여러 엔드포인트에서 분리 시작/중지 자동화

목차

소개

이 문서에서는 Cisco Secure Endpoint용 API를 사용하여 여러 엔드포인트에서 중단/시작 격리를 자동화하는 방법에 대해 설명합니다.

사전 요구 사항

요구 사항

다음 주제에 대한 지식을 보유하고 있으면 유용합니다.

- Cisco Secure Endpoint
- Cisco Secure Endpoint 콘솔
- Cisco Secure Endpoint API
- 비단뱀

사용되는 구성 요소

이 문서의 정보는 다음 소프트웨어 버전을 기반으로 합니다.

- Cisco Secure Endpoint 8.4.0.30201
- Python 환경을 호스팅할 엔드포인트
- Python 3.11.7

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

배경 정보

- PUT 요청을 사용하여 격리를 시작합니다.
- 격리를 중지하는 데 DELETE 요청이 사용됩니다.
- 자세한 내용은 API 설명서를 참조하십시오.

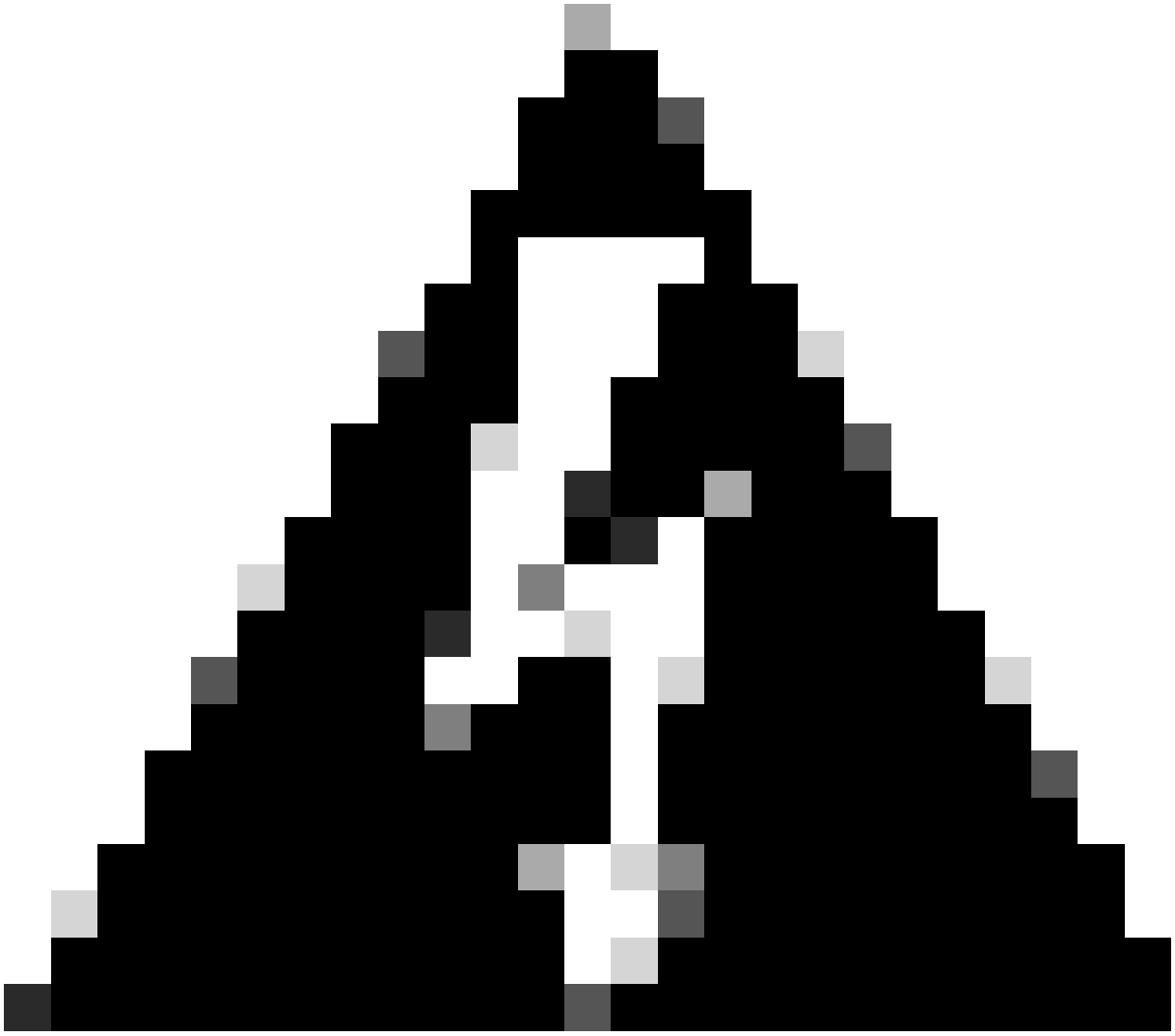
문제

Cisco Secure Endpoint는 한 번에 한 시스템에서 격리를 시작/중지할 수 있습니다. 그러나 보안 사고 중에는 잠재적 위협을 효과적으로 억제하기 위해 여러 엔드포인트에서 이러한 작업을 동시에 수행해야 하는 경우가 많습니다. API를 사용하여 대량 엔드포인트에 대한 시작/중지 격리 프로세스를 자동화하면 사고 대응 효율성을 크게 높이고 네트워크에 대한 전반적인 위험을 줄일 수 있습니다.

