Configura funzionalità flusso eventi AMP for Endpoints

Sommario

[Introduzione](#)
[Prerequisiti](#)
[Requisiti](#)
[Componenti usati](#)
[Configurazione](#)
[Esempio di rete](#)
[Configurazioni](#)
[Crea credenziali API](#)
[Crea flusso eventi](#)
[Verifica](#)
[Risoluzione dei problemi](#)
[Codici di stato](#)

Introduzione

In questo documento viene descritto come configurare e utilizzare la funzione Event Stream per Advanced Malware Protection (AMP) for Endpoints.

Prerequisiti

Requisiti

Cisco raccomanda la conoscenza dei seguenti argomenti:

- AMP for Endpoints
- Conoscenze base della programmazione Python

Componenti usati

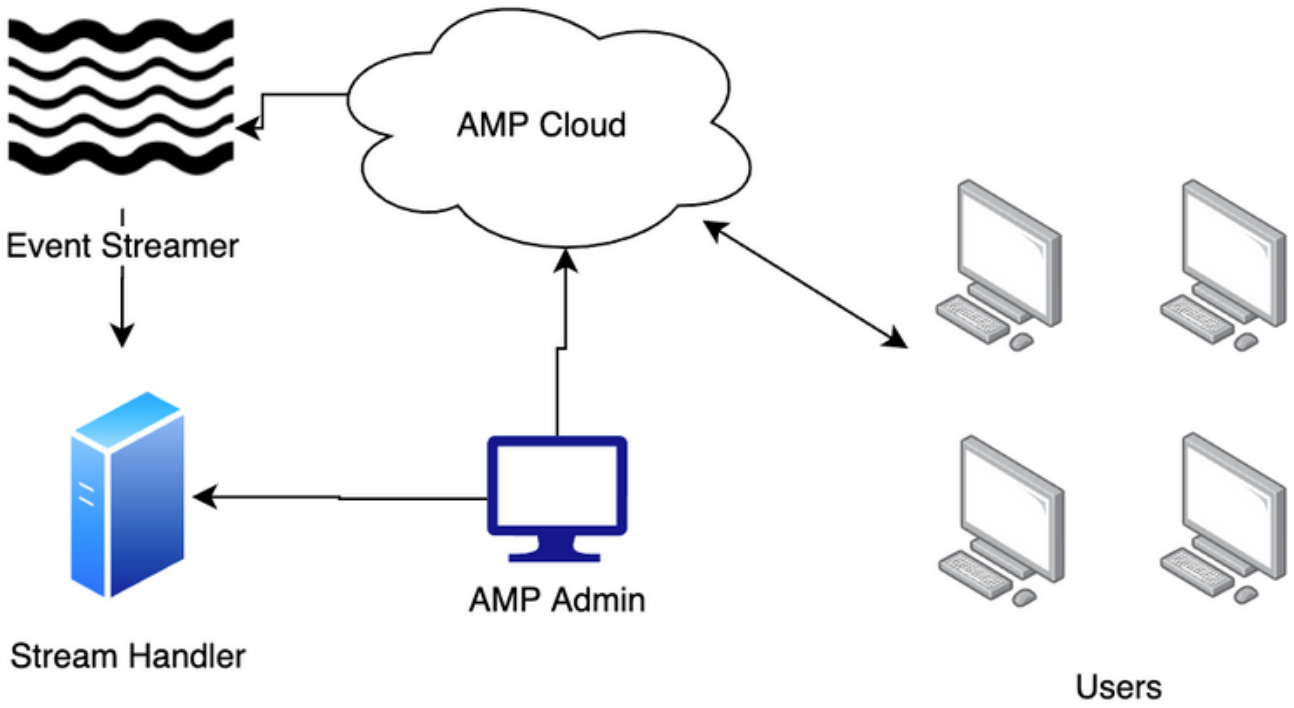
Le informazioni fornite in questo documento si basano su Python 3.7 con le librerie esterne **pika** (versione 1.1.0) e **request** (versione 2.2.0).

Le informazioni discusse in questo documento fanno riferimento a dispositivi usati in uno specifico ambiente di emulazione. Su tutti i dispositivi menzionati nel documento la configurazione è stata ripristinata ai valori predefiniti. Se la rete è operativa, valutare attentamente eventuali conseguenze derivanti dall'uso dei comandi.

Configurazione

Esempio di rete

Questa immagine fornisce un esempio di sequenziamento del flusso di eventi:



Configurazioni

Crea credenziali API

1. Accedere al portale AMP for Endpoints
2. In **Account,** scegliere **Credenziali API**
3. Fare clic su **Nuove credenziali API**
4. Immettere un valore nel campo **Nome applicazione**
5. Selezionare **Lettura e scrittura** per l'**ambito**
6. Fare clic su **Crea**
7. Memorizza queste credenziali in un gestore delle password o in un file crittografato

Crea flusso eventi

1. Aprire una shell Python e importare le librerie **json**, **ssl**, **pika** e **request**.

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

2. Memorizzare i valori per url, client_id e api_key. L'URL può variare se non si utilizza il cloud nordamericano. Inoltre, client_id e api_key sono univoci per l'ambiente in uso.

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

3. Creare l'oggetto dati da passare alla richiesta. Deve includere il nome e può includere event_type e group_guid per limitare gli eventi e i gruppi inclusi nel flusso. Se non viene passato alcun group_guid o event_type, il flusso di eventi includerà tutti i gruppi e i tipi di evento.

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

4. Eseguire la chiamata alla richiesta POST e memorizzare il valore in una variabile.

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

5. Stampa il codice di stato. Confermare che il codice sia 201.

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

6. Caricare il contenuto della risposta in un oggetto JSON e archiviare tale oggetto in una variabile.

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

7. Esaminare il contenuto dei dati di risposta.

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

8. I dati Advanced Message Queuing Protocol (AMQP) sono all'interno della risposta. Estrarre i dati nelle rispettive variabili.

```
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. Definire una funzione di richiamata con questi parametri. In questa configurazione, il corpo dell'evento viene stampato sullo schermo. È tuttavia possibile modificare il contenuto di questa funzione in base agli obiettivi desiderati.

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

10. Preparare l'URL AMQP dalle variabili create.

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

11. Preparare il contesto SSL

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

12. Preparare il flusso AMQP con i metodi della libreria 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. Avviare lo streaming.

```
channel.start_consuming()
```

14. Il flusso è ora in diretta e in attesa di eventi.

Verifica

Attivare un evento su un endpoint nell'ambiente. Ad esempio, avviare una scansione flash. I dati dell'evento vengono stampati sullo schermo.

Premete **Ctrl+C** (Windows) o **Comando-C** (Mac) per interrompere il flusso.

Risoluzione dei problemi

Codici di stato

- Il codice di stato 401 indica che si è verificato un problema con l'autorizzazione. Controllare **client_id** e **api_key** o generare nuove chiavi.
- Il codice di stato 400 indica che si è verificato un problema di richiesta non valida. Verificare che non sia stato creato un flusso di eventi con tale nome o che non siano stati creati più di 5 flussi di eventi. Un altro possibile rimedio per il codice di stato 400 consiste nell'aggiungere la seguente variabile:

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

e aggiornare la richiesta post per riflettere la dichiarazione di intestazione:

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