FLEMING

Lab 4: Gateway Log Investigation

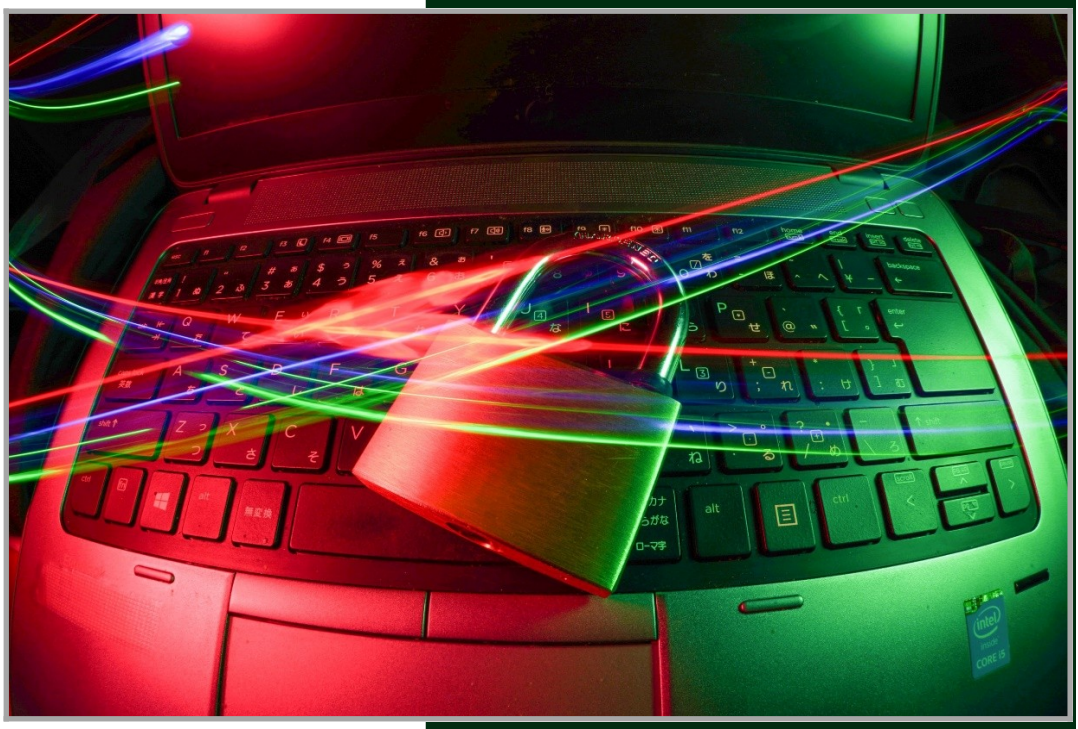


Photo by FLY:D on Unsplash
Free to use under the Unsplash License

COMP 593 - Scripting Applications

TABLE OF CONTENTS

LEARNING OBJECTIVES

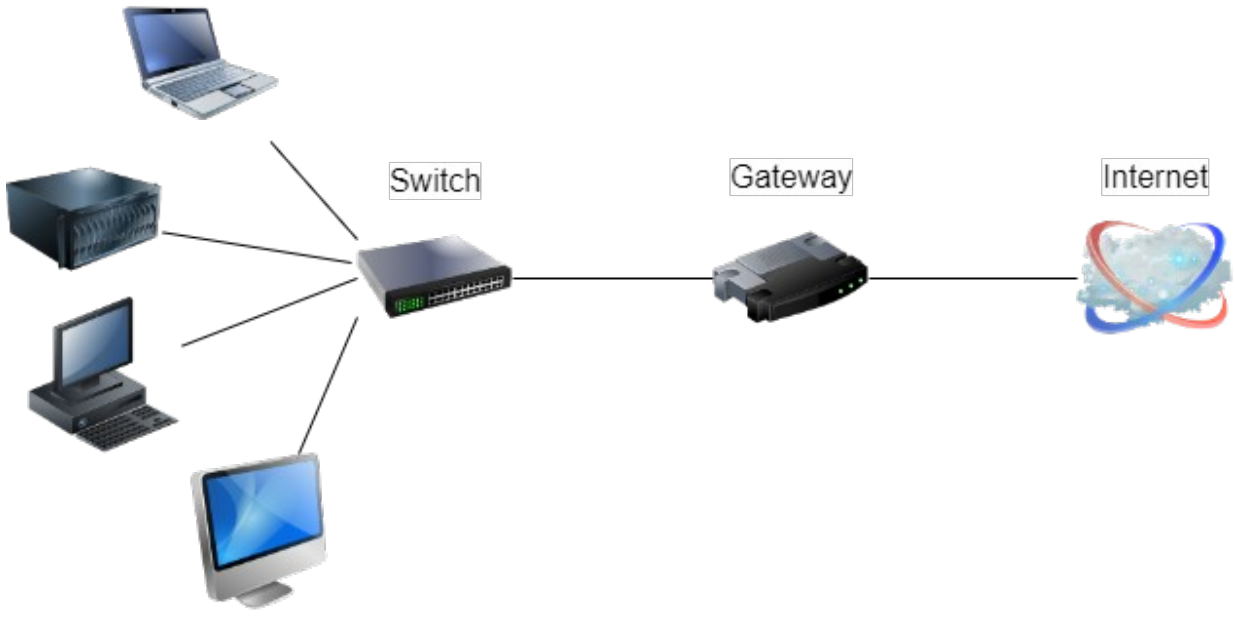
Upon completion of this lab assignment, students should be able to:

- Write a Python script that process a text file line by line
- Incorporate regex matching operations into a Python script using the functions and classes defined in the **re** module, including:
    - Matching text in a string using a regex pattern,
    - Extract text from a string using regex capture groups,
    - Replacing text in a string using a regex pattern to locate the text to be replaced.
- Generate a CSV file using DataFrame class methods from the pandas module.
- Generate a plain text file using DataFrame class methods from the pandas module.
- Discuss the purpose and format of a function docstring.

INTRODUCTION

This lab assignment simulates a computer security investigation that involves developing a Python script to locate and extract information from a log file produced by a network gateway firewall.

A gateway is a node that connects two networks together, serving as an entry and exit point that all data passes through before being routed to its destination. An example network architecture that incorporates a gateway is depicted in the figure below.



A firewall is a network security device that monitors and filters incoming and outgoing network traffic based on an established security policies. Some network gateways have a built-in firewall.