솔루션

- 이 문서에 제공된 Python 스크립트를 사용하여 보안 엔드포인트 API 자격 증명을 사용하여 조직의 여러 엔드포인트에서 격리를 시작/종료할 수 있습니다.
- AMP API 자격 증명을 생성하려면 Cisco AMP for Endpoints API 개요를 참조하십시오
- 제공된 스크립트를 사용하려면 엔드포인트에 pythonon을 설치해야 합니다.
- Python 설치 후 요청 모듈 설치

```
pip install requests
```



경고: 이 스크립트는 예시 목적으로만 제공되며 API를 사용하여 엔드포인트 격리 기능을 자동화하는 방법을 시연하기 위한 것입니다. CiscoTAC(Technical Assistance Center)는 이 스크립트와 관련된 문제 해결에 관여하지 않습니다. 사용자는 스크립트를 프로덕션 환경에 배포하기 전에 안전한 환경에서 신중하게 테스트해야 합니다.

스크립트

제공된 스크립트를 사용하여 비즈니스의 여러 엔드포인트에서 격리를 시작할 수 있습니다.

```
import requests

def read_config(file_path):
    """
    Reads the configuration file to get the API base URL, client ID, and API key.
    """
    config = {}
    try:
        with open(file_path, 'r') as file:
```

```
            for line in file:
                # Split each line into key and value based on '='
                key, value = line.strip().split('=')
                config[key] = value
    except FileNotFoundError:
        print(f"Error: Configuration file '{file_path}' not found.")
        exit(1)  # Exit the script if the file is not found
    except ValueError:
        print(f"Error: Configuration file '{file_path}' is incorrectly formatted.")
        exit(1)  # Exit the script if the file format is invalid
    return config

def read_guids(file_path):
    """
    Reads the file containing GUIDs for endpoints to be isolated.
    """
    try:
        with open(file_path, 'r') as file:
            # Read each line, strip whitespace, and ignore empty lines
            return [line.strip() for line in file if line.strip()]
    except FileNotFoundError:
        print(f"Error: GUIDs file '{file_path}' not found.")
        exit(1)  # Exit the script if the file is not found
    except Exception as e:
        print(f"Error: An unexpected error occurred while reading the GUIDs file: {e}")
        exit(1)  # Exit the script if an unexpected error occurs

def isolate_endpoint(base_url, client_id, api_key, connector_guid):
    """
    Sends a PUT request to isolate an endpoint identified by the connector GUID.
    Args:
        base_url (str): The base URL for the API.
        client_id (str): The API client ID for authentication.
        api_key (str): The API key for authentication.
        connector_guid (str): The GUID of the connector to be isolated.
    """
    url = f"{base_url}/{connector_guid}/isolation"
    try:
        # Send PUT request with authentication
        response = requests.put(url, auth=(client_id, api_key))
        response.raise_for_status()  # Raise an HTTPError for bad responses (4xx and 5xx)

        if response.status_code == 200:
            print(f"Successfully isolated endpoint: {connector_guid}")
        else:
            print(f"Failed to isolate endpoint: {connector_guid}. Status Code: {response.status_code},
    except requests.RequestException as e:
        print(f"Error: An error occurred while isolating the endpoint '{connector_guid}': {e}")

if __name__ == "__main__":
    # Read configuration values from the config file
    config = read_config('config.txt')

    # Read list of GUIDs from the GUIDs file
    connector_guids = read_guids('guids.txt')

    # Extract configuration values
    base_url = config.get('BASE_URL')
    api_client_id = config.get('API_CLIENT_ID')
    api_key = config.get('API_KEY')

    # Check if all required configuration values are present
```

```
    if not base_url or not api_client_id or not api_key:
        print("Error: Missing required configuration values.")
        exit(1)  # Exit the script if any configuration values are missing

    # Process each GUID by isolating the endpoint
    for guid in connector_guids:
        isolate_endpoint(base_url, api_client_id, api_key, guid)
```

