Log Monitoring Workflow  
Project Report

**Created by: Harsh Patel  
Date: January 6th, 2025**

**Table of Contents**

1. **Executive Summary**
2. **Solution Section**
3. **Potential iterations Section**
4. **Conclusion**
5. **References**

**Executive Summary**

In this project, we have developed an automated workflow to monitor network traffic for unusual activity, focusing on failed login attempts across Linux and Windows systems. By leveraging Bash and Python scripting, we provide an efficient mechanism to detect potential security threats, alert the relevant personnel, and document findings in a structured manner. This workflow minimizes manual effort, ensures compliance with regulatory requirements, and enhances the organization's overall security posture.

**Solution Section**  
**Script**

A computer screen shot of a program

Description automatically generated

**Explanation**

1. ***LOG\_FILE="/var/log/auth.log"***This defines the path to the log file where authentication attempts (successful and failed) are recorded./var/log/auth.log is commonly used on Linux systems to store authentication logs, including login attempts.
2. ***ALERT\_THRESHOLD=3***  
   This sets the threshold for failed login attempts. An alert will be triggered if the number of failed attempts exceeds this value.
3. ***FAILED\_COUNT=$(grep "Failed password" $LOG\_FILE | wc -1)***

**grep "Failed password" $LOG\_FILE:**Searches the log file for lines containing the phrase "Failed password", which indicates a failed login attempt.  
**| wc -1:**The pipe (|) takes the output of the grep command and sends it to wc -1, which counts the number of lines.  
The result is stored in the variable FAILED\_COUNT, representing the total number of failed login attempts.

1. ***if [ $FAILED\_COUNT -gt $ALERT\_THRESHOLD]; then***This if statement checks if the number of failed login attempts (FAILED\_COUNT) is greater than the threshold (ALERT\_THRESHOLD).
2. ***echo "ALERT: Unusual number of failed logins detected - $FAILED\_COUNT failed attempts."***If the condition is true, it prints an alert message to the terminal, showing the number of failed login attempts.

**Script**

A computer screen shot of a program code

Description automatically generated

**Explanation**

***log\_file="/var/log/auth.log"***This variable defines the path to the log file (auth.log) where login attempts are recorded. This file is typically found in /var/log/ on Linux systems.

***threshold=3***This variable sets the threshold for the number of failed login attempts. If the number of failed attempts exceeds this threshold, an alert will be sent.

***alert\_email=***[***manager@turnanewleaf.com***](mailto:manager@turnanewleaf.com)This variable holds the email address where the alert will be sent.

***failed\_count=$(grep "Failed password" "$log\_file" | wc -l)*grep "Failed password" "$log\_file"**: This command searches for all occurrences of the phrase "Failed password" in the specified log file (auth.log). Each occurrence represents a failed login attempt.**wc -l**: This command counts the number of lines output by grep, which corresponds to the number of failed login attempts.The result is stored in the variable **failed\_count.**

***if [ $failed\_count -gt $threshold ]; then***This conditional checks if the number of failed login attempts (failed\_count) is greater than the threshold **(3)**.

***echo "Security Alert: $failed\_count failed login attempts" | mail -s "Security Alert" "$alert\_email"***If the condition is true, this line sends an email to the specified alert\_email with the subject "Security Alert" and a message that includes the number of failed login attempts.

