Lab 4: Introduction to Operating Systems and File Systems in Linux

Objective:

This lab aims to provide students with foundational knowledge of Linux operating systems, focusing on file systems, directory structures, and essential commands. Students will learn about the benefits and challenges of using Linux for digital forensics and gain hands-on experience in managing files, performing system updates, and understanding networking commands in a Linux environment.

Part 1: Understanding Linux for Digital Forensics

1. Good and Bad Aspects of Linux in Digital Forensics:

Discuss the advantages of using Linux for digital forensics, such as its open-source nature, powerful command-line tools, and extensive community support.

Advantages of Linux in Digital Forensics:

- **Open-Source Nature:** Linux is free and open-source, allowing forensic investigators to inspect, modify, and customize tools for specific case requirements without licensing restrictions.
- **Powerful Command-Line Tools:** Linux provides numerous built-in forensic utilities which are essential for imaging, evidence analysis, and network monitoring.
- **Stability and Reliability:** Linux is less prone to crashes and malware, making it ideal for sensitive evidence handling.
- **Community and Documentation:** A vast global community offers support, tutorials, and scripts that help investigators troubleshoot and improve workflows

Address the potential challenges, including compatibility issues with certain file systems and software, and the learning curve for users new to Linux.

Disadvantages and Challenges:

- Compatibility Issues: Some proprietary forensic software (e.g., EnCase, FTK) and certain file systems like NTFS, APFS, or exFAT may not always be fully compatible or may require additional drivers or tools for full functionality.
- Learning Curve: Users new to Linux often find the command-line interface complex compared to Windows' graphical environment, leading to potential operational errors during investigations.
- Hardware and Software Limitations: Some specialized hardware or software tools are designed primarily for Windows, limiting direct use on Linux systems.

Task:

- Write a brief summary of at least two advantages and two challenges of using Linux for digital forensics.

Task: Advantages and Challenges of Using Linux for Digital Forensics

Advantages:

- **Open-source and customizable:** Linux allows investigators to modify and adapt forensic tools freely without licensing restrictions, making it ideal for flexible investigations.
- **Powerful forensic tools:** Linux provides built-in and pre-installed tools such as dd, grep, and Autopsy, enabling efficient data acquisition, analysis, and recovery.

Challenges:

- **Compatibility issues:** Some forensic tools and file systems (like NTFS or APFS) are not fully supported in Linux without additional drivers or software, limiting accessibility to certain evidence sources.
- **Steep learning curve:** For users accustomed to Windows, Linux's command-line environment can be difficult to master, potentially leading to operational mistakes during forensic analysis.

Part 2: Virtual File System and File Structure

1. Virtual File System (VFS):

- **Explain what a Virtual File System is and its role in Linux.**

  A Virtual File System (VFS**)** is an abstraction layer within the Linux kernel that provides a unified interface for interacting with different file systems. It allows applications and users to access files in the same way, regardless of the underlying file system type (e.g., ext4, NTFS, FAT32). The VFS manages how files are opened, read, written, and closed, translating these operations into specific actions for each supported file system.

  T he main role of the VFS is to ensure compatibility and flexibility enabling Linux to mount and use multiple file systems simultaneously. For example, it lets a Linux system read from an ext4 drive and a USB formatted with FAT32 without requiring the user to understand the differences between them. Essentially, VFS acts as a bridge between user-level processes and the low-level file system drivers, making Linux versatile and efficient in handling diverse storage types.

- **Discuss how VFS allows Linux to support multiple file systems seamlessly.**

  The **Virtual File System (VFS)** enables Linux to support multiple file systems seamlessly by acting as a **common interface** between user applications and the various file system implementations. Instead of applications needing to understand the specifics of each file system (like ext4, NTFS, FAT32, or XFS), they interact only with the VFS layer.

When a user performs an action such as opening or reading a file the VFS translates that request into the appropriate commands for the underlying file system. This allows Linux to mount and access different storage types simultaneously without compatibility issues. For instance, a Linux machine can read files from a Windows NTFS partition and write to an ext4 partition at the same time.

In essence, the VFS provides uniformity, flexibility, and interoperability, making it possible for Linux to manage various storage devices and formats efficiently under one unified system.

2. File Structure:

- Describe the hierarchical file structure in Linux and the significance of the root directory (/).

File Structure in Linux

Linux uses a **hierarchical file structure**, meaning all files and directories are organized in a tree-like format that begins at a single starting point called the **root directory (/)**. Every file and folder in the system branches out from this root, forming a structured and organized path.

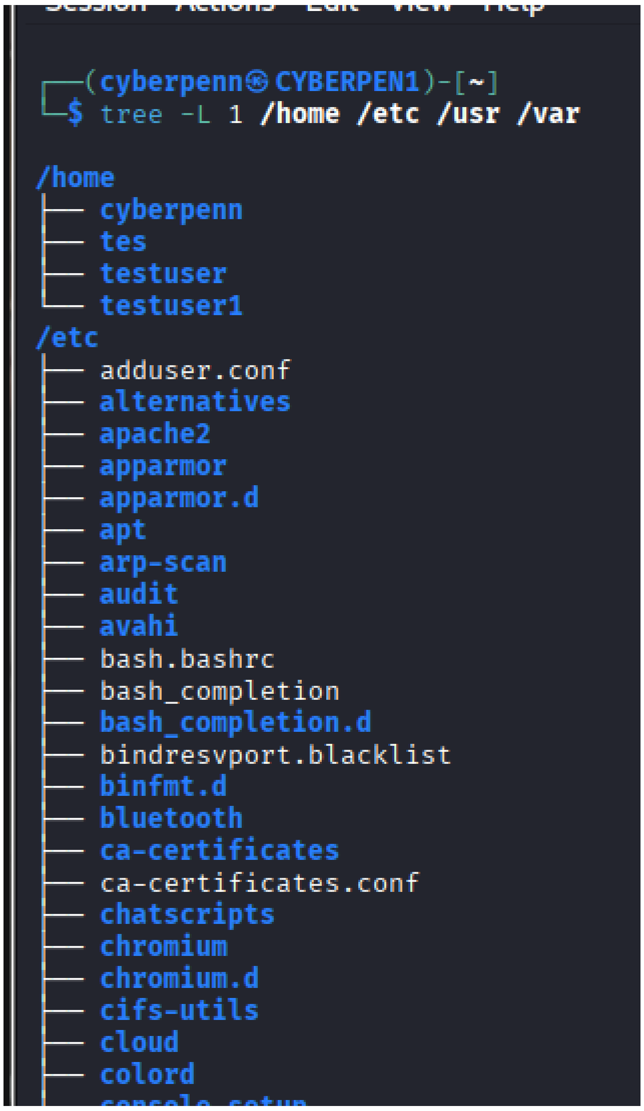
The **root directory (/**) is the topmost level of the Linux file system — all other directories, devices, and partitions are mounted under it. Key subdirectories include:

- **/home** – contains personal files for each user.
- **/etc** – stores system configuration files.
- **/bin** – holds essential user command binaries.
- **/var** – contains variable data like logs and temporary files.
- **/dev** – represents hardware devices as files.

The hierarchical structure ensures efficient organization, easy navigation, and consistent access control. The root directory's significance lies in its role as the **foundation of the entire Linux file system**, ensuring that all resources are accessible through a unified structure.

Task:

- Create a diagram representing the Linux file structure, highlighting key directories such as /home, /etc, /usr, and /var.



```
┌──(cyberpenn㉿CYBERPEN1)-[~]
└─$ tree -L 1 /home /etc /usr /var

/home
├── cyberpenn
├── tes
├── testuser
└── testuser1
/etc
├── adduser.conf
├── alternatives
├── apache2
├── apparmor
├── apparmor.d
├── apt
├── arp-scan
├── audit
├── avahi
├── bash.bashrc
├── bash_completion
├── bash_completion.d
├── bindresvport.blacklist
├── binfmt.d
├── bluetooth
├── ca-certificates
├── ca-certificates.conf
├── chatscripts
├── chromium
├── chromium.d
├── cifs-utils
├── cloud
├── colord
```

Part 3: Path and Path Variable

1. Path Variable:

- Define what the path variable is in Linux and its importance in executing commands.

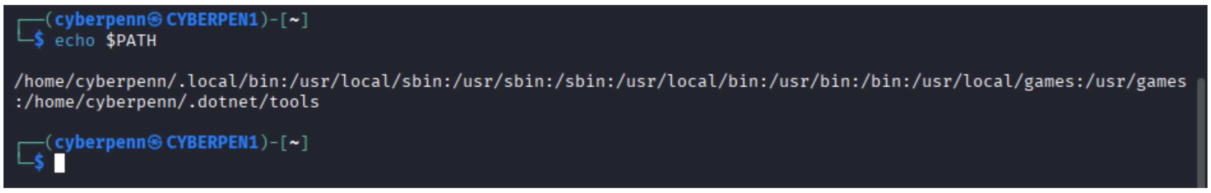
Path Variable in Linux

- The **PATH variable** in Linux is an **environment variable** that specifies a list of directories where the system looks for executable files when a command is entered in the terminal

- The PATH variable is essential because it allows users to run commands without needing to type their full file paths every time. For instance, instead of typing `/bin/ls`, a user can simply type `ls` because `/bin` is included in the PATH.
- this variable helps improve efficiency, convenience, and flexibility in the command-line environment

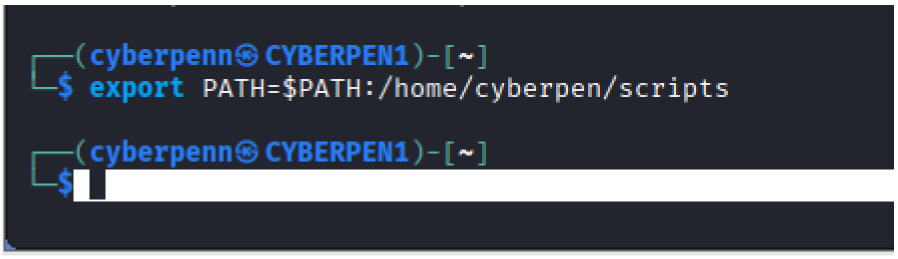
Task:

- Display the current path variable by executing the command: echo $PATH



```
┌──(cyberpenn㉿CYBERPEN1)-[~]
└─$ echo $PATH
/home/cyberpenn/.local/bin:/usr/local/sbin:/usr/sbin:/sbin:/usr/local/bin:/usr/bin:/bin:/usr/local/games:/usr/games
:/home/cyberpenn/.dotnet/tools

┌──(cyberpenn㉿CYBERPEN1)-[~]
└─$ 
```

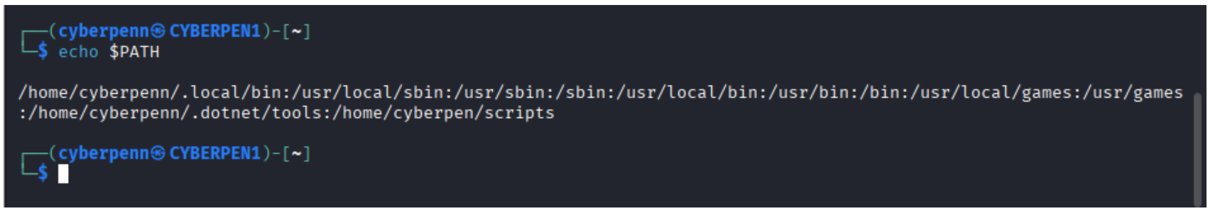
- Add a new directory (e.g., /home/yourusername/scripts) to the path variable temporarily for the session: export PATH=$PATH:/home/yourusername/scripts



```
┌──(cyberpenn㉿CYBERPEN1)-[~]
└─$ export PATH=$PATH:/home/cyberpen/scripts

┌──(cyberpenn㉿CYBERPEN1)-[~]
└─$ 
```

Task:

- Confirm the new path by executing the echo $PATH command again.

