Python Script for Creating Akamai Access Groups

# Overview

This Python script allows for the creation of an Akamai access group using Akamai's API. The script accepts command-line parameters including the access group name, its description, and the environment (e.g., default or production). The script checks if the access group already exists before attempting to create it, ensuring no duplicate groups are created.

A screenshot of a computer screen

Description automatically generated

# Imports and Setup

The script begins by importing necessary libraries:

import os  
import requests  
import argparse  
from akamai.edgegrid import EdgeGridAuth, EdgeRc

- os: Used for handling file paths.  
- requests: Used for making HTTP requests to the Akamai API.  
- argparse: Used for parsing command-line arguments.  
- akamai.edgegrid: Used for handling Akamai EdgeGrid authentication with the .edgerc file.

# Loading Credentials from .edgerc

The load\_edgerc function loads credentials from the .edgerc file based on the specified environment:

def load\_edgerc(section="default"):  
 edgerc = EdgeRc(os.path.expanduser("~/.edgerc"))  
 base\_url = f"https://{edgerc.get(section, 'host')}/"  
 session = requests.Session()  
 session.auth = EdgeGridAuth.from\_edgerc(edgerc, section)  
 return session, base\_url

# Checking If the Access Group Exists

The check\_access\_group\_exists function checks whether an access group with the specified name already exists:

def check\_access\_group\_exists(session, base\_url, group\_name):  
 endpoint = f"{base\_url}identity/v1/groups"  
 response = session.get(endpoint)  
  
 if response.status\_code == 200:  
 groups = response.json()  
 for group in groups['groups']:  
 if group['groupName'] == group\_name:  
 return True  
 else:  
 print(f"Error fetching access groups: {response.status\_code}")  
 return False

# Creating the Access Group

The create\_access\_group function creates a new access group if it doesn’t already exist:

def create\_access\_group(session, base\_url, group\_name, description):  
 if not check\_access\_group\_exists(session, base\_url, group\_name):  
 endpoint = f"{base\_url}identity/v1/groups"  
 payload = {  
 "groupName": group\_name,  
 "description": description  
 }  
 response = session.post(endpoint, json=payload)  
 if response.status\_code == 201:  
 print(f"Access group '{group\_name}' created successfully.")  
 else:  
 print(f"Failed to create access group. Status code: {response.status\_code}")  
 print(f"Response: {response.text}")  
 else:  
 print(f"Access group '{group\_name}' already exists.")

# Command-Line Argument Parsing

The main function handles the command-line arguments:

def main():  
 parser = argparse.ArgumentParser(description="Create an Akamai access group.")  
 parser.add\_argument("group\_name", help="Name of the access group to be created")  
 parser.add\_argument("description", help="Description for the new access group")  
 parser.add\_argument("--env", default="default", help="Environment section in .edgerc file (default or prod)")  
  
 args = parser.parse\_args()  
  
 session, base\_url = load\_edgerc(section=args.env)  
  
 create\_access\_group(session, base\_url, args.group\_name, args.description)  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 main()

# Running the Script

To run the script, use the following command:  
```bash  
python akamai\_access\_group.py "NewAccessGroup" "This is a description for the new access group"  
```  
This will create an access group named "NewAccessGroup" with the specified description using credentials from the "default" section of `.edgerc`.

To use the production environment, run:  
```bash  
python akamai\_access\_group.py "NewAccessGroup" "This is a description for the new access group" --env prod  
```

# Summary

This script is designed to be flexible and secure, allowing you to manage Akamai access groups easily. It leverages the `.edgerc` file for credential management and uses Python's `argparse` module to accept command-line arguments, making it adaptable to various environments and use cases.