在多个终端上自动启动/停止隔离

目录

简介

本文档介绍如何使用思科安全终端API在多个终端上自动停止/启动隔离。

先决条件

要求

Cisco 建议您了解以下主题：

- Cisco Secure Endpoint
- 思科安全终端控制台
- 思科安全终端API
- Python

使用的组件

本文档中的信息基于以下软件版本：

- 思科安全终端8.4.0.30201
- 终端到主机Python环境
- Python 3.11.7

本文档中的信息都是基于特定实验室环境中的设备编写的。本文档中使用的所有设备最初均采用原始（默认）配置。如果您的网络处于活动状态，请确保您了解所有命令的潜在影响。

背景信息

- 使用PUT请求启动隔离。
- DELETE请求用于停止隔离。
- 有关详细信息，请参阅API文档。

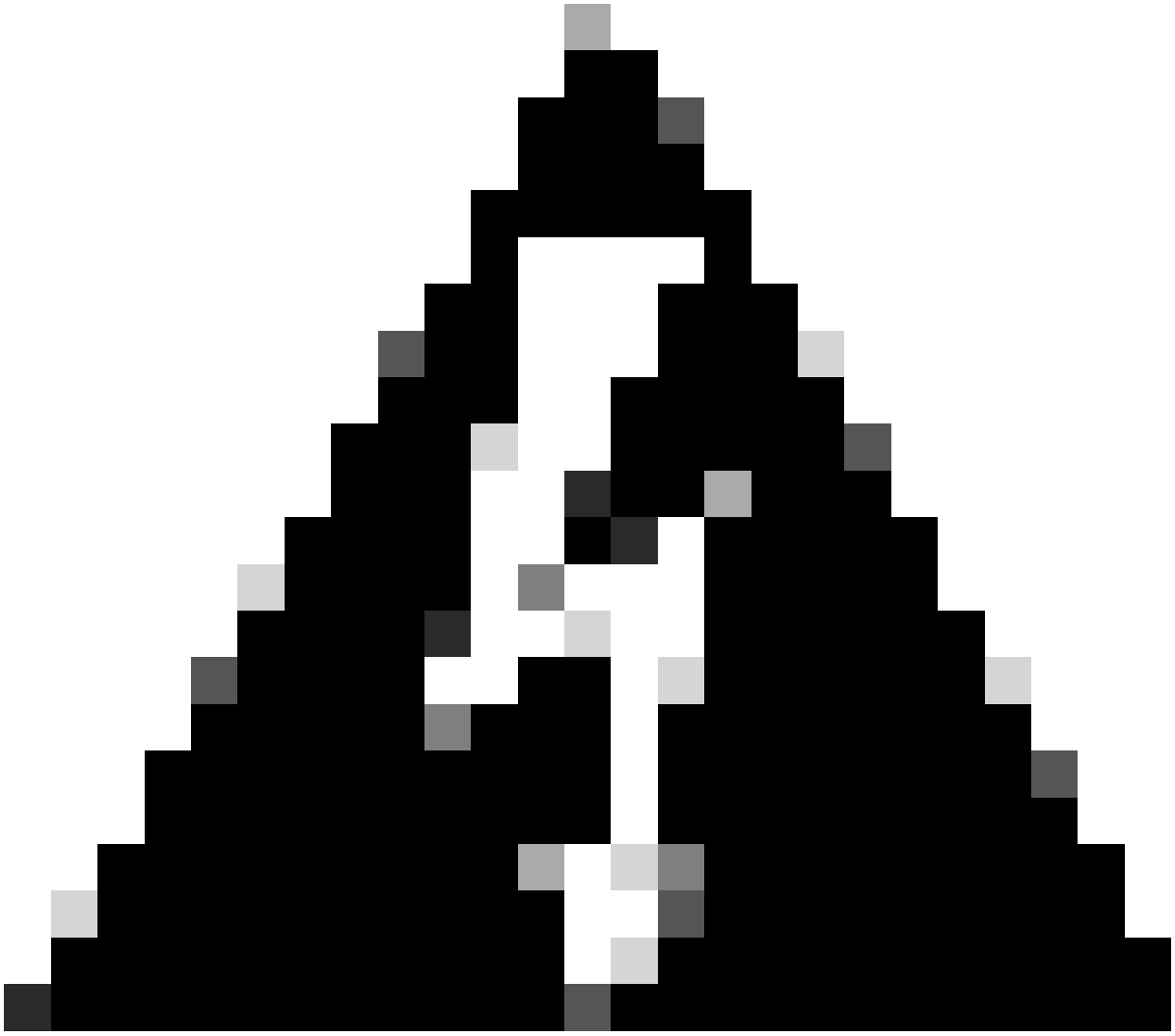
问题

思科安全终端允许一次在一台计算机上启动/停止隔离。但是，发生安全事件时，通常需要在多个终端上同时执行这些操作，才能有效遏制潜在威胁。使用API自动执行批量终端的启动/停止隔离流程，可以显著提高事件响应效率并降低网络的整体风险。