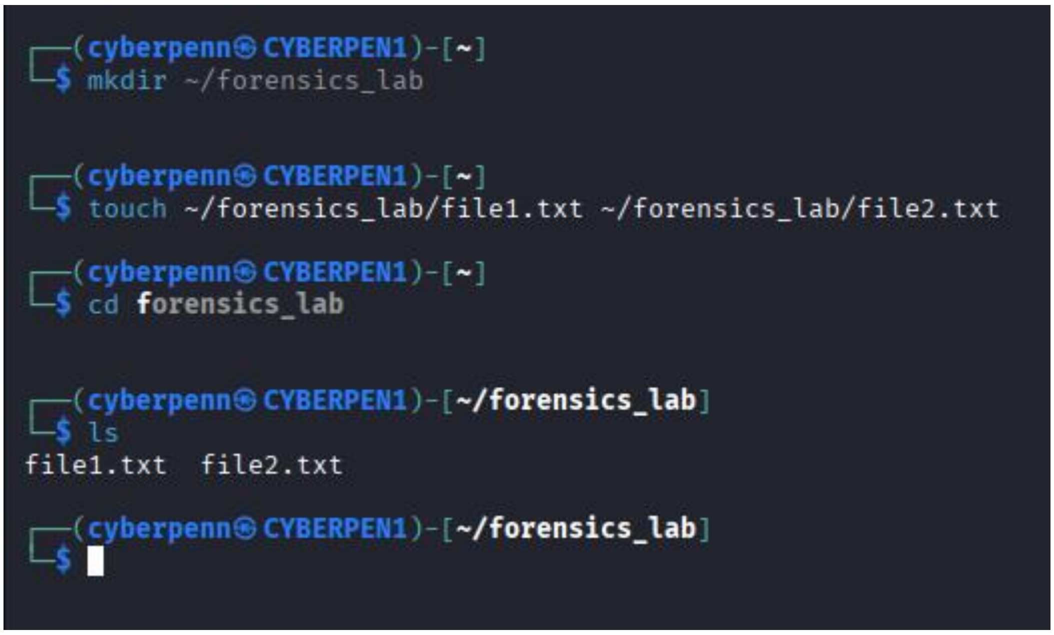


```
┌──(cyberpenn㉿CYBERPEN1)-[~]
└─$ echo $PATH
/home/cyberpenn/.local/bin:/usr/local/sbin:/usr/sbin:/sbin:/usr/local/bin:/usr/bin:/bin:/usr/local/games:/usr/games
:/home/cyberpenn/.dotnet/tools:/home/cyberpen/scripts

┌──(cyberpenn㉿CYBERPEN1)-[~]
└─$ 
```

Part 4: Essential Linux Commands

1. Creating Folders and Files:

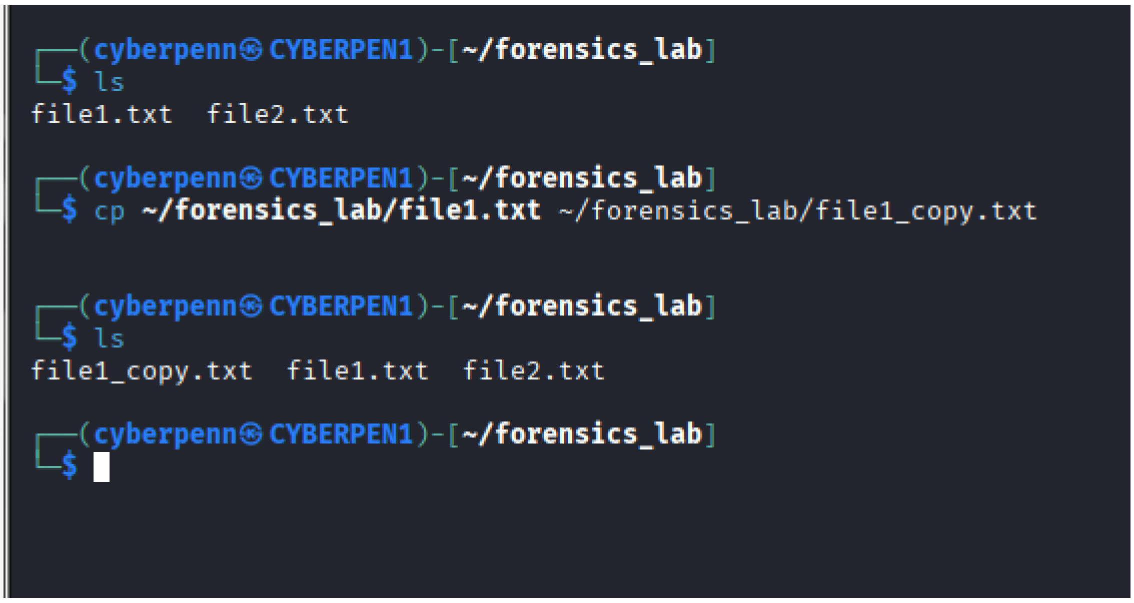
- Use the following commands to create a new directory named forensics_lab and create two text files within it:
- mkdir ~/forensics_lab touch ~/forensics_lab/file1.txt ~/forensics_lab/file2.txt



```
┌──(cyberpenn㉿CYBERPEN1)-[~]
└─$ mkdir ~/forensics_lab


┌──(cyberpenn㉿CYBERPEN1)-[~]
└─$ touch ~/forensics_lab/file1.txt ~/forensics_lab/file2.txt

┌──(cyberpenn㉿CYBERPEN1)-[~]
└─$ cd forensics_lab


┌──(cyberpenn㉿CYBERPEN1)-[~/forensics_lab]
└─$ ls
file1.txt  file2.txt

┌──(cyberpenn㉿CYBERPEN1)-[~/forensics_lab]
└─$ 
```

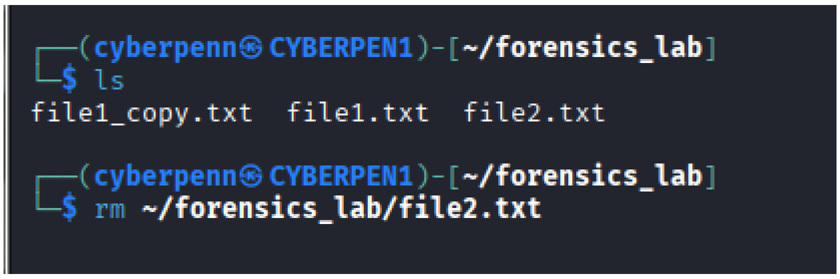
2. File Copying and Deletion:

- Copy file1.txt to a new file named file1_copy.txt: cp ~/forensics_lab/file1.txt ~/forensics_lab/file1_copy.txt



```
┌──(cyberpenn㉿CYBERPEN1)-[~/forensics_lab]
└─$ ls
file1.txt  file2.txt

┌──(cyberpenn㉿CYBERPEN1)-[~/forensics_lab]
└─$ cp ~/forensics_lab/file1.txt ~/forensics_lab/file1_copy.txt


┌──(cyberpenn㉿CYBERPEN1)-[~/forensics_lab]
└─$ ls
file1_copy.txt  file1.txt  file2.txt

┌──(cyberpenn㉿CYBERPEN1)-[~/forensics_lab]
└─$ 
```

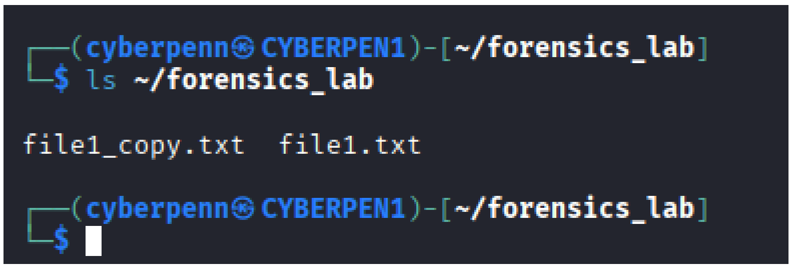
- Delete file2.txt: rm ~/forensics_lab/file2.txt



```
┌──(cyberpenn㉿CYBERPEN1)-[~/forensics_lab]
└─$ ls
file1_copy.txt  file1.txt  file2.txt

┌──(cyberpenn㉿CYBERPEN1)-[~/forensics_lab]
└─$ rm ~/forensics_lab/file2.txt
```

3. Task:

- Verify that the copy and deletion were successful by listing the contents of the forensics_lab directory: ls ~/forensics_lab

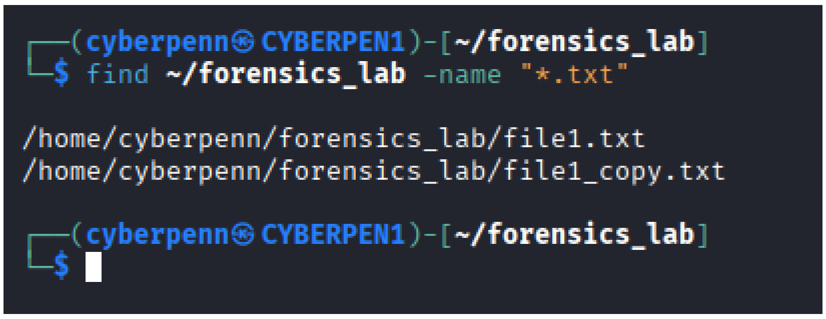


```
┌──(cyberpenn㉿CYBERPEN1)-[~/forensics_lab]
└─$ ls ~/forensics_lab

file1_copy.txt  file1.txt

┌──(cyberpenn㉿CYBERPEN1)-[~/forensics_lab]
└─$ 
```

