Activity: Algorithms to sort IP Addresses

This lab was completed using Jupyter Notebook

Introduction

Security logs are often stored in text files. To analyze the security logs in these files, security analysts have to import and parse these files. Python has some functions that come in handy for these tasks, allowing analysts to efficiently access information from text files.

In this lab, you'll practice using functions and other syntax in Python to import and parse text files.

Note: *Have you already completed this lab once?* Due to how Coursera handles files, you will need to reset the `login.txt"` file used in this lab to its original contents if you want to complete this lab more than once. The File contents reset section at the end of this notebook contains code that allows you to reset the `login.txt"` file to its original contents. After you have run the code in that section, you can begin the lab again.

Scenario

In this lab, you're working as a security analyst. You're responsible for preparing a security log file for analysis and creating a text file with IP addresses that are allowed to access restricted information.

Task 1

In this task, you'll import a security log text file and store it as a string to prepare it for analysis.

In Python, a `with` statement is often used in file handling to open a file and then automatically close the file after reading it.

You're given a variable named `import_file` that contains the name of the log file that you want to import. Start by writing the first line of the `with` statement in the following code cell. Use the `open()` function, setting the second parameter to `"r"`. Note that running this code will produce an error because it will only contain the first line of the `with` statement; you'll complete this `with` statement in the task after this.

```
In [3]:
# Assign `import_file` to the name of the text file that contains the
security log file

import_file = "login.txt"
# First line of the `with` statement
```

```
# Use `open()` to import security log file and store it as a string

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

```
 File "<ipython-input-3-59e1e6536a39>", line 7
   with open(import_file,"r") as file:
                                      ^
SyntaxError: unexpected EOF while parsing
```

Task 2

Now, you'll use the `.read()` method to read the imported file, and you'll store the result in a variable named `text`. Afterwards, display the `text` and explore what it contains by running the cell.

In [4]:

```
# Assign `import_file` to the name of the text file that contains the
security log file

import_file = "login.txt"

# The `with` statement
# Use `open()` to import security log file and store it as a string

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

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

 text = file.read()

# Display the contents of `text`

print(text)
```

```
username,ip_address,time,date
tshah,192.168.92.147,15:26:08,2022-05-10
dtanaka,192.168.98.221,9:45:18,2022-05-09
tmitchel,192.168.110.131,14:13:41,2022-05-11
daquino,192.168.168.144,7:02:35,2022-05-08
eraab,192.168.170.243,1:45:14,2022-05-11
```

jlansky,192.168.238.42,1:07:11,2022-05-11
acook,192.168.52.90,9:56:48,2022-05-10
asundara,192.168.58.217,23:17:52,2022-05-12
jclark,192.168.214.49,20:49:00,2022-05-10
cjackson,192.168.247.153,19:36:42,2022-05-12
jclark,192.168.197.247,14:11:04,2022-05-12
apatel,192.168.46.207,17:39:42,2022-05-10
mabadi,192.168.96.244,10:24:43,2022-05-12
iuduike,192.168.131.147,17:50:00,2022-05-11
abellmas,192.168.60.111,13:37:05,2022-05-10
gesparza,192.168.148.80,6:30:14,2022-05-11
cgriffin,192.168.4.157,23:04:05,2022-05-09
alevitsk,192.168.210.228,8:10:43,2022-05-08
eraab,192.168.24.12,11:29:27,2022-05-11
jsoto,192.168.25.60,5:09:21,2022-05-09

Task 3

The output in the previous step is one big string. In this task, you'll explore how you can split the string that contains the entire imported log file into a list of strings, one string per line.

Use the `.split()` method to perform this split and then display the result. Note that displaying `.split()` doesn't change what is stored in the `text` variable. Variable reassignment would be necessary if you want to store the result after splitting.

In [5]:

```
# Assign `import_file` to the name of the text file that contains the
security log file

import_file = "login.txt"

