**Cybersecurity Log Monitoring Workflow for "Turn a New Leaf"**

**Subject**: Weekly Log Monitoring Report - 2024-03-19

**Dear LHL manager,**

I hope this message finds you well. Enclosed in this email is the weekly Log Monitoring Report for the "Turn a New Leaf" web servers.

**Key Findings This Week**

* We are monitoring Apache web server logs, specifically the `access.log` file.
* The items of interest are HTTP status codes that may indicate errors (like 404 for Not Found, 403 for Forbidden, 500 for Internal Server Error) and patterns suggesting potential security threats (such as repeated failed access attempts from the same IP address).
* Logs are collected using a Bash script that copies the `access.log` file from its default location (`/var/log/apache2/access.log`) to a centralized location for processing.
* This process is done daily to ensure that the most recent logs are always analyzed.

:

* We use `grep` combined with regular expressions within a Bash script to extract log entries that match our criteria from the centralized log file.
* An example regex pattern might be `grep "404\|500" /centralized/location/access.log` to find all 404 and 500 status codes.
* "Interesting" items are defined as patterns or anomalies in the logs that deviate from normal operational behavior, which could indicate potential security incidents or system misconfigurations.
* A Python script, `log\_analysis.py`, takes the filtered logs and performs a detailed analysis.
* It uses regular expressions to parse each log entry, extracts the IP addresses and status codes, and then quantifies each unique occurrence.
* The script counts the frequency of status codes and IP addresses to identify any unusual or unexpected patterns.
* Cron is used on Linux systems to automate the execution of the log collection and analysis scripts. For example, to run the collection script every hour, you would use `0 \* \* \* \* /path/to/collect\_logs.sh`.
* On Windows, you would use the Task Scheduler to automate running the `log\_analysis.py` script daily, ensuring consistent analysis without manual input.
* A weekly summary report generated by the Python script is sent via email to the manager. This report consolidates the findings and provides insights into the overall health and security of the web services.
* For immediate issues, such as a detected security breach, an alert is sent out promptly to initiate a rapid response.

By setting up this workflow, "Turn a New Leaf" ensures that its web servers are under constant surveillance for potential issues, thus maintaining the integrity and performance of its digital infrastructure while complying with regulatory requirements. The documentation accompanying this workflow will provide all necessary instructions and explanations for the technical staff to maintain and potentially enhance the system.

**Actions Taken:**

* All error occurrences have been documented and timestamped.
* The origin of suspicious traffic has been logged and is being monitored closely.
* Relevant system administrators and web developers have been notified where appropriate.

Please review the attached documents at your earliest convenience. Should you require a more detailed account or if you have any questions regarding this report, feel free to contact me directly.

**Looking Ahead:**

* We are exploring the integration of a real-time alerting system for immediate notification of such irregularities.
* Plans are underway to implement a more sophisticated analysis using machine learning techniques for predictive threat detection.

Thank you for your attention to this matter. I will continue to closely monitor our systems and report any further anomalies as necessary.

**Best Regards,**

Reniel Villahermosa

**Introduction:**

This project outlines a comprehensive workflow to monitor and analyze the web server logs of "Turn a New Leaf," a non-profit organization. It encapsulates the entire process, from data collection to reporting, to detect and flag unusual activities potentially indicative of cybersecurity threats.

**Workflow Overview:**

The monitoring workflow is an automated process that performs daily log analysis to ensure consistency in performance and security monitoring. The steps include:

1: Automatically retrieve access logs daily from the web servers for data collection

2. Parse the logs using Python to identify common HTTP methods, status codes, and transmitted data volumes which is a form of data analysis

3. Trigger alerts for unusual patterns such as a high number of 404 errors which might indicate a scanning attack or probing of the network for alert generation

4. Generate and send a weekly summary report to the manager detailing the findings which includes a report

**Programming Tools and Scripts**:

The process employs several tools:

* **Python:** For writing scripts to parse logs and analyze data.
* **Bash:** To automate log retrieval and manage file operations in Linux environments.
* **Cron:** For scheduling tasks in Unix-based systems, ensuring the Python script runs as planned..
* **Regular Expressions:** Used within the Python script to extract relevant data from the logs.