Part 5: Searching for Information

1. Searching for Files:

- Use the find command to search for all .txt files in the forensics_lab directory: find ~/forensics_lab -name "*.txt"



```
┌──(cyberpenn㉿CYBERPEN1)-[~/forensics_lab]
└─$ find ~/forensics_lab -name "*.txt"

/home/cyberpenn/forensics_lab/file1.txt
/home/cyberpenn/forensics_lab/file1_copy.txt

┌──(cyberpenn㉿CYBERPEN1)-[~/forensics_lab]
└─$ 
```

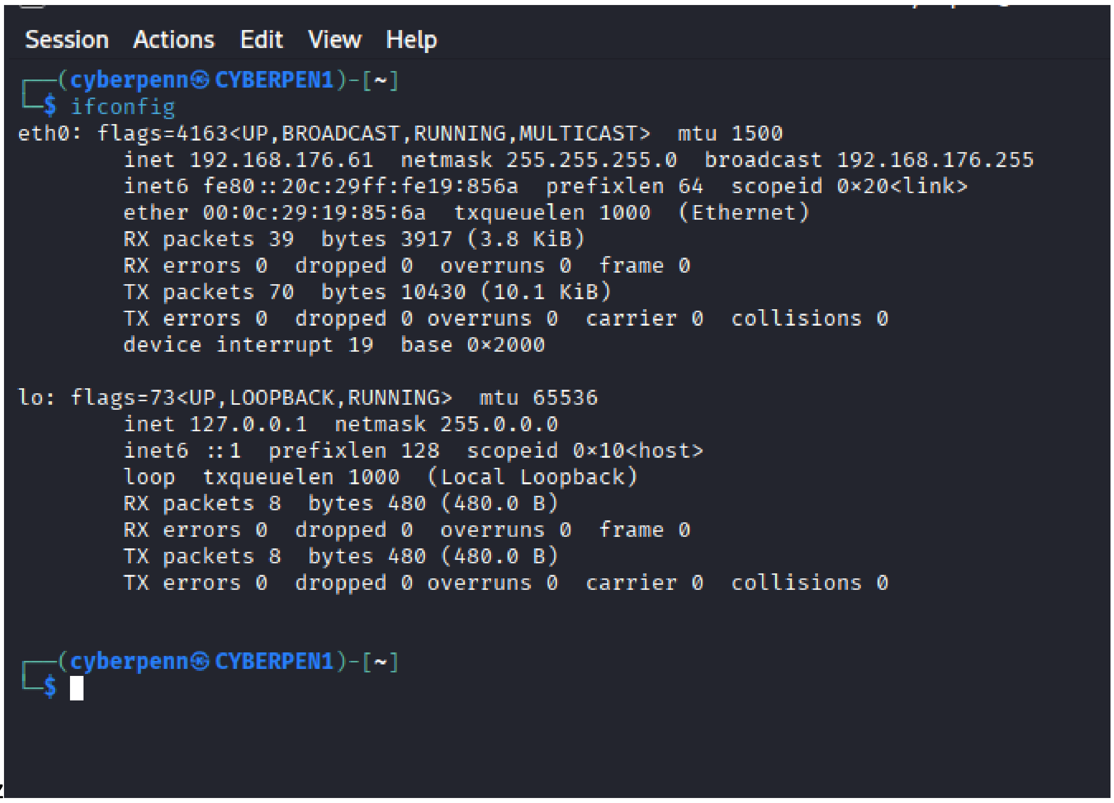
2. Task:

- Document the output of the command and explain what it shows.

Part 6: Networking in Linux

1. Basic Networking Commands:

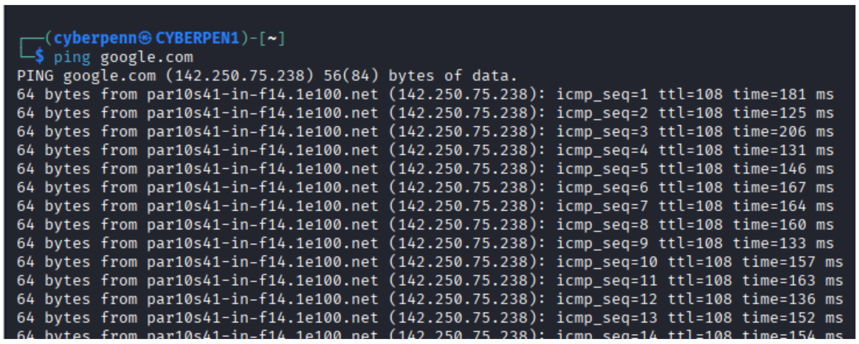
- Execute the following commands to gather network information:
  - ifconfig: Display network interfaces and their configurations.



```
┌──(cyberpenn㉿CYBERPEN1)-[~]
└─$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.176.61  netmask 255.255.255.0  broadcast 192.168.176.255
        inet6 fe80::20c:29ff:fe19:856a  prefixlen 64  scopeid 0x20<link>
        ether 00:0c:29:19:85:6a  txqueuelen 1000  (Ethernet)
        RX packets 39  bytes 3917 (3.8 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 70  bytes 10430 (10.1 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
        device interrupt 19  base 0x2000

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 8  bytes 480 (480.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 8  bytes 480 (480.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0


┌──(cyberpenn㉿CYBERPEN1)-[~]
└─$ 
```

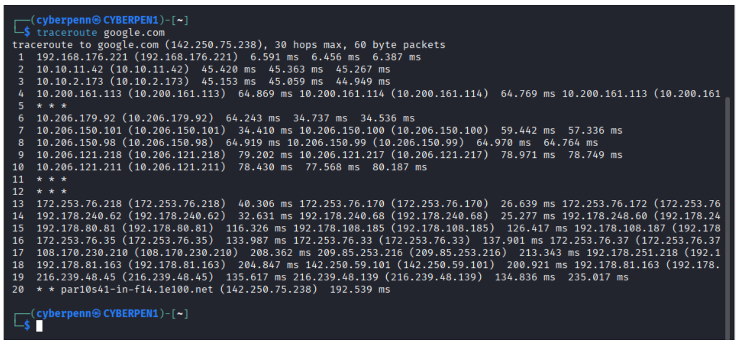
z

- ping google.com: Check connectivity to Google's servers.



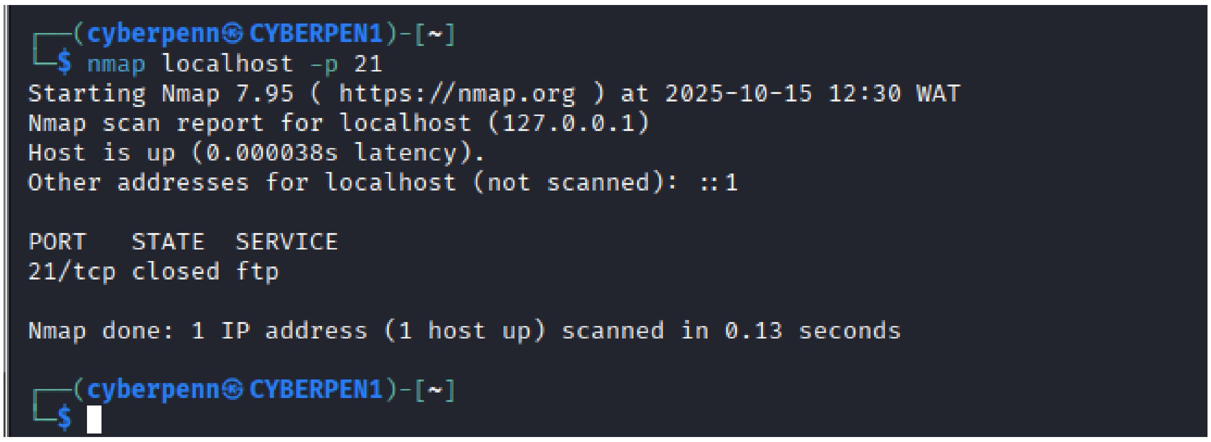
```
┌──(cyberpenn㉿CYBERPEN1)-[~]
└─$ ping google.com
PING google.com (142.250.75.238) 56(84) bytes of data.
64 bytes from par10s41-in-f14.1e100.net (142.250.75.238): icmp_seq=1 ttl=108 time=181 ms
64 bytes from par10s41-in-f14.1e100.net (142.250.75.238): icmp_seq=2 ttl=108 time=125 ms
64 bytes from par10s41-in-f14.1e100.net (142.250.75.238): icmp_seq=3 ttl=108 time=206 ms
64 bytes from par10s41-in-f14.1e100.net (142.250.75.238): icmp_seq=4 ttl=108 time=131 ms
64 bytes from par10s41-in-f14.1e100.net (142.250.75.238): icmp_seq=5 ttl=108 time=146 ms
64 bytes from par10s41-in-f14.1e100.net (142.250.75.238): icmp_seq=6 ttl=108 time=167 ms
64 bytes from par10s41-in-f14.1e100.net (142.250.75.238): icmp_seq=7 ttl=108 time=164 ms
64 bytes from par10s41-in-f14.1e100.net (142.250.75.238): icmp_seq=8 ttl=108 time=160 ms
64 bytes from par10s41-in-f14.1e100.net (142.250.75.238): icmp_seq=9 ttl=108 time=133 ms
64 bytes from par10s41-in-f14.1e100.net (142.250.75.238): icmp_seq=10 ttl=108 time=157 ms
64 bytes from par10s41-in-f14.1e100.net (142.250.75.238): icmp_seq=11 ttl=108 time=163 ms
64 bytes from par10s41-in-f14.1e100.net (142.250.75.238): icmp_seq=12 ttl=108 time=136 ms
64 bytes from par10s41-in-f14.1e100.net (142.250.75.238): icmp_seq=13 ttl=108 time=152 ms
64 bytes from par10s41-in-f14.1e100.net (142.250.75.238): icmp_seq=14 ttl=108 time=154 ms
```

- traceroute google.com: Trace the route to Google.



