Konfigurieren der Event Stream-Funktion für AMP für Endgeräte

Inhalt

Einführung

In diesem Dokument wird beschrieben, wie die Event Stream-Funktion für Advanced Malware Protection (AMP) für Endgeräte konfiguriert und genutzt wird.

Voraussetzungen

Anforderungen

Cisco empfiehlt, die folgenden Themen zu kennen:

- AMP für Endgeräte
- Grundkenntnisse der Python-Programmierung

Verwendete Komponenten

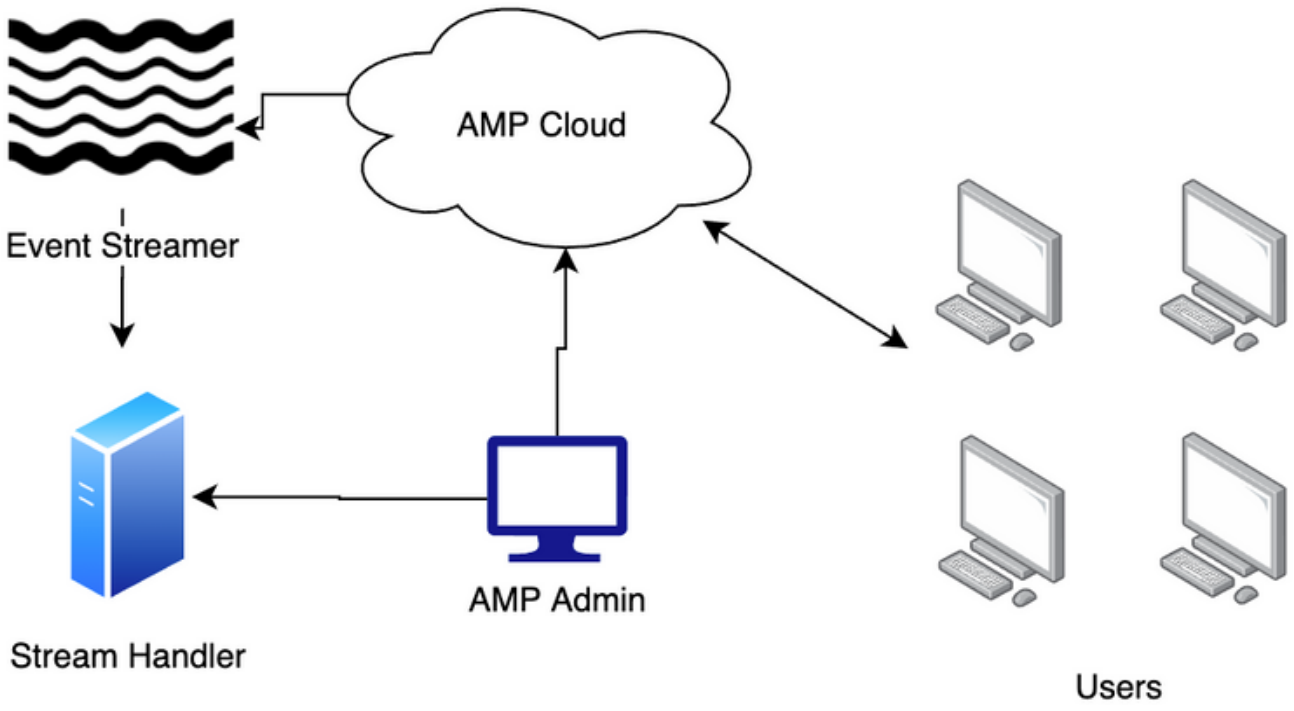
Die Informationen in diesem Dokument basieren auf Python 3.7 mit den **pika** (Version 1.1.0) und **Anfragen** (Version 2.22.0) externen Bibliotheken.

Die Informationen in diesem Dokument wurden von den Geräten in einer bestimmten Laborumgebung erstellt. Alle in diesem Dokument verwendeten Geräte haben mit einer leeren (Standard-)Konfiguration begonnen. Wenn Ihr Netzwerk in Betrieb ist, stellen Sie sicher, dass Sie die potenziellen Auswirkungen eines Befehls verstehen.

