Algorithm for IP File Updates in Python

Google Cybersecurity Professional Certification

Project description

In this project, I utilized Python to manage IP traffic as part of a cybersecurity workflow. This process involved reading and processing an allowlist of IP addresses, identifying and removing IPs flagged for removal, and updating the allowlist accordingly. The solution demonstrated Python's proficiency for file handling, list manipulation and automating network security tasks.

Open the file that contains the allow list

The file that you want to open is called **"allow_list.txt"**. Assign a string containing this file name to the **import_file** variable. Then, use a **with** statement to open it. Use the variable **file** to store the file while you work with it inside the **with** statement.

```python
# Assign `import_file` to the name of the file
import_file = "allow_list.txt"

# Assign `remove_list` to a list of IP addresses that are no longer allowed to access restricted information.
remove_list = ["192.168.97.225", "192.168.158.170", "192.168.201.40", "192.168.58.57"]

# First line of `with` statement
with open(import_file, "r") as file:
```

```python
import_file = "allow_list.txt"
```

- Assigns the name of the file (`allow_list.txt`) containing the allowlist of IPs to the variable `import_file`.
- **Purpose:** Stores the filename in a variable for easy access.

```python
remove_list = [...]
```

- Assigns a list of IP addresses to the variable `remove_list`.
- **Purpose:** Creates a Python list to store IPs that need to be removed.

```python
with open(import_file, "r") as file:
```

- **open() function:** Opens a file. The "r" mode specifies "read-only" access.
- **with statement:** A context manager to manage the file's lifecycle. It ensures that the file is properly closed after operations are completed, even if an error occurs.
- **file variable:** A temporary variable representing the opened file object.

Read the file contents

Next, use the `.read()` method to convert the contents of the allow list file into a string so that you can read them. Store this string in a variable called `ip_addresses`.

```python
# Assign `import_file` to the name of the file

import_file = "allow_list.txt"

# Assign `remove_list` to a list of IP addresses that are no longer allowed to access restricted information.

remove_list = ["192.168.97.225", "192.168.158.170", "192.168.201.40", "192.168.58.57"]

# Build `with` statement to read in the initial contents of the file

with open(import_file, "r") as file:

    # Use `.read()` to read the imported file and store it in a variable named `ip_addresses`

    ip_addresses = file.read()

# Display `ip_addresses`

print(ip_addresses)
```

```
ip_address
192.168.25.60
192.168.205.12
192.168.97.225
192.168.6.9
192.168.52.90
```

`ip_addresses = file.read()`

- This method reads the entire content of the file as a single string.
- **Purpose:** Converts the contents of the allowlist file into a string so it can be processed.

`print(ip_addresses)`

- Displays the content of the file stored in `ip_addresses`.
- Useful for verifying the file's contents during development or debugging.

Convert the string into a list

In order to remove individual IP addresses from the allow list, the IP addresses need to be in a list format. Therefore, use the `.split()` method to convert the `ip_addresses` string into a list.

```python
# Assign `import_file` to the name of the file

import_file = "allow_list.txt"

# Assign `remove_list` to a list of IP addresses that are no longer allowed to access restricted information.

remove_list = ["192.168.97.225", "192.168.158.170", "192.168.201.40", "192.168.58.57"]

# Build `with` statement to read in the initial contents of the file

with open(import_file, "r") as file:

  # Use `.read()` to read the imported file and store it in a variable named `ip_addresses`

  ip_addresses = file.read()

# Use `.split()` to convert `ip_addresses` from a string to a list

ip_addresses = ip_addresses.split()

# Display `ip_addresses`

print(ip_addresses)
```
```
['ip_address', '192.168.25.60', '192.168.205.12', '192.168.97.225', '192.168.6.9', '192.168.52.90', '192.168.158.170', '192.16
8.90.124', '192.168.186.176', '192.168.133.188', '192.168.203.198', '192.168.201.40', '192.168.218.219', '192.168.52.37', '192.
168.156.224', '192.168.60.153', '192.168.58.57', '192.168.69.116']
```

Iterate through the remove list

A second list called `remove_list` contains all of the IP addresses that should be removed from the `ip_addresses` list. Set up the header of a `for` loop that will iterate through the `remove_list`. Use `element` as the loop variable.

```python
# Assign `import_file` to the name of the file

import_file = "allow_list.txt"

# Assign `remove_list` to a list of IP addresses that are no longer allowed to access restricted information.

remove_list = ["192.168.97.225", "192.168.158.170", "192.168.201.40", "192.168.58.57"]

# Build `with` statement to read in the initial contents of the file

with open(import_file, "r") as file:

  # Use `.read()` to read the imported file and store it in a variable named `ip_addresses`

  ip_addresses = file.read()

# Use `.split()` to convert `ip_addresses` from a string to a list

ip_addresses = ip_addresses.split()

# Build iterative statement
# Name loop variable `element`
# Loop through `ip_addresses`

for element in ip_addresses:

    # Display `element` in every iteration

    print(element)
```

```
ip_address
192.168.25.60
192.168.205.12
192.168.97.225
192.168.6.9
192.168.52.90
192 168 158 170
```