Firewalls can typically be configured to record a log of all network traffic that passes through it, or only traffic that meets some specific criteria, e.g., record all network traffic that is blocked by the firewall. Such a log may be saved as a plain text file, where key information about network traffic is recorded, such as source and destination IP addresses, which can be useful for detecting and tracking malicious attacks on the network from external sources.

Firewall logs may contain a large amount of data, making them impractical to search through manually. For this reason, some type of software tool is typically needed to facilitate a timely and accurate investigation. In some cases, e.g., when the log uses a CSV format, it may be sufficient to open the log file in a spreadsheet application and use its built-in search and filtering capabilities to locate records of interest; however, the log file may not be formatted in a way that allows the spreadsheet application to automatically separate values into individual columns, severely limit its filtering capabilities. Another approach could involve using a log analysis tool, but this would involve learning how to use the tool and may be overkill for a simple log investigation.

Another approach is to write a Python script that processes the log file line by line and uses regular expressions to locate and extract information of interest. This approach provides endless possibilities for text searching and report generation and can be tailored to be as simple or complex as needed for the task at hand. This is the log analysis approach that students will perform for this lab assignment.

INSTRUCTIONS

STEP 1: DOWNLOAD THE GATEWAY FIREWALL LOG

Download the **gateway.log** file from the Module 4 folder in D2L.

This log file includes records from a network firewall. Many of the log records describe dropped packets, but there are also some other records of security interest. Dealing with log files like this is a common task for security administrators and investigators.

