

**Name:** Bint e Fatima Niazi

**Date:** July 2025

**Contact:**

-Email: fatimabinte4@gmail.com

-LinkedIn: linkedin.com/in/bint-e-fatima-niazi

-GitHub: github.com/fatima-niazi

Project: SSH Brute Force Simulation on Laptop and Android and SSH log capture

# Introduction

This cybersecurity lab simulates a brute-force SSH login attack in two scenarios: 1) against a Linux laptop (Kali Linux) with OpenSSH server, and 2) against an Android device using Termux and OpenSSH. The objective is to understand how brute-force attacks work and how to monitor them like a SOC analyst.

# Objectives

- Simulate SSH brute-force attacks using Hydra  
- Set up targets on both Linux and Android environments  
- Capture and analyze logs on Kali Linux (SOC behavior)  
- Understand attacker techniques and defender detection strategies

# Tools & Environment

- Kali Linux (Attacker & Monitoring)  
- Android Phone with Termux (Secondary Target)  
- OpenSSH (installed via `openssh-server` on Kali and `openssh` on Termux)  
- Hydra (password brute-forcing tool)  
- password.txt (custom wordlist)  
- Same Wi-Fi Network for communication

# Step-by-Step Execution

1. **Installed OpenSSH server on Kali Linux: `**sudo apt install openssh-server**`**

A screen shot of a computer

AI-generated content may be incorrect.

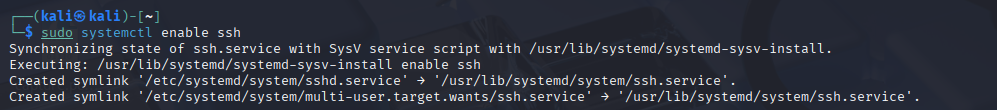
1. **Created a weak user on Kali: `**sudo adduser testuser**` with password `**12345`

A computer screen shot of a computer

AI-generated content may be incorrect.

1. **Started SSH service: `**sudo systemctl start ssh**` and ensured it was active.**





A computer screen shot of a computer program

AI-generated content may be incorrect.

1. **Created password list `**password.txt**` in Kali with test entries: `**1234`, `qwerty`, `hello`, `user1234`

A computer screen shot of a computer code

AI-generated content may be incorrect.

A computer screen shot of a black box

AI-generated content may be incorrect.

1. **Launched brute-force using Hydra: `**hydra -l testuser -P password.txt ssh://127.0.0.1**`(for same machine since I am doing on local host)**

A computer screen shot of a program

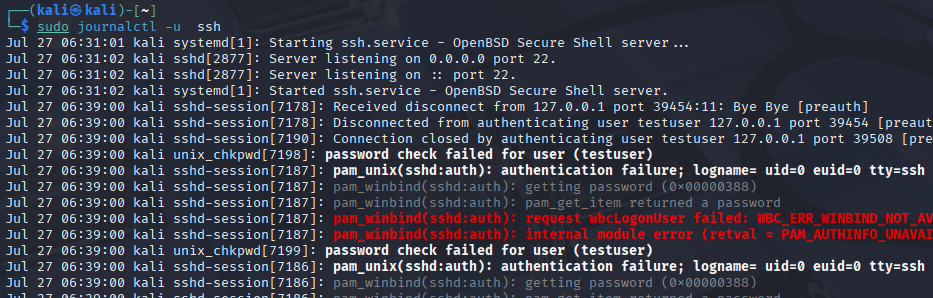
AI-generated content may be incorrect.

1. **Captured real-time logs using:** `sudo journalctl -f | grep sshd**`(live monitoring)**

A computer screen shot of a number

AI-generated content may be incorrect.

**You can check full history of ssh logs using: `**sudo journalctl -u ssh**`**



1. **Observed failed login attempts followed by successful login log from SSH in logs (simulating SOC detection)**

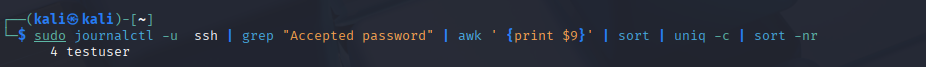
A computer screen shot of a number

AI-generated content may be incorrect.

1. **Applied the following filters:**

* **To detect which IP addresses are attempting brute force and how many times using:**

sudo journalctl -u ssh | grep "Accepted password" | awk '{print $9}' | sort | uniq -c | sort -nr

****

grep "Accepted password" → gets only successful login lines

awk '{print $9}' → picks the 9th word, which is usually the username

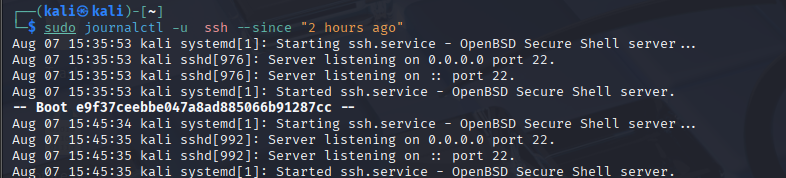
sort → sorts usernames alphabetically

uniq -c → counts how many times each unique username appeared

sort -nr → sorts counts in reverse order (most frequent first)

* **Filter logs from past two hours using:**

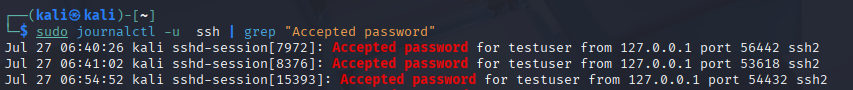
sudo journalctl -u ssh --since "2 hours ago"



Shows all SSH-related log entries from the past 2 hours, helping you monitor recent or ongoing activity in real time.

