LIGHTHOUSE LAB

CYBERSECURITY

LOG MONITORING WORKFLOW

OLADAPO OLUWAYALE

Content **Page**

1. Executive Summary

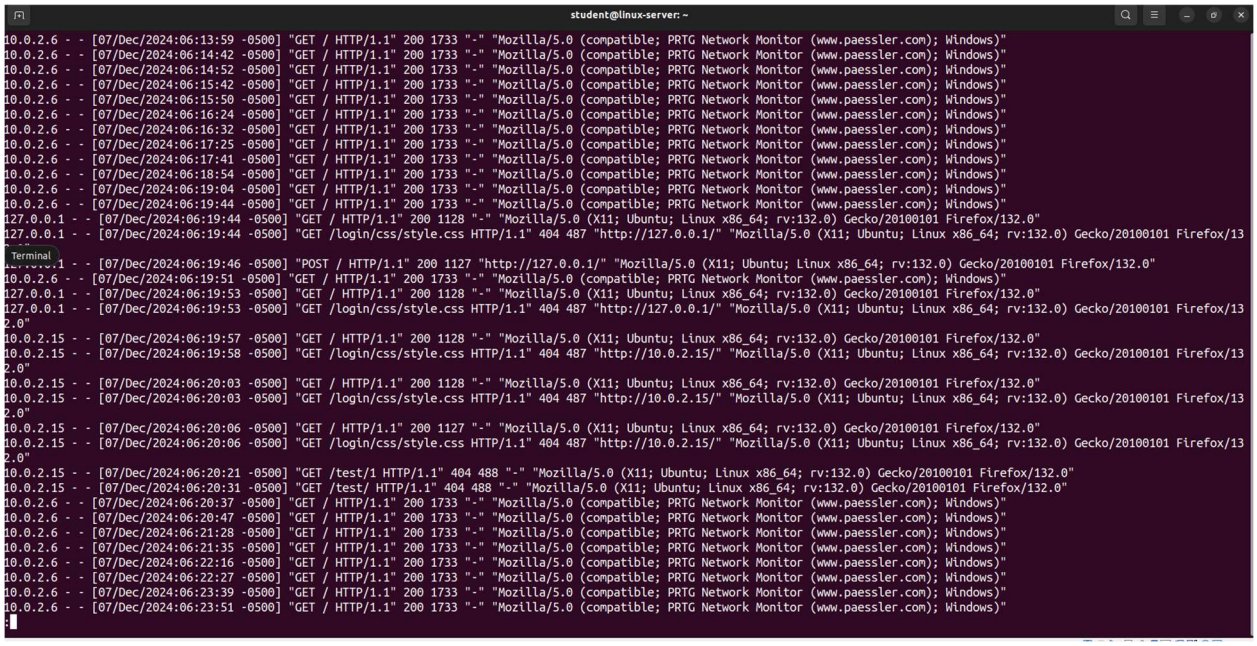
The concept of logging and monitoring isn't new, but organizations still struggle to formulate and implement a security-focused logging and monitoring structure. Security teams need to build logging and monitoring programs that not only collect traditional operational metrics, but are also capable of storing, analyzing, and even mitigating a variety of attacks. (Black Duck, 2024)

This project aims to establish an efficient and automated workflow for monitoring unusual network traffic and failed server access attempts at Turn a New Leaf, a medium-sized non-profit supporting youth employment. As an Access Log Analyst, the task is to create and apply scripting with the use of python and bash for monitoring logs of access activities on both Windows and Linux webservers; it is also imperative to have the logs from the two servers centralized in a shared file. Centralized logging best practices recommend routing log data to a location that is separate from the production environment. This enables IT teams to test and debug issues without impacting business-critical systems. It also prevents hackers from deleting log data and mitigates the risk of losing data in an auto scaled environment (StrongDM, 2024). The objective is to detect anomalies in network traffic, especially failed access attempts that could be an indication of compromise like DDoS and alert management accordingly. Documentation of the process will also be provided weekly.

