Gestire gli elenchi di destinazioni di accesso sicuro tramite l'API Python e REST

Sommario

Spoiler (Evidenziato da leggere)
Si tenga presente che Cisco non fornisce supporto ufficiale per questo progetto di sviluppo. È inteso esclusivamente come esempio di riferimento per facilitare la comprensione del modo in cui l'API interagisce con le applicazioni. Gli utenti devono utilizzare questo progetto solo a scopo didattico e non come base per l'implementazione a livello di produzione. L'esecuzione del codice presentato in questo articolo avviene a proprio rischio e Cisco declina espressamente qualsiasi responsabilità per eventuali problemi derivanti dal suo utilizzo.
Si tenga presente che Cisco non fornisce supporto ufficiale per questo progetto di sviluppo. È inteso esclusivamente come esempio di riferimento per facilitare la comprensione del modo in cui l'API interagisce con le applicazioni. Gli utenti devono utilizzare questo progetto solo a scopo didattico e non come base per l'implementazione a livello di produzione. L'esecuzione del codice presentato in questo articolo avviene a proprio rischio e Cisco declina espressamente qualsiasi responsabilità per eventuali problemi derivanti dal suo utilizzo.

Introduzione

Questo documento descrive come eseguire tutte le possibili operazioni sugli elenchi di destinazione utilizzando Python e l'API REST.

Prerequisiti

Cisco raccomanda la conoscenza dei seguenti argomenti:

1. Python
2. API REST
3. Cisco Secure Access

Requisiti

Prima di procedere, è necessario soddisfare i seguenti requisiti:

- Account utente Cisco Secure Access con il ruolo di amministratore completo.
- Account Cisco Security Cloud Single Sign On (SCSO) per accedere a Secure Access.
- Creare le chiavi dell'API Secure Access

Componenti usati

Le informazioni fornite in questo documento si basano sulle seguenti versioni software e hardware:

- Dashboard di accesso protetto
- Python 3.x

Le informazioni discusse in questo documento fanno riferimento a dispositivi usati in uno specifico ambiente di emulazione. Su tutti i dispositivi menzionati nel documento la configurazione è stata ripristinata ai valori predefiniti. Se la rete è operativa, valutare attentamente eventuali conseguenze derivanti dall'uso dei comandi.

Configurazione

Esistono diversi modi per scrivere questo codice considerando diversi aspetti come la gestione degli errori, la validità del token (3600 secondi) e così via.

Assicurarsi che le seguenti librerie Python siano installate prima di eseguire lo script:

```
pip install requests
pip install oauthlib
pip install requests_oauthlib
```