Konfigurieren

Netzwerkdiagramm

Dieses Bild enthält ein Beispiel für die Ereignisstream-Sequenzierung:



Konfigurationen

Erstellen von API-Anmeldeinformationen

1. Navigieren Sie zu Ihrem AMP für Endgeräte-Portal, und melden Sie sich an.
2. Wählen Sie unter **Accounts (Konten)** die Option **API Credentials (API-Anmeldeinformationen) aus.**
3. Klicken Sie auf **Neue API-Anmeldeinformationen.**
4. Geben Sie im Feld **Anwendungsname** einen Wert ein.
5. Wählen Sie **Lese- und Schreibzugriff** für den **Bereich aus.**
6. Klicken Sie auf **Erstellen**
7. Speichern dieser Anmeldeinformationen in einem Passwort-Manager oder in einer verschlüsselten Datei

Ereignisstream erstellen

1. Öffnen Sie eine Python-Shell, und importieren Sie die **json**, **ssl**, **pika** und **anfordernden** Bibliotheken.

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

2. Speichern Sie die Werte für url, client_id und api_key. Ihre URL kann variieren, wenn Sie nicht die North American Cloud verwenden. Außerdem sind Ihre client_id und api_key nur in Ihrer Umgebung verfügbar.

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

3. Erstellen Sie das Datenobjekt, das an die Anforderung übergeben werden soll. Dies muss den Namen enthalten und kann event_type und group_guid enthalten, um die im Stream enthaltenen Ereignisse und Gruppen einzuschränken. Wenn group_guid oder event_type nicht übergeben wird, enthält der Ereignisstream alle Gruppen und Ereignistypen.

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

4. Führen Sie den POST-Anforderungsaufruf aus, und speichern Sie den Wert in einer Variablen.

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

5. Drucken Sie den Statuscode. Bestätigen Sie, dass der Code 201 lautet.

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

6. Laden Sie den Inhalt der Antwort in ein json-Objekt, und speichern Sie dieses Objekt in einer Variablen.

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

7. Überprüfen Sie den Inhalt der Antwortdaten.

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

8. Die Daten des Advanced Message Queuing Protocol (AMQP) befinden sich in der Antwort. Extrahieren Sie die Daten in die entsprechenden Variablen.

```
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. Definieren Sie eine Rückruffunktion mit diesen Parametern. In dieser Konfiguration drucken

Sie den Text des Ereignisses auf den Bildschirm. Sie können diesen Inhalt dieser Funktion jedoch an Ihre Ziele anpassen.

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

10. Bereiten Sie die AMQP-URL aus den von Ihnen erstellten Variablen vor.

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

11. Bereiten Sie den SSL-Kontext vor

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

12. Bereiten Sie den AMQP-Stream mit den Methoden der Pika-Bibliothek vor.

```
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. Initiieren Sie den Stream.

```
channel.start_consuming()
```

14. Der Stream ist jetzt live und wartet auf Ereignisse.

Überprüfen

Auslösen eines Ereignisses auf einem Endpunkt in der Umgebung Starten Sie z. B. einen Flash-Scan. Beachten Sie, dass der Stream die Ereignisdaten an den Bildschirm ausgibt.

Drücken Sie **Strg+C** (Windows) oder **Command-C** (Mac), um den Stream zu unterbrechen.

Fehlerbehebung

Statuscodes

- Der Statuscode 401 weist auf ein Problem mit der Autorisierung hin. Überprüfen Sie Ihre **client_id** und **api_key**, oder generieren Sie neue Schlüssel.
- Ein Statuscode von 400 weist darauf hin, dass ein Problem mit der fehlerhaften Anforderung vorliegt. Stellen Sie sicher, dass kein Event Stream mit diesem Namen erstellt wurde oder dass nicht mehr als 5 Event Streams erstellt wurden. Eine weitere mögliche Abhilfe für

Statuscode 400 wäre das Hinzufügen der folgenden Variablen:

```
headers = {
    'content-type': 'application/json'
}
```

und aktualisieren Sie Ihre Postanfrage, um die Kopfzeilendeklaration wiederzugeben:

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