# Log Monitoring Workflow for

# Turn a New Leaf (TaNL)

Prepared by: Esther Ogunlana for LHL Project

# Table of Contents

# Log Monitoring Workflow for Turn a New Leaf (TaNL)

## Executive Summary

Turn a New Leaf is a non-profit organization that supports youth in a range of rural communities to seek employment.
This report is based on the analysis performed on Linux and Windows, and the two servers operated by Turn a New Leaf. The main objective of this report is to establish a workflow and monitor for unusual traffic which could be an Indicator of Compromise (IoC) that unscrupulous entities are trying to gain access to the server which contains sensitive and personally identifiable information (PII) such as social security numbers, full names, addresses, etc.
This report also indicates that implementing automated scripts can lead to improved resource management and proactive issue resolution, ultimately contributing to better system reliability and performance. The ability to adapt and iterate on these scripts further empowers administrators to tailor solutions to their specific needs, fostering a more responsive and effective operational environment.

## Introduction

Log files are digital records that capture events, processes, and transactions occurring within a system or application. They provide a chronological account of activities, often including timestamps, user interactions, error messages, and system performance metrics. Log files are essential for troubleshooting, monitoring, and auditing system behaviour (Zhang et al., 2020).

An access log file typically records details about user login attempts, successful and unsuccessful. It indicates when the attempt occurred, the IP address identifies the source of the attempt, the username shows which account was targeted, and the status and reason provide insights as to why the login failed, for example, an invalid password.

Monitoring such logs is crucial for identifying potential security threats, as repeated unsuccessful login attempts can indicate brute-force attacks or unauthorized access attempts (Smith, 2022).

Turn a New Leaf (TaNL) members have between 8:00 am and 12:00 noon on Thursdays to provide their weekly reports. For this report, I start to monitor the logs for any unusual traffic in the afternoon, after the weekly check-in.
I have saved an **access.log** file in my Desktop folder for this case study. Hence the file path is ***/home/student/Desktop/TaNL/access.log***

For this report, I will be focused on finding any unusual traffic outside of the active user period, which for this study is 8:00 am to 12:00 noon.
To use a bash script to analyze the log, I would first create a bash file by using the nano command, and including the instructions which in this case is analyzing my log file to capture failed log-in attempts.

The result of my log analysis is the basis for this report. If unusual traffic is discovered this will be included in the report. Since this report focuses solely on failed login attempts, all data relevant to this event will be recorded. If failed log-in attempts are recorded, it is important to notify management immediately. I could either include the command in my Bash script code or I could send the email manually.

Finally, it is essential to record this process for record-keeping purposes and to ensure that problems and issues are not repeated and it also provides precedents for solving future problems.

## Solutions

A Bash script is a plain text file that contains a series of commands written in the Bash (Bourne Again SHell) programming language. It automates tasks by executing these commands sequentially when the script is run. Bash scripts are commonly used for system administration, file manipulation, program execution, and managing system processes (Pope et al., 2020).
A Bash script is essential for this report since our objective is to automate the monitoring process.

### Script Overview

This script is designed to monitor failed login attempts on a Linux system by analyzing the authentication log file. After the log file is analyzed, failed login attempts are recorded, and an email alert is sent to a manager, Sharon Oscar if unusual activity is detected.

### Script Breakdown

1. Shebang Line: All bash scripts begin with this line as it indicates that the script should be run in the bash shell. Without this line, the script might be executed with a different shell which could lead to errors if the script contains elements specific to Bash (Tanenbaum, 2016). The shebang line looks like this: **#!/bin/bash.**
2. Variables: In Bash scripts, variables are used to store data that can be referenced and manipulated throughout the script. Variables are defined by assigning value with the “**=**” sign.
   Once variables are defined, it allows for more flexible and maintainable scripts, as you can change the value of the variable in one place rather than throughout the entire script(Robinson, 2020).
   For this report, I am using an Access log file saved on my Desktop, hence,
   ***Log_File=/home/student/Desktop/TaNL/access.log***
   Other variables for this report are the output file, which is the result of the analysis as the name implies, email, this report is designed to alert a manager Sharon Oscar whose email is provided in the script, and the current date which is the date the script is set to run.

