

Turn A New Leaf

TABLE OF CONTENTS

INTRODUCTION.....1

METHODOLOGY.....1

WORKFLOW.....1

PROGRAMMING1-5

 1.a. Bash Scripting1

 1.b. Python Scripting3

 1.c. Corn Job.....4

 1.d. Task Scheduler.....5

EXPECTED OUTPUT.....5

UNUSUAL BEHAVIOR.....5

POTENTIAL ITERATIONS6

REFERENCES6

Introduction:

In response to the management's request to monitor network logs for unusual activity at Turn a New Leaf, it's essential to establish an efficient workflow combined with programming tools to streamline the process. As an Access Log Analyst, the primary objective is to detect any abnormal network traffic, particularly focusing on failed login attempts, and provide timely documentation for managerial review. This comprehensive approach aims to ensure compliance with regulatory requirements and maintain the integrity of the organization's network infrastructure.

Methodology:

This workflow involves a systematic approach to monitor the network logs of Turn a new leaf Organization. The following steps outline the methodology used for this workflow.

- Reviewed the project description to understand the objectives. Identified the need to monitor access logs for unusual network traffic and generate alerts for failed logins.
- Designed a straightforward workflow to monitor access logs on a weekly basis. Specified that monitoring will occur every Thursday to coincide with the company's login requirements.
- Selected basic programming tools such as Bash scripting for log analysis. Developed a simple Bash script to grep access logs for failed login attempts and 404 errors.
- Expected the script to generate alerts if the number of failed logins attempts or 404 errors exceeds predefined thresholds. Defined thresholds for flagging unusual network traffic, such as more than 6 failed login attempts per day.
- Routine reviews of our monitoring process will occur to refine criteria, optimize script performance, and implement additional security measures as needed. Feedback and ongoing training will drive continuous enhancement of our workflow.

Workflow:

The log monitoring process will be conducted on a weekly basis, specifically every Thursday. This timing aligns with the company's requirement to provide a weekly update on network traffic anomalies. We will monitor logs from both Windows and Linux machines within the organization's network. Additionally, logs from the two web servers will be included in the monitoring process.

For Linux machines: Bash scripting will be employed to parse and extract relevant information from the access logs.

For Windows machines: Python scripting will be utilized to parse Windows event logs and extract pertinent data related to login attempts.

We will primarily look for unusual patterns in network traffic, paying special attention to detecting too many failed login attempts. We'll set thresholds for what's considered normal based on past data and industry standards. If the number of failed login attempts goes beyond these thresholds, we'll send an alert.

Programming:

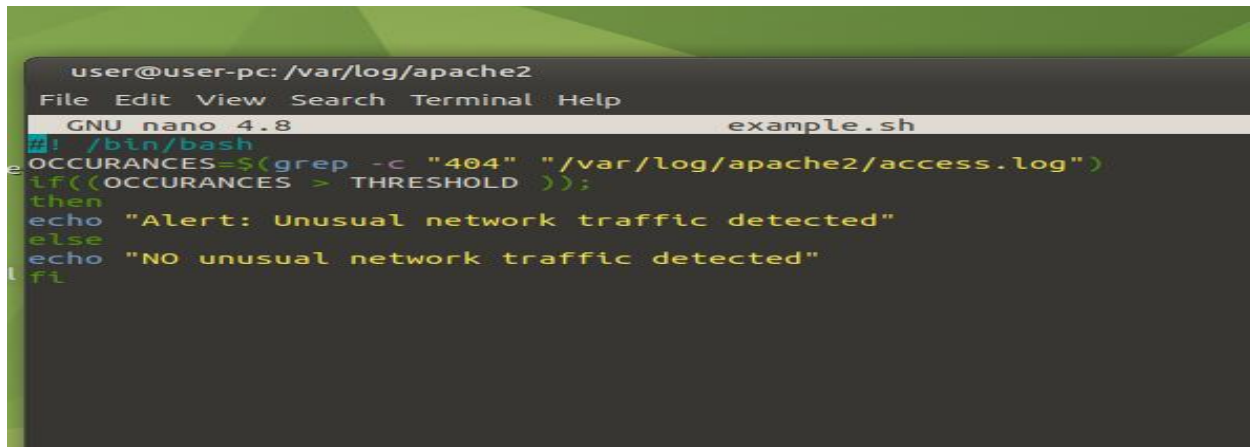
In this section, we outline the tools, languages, and scripts that will be used to implement the monitoring and alerting system for detecting unusual network traffic.