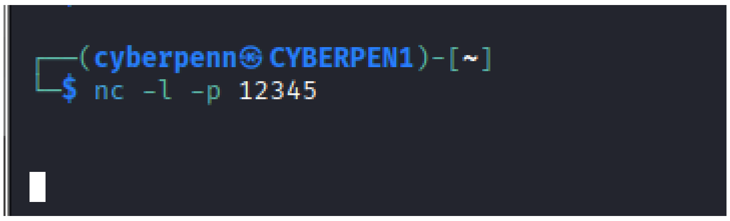
```
┌──(cyberpenn㉿CYBERPEN1)-[~]
└─$ traceroute google.com
traceroute to google.com (142.250.75.238), 30 hops max, 60 byte packets
 1  192.168.176.221 (192.168.176.221)  6.591 ms  6.456 ms  6.387 ms
 2  10.10.11.42 (10.10.11.42)  45.420 ms  45.363 ms  45.267 ms
 3  10.10.2.173 (10.10.2.173)  45.153 ms  45.059 ms  44.949 ms
 4  10.200.161.113 (10.200.161.113)  64.869 ms 10.200.161.114 (10.200.161.114)  64.769 ms 10.200.161.113 (10.200.161
 5  * * *
 6  10.206.179.92 (10.206.179.92)  64.243 ms  34.737 ms  34.536 ms
 7  10.206.150.101 (10.206.150.101)  34.410 ms 10.206.150.100 (10.206.150.100)  59.442 ms  57.336 ms
 8  10.206.150.98 (10.206.150.98)  64.919 ms 10.206.150.99 (10.206.150.99)  64.970 ms  64.764 ms
 9  10.206.121.218 (10.206.121.218)  79.202 ms 10.206.121.217 (10.206.121.217)  78.971 ms  78.749 ms
10  10.206.121.211 (10.206.121.211)  78.430 ms  77.568 ms  80.187 ms
11  * * *
12  * * *
13  172.253.76.218 (172.253.76.218)  40.306 ms 172.253.76.170 (172.253.76.170)  26.639 ms 172.253.76.172 (172.253.76
14  192.178.240.62 (192.178.240.62)  32.631 ms 192.178.240.68 (192.178.240.68)  25.277 ms 192.178.248.60 (192.178.24
15  192.178.80.81 (192.178.80.81)  116.326 ms 192.178.108.185 (192.178.108.185)  126.417 ms 192.178.108.187 (192.178
16  172.253.76.35 (172.253.76.35)  133.987 ms 172.253.76.33 (172.253.76.33)  137.901 ms 172.253.76.37 (172.253.76.37
17  108.170.230.210 (108.170.230.210)  208.362 ms 209.85.253.216 (209.85.253.216)  213.343 ms 192.178.251.218 (192.1
18  192.178.81.163 (192.178.81.163)  204.847 ms 142.250.59.101 (142.250.59.101)  200.921 ms 192.178.81.163 (192.178.
19  216.239.48.45 (216.239.48.45)  135.617 ms 216.239.48.139 (216.239.48.139)  134.836 ms  235.017 ms
20  * * par10s41-in-f14.1e100.net (142.250.75.238)  192.539 ms

┌──(cyberpenn㉿CYBERPEN1)-[~]
└─$ 
```

- Use nmap to scan the localhost for port 21: nmap localhost -p 21



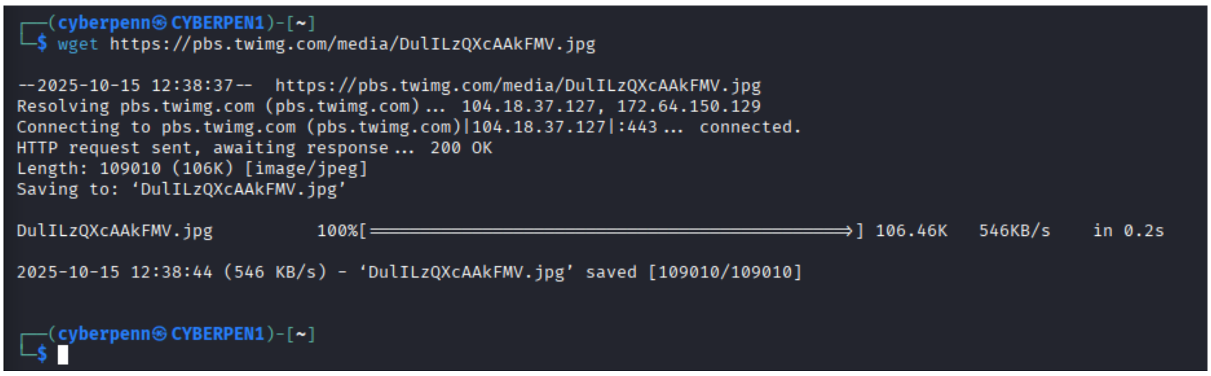
```
┌──(cyberpenn㉿CYBERPEN1)-[~]
└─$ nmap localhost -p 21
Starting Nmap 7.95 ( https://nmap.org ) at 2025-10-15 12:30 WAT
Nmap scan report for localhost (127.0.0.1)
Host is up (0.000038s latency).
Other addresses for localhost (not scanned): ::1

PORT   STATE  SERVICE
21/tcp closed ftp

Nmap done: 1 IP address (1 host up) scanned in 0.13 seconds

┌──(cyberpenn㉿CYBERPEN1)-[~]
└─$ 
```

- Use netcat to set up a simple TCP connection: nc -l -p 12345



```
┌──(cyberpenn㉿CYBERPEN1)-[~]
└─$ nc -l -p 12345

```

- Use wget to download an image: wget https://pbs.twimg.com/media/DulILzQXcAAkFMV.jpg



```
┌──(cyberpenn㉿CYBERPEN1)-[~]
└─$ wget https://pbs.twimg.com/media/DulILzQXcAAkFMV.jpg

--2025-10-15 12:38:37--  https://pbs.twimg.com/media/DulILzQXcAAkFMV.jpg
Resolving pbs.twimg.com (pbs.twimg.com)... 104.18.37.127, 172.64.150.129
Connecting to pbs.twimg.com (pbs.twimg.com)|104.18.37.127|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 109010 (106K) [image/jpeg]
Saving to: ‘DulILzQXcAAkFMV.jpg’

DulILzQXcAAkFMV.jpg         100%[===================================================>] 106.46K   546KB/s    in 0.2s

2025-10-15 12:38:44 (546 KB/s) - ‘DulILzQXcAAkFMV.jpg’ saved [109010/109010]


┌──(cyberpenn㉿CYBERPEN1)-[~]
└─$ 
```

2. Task: Document the outputs of these commands and explain the significance of the information provided.

- **Ifconfig:** shows network interfaces and their settings (IP addresses, netmasks, MAC addresses, interface status, RX/TX counters). Identifies how the host is connected to a network, which interfaces were active, and basic traffic counters.
- **ping google.com:** tests reachability and round-trip time to a remote host; reports packets sent/received, packet loss, and latency statistics. Verifies Internet connectivity

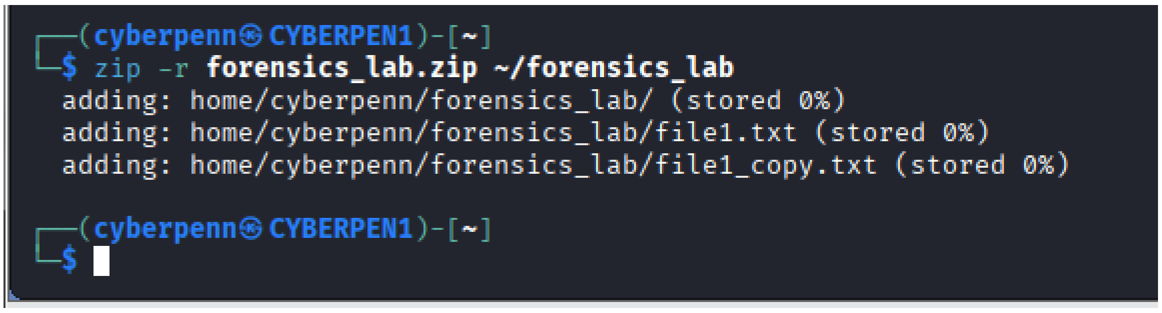
- **traceroute google.com:** lists each network hop between the host and the destination, with per-hop latency. Reveals the packet route and intermediate networks (ISPs, gateways).
- **nmap localhost -p 21**: scans the specified port on the local machine and reports its state (open/closed/filtered) and service information if available.
  Identifies active services and potential attack surfaces on the host
- **nc -l 12345: (netcat listener)** opens a TCP port and waits for an incoming connection; produces no output until a client connects or data is exchanged. Demonstrates a socket listening state and enables simple data transfer/testing;
- **wget**: downloads a remote file and reports connection steps (DNS resolution, HTTP response code), file size, transfer progress, and saved filename**.**

For each command, save the exact stdout/stderr (or take screenshots) and record timestamps and file hashes where relevant; these artifacts (command output, saved files, hashes) strengthen evidence integrity and the lab report.

Part 7: File Management

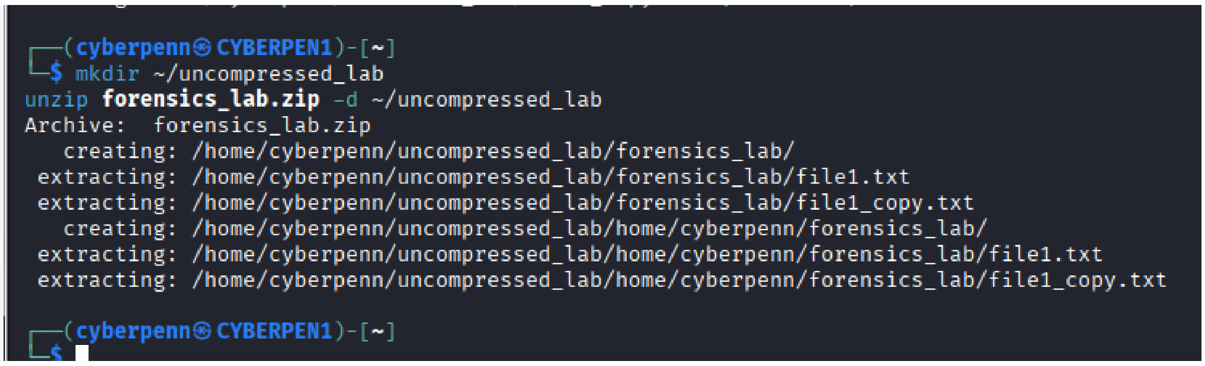
1. Zipping and Unzipping Files:

- Create a compressed archive of the forensics_lab directory: zip -r forensics_lab.zip ~/forensics_lab



```
┌──(cyberpenn㉿CYBERPEN1)-[~]
└─$ zip -r forensics_lab.zip ~/forensics_lab
  adding: home/cyberpenn/forensics_lab/ (stored 0%)
  adding: home/cyberpenn/forensics_lab/file1.txt (stored 0%)
  adding: home/cyberpenn/forensics_lab/file1_copy.txt (stored 0%)

┌──(cyberpenn㉿CYBERPEN1)-[~]
└─$ 
```

