PROJECT TITLE

Log Analyzer Bot

T. SURYA AJAY KUMAR SHASHIDHAR

CYBER SECURITY CYBER SECURITY

TAH3006251682 TRAINER

TEKS ACADAMY 9603343287

9502264777

**Abstract:**

The Log Analyzer Bot is an automated tool running on Linux to monitor SSH login attempts. It parses /var/log/auth.log to identify suspicious or unauthorized access attempts by looking for failed logins. The bot collects data on the IP address, username, and frequency of failed attempts and stores it as variables in order to determine there are repeated too many failures (over a threshold). If it detects repeated failures, the bot blocks the offending IPs via UFW (Uncomplicated Firewall) based on the type of access they were attempting, thereby improving the overall security of the server. The bot was built with bash scripting, and utilises some basic tools like grep, awk, and cron, and is a lightweight but effective way to continuously monitor for and automate a response to SSH login attempts and threats to Linux systems.

**Problem Statement:**

In a new-age Linux-based platform, especially on servers, the major threat from an SSH perspective is unauthorized SSH login attempts. Hackers and bots that are mandated to discover vulnerability within the SSH login method usually do so using brute force in the form of either a large collection of accounts or a series of username/password combinations. Key system administrators are often inundated with attacks/issues that stem from the exponential amount of log data, thus making it impossible to outline these attacks manually.

There is a recognized need for a semi-automated system to detect suspicious/failed SSH login attempts, outline potential forms of attack, provide details to generate actionable metrics and insights, maintain an auditable track record, provide insight and act on identified malicious IP's within that audit trail.

**Project Objective(s):**

* Log Monitoring Automation:

Create a bash script that can perform log monitoring of /var/log/auth.log for failed and suspicious SSH login attempts on an ongoing basis.

* Intrusion Pattern Detection:

Identify and capture key entries of the extra data such as IP address, user name, time, and frequency of failed log in attempts using commands such as grep and awk.

* Malicious IP Address Auto-Blocking :

Integrate with USF (Uncomplicated Firewall) to block any IP address encountered during the monitoring period that exceeds a defined amount of failed login requests.

* Admin Alert Notification:

Send email notifications and/or log alerts in circumstance of abnormal login activity or when actions are taken (e.g. blocking an IP).

* Continuous Operation

Run the script automatically at periodic intervals using cron which would permit responding in real-time and 24/7 log analysis.

* Log Detected Threats and Actions

Log detected threats and actions taken in a secure backup file for the period of review/auditing and information for possible future use.

**Project Scope:**

* check /var/log/auth.log for failed ssh login attempts.
* Detect and pull out suspicious IP addresses.
* Auto-block malicious IPs using ufw.
* Add a cron with periodic checking.
* Log actions and notify the admin.
* Designed for Linux systems Kali).
* Concentrating on ssh-based intrusion detection.

**Tools & Technologies:**

* Bash Scripting - Run logging parsing/logging and system commands.
* Grep & Awk - Get the relevant data from /var/log/auth.log.
* UFW (Uncomplicated Firewall) - Block suspicious IP addresses.
* Cron - Call the bot to check regularly.
* Mail / Mailx - Send email notifications to administrators.
* Logger - Log events to system logs for tracking.
* Linux OS - Specifically, Debian-based systems like KalI

**Timeline (Tentative):**

**Days**  **Activity**

Day 1 Planning the project, configuring environment, and gathering sample logs

Day 2 Retrofitting a bash script to parse /var/log/auth.log with grep and awk

Day 3 Integrating ufw to auto-block suspicious Ips

Day 4 Adding email alerts and further logging with mail and logger

Day 5 Triggering with cron, final testing, and documenting

**Deliverables:**

* Bash script to analyze SSH login attempts from /var/log/auth.log
* Cron job for automated period execution
* Integrated with ufw to block suspicious Ips
* logging file to incriminate detected intrusions and action taken
* email or system alerts using mail or logger
* User manual / README with setup and usage instructions
* Final project report detailing objective, scope, implementation & results

**Execution:**

**Step 1:** Enable and Verify UFW (Firewall) and installation process

**A computer screen shot of a computer screen

AI-generated content may be incorrect.**

**Step 2:** create file and write bash script save and exit

A computer screen shot of a computer screen

AI-generated content may be incorrect.

****

**Step 3:** Give Execution Permissions



**Step 4:** check ufw firewall status and Automate the Script with Cron tab

****

****

**Step 5:** check the cron tab and all the running process and the details

****

**Front end implemetation:**

The front-end implementation of the GUI-Based Log Analyzer Bot on Linux front end was developed incorporating Python's Tkinter library. Tkinter offers a simple and effective way to build desktop applications. The GUI allows users to interact with the log analyzer in an easy-to-use GUI so that they do not need to use the terminal. The front end has various components: input fields so users can set a threshold for failed login attempts and buttons to run the log analysis, export results to CSV, and even manually block IP address using the system’s firewall (ufw). The results are displayed in a scrollable text area showing the IP addresses and the number of failed SSH login attempts, parsed from /var/log/auth.log . The application also has a back-end parser script that implements regular expressions and Python’s collections.Counter to scan the log file for suspicious activity. Overall, the GUI eases the monitoring of login attempts and user actions; making overall usability easier to apply accessible controls to review logs and follow up with appropriate security actions.

**GUI:**

A screenshot of a computer

AI-generated content may be incorrect.