Log Monitoring Workflow

Turn a New Leaf

By Heather Fincati

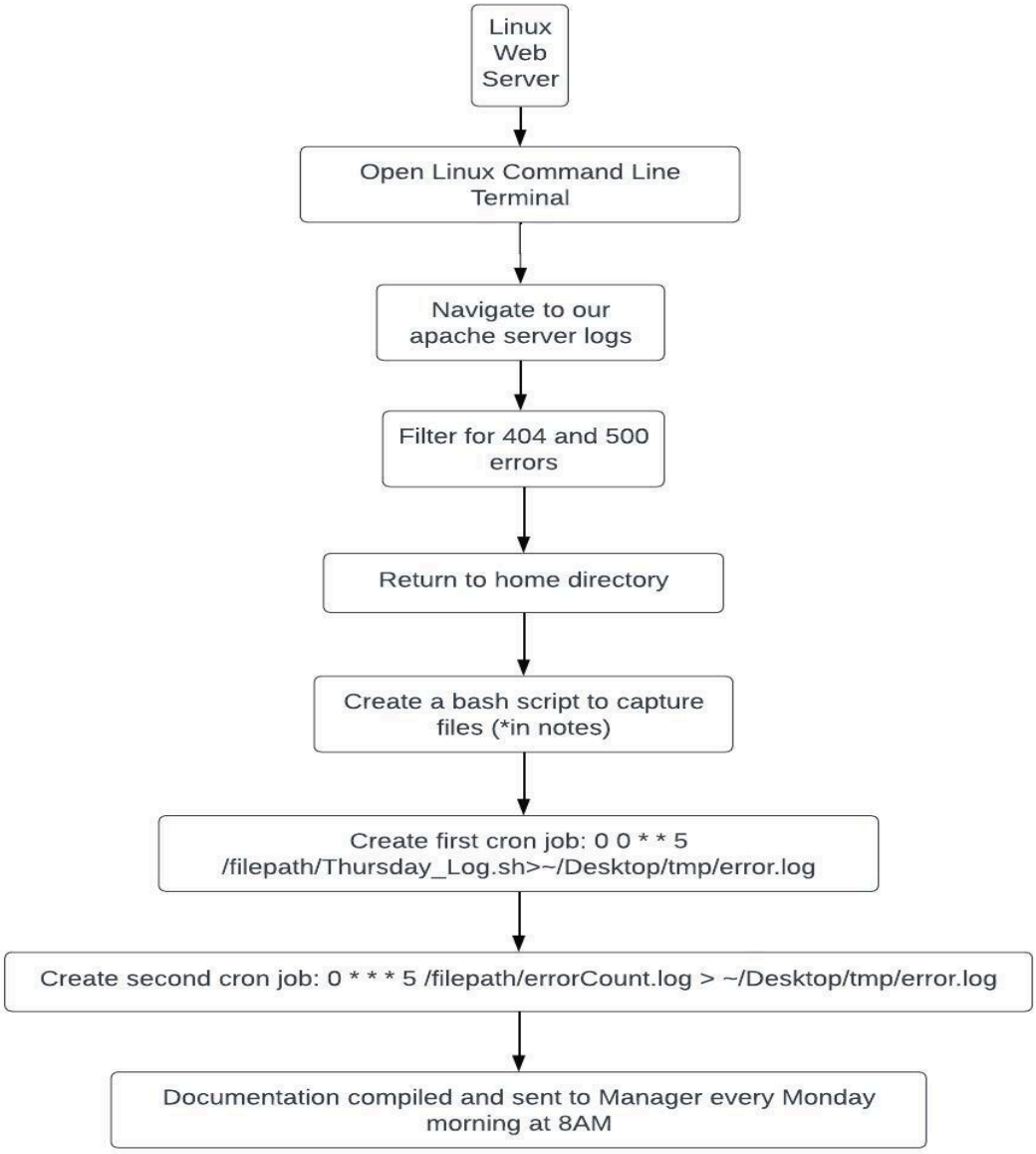
February 26, 2024

Table of Contents

Executive Summary

This report details the planned implementation of ongoing web server log monitoring at Turn a New Leaf. Its primary objectives are to identify potential security vulnerabilities, track user activity trends, and optimize server performance. By creating an automated monitoring system we can make this process as efficient as possible. And to do this we will be employing a tailored selection of tools specific to each web server's operating system (Windows or Linux), this initiative will generate valuable insights to strengthen both server security and user experience. We created a workflow to better explain our goals, the tools we used, and the steps we took to perform the network monitoring as requested.