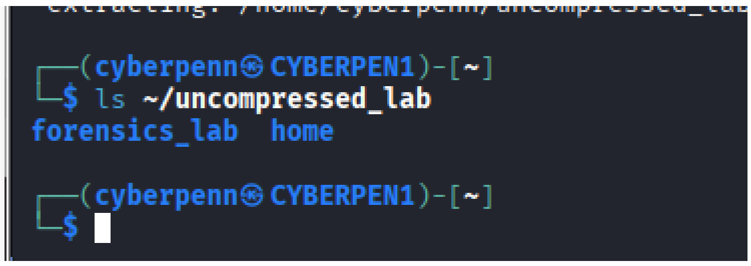
- Unzip the created archive into a new directory: mkdir ~/uncompressed_lab unzip forensics_lab.zip -d ~/uncompressed_lab



```
┌──(cyberpenn㉿CYBERPEN1)-[~]
└─$ mkdir ~/uncompressed_lab
unzip forensics_lab.zip -d ~/uncompressed_lab
Archive:  forensics_lab.zip
   creating: /home/cyberpenn/uncompressed_lab/forensics_lab/
 extracting: /home/cyberpenn/uncompressed_lab/forensics_lab/file1.txt
 extracting: /home/cyberpenn/uncompressed_lab/forensics_lab/file1_copy.txt
   creating: /home/cyberpenn/uncompressed_lab/home/cyberpenn/forensics_lab/
 extracting: /home/cyberpenn/uncompressed_lab/home/cyberpenn/forensics_lab/file1.txt
 extracting: /home/cyberpenn/uncompressed_lab/home/cyberpenn/forensics_lab/file1_copy.txt

┌──(cyberpenn㉿CYBERPEN1)-[~]
└─$ 
```

2. Task:

- Verify the contents of the uncompressed directory to ensure all files were restored correctly: ls ~/uncompressed_lab



```
┌──(cyberpenn㉿CYBERPEN1)-[~]
└─$ ls ~/uncompressed_lab
forensics_lab  home

┌──(cyberpenn㉿CYBERPEN1)-[~]
└─$ 
```

Part 8: Creating a Shell Script

1. Creating a Shell Script:

- Create a shell script named sys_info.sh with the following content:

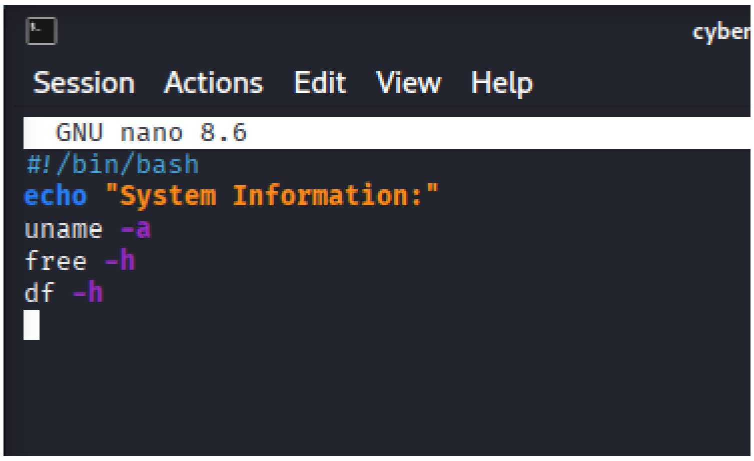
2. #!/bin/bash

3. echo "System Information:"

4. uname -a

5. free -h

df -h

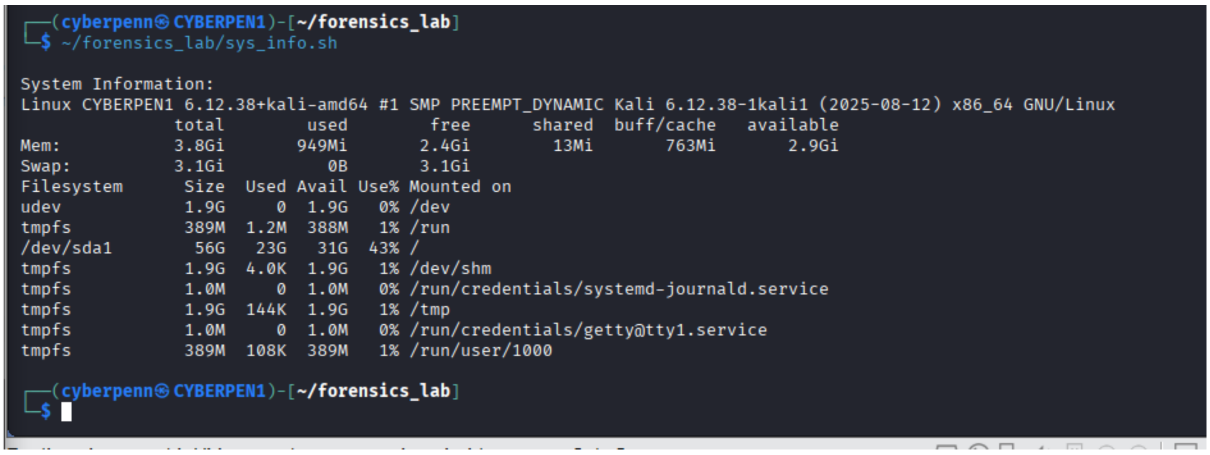


```
  GNU nano 8.6
#!/bin/bash
echo "System Information:"
uname -a
free -h
df -h
```

- Make the script executable: chmod +x ~/forensics_lab/sys_info.sh

Task:

- Run the script from the terminal: ~/forensics_lab/sys_info.sh



```
┌──(cyberpenn㉿CYBERPEN1)-[~/forensics_lab]
└─$ ~/forensics_lab/sys_info.sh

System Information:
Linux CYBERPEN1 6.12.38+kali-amd64 #1 SMP PREEMPT_DYNAMIC Kali 6.12.38-1kali1 (2025-08-12) x86_64 GNU/Linux
               total        used        free      shared  buff/cache   available
Mem:           3.8Gi       949Mi       2.4Gi        13Mi       763Mi       2.9Gi
Swap:          3.1Gi          0B       3.1Gi
Filesystem      Size  Used Avail Use% Mounted on
udev            1.9G     0  1.9G   0% /dev
tmpfs           389M  1.2M  388M   1% /run
/dev/sda1        56G   23G   31G  43% /
tmpfs           1.9G  4.0K  1.9G   1% /dev/shm
tmpfs           1.0M     0  1.0M   0% /run/credentials/systemd-journald.service
tmpfs           1.9G  144K  1.9G   1% /tmp
tmpfs           1.0M     0  1.0M   0% /run/credentials/getty@tty1.service
tmpfs           389M  108K  389M   1% /run/user/1000

┌──(cyberpenn㉿CYBERPEN1)-[~/forensics_lab]
└─$ 
```

- Document the output observed from executing the script.

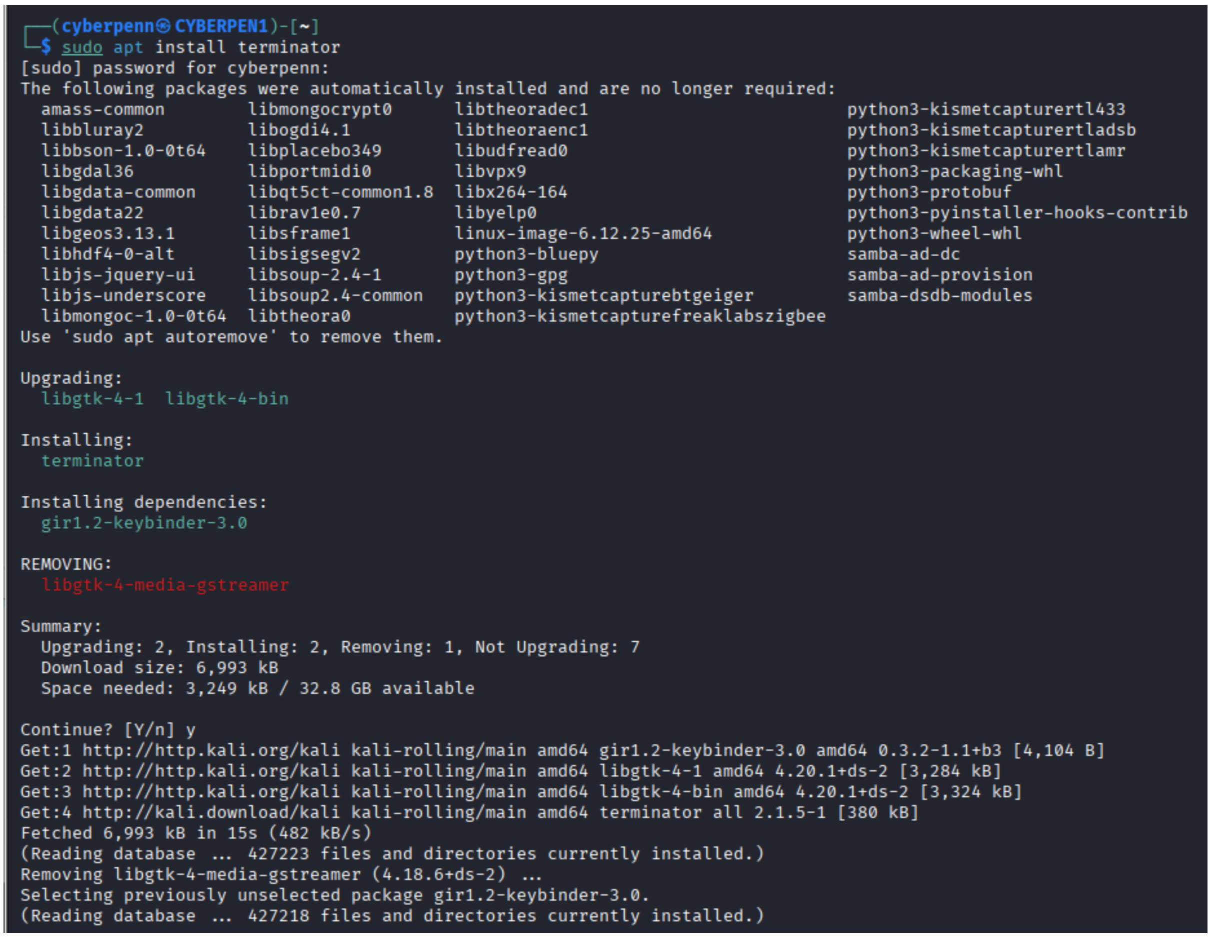
Explanation of the Output:

- The first section shows detailed system information, like:
  - Operating System: Kali Linux
  - Kernel Version: 6.12.38
  - Architecture: x86_64
  - Hostname: CYBERPEN1
- The memory section displays RAM and swap memory usage, indicating:
  - Total memory available (3.8 GiB)
  - Used and free memory
  - No swap memory currently in use
- The storage section lists mounted file systems and their disk usage:
  - `/dev/sda1` is the main partition with 56 GB total and 43% used.
  - Temporary and virtual file systems (`/run`, `/tmp`, `/dev/shm`) are also listed with their respective usage.