1.a. Bash Scripting:

Bash scripting will be utilized for its simplicity and efficiency in automating command-line tasks. We use Bash scripting to parse access logs on Linux machines. The following key commands and scripts will be employed:

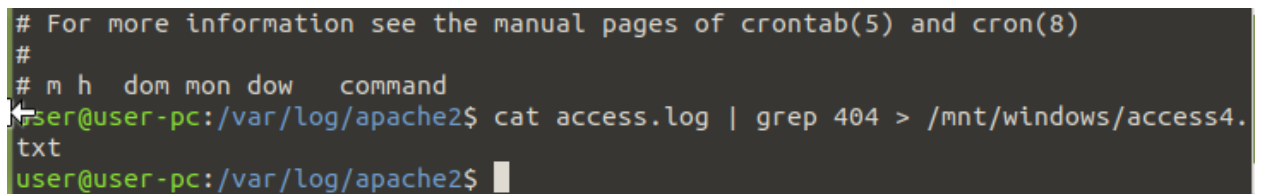
- grep: Used to search for specific patterns in log files, such as failed login attempts and HTTP status codes.

Custom Bash scripts will be developed to automate repetitive tasks and streamline the monitoring process.



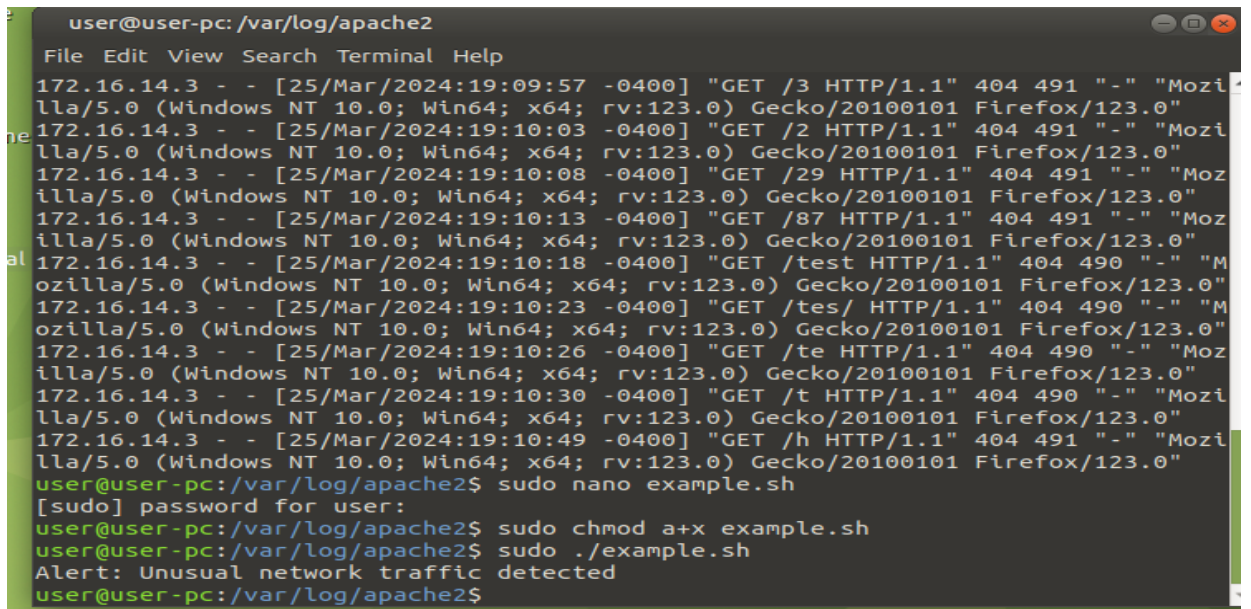
```
user@user-pc: /var/log/apache2
File Edit View Search Terminal Help
GNU nano 4.8 example.sh
#!/bin/bash
OCCURANCES=$(grep -c "404" "/var/log/apache2/access.log")
if((OCCURANCES > THRESHOLD ));
then
echo "Alert: Unusual network traffic detected"
else
echo "NO unusual network traffic detected"
fi
```

Fig 1: Bash script to check the unusual network traffic.



```
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h dom mon dow   command
user@user-pc:/var/log/apache2$ cat access.log | grep 404 > /mnt/windows/access4.txt
user@user-pc:/var/log/apache2$
```

