Turn A New Leaf Log Monitoring

Table of Contents

Executive Summary

Turn a New Leaf is a medium-sized non-profit organization that supports youth in a range of rural communities to seek employment. In order to support government regulatory requirements, their members must log into the company system every Thursday to confirm or update their employment status, and input any updates on their job searches, including links to job listings they are actively pursuing.

Turn a new leaf has requested that the logs be monitored for any unusual traffic and to send an alert for unusual failed logins. They've also asked for documentation and a weekly update by email.

WorkFlow

1. Monitoring Logs
2. Analyze Logs
3. Determining Escalation
4. Weekly update? If yes send the documentation via email go back to 1.
5. Weekly update? If not, continue back to 1

Step 1.Monitoring Logs.

Monitoring the traffic logs specifically the error 400 and 500 error code series. The 400 error series, namely the 401-403 error codes, can be used to check if the user's silent is connecting with the server, so when a user account tries to access the server, if it fails the user would not be able to access the server and would return an error 401-403 code. The error 500 series ranging from 500-599 will help monitor the traffic to the server to help protect against an overflow of network traffic to the server. Filtering and monitoring for both these error types will help a log analyst detect possible attacks on the user accounts or the server itself. Using programming through bash scripts, and python programming techniques will help for monitoring the logs we will be analyzing.

Step 2. Analyze Logs.

Analyzing the filtered logs from step 1, helps log analysts look for any IoC's (Indicators of Compromise). Setting a certain threshold for what would be considered an IoC will help determine what should be escalated to a manager. For example, setting a threshold of 3 failed attempts from a user account that has been exceeded could indicate an attack, or that a user forgot their password. Either way it should be escalated to a manager as a possible IoC.

Step 3. Determining Escalation.

After analyzing the logs from steps 1 and 2, determining whether or not to escalate will be the next step. Certain investigations can be performed to help determine the escalation type. For example, if there are more 401- 403 error codes originating from the same IP address that could be an IoC (Indicator of Compromise), but further investigation to whether or not that IP address has always been used or if its different from what has been recently used could be an indicator of a possible attacker trying to breach a users account. If the IP address is different from what is commonly used then it could be an IoC in which case further escalation would be required.

Step 5 & 6. Weekly Update.