Part 9: Installing Software

1. Installing Terminator:

- Use the following command to install Terminator, a terminal emulator: sudo apt install terminator



```
┌──(cyberpenn㉿CYBERPEN1)-[~]
└─$ sudo apt install terminator
[sudo] password for cyberpenn:
The following packages were automatically installed and are no longer required:
  amass-common        libmongocrypt0      libtheoradec1                      python3-kismetcapturertl433
  libbluray2          libogdi4.1          libtheoraenc1                      python3-kismetcapturertladsb
  libbson-1.0-0t64    libplacebo349       libudfread0                        python3-kismetcapturertlamr
  libgdal36           libportmidi0        libvpx9                            python3-packaging-whl
  libgdata-common     libqt5ct-common1.8  libx264-164                        python3-protobuf
  libgdata22          librav1e0.7         libyelp0                           python3-pyinstaller-hooks-contrib
  libgeos3.13.1       libsframe1          linux-image-6.12.25-amd64          python3-wheel-whl
  libhdf4-0-alt       libsigsegv2         python3-bluepy                     samba-ad-dc
  libjs-jquery-ui     libsoup-2.4-1       python3-gpg                        samba-ad-provision
  libjs-underscore    libsoup2.4-common   python3-kismetcapturebtgeiger      samba-dsdb-modules
  libmongoc-1.0-0t64  libtheora0          python3-kismetcapturefreaklabszigbee
Use 'sudo apt autoremove' to remove them.

Upgrading:
  libgtk-4-1  libgtk-4-bin

Installing:
  terminator

Installing dependencies:
  gir1.2-keybinder-3.0

REMOVING:
  libgtk-4-media-gstreamer

Summary:
  Upgrading: 2, Installing: 2, Removing: 1, Not Upgrading: 7
  Download size: 6,993 kB
  Space needed: 3,249 kB / 32.8 GB available

Continue? [Y/n] y
Get:1 http://http.kali.org/kali kali-rolling/main amd64 gir1.2-keybinder-3.0 amd64 0.3.2-1.1+b3 [4,104 B]
Get:2 http://http.kali.org/kali kali-rolling/main amd64 libgtk-4-1 amd64 4.20.1+ds-2 [3,284 kB]
Get:3 http://http.kali.org/kali kali-rolling/main amd64 libgtk-4-bin amd64 4.20.1+ds-2 [3,324 kB]
Get:4 http://kali.download/kali kali-rolling/main amd64 terminator all 2.1.5-1 [380 kB]
Fetched 6,993 kB in 15s (482 kB/s)
(Reading database ... 427223 files and directories currently installed.)
Removing libgtk-4-media-gstreamer (4.18.6+ds-2) ...
Selecting previously unselected package gir1.2-keybinder-3.0.
(Reading database ... 427218 files and directories currently installed.)
```

Task:

- Document the installation steps and any outputs observed during the installation process.

Task: Document the installation steps and any outputs observed during the installation process

- **Step 1:** Update the package list
  - **Command:**
  - sudo apt update

- **Step 2:** Install Terminator package
  - **Command:**

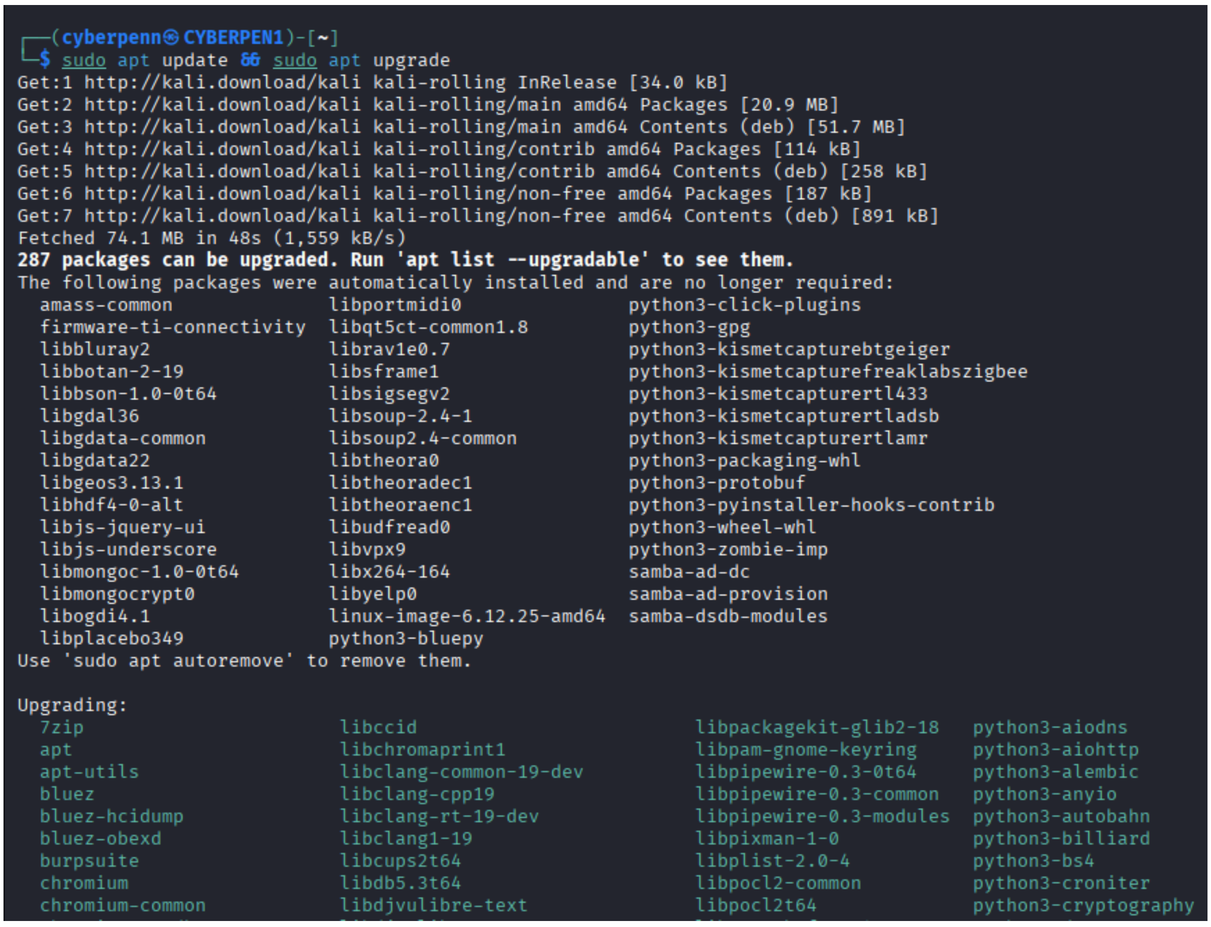
    - sudo apt install terminator
    - **Output Observed:**
        - The terminal displays the dependencies to be installed.
        - Prompts for confirmation:
        - Do you want to continue? [Y/n]
        - After confirmation (Y), installation proceeds
    - **Significance:** Installs Terminator and configures it within the system's application menu.

- **Step 3:** Verify installation
    - **Command:**
    - terminator --version
    - **Output Observed:**
    - terminator 2.1.5
    - **Significance:** Confirms that Terminator is installed successfully and ready to use.

Part 10: Updating/Installing Software

1. Updating Software:

- Use the following command to update the package list and upgrade installed packages: sudo apt update && sudo apt upgrade



```
┌──(cyberpenn㉿CYBERPEN1)-[~]
└─$ sudo apt update && sudo apt upgrade
Get:1 http://kali.download/kali kali-rolling InRelease [34.0 kB]
Get:2 http://kali.download/kali kali-rolling/main amd64 Packages [20.9 MB]
Get:3 http://kali.download/kali kali-rolling/main amd64 Contents (deb) [51.7 MB]
Get:4 http://kali.download/kali kali-rolling/contrib amd64 Packages [114 kB]
Get:5 http://kali.download/kali kali-rolling/contrib amd64 Contents (deb) [258 kB]
Get:6 http://kali.download/kali kali-rolling/non-free amd64 Packages [187 kB]
Get:7 http://kali.download/kali kali-rolling/non-free amd64 Contents (deb) [891 kB]
Fetched 74.1 MB in 48s (1,559 kB/s)
287 packages can be upgraded. Run 'apt list --upgradable' to see them.
The following packages were automatically installed and are no longer required:
  amass-common              libportmidi0              python3-click-plugins
  firmware-ti-connectivity  libqt5ct-common1.8        python3-gpg
  libbluray2                librav1e0.7               python3-kismetcapturebtgeiger
  libbotan-2-19             libsframe1                python3-kismetcapturefreaklabszigbee
  libbson-1.0-0t64          libsigsegv2               python3-kismetcapturertl433
  libgdal36                 libsoup-2.4-1             python3-kismetcapturertladsb
  libgdata-common           libsoup2.4-common         python3-kismetcapturertlamr
  libgdata22                libtheora0                python3-packaging-whl
  libgeos3.13.1             libtheoradec1             python3-protobuf
  libhdf4-0-alt             libtheoraenc1             python3-pyinstaller-hooks-contrib
  libjs-jquery-ui           libudfread0               python3-wheel-whl
  libjs-underscore          libvpx9                   python3-zombie-imp
  libmongoc-1.0-0t64        libx264-164               samba-ad-dc
  libmongocrypt0            libyelp0                  samba-ad-provision
  libogdi4.1                linux-image-6.12.25-amd64  samba-dsdb-modules
  libplacebo349             python3-bluepy
Use 'sudo apt autoremove' to remove them.

Upgrading:
  7zip                       libccid                        libpackagekit-glib2-18   python3-aiodns
  apt                        libchromaprint1                libpam-gnome-keyring     python3-aiohttp
  apt-utils                  libclang-common-19-dev         libpipewire-0.3-0t64     python3-alembic
  bluez                      libclang-cpp19                 libpipewire-0.3-common   python3-anyio
  bluez-hcidump              libclang-rt-19-dev             libpipewire-0.3-modules  python3-autobahn
  bluez-obexd                libclang1-19                   libpixman-1-0            python3-billiard
  burpsuite                  libcups2t64                    libplist-2.0-4           python3-bs4
  chromium                   libdb5.3t64                    libpocl2-common          python3-croniter
  chromium-common            libdjvulibre-text              libpocl2t64              python3-cryptography
```

Lab 5: Remote Evidence Acquisition and Digital Forensics Techniques

Objective:

This lab aims to provide students with hands-on experience in remote file acquisition, using network utilities and tools to access files on a Windows machine from a Kali Linux system. Students will learn about standard input/output/error devices and the use of tools like Zenmap and netcat for remote access and file transfers.

Part 1: Understanding Standard Input/Output/Error Devices

1. **Standard Devices:**

Explain the concepts of standard input, standard output, and standard error devices in Linux.