Workflow


Linux Web Server
Open Linux Command Line Terminal
Navigate to our apache server logs
Filter for 404 and 500 errors
Return to home directory
Create a bash script to capture files (*in notes)
Create first cron job: 0 0 * * 5 /filepath/Thursday_Log.sh>~/Desktop/tmp/error.log
Create second cron job: 0 * * * 5 /filepath/errorCount.log > ~/Desktop/tmp/error.log
Documentation compiled and sent to Manager every Monday morning at 8AM

Programming and Expected Output

Because there is a large amount of data from the log files to go through we wrote scripts to par the information. For our Windows web server, we created our Python script in VS Code. We wanted to filter for IP addresses and the occurrences of the amount of 404 and 500 errors occurred in total and for each IP address, see script and outcome below:

```
> import re ···
  status_count = {"404": 0, "500": 0}

  ip_addresses = []
  with open(r"c:\shared\apache_logs.txt", "r") as logfile:
      for line in logfile:
          match = re.search(r'(\d+\.\d+\.\d+\.\d+).*\s(404|500)\s', line)
          if match:
              ip = match.group(1)
              status = match.group(2)
              status_count[status] += 1
              ip_addresses.append(ip)
  print("Number of occurrences of '404':", status_count["404"])
  print("Number of occurrences of '500':", status_count["500"])
  ip_counts = Counter(ip_addresses)
  sorted_ips = sorted(ip_counts, key=ip_counts.get, reverse=True)
  print("Sorted IP addresses (most common to least):")
  for ip in sorted_ips:
      print(ip, ":", ip_counts[ip])
```

The outcome:

```
Number of occurrences of '404': 213
Number of occurrences of '500': 3
Sorted IP addresses (most common to least):
208.91.156.11 : 60
144.76.95.39 : 14
66.249.73.135 : 10
91.236.75.25 : 8
75.97.9.59 : 6
176.92.75.62 : 5
```

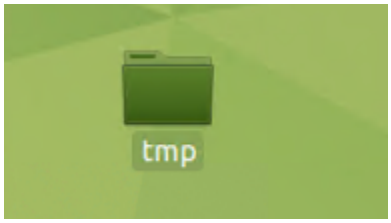
We then used the command line terminal to create a Bash script* with Regular Expressions for our Linux web server. The code we created is:

```
cat ~/Desktop/tmp/apache_logs.txt | grep -Eo '^([0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}) - -
\[([^]]+)\] "GET\s+\/([^"]+)" (4[0-9]{2}|5[0-9]{2})'
```

Expected outcome:

```
188.165.243.45 - - [20/May/2015:02:05:18 +0000] "GET /admin.php HTTP/1.1" 404
208.91.156.11 - - [20/May/2015:03:05:54 +0000] "GET /files/logstash/logstash-1.3
.2-monolithic.jar HTTP/1.1" 404
69.175.14.230 - - [20/May/2015:03:05:13 +0000] "GET /wp-login.php?action=registe
r HTTP/1.0" 404
208.91.156.11 - - [20/May/2015:04:05:43 +0000] "GET /files/logstash/logstash-1.3
.2-monolithic.jar HTTP/1.1" 404
208.115.113.88 - - [20/May/2015:05:05:01 +0000] "GET /blog/geekery/jquery-i**ter
face-puffer.html HTTP/1.1" 404
208.91.156.11 - - [20/May/2015:06:05:37 +0000] "GET /files/logstash/logstash-1.3
.2-monolithic.jar HTTP/1.1" 404
```

We saved directed it to go to our Desktop:



Documentation

In order to streamline the documentation process we have created this command as another way to filter out the log file into a CSV file that could be open by Excel or LibreOffice and sent to our Manager in a report:

cat ~/Desktop/tmp/apache_logs.txt | grep -Eo '^([0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}) - -\[([^]]+)\] "GET\s+\/([^"]+)" (3[0-9]{2}|4[0-9]{2}|5[0-9]{2})' | awk '{print $1","$4","$5","$6" "$7","$8","$9}' > errors_20240224-1724.csv

Unusual Behaviour

We have a baseline of 5 failed login attempts by the same IP address so anything over that would be considered unusual behaviour. This is especially true if conducted within seconds of each other as that is not humanly possible. And would also be considered an indicator of compromise. Seeing the 404 error code is also very unusual as this means there were attempts to reach the URL and it was invalid and unreachable. We would need to look further into these behaviours to see if there are bad actors at play here and the potential of an attack either in progress or already completed.

Potential Iterations

I would streamline my workflow even further to create an even more efficient process by automating more tasks. If I knew Python better I would be able to write a script to automate the email going to my manager each week.

Citations

Google Gemini

1:https://gemini.google.com/app/199005bce5972d1d

I wanted to find a professional technical report format so I used google gemini to help me.

Crontab Guru

2:https://crontab.guru/#0_0_*_*_5

I used Crontab Guru to create the formula for both of the cron jobs used for monitoring in Linux.

Lucidchart

3:https://lucid.app/lucidchart/2645957f-86d4-4c1e-b3ae-3df968f72492/edit?invitationId=inv_179c40fe-ce51-4b8b-bb84-654b8ca2db13&page=0_0#

I used Lucid chart to create a flowchart of our workflow to make it look cleaner and easier to read.

Linux Command Line

4:https://drive.google.com/file/d/1VJa_LGtTaZmOy9H4unzVqFZFCB_CC14B/view

I used the Linux Command Line document to help me find the right commands, such as; grep, cat, and mkdir and also to further understand their proper uses.