**INTRODUCTION TO INFORMATION SECURITY AND FORENSIC**

**LAB NO 1**



**SUBMITTED TO:SIR MOEEZ JAVED**

**LAB NO 1**

**AMNA SALEEM HAYAT**

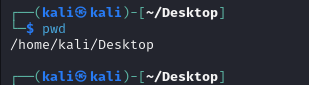
**COMMANDS:**

1. **Cd:** move into directory

Cd Desktop



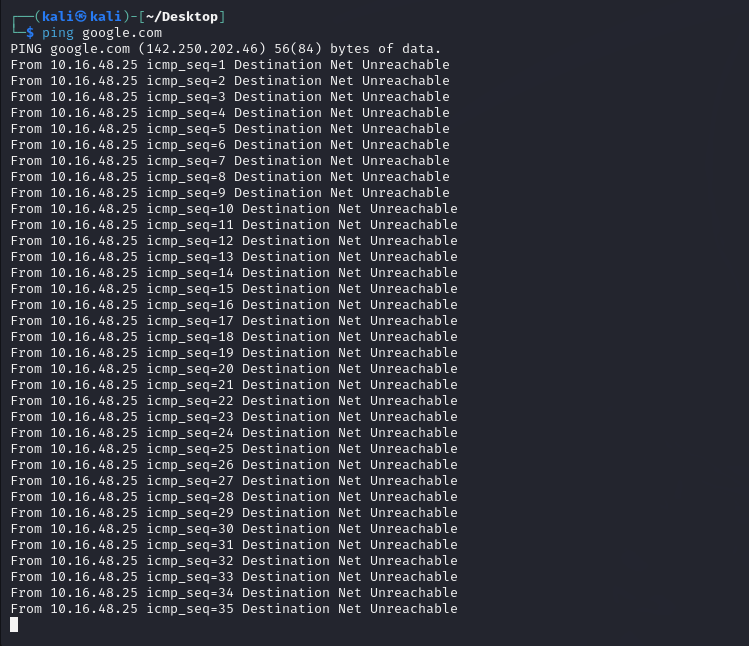
1. **Pwd:** location



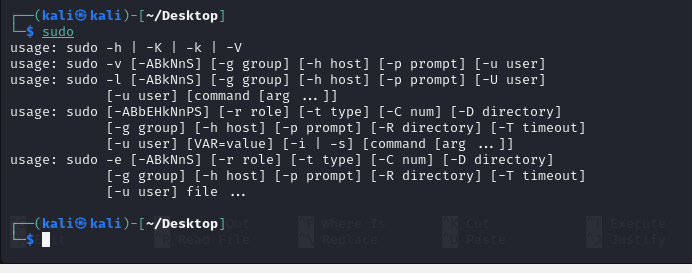
1. Ls: show files on directory



1. ping : check internet connection by sending packet

Ping google

1. sudo: move to root user

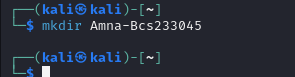


1. nano:



Mkdir:folder creation

Mkdir amna bcs233045



**WALKTHROUGH TASK**

Step 1: Data Collection Create a simulated log file to represent raw data. Open a terminal and enter:

1. Start with cat>auth.log<EOF

**COMMAND**

cat > auth.log <<'EOF'

2025-09-02 09:00:01 host sshd[1234]: Failed password for invalid user admin from 192.168.1.100 port

53718

2025-09-02 09:01:05 host sshd[1234]: Failed password for invalid user admin from 192.168.1.100 port

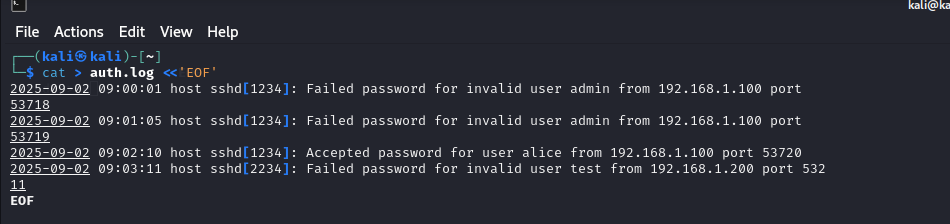
53719

2025-09-02 09:02:10 host sshd[1234]: Accepted password for user alice from 192.168.1.100 port 53720

2025-09-02 09:03:11 host sshd[2234]: Failed password for invalid user test from 192.168.1.200 port 532

11

EOF



**Step 2:** Information Extraction with Python (7 minutes) Write and run a Python script to parse the log and extract patterns, transforming data into information. Create a file walkthrough.py using:

**COMMAND**

cat > walkthrough.py <<'PY'

