Importer des données externes dans votre système ExtraHop

Publié: 2023-09-19

L'API ExtraHop Open Data Context vous permet d'importer des données d'un hôte externe dans la table des sessions de votre capteur ExtraHop. Ces données peuvent ensuite être utilisées pour créer des mesures personnalisées que vous pouvez ajouter aux graphiques ExtraHop, stocker dans des enregistrements sur un magasin d'enregistrements ou exporter vers un outil d'analyse externe.

Après avoir activé l'API Open Data Context sur votre capteur, 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 par l'écriture d'un déclencheur.

Par exemple, vous pouvez exécuter un script client memcached sur un hôte externe pour importer des données de charge CPU dans la table de session de votre capteur. Ensuite, vous pouvez écrire un déclencheur qui accède à la table de session et commute les données en tant que 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 capteur avant qu'il ne puisse recevoir des données d'un hôte externe.

Avant de commencer

- Vous devez disposer de privilèges de configuration ou d'administration du système et des accès pour accéder à la page Administration de votre système ExtraHop.
- Si vous disposez d'un pare-feu, vos règles de pare-feu doivent permettre aux hôtes externes d'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 la section Configuration du système, cliquez sur **Capture**.
3. Cliquez sur **Open Data Context API**.
4. Cliquez sur **Activer Open Data Context API**.
5. Configurez chaque protocole par lequel vous souhaitez autoriser les transmissions de données externes :

Option	Description
TCP	1. Cochez la case **Port TCP activé**. 2. Dans le champ **Port TCP**, saisissez le numéro du port qui recevra les données externes.
UDP	1. Cochez la case **Port UDP activé**. 2. Dans le champ **Port UDP**, saisissez le numéro du port qui recevra les données externes.

6. Cliquez sur **Enregistrer et redémarrer la capture**.

   Important: Le capteur ne collectera pas de métriques pendant le redémarrage.

7. Cliquez sur **Terminé**.

Écrire un script Python pour importer des données externes

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

Cette rubrique fournit des conseils syntaxiques et des bonnes pratiques pour l'écriture du 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. Le capteur 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 le capteur. Certains clients memcached tentent de stocker des informations de type dans les valeurs. Par exemple, la bibliothèque memcache de Python stocke les flottants en tant que valeurs décapées, ce qui entraîne des résultats non valides lors de l'appel à `Session.lookup` dans les déclencheurs. La syntaxe Python suivante insère correctement un flottant sous la forme d'une chaîne

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

)
- Bien que les valeurs de la table de session puissent être d'une taille pratiquement illimitée, l'enregistrement de valeurs importantes dans la table de session peut entraîner une dégradation des performances. En outre, les mesures transmises au magasin de données doivent être inférieures ou égales à 4096 octets, et des valeurs de table surdimensionnées peuvent donner lieu à des mesures tronquées ou imprécises.
- Les rapports statistiques de base sont pris en charge, mais les rapports statistiques détaillés par taille d'élément ou préfixe de 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 en bloc par le biais de la commande `flush` n'est pas prise en charge.
- Les clés expirent toutes les 30 secondes. Par exemple, si une clé est définie pour expirer dans 50 secondes, elle peut prendre entre 50 et 79 secondes pour expirer.
- Toutes les clés définies à l'aide de l'API Open Data Context sont exposées par le biais de l'événement déclencheur `SESSION_EXPIRE` au fur et à mesure de leur expiration. Ce comportement est différent de celui de l'API Trigger, qui n'expose pas les clés arrivant à expiration par le biais de l'événement `SESSION_EXPIRE`.

