PythonおよびAsyncOS REST APIを使用したESAディクショナリの設定

内容

はじめに

このドキュメントでは、AsyncOS API(REST API)を使用してPythonスクリプトを使用し、新しいディクショナリをCisco Eメールセキュリティアプライアンス(ESA)に追加する方法について説明します。

前提条件

次の前提条件を確認してください。

- Pythonプログラミング言語
- Python `requests`モジュール
- JSON
- Cisco Email Security Appliance (ESA)

要件

スクリプトを実行するには、次のソフトウェアとアクセス権を使用します。

- AsyncOS 14.x以降を実行するCisco Eメールセキュリティアプライアンス(ESA)
- Python 3

- Python `requests`モジュール
- AsyncOS API(REST API)を使用する権限を持つESA管理者クレデンシャル
- ポート`6443`（推奨）でのHTTPSによるESA管理インターフェイスへのアクセス

使用するコンポーネント

この実習では、次のコンポーネントを使用しました。

1) Cisco ESA - 15.0.0-104

2)崇高なテキストエディタ

このドキュメントの情報は、特定のラボ環境にあるデバイスに基づいて作成されたものです。このドキュメントで使用するすべてのデバイスは、クリアな（デフォルト）設定で作業を開始しています。本稼働中のネットワークでは、各コマンドによって起こる可能性がある影響を十分確認してください。

転送と認証

- トランスポート：AsyncOS APIのポート`6443`でHTTPSを使用します（使用可能な場合）。HTTP/6080は、非実稼働環境またはラボ環境でのみ使用してください。
- 認証：この例では、HTTP Basic認証を使用します。ヘッダー「`Authorization: Basic <base64(username:password)>`」を指定します。実際のクレデンシャルをスクリプトに貼り付けないでください。

バックグラウンド情報

このドキュメントの内容：

目標：PythonスクリプトとAsyncOS API(REST API)を使用して、ESAにディクショナリを追加します。

期待される結果：APIは正常な応答を返し、新しいディクショナリがESA設定に表示されます。

設定

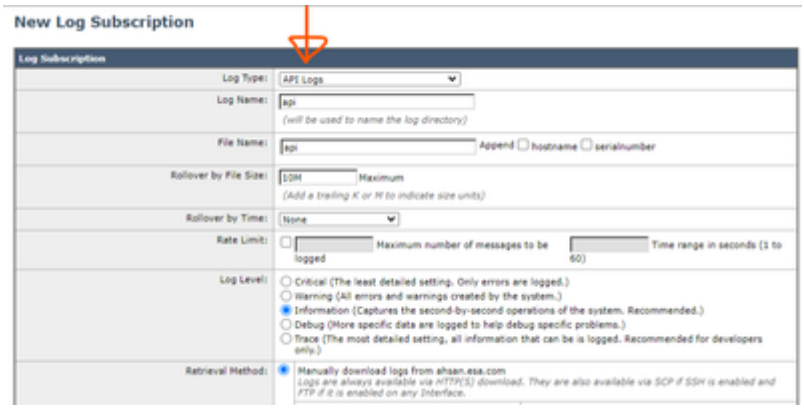
スクリプトを実行する前に、ESA管理インターフェイスでAsyncOS API(REST API)サービスを有効にします。

1. グラフィカルユーザインターフェイス(GUI)で、AsyncOS API HTTPおよびAsyncOS API HTTPS設定が含まれているネットワークページまたはインターフェイスページに移動します。
2. AsyncOS API HTTPSを有効にして、ポート6443を確認します。
3. （オプション）AsyncOS API HTTPを有効にして、ラボで使用するポート6080を確認します。
4. [Save] をクリックします。

図1：管理インターフェイスでAsyncOS API HTTPS（ポート6443）が有効になっていることを確認します。

AsyncOS API要求をトラブルシューティングするためのAPIロギングの設定

1. グラフィカルユーザインターフェイス(GUI)で、System Administration > Log Subscriptionsに移動します。
2. Add Log Subscriptionをクリックします。
3. Log Typeドロップダウンリストから、API Logsを選択します。
4. ログ名やファイル名などの残りの必須フィールドに入力し、Submitをクリックします。