#!/usr/bin/env python3

import re

from collections import defaultdict

logfile = 'auth.log'

pattern = re.compile(r"^(?P<timestamp>\d{4}-\d{2}-\d{2} \d{2}:\d{2}:\d{2}).\*?(Failed password|Accept

ed password).\*from (?P<ip>\d+\.\d+\.\d+\.\d+).\*(?:port (?P<port>\d+))")

results = []

with open(logfile, 'r') as f:

for line in f:

m = pattern.search(line)

if m:

status = 'Failed' if 'Failed password' in line else 'Accepted'

results.append({'timestamp': m.group('timestamp'), 'ip': m.group('ip'), 'status': status})

# Print extracted information (Information layer)

print("Extracted entries:")

for r in results:

print(f"{r['timestamp']} {r['ip']} {r['status']}")

# Aggregate per IP (helpful for knowledge layer)

counts = defaultdict(lambda: {'Failed': 0, 'Accepted': 0})

for r in results:

counts[r['ip']][r['status']] += 1

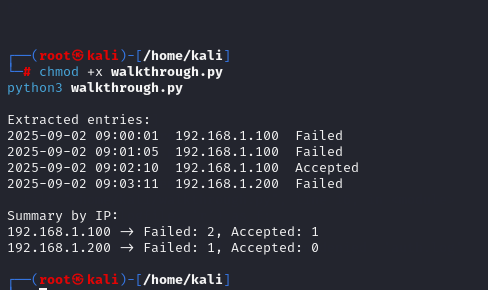
print('\nSummary by IP:')

for ip, c in counts.items():

print(f"{ip} -> Failed: {c['Failed']}, Accepted: {c['Accepted']}")

PY



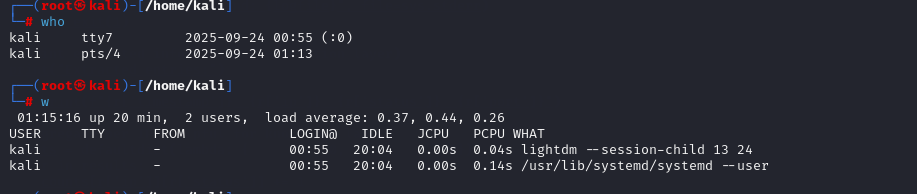


**3.COMANDS:**

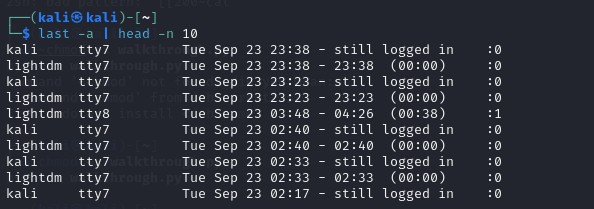
**Step 3: Knowledge Application (5 minutes)**

WHO

W



**Last -a | head -n 10**



**Step 4: Wisdom**

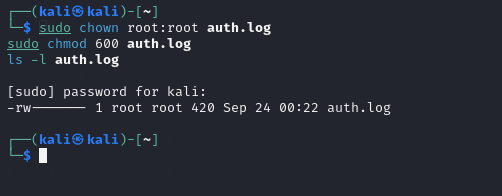
ha256sum auth.log > auth.log.sha256

cat auth.log.sha256

sha256sum: auth.log:

**2.sha256sum auth.log > auth.log.sha256**

**cat auth.log.sha256**

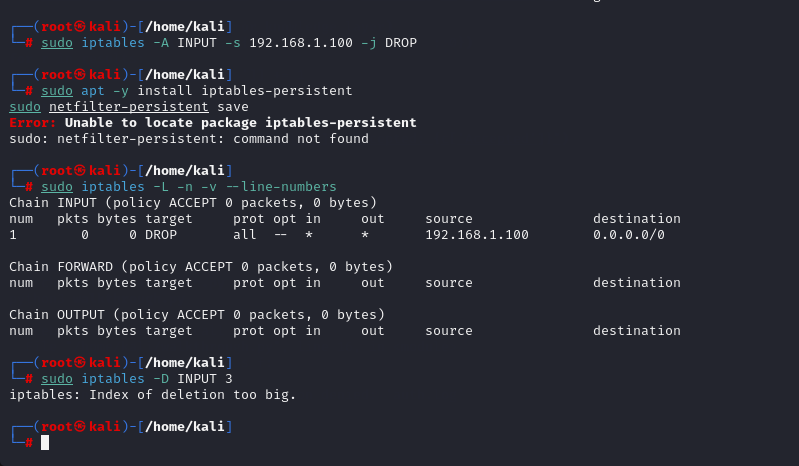


**1.sudo iptables -A INPUT -s 192.168.1.100 -j DROP**

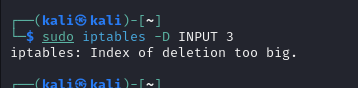
**2.** **sudo apt -y install iptables-persistent**

**sudo netfilter-persistent save**

**3.sudo iptables -L -n -v --line-numbers**



**sudo iptables -D INPUT 3**



****

****