지침

- AMP API 자격 증명을 생성하려면 Cisco AMP for Endpoints API 개요를 참조하십시오
- 해당 지역에 대해 언급된 BASE_URL 사용:

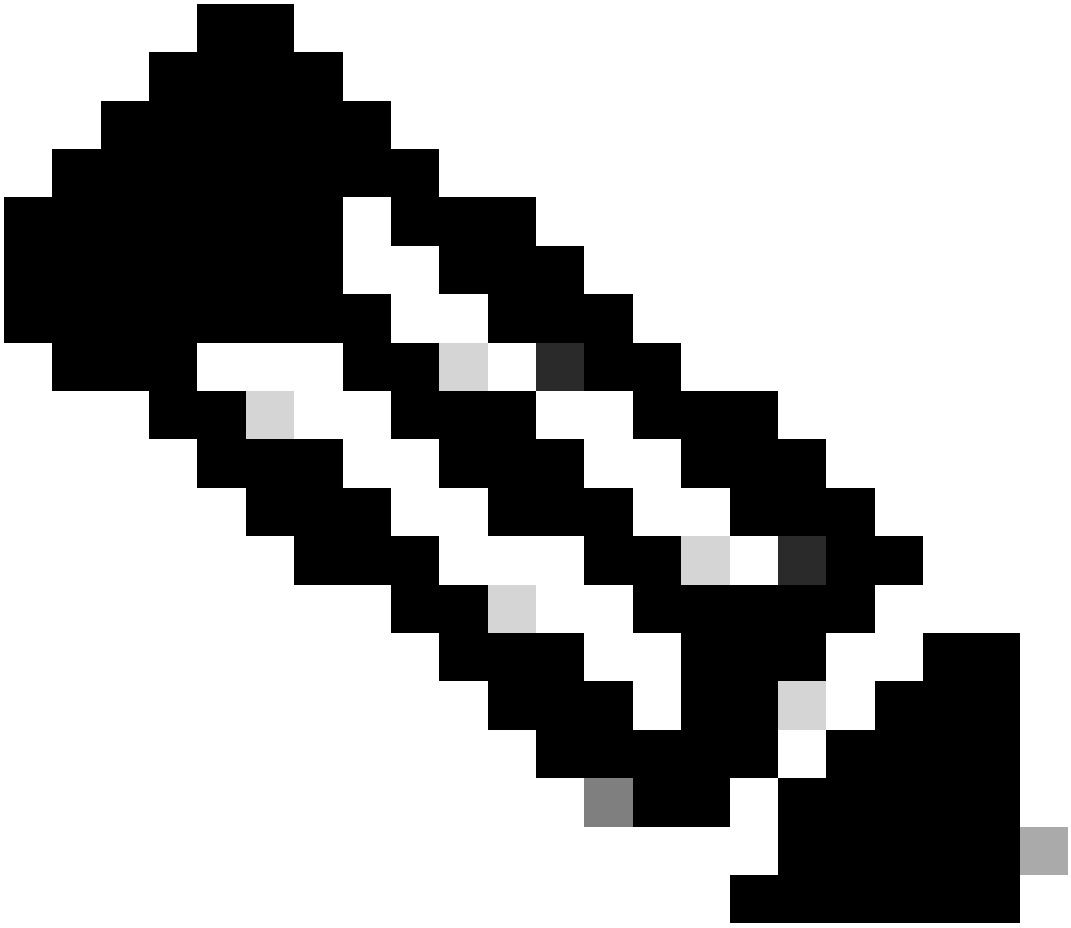
```
NAM - https://api.amp.cisco.com/v1/computers/
EU - https://api.eu.amp.cisco.com/v1/computers/
APJC - https://api.apjc.amp.cisco.com/v1/computers/
```

- 앞서 언급한 내용이 있는 스크립트와 같은 디렉토리에 config.txt 파일을 생성합니다. config.txt 파일의 예:

```
BASE_URL=https://api.apjc.amp.cisco.com/v1/computers/
API_CLIENT_ID=xxxxxxxxxxxxxxxxxxxx
API_KEY=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
```

- 커넥터 GUID 목록이 있는 스크립트와 같은 디렉터리에 한 줄에 하나씩 GUID.txt 파일을 만듭니다. 필요한 만큼 GUID를 추가합니다. guids.txt 파일의 예:

```
abXXXXXXXXXXXXcd-XefX-XghX-X12X-XXXXXX567XXXXXXX
yzXXXXXXXXXXXXlm-XprX-XmnX-X34X-XXXXXX618XXXXXXX
```



