans ce champ, saisissez le numéro de port qui recevra les données externes.
UDP	1. Sélectionnez le **Port UDP activé** case à cocher. 2. Dans le **Port UDP** dans ce champ, saisissez le numéro de port qui recevra les données externes.

6. Cliquez **Enregistrez et redémarrez Capture**.

Important: La sonde ne collectera pas de métriques lors du redémarrage.

7. Cliquez **Terminé**.

Écrire un script Python pour importer des données externes

Avant de pouvoir importer des données externes dans le tableau des sessions de votre sonde, vous devez écrire un script Python qui identifie votre sonde et contient les données que vous souhaitez importer dans le tableau de session. Le script est ensuite exécuté à partir d'un client Memcached sur l'hôte externe .

Cette rubrique fournit des conseils sur la syntaxe et les meilleures pratiques pour écrire le script Python. UN exemple de script complet est disponible à la fin de ce guide.

Avant de commencer

Assurez-vous que vous disposez d'un client Memcached sur la machine hôte externe. Vous pouvez installer n'importe quelle bibliothèque client Memcached standard, telle que http://libmemcached.org/ ou https://pypi.python.org/pypi/pymemcache. La sonde agit comme un serveur Memcached version 1.4.

Voici quelques considérations importantes concernant l'API Open Data Context :

- L'API Open Data Context prend en charge la plupart des commandes memcached, telles que `get`, `set`, et `increment`.
- Toutes les données doivent être insérées sous forme de chaînes lisibles par sonde. Certains clients Memcached tentent de stocker des informations de type dans les valeurs. Par exemple, la bibliothèque de cache de Python stocke les flottants sous forme de valeurs sélectionnées, ce qui entraîne des résultats non valides lors de l'appel `Session.lookup` dans les déclencheurs. La syntaxe Python suivante insère correctement un float sous forme de chaîne :

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

- Bien que la taille des valeurs de la table de session puisse être presque illimitée, l'ajout de valeurs importantes à la table de session peut entraîner une dégradation des performances. En outre, les métriques enregistrées dans la banque de données doivent être de 4 096 octets ou moins, et les valeurs de table surdimensionnées peuvent entraîner des métriques tronquées ou imprécises.
- Les rapports statistiques de base sont pris en charge, mais les rapports statistiques détaillés par taille d'élément ou par préfixe clé ne sont pas pris en charge.
- La définition de l'expiration des éléments lors de l'ajout ou de la mise à jour d'éléments est prise en charge, mais l' expiration groupée via `flush` la commande n'est pas prise en charge.
- Les clés expirent toutes les 30 secondes. Par exemple, si une clé est configurée pour expirer dans 50 secondes, elle peut prendre de 50 à 79 secondes pour expirer.
- Toutes les clés définies avec l'API Open Data Context sont exposées via `SESSION_EXPIRE` événement déclencheur lorsqu'ils expirent. Ce comportement contraste avec l'API Trigger, qui n'expose pas les clés arrivant à expiration via le `SESSION_EXPIRE` événement.

1. Dans un éditeur Python, ouvrez un nouveau fichier.
2. Ajoutez l'adresse IP de votre sonde et le numéro de port vers lequel le client memcached enverra les données, selon la syntaxe suivante :

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

3. Ajoutez les données que vous souhaitez importer à la table de session via le memcached `set` commande, formatée en paires clé-valeur, similaire à la syntaxe suivante :

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

4. Enregistrez le fichier.
5. Exécutez le script Python depuis le client memcached sur l'hôte externe.

Écrire un déclencheur pour accéder aux données importées

Vous devez écrire un déclencheur avant de pouvoir accéder aux données de la table de session.

Avant de commencer
Cette rubrique suppose une expérience de l'écriture de déclencheurs. Si les déclencheurs ne vous sont pas familiers, consultez les rubriques suivantes :

- déclencheurs
- Créez un déclencheur
- Découvrez comment créer un déclencheur pour collecter des métriques personnalisées

1. Connectez-vous au système ExtraHop via `https://<extrahop-hostname-or-IP-address>`.
2. Cliquez sur l'icône des paramètres système puis cliquez sur **DÉCLENCHEURS**.
3. Cliquez **Nouveau**, puis cliquez sur Configuration onglet.
4. Dans le **Nom** champ, saisissez un nom unique pour le déclencheur.
5. Dans le **Évènements** champ, commencez à saisir le nom d'un événement, puis sélectionnez un événement dans la liste filtrée.
6. Cliquez sur **Rédacteur** onglet.
7. Dans le Script de déclenchement zone de texte, écrivez un script déclencheur qui accède aux données de la table de session et les applique. UN exemple de script complet est disponible à la fin de ce guide.

   Le script doit inclure `Session.lookup` méthode pour localiser une clé spécifiée dans la table de session et renvoyer la valeur correspondante.

   Par exemple, le code suivant recherche une clé spécifique dans la table de session pour renvoyer la valeur correspondante, puis valide la valeur dans une application sous forme de métrique personnalisée :

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

   Conseil Vous pouvez également ajouter, modifier ou supprimer des paires clé-valeur dans le tableau de session à l'aide des méthodes décrites dans Session classe du Référence de l'API ExtraHop Trigger.
8. Cliquez **Enregistrer et fermer**.

Prochaines étapes
Vous devez attribuer le déclencheur à un équipement ou à un groupe de dispositifs. Le déclencheur ne sera pas lancé tant qu'il n'aura pas été attribué.

Exemple d'API Open Data Context

Dans cet exemple, vous allez apprendre à vérifier le score de réputation et le risque potentiel des domaines qui communiquent avec les appareils de votre réseau. Tout d'abord, l'exemple de script Python vous montre comment importer des données de réputation de domaine dans la table de session de votre sonde. L'exemple de script déclencheur vous montre ensuite comment vérifier les adresses IP des événements DNS par rapport aux données de réputation de domaine importées et comment créer une métrique personnalisée à partir des résultats.

Exemple de script Python

Ce script Python contient une liste de 20 noms de domaine populaires et peut faire référence aux scores de réputation de domaine obtenus à partir d'une source telle que Outils de domaine.

Ce script est une API REST qui accepte une opération POST dont le corps est le nom de domaine. Lors d'une opération POST, le client memcached met à jour la table de session avec les informations de domaine .

```
#!/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')
```

Exemple de script de déclencheur

Cet exemple de script de déclencheur canonise (ou convertit) les adresses IP renvoyées lors d'événements DNS en noms de domaine, puis vérifie le domaine et son score de réputation dans le tableau de session. Si la valeur du score est supérieure à 75, le déclencheur ajoute le domaine à un conteneur d'application appelé « DNSReputation » sous la forme d'une métrique détaillée appelée « Mauvaise réputation DNS ».

```
//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);
    }
}
```