Advanced Malware Protection voor endpoints

Inhoud

Inleiding

Dit document beschrijft hoe u de optie Event Stream (AAMP) kunt configureren en gebruiken voor Advanced Malware Protection (AMP) voor endpoints.

Voorwaarden

Vereisten

Cisco raadt u aan om kennis te hebben van de volgende onderwerpen:

- Advanced Malware Protection voor endpoints
- Basiskennis van de programmering van Python

Gebruikte componenten

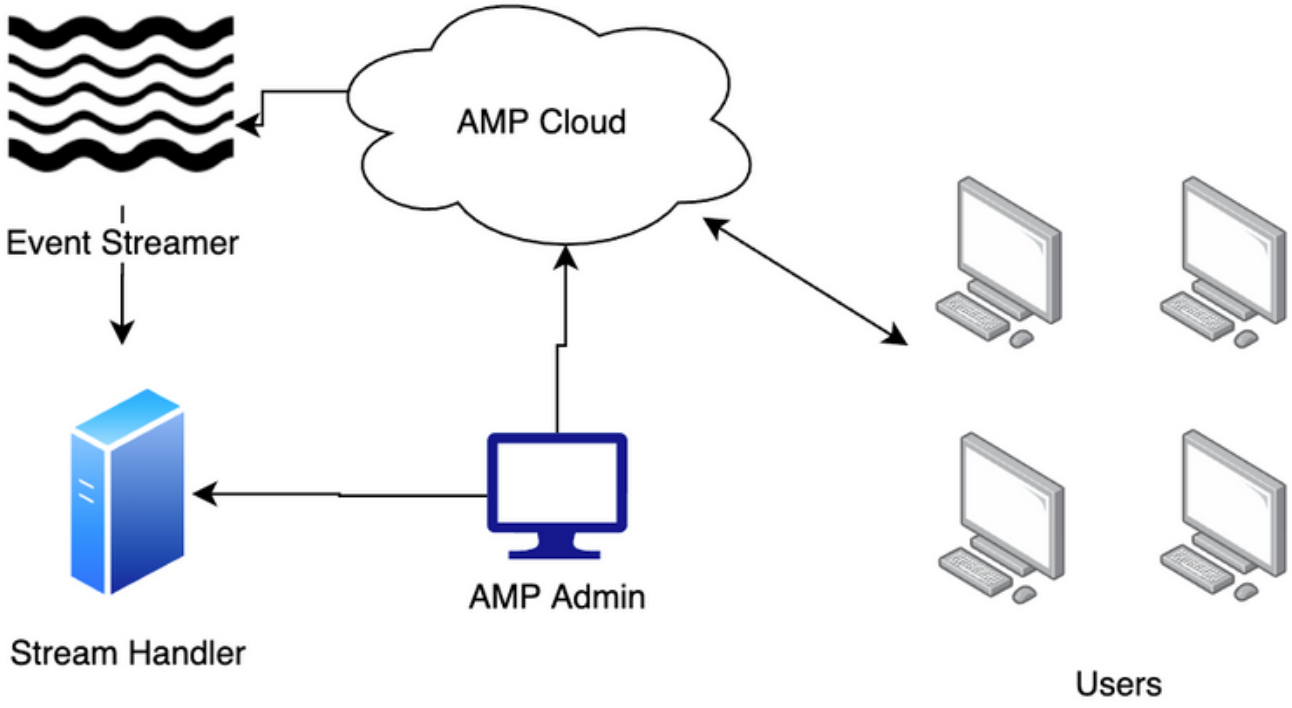
De informatie in dit document is gebaseerd op Python 3.7 met de **pika** (versie 1.1.0) en **verzoeken** (versie 2.2.2.0) externe bibliotheken.

De informatie in dit document is gebaseerd op de apparaten in een specifieke laboratoriumomgeving. Alle apparaten die in dit document worden beschreven, hadden een opgeschoonde (standaard)configuratie. Als uw netwerk levend is, zorg er dan voor dat u de mogelijke impact van om het even welke opdracht begrijpt.

Configureren

Netwerkdiagram

Dit beeld geeft een voorbeeld van de sequentie van Event Stream:



Configuraties

API-referenties maken

1. Navigeer naar uw Advanced Malware Protection voor endpoints en inloggen
2. Kies onder **Account API-Credentials**
3. Klik op **Nieuwe API-lettertypen**
4. Voer een waarde in in het veld **Toepassingsnaam**.
5. Selecteer **Lezen en schrijven** voor **toepassingsgebied**
6. Klik op **Maken**
7. Bewaar deze referenties in een wachtwoordbeheerder of een versleuteld bestand

Event Stream maken

1. Open een schildpad van Python en voer de **zoon**, **ssl**, **pika** en **verzoekt** bibliotheken in.

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

2. Bewaar de waarden voor de URL, client_id en api_key. Uw URL kan variëren als u niet de Noord-Amerikaanse wolk gebruikt. Uw client_id en api_key zijn ook uniek voor uw omgeving.

```
url = "https://api.amp.cisco.com/v1/event_streams"
client_id = "d16aff14860af496e848"
```

```
api_key = "d01ed435-b00d-4a4d-a299-1806ac117e72"
```

3. Maak het gegevensobject aan het verzoek door te geven. Dit moet naam bevatten, en kan event_type en group_guid omvatten om de gebeurtenissen en groepen in de stream te beperken. Als er geen group_guid of event_type wordt meegegeven, zal de eventstream alle groepen en eventtypen omvatten.

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

4. Maak de POST vraag, en bewaar de waarde in een variabele.

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

5. Druk de statuscode af. Bevestig dat de code 201 is.

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

6. Laad de inhoud van de reactie in een klein object en bewaar dat object in een variabele.

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

7. Bekijk de inhoud van de responsgegevens.

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

8. De Advanced Message Queuing Protocol (AMQP)-gegevens bevinden zich in de respons. Verdeel de gegevens in de respectievelijke variabelen.

```
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. Defineert een callback functie met deze parameters. In deze instelling drukt u de inhoud van de gebeurtenis op het scherm af. U kunt de inhoud van deze functie echter wijzigen om aan uw doelstellingen te voldoen.

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

10. Bereid de AMQP URL voor van de variabelen die u hebt gemaakt.

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

11. Maak de SSL-context klaar

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

12. Bereid de AMQP-stroom voor met de methoden van de poolbibliotheek.

```
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. Start de stream.

```
channel.start_consuming()
```

14. De stroom is nu live en wacht op gebeurtenissen.

Verifiëren

Probeer een gebeurtenis op een eindpunt in uw omgeving. Stel een flash-scan in. Merk op dat de stream de gebeurtenissen op het scherm afdrukt.

Druk op **Ctrl+C** (Windows) of **Opdracht-C** (Mac) om de stream te onderbreken.

Problemen oplossen

Statuscodes

- Een statuscode van 401 geeft aan dat er een probleem is met een vergunning. Controleer je **client_id** en **api_key**, of genereer nieuwe keys.
- Een statuscode van 400 geeft aan dat er een probleem is met een slecht verzoek. Controleer of je geen Event Stream hebt gemaakt met die naam, of dat je niet meer dan 5 Event Streams hebt gemaakt. Een andere mogelijke oplossing voor statuscode 400 zou zijn de volgende variabele toe te voegen:

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

en update uw postverzoek om de headerverklaring weer te geven:

```
r = requests.post(
    url = url,
    headers = headers,
```

```
    data = data,
    auth = (client_id, api_key)
)
```