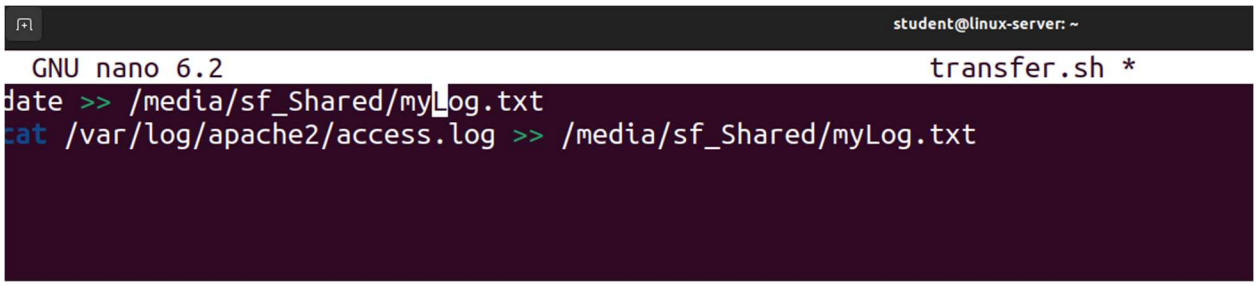
2. Workflow

- Log Access: Connect to Linux to access logs in /var/log/apache2/access.logs and move to shared file on Windows VM to be automated together with that of windows and cronjob used to automate the movement of the log every hour.

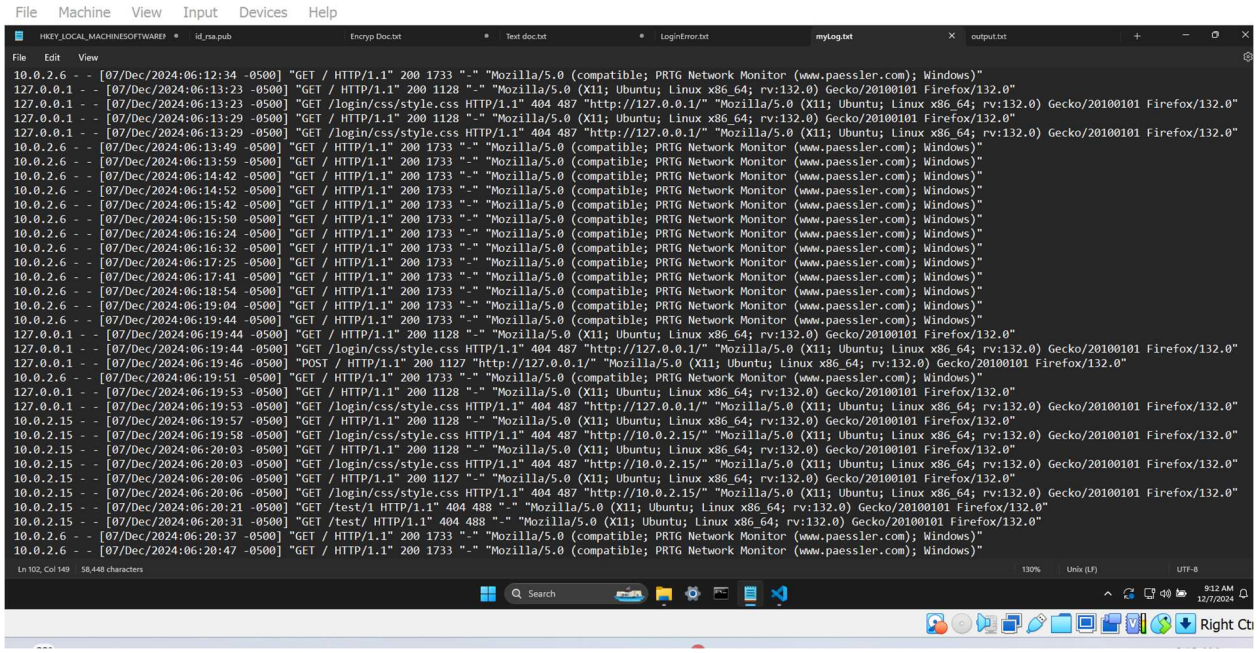
  The screenshot below is the log file in Linux before it was copied to a shared folder with windows for centralization as discussed earlier.