Script

Assicurarsi di sostituire il`client_id`e il`client_secret`con il`API Key`e`Key Secret`in questo script, rispettivamente.

```
from oauthlib.oauth2 import BackendApplicationClient
from oauthlib.oauth2 import TokenExpiredError
from requests_oauthlib import OAuth2Session
from requests.auth import HTTPBasicAuth
import time
import requests
import pprint
import json
```

```
def fetch_headers(BToken):
    BT = f"Bearer {BToken}"
    headers = { 'Authorization':BT,
            "Content-Type": "application/json",
            "Accept": "application/json"
            }
    return headers

# GET OAUTH 2.0 TOKEN
def getToken():
    token_url = 'https://api.sse.cisco.com/auth/v2/token'
    try:
        #ASSIGN your API Key to the variable client_id and Secret Key to the variable client_secret
        client_id = "XXXXXXXXXXXXXXXXXXXXXXXXXXXX"
        client_secret = "XXXXXXXXXXXXXXXXXXXXXXXXXXXX"

        auth = HTTPBasicAuth(client_id, client_secret)
        client = BackendApplicationClient(client_id=client_id)
        oauth = OAuth2Session(client=client)
        token = oauth.fetch_token(token_url=token_url, auth=auth)
        print("\n######Token Generated Successfully######\n")
        return token
    except e as Exception:
        print(f"Encountered an error while Fetching the TOKEN :: {e}")


# 1 - GET DESTINATION LISTS
def fetch_destinationlists(h):
    url = "https://api.sse.cisco.com/policies/v2/destinationlists"
    try:
        response = requests.request('GET', url, headers=h)
        json_object = json.loads(response.content)

        #pprint.pprint(json_object)
        x=1
        for item in json_object["data"]:
            print(f"Destination List : {x}")
            pprint.pprint(f"Name : {item['name']}")
            pprint.pprint(f"ID : {item['id']}")
            #pprint.pprint(f"Destination Count : {item['meta']['destinationCount']}")
            print("\n")
            x+=1
    except e as Exception:
        print(f"Encountered an Error while Fetching the Destination Lists :: {e}")


# 2 - GET DESTINATION LIST
def get_destinationlist(h):
    try:
        choice = input("Enter the ID of the DestinationList:: ")
        url = "https://api.sse.cisco.com/policies/v2/destinationlists/" + choice
        response = requests.request('GET', url, headers=h)
        json_object = json.loads(response.content)
        print("\n\n")
        pprint.pprint(json_object)
        print("\n\n")
    except e as Exception:
        print(f"Encountered an Error while Fetching the Destination List Details :: {e}")


# 3 - CREATE DESTINATION LIST
def create_destinationlist(h):
```

```
    url = "https://api.sse.cisco.com/policies/v2/destinationlists"
    try:
        naav = input("Name of the DestinationList :: ")
        payload = {
            "access": "none",
            "isGlobal": False,
            "name": naav,
            }
        response = requests.request('POST', url, headers=h, data = json.dumps(payload))
        json_object = json.loads(response.content)
        print("\n\n")
        pprint.pprint(json_object)
        print("\n\n")
    except e as Exception:
        print(f"Encountered an Error while Creating the Destination List :: {e}")


# 4 - UPDATE DESTINATION LIST NAME
def patch_destinationlist(h):
    try:
        choice = input("Enter the ID of the DestinationList for changing it's name :: ")
        url = "https://api.sse.cisco.com/policies/v2/destinationlists/" + choice
        naav = input("Enter New Name :: ")
        payload = {"name": naav}

        response = requests.request('PATCH', url, headers=h, data = json.dumps(payload))
        json_object = json.loads(response.content)
        print("\n\n")
        pprint.pprint(json_object)
        print("\n\n")
    except e as Exception:
        print(f"Encountered an Error while Updating the Destination List :: {e}")


# 5 - DELETE DESTINATION LIST
def delete_destinationlist(h):
    try:
        choice = input("Enter the ID of the DestinationList for DELETION :: ")
        url = "https://api.sse.cisco.com/policies/v2/destinationlists/" + choice

        response = requests.request('DELETE', url, headers=h)
        json_object = json.loads(response.content)
        print("\n\n")
        pprint.pprint(json_object)
        print("\n\n")
    except Exception as e:
        print(f"Encountered an Error while Deleting the Destination List :: {e}")


# 6 - GET DESTINATIONS FROM A DESTINATION LIST
def fetch_detail(h):
    try:
        choice = input("DestinationList ID: ")

        url2 = "https://api.sse.cisco.com/policies/v2/destinationlists/" + choice + "/destinations"

        response = requests.request('GET', url2, headers=h)
        print("\n")
        json_dest = json.loads(response.content)
        pprint.pprint(json_dest)
        print("\n\n")
    except e as Exception:
```

```
        print(f"Encountered an Error while Fetching the Destinations from the Destination List :: {e}")


# 7 - ADD DESTINATIONS TO A DESTINATION LIST
def add_destinations(h):
    try:
        choice = input("Enter the ID of the DestinationList :: ")
        url = "https://api.sse.cisco.com/policies/v2/destinationlists/" + choice + "/destinations"
        destination_to_add = input("\nEnter the destination that you want to add :: ")
        payload = [{"destination": destination_to_add}]

        response = requests.request('POST', url, headers=h, data = json.dumps(payload))
        print("\n")
        json_dest = json.loads(response.content)
        pprint.pprint(json_dest)
        print("\n\n")
    except e as Exception:
        print(f"Encountered an Error while Adding the Destination to the Destination List :: {e}")


# 8 - DELETE DESTINATIONS FROM DESTINATION LIST
def delete_entry(h):
    try:
        choice_del = input("\nCONFIRM DestinationList ID from which you want to delete the Destination
        url3 = "https://api.sse.cisco.com/policies/v2/destinationlists/" + choice_del + "/destinations/
        dest = int(input("ID of the Destination that you want to remove: "))

        payload = f"[{dest}]"
        response = requests.request('DELETE', url3, headers=h, data=payload)
        json_del = json.loads(response.content)
        print("\n\n")
        pprint.pprint(json_del)
        print("\n\n")
    except e as Exception:
        print(f"Encountered an Error while Deleting a Destination from the Destination List :: {e}")



#FETCH COOKIE
possess_cookie = " "
while possess_cookie not in ["Y","y","N","n"]:
    possess_cookie = input("Token Already Generated? (Y/N) :: ")
    if possess_cookie.upper() =="N":
        cook = getToken()
        with open("cookie.txt","w") as wr:
            wr.writelines(cook["access_token"])
#         print(f"Access Token = {cook["access_token"]}")


#FETCH HEADERS
with open("cookie.txt","r") as ree:
    h = fetch_headers(ree.readline())


print("\n")
while True:
    action = input("""Available operations:
1. Get Destination Lists
2. Get Destination List
3. Create Destination List
4. Update Destination List Name
5. Delete Destination List
```

```
6. Get Destinations from Destination List
7. Add Destinations to a Destination List
8. Delete Destinations from Destination List
9. Exit

Enter Your Choice :: """)

    print("\n")
    operation = action.replace(" ","")
    if operation == "1":
        fetch_destinationlists(h)

    elif operation == "2":
        fetch_destinationlists(h)
        get_destinationlist(h)

    elif operation == "3":
        create_destinationlist(h)

    elif operation == "4":
        fetch_destinationlists(h)
        patch_destinationlist(h)

    elif operation == "5":
        fetch_destinationlists(h)
        delete_destinationlist(h)

    elif operation == "6":
        fetch_destinationlists(h)
        fetch_detail(h)

    elif operation == "7":
        fetch_destinationlists(h)
        add_destinations(h)

    elif operation == "8":
        fetch_destinationlists(h)
        fetch_detail(h)
        delete_entry(h)

    elif operation == "9":
        break

    else:
        print("\n")
        print("==========INCORRECT INPUT==========")
        print("\n")

print("Thank You!!! Good Bye!!!")
time.sleep(5)
```

