AMP for Endpoints 이벤트 스트림 기능 구성

목차

소개

이 문서에서는 엔드포인트용 AMP(Advanced Malware Protection)를 위한 이벤트 스트림 기능을 구성하고 사용하는 방법에 대해 설명합니다.

사전 요구 사항

요구 사항

Cisco에서는 다음 주제에 대해 알고 있는 것이 좋습니다.

- AMP for Endpoints
- Python 프로그래밍에 대한 기본 지식

사용되는 구성 요소

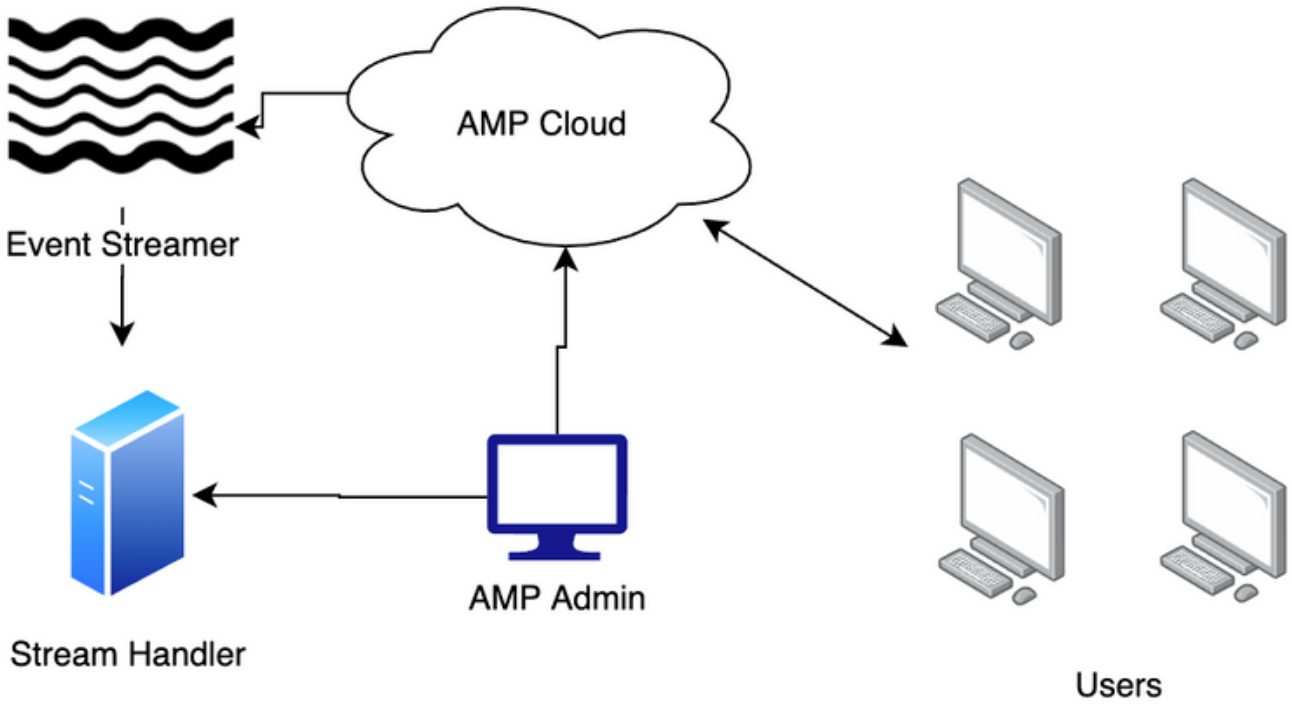
이 문서의 정보는 **pika**(버전 1.1.0) 및 **요청**(버전 2.22.0) 외부 라이브러리를 사용하는 Python 3.7을 기반으로 합니다.

