# Information Security Project

# (SSH Brute Force Attack Detection and Prevention on Kali Linux)



# Session 2023 - 2027

# Submitted by:

Munees Tariq 2023-CS-32

Asad Tayyab 2023-CS-14

Muhammad Uzair 2023-CS-22

# Supervised by:

Mam. Faiza Iqbal

# Course:

Information Security

# Department of Computer Science

# University of Engineering and Technology, Lahore, Pakistan

# Project Title:

# SSH Brute Force Attack Detection and Prevention on Kali Linux

# Description:

This project provides a GUI-based security tool for hardening Linux systems against SSH brute force attacks and improving overall system security. The application combines two Python scripts:

* **gui-app.py:** A Tkinter-based GUI tool that offers

1. SSH security configuration (limiting auth attempts, sessions, and grace time)
2. Password policy enforcement (complexity requirements, length rules)
3. Secure user creation with password validation
4. Password generation functionality

* **pass-gen.py:** A standalone password generator that creates strong random passwords

# Problems and Definition:

**1. SSH Brute Force Attacks**

Repeated automated attempts to guess SSH login credentials (username/password) to gain unauthorized access.  
**Impact:** Can lead to system compromise, data theft, or malware deployment.

**2. Weak Password Policies**

Lack of strong password requirements (short length, no complexity) makes brute-forcing easier.  
**Impact:** Easier for attackers to guess or crack passwords.

**3. Excessive Authentication Attempts**

No limit on failed login attempts allows attackers to try unlimited password combinations.  
**Impact:** Increases success rate of brute force attacks.

**4. Unlimited SSH Sessions**

No restriction on concurrent SSH sessions per user.  
**Impact:** Attackers can open multiple connections for exploitation.

**5. Long Login Grace Time**

Extended time window for users to authenticate before session timeout.  
**Impact:** Gives attackers more time to guess passwords.

**6. Default or Weak SSH Configurations**

SSH service running with insecure default settings (e.g., allowing root login, weak encryption).  
**Impact:** Increases vulnerability to attacks.

**7. Lack of Automated Security Hardening**

Manual security configurations are time-consuming and error-prone.  
**Impact:** System remains vulnerable due to misconfigurations or oversight.

# Security Effectiveness:

The program implements **advance SSH brute-force prevention** and **password security measures**, but with **some limitations**. The SSH hardening rules (1,MaxAuthTries 2,MaxSessions 3,LoginGraceTime) effectively slow down brute-force attacks, while the password policies enforce minimal complexity (6+ chars, mixed cases, digits). The tool is **good for personal Kali Linux hardening**. Overall, it provides a **decent baseline** but needs a little bit more advancements.

# Objectives:

The goal of this project is to simulate a brute force attack on an SSH service using Hydra, analyze its impact, and implement a solution to detect and protect against such attacks. A custom app was also developed to monitor SSH login attempts in real-time.

# Tools & Environment:

| Tool | Purpose |
| --- | --- |
| Kali Linux | Attacker machine |
| OpenSSH Server | Target service |
| Hydra | Brute force tool |
| Python | App development |
| /var/log/auth.log | Log file to monitor attempts |

# Vulnerability & Exploit:

Exploit: Brute Force SSH Login

SSH is a secure protocol but vulnerable when weak passwords are used.

Hydra can attempt thousands of passwords quickly, leading to unauthorized access.