解决方案

- 本文中提供的Python脚本可用于使用安全终端API凭证在组织中的多个终端上启动/终止隔离。
- 要生成AMP API凭证，请参阅面向终端的思科AMP思科AMP的终端API概述
- 要使用提供的脚本，您需要在终端上安装pythonon。
- 安装python后，请安装请求模块

```
pip install requests
```



警告：提供该脚本只是为了说明目的，旨在演示如何使用API自动执行终端隔离功能。思科技术支持中心(TAC)不参与与该脚本相关的故障排除问题。在生产环境中部署脚本之前，用户必须小心谨慎地在安全的环境中对脚本进行全面测试。

脚本

您可以使用提供的脚本在企业的多个终端上开始隔离：

```
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)
```

说明

- 要生成AMP API凭证，请参阅面向终端的思科AMP思科AMP的终端API概述
- 使用针对您所在地区提及的BASE_URL：

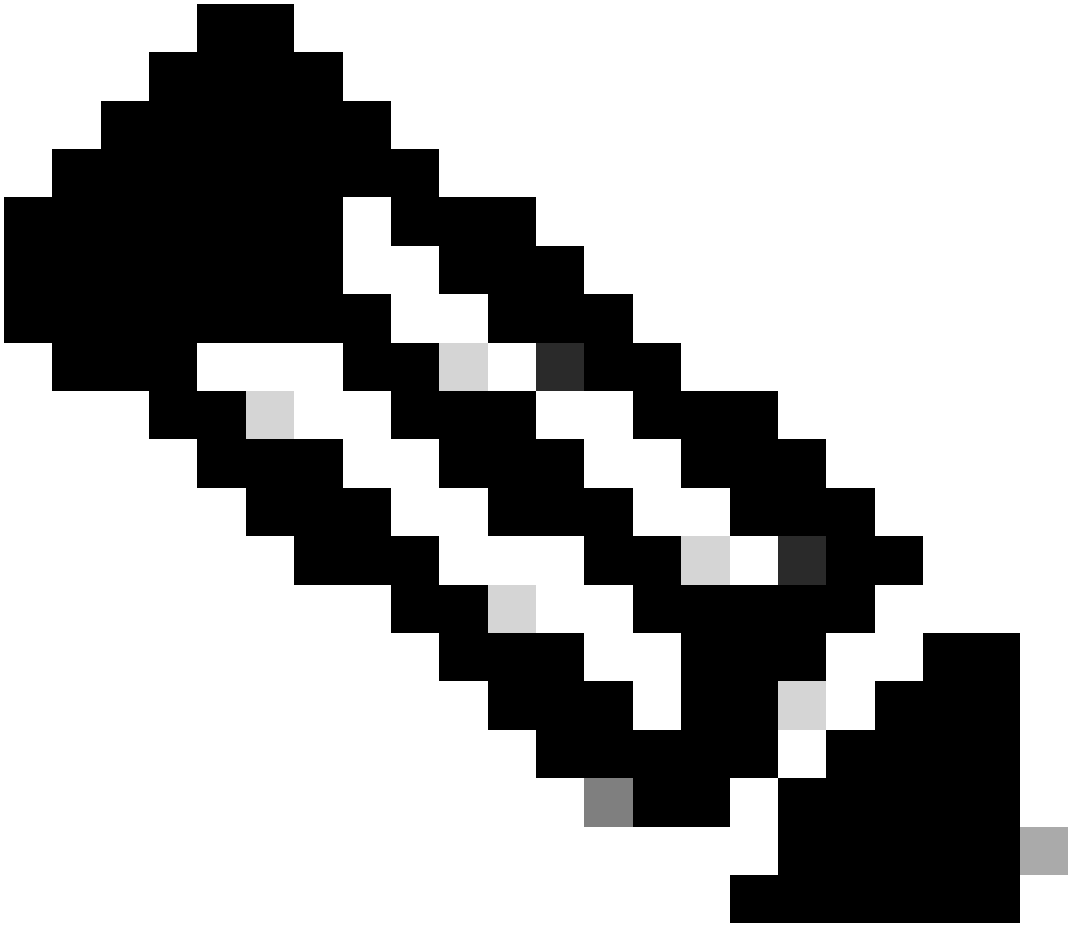
```
NAM - https://api.amp.cisco.com/v1/computers/
EU - https://api.eu.amp.cisco.com/v1/computers/
APJC - https://api.apjc.amp.cisco.com/v1/computers/
```

