Custom Tooling to Explore AWS

Analyze the challenge

The challenge is asking to create a Python script to enumerate and download files from an S3 bucket. The script should mimic the behavior of the AWS CLI but with more control and the ability to handle different response codes and parse the XML response for directories and files. The final step is to retrieve the MD5 hash of a specific file from the S3 bucket as the flag.

Plan of action

1. Set up the Environment

- **Install Burp Suite Community Edition:** Download and install from the provided link.
- **Install AWS CLI:** Follow the instructions in the provided link to install the AWS CLI for your operating system.
- **Install VS Code with Python and Python Debugger extensions:** Set up VS Code for Python development.
- **Install Jython in Burp Suite:** Add the standalone Jython file as instructed in the guide so that the "Copy as Python-Requests" extension works correctly.
- **Install BApp Store extension "Copy as Python-Requests":** Install this extension in Burp Suite to help generate the Python request code.
- **Configure Burp Proxy:**
  - Enable the Burp proxy to intercept traffic.
  - Configure your terminal to proxy traffic through Burp using the provided `export` commands.
  - Download and export the Burp certificate to handle HTTPS traffic, ensuring to run these commands.

2. Capture and Analyze AWS CLI Requests

- **Run `aws s3 ls s3://dev.challenge.com`:** Execute this command in your terminal while Burp is intercepting.
- **Observe the requests in Burp:** Forward through the requests until you see the one that lists the bucket contents. This request will be used as the basis for our Python script.
- **Send the request to Repeater:** Right-click the relevant request in Burp and select "Send to Repeater."
- **Test the request in Repeater:** Click "Send" in Repeater to view the response, which should list the S3 bucket contents.

3. Create a Basic Python Script

- **Copy the request as Python code:** In Repeater, right-click the request, go to Extensions -> "Copy as Python-Requests", and select "Copy as requests."
- **Paste into VS Code:** Create a new file named `s3enum.py` in VS Code and paste the copied code.
- **Initial cleanup:**

- Replace `burp0_url` with a more descriptive variable name like `bucket_url`.
- Remove unnecessary Burp headers.
- Store the `requests.get()` call in a variable named `response`.
- Add a `print(response)` statement to see the response.

```
import requests

bucket_url = "https://s3.amazonaws.com:443/dev.challenge.com?list-
type=2&prefix=&delimiter=%2F&encoding-type=url"
response = requests.get(bucket_url)
print(response)
```

The initial code should successfully make a request to the S3 bucket and print the response object. This is a good starting point to build upon. We need to check if the response code is being printed as expected and confirm that the request is identical to the one made by the AWS CLI.

4. Refine the Script to Handle Response Codes

- **Add conditional statements:** Introduce `if`, `elif`, and `else` statements to handle different response codes (200, 404, 403).
- **Use `response.status_code`:** Access the HTTP status code using `response.status_code`.
- **Test with different bucket names:** Modify the `bucket_url` to test various scenarios (valid bucket, non-existent bucket, unauthorized bucket).

```
import requests

bucket_url = "https://s3.amazonaws.com/dev.challenge.com?list-
type=2&prefix=&delimiter=%2F&encoding-type=url"
response = requests.get(bucket_url)
print(response)
if response.status_code == 200:
    print('Starting here with valid buckets')
elif response.status_code == 404:
    print('Bucket does not exist.')
elif response.status_code == 403:
    print('We do not have permissions to access this bucket.')
else:
    print('Something is causing us to not be able to access the
bucket.')
```

The code now correctly identifies and prints messages based on the HTTP response code. We need to ensure that the logic correctly distinguishes between a successful response (200), a non-existent bucket (404), and an unauthorized access attempt (403). The `else` statement should handle any other unexpected response codes.