Open the log file in a text editor (e.g., VS Code) and inspect the information that is contains. Notice that:

- Records are separated by a new line,
- Records do not all have the same format,
- Records do not all contain the same information,

- Some data items are consistently labeled using an uppercase word followed by an equal sign, e.g., **SRC=**89.49.78.88, and is separated from the next data item by a space,
- The data is not formatted in a way that would allow a spreadsheet application to automatically separate data items into columns (like a CSV file), or for the data to be imported into a pandas DataFrame object, and
- There is too much information for a manual search to be feasible.

Step 2: Get the Script Templates from D2L

Create a new repository, save the script templates into the repository folder, and open the folder in VS Code.

Step 3: Create Function that Gets the Log File Path

Define a function that gets the log file path from the command line parameters. The function must:

- Accept one parameters: The parameter number from which to get the log file path
- Return the log file path input as a command line parameter
- Output an error message and abort script execution if no command line parameter is provided
- Output an error message and abort script execution if the command line parameter is not the path of an existing file

Call this function from **main()**, assigning its return value to a local variable.

Step 4: Create Function that Filters Log Message Records

Define a function that lists all records in a log file that match a specified regex. The function must:

- Accept the following parameters:
  - Log file path
  - Regex
  - Enable/disable case-sensitive regex matching
  - Enable/disable printing records that pass through the filter
  - Enable/disable printing a summary sentence
- Return a list of all records from the log file that match the regex
- If enabled, print each of the records that match the regex

- If enabled, print a summary sentence that indicates how many records in the log file match the regex, and whether case-sensitive or case-insensitive regex matching was performed

STEP 5: INVESTIGATE THE GATEWAY FIREWALL LOG

Bill, the network administrator, believes somebody has been attempting to crack the server by breaking into the secure shell server, so let's use the function created in step 4 to print all records that case-insensitive match the regex **'sshd'**. The script output should be as shown below.

Windows PowerShell

```
PS C:\> python lab4.py gateway.log
Jan 29 04:00:23 myth sshd[10124]: Accepted publickey for root from 192.168.17.8 port
40555 ssh2
Jan 29 04:01:00 myth sshd[10136]: Accepted publickey for root from 192.168.17.8 port
40556 ssh2
...
Jan 29 13:26:58 myth sshd[12502]: Invalid user patriciar from 220.195.35.40
Jan 29 13:27:02 myth sshd[12504]: Invalid user porteria from 220.195.35.40
The log file contains 338 records that case-insensitive match the regex "sshd".
```

It looks like someone is running through a list of common usernames to try to sign in, so let's use the function to print all records that case-insensitive match the regex **'invalid user'**. The script output should be as shown below.

Windows PowerShell

```
PS C:\> python lab4.py gateway.log
Jan 29 13:05:04 myth sshd[11825]: Invalid user anonymous from 220.195.35.40
Jan 29 13:05:10 myth sshd[11827]: Invalid user passwd from 220.195.35.40
...
Jan 29 13:26:58 myth sshd[12502]: Invalid user patriciar from 220.195.35.40
Jan 29 13:27:02 myth sshd[12504]: Invalid user porteria from 220.195.35.40
The log file contains 318 records that case-insensitive match the regex "invalid user".
```

It looks like all those invalid login attempts are coming from the same IP address, but it would be a good idea to confirm that all 318 of them are in fact from the same IP address. Let's check that by printing all records that case-insensitive match the regex **'invalid user.*220.195.35.40'**. The script output should be as shown below.

Windows PowerShell

```
PS C:\> python lab4.py gateway.log
Jan 29 13:05:04 myth sshd[11825]: Invalid user anonymous from 220.195.35.40
Jan 29 13:05:10 myth sshd[11827]: Invalid user passwd from 220.195.35.40
...
Jan 29 13:26:58 myth sshd[12502]: Invalid user patriciar from 220.195.35.40
Jan 29 13:27:02 myth sshd[12504]: Invalid user porteria from 220.195.35.40
The log file contains 318 records that match the regex "Invalid user.*220.195.35.40".
```

That confirms that all invalid login attempts are coming from IP address 220.195.35.40. We should let Bill know about it so the gateway firewall can be configured to block all network traffic from that IP address.