Fig 2: Using Grep command to get 404 errors

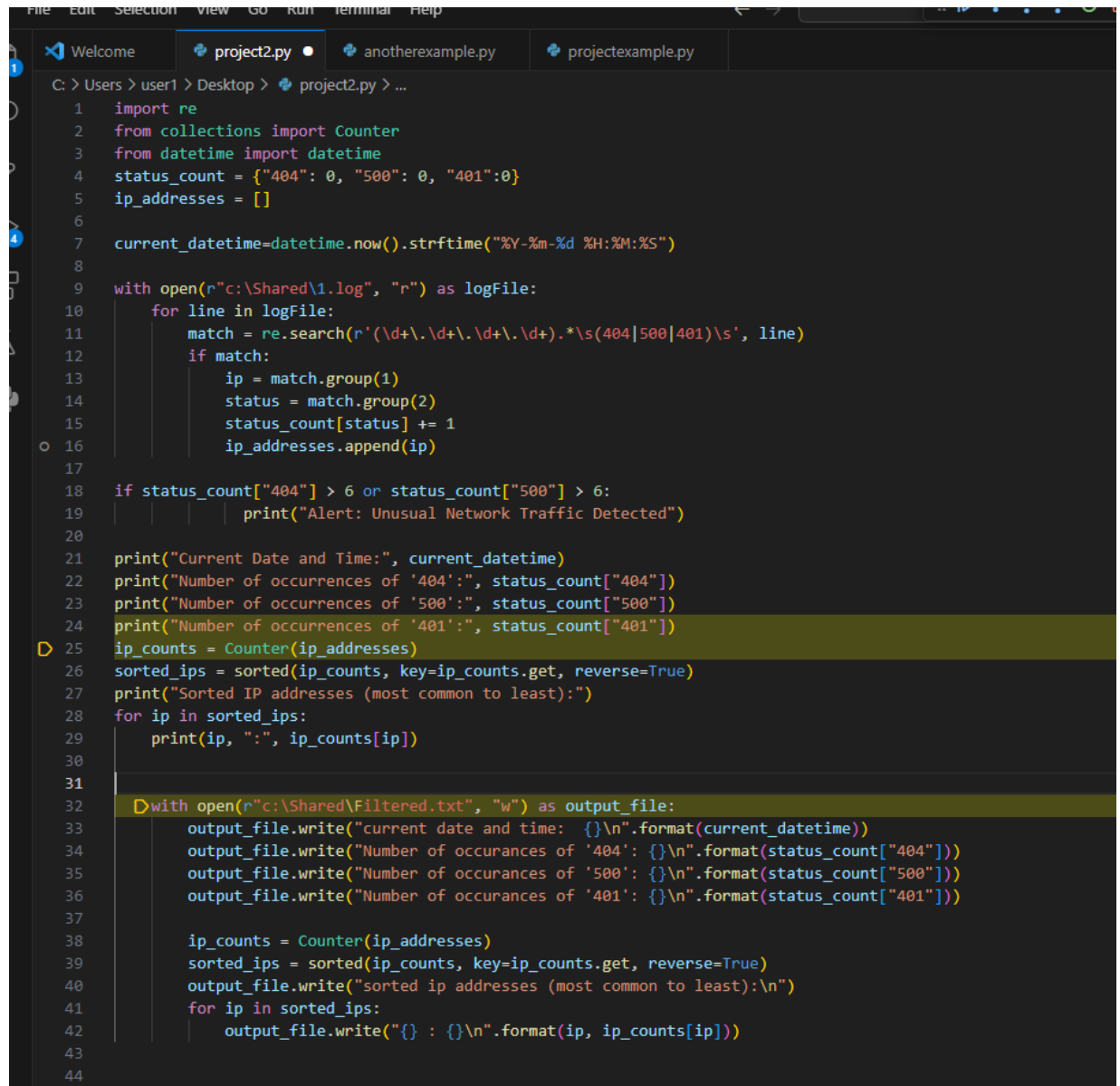


```
user@user-pc: /var/log/apache2
File Edit View Search Terminal Help
172.16.14.3 - - [25/Mar/2024:19:09:57 -0400] "GET /3 HTTP/1.1" 404 491 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:123.0) Gecko/20100101 Firefox/123.0"
172.16.14.3 - - [25/Mar/2024:19:10:03 -0400] "GET /2 HTTP/1.1" 404 491 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:123.0) Gecko/20100101 Firefox/123.0"
172.16.14.3 - - [25/Mar/2024:19:10:08 -0400] "GET /29 HTTP/1.1" 404 491 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:123.0) Gecko/20100101 Firefox/123.0"
172.16.14.3 - - [25/Mar/2024:19:10:13 -0400] "GET /87 HTTP/1.1" 404 491 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:123.0) Gecko/20100101 Firefox/123.0"
172.16.14.3 - - [25/Mar/2024:19:10:18 -0400] "GET /test HTTP/1.1" 404 490 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:123.0) Gecko/20100101 Firefox/123.0"
172.16.14.3 - - [25/Mar/2024:19:10:23 -0400] "GET /tes/ HTTP/1.1" 404 490 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:123.0) Gecko/20100101 Firefox/123.0"
172.16.14.3 - - [25/Mar/2024:19:10:26 -0400] "GET /te HTTP/1.1" 404 490 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:123.0) Gecko/20100101 Firefox/123.0"
172.16.14.3 - - [25/Mar/2024:19:10:30 -0400] "GET /t HTTP/1.1" 404 490 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:123.0) Gecko/20100101 Firefox/123.0"
172.16.14.3 - - [25/Mar/2024:19:10:49 -0400] "GET /h HTTP/1.1" 404 491 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:123.0) Gecko/20100101 Firefox/123.0"
user@user-pc:/var/log/apache2$ sudo nano example.sh
[sudo] password for user:
user@user-pc:/var/log/apache2$ sudo chmod a+x example.sh
user@user-pc:/var/log/apache2$ sudo ./example.sh
Alert: Unusual network traffic detected
user@user-pc:/var/log/apache2$
```