- Standard Input (stdin): The default input stream from which a program reads data.Typically connected to the keyboard when a command runs in a terminal.
- Standard Output (stdout): The default output stream where programs send their output data. Usually displayed on the terminal screen.
- Standard Error (stderr): The default stream where programs write error messages or diagnostics. Helps separate normal output from errors.

Discuss how network utilities utilize these standard devices to communicate and process data.

Combined Use in Network Communication

- **Stdin**: accepts data or commands to send.
- **Stdout**: shows received responses.
- **Stderr**: logs connection or configuration errors.
- Example: nc example.com 80
    - stdin: sends your HTTP request.
    - stdout: displays server response.
    - stderr: shows any connection errors.

Task: Write a short paragraph explaining how standard devices work in the context of command-line operations and networking.

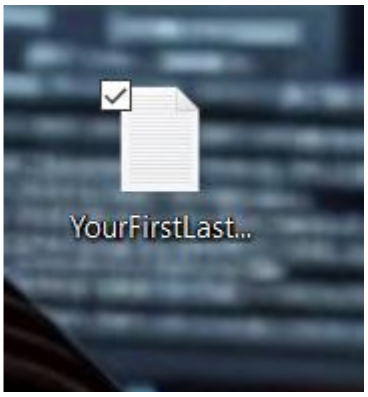
In Linux command-line operations, standard devices standard input (stdin), standard output (stdout), and standard error (stderr) play a central role in how data is processed and displayed. Stdin is used to take input from the user or another program, stdout handles the display of normal command results, and stderr outputs error messages separately. This separation ensures that users and system administrators can clearly distinguish between successful results and potential problems during command execution.

In networking, these standard devices are equally important for communication between local and remote systems. Tools like Netcat, Nmap, and Ping rely on stdin to receive commands or data, stdout to display responses from remote hosts, and stderr to log connection issues or permission errors. By managing data flow in this way, Linux allows network utilities to operate efficiently, automate tasks, and provide detailed feedback — making them valuable in digital forensics and remote evidence acquisition.

Part 2: Setting Up Windows for Remote Access

1. Creating an Evidence File in Windows:

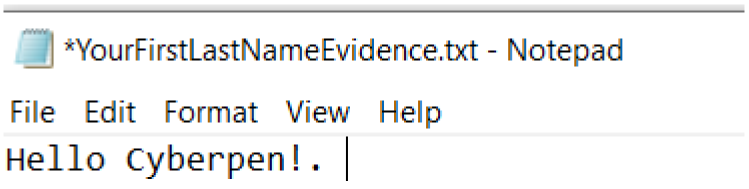
- On your Windows 10 machine, create a new text file named YourFirstLastNameEvidence.txt.



- In the file, add the content: Hello [Your Name]!.

Task:

- Verify the file has been created successfully with the correct content.


*YourFirstLastNameEvidence.txt - Notepad
File Edit Format View Help

```
Hello Cyberpen!.
```

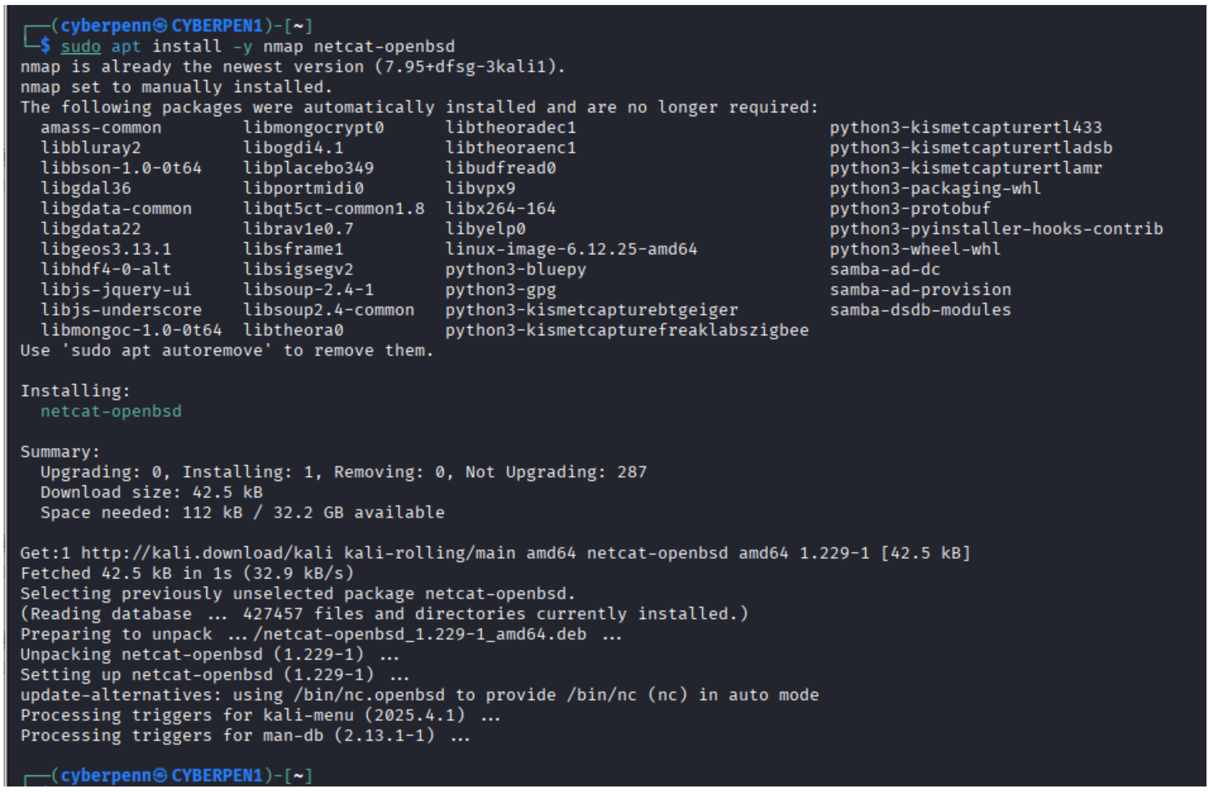
2. Restrictions on Direct Access:

- Note that you are not allowed to touch the Windows machine directly after creating the file. Instead, you will access it remotely from Kali Linux.

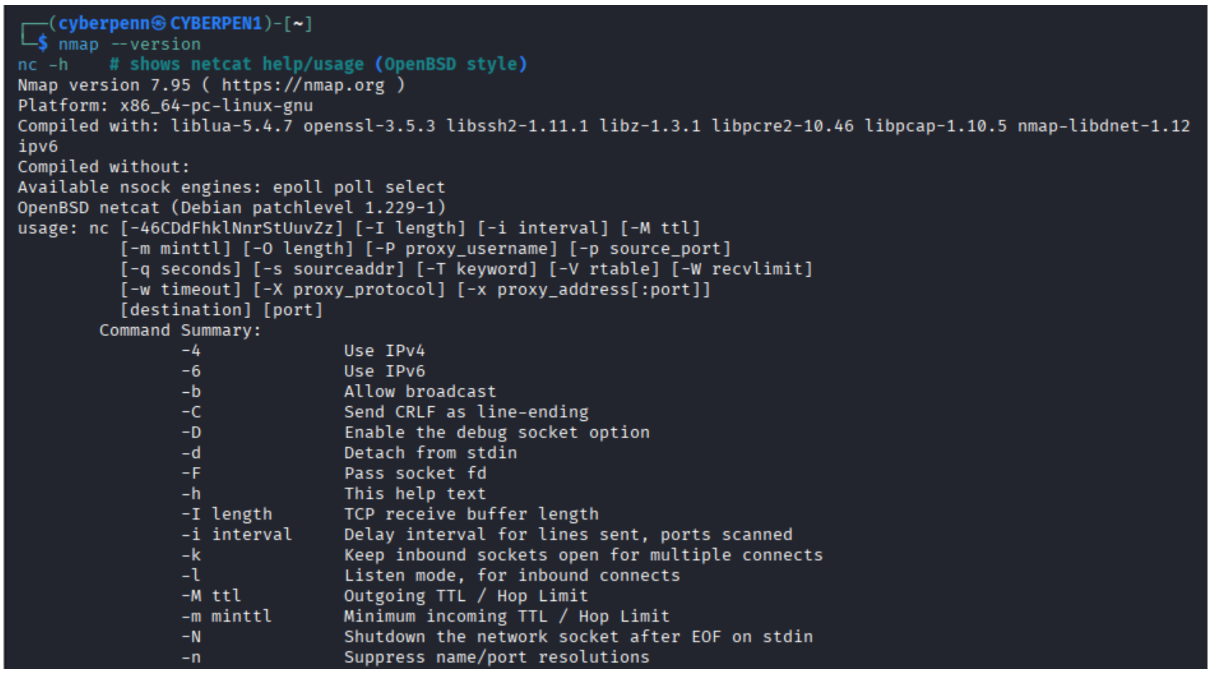
Part 3: Setting Up Kali Linux for Remote Access

1. Installing Necessary Tools:

- Ensure that nmap and netcat are installed on your Kali Linux system. If not, install them using:
- sudo apt update sudo apt install nmap netcat