5. Parse the XML Response

- **Import `re` module:** Add `import re` at the beginning of your script to use regular expressions.
- **Print the response text:** Add `print(response.text)` within the `if response.status_code == 200:` block to see the XML response.
- **Extract prefixes (directories):** Use `re.findall()` to find all occurrences of text between `<Prefix>` and `</Prefix>` tags.
- **Handle the root directory:** Add logic to print "/" for empty prefixes, which represent the root directory.

```
import requests
import re

bucket_url = "https://s3.amazonaws.com/dev.challenge.com?list-
type=2&prefix=&delimiter=%2F&encoding-type=url"
response = requests.get(bucket_url)

if response.status_code == 200:
    print('Starting here with valid buckets')
    print(response.text)
    prefixes = re.findall(r'<Prefix>(.*?)</Prefix>', response.text)
    print('Directories discovered:')
    for prefix in prefixes:
        if prefix == '':
            print('/')
        else:
            print(prefix)
elif response.status_code == 404:
    print('Bucket does not exist.')
elif response.status_code == 403:
    print('We do not have permissions to access this bucket.')
else:
    print('Something is causing us to not be able to access the
bucket.')
```

The code now successfully parses the XML response and extracts the directory prefixes. The regular expression correctly identifies the content between the `` tags. The logic for handling the root directory ensures that it's represented by "/" in the output.

6. Enumerate Directories and Files

- **Create `new_url` for subdirectory requests:** Inside the `for prefix in prefixes:` loop, create a `new_url` variable that includes the current `prefix` in the URL.
- **Make requests to `new_url`:** Use `requests.get(new_url)` to get the contents of each subdirectory.
- **Extract keys (files):** Use `re.findall()` to find text between `<Key>` and `</Key>` tags in the response from `new_url`.

- **Print discovered files:** Print each discovered file using `print(f'File Discovered - {key}')`.
- **Handle permissions for subdirectories:** Add `if`, `elif`, `else` statements to handle 403 and other errors when accessing subdirectories.

```
import requests
import re

bucket_url = "https://s3.amazonaws.com/dev.challenge.com?list-
type=2&prefix=&delimiter=%2F&encoding-type=url"
response = requests.get(bucket_url)

if response.status_code == 200:
    prefixes = re.findall(r'<Prefix>(.*?)</Prefix>', response.text)
    print('Directories discovered:')
    for prefix in prefixes:
        if prefix == '':
            print('/')
        else:
            print(prefix)
        new_url = f"https://s3.amazonaws.com/dev.challenge.com?list-
type=2&prefix={prefix}&delimiter=%2F&encoding-type=url"
        response = requests.get(new_url)
        if response.status_code == 200:
            keys = re.findall(r'<Key>(.*?)</Key>', response.text)
            for key in keys:
                print(f'File Discovered - {key}')
        elif response.status_code == 403:
            print('We do not have permissions to access this
directory.')
        else:
            print('Something is causing us to not be able to access
the bucket.')
elif response.status_code == 404:
    print('Bucket does not exist.')
elif response.status_code == 403:
    print('We do not have permissions to access this bucket.')
else:
    print('Something is causing us to not be able to access the
bucket.')
```

The code now correctly enumerates both directories and files within each directory. The `new_url` is constructed correctly to access the contents of each subdirectory. The regular expression for extracting keys works as intended. The error handling for subdirectories ensures that permission issues are reported.

7. Download Files

- **Import `os` and `urllib3` modules:** Add `import os` and `import urllib3` at the beginning of your script.