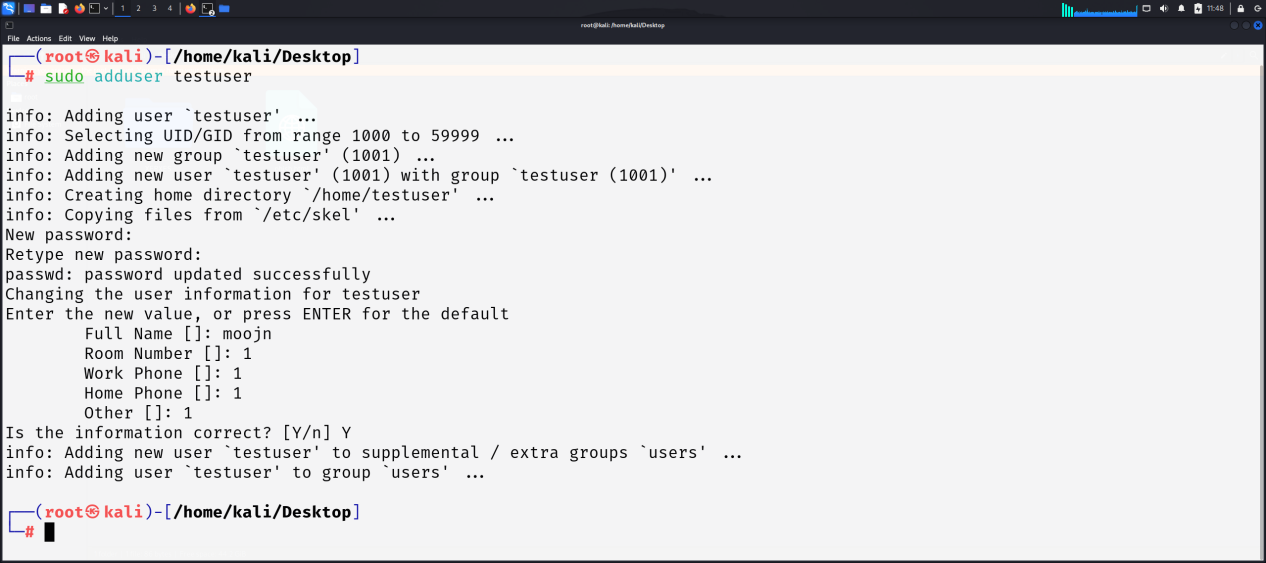
Target:

SSH server running on Kali Linux

Weak user testuser with password 1234

# Step-by-Step Execution:

Step 1: Create a Weak User



Step 2: Start SSH Server

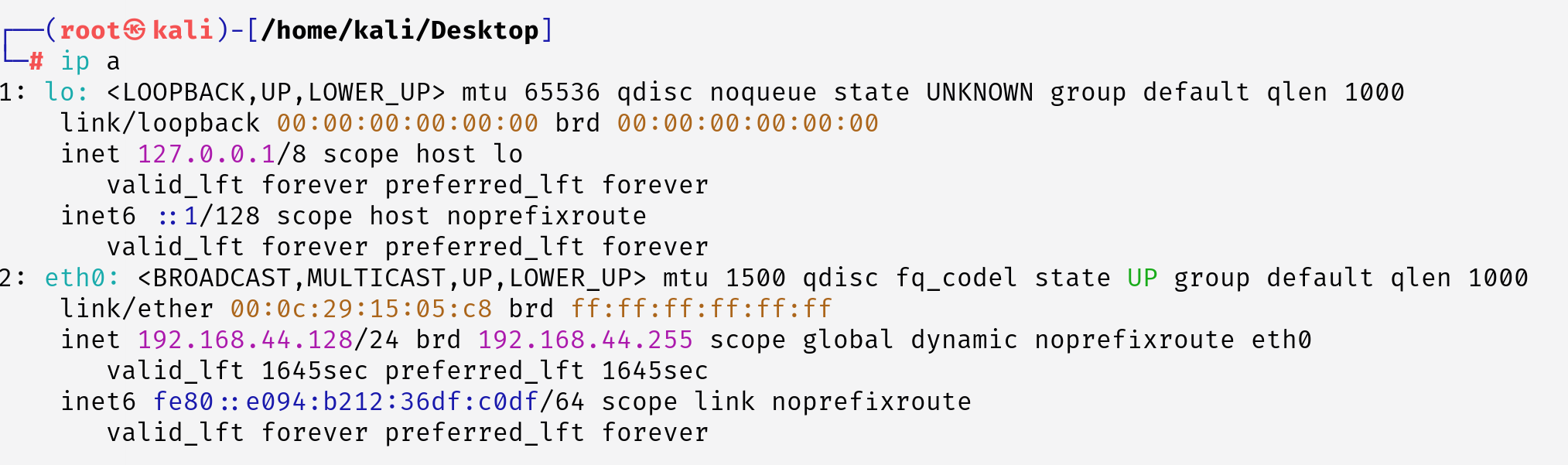
**Packages Used:**

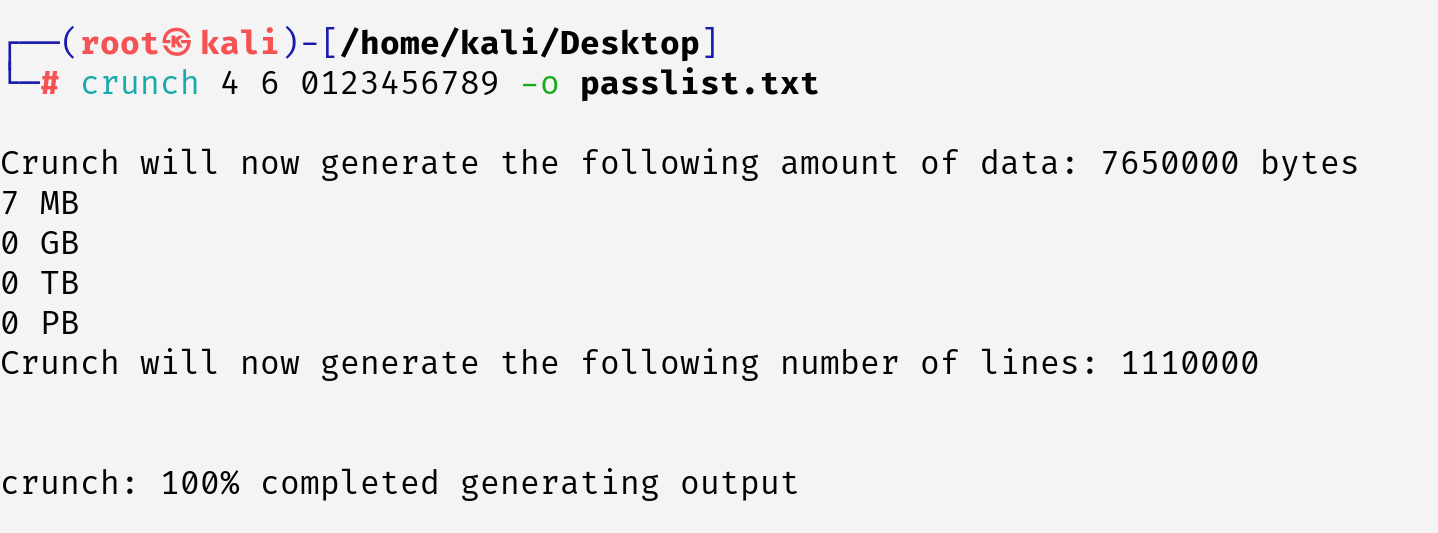
sudo apt update

sudo apt install openssh-server

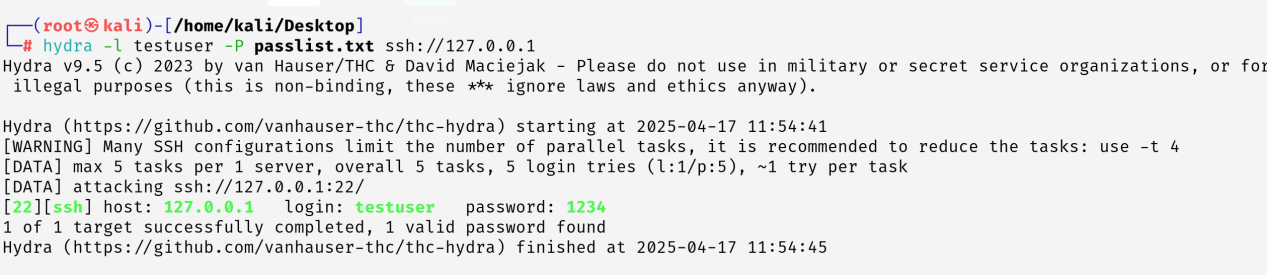
sudo systemctl start ssh



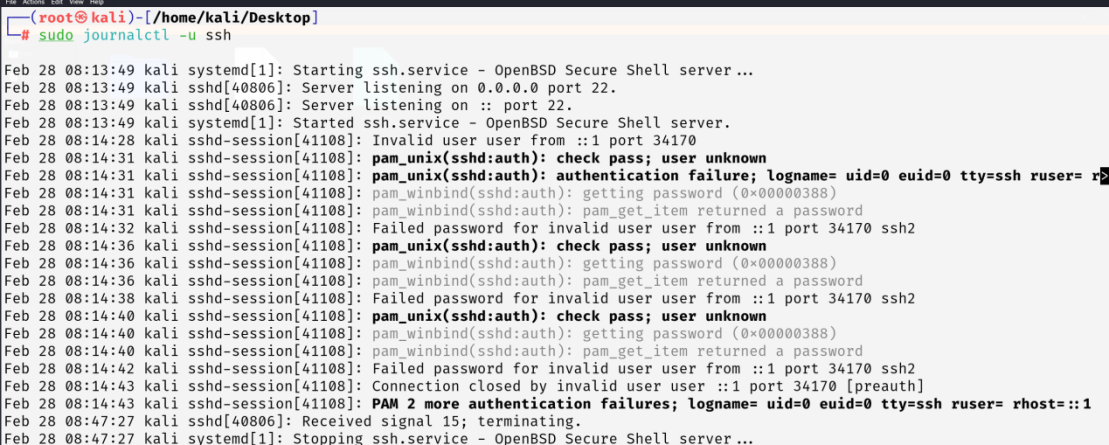