Uscita:

L'output di questo script deve essere simile al seguente:

```
Token Already Generated? (Y/N) :: y
```

```
Available operations:
1. Get Destination Lists
2. Get Destination List
3. Create Destination List
4. Update Destination List Name
5. Delete Destination List
6. Get Destinations from Destination List
7. Add Destinations to a Destination List
8. Delete Destinations from Destination List
9. Exit

Enter Your Choice :: 1
```

Una volta che questo programma è stato eseguito con successo, viene posta una domanda all'inizio circa il Cookie -Cookie Already Generated? (Y/N). Questa domanda viene posta per evitare che il cookie venga generato più volte, poiché una volta generato, il cookie è valido per 3600 secondi (1 ora). Se si immetteyoY, non viene generato un nuovo cookie. Tuttavia, se si immettenoN, viene generato un nuovo cookie che viene salvato in un file di testo locale nella stessa directory/cartella. Il cookie di questo file viene utilizzato nelle richieste successive.

Errori

È possibile che questo errore si verifichi se si immette un ID non corretto per qualsiasi operazione che richiede di indicare l'ID DestinationList:

```
{'message': 'no Route matched with those values'}
```

Durante la creazione di un oggetto DestinationList, se si specifica il nome di un oggetto DestinationList che supera i 255 caratteri, verrà visualizzato il seguente errore:

```
{'code': 400,
'code_text': 'Bad Request',
'error': 'invalid_request',
'message': {'name': {'code': 'string_invalid',
'code_text': 'must be fewer than 255 characters long'}},
'statusCode': 400,
'txId': 'l23f0cjk62fe'}
```

È inoltre possibile recuperare informazioni su criteri, computer mobili, report e così via, tramite la Guida per l'utente degli sviluppatori Secure Access.

Risoluzione dei problemi

Gli endpoint API di accesso sicuro utilizzano i codici di risposta HTTP per indicare l'esito positivo o

negativo di una richiesta API. In generale, i codici nell'intervallo 2xx indicano un esito positivo, i codici nell'intervallo 4xx indicano un errore risultante dalle informazioni fornite e i codici nell'intervallo 5xx indicano errori del server. L'approccio per risolvere il problema dipende dal codice di risposta ricevuto:

200	**OK**	Success. Everything worked as expected.
201	**Created**	New resource created.
202	**Accepted**	Success. Action is queued.
204	**No Content**	Success. Response with no message body.
400	**Bad Request**	Likely missing a required parameter or malformed JSON. The syntax of your query may need to be revised. Check for any spaces preceding, trailing, or in the domain name of the domain you are trying to query.
401	**Unauthorized**	The authorization header is missing or the key and secret pair is invalid. Ensure your API token is valid.
403	**Forbidden**	The client is unauthorized to access the content.
404	**Not Found**	The requested resource doesn't exist. Check the syntax of your query or ensure the IP and domain are valid.
409	**Conflict**	The client requests that the server create the resource, but the resource already exists in the collection.
429	**Exceeded Limit**	Too many requests received in a given amount of time. You may have exceeded the rate limits for your organization or package.
413	**Content Too Large**	The request payload is larger than the limits defined by the server.

API REST - Codici di risposta 1

500	**Internal Server Error**	Something wrong with the server.
503	**Service Unavailable**	Server is unable to complete request.

API REST - Codici di risposta 2

Informazioni correlate

- Guida per l'utente di Cisco Secure Access
- Supporto tecnico Cisco e download

- Aggiungi chiavi API di accesso sicuro
- Guida per l'utente per gli sviluppatori
- Configurazione di Secure Access per l'utilizzo dell'API REST con Python
- Gestisci elenchi di destinazione tramite cURL

Informazioni su questa traduzione

Cisco ha tradotto questo documento utilizzando una combinazione di tecnologie automatiche e umane per offrire ai nostri utenti in tutto il mondo contenuti di supporto nella propria lingua. Si noti che anche la migliore traduzione automatica non sarà mai accurata come quella fornita da un traduttore professionista. Cisco Systems, Inc. non si assume alcuna responsabilità per l’accuratezza di queste traduzioni e consiglia di consultare sempre il documento originale in inglese (disponibile al link fornito).