- **Create `directory_name` variable:** Define a variable to store the name of the directory where files will be downloaded.
- **Disable SSL warnings:** Add `urllib3.disable_warnings(urllib3.exceptions.InsecureRequestWarning)` to suppress warnings related to unverified SSL certificates.
- **Construct `download_url`:** Inside the `for key in keys:` loop, create a `download_url` using the `key`.
- **Make a request to `download_url`:** Use `requests.get(download_url, stream=True, verify=False)` to download the file.
- **Handle 403 and other errors for download:** Wrap the download code in conditional statements to handle errors.
- **Create directory structure:** Use `os.makedirs()` to create the necessary directory structure based on the `key`.
- **Save the file:** Use `with open(file_path, 'wb') as file:` to open the file in binary write mode and write the downloaded data using `file.write(data)`.

```
import requests
import re
import os
import urllib3

urllib3.disable_warnings(urllib3.exceptions.InsecureRequestWarning)

bucket_url = "https://s3.amazonaws.com/dev.challenge.com?list-
type=2&prefix=&delimiter=%2F&encoding-type=url"
directory_name = 'dev.huge-logistics.com'
response = requests.get(bucket_url)

if response.status_code == 200:
    prefixes = re.findall(r'<Prefix>(.*?)</Prefix>', response.text)
    print('Directories discovered:')
    for prefix in prefixes:
        if prefix == '':
            print('/')
        else:
            print(prefix)
        new_url = f"https://s3.amazonaws.com/dev.challenge.com?list-
type=2&prefix={prefix}&delimiter=%2F&encoding-type=url"
        response = requests.get(new_url)
        if response.status_code == 200:
            keys = re.findall(r'<Key>(.*?)</Key>', response.text)
            for key in keys:
                if key == prefix:
                    pass
                else:
                    print(f'File Discovered - {key}')
```

```
                    download_url =
f'https://s3.amazonaws.com/dev.challenge.com/{key}'
                    print(f'Downloading {key}')
                    file_response = requests.get(download_url,
stream=True, verify=False)
                    if file_response.status_code == 200:
                        if not key.endswith('/'):
                            file_path = os.path.join(directory_name,
key)
                            os.makedirs(os.path.dirname(file_path),
exist_ok=True)
                            with open(file_path, 'wb') as file:
                                for data in
file_response.iter_content(1024):
                                    file.write(data)
                                print(f'{key} downloaded.')
        elif response.status_code == 403:
            print('We do not have permissions to access this
directory.')
        else:
            print('Something is causing us to not be able to access
the bucket.')
elif response.status_code == 404:
    print('Bucket does not exist.')
elif response.status_code == 403:
    print('We do not have permissions to access this bucket.')
else:
    print('Something is causing us to not be able to access the
bucket.')
```

The code now successfully downloads files from the S3 bucket and saves them to the specified directory. The `download_url` is constructed correctly. The `os.makedirs()` function ensures that the directory structure is created before saving the file. The error handling for file downloads is in place. The use of `stream=True` and `verify=False` allows for downloading files efficiently and handling potential SSL issues.

8. Retrieve the Flag

- **Run `md5sum` command:** Execute the command `md5sum dev.challenge.com/shared/hl_migration_project.zip` in your terminal to get the MD5 hash of the specified file.

The `md5sum` command should output the correct MD5 hash of the file, which is the flag for the lab. We need to verify that the file was downloaded correctly in the previous step and that the `md5sum` command is executed in the correct directory.

OUTPUT

```
import requests
import re
```

```
import os
import urllib3

urllib3.disable_warnings(urllib3.exceptions.InsecureRequestWarning)

bucket_url = "https://s3.amazonaws.com/dev.challenge.com?list-
type=2&prefix=&delimiter=%2F&encoding-type=url"
directory_name = 'dev.challenge.com'
response = requests.get(bucket_url)

if response.status_code == 200:
    prefixes = re.findall(r'<Prefix>(.*?)</Prefix>', response.text)
    print('Directories discovered:')
    for prefix in prefixes:
        if prefix == '':
            print('/')
        else:
            print(prefix)
        new_url = f"https://s3.amazonaws.com/dev.challenge.com?list-
type=2&prefix={prefix}&delimiter=%2F&encoding-type=url"
        response = requests.get(new_url)
        if response.status_code == 200:
            keys = re.findall(r'<Key>(.*?)</Key>', response.text)
            for key in keys:
                if key == prefix:
                    pass
                else:
                    print(f'File Discovered - {key}')
                    download_url =
f'https://s3.amazonaws.com/dev.challenge.com/{key}'
                    print(f'Downloading {key}')
                    file_response = requests.get(download_url,
stream=True, verify=False)
                    if file_response.status_code == 200:
                        if not key.endswith('/'):
                            file_path = os.path.join(directory_name, key)
                            os.makedirs(os.path.dirname(file_path),
exist_ok=True)
                            with open(file_path, 'wb') as file:
                                for data in
file_response.iter_content(1024):
                                    file.write(data)
                                print(f'{key} downloaded.')
        elif response.status_code == 403:
            print('We do not have permissions to access this directory.')
        else:
            print('Something is causing us to not be able to access the
bucket.')
elif response.status_code == 404:
    print('Bucket does not exist.')
elif response.status_code == 403:
    print('We do not have permissions to access this bucket.')
else:
    print('Something is causing us to not be able to access the bucket.')
```

Flag:

Run the following command in your terminal after running the script:

```
md5sum dev.challenge.com/shared/hl_migration_project.zip
```

The output will be the MD5 hash, which is the flag.