Step 3: Check IP Address

Step 4: Create a Password List

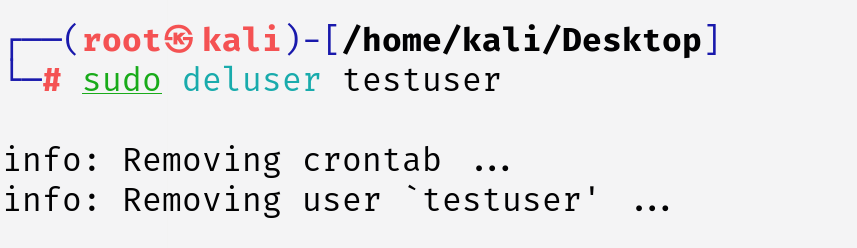
This creates a password list file with a minimum of 4 characters and a maximum of 6, containing all combinations of numbers from 0 to 9.

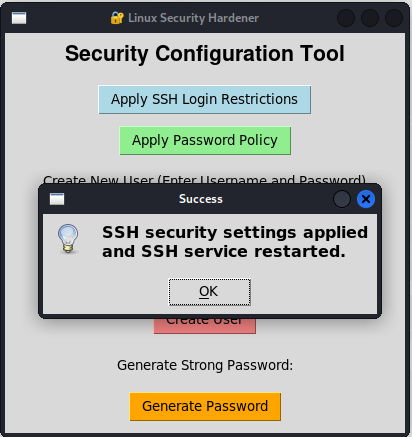
Step 5: Run Hydra

# Data Analysis:

Logs Analysis:

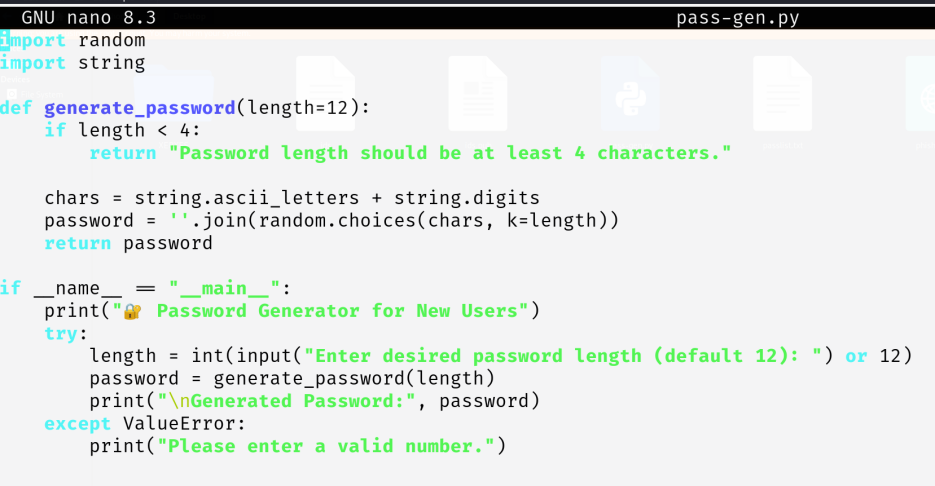
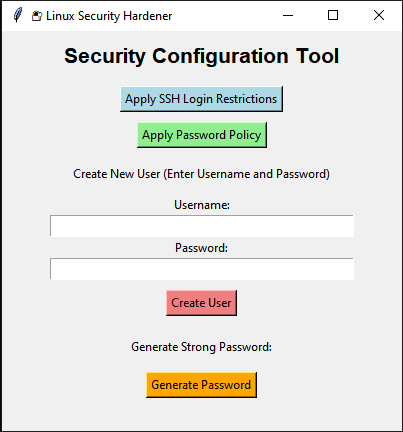
# Solution & Protection:

1. Remove the weak user
2. Use SSH Key Authentication Instead
3. Configure SSH security settings



What it does:

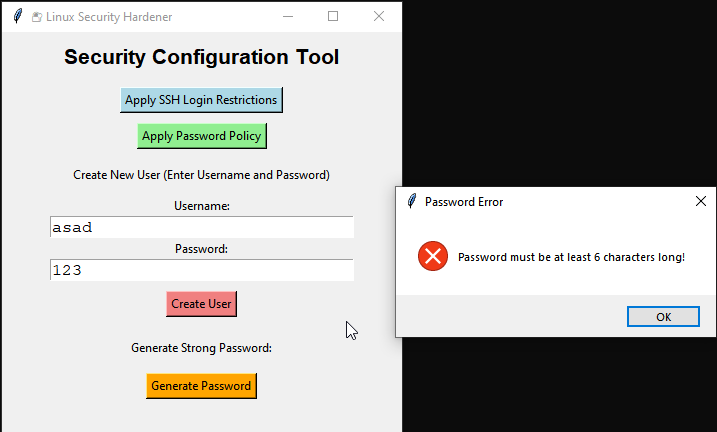
* Modifies /etc/ssh/sshd\_config for enhanced server protection.
* Limits authentication attempts (MaxAuthTries 1) to prevent brute-force attacks.
* Restricts concurrent sessions per connection (MaxSessions 2) to minimize abuse.
* Enforces short login grace time (LoginGraceTime 30) to limit unauthorized access.
* Hardens SSH service, ensuring access only for legitimate users within a brief time frame.

1. Password Generator for secure pass
2. Removing weak users using python app

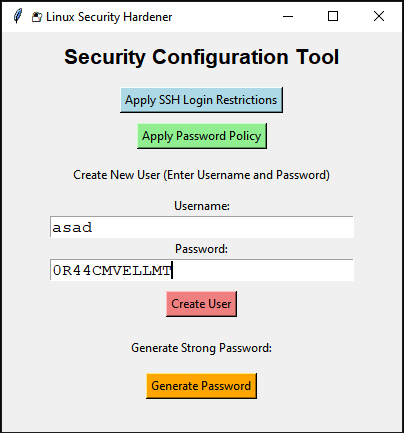
Gui App:

The python app allows to make users directly and easily from the UI provided and with proper password policies listed below

* Allowing only Strong passwords



It does not allow users to create weak password with length less than 6

* Suggesting some strong passwords

It also generates some strong passwords for the user that are not easy to be cracked using brute force

# Conclusion:

A detection tool was developed in Python to help monitor and alert against brute-force attempts in real-time. Preventive measures, including disabling password-based login, were implemented to protect the system.

# Future Plans:

In future iterations, the project can be enhanced by adding

* Logging functionality to track all actions performed through the GUI for audit purposes.
* Integration with LDAP or Active Directory for centralized user management could also be implemented.
* Additional security features such as firewall configuration, automatic updates, and intrusion detection setup can further strengthen system protection.
* To improve usability, role-based access controls and a dashboard summary of applied settings could be included.
* Cross-platform support and packaging the tool as a standalone executable would also broaden its usability.