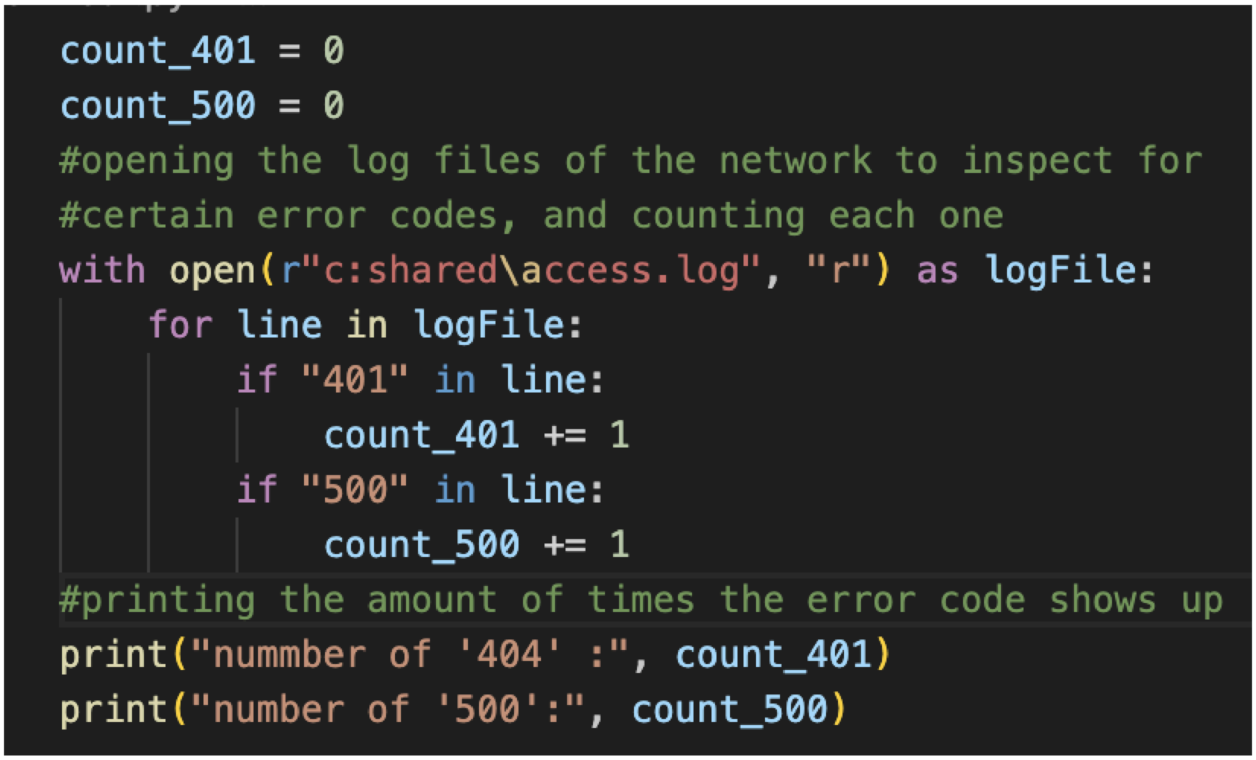
After we have determined whether or not to escalate, the process will start over again. If further escalation has been determined, then sending a weekly update on Friday's, the day after all the server traffic to be accumulated and then analyzed. After the company expands and grows in size, the addition of automation will help keep up with the additional work flow. As well as the larger the company grows the proportional time should be allocated to analyzing the error log files. After the escalation process is done, then log analysts may return to step 1 of the workflow.

Programming

For the programming aspect of things, first we need to go to the machine and used for to monitor the network and insert the following code:

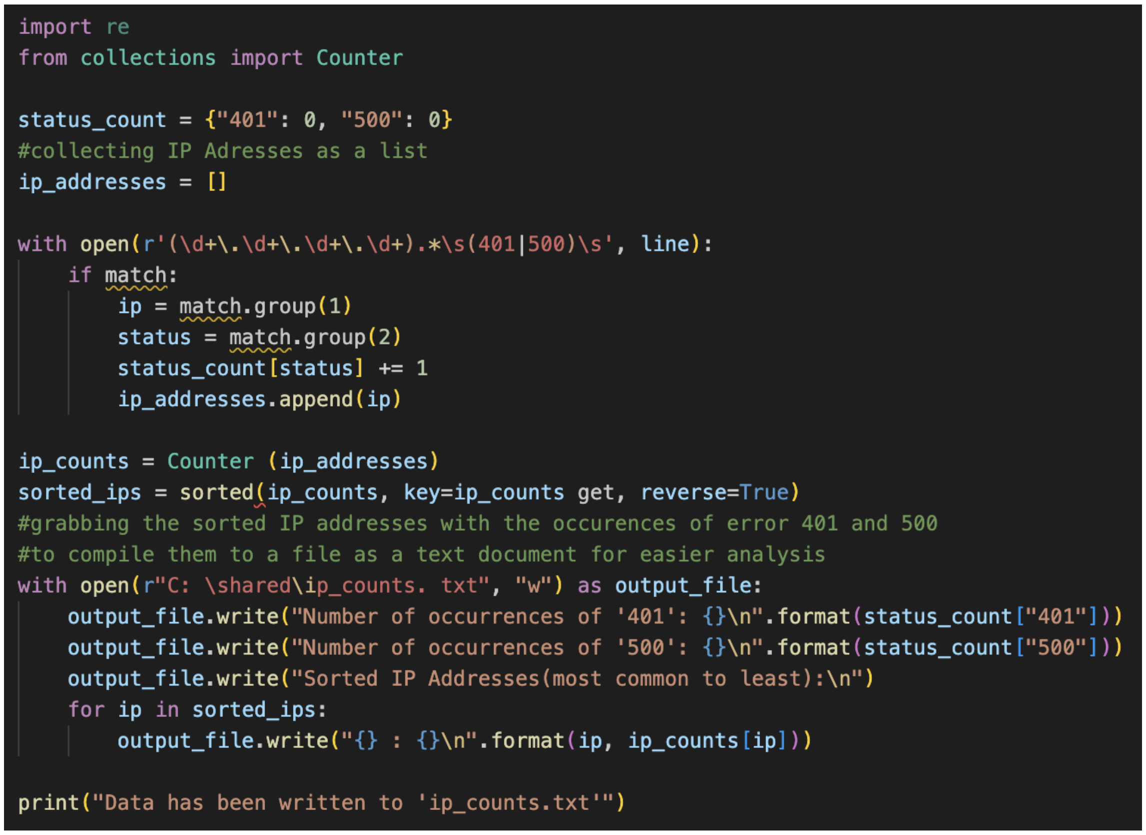
```
$ tail -f /var/log/apache2/error.log | egrep -v "(.gif|.jpg|.png|.swf|.ico)"
```

