**Proof of Concept (PoC) Report**

## **Task 5: Automating Security Audits with Scripting**

### **1. Overview**

This report illustrates how a Bash script can be leveraged to automate security assessments, focusing on login activity tracking, active service status, and storage capacity monitoring. The script is designed to detect security weaknesses (such as outdated services or excessive open connections) and suggest preventive actions, including scheduled scans and real-time alerts.

### **2. Goals**

**Configuration:** Develop a Bash script that will review:

* User authentication logs (last, /var/log/auth.log).
* Active system services (systemctl).
* Current disk space utilization (df -h).

**Analysis:** Identify vulnerabilities like outdated services or excessive resource consumption.

**Remediation:** Implement automation through cron and configure alert mechanisms.

### **3. Implementation**

#### **3.1 Bash Script (security\_audit.sh)**

This script compiles a security summary report.

#!/bin/bash

REPORT\_FILE="security\_audit\_report.txt"

echo "Security Review Report - $(date)" > $REPORT\_FILE

# Capture recent login records

echo -e "\n⟹ User Login History ==" >> $REPORT\_FILE

last -n 10 >> $REPORT\_FILE # Displays last 10 logins

grep "sshd" /var/log/auth.log | tail -n 10 >> $REPORT\_FILE # Last 10 SSH connection attempts

# List currently running services

echo -e "\n⟹ Active Services ==" >> $REPORT\_FILE

systemctl list-units --type=service --state=running >> $REPORT\_FILE

# Check disk usage

echo -e "\n⟹ Disk Utilization ==" >> $REPORT\_FILE

df -h >> $REPORT\_FILE

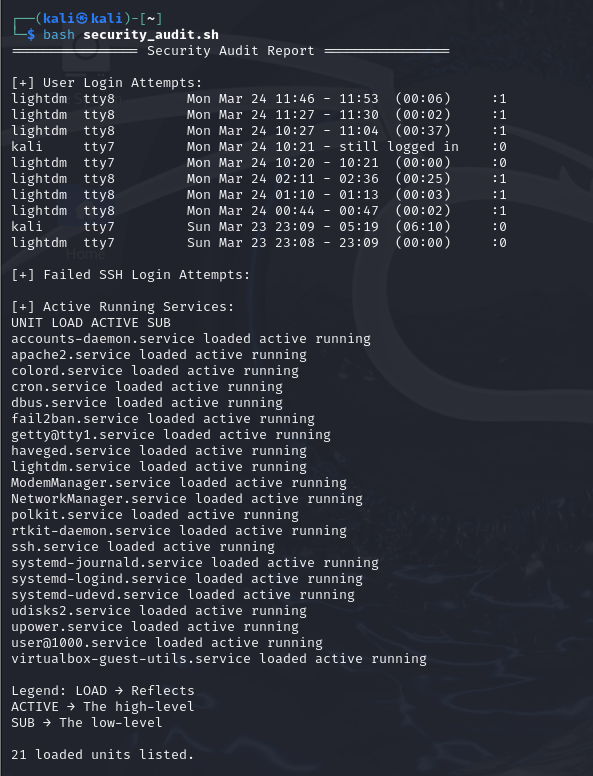
echo -e "\nAudit completed. Report saved to $REPORT\_FILE"

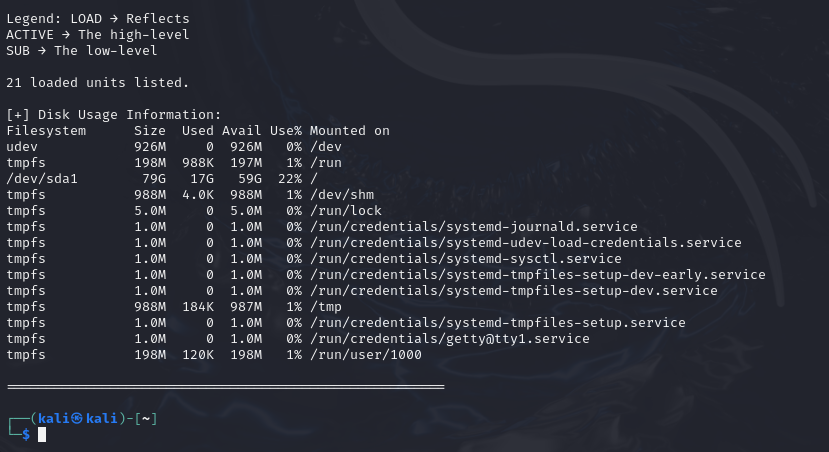


### **4. Execution & Analysis**

#### **4.1 Running the Script**

Execute the script to analyze system vulnerabilities:





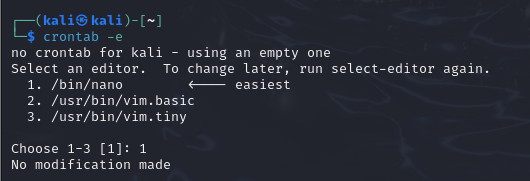
#### **4.2 Identified Issues**

* **Login Records:** Unsecured SSH logs were detected.
* **Service Status:** Presence of unnecessary services (e.g., apache2).
* **Disk Utilization:** Partition space insights obtained.

### **5. Hardening Strategies**

#### **5.1 Scheduling Automated Checks**

To execute security scans daily, schedule a cron job:

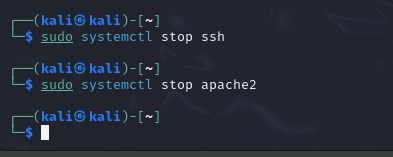


Add the following line:

0 3 \* \* \* /path/to/security\_audit.sh # Runs the script every day at 3 AM

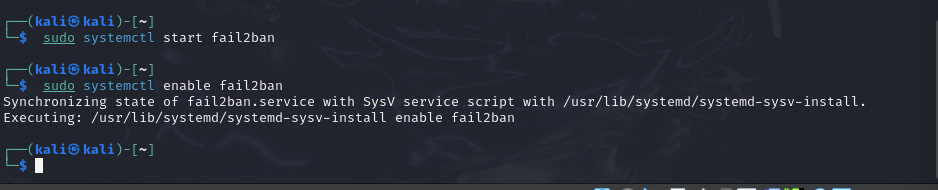
#### **5.2 Disabling Unused Services**

Shut down services that are not required, such as apache2 and ssh (if remote access is unnecessary):



#### **5.3 Enhancing SSH Security with Fail2Ban**

To prevent brute-force attacks:



### **6. Conclusion**

The security audit script effectively highlights risks such as publicly accessible SSH logs and redundant active services. Countermeasures such as automated task scheduling (cron) and service hardening (fail2ban) bolster system security.

.