3. Report Header: A report header in a Bash script serves as an introductory section that provides essential information about the content of the report. It typically includes details such as the title, date, and any relevant context that helps the reader understand what the report entails. In a Bash script, a report header can be defined using the echo statement to output formatted text to a file. This command creates a header for a report on failed login attempts, including the current date for reference. The use of a header not only improves readability but also provides crucial context for the data that follows, making it easier for stakeholders to interpret the results (Miller, 2018).
   For example, ***echo "Failed Login Attempts Report for $(date + "%Y-%m-%d")" > "$Output_File***" which is named as failedlogins.
4. Analyze Log File: Analyzing a log file in a Bash script involves examining the contents of the log to extract useful information, such as errors, warnings, or specific events. Such analysis is crucial for monitoring system security and troubleshooting issues, as it enables administrators to quickly identify and respond to potential problems (Kumar, 2021). This process often utilizes commands like `grep` to search for keywords or patterns that indicate particular types of log entries. For this report, we are analyzing the log file for failed login attempts.
   For this script, the command is ***grep"failed"/home/student/Desktop/TaNL/access.log.***

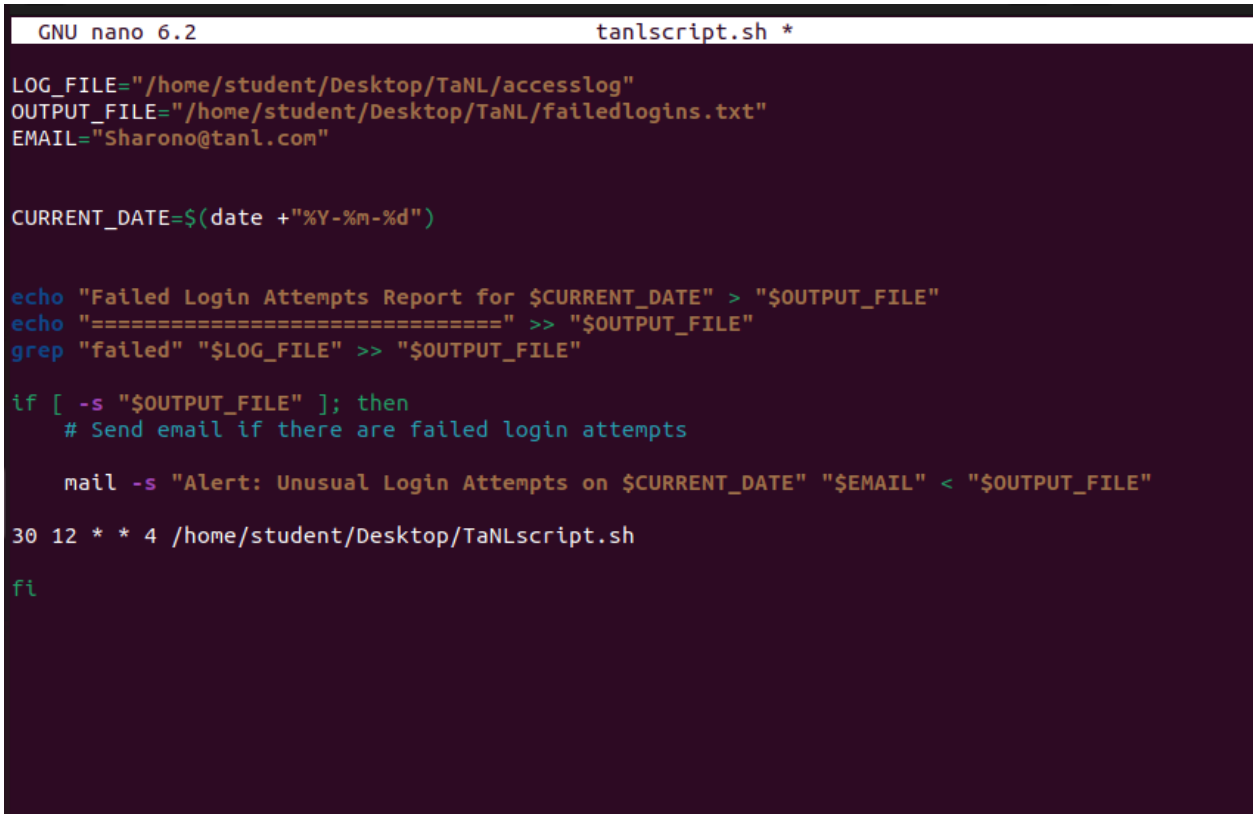
5. Failed Login Attempts: In Bash scripting, if conditions are used to perform conditional evaluations, allowing the script to execute certain commands based on whether a specified condition is true or false. This enables decision-making capabilities within the script (Ramey, 2019).
   For this script, the if statement used states that if there is a value greater than 0 in the output file, an email should be sent to the indicated email address SharonO@tanl.com.
   ***If [-s "$failedlogins" ]; then mail -s "Alert: Unusual Login Attempts" "$sharon0@tanl.com" < "failedlogins"***
6. Send Email Alert: If there are any failed login attempts, this command sends an email to a specified receiver with a subject line indicating that unusual login attempts were detected, including the contents of the report.
   For this script, the command ***mail -s "Alert: Unusual Login Attempts" "$sharon0@tanl.com" < "failedlogins"*** ensures that the result of the analysis is promptly sent to Sharon, the manager.
7. Automate the Script: Cron jobs are useful for automating routine tasks such as log monitoring without manual intervention. For this report, since our goal is to monitor the log after users have updated their information, we can schedule our script to run right after with the help of a cron job (Smith, 2021). For this script, the script is scheduled to automatically run at 12:30 pm on Thursdays. It appears as " ***30 12 * * 4 /home/student/Desktop/TaNL/access.log"***