Key commands will include `crontab` entries for Unix systems and `Scheduled Task` configurations for Windows systems.

**Expected Output:**

The Python script produces an output file summarizing:

* Counts of HTTP methods.
* Status code occurrences.
* Total bytes sent.

This output is important to quickly assess server activity and highlight potential issues.

**Documentation:**

Every step of the process will be documented. The code will be commented for clarity, and a README file will include:

* Instructions for setting up and running the scripts.
* Descriptions of output files and how to interpret them.

Weekly reports will be emailed to the manager, and critical alerts will be sent immediately.

**Unusual Behaviour Flags:**

The system will flag:

* Unusually high traffic volumes.
* Frequent access from unrecognized IP addresses.
* A surge in error codes like 404 or 503, which might indicate potential threats or server health issues.

**Potential Iterations:**

Future iterations could introduce:

* Machine learning to predict and adapt to new threat patterns.
* Real-time alerting systems for immediate response.
* A web interface for non-technical staff to monitor reports.

**Conclusion:**

This workflow aims to provide "Turn a New Leaf" with a robust cyber security monitoring system that is both comprehensive and adaptable to evolving threats, ensuring the safety of their digital infrastructure.

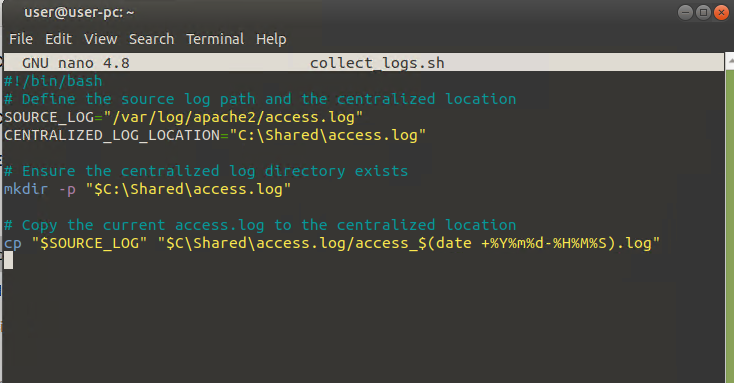
Appendix:

* Sample codes and scripts.
* Cron job entries and Task Scheduler settings.
* Sample output file format.

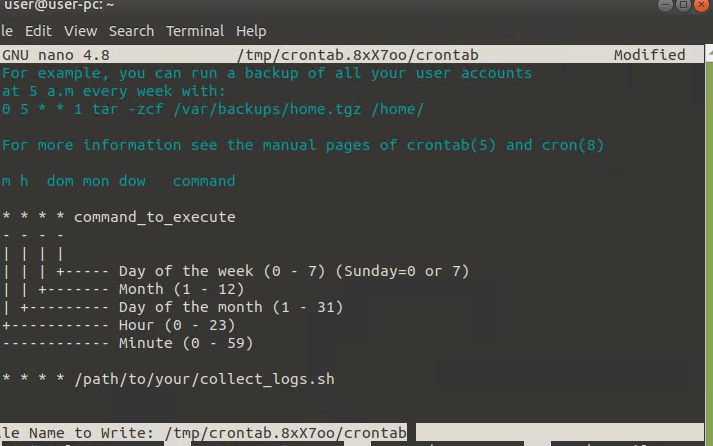
**Screenshots on the Python script:**

**Step 1:**

First was to create a bash to write the script then transfer it to the Windows shared folder.

