Nasir Ali

Algorithm for file updates in Python

Project description

In this project, I will develop an algorithm that updates a file. There will be two files: one containing a list of IP addresses that have access to the restricted area, and another containing IP addresses that should have their access revoked. My algorithm will remove IP addresses from the allow list if they appear on the remove list.

Open the file that contains the allow list

To open the file ‘allow_list.txt’, which contains the allow list, I first assign the file name as a string and the variable ‘import_file’. I then use ‘with statement along with the open() function to open the file. The With statement manages resources, for example, it opens the file and assigns it to the variable ‘file’. The open() function takes two arguments: the first is the file name(stored in import_file), and the second is the mode - in this case ‘r’ opens the file for reading. The ‘as’ keyword assigns a variable called ‘file’, file stores the output of the open() function.

```
import_file = "allow_list.txt"

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

Read the file contents

A variable named file is used to reference the opened allow_list.txt file. To read its contents, the read() function is used, and the result is stored in a variable named ip_addresses. The read() function reads the entire content of the file as a single string, which can then be processed or printed. This string is assigned to the ip_addresses variable, and the print() function is used to display its contents. Converting the file contents into a string allows us to work with and extract data more easily in later steps.

```
ip_addresses = file.read()
print(ip_addresses)
```

Convert the string into a list

To convert the string into a list, I use split() function which splits the text at each whitespace character by default. To remove individual Ip addresses from the allowed list it must be in list format.

ip_addresses = ip_addresses.split()

['ip_address', '192.168.205.12', '192.168.6.9', '192.168.52.90', '192.168.90.124', '192.168.186.176', '192.168.133.188', '192.168.218.219', '192.168.52.37', '192.168.156.224', '192.168.60.153', '192.168.69.116']

Iterate through the remove list

This for loop iterates through ip_addresse list. Each IP address is temporarily stored in a variable element. If the IP is also found in the remove_list, the condition is met and can be handled accordingly.

```
for element in ip_addresses:
        if element in remove_list:
```

Remove IP addresses that are on the remove list

If an IP address in the ip_addressses list is also found in the remove_list, it should be removed. This can be done using the remove() function with element as its argument. Applying remove function in this is okay because there are no duplicates in the Ip_addresses list.

ip_addresses.remove(element)

Update the file with the revised list of IP addresses

To convert the list of IP addresses back into a string, I use join() function. This joins the list elements into a single string, with a (“ ”) used as the separator. The updated string is then written back to the file using the write() function. The file.write() method is used to override the content of the original file with the new string.

```
ip_addresses = " ".join(ip_addresses)
file.write(ip_addresses)
```

Summary

In conclusion, using a for loop to iterate over the list of IP addresses made it easier to identify those that should no longer have access to the restricted area. As a security analyst, It's important to efficiently update access lists - doing it manually would be time-consuming and error prone.