- 使用上述内容，在脚本所在的目录中创建config.txt文件。config.txt文件示例：

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

- 使用连接器GUID列表（每行一个）在脚本所在的目录中创建guids.txt文件。根据需要添加多个GUID。guids.txt文件示例：

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



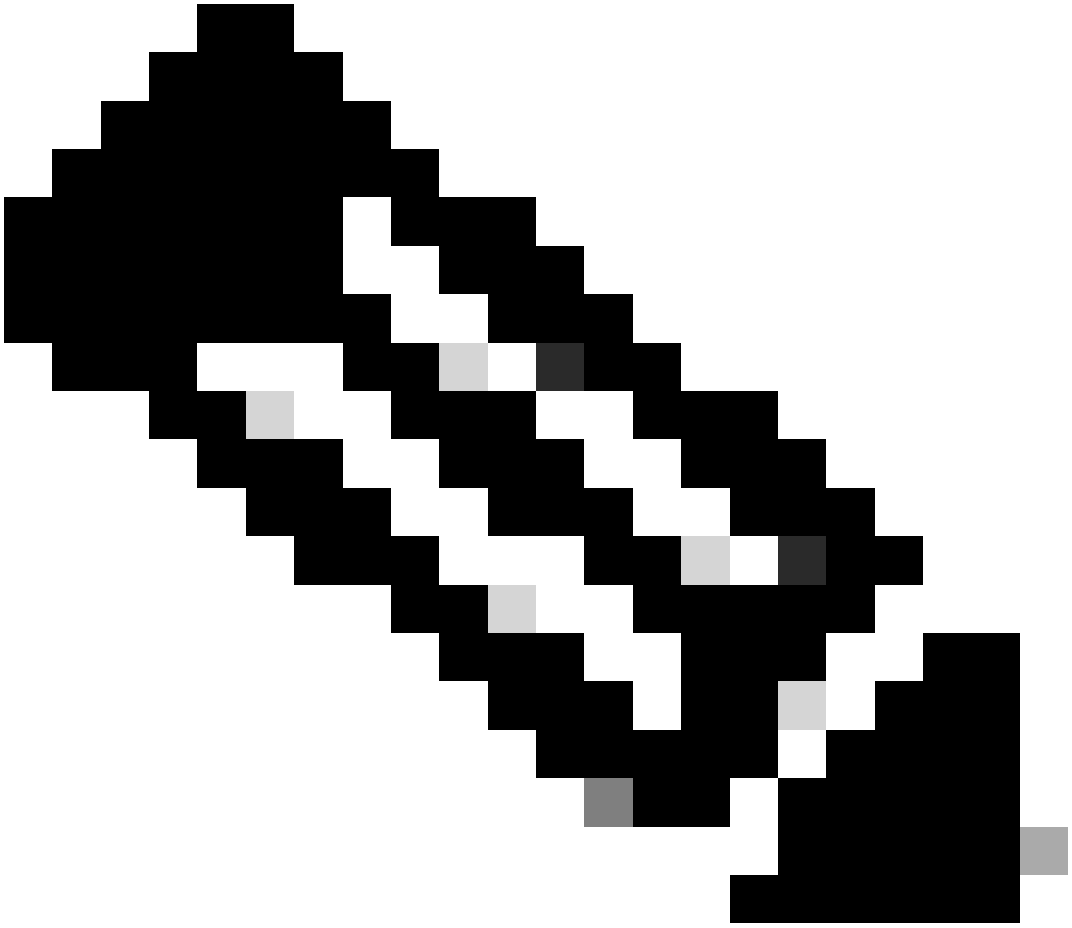
注意：您可以通过API GET /v1/computers或从Cisco安全终端控制台收集终端的GUID，方法是导航到管理>计算机，展开特定终端的条目，然后复制连接器GUID。

- 打开终端或命令提示符。导航到start_isolation_script.py所在的目录。
- 通过运行前面提到的命令执行脚本：

```
python start_isolation_script.py
```

验证

- 该脚本会尝试隔离guids.txt文件中指定的每个端点。
- 检查terminal或command prompt，了解每个终端的成功或错误消息。



注意：附加的脚本start_isolation.py可用于在端点上启动隔离，而stop_isolation.py旨在在端点上停止隔离。运行和执行脚本的所有指令都保持不变。

关于此翻译
思科采用人工翻译与机器翻译相结合的方式将此文档翻译成不同语言，希望全球的用户都能通过各自的语言得到支持性的内容。

请注意：即使是最好的机器翻译，其准确度也不及专业翻译人员的水平。

Cisco Systems, Inc. 对于翻译的准确性不承担任何责任，并建议您总是参考英文原始文档（已提供链接）。