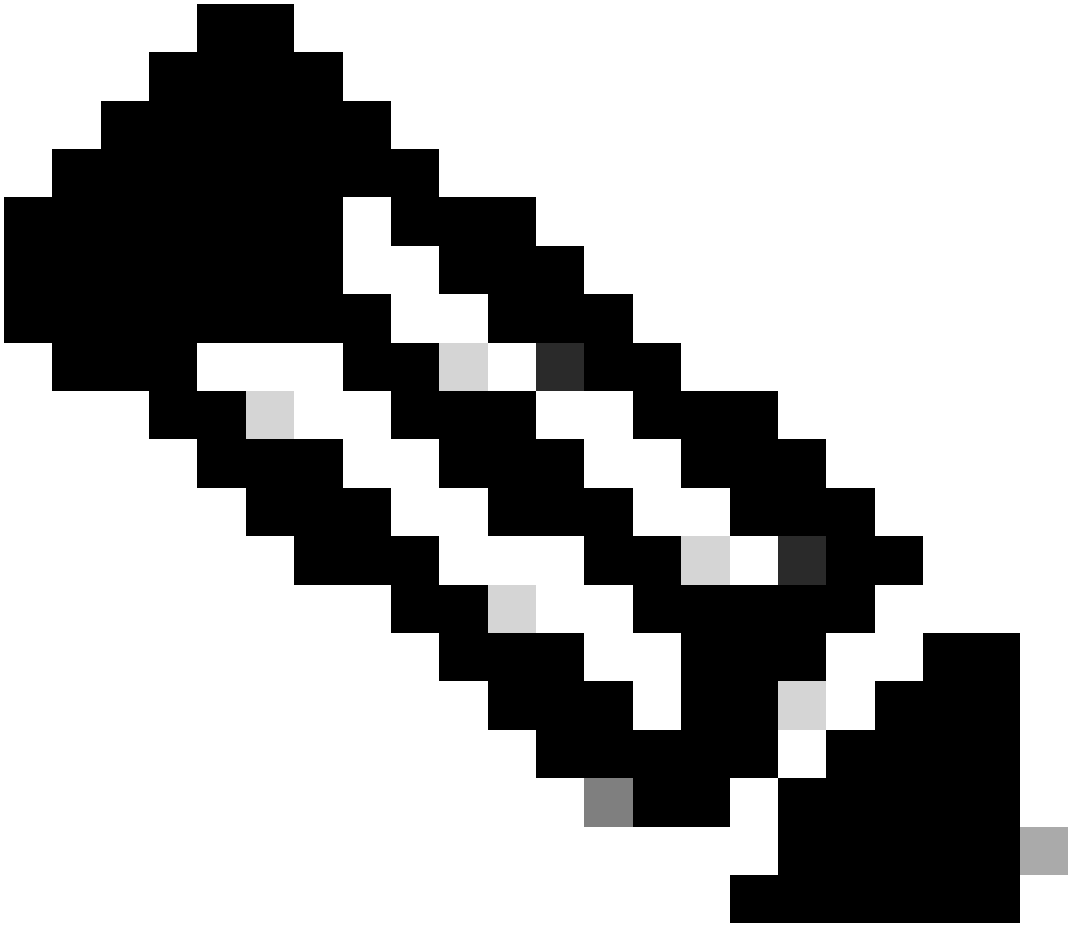
참고: API GET /v1/computers를 통해 또는 Cisco Secure Endpoint Console에서 Management(관리) > Computers(컴퓨터)로 이동하고 특정 엔드포인트에 대한 항목을 확장하며 Connector GUID를 복사하여 엔드포인트의 GUID를 수집할 수 있습니다.

- 터미널 또는 명령 프롬프트를 엽니다. start_isolation_script.py가 있는 디렉토리로 이동합니다.
- 위에서 언급한 명령을 실행하여 스크립트를 실행합니다.

```
python start_isolation_script.py
```

다음을 확인합니다.

- 스크립트는 guids.txt 파일에 지정된 각 엔드포인트를 격리합니다.
- 각 엔드포인트의 성공 또는 오류 메시지에 대해 터미널 또는 명령 프롬프트를 확인합니다.



참고: 연결된 스크립트 start_isolation.py를 사용하여 엔드포인트에서 격리를 시작할 수 있으며 stop_isolation.py는 엔드포인트에서 격리를 중지하도록 설계되었습니다. 스크립트를 실행 및 실행하기 위한 모든 명령은 동일하게 유지됩니다.

이 번역에 관하여

Cisco는 전 세계 사용자에게 다양한 언어로 지원 콘텐츠를 제공하기 위해 기계 번역 기술과 수작업 번역을 병행하여 이 문서를 번역했습니다. 아무리 품질이 높은 기계 번역이라도 전문 번역가의 번역 결과물만큼 정확하지는 않습니다. Cisco Systems, Inc.는 이 같은 번역에 대해 어떠한 책임도 지지 않으며 항상 원본 영문 문서(링크 제공됨)를 참조할 것을 권장합니다.