1. Dans un éditeur Python, ouvrez un nouveau fichier.
2. Ajoutez l'adresse IP de votre capteur et le numéro de port où le client memcached enverra les données, en suivant la syntaxe suivante :

   ```
   client = memcache.Client(["eda_ip_address:eda_port"])
   ```
3. Ajoutez les données que vous souhaitez importer dans la table de session via la commande memcached `set`, formatées en paires clé-valeur, comme dans la syntaxe suivante :

   ```
   client.set("some_key", "some_value")
   ```
4. Enregistrez le fichier.
5. Exécutez le script Python à partir du 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 certaine expérience de l'écriture de déclencheurs. Si vous n'êtes pas familiarisé avec les déclencheurs, consultez les rubriques suivantes :

- Déclencheurs
- Créer un déclencheur
- Apprendre à créer un déclencheur pour collecter des mesures personnalisées

1. Connectez-vous au système ExtraHop via `https://<extrahop-hostname-or-IP-address>`.
2. Cliquez sur l'icône Paramètres système, puis sur **Déclencheurs**.
3. Cliquez sur **Nouveau**, puis sur l'onglet Configuration.
4. Dans le champ **Nom**, saisissez un nom unique pour le déclencheur.
5. Dans le champ **Evénements**, commencez à saisir un nom d'événement, puis sélectionnez un événement dans la liste filtrée.
6. Cliquez sur l'onglet **Editeur**.
7. Dans la zone de texte Script de déclenchement, écrivez un script de déclenchement 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 la méthode `Session.lookup` 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 enregistre la valeur dans une application en tant que 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 la table de session grâce aux méthodes décrites dans la classe Session du site Référence API ExtraHop Trigger.
8. Cliquez sur **Enregistrer et fermer**.

Prochaines étapes
Vous devez affecter le déclencheur à un appareil ou à un groupe d'appareils. Le déclencheur ne s'exécutera pas tant qu'il n'aura pas été affecté.

Exemple d'API Open Data Context

Dans cet exemple, vous apprendrez à vérifier le score de réputation et le risque potentiel des domaines qui communiquent avec les appareils de votre réseau. L'exemple de script Python vous montre tout d'abord comment importer des données de réputation de domaine dans la table de session de votre capteur. Ensuite, l'exemple de script de déclenchement vous montre comment vérifier les adresses IP sur les 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 référencer des scores de réputation de domaine obtenus à partir d'une source telle que DomainTools.

Ce script est une API REST qui accepte une opération POST où 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 sur le 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 'ajouté à la table des sessions'
 api.add_resource(DnsReputation, '/dnsreputation') if __name__ ==
 '__main__' :
     app.run(debug=True,host='0.0.0.0')
```

Exemple de script de déclenchement

Cet exemple de script déclencheur canonise (ou convertit) les adresses IP renvoyées par les événements DNS en noms de domaine, puis vérifie le domaine et son score de réputation dans la table des sessions. Si la valeur du score est supérieure à 75, le déclencheur ajoute le domaine à un conteneur d'application appelé "DNSReputation" en tant que métrique détaillée appelée "Mauvaise réputation DNS".

```
//Configurer les paramètres de déclenchement suivants : //Nom :
 DNSReputation //Debugging : Activé //Evénements : 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 )
 { // erreur ou null ou recherche inversée, ou recherche de nom local return
 return ; } //var canonicalname = DNS.qname.split('.').slice(-2).join('.') ;
 var canonicalname = DNS.qname.substring(DNS.qname.lastIndexOf('.',
 DNS.qname.lastIndexOf('.')-1)+1) //debug(canonicalname) ; /Recherche de ce
 nom DNS dans la table de session var score = Session.lookup(canonicalname)
 if (score === null) { // Envoyer au service pour la recherche
 Remote.HTTP("dnsrep").post({path : "/dnsreputation", payload :
 canonicalname}) ; } else { debug(canonicalname + ':' +score) ; if
 (parseFloat(score) > 75) { //Créer une application dans le système ExtraHop
 et ajouter des métriques personnalisées //Note : L'application n'est pas
 affichée dans le système ExtraHop après la //requête initiale, mais elle
 est affichée après les requêtes suivantes.
       Application('DNSReputation').metricAddDetailCount('Bad DNS
 reputation', canonicalname + ':' + score, 1) ; } }
```