Fig 3: The output of the bash script.

1.b. Python Scripting:

Developed a Python script to parse access logs on Windows machines and perform analysis for unusual network traffic. Python may be employed for more complex log analysis and additional functionality.



```

1  import re
2  from collections import Counter
3  from datetime import datetime
4  status_count = {"404": 0, "500": 0, "401": 0}
5  ip_addresses = []
6
7  current_datetime=datetime.now().strftime("%Y-%m-%d %H:%M:%S")
8
9  with open(r"c:\Shared\l.log", "r") as logFile:
10     for line in logFile:
11         match = re.search(r'(\d+\.\d+\.\d+\.\d+).*s(404|500|401)s', line)
12         if match:
13             ip = match.group(1)
14             status = match.group(2)
15             status_count[status] += 1
16             ip_addresses.append(ip)
17
18     if status_count["404"] > 6 or status_count["500"] > 6:
19         print("Alert: Unusual Network Traffic Detected")
20
21     print("Current Date and Time:", current_datetime)
22     print("Number of occurrences of '404':", status_count["404"])
23     print("Number of occurrences of '500':", status_count["500"])
24     print("Number of occurrences of '401':", status_count["401"])
25     ip_counts = Counter(ip_addresses)
26     sorted_ips = sorted(ip_counts, key=ip_counts.get, reverse=True)
27     print("Sorted IP addresses (most common to least):")
28     for ip in sorted_ips:
29         print(ip, ":", ip_counts[ip])
30
31
32     with open(r"c:\Shared\Filtered.txt", "w") as output_file:
33         output_file.write("current date and time: {}\n".format(current_datetime))
34         output_file.write("Number of occurrences of '404': {}\n".format(status_count["404"]))
35         output_file.write("Number of occurrences of '500': {}\n".format(status_count["500"]))
36         output_file.write("Number of occurrences of '401': {}\n".format(status_count["401"]))
37
38         ip_counts = Counter(ip_addresses)
39         sorted_ips = sorted(ip_counts, key=ip_counts.get, reverse=True)
40         output_file.write("sorted ip addresses (most common to least):\n")
41         for ip in sorted_ips:
42             output_file.write("{} : {}\n".format(ip, ip_counts[ip]))
43
44

```

Fig 4: The python script to look for 404, 500, 401 error codes.

```
File Edit Format View Help
current date and time: 2024-03-25 18:43:02
Number of occurrences of '404': 9
Number of occurrences of '500': 0
Number of occurrences of '401': 0
sorted ip addresses (most common to least):
172.16.14.3 : 9
```

Fig 5: The output on the alerts including the Ip addresses.

1.c. Cron Job:

Scheduled the Bash script using cron jobs for automated execution on Linux machines.

Each task to run has to be defined through a single line indicating with different fields when the task will be run and what command to run for the task

To define the time you can provide concrete values for minute (m), hour (h), day of month (dom), month (mon), and day of week (dow) or use '*' in these fields (for 'any').

Notice that tasks will be started based on the cron's system daemon's notion of time and timezones.

Output of the crontab jobs (including errors) is sent through email to the user the crontab file belongs to (unless redirected).

For example, you can run a backup of all your user accounts at 5 a.m every week with:

```
0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
```

```
/5 * * * * /home/user/move_Log.sh
```

For more information see the manual pages of crontab(5) and cron(8)

Fig 6: The automated corn tab.