Now let's check if the log contains any error messages by printing all records that case-insensitive match the regex **'error'**. The script output should be as shown below.

Windows PowerShell

```
PS C:\> python lab4.py gateway.log
Jan 29 08:17:05 myth kernel: SFW2-OUT-ERROR IN= OUT=ppp0 SRC=216.58.112.55 DST=77.42.129.124 …
Jan 29 08:17:36 myth kernel: SFW2-OUT-ERROR IN= OUT=ppp0 SRC=216.58.112.55 DST=77.42.129.124 …
...
Jan 29 10:29:11 myth sshd[11035]: error: PAM: Authentication failure for root from
cmbw95.fivefortyfour.com
Jan 29 10:29:22 myth sshd[11035]: error: PAM: Authentication failure for root from
cmbw95.fivefortyfour.com
Jan 29 10:56:05 myth sshd[11136]: error: PAM: Authentication failure for root from
cmbw95.fivefortyfour.com
Jan 29 10:58:59 myth sshd[11161]: error: PAM: Authentication failure for root from
cmbw95.fivefortyfour.com
...
Jan 29 14:18:14 myth kernel: SFW2-OUT-ERROR IN= OUT=ppp0 SRC=216.58.112.55 DST=193.227.243.149 …
Jan 29 14:19:53 myth kernel: SFW2-OUT-ERROR IN= OUT=ppp0 SRC=216.58.112.55 DST=193.227.243.149 …
The log file contains 14 records that case-insensitive match the regex "error".
```

We should let Bill know about those SFW2-OUT-ERROR logs, but they don't seem to be of any security concern.

Those authentication failures look interesting, so let's print all records that case-insensitive match the regex **'pam'**. The script output should be as shown below.

Windows PowerShell

```
PS C:\> python lab4.py gateway.log
Jan 29 10:26:59 myth sshd[11004]: Accepted keyboard-interactive/pam for root from 192.168.17.11 port
3151 ssh2
Jan 29 10:29:11 myth sshd[11035]: error: PAM: Authentication failure for root from
cmbw95.fivefortyfour.com
Jan 29 10:29:22 myth sshd[11035]: error: PAM: Authentication failure for root from
cmbw95.fivefortyfour.com
Jan 29 10:29:28 myth sshd[11035]: Accepted keyboard-interactive/pam for root from 192.168.17.11 port
3152 ssh2
Jan 29 10:50:01 myth sshd[11119]: Accepted keyboard-interactive/pam for root from 192.168.17.11 port
3153 ssh2
Jan 29 10:56:05 myth sshd[11136]: error: PAM: Authentication failure for root from
cmbw95.fivefortyfour.com
Jan 29 10:56:12 myth sshd[11136]: Accepted keyboard-interactive/pam for root from 192.168.17.11 port
3154 ssh2
Jan 29 10:58:59 myth sshd[11161]: error: PAM: Authentication failure for root from
cmbw95.fivefortyfour.com
Jan 29 10:59:04 myth sshd[11161]: Accepted keyboard-interactive/pam for root from 192.168.17.11 port
3156 ssh2
Jan 29 11:02:19 myth sshd[11208]: Accepted keyboard-interactive/pam for root from 192.168.17.11 port
3157 ssh2
The log file contains 10 records that case-insensitive match the regex "pam".
```

Based on those timestamps showing successful logins shortly after the failures, it looks like Bill has a habit of mistyping his password when trying to login from his Windows 95 machine.

Step 6: Move Function into Reusable Module

That function you just created is very useful and very flexible. It could be used to search any plain text file, so let's move it into a separate **.py** file to make it easily reusable for any log investigation. Also, move the function that gets the log file path from the command line to the same .py file, and modify it such that it can get a file path from any command line parameter.