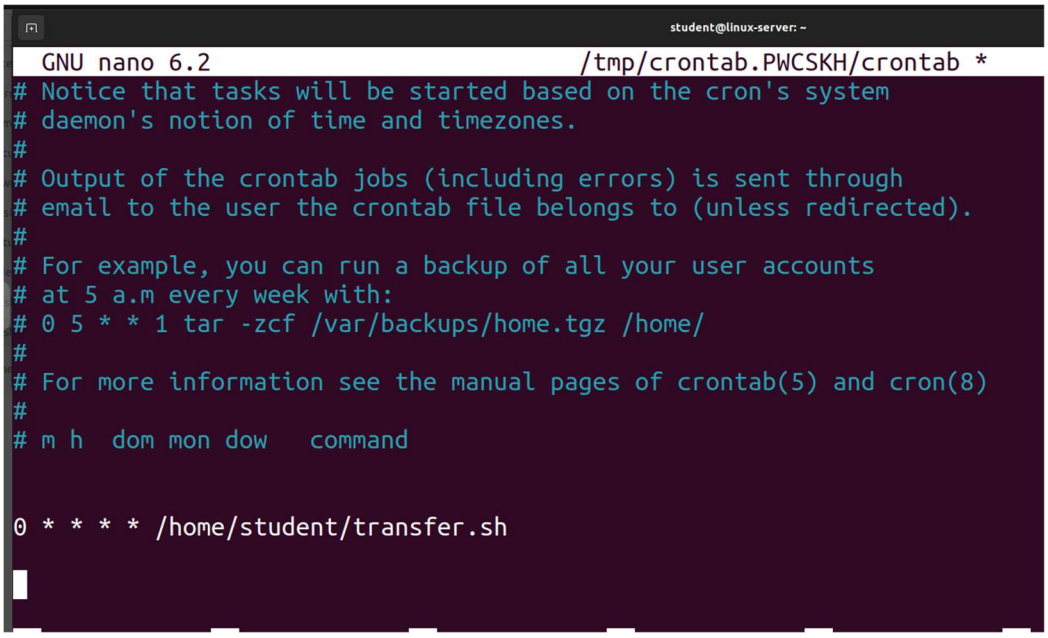
Below is the script in place to have the log file moved to the shared folder.



Below is the log file in the shared file after it had been moved



Below is the cronjob to automate the task of having the log file moved to the shared file every hour instead of having someone do it manually. Managing repetitive tasks using an automated process is a common need for Cyber Security analysts. If you use a Unix-like OS, a Cron Job can save you time by running a task automatically. With Cron, you can automate system maintenance, monitor disk space, and schedule backups. By their nature, Cron Jobs are great for computers that work every day and hour of the week – often, this will be a server. (Compass, n.