1.d. Task Scheduler:

Schedule the Python script using Task Scheduler for automated execution on Windows machines.

General	Triggers	Actions	Conditions	Settings	History (disabled)
Name:	project2				
Location:	\				
Author:	POD1156\user1				
Description:	This task runs a Python script to analyze network traffic logs and generate a report. It counts occurrences of HTTP status codes 404, 500 and 401, identifies potentially unusual traffic patterns, and logs the results to a file.				

Fig 7: Using Windows Task Scheduler to run the code every week.

Expected output:

In this section, we outline the anticipated results and outputs of the monitoring and analysis efforts, focusing on the processing of access logs from our Linux machine hosting the webserver, identification of specific HTTP status codes (401, 404, or 500), and generation of alerts or warnings in case of potential security threats.

The expected output is the retrieval of access logs from our Linux machine hosting the webserver at regular intervals, approximately every five minutes. These logs will be transferred to a more secure shared folder for processing and analysis. Upon retrieval, the program will filter through the access logs to extract information pertaining to specific HTTP status codes: 401 (Unauthorized), 404 (Not Found), or 500 (Internal Server Error).

Logs containing these status codes will be isolated for further analysis, while irrelevant entries will be discarded. The program will analyze the filtered logs to detect patterns indicative of potential security threats or attacks. In case of anomalous behavior or a significant number of occurrences of the specified status codes, alerts or warnings will be generated.

Unusual Behavior:

Spike in Failed Login Attempts: An unusual increase in the number of failed login attempts, especially during non-peak hours or from unexpected IP addresses, could indicate potential brute-force attacks or unauthorized access attempts. These attempts will be flagged for immediate investigation and potential mitigation measures.

Surge in 404 Errors: A sudden surge in HTTP 404 errors, indicating "Page Not Found" responses, may signify attempts to access non-existent resources or pages on the web servers. While occasional 404 errors are common, a significant increase in their frequency or concentration within a short period could suggest probing or reconnaissance activities by malicious actors.

Anomalous Traffic Patterns: Any deviations from normal traffic patterns, such as a significant increase in traffic from specific IP ranges or unusual user-agent strings not associated with legitimate user activity, will be considered as potential indicators of suspicious behavior. These anomalies will be closely monitored and investigated further to determine their nature and potential threat level.

By actively monitoring these indicators of unusual behavior in the access logs, we aim to promptly detect and respond to any potential security incidents, thereby enhancing the overall cybersecurity posture of Turn a New Leaf.

Potential Iterations:

Dedicate time to continuously improve skills in log analysis, scripting, and cybersecurity best practices. Stay updated with the latest advancements in cybersecurity technologies and methodologies through training, certifications, and participation in relevant industry events and communities. This ongoing learning process will enable us to adapt to evolving threats and effectively mitigate security risks.

Explore opportunities to enhance automated response mechanisms for detected security incidents. This may include the development of automated scripts or playbooks to execute predefined response actions, such as blocking suspicious IP addresses, updating firewall rules, or initiating incident response procedures. By automating response actions, we can reduce response times and mitigate the impact of security incidents more effectively.

By incorporating these potential iterations into the monitoring and alerting workflow, we can continually enhance the effectiveness and efficiency of our cybersecurity practices, ensuring robust protection of Turn a New Leaf's network infrastructure against emerging threats.

References:

<https://www.suprsend.com/post/how-to-send-email-notifications-using-python-with-code-examples-> create email alerts using python

<https://www.freecodecamp.org/news/bash-scripting-tutorial-linux-shell-script-and-command-line-for-beginners/>- bash scripting basics

[https://www.netwitness.com/blog/an-introduction-to-siem-integrations/#:~:text=Security%20Information%20and%20Event%20Management%20\(SIEM\)%20integrations%20are%20an%20essential,response%20when%20a%20breach%20occurs.-](https://www.netwitness.com/blog/an-introduction-to-siem-integrations/#:~:text=Security%20Information%20and%20Event%20Management%20(SIEM)%20integrations%20are%20an%20essential,response%20when%20a%20breach%20occurs.-) Integration with SIEM

https://en.wikipedia.org/wiki/HTTP_404- Understanding 404 errors

https://en.wikipedia.org/wiki/List_of_HTTP_status_codes#500- Understanding 500 error

<https://developer.mozilla.org/en-US/docs/Web/HTTP/Status/401-> Understanding 401 error

<https://www.hostinger.com/tutorials/cron-job-> Understanding cron job