## Potential Iterations

There is a possibility that the set threshold may be too low to indicate malicious intent, so after reviewing the output pattern, there is a likelihood of increasing the threshold to seven(7) or ten(10) as the basis for alerts.
Depending on the result of the analysis, we may decide to increase or decrease the frequency of running the script to suit our needs.

## Conclusion

The expected output for this process is a report containing details of the report such as date, time, source, reason, and list of failed login attempts. This output narrows down a list of processes and provides the manager with only the important details that can be treated promptly.
For this report, I have created a folder for Turn a New Leaf which I have saved on my Desktop. All files will be saved in the folder and the generated report will also be included in this directory. This folder will include all the files concerning Turn a New Leaf. The output of the script will be saved in ***TaNLfailedlogins.txt***
A command line was also included in the Bash script to send the report to the manager for prompt action.
The purpose of this report is to find and report unusual behaviour and to find out what unusual behaviour is, a criterion needs to be set. It is possible to input the wrong password when trying to log into an account so the threshold for this report will be set at five(5) failed attempts originating from an unfamiliar IP address.
By creating a Bash script, we can automate the log-monitoring process and integrating a cron job further enables the script to run at scheduled intervals without manual intervention.



```
  GNU nano 6.2                                  tanlscript.sh *

LOG_FILE="/home/student/Desktop/TaNL/accesslog"
OUTPUT_FILE="/home/student/Desktop/TaNL/failedlogins.txt"
EMAIL="Sharono@tanl.com"


CURRENT_DATE=$(date +"%Y-%m-%d")


echo "Failed Login Attempts Report for $CURRENT_DATE" > "$OUTPUT_FILE"
echo "==============================" >> "$OUTPUT_FILE"
grep "failed" "$LOG_FILE" >> "$OUTPUT_FILE"

if [ -s "$OUTPUT_FILE" ]; then
    # Send email if there are failed login attempts

    mail -s "Alert: Unusual Login Attempts on $CURRENT_DATE" "$EMAIL" < "$OUTPUT_FILE"

30 12 * * 4 /home/student/Desktop/TaNLscript.sh

fi
```

**Sample Script**

References

Kumar, R. (2021). *Linux Log Management: A Comprehensive Guide to Logging on Linux Systems*. Packt Publishing.

Miller, B. (2018). *Learning Bash Scripting: A Beginner's Guide*. O'Reilly Media.

Pope, R., Smith, J., & Patel, L. (2020). *Mastering Bash Scripting*. Programming Press.

Ramey, T. (2019). *Bash Cookbook: Solutions and Examples for Bash Users*. O'Reilly Media.

Robinson, J. (2020). *Bash Scripting and Shell Programming (Linux Command Line)*. Independently published.

Smith, A. (2021). *Automating Tasks with Cron Jobs*. System Administration Handbook.

Smith, R. (2022). *Defending Against Unauthorized Access: Strategies and Techniques*. Information Security Review, 10(2), 123-135.

Tanenbaum, A. S. (2016). *Operating Systems: Design and Implementation*. Pearson.De

Zhang, L., Chen, R., & Wang, Y. (2020). *Log File Analysis for System Monitoring*. Journal of Systems and Software, 170, 110-120.