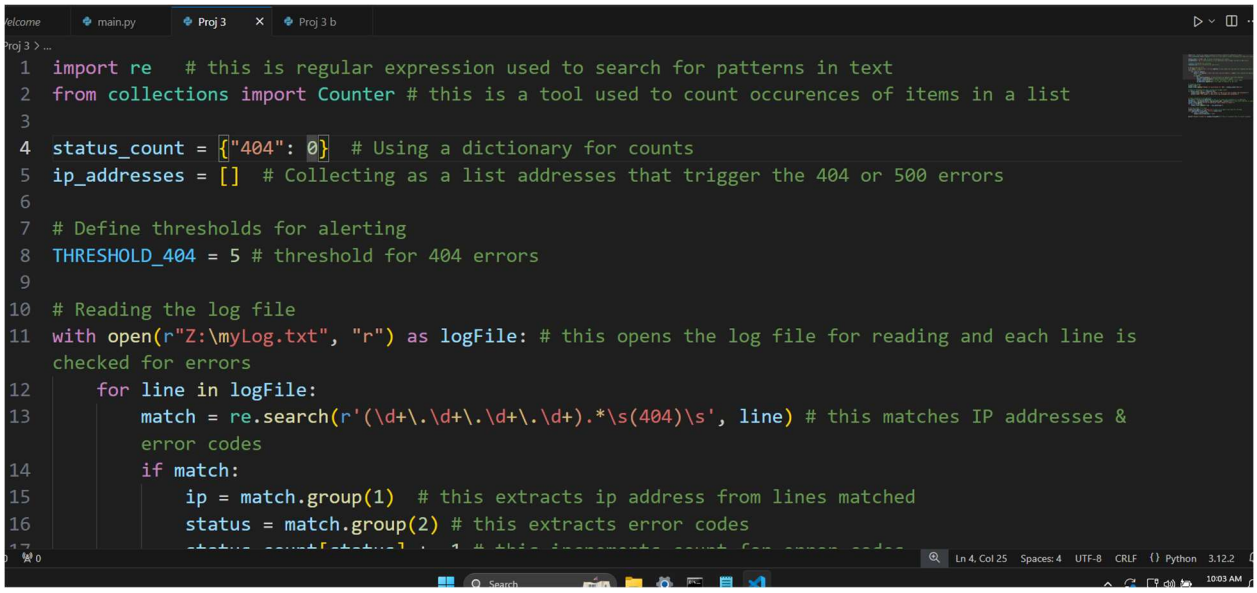
- Monitor for failed webserver attempts: Use Python scripts to identify failed server access attempts and compare them to baseline normal activity.
- Detect Unusual Behavior: Established a threshold of 5 (five) failed attempts for “404”. If the threshold is exceeded, alert is triggered
- Document Findings: Document the number of failed attempts and any other suspicious items identified during monitoring.
- Weekly Report: Summarize findings and email to my manager with insights and any necessary steps to be taken