**Outcome**

**First Bash Script Output:** An alert message indicating the number of failed login attempts.

**Second Bash Script Output:** Email containing weekly report sent to responsible personnel.

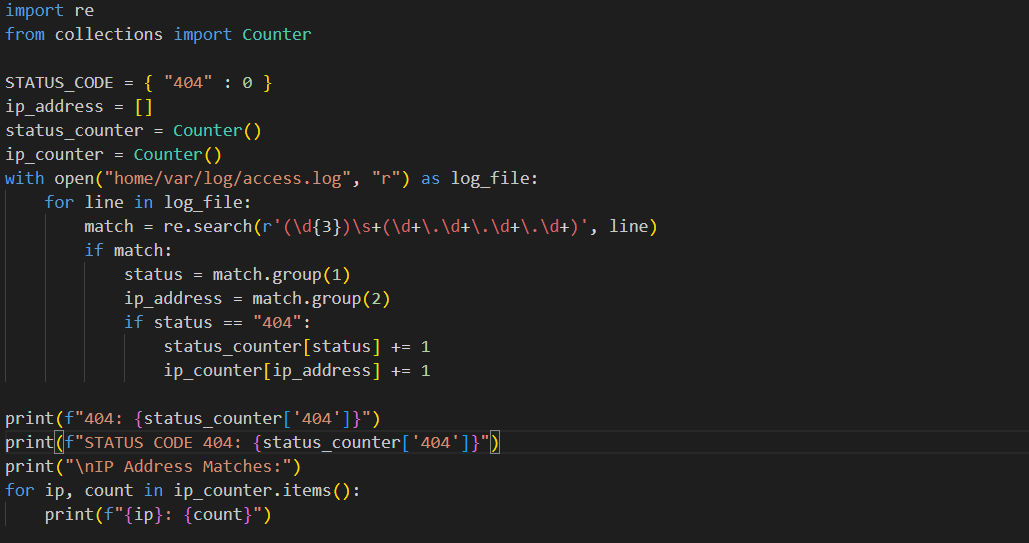
**Script**

A computer screen with text

Description automatically generated

This script ensures that the Failed Login script runs every 5 minutes to check for failed or bad passwords used by someone so it can detect any Brute-Force attack or someone trying to attack.

**Script**



This Python script is designed to parse a web server access log file and count how many times the HTTP status code "404" (Not Found Error) occurs, along with counting how often each IP address is associated with a 404 error.

**Explanation**

***STATUS\_CODE = { "404" : 0 }***: A dictionary initialized to track the count of 404 status codes.

***status\_counter = Counter()***: A Counter to count occurrences of each HTTP status code. Specifically, it is used here to track how many times status code "404" occurs.

***ip\_counter = Counter()***: A Counter to count occurrences of each IP address that is associated with a 404 status code.

***match = re.search(r'(\d{3})\s+(\d+\.\d+\.\d+\.\d+)', line)*:**This regular expression searches for:A three-digit status code (\d{3}), followed by a space (\s+), and An IP address in the form of xxx.xxx.xxx.xxx (\d+\.\d+\.\d+\.\d+).  
If a match is found, it extracts the status code and IP address from the line.

***status = match.group(1)***: This extracts the first capturing group from the regex match, which is the HTTP status code.

***ip\_address = match.group(2)****:* This extracts the second capturing group, which is the IP address.  
The script then checks if the status code is "404":

***if status == "404":*** If true, the script increments the count for "404" in the status\_counter.

***status\_counter[status] += 1***: Increments the count of the "404" status code.  
It also increments the count of the specific IP address in the ip\_counter:

***ip\_counter[ip\_address] += 1***: Increments the count of the IP address that triggered the 404 error.

**Outcome**A screenshot of a computer

Description automatically generated

**Python Script Output:** Gives you Occurrences and list of IP Address related to 404 error code.

**Documentation**

Bash Script is set to run every 5 minutes and generate and send weekly reports and alerts to the Manager.

**Unusual Behavior Criteria**

* Failed login attempts exceeding the threshold of 3 in a day.
* Repeated Attempts from a Single IP.
* Multiple failed attempts from the same IP address.
* Off-Hours Access: Failed login attempts during non-business hours.

**Potential Iterations**

**Add Email Alerts**: Modify the Bash script to send an email when an alert is triggered instead of a weekly report.

**Include More Details**: Enhance the Python script to include timestamps and IP addresses.

**Real-Time Alerts**: Use a real-time monitoring tool to send instant alerts.

**Real-Time Monitoring**: Implement real-time alerts using tools like tail.

**IP Address Blocking**: Implement a feature to block IP addresses after multiple failed attempts.

**Improved Report Format**: Add graphical summaries to the weekly report.

**Conclusion**

This simplified solution using Bash and Python scripts provides an efficient and easy-to-understand approach to monitoring login attempts. It ensures timely detection of unusual activities and helps maintain system security. Future fixes can focus on improving report visualization and real-time monitoring and alerts.

**References**

*CronHowTo - Community Help Wiki*. (n.d.). <https://help.ubuntu.com/community/CronHowto>

*7. Input and output*. (n.d.). Python Documentation. <https://docs.python.org/3/tutorial/inputoutput.html>

*Bash Reference Manual*. (n.d.). <https://www.gnu.org/software/bash/manual/bash.html>

Bash scripting cheatsheet. (n.d.). *Devhints.io Cheatsheets*. <https://devhints.io/bash>

Rane, M. (2020, April 2). A Beginner's Guide to Grep: Basics and Regular Expressions. *Open Source For You*. <https://www.opensourceforu.com/2012/06/beginners-guide-gnu-grep-basics/>

J, H. (2023, January 31). *Linux Commands cheat sheet - Quickly reference common commands!* Websentra. <https://www.websentra.com/linux-commands-cheat-sheet/>