Configurez l'AMP pour la caractéristique de flot d'événement de points finaux

Contenu

[Introduction](#)
[Conditions préalables](#)
[Conditions requises](#)
[Composants utilisés](#)
[Configurer](#)
[Diagramme du réseau](#)
[Configurations](#)
[Créez les qualifications API](#)
[Créez le flot d'événement](#)
[Vérifier](#)
[Dépanner](#)
[Codes d'état](#)

Introduction

Ce document décrit comment configurer et consommer la caractéristique de flot d'événement pour la protection avancée de malware (AMP) pour des points finaux.

Conditions préalables

Exigences

Cisco recommande que vous ayez la connaissance des thèmes suivants :

- AMP pour les points terminaux
- Connaissance de base de la programmation de python

Composants utilisés

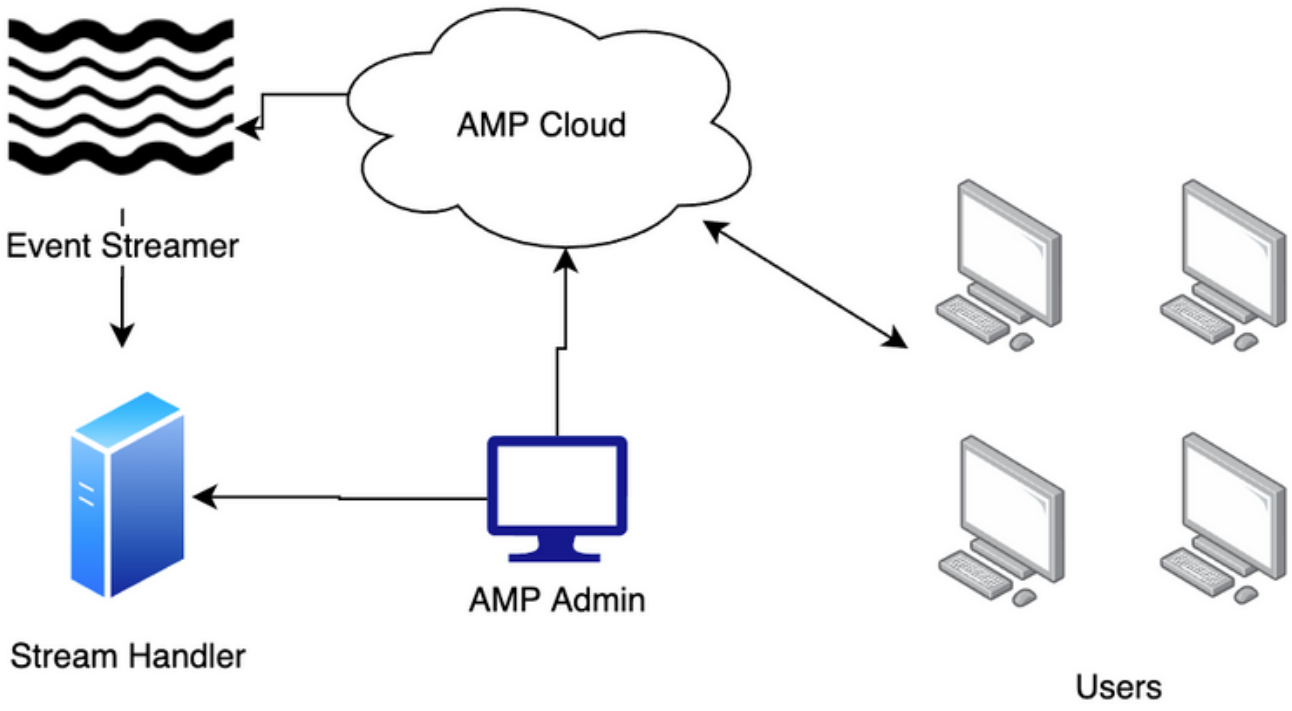
Les informations dans ce document sont basées sur le python 3.7 avec le **pika** (version 1.1.0) et les bibliothèques externes de **demandes** (version 2.22.0).

Les informations contenues dans ce document ont été créées à partir des périphériques d'un environnement de laboratoire spécifique. Tous les périphériques utilisés dans ce document ont démarré avec une configuration effacée (par défaut). Si votre réseau est en ligne, assurez-vous de bien comprendre l'incidence possible des commandes.