Step 7: Add Capture Group Support to the Function

That function would be even more useful if it supported regex capture groups to extract data from each record. And, if it returns the extracted data as a list of tuples (see example below), it would be in a format that can easily be converted into a **DataFrame** object, which would open up a lot of data analysis possibilities using the pandas package.

Extracted Data Example

Regex with capture groups:

```
'SRC=(.*?) DST=(.*?) LEN=(.*?) '
```

List of tuples:

```
[('24.64.208.134', '216.58.112.55', '512'),  ← Data extracted from first record
 ('24.64.208.134', '216.58.112.55', '512'),  ← Data extracted from second record
 ('24.64.208.134', '216.58.112.55', '512'),  ← Data extracted from third record
...                                          ← Data omitted for brevity
 ('192.168.17.24', '192.168.10.60', '235'),  ← Data extracted from third last
record
 ('192.168.17.24', '192.168.9.51',  '204'),  ← Data extracted from second last
record
 ('192.168.17.24', '192.168.10.60', '204')]  ← Data extracted from last record
```

Modify the function created in step 4 such that it also returns any data extracted using regex capturing groups as a list of tuples.

Step 8: Create Function that Tallies Traffic by Port

Bill wants you to provide him with a report (content described in next step) for each of the destination ports that are experiencing high volumes of network traffic. To do this, you will first have to determine how many records there are in the log file for each destination port. One simple way to do this is to create a dictionary that uses each destination port number as a key, where its respective value is the number of records in the log file that contain that destination port number.

Define a function that processes a log file to create a dictionary of record tallies for each destination port as described above. The function must:

- Accept the log file path as a parameter
- Process the log file line by line tallying the number of records that contain each destination port number (DPT)
- Return a dictionary of destination port number records counts

Step 9: Create Function that Generates Destination Port Reports

Bill wants each report to be a CSV file that contains the following information extracted from each record in the log file that contains a specified destination port number:

- Date
- Time
- Source IP address
- Destination IP address

- Source port number
- Destination port number

For example, the first few rows of the CSV report for destination port number 40686 should look like this (when opened in Excel).

	A	B	C	D	E	F
1	Date	Time	Source IP Address	Destination IP Address	Source Port	Destination Port
2	29-Jan	0:04:42	72.197.8.56	216.58.112.55	9258	40686
3	29-Jan	0:04:42	71.228.199.109	216.58.112.55	37091	40686
4	29-Jan	0:04:42	99.248.20.48	216.58.112.55	48725	40686
5	29-Jan	0:04:42	75.117.31.43	216.58.112.55	3122	40686
6	29-Jan	0:04:42	58.160.93.74	216.58.112.55	24824	40686

Define a function that generates a CSV file containing the information described above for a specified destination port number that is extracted from a specified log file. The function must:

- Accept the following parameters:
  - Log file path
  - Destination port number
- Generate a CSV file containing the information described above
- Save the CSV file in the same directory in which the script resides under the filename **destination_port_{number}_report.csv**, where **{number}** is the destination port number.

Step 10: Generate Destination Port Reports

Bill wants you to provide a separate report for each destination port for which there are 100 or more records in the log file.

Within the **main()** function, use a **for** loop to iterate through the dictionary of destination port number records counts, and for each port having a record count of 100 or more, call the function created in step 9 to generate a report.

Step 11: Create Function that Generates Invalid User Report

Bill also wants a CSV report that contains the following information extracted from the log file that indicates an attempt to login as an invalid user:

- Date
- Time

- Username
- IP address

The first few rows of the CSV report should look like this (when opened in Excel).