* **Filter Accepted Password using:**

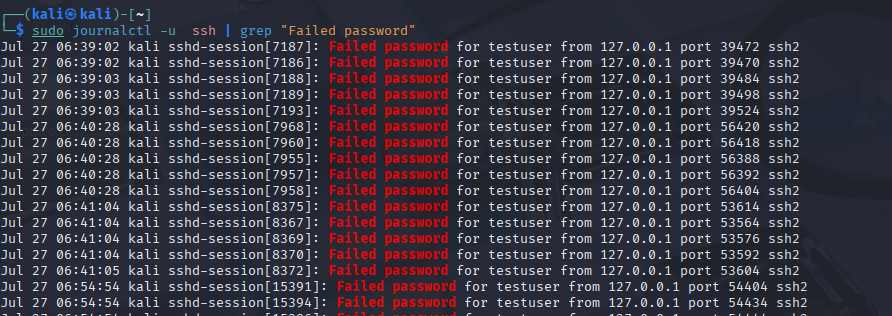
sudo journalctl -u ssh | grep "Accepted password"



Shows all successful password-based logins. Helps identify if any unauthorized users gained access.

* **Filter Failed Password using:**

sudo journalctl -u ssh | grep "Failed password"



Lists incorrect login attempts. A high frequency may indicate a brute-force attack.

* **Filter Connection from using:**

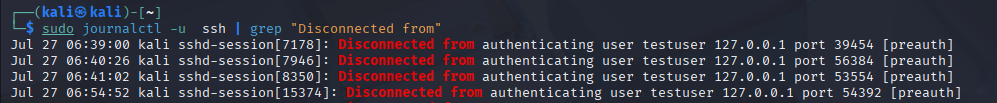
sudo journalctl -u ssh | grep "Connection from"



Logs initial connection attempts, even if they don’t proceed to authentication. Useful for detecting scanning or probing activity.

* **Filter Disconnected from using:**

sudo journalctl -u ssh | grep "Disconnected from"



Displays when a user disconnects — helpful for tracking session durations or spotting failed logins followed by disconnections.

* **Filter Invalid User using:**

sudo journalctl -u ssh | grep "Invalid user"



Logs attempts made with non-existent usernames (e.g., bots trying admin, test, etc.). Useful for detecting dictionary attacks.

* **Filter Accepted Public Key using:**

sudo journalctl -u ssh | grep "Accepted publickey"



Shows users who logged in via SSH key-based authentication instead of passwords. Good to check for authorized automation or DevOps access.

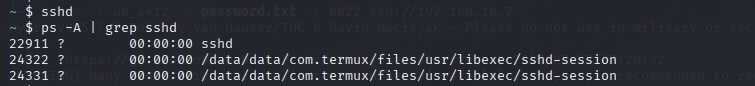
1. **Deleted user from Kali after test: `**sudo deluser testuser --remove-home**`**

A computer screen shot of white text

AI-generated content may be incorrect.

1. **Bonus Test:**

**Installed Termux on Android and ran SSH server using `sshd`**



1. **Discovered Termux username using `**whoami**` and IP with `**ip route | grep wlan0**`**
2. **Ran Hydra against Android phone: `**hydra -l u0\_aXXX -P password.txt -s 8022 ssh://<phone\_ip>**`**

A screen shot of a computer code

AI-generated content may be incorrect.

(Note**:** Logs were not monitored on phone; used for understanding mobile SSH brute-forcing only)

1. **Stopped SSH on Termux: `**pkill sshd**` and deleted password list**

A close up of a computer screen

AI-generated content may be incorrect.

# SOC Monitoring Insight

Logs from Kali Linux were monitored live using `journalctl`, showing multiple failed logins attempts and a successful SSH login. Their history was also monitored and filters were applied to find failed, passed attempts etc. This is a signature of brute-force activity. Such logs can trigger alerts in a SIEM platform used by SOC analysts. In contrast, SSH on Termux was used purely for fun testing and logs were not monitored.

# Ethical Disclaimer

This lab was performed in a controlled environment with personal devices. No unauthorized access was attempted. All simulations were conducted for educational purposes only.

# Cleanup Actions

- Kali: Deleted `testuser`, stopped ssh service if not needed.  
- Android: Ran `pkill sshd` in Termux, deleted password list, reset password.

# Conclusion

This lab demonstrated how SSH brute-force attacks are carried out and how SOC teams detect such events through log analysis. Two platforms were tested – a Linux system (with logging) and an Android phone (for tool behavior). This dual simulation built both attacker and defender perspectives in real-world conditions.