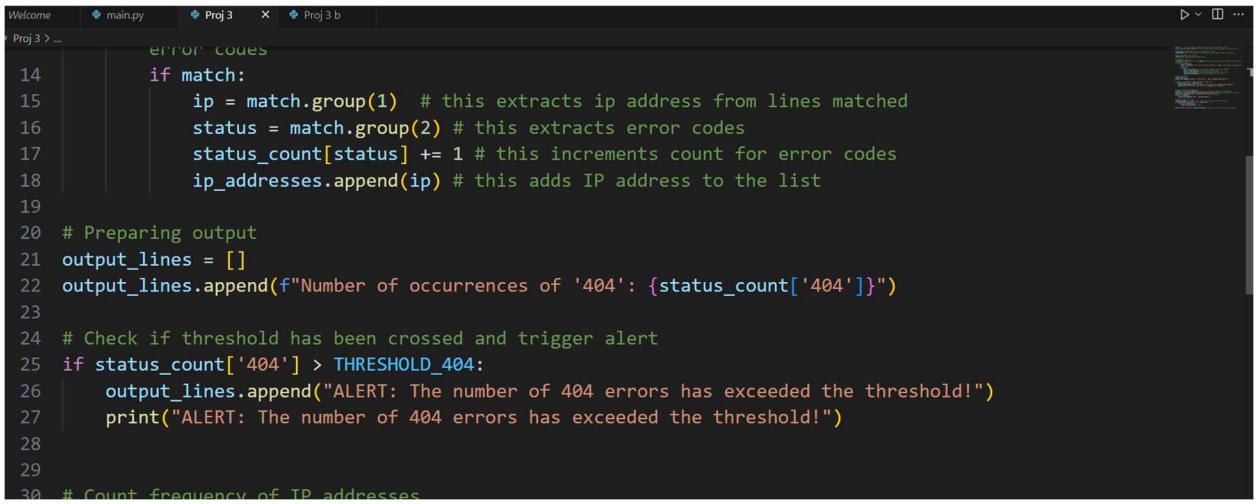
3. Programming Tools and Scripts

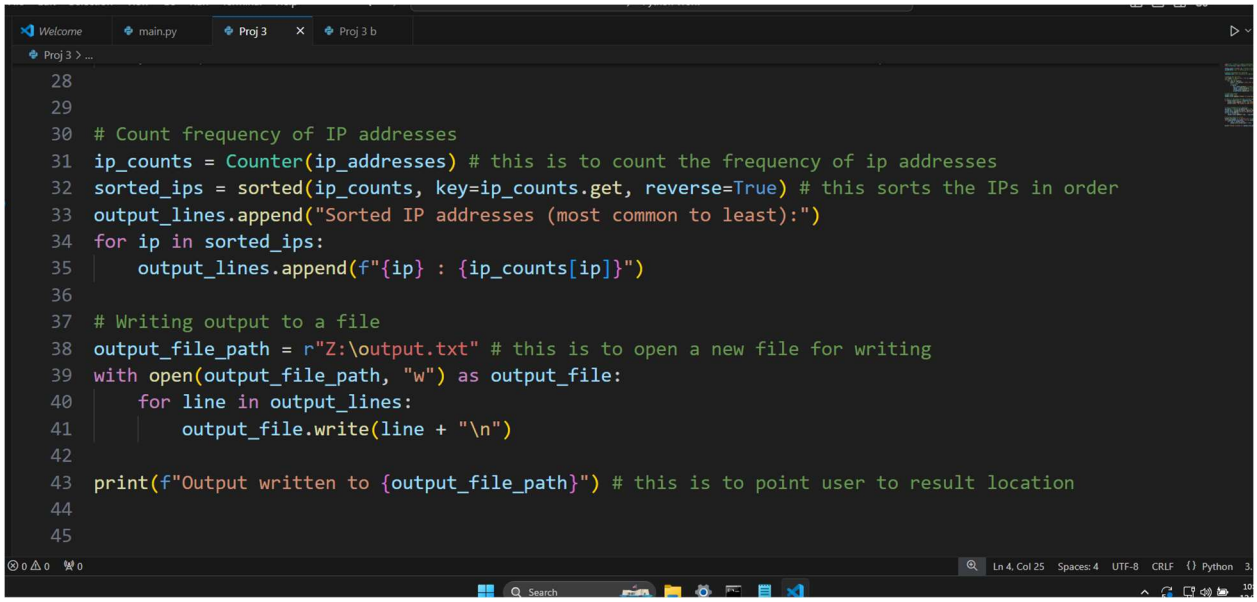
Bash scripting and Python are two different ways to write programs and automate tasks on Linux and Windows systems. Many Linux users learn one or both. Bash scripting is basically a way to put together a series of simple commands, sometimes using loops or decision-making (like "if this happens, do that"). It’s like giving a list of instructions to the system to perform tasks. On the other hand, Python is a more powerful and complete programming language. It’s capable of doing a lot more, from simple tasks like automation to creating full programs that have visual interfaces, like apps you can click on. (LinuxConfig.org, 2020).

3.1 Python Script

The Python script in the screenshot is designed to open and check a centralized log file for “404” error codes, which indicate failed attempts to access the web servers. The script counts how many times the "404" error occurs. If the count exceeds a set threshold of five attempts, it prints an alert saying, “ALERT: The error code '404' has exceeded the threshold!”. Additionally, the script lists the IP addresses associated with these errors, sorting them from the most common to the least common and sends and stores the outcome in a file named output. The expected outcome of the script can be seen in the screenshot provided in the expected outcome section.