This code will be able to sort the logs into a select file, this will allow us to then enact our python script in order to sort through them all and get the main error codes we are concerned with, the 400 series, and the 500 series. These specific codes such as the classic “404” error codes that indicate that the server is not responding, which will allow us the ability to monitor the logs and the amount of logs in the file. In this bit of coding we will use “Error 401” as our target for monitoring.



```
count_401 = 0
count_500 = 0
#opening the log files of the network to inspect for
#certain error codes, and counting each one
with open(r"c:shared\access.log", "r") as logFile:
    for line in logFile:
        if "401" in line:
            count_401 += 1
        if "500" in line:
            count_500 += 1
#printing the amount of times the error code shows up
print("nummber of '404' :", count_401)
print("number of '500':", count_500)
```

This first snippet of code is an example of how we can filter for certain error codes such as error codes 401 and 500. It will also tell us the success of finding the error codes so that we can check if the code is functioning as it should. Instead of returning as a print line in the Python terminal, we could then filter the error codes into a file as a text document, for easier analysis of all the relevant data, for example, IP address and the error codes they generated.



```python
import re
from collections import Counter

status_count = {"401": 0, "500": 0}
#collecting IP Adresses as a list
ip_addresses = []

with open(r'(\d+\.\d+\.\d+\.\d+).*\s(401|500)\s', line):
    if match:
        ip = match.group(1)
        status = match.group(2)
        status_count[status] += 1
        ip_addresses.append(ip)

ip_counts = Counter (ip_addresses)
sorted_ips = sorted(ip_counts, key=ip_counts get, reverse=True)
#grabbing the sorted IP addresses with the occurences of error 401 and 500
#to compile them to a file as a text document for easier analysis
with open(r"C: \shared\ip_counts. txt", "w") as output_file:
    output_file.write("Number of occurrences of '401': {}\n".format(status_count["401"]))
    output_file.write("Number of occurrences of '500': {}\n".format(status_count["500"]))
    output_file.write("Sorted IP Addresses(most common to least):\n")
    for ip in sorted_ips:
        output_file.write("{} : {}\n".format(ip, ip_counts[ip]))

print("Data has been written to 'ip_counts.txt'")
```

The updated line of code above will allow us to use a coding technique called “Regular Expression” or RegEX for short, as a way of filtering out key occurrences of the “401” error code and “500” error codes respectively. Using RegEx for the filtering we can then take the data linked with those error codes such as the IP address that generated the error code, and sort them into a file as a text document to see the instances of those error codes being generated along with the information that we decided to gather from it. In the case above we only wanted to acquire the IP address connected with each error code generated.