이 문서의 정보는 특정 랩 환경의 디바이스를 토대로 작성되었습니다.이 문서에 사용된 모든 디바이스는 초기화된(기본) 컨피그레이션으로 시작되었습니다.네트워크가 작동 중인 경우 모든 명령의 잠재적인 영향을 이해해야 합니다.

구성

네트워크 다이어그램

이 이미지는 이벤트 스트림 시퀀싱의 예를 제공합니다.



구성

API 자격 증명 생성

1. AMP for Endpoints 포털로 이동하여 로그인합니다.
2. Accounts(**어카운트)에서 API Credentials(API 자격 증명)**를 선택합니다.
3. 새 **API 자격 증명을 클릭합니다.**
4. 애플리케이션 **이름** 필드에 값을 **입력합니다.**
5. 범위를 **읽기 및 쓰기**를 **선택합니다.**
6. Create(생성)를 **클릭합니다.**
7. 이러한 자격 증명을 암호 관리자 또는 암호화된 파일에 저장

이벤트 스트림 생성

1. Python 셸을 열고 **json**, **ssl**, **pika** 및 **요청** 라이브러리를 가져옵니다.

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

2. url, client_id 및 api_key 값을 저장합니다. 북미 클라우드를 사용하지 않는 경우 URL이 달라질 수 있습니다. 또한 client_id 및 api_key는 사용자 환경에 고유한 특성을 갖습니다.

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

3. 요청에 전달할 데이터 개체를 만듭니다. 여기에는 이름이 포함되어야 하며, 스트림에 포함된

이벤트와 그룹을 제한하려면 event_type 및 group_guid를 포함할 수 있습니다. group_guid 또는 event_type이 전달되지 않으면 이벤트 스트림에는 모든 그룹 및 이벤트 유형이 포함됩니다.

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

4. POST 요청 호출을 수행하고 값을 변수에 저장합니다.

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

5. 상태 코드를 인쇄합니다. 코드가 201인지 확인합니다.

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

6. 응답 내용을 json 개체로 로드하고 해당 개체를 변수에 저장합니다.

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

7. 응답 데이터의 내용을 검토합니다.

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

8. AMQP(Advanced Message Queuing Protocol) 데이터는 응답 내부에 있습니다. 데이터를 각 변수로 추출합니다.

```
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. 이러한 매개 변수를 사용하여 콜백 함수를 정의합니다. 이 설정에서는 이벤트의 본문을 화면에 인쇄합니다. 그러나 이 함수의 내용을 목표에 맞게 변경할 수 있습니다.

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

10. 생성한 변수에서 AMQP URL을 준비합니다.

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

11. SSL 컨텍스트 준비

```
context = ssl.SSLContext(ssl.PROTOCOL_TLSv1_2)
```

```
amqp_ssl = pika.SSLOptions(context)
```

12. pika 라이브러리 메서드를 사용하여 AMQP 스트림을 준비합니다.

```
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. 스트림을 시작합니다.

```
channel.start_consuming()
```

14. 스트림이 현재 라이브 상태로 이벤트를 기다리고 있습니다.

다음을 확인합니다.

환경의 엔드포인트에서 이벤트를 트리거합니다. 예를 들어 플래시 스캔을 시작합니다. 스트림이 이벤트 데이터를 화면에 출력합니다.

Ctrl **+C**(Windows) 또는 **Command-C**(Mac)를 눌러 스트림을 중단합니다.

문제 해결

상태 코드

- 상태 코드 401은 권한 부여에 문제가 있음을 나타냅니다. client_id 및 **api_key**를 확인하거나 새 키를 생성합니다.
- 상태 코드 400은 잘못된 요청 문제가 있음을 나타냅니다. 해당 이름으로 만든 이벤트 스트림이 없거나 생성된 이벤트 스트림이 5개 이상인지 확인합니다. 상태 코드 400에 대한 또 다른 가능한 해결 방법은 다음 변수를 추가하는 것입니다.

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

헤더 선언을 반영하도록 게시 요청을 업데이트합니다.

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