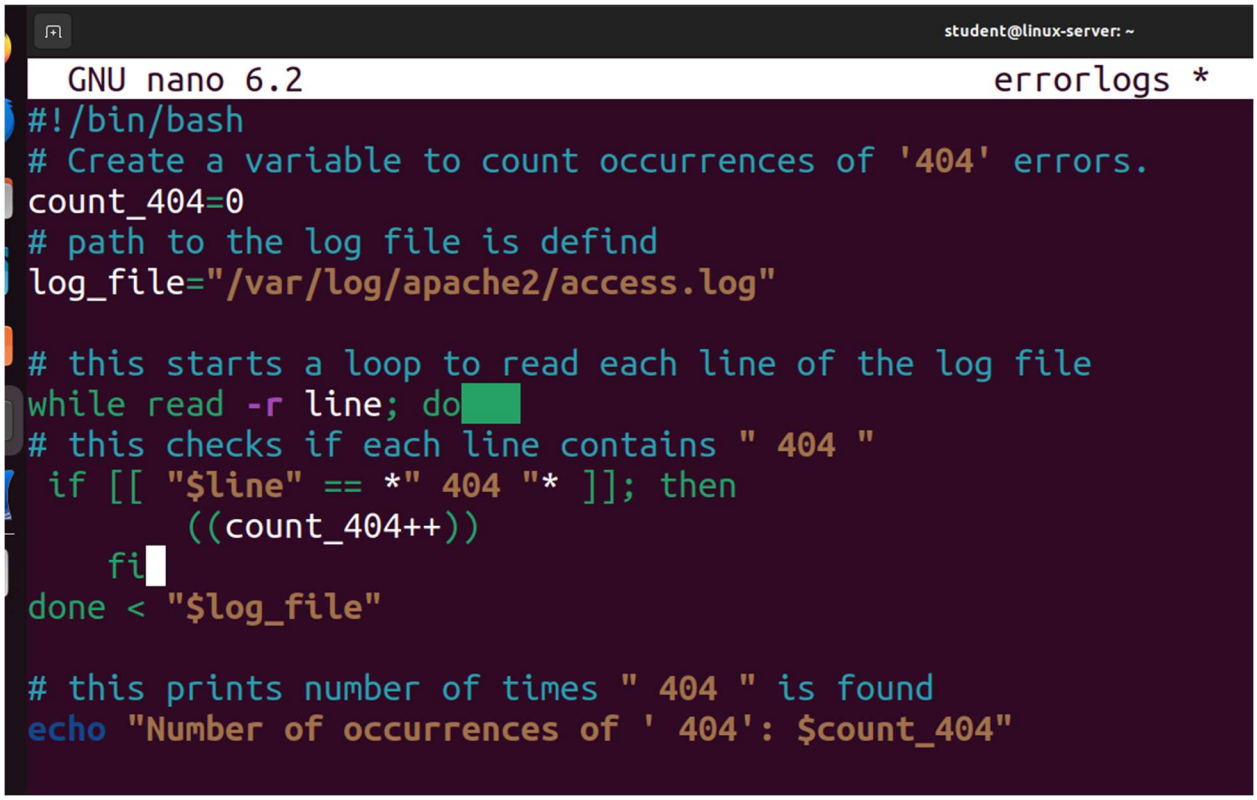


```
28
29
30  # Count frequency of IP addresses
31  ip_counts = Counter(ip_addresses) # this is to count the frequency of ip addresses
32  sorted_ips = sorted(ip_counts, key=ip_counts.get, reverse=True) # this sorts the IPs in order
33  output_lines.append("Sorted IP addresses (most common to least):")
34  for ip in sorted_ips:
35      output_lines.append(f"{ip} : {ip_counts[ip]}")
36
37  # Writing output to a file
38  output_file_path = r"Z:\output.txt" # this is to open a new file for writing
39  with open(output_file_path, "w") as output_file:
40      for line in output_lines:
41          output_file.write(line + "\n")
42
43  print(f"Output written to {output_file_path}") # this is to point user to result location
44
45
```

3.2 Bash Script

This Bash script below counts how many times a "404" error appears in a specific log file, which records server activity. It starts by setting up a counter and selecting the log file (/var/log/apache2/access.log) to check. The script reads through the log file line by line, and each time it finds a line with the "404" error code (which means a page wasn't found), it increases the counter by one. Once the script finishes going through the file, it outputs the total number of "404" errors it found.