Configurer

Diagramme du réseau

Cette image fournit un exemple du séquençage de flot d'événement :



Configurations

Créez les qualifications API

1. Naviguez vers votre AMP pour le portail et la procédure de connexion de points finaux
2. Sous des **comptes,** choisissez les **qualifications API**
3. Cliquez sur New les **qualifications API**
4. Écrivez une valeur dans le domaine de **nom d'application**
5. Sélectionnez **lu et écrivez** pour la **portée**
6. Le clic **créent**
7. Enregistrez ces qualifications dans un gestionnaire ou un fichier crypté de mot de passe

Créez le flot d'événement

1. Ouvrez un shell de python et importez le **json**, le **SSL**, **pika** et **demande des** bibliothèques.

```
import json
import pika
import requests
import ssl
```

2. Enregistrez les valeurs pour l'URL, le client_id, et l'api_key. Votre URL peut varier si vous n'utilisez pas le nuage nord-américain. En outre, votre client_id et api_key est seul à votre environnement.

```
url = "https://api.amp.cisco.com/v1/event_streams"
client_id = "d16aff14860af496e848"
api_key = "d01ed435-b00d-4a4d-a299-1806ac117e72"
```

3. Créez l'élément de données pour passer à la demande. Ceci doit inclure le nom, et peut inclure l'event_type et le group_guid pour limiter les événements et les groupes inclus dans le flot. Si aucun group_guid ou event_type n'est passé, le flot d'événement inclura tous les groupes et types d'événement.

```
data = {
    "name": "Event Stream for ACME Inc",
    "group_guid": ["5cdf70dd-1b14-46a0-be90-e08da14172d8"],
    "event_type": [1090519054]
}
```

4. Faites le POST demander l'appel, et enregistrez la valeur dans une variable.

```
r = requests.post(
    url = url,
    data = data,
    auth = (client_id, api_key)
)
```

5. Imprimez code d'état. Confirmez que le code est 201.

```
print(r.status_code)
```

6. Chargez le contenu de la réponse dans un objet de json, et l'enregistrez qu'objet dans une variable.

```
j = json.loads(r.content)
```

7. Passez en revue le contenu des données de réponse.

```
for k, v in j.items():
    print(f"{k}: {v}")
```

8. Les données avancées de Protocol de file d'attente de messages (AMQP) sont à l'intérieur de la réponse. Extrayez les données dans des variables respectives.

```
user_name = j["data"]["amqp_credentials"]["user_name"]
queue_name = j["data"]["amqp_credentials"]["queue_name"]
password = j["data"]["amqp_credentials"]["password"]
host = j["data"]["amqp_credentials"]["host"]
port = j["data"]["amqp_credentials"]["port"]
proto = j["data"]["amqp_credentials"]["proto"]
```

9. Définissez une fonction de rappel avec ces paramètres. Dans cette installation, vous imprimez le corps de l'événement à l'écran. Cependant, vous pouvez changer ce contenu de cette fonction pour adapter à vos objectifs.

```
def callback(channel, method, properties, body):
```

```
    print(body)
```

10. Préparez l'URL AMQP des variables que vous avez créées.

```
amqp_url = f"amqps://{user_name}:{password}@{host}:{port}"
```

11. Préparez le contexte SSL

```
context = ssl.SSLContext(ssl.PROTOCOL_TLSv1_2)
amqp_ssl = pika.SSLOptions(context)
```

12. Préparez le flot AMQP avec les méthodes de bibliothèque de pika.

```
params = pika.URLParameters(amqp_url)
params.ssl_options = amqp_ssl

connection = pika.BlockingConnection(params)
channel = connection.channel()

channel.basic_consume(
    queue_name,
    callback,
    auto_ack = False
)
```

13. Initiez le flot.

```
channel.start_consuming()
```

14. Le flot est maintenant vivant et attendant des événements.

Vérifiez

Déclenchez un événement sur un point final dans votre environnement. Par exemple, initiez un balayage instantané. Notez que le flot imprime les données d'événement à l'écran.

Appuyez sur **Ctrl+C** (Windows) ou **commande-C** (MAC) d'interrompre le flot.

Dépanner

Codes d'état

- Code d'état de 401 indique qu'il y a une question avec l'autorisation. Vérifiez votre **client_id** et **api_key**, ou générez les nouvelles clés.
- Code d'état de 400 indique qu'il y a une mauvaise question de demande. Vérifiez que vous n'avez pas un flot d'événement créé avec ce nom, ou que vous n'avez pas plus de 5 flots d'événement créés.