Remove IP addresses that are on the remove list

In the body of your iterative statement, add code that will remove all the IP addresses from the allow list that are also on the remove list. First, create a conditional that evaluates if the loop variable `element` is part of the `ip_addresses` list. Then, within that conditional, apply the `.remove()` method to the `ip_addresses` list and remove the IP addresses identified in the loop variable `element`.

```python
# Assign `import_file` to the name of the file

import_file = "allow_list.txt"

# Assign `remove_list` to a list of IP addresses that are no longer allowed to access restricted information.

remove_list = ["192.168.97.225", "192.168.158.170", "192.168.201.40", "192.168.58.57"]

# Build `with` statement to read in the initial contents of the file

with open(import_file, "r") as file:

  # Use `.read()` to read the imported file and store it in a variable named `ip_addresses`

  ip_addresses = file.read()

# Use `.split()` to convert `ip_addresses` from a string to a list

ip_addresses = ip_addresses.split()

# Build iterative statement
# Name loop variable `element`
# Loop through `ip_addresses`

for element in ip_addresses:

  # Build conditional statement
  # If current element is in `remove_list`,

    if element in remove_list:

        # then current element should be removed from `ip_addresses`

        ip_addresses.remove(element)

# Display `ip_addresses`

print(ip_addresses)

['ip_address', '192.168.25.60', '192.168.205.12', '192.168.6.9', '192.168.52.90', '192.168.90.124', '192.168.186.176', '192.16
8.133.188', '192.168.203.198', '192.168.218.219', '192.168.52.37', '192.168.156.224', '192.168.60.153', '192.168.69.116']
```

Update the file with the revised list of IP addresses

Now that you have removed these IP addresses from the `ip_address` variable, you can complete the algorithm by updating the file with this revised list. To do this, you must first convert the `ip_addresses` list back into a string using the `.join()` method. Apply `.join()` to the string `"\n"` in order to separate the elements in the file by placing them on a new line.

Then, use another `with` statement and the `.write()` method to write over the file assigned to the `import_file` variable.

```python
# Assign `import_file` to the name of the file

import_file = "allow_list.txt"

# Assign `remove_list` to a list of IP addresses that are no longer allowed to access restricted information.

remove_list = ["192.168.97.225", "192.168.158.170", "192.168.201.40", "192.168.58.57"]

# Build `with` statement to read in the initial contents of the file

with open(import_file, "r") as file:

  # Use `.read()` to read the imported file and store it in a variable named `ip_addresses`

  ip_addresses = file.read()

# Use `.split()` to convert `ip_addresses` from a string to a list

ip_addresses = ip_addresses.split()

# Build iterative statement
# Name loop variable `element`
# Loop through `ip_addresses`

for element in ip_addresses:

  # Build conditional statement
  # If current element is in `remove_list`,

    if element in remove_list:

        # then current element should be removed from `ip_addresses`

        ip_addresses.remove(element)

# Convert `ip_addresses` back to a string so that it can be written into the text file

ip_addresses = " ".join(ip_addresses)

# Build `with` statement to rewrite the original file

with open(import_file, "w") as file:

  # Rewrite the file, replacing its contents with `ip_addresses`

  file.write(ip_addresses)
```

Bringing it to one function

```python
# Define a function named `update_file` that takes in two parameters: `import_file` and `remove_list`

def update_file(import_file, remove_list):

    # Build `with` statement to read in the initial contents of the file

    with open(import_file, "r") as file:

        # Use `.read()` to read the imported file and store it in a variable named `ip_addresses`

        ip_addresses = file.read()

    # Use `.split()` to convert `ip_addresses` from a string to a list

    ip_addresses = ip_addresses.split()

    # Build iterative statement
    # Name loop variable `element`
    # Loop through `ip_addresses`

    for element in ip_addresses:

        # Build conditional statement
        # If current element is in `remove_list`,

        if element in remove_list:

            # then current element should be removed from `ip_addresses`

            ip_addresses.remove(element)

    # Convert `ip_addresses` back to a string so that it can be written into the text file

    ip_addresses = " ".join(ip_addresses)

    # Build `with` statement to rewrite the original file

    with open(import_file, "w") as file:

        # Rewrite the file, replacing its contents with `ip_addresses`

        file.write(ip_addresses)
```

By defining a function named `update_file(import_file, remove_list)`, it takes in 2 parameters. The first is the name of the text file that contains the IP addresses and the second contains a list of IP addresses to be removed.

When the function is called, it automatically updates the allowed list of IP addresses after passing in the 2 arguments.

Summary

Developed a Python function to automate IP allowlist management for cybersecurity. The function reads the allowlist file, converts its contents into a list, removes specified IP addresses using iteration and conditional logic, and updates the file with the revised list. Key techniques include file handling `with open()`, string manipulation using `.split()` and `" ".join()`, and modular function design for reusability. This project highlights Python's utility in automating essential network security tasks efficiently.