Log Typeドロップダウンリストから API Logsを選択し、残りの必須詳細を入力します。

注：ログはGUIから設定できます。

スクリプト

Pythonのrequestsモジュールから HTTPのPOSTメソッドを使用して、ディクショナリを作成します。

POST要求には次のパラメータを使用します。

項目	場所	Required	注意事項
device_type（デバイスのタイプ）	URLクエリ文字列	Yes	esaを設定します。
無視する	JSONペイロード	Yes	0 =大文字と小文字を区別する、1 =大文字と小文字を区別しない。
全ての語	JSONペイロード	Yes	0 =部分文字列の一致、1 =単語単位の一致。
符号化	JSONペイロード	Yes	別のエンコーディングが必要でない場合は、utf-8を使用します。
言葉	JSONペイロード	Yes	エントリの配列。リテラル用語には["term"]を使用します。スマートIDには、ESAでサポートされている["*credit", 2, "prefix"]などのタプル形式を使用します。
モード	URLクエリ文字列	いいえ	クラスタ化された/グループの展開にのみ必要です（たとえば、mode=cluster）。

```
import base64
import requests

# ESA management IP address or FQDN
host = "



     "

# Dictionary name to create
dictionary_name = "



      "

# AsyncOS API endpoint (HTTPS recommended)
url = f"https://{host}:6443/esa/api/v2.0/config/dictionaries/{dictionary_name}?device_type=esa"

payload = {
    "data": {
        "ignorecase": 0,
        "wholewords": 1,
        "words": [
            ["*credit", 2, "prefix"],
            ["*aba"],
            ["Example Term"]
```

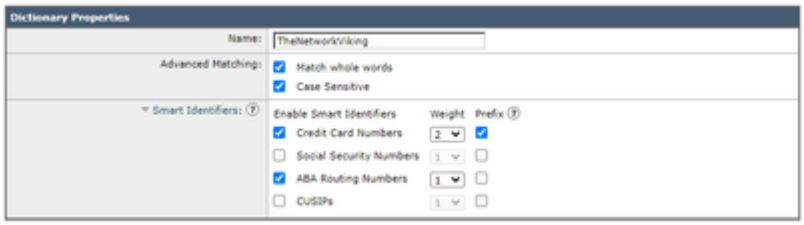
```
        ],
        "encoding": "utf-8"
    }
}

# Basic authentication header
username = "



       "
password = "



        "
creds = base64.b64encode(f"{username}:{password}".encode()).decode()

headers = {
    "Content-Type": "application/json",
    "Authorization": f"Basic {creds}"
}

response = requests.post(url, headers=headers, json=payload, timeout=30, verify=False)
print(f"Status code: {response.status_code}")
try:
    print(response.json())
except ValueError:
    print(response.text)

if response.status_code != 201:
    raise SystemExit("Dictionary creation failed.")
```

スクリプトでは、ディクショナリ名(**<DICT_NAME>**)が要求URLパスに含まれています。

必要なパラメータ（**ignorecase**、**wholewords**、**words**、および**encoding**）をJSONペイロードに含めます。URLクエリ文字列に**device_type**を追加します。

mode パラメータは、ESAがクラスタまたはグループ内にある場合にのみ必要です。その場合は、device_typeと同じ方法で、URLクエリ文字列にmode=clusterを追加します。

*credit と *aba がペイロードに追加され、クレジットカード番号とABAルーティング番号のスマート識別子を有効にします（スクリーンショットを参照）。



Example Termという用語が、照合する辞書項目としてペイロードに追加されます。

確認

POST要求が成功すると、APIログに次のようなエントリが記録されます。

```
API log example
Thu October 5 12:58:19 2023 Info: 198.51.100.10 - - 05/Oct/2023 12:58:19 +0000 POST /esa/api/v2.0/confi
```

予測される応答:

```
{"data": {"message": "Added Successfully"}}
```

ディクショナリがすでに存在し、POST要求が再実行される場合、APIログには次のようなエントリが含まれます。

```
API log example
Thu October 5 13:00:31 2023 Info: 198.51.100.10 - - 05/Oct/2023 13:00:31 +0000 POST /esa/api/v2.0/confi
```

予測される応答:

```
{"error": {"message": "Dictionary already exists.", "code": "409", "explanation": "409 = Request confli
```

関連情報

追加の背景情報については、次の外部リソースを使用してください（このドキュメントの手順は

完全独立型です）。

REST API + Pythonを使用した辞書の追加

このドキュメントには、必要な認証方式と最小限の手順が記載されています。ヘッダー `Authorization: Basic <base64(username:password)>`の付いたHTTP Basic認証を使用し、ポート `6443`でHTTPSを優先します。

REST APIとCisco ESA

さらに、この手順で使用されるPythonコードはPostmanから抽出できます。必要に応じて、このビデオチュートリアルを確認してください。

Postman - Pythonスクリプトの生成方法

翻訳について
シスコは世界中のユーザにそれぞれの言語でサポート コンテンツを提供するために、機械と人による翻訳を組み合わせて、本ドキュメントを翻訳しています。ただし、最高度の機械翻訳であっても、専門家による翻訳のような正確性は確保されません。シスコは、これら翻訳の正確性について法的責任を負いません。原典である英語版（リンクからアクセス可能）もあわせて参照することを推奨します。