	A	B	C	D
1	Date	Time	Username	IP Address
2	29-Jan	13:05:04	anonymous	220.195.35.40
3	29-Jan	13:05:10	passwd	220.195.35.40
4	29-Jan	13:05:14	chuck	220.195.35.40
5	29-Jan	13:05:18	darkman	220.195.35.40
6	29-Jan	13:05:22	hostmaster	220.195.35.40
7	29-Jan	13:05:26	jeffrey	220.195.35.40

Define a function that generates the report described above using information extracted from a specified log file. The function must:

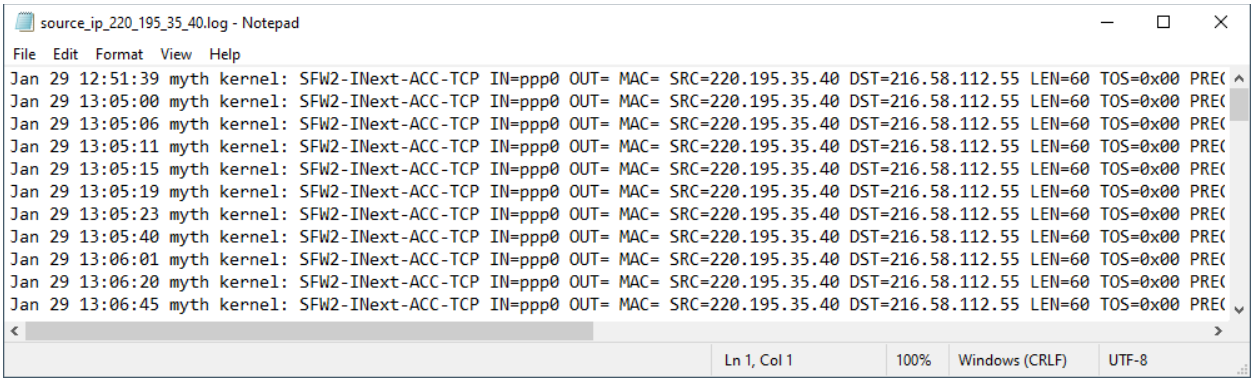
- Accept the log file path as a parameter
- Generate a CSV file containing the information described above
- Save the CSV file in the same directory in which the script resides under the filename **invalid_users.csv**.

Call the function from **main()** to generate the file.

Hint: This function will be very similar to the function created in step 9.

STEP 12: CREATE FUNCTION THAT EXTRACTS AND SAVES SOURCE IP RECORDS

To further investigate the invalid user logins, Bill wants a plain text **.log** file that contains all records from the log file that contain the source (SRC) IP address 220.195.35.40. The first few rows of the plain text report should look like this.



Define a function that generates the plain text **.log** file described above using information extracted from a specified log file. Just in case Bill asks for a similar file for another source IP address in the future, let's make the source IP address a parameter. The function must:

- Accept the following parameters:
  - Log file path
  - Source IP address
- Generate a plain text file containing the information described above
- Save the file in the same directory in which the script resides under the filename **source_ip_{address}.log**, where **{address}** is the source IP address with all periods replaced by underscores, e.g., **source_ip_220_195_35_40.log**.

Call the function from **main()** to generate the file.

Hints:

- *This function will be similar to the function created in step 9.*
- *The first parameter returned by the (extremely useful) function created in steps 4/7 is a list containing each record that matches the specified regex, which is exactly the information that needs to be saved in the* ***.log*** *file.*
- *Since CSV files are plain text files, the* ***to_csv()*** *method of the* ***Dataframe*** *class can be used to save the file. This StackOverflow thread describes one way it can be done.*
- *The* ***re.sub()*** *function from the* ***re*** *module could be used to replace all periods in the source IP address with underscores.*

Dropbox Submission

Submit the URL of the GitHub repository that contains your script and all report files generated by your script, e.g., https://github.com/BobLoblaw/COMP593-Lab5

Note: Files produced by a script should not typically be included in its source repository but including them in the repository for this lab will simplify lab submission and grading.

Assessment

Item	Out Of	Assessment Criteria

GitHub	2	• GitHub repository is public
		• Repository contains
		  o Two .py files
		  o Five .csv files generated by script
		  o One .log file generated by script
		• Repository does not contain __pycache__ folder

Collaborative Portion	2	• Script implemented as described in steps 2-10
Invalid User Report	3	• Function defined and called as described in step 11 • Report contains required information and format • Report filename as required
Source IP Log	3	• Function defined and called as described in step 12 • Report contains required information and format • Report filename as specified
Total:	**10**	