****

After using chmod +x collect\_logs.sh to make script executable

Using cron to schedule the frequency

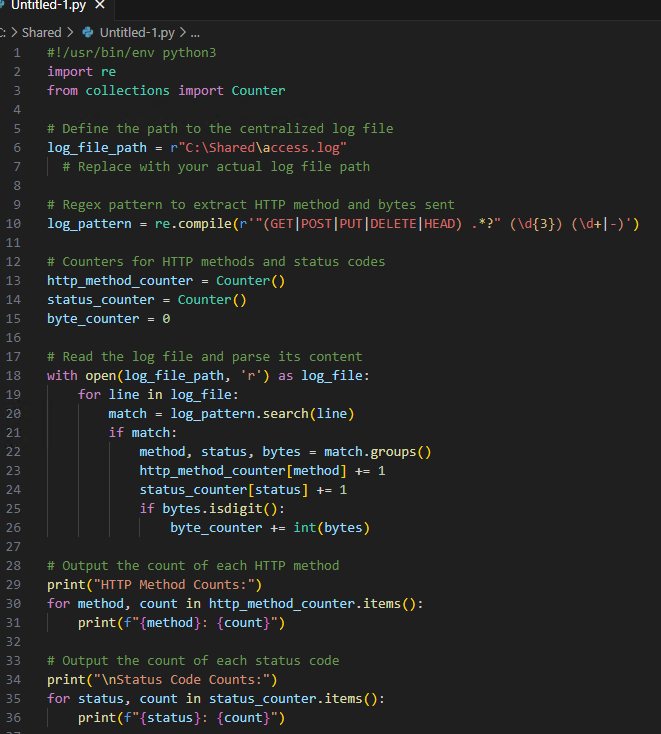
So I would like to run a script at 4:30 pm on th first of ever month:

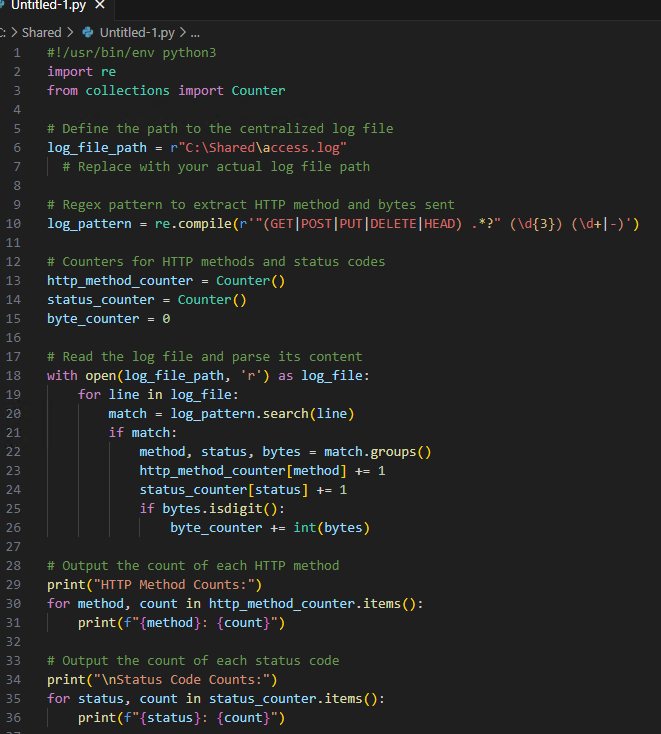
The script would like something like this:

30 16 1 \* \* /collect\_logs.sh script.

**Screenshots of the Python Script**

I’ve developed a Python script that serves as a log analyzer. This script is designed to parse server access logs to extract and quantify HTTP methods and status codes, and to calculate the total bytes sent over the network. It uses regular expressions to identify patterns within the log entries, isolating HTTP methods (like GET, POST, PUT, DELETE, HEAD), status codes (such as 200, 404), and the volume of data transferred as represented by the byte size at the end of each log entry. After parsing the logs, it tallies the occurrences of each HTTP method and status code using Python's `Counter` class, and sums the total bytes sent. Finally, it outputs these statistics to the console and also writes them to an output file for easy review and archival purposes. The output is essential for monitoring network traffic and identifying potential anomalies or issues with the web servers, and it aids in proactive cybersecurity measures.





**Citations**

"Crontab Guru - Quick and Easy Crontab Scheduling." Accessed March 19, 2024.<https://crontab.guru/>.

"RegExr: Learn, Build, & Test RegEx." Accessed March 20, 2024.<https://regexr.com/>.