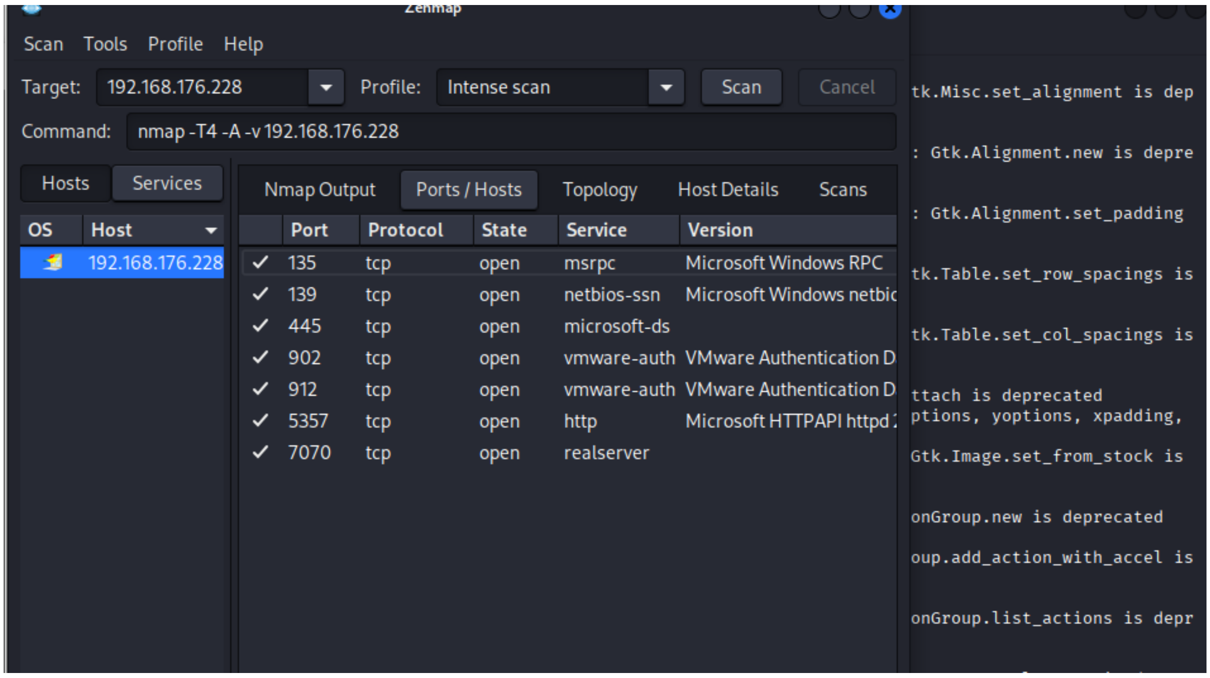
```
┌──(cyberpenn㉿CYBERPEN1)-[~]
└─$ sudo apt install -y nmap netcat-openbsd
nmap is already the newest version (7.95+dfsg-3kali1).
nmap set to manually installed.
The following packages were automatically installed and are no longer required:
  amass-common        libmongocrypt0      libtheoradec1                     python3-kismetcapturertl433
  libbluray2          libogdi4.1          libtheoraenc1                     python3-kismetcapturertladsb
  libbson-1.0-0t64    libplacebo349       libudfread0                       python3-kismetcapturertlamr
  libgdal36           libportmidi0        libvpx9                           python3-packaging-whl
  libgdata-common     libqt5ct-common1.8  libx264-164                       python3-protobuf
  libgdata22          librav1e0.7         libyelp0                          python3-pyinstaller-hooks-contrib
  libgeos3.13.1       libsframe1          linux-image-6.12.25-amd64         python3-wheel-whl
  libhdf4-0-alt       libsigsegv2         python3-bluepy                    samba-ad-dc
  libjs-jquery-ui     libsoup-2.4-1       python3-gpg                       samba-ad-provision
  libjs-underscore    libsoup2.4-common   python3-kismetcapturebtgeiger     samba-dsdb-modules
  libmongoc-1.0-0t64  libtheora0          python3-kismetcapturefreaklabszigbee
Use 'sudo apt autoremove' to remove them.

Installing:
  netcat-openbsd

Summary:
  Upgrading: 0, Installing: 1, Removing: 0, Not Upgrading: 287
  Download size: 42.5 kB
  Space needed: 112 kB / 32.2 GB available

Get:1 http://kali.download/kali kali-rolling/main amd64 netcat-openbsd amd64 1.229-1 [42.5 kB]
Fetched 42.5 kB in 1s (32.9 kB/s)
Selecting previously unselected package netcat-openbsd.
(Reading database ... 427457 files and directories currently installed.)
Preparing to unpack .../netcat-openbsd_1.229-1_amd64.deb ...
Unpacking netcat-openbsd (1.229-1) ...
Setting up netcat-openbsd (1.229-1) ...
update-alternatives: using /bin/nc.openbsd to provide /bin/nc (nc) in auto mode
Processing triggers for kali-menu (2025.4.1) ...
Processing triggers for man-db (2.13.1-1) ...

┌──(cyberpenn㉿CYBERPEN1)-[~]
```



```
┌──(cyberpenn㉿CYBERPEN1)-[~]
└─$ nmap --version
nc -h    # shows netcat help/usage (OpenBSD style)
Nmap version 7.95 ( https://nmap.org )
Platform: x86_64-pc-linux-gnu
Compiled with: liblua-5.4.7 openssl-3.5.3 libssh2-1.11.1 libz-1.3.1 libpcre2-10.46 libpcap-1.10.5 nmap-libdnet-1.12
ipv6
Compiled without:
Available nsock engines: epoll poll select
OpenBSD netcat (Debian patchlevel 1.229-1)
usage: nc [-46CDdFhklNnrStUuvZz] [-I length] [-i interval] [-M ttl]
          [-m minttl] [-O length] [-P proxy_username] [-p source_port]
          [-q seconds] [-s sourceaddr] [-T keyword] [-V rtable] [-W recvlimit]
          [-w timeout] [-X proxy_protocol] [-x proxy_address[:port]]
          [destination] [port]
        Command Summary:
                -4              Use IPv4
                -6              Use IPv6
                -b              Allow broadcast
                -C              Send CRLF as line-ending
                -D              Enable the debug socket option
                -d              Detach from stdin
                -F              Pass socket fd
                -h              This help text
                -I length       TCP receive buffer length
                -i interval     Delay interval for lines sent, ports scanned
                -k              Keep inbound sockets open for multiple connects
                -l              Listen mode, for inbound connects
                -M ttl          Outgoing TTL / Hop Limit
                -m minttl       Minimum incoming TTL / Hop Limit
                -N              Shutdown the network socket after EOF on stdin
                -n              Suppress name/port resolutions
```

2. Using Zenmap:

- Open Zenmap on Kali Linux to perform a scan of the Windows machine to identify open ports and services. Input the Windows machine's IP address and scan for common ports.



Target: 192.168.176.228 Profile: Intense scan

Command: nmap -T4 -A -v 192.168.176.228

OS	Host
	192.168.176.228

	Port	Protocol	State	Service	Version
✓	135	tcp	open	msrpc	Microsoft Windows RPC
✓	139	tcp	open	netbios-ssn	Microsoft Windows netbic
✓	445	tcp	open	microsoft-ds	
✓	902	tcp	open	vmware-auth	VMware Authentication D
✓	912	tcp	open	vmware-auth	VMware Authentication D
✓	5357	tcp	open	http	Microsoft HTTPAPI httpd 2
✓	7070	tcp	open	realserver	

```
tk.Misc.set_alignment is dep

: Gtk.Alignment.new is depre

: Gtk.Alignment.set_padding

tk.Table.set_row_spacings is

tk.Table.set_col_spacings is

ttach is deprecated
ptions, yoptions, xpadding,

Gtk.Image.set_from_stock is

onGroup.new is deprecated

oup.add_action_with_accel is

onGroup.list_actions is depr
```

Task:

- Document the scan results, focusing on any open ports related to file sharing or remote access (e.g., port 139 for SMB, port 445 for SMB over TCP).

Open Ports and Services Found

- **Port 135 (TCP):** Microsoft Windows RPC – used for remote procedure calls.
- **Port 139 (TCP):** NetBIOS-SSN – supports SMB file and printer sharing.
- **Port 445 (TCP):** Microsoft-DS – SMB over TCP, used for Windows file sharing.
- **Port 902 (TCP):** VMware Authentication Daemon (SSL) – remote management service.
- **Port 912 (TCP):** VMware Authentication Daemon (non-SSL).
- **Port 5357 (TCP):** HTTPAPI service – Windows UPnP / device sharing.
- **Port 7070 (TCP):** AnyDesk Client – remote desktop service.

Observations

- SMB-related ports (139, 445) are open, confirming active file sharing.
- SMB signing is enabled but not required, which may reduce security.

- AnyDesk detected on port 7070, indicating remote desktop software is active.
- VMware services (ports 902, 912) show virtualization or remote console activity.
- Host details:
  - Hostname: DESKTOP-UPP58V2
  - Workgroup: WORKGROUP
  - MAC Address: 7C:2A:31:C6:50:F7 (Intel)
  - OS Detected: Windows 10 / 11 / Server 2019

Security Notes

- SMB ports may allow remote access or file retrieval if misconfigured.
- AnyDesk and VMware ports can serve as remote access entry points.
- RPC (port 135) could be used for administrative control or remote execution.
- These open ports are critical for remote evidence acquisition but also represent potential vulnerabilities.

Conclusion

- The Windows system (192.168.176.228) is reachable and running multiple network services.
- The most relevant for remote evidence collection are:
  - Port 139 (NetBIOS)
  - Port 445 (SMB)
- The presence of AnyDesk (7070) and VMware services (902/912) confirms remote access capabilities.
- The system should be secured and monitored before data extraction to avoid unauthorized access.

Part 4: Establishing Remote Access

1. Using Netcat for Remote Access:

- On the Windows machine, you need to set up netcat to listen for incoming connections. Open a command prompt and run:
```
nc -l -p 12345 -w 10 < YourFirstLastNameEvidence.txt
```

2. Note: Ensure that Windows Firewall allows this connection or temporarily disable it for this exercise.

3. Connecting from Kali:

- On the Kali Linux machine, use netcat to connect to the Windows machine and acquire the file:
  nc [Windows_IP_Address] 12345 > YourFirstLastNameEvidence.txt

4. Task:

- After running the command, verify that YourFirstLastNameEvidence.txt has been successfully downloaded to the Kali system by checking its content:
  cat YourFirstLastNameEvidence.txt

Part 6: Digital Forensics and Remote Evidence Acquisition

1. Discussing Remote Evidence Acquisition:

- Define what remote evidence acquisition means in the context of digital forensics. Highlight the importance of maintaining evidence integrity and legal implications.

Remote Evidence Acquisition

Remote evidence acquisition in digital forensics refers to the process of collecting digital data from a remote system or networked device without physically accessing it. This method is commonly used when systems are geographically dispersed, active on live networks, or when immediate access is required to prevent data loss or tampering. It allows forensic investigators to obtain critical evidence such as log files, user activity, or volatile memory over secure network connections.

However, remote acquisition presents several challenges. Network latency, bandwidth limitations, and encryption can affect the completeness and integrity of data. There is also the risk of altering volatile evidence during transmission or violating privacy and jurisdictional laws if the system is in another region.

To maintain evidence integrity, best practices include using secure, forensically sound tools that generate cryptographic hashes, documenting every step of the process, ensuring chain of custody, and using encrypted channels for data transfer. Legal authorization, such as warrants or consent, must always be obtained to ensure admissibility of the evidence in court.

Task:

Write a brief explanation (150-200 words) discussing the challenges and best practices of remote evidence acquisition.

Challenges and Best Practices of Remote Evidence Acquisition

Remote evidence acquisition involves collecting digital evidence from a system or network without physical access to the device. It is often used in cybercrime investigations, incident response, and digital forensics to gather data such as system logs, memory dumps, and user activity across remote environments.

One of the main challenges in remote acquisition is maintaining the integrity and authenticity of evidence during transfer. Network latency, encryption, and unstable connections can lead to incomplete or corrupted data. Additionally, accessing systems across jurisdictions raises legal and privacy concerns, requiring proper authorization or court orders. Another challenge is the risk of altering volatile data, such as active processes or RAM contents, which may affect admissibility in court.

To overcome these issues, investigators should use forensically sound tools that create cryptographic hash values to verify data integrity. Evidence should be transmitted over secure and encrypted channels (e.g., SSH, TLS