Expected Output

The expected output of the programming above is to filter out the essential data we want to find from the server logs, namely the 401 error codes and the 500 error codes. While these codes are being filtered into a file as a text document, we are also extracting the important data from what generated those specific error messages such as the IP address. When we extract the useful data and compile them into a file, it will then allow log analysts to monitor and categorize the logs accordingly and report any unusual behavior or traffic amongst the server. “Printing” these lines of codes will also allow us to check the programming terminal to make sure our code is running as intended.

Documentation

With the tools implemented, the expectation is to cut the time it takes to analyze the logs in the file into a small condensed search through the file logs. The reason why we specifically want to look through the 400 & 500 series logs, is because those types of logs mainly deal with the servers traffic. The 500 series deals with client error logs, while the 400 series deals with the server error logs. If there is a suspicious amount of activity in either field we can log these into the file, and communicate with higher ups to take the necessary precautions. As outlined in the workflow portion of this document, the recommended time for documentation and escalation would be to allow till Friday for the accumulation of the server logs on thursday, then allow Log analysts Friday to review the filtered logs for any unusual behavior and to escalate the situation if something is deemed to be a potential attack. After that has been determined, emailing the manager would be the next course of action to the documentation process, to have records of the potential attack and to get any direction from the manager as to next steps of mitigation.

Unusual Behavior

When monitoring the error logs some examples of suspicious activity mainly come from the error 400 series code. The 400 series error codes deal with client or user based errors when connecting to a server. “Error 404” being one of the most common error messages is when there is a problem with the user connectivity or the server connectivity when trying to access a server. One code we can look at is “Error 401” as it deals with authentication, this error occurs when the user is not granted access to a server or domain because of the lack of authentication. When monitoring the logs a possible indicator of unusual behavior would be seeing a multitude (for example 5 or more) of these logs appearing from a certain IP Address trying to access a user account. The number of attempts could indicate either someone who has forgotten their password, or that there is an unauthorized user attempting to gain access to a user account which could then lead to different types of attacks such as a DDoS attack (Distributed Denial of Service attack), which would attack the server itself and cause the server to be inaccessible to other users and workers alike.

Potential Iterations

There could be a more efficient way of coding to sort the error logs, such as to enable a specific search to the 400 series or the 500 series. As more users generate more traffic to the server, we would need a more efficient way to search through the log files. These fixes can be enabled when the traffic starts to see an increase in users. Another possibility is setting up automated emailing using CRON job coding, which is a technique of coding for setting time intervals. Combining CRON job coding, with bash scripting for file sorting, would greatly improve the amount of time and resources spent on sorting through the logs.

Work Cited

MozDevNet. (n.d.-b). *HTTP response status codes - HTTP: MDN*. HTTP | MDN. https://developer.mozilla.org/en-US/docs/Web/HTTP/Status

Advocate, A. C. D., & Andrea ChiarelliPrincipal Developer AdvocateI have over 20 years of experience as a software engineer and technical author. Throughout my career. (2021, December 20). *Forbidden (403), unauthorized (401), or what else?*. Auth0. https://auth0.com/blog/forbidden-unauthorized-http-status-codes/

Compass. Welcome to Compass. (n.d.). https://cyber.compass.lighthouselabs.ca/p/2/days/w03d5/activities/2862

Jon Penland Jon is the Chief Operating Officer at Kinsta and is involved with Kinsta's sales. (2023, August 15). *A complete guide and list of HTTP status codes*. Kinsta®. https://kinsta.com/blog/http-status-codes/

Inc, C. (2022, March 23). *500 series server error status codes*. 500 Series Server Error Status Codes. https://techdocs.broadcom.com/us/en/ca-enterprise-software/it-operations-management/application-performance-management/10-7/administrating/cem-configuration/transaction-definition/http-status-codes/500-series-server-error-status-codes.html