In short, the script helps track how many failed attempts (page not found) were made on a server by counting "404" errors in the log file.

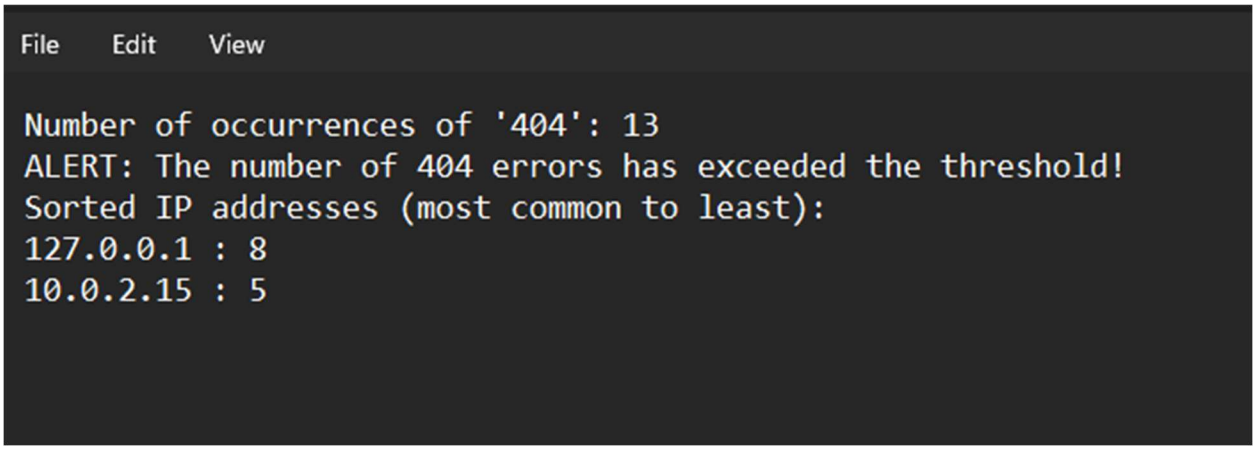


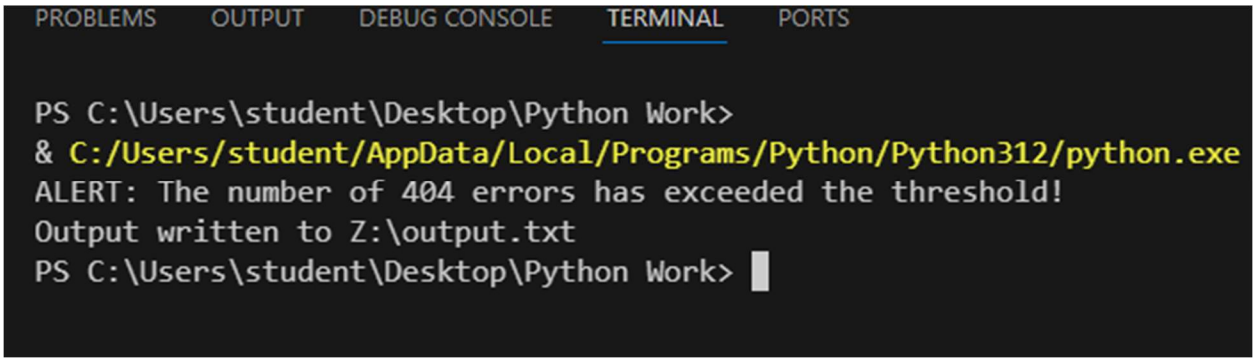
4.0 Expected Outcome

This refers to what the scripts accomplish eventually.