# The `with` statement
# Use `open()` to import security log file and store it as a string

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

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

 text = file.read()
```

```
print(text.split())
```

['username,ip_address,time,date', 'tshah,192.168.92.147,15:26:08,2022-05-10', 'dtanaka,192.168.98.221,9:45:18,2022-05-09', 'tmitchel,192.168.110.131,14:13:41,2022-05-11', 'daquino,192.168.168.144,7:02:35,2022-05-08', 'eraab,192.168.170.243,1:45:14,2022-05-11', 'jlansky,192.168.238.42,1:07:11,2022-05-11', 'acook,192.168.52.90,9:56:48,2022-05-10', 'asundara,192.168.58.217,23:17:52,2022-05-12', 'jclark,192.168.214.49,20:49:00,2022-05-10', 'cjackson,192.168.247.153,19:36:42,2022-05-12', 'jclark,192.168.197.247,14:11:04,2022-05-12', 'apatel,192.168.46.207,17:39:42,2022-05-10', 'mabadi,192.168.96.244,10:24:43,2022-05-12', 'iuduike,192.168.131.147,17:50:00,2022-05-11', 'abellmas,192.168.60.111,13:37:05,2022-05-10', 'gesparza,192.168.148.80,6:30:14,2022-05-11', 'cgriffin,192.168.4.157,23:04:05,2022-05-09', 'alevitsk,192.168.210.228,8:10:43,2022-05-08', 'eraab,192.168.24.12,11:29:27,2022-05-11', 'jsoto,192.168.25.60,5:09:21,2022-05-09']

Question 1

What do you notice about the output before and after using the `.split()` method?

- The code is more organized and readable---parsed.

Task 4

There is a missing entry in the log file. You'll need to account for that by appending it to the log file. You're given the missing entry stored in a variable named `missing_entry`.

Use the `.write()` method and the parameter `"a"` in the `open()` function.

After the portion of the code that writes to the file, another with statement uses the `.read()` method to read the updated file into the `text` variable and then display it.

In [10]:

```
# Assign `import_file` to the name of the text file that contains the
security log file

import_file = "login.txt"

# Assign `missing entry` to a log that was not recorded in the log file

missing_entry = "jrafael,192.168.243.140,4:56:27,2022-05-09"

# Use `open()` to import security log file and store it as a string
# Pass in "a" as the second parameter to indicate that the file is being
opened for appending purposes
```

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

    # Use `.write()` to append `missing_entry` to the log file

    file.write(missing_entry)

# Use `open()` with the parameter "r" to open the security log file for
reading purposes

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

    # Use `.read()` to read in the contents of the log file and store in a
variable named `text`

    text = file.read()

# Display the contents of `text`

print(text)
```

Output:

```
username,ip_address,time,date
tshah,192.168.92.147,15:26:08,2022-05-10
dtanaka,192.168.98.221,9:45:18,2022-05-09
tmitchel,192.168.110.131,14:13:41,2022-05-11
daquino,192.168.168.144,7:02:35,2022-05-08
eraab,192.168.170.243,1:45:14,2022-05-11
jlansky,192.168.238.42,1:07:11,2022-05-11
acook,192.168.52.90,9:56:48,2022-05-10
asundara,192.168.58.217,23:17:52,2022-05-12
jclark,192.168.214.49,20:49:00,2022-05-10
cjackson,192.168.247.153,19:36:42,2022-05-12
jclark,192.168.197.247,14:11:04,2022-05-12
apatel,192.168.46.207,17:39:42,2022-05-10
mabadi,192.168.96.244,10:24:43,2022-05-12
iuduike,192.168.131.147,17:50:00,2022-05-11
abellmas,192.168.60.111,13:37:05,2022-05-10
gesparza,192.168.148.80,6:30:14,2022-05-11
cgriffin,192.168.4.157,23:04:05,2022-05-09
alevitsk,192.168.210.228,8:10:43,2022-05-08
eraab,192.168.24.12,11:29:27,2022-05-11
jsoto,192.168.25.60,5:09:21,2022-05-09
jrafael,192.168.243.140,4:56:27,2022-05-09
```

Question 2

What do you notice about the position of the entry that was added to the log file?

- **The entry was appended, or added to the end of the list.**

Task 5

The next task you're responsible for is creating a text file. This text file should include a list of IP addresses that are allowed to access restricted information. Documenting this in a text file will help you communicate your findings to your security team.

Start by creating a variable named `import_file` that stores the name of the file, which should be `"allow_list.txt"`.

You're also given a variable named `ip_addresses` that stores a string containing the IP addresses that are allowed.

Run the code to display the two variables and explore what they contain. Be sure to replace the `### YOUR CODE HERE ###` with your own code before you run the following cell.

In [ ]:

```
# Assign `import_file` to the name of the text file that you want to create

import_file = "allow_list.txt"

# Assign `ip_addresses` to a list of IP addresses that are allowed to access
the restricted information

ip_addresses = "192.168.218.160 192.168.97.225 192.168.145.158 192.168.108.13
192.168.60.153 192.168.96.200 192.168.247.153 192.168.3.252 192.168.116.187
192.168.15.110 192.168.39.246"

# Display `import_file`

print(import_file)

# Display `ip_addresses`

print(ip_addresses.split())
```

Output:
allow_list.txt

['192.168.218.160', '192.168.97.225', '192.168.145.158', '192.168.108.13', '192.168.60.153', '192.168.96.200', '192.168.247.153', '192.168.3.252', '192.168.116.187', '192.168.15.110', '192.168.39.246']

Task 6

Your next goal is to create a `with` statement in order to write the IP addresses to the text file you created in the previous step.

You'll first open the file using the `"w"` parameter. Then, you'll write the IP addresses to the file.

In [ ]:

```
# Assign `import_file` to the name of the text file that you want to create

import_file = "allow_list.txt"

# Assign `ip_addresses` to a list of IP addresses that are allowed to access the restricted information

ip_addresses = "192.168.218.160 192.168.97.225 192.168.145.158 192.168.108.13 192.168.60.153 192.168.96.200 192.168.247.153 192.168.3.252 192.168.116.187 192.168.15.110 192.168.39.246"

# Create a `with` statement to write to the text file

with open("allow_list.txt", "w") as file:

      file.write(ip_addresses)

print(ip_addresses)
```

Output:
['192.168.218.160', '192.168.97.225', '192.168.145.158', '192.168.108.13', '192.168.60.153', '192.168.96.200', '192.168.247.153', '192.168.3.252', '192.168.116.187', '192.168.15.110', '192.168.39.246']

- These ip addresses are now written to the created allow list of ip addresses.

Task 7

In this final step, you'll complete the code you've been writing up to this point. You'll add code to read the file containing IP addresses.

Complete a `with` statement that reads the text file and stores it in a new variable called `text`.

Afterwards, display the contents of `text` and run the cell to explore the result..

```
In [19]: # Assign `import_file` to the name of the text file that you want to create

         import_file = "allow_list.txt"

         # Assign `ip_addresses` to a list of IP addresses that are allowed to access the restricted information

         ip_addresses = "192.168.218.160 192.168.97.225 192.168.145.158 192.168.108.13 192.168.60.153 192.168.96.200 192.168.

         # Create a `with` statement to write to the text file

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

             # Write `ip_addresses` to the text file

             file.write(ip_addresses)

         # Create a `with` statement to read in the text file

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

             # Read the file and store the result in a variable named `text`

             text = file.read()

         # Display the contents of `text`

         print(text)

         192.168.218.160 192.168.97.225 192.168.145.158 192.168.108.13 192.168.60.153 192.168.96.200 192.168.247.153 192.16
         8.3.252 192.168.116.187 192.168.15.110 192.168.39.246
```

Summary:

The Python lab focused on cybersecurity log file management aimed to empower security analysts with robust skills in importing and parsing security logs for comprehensive analysis. Framed within a cybersecurity context, the scenario placed participants in the role of security analysts entrusted with preparing log files crucial for identifying potential threats and vulnerabilities. The structured tasks guided analysts through crucial steps, starting with the utilization of Python's 'open()' function within a 'with' statement, set in read mode, to securely import a designated log file. Following this, analysts leveraged the '.read()' method to retrieve and encapsulate the log's contents into a string variable named 'text.' This content, representing critical security-related data such as usernames, IP addresses, timestamps, and dates, was then methodically analyzed. Participants applied the '.split()' method to transform the consolidated string into a list of discrete entries, enabling a meticulous examination of each log's details. By mastering these fundamental file handling techniques, analysts fortified their abilities to uncover insights crucial for proactive threat detection and fortifying cybersecurity measures." This revised summary emphasizes the significance of handling security logs in cybersecurity operations, highlighting the pivotal role of analysts in leveraging Python to manage and analyze security-related data for threat identification and proactive cybersecurity measures.