Automate Start/Stop Isolation on Multiple Endpoints

Contents

Introduction

This document describes how to automate the stop/start isolation on multiple endpoints using the API for Cisco Secure Endpoint.

Prerequisites

Requirements

Cisco recommends that you have knowledge of these topics:

- Cisco Secure Endpoint
- Cisco Secure Endpoint Console
- Cisco Secure Endpoint API
- Python

Components Used

The information in this document is based on these software verisons:

- Cisco Secure Endpoint 8.4.0.30201
- Endpoint to host python environment
- Python 3.11.7

The information in this document was created from the devices in a specific lab environment. All of the devices used in this document started with a cleared (default) configuration. If your network is live, ensure that you understand the potential impact of any command.

Background Information

- You use a PUT request to initiate the isolation.
- A DELETE request is used to stop the isolation.
- Check the API documentation for more information.

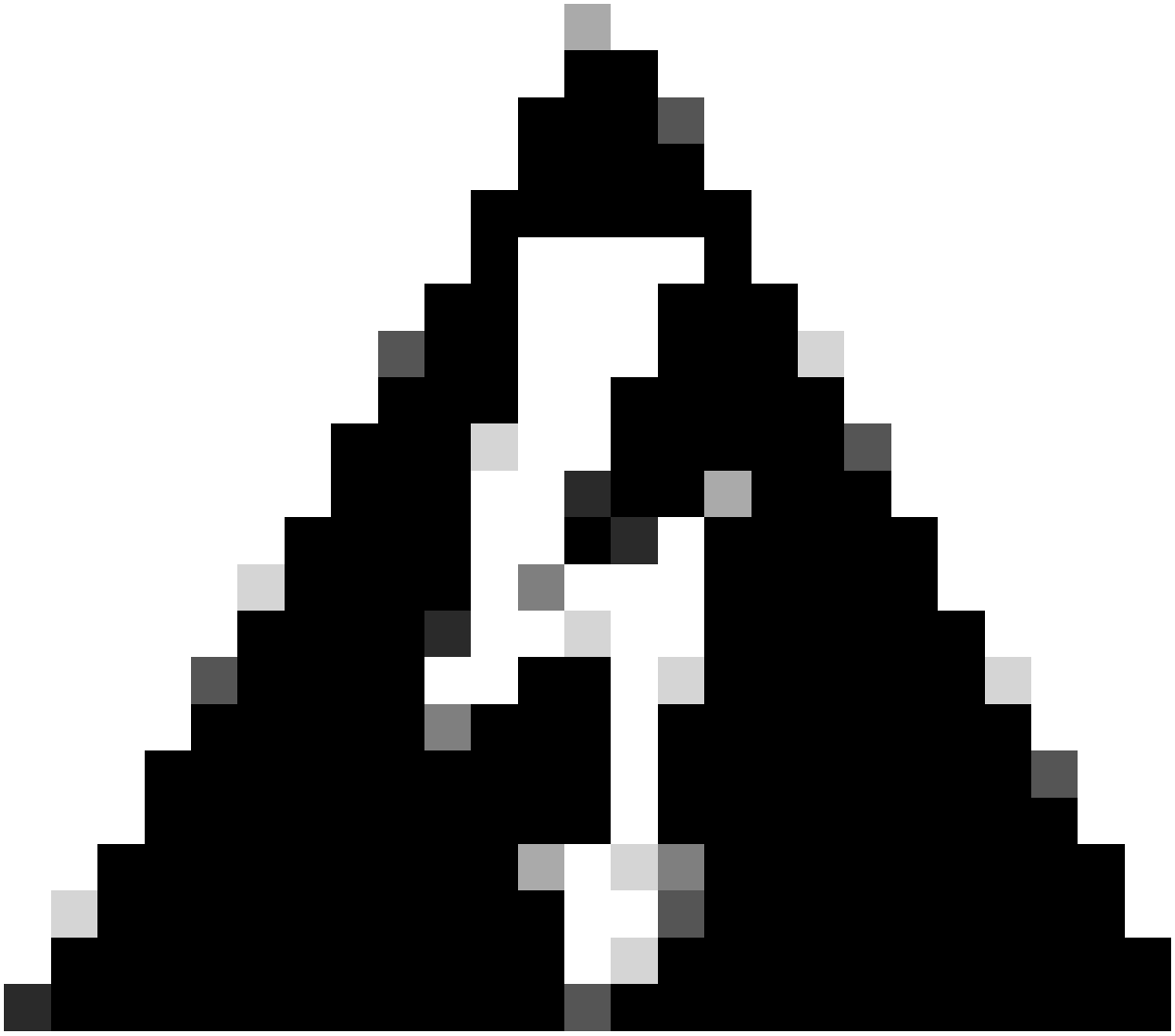
Problem

Cisco Secure Endpoint allows start/stop isolation on one machine at a time. However, during a security incident, it is often necessary to perform these operations on multiple endpoints simultaneously to effectively contain potential threats. Automating the start/stop isolation process for bulk endpoints using the API can significantly enhance incident response efficiency and reduce the overall risk to the network.

Solution

- The Python script provided in this article can be used to initiate/terminate isolation on multiple endpoints in your organization using Secure Endpoint API credentials.
- To generate the AMP API credentials, please refer to Overview of the Cisco AMP for Endpoints API
- To use the provided script, you need to installpythonon your endpoints.
- After installing python, please install **requests** module

```
pip install requests
```



Warning: The script is provided solely for illustrative purposes and is intended to demonstrate how to automate the endpoint isolation feature using the API. CiscoTechnical Assistance Center (TAC) is not involved in troubleshooting issues related to this script. Users must exercise caution and thoroughly test the script in a safe environment before deploying it in a production setting.

Script

You can use the provided script to start isolation on multiple endpoints in your business:

```
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)
```

Instructions

- To generate the AMP API credentials, please refer to [Overview of the Cisco AMP for Endpoints API]
- Use BASE_URL mentioned for your region:

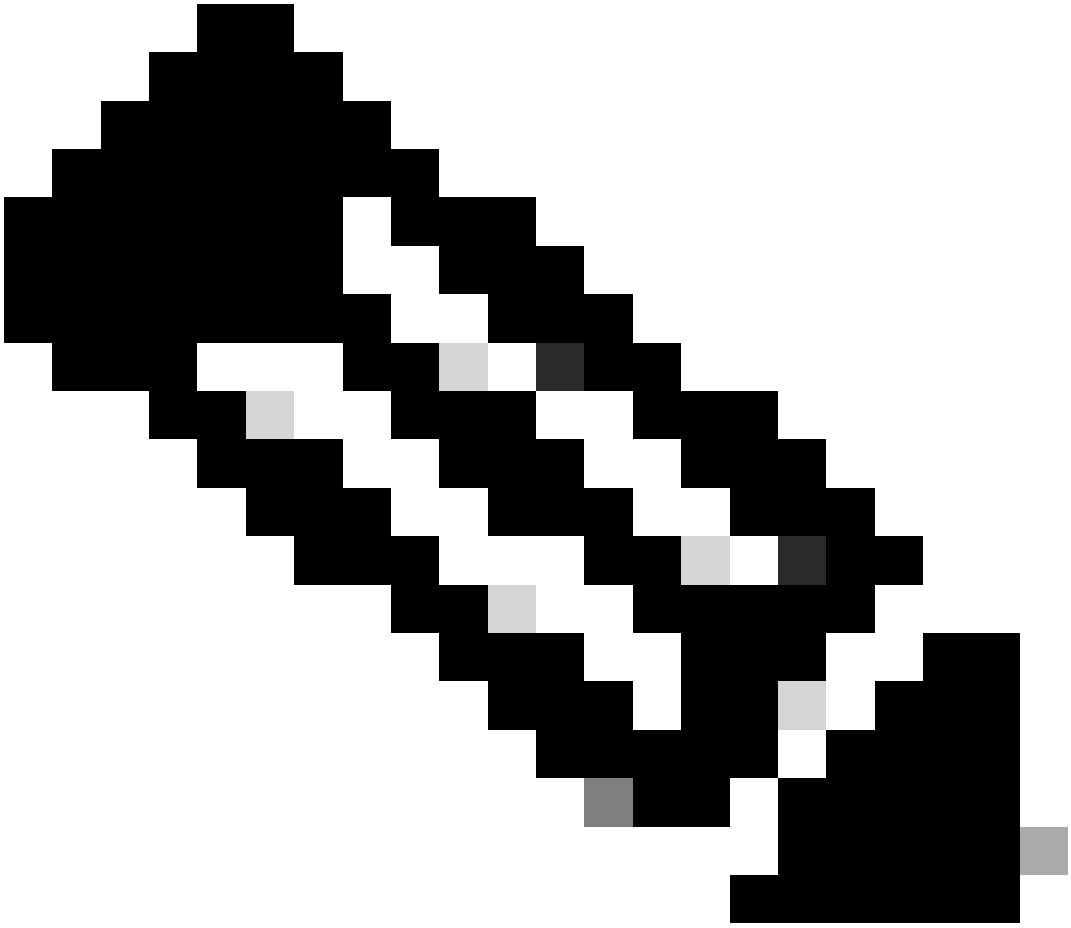
```
NAM - https://api.amp.cisco.com/v1/computers/
EU - https://api.eu.amp.cisco.com/v1/computers/
APJC - https://api.apjc.amp.cisco.com/v1/computers/
```

- Create a **config.txt** file in the same directory as the script with the mentioned content. Example of config.txt file:

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

- Create a **guids.txt** file in the same directory as the script with the list of connector GUIDs, one per line. Add as much GUIDs as needed. Example of guids.txt file:

```
abXXXXXXXXXXXcd-XefX-XghX-X12X-XXXXXX567XXXXXXX
yzXXXXXXXXXXXlm-XprX-XmnX-X34X-XXXXXX618XXXXXXX
```



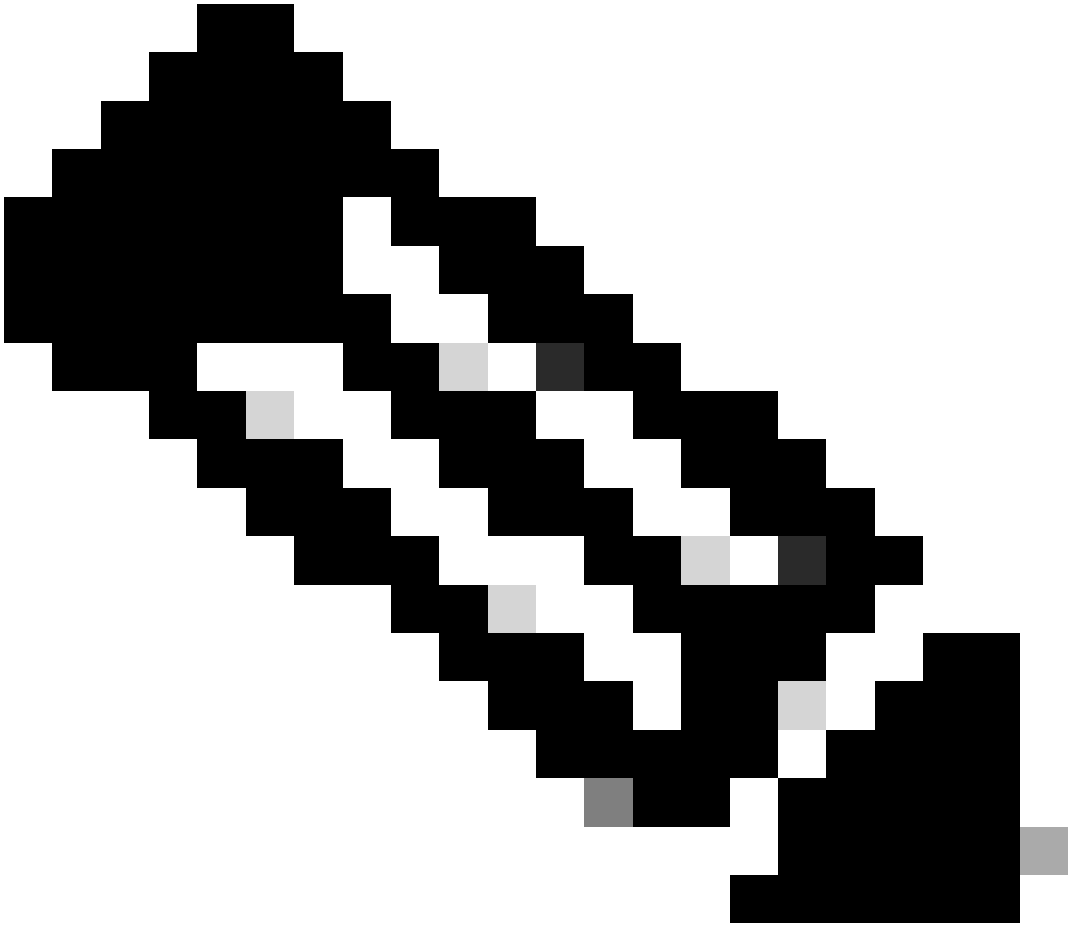
Note: You can collect the GUIDs of your endpoints either through the API GET /v1/computers or from the Cisco Secure Endpoint Console by navigating to **Management > Computers**, expanding the entry for a specific endpoint, and copying the Connector GUID.

- Open a Terminal or Command Prompt. Navigate to the **directory** where start_isolation_script.py is located.
- Execute the **script** by running the mentioned command:

```
python start_isolation_script.py
```

Verify

- The script attempts to isolate each endpoint specified in guids.txt file.
- Check the **terminal** or **command prompt** for success or error messages for each endpoint.



Note: The attached script start_isolation.py can be used to initiate isolation on endpoints, while stop_isolation.py is designed to stop the isolation on endpoints. All instructions for running and executing the script remain the same.