4.1 Python

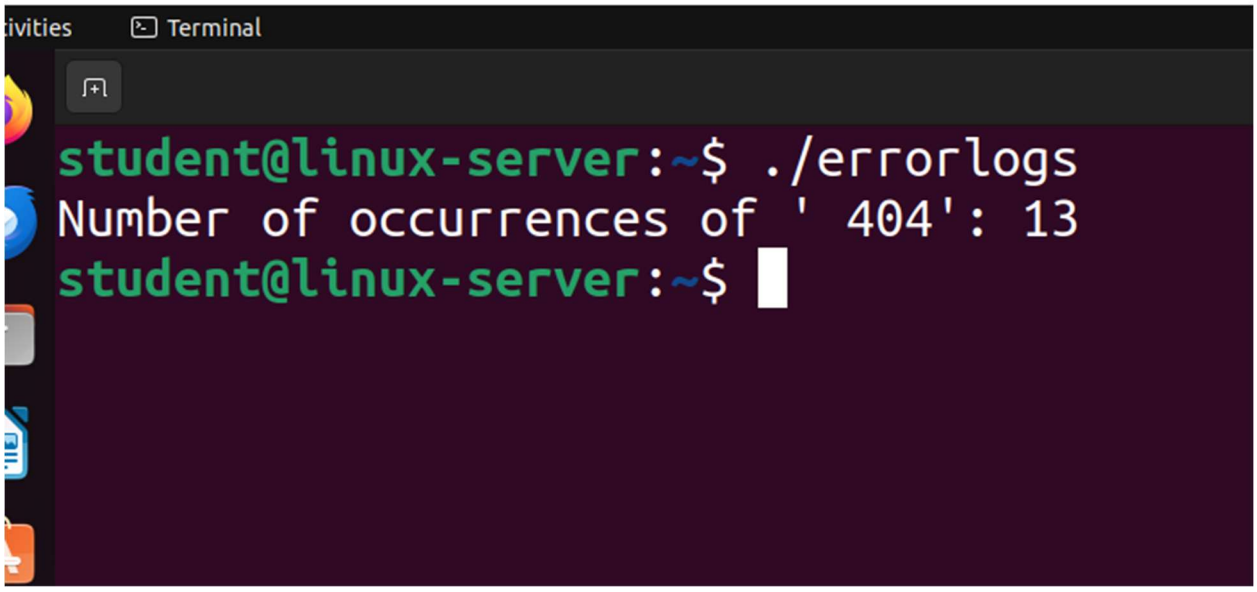
- Triggers alert when threshold are exceeded
- The outcome is written to a designated file Z:\output.txt
- The expected output is the identification of suspicious IP addresses, error codes and their frequency.





4.2 Bash

Below is the outcome of the bash script which counts a total of 13 ‘404’ error code which is consistent with the amount count by the python script



5. **Documentation**
   - Log Documentation: The output id logged into a file with timestamp every time the script runs,
   - Weekly Reports: A weekly email summarizing the logs of failed attempts and all other unusual activity is compiled

6. **Unusual Behavior**
   - Repeated access failures from the same IP address indicate a possible DDoS attack
   - Unusually high number of failed attempts within a short period of time

7. **Potential Iterations**
   - An SMTP server needs to send email alerts which would make operations seamless, and response quick as opposed to sending emails manually.
   - Learning & Development is essential like studying common attack patterns to better detection skills

Reference

Black Duck. (2024, December 6). *Logging and monitoring best practices*. Black Duck Software. https://www.blackduck.com/blog/logging-and-monitoring-best-practices.html

StrongDM. (2024, December 6). *Log management best practices*. StrongDM. https://www.strongdm.com/blog/log-management-best-practices

Compass. (n.d.). *Day 5: Activities*. Lighthouse Labs. https://web.compass.lighthouselabs.ca/p/cyber/days/w03d5/activities/2939

LinuxConfig.org. (2020, November 6). *Bash scripting vs Python*. LinuxConfig.org. https://linuxconfig.org/bash-scripting-vs-python

Microsoft. (2024). *Visual Studio Code* (Version 1.76) [Computer software]. https://code.visualstudio.com/

Ubuntu. (2022). *Ubuntu* (Version 22.04 LTS) [Operating system]. Canonical. https://ubuntu.com/

Microsoft. (2021). *Windows 11* [Operating system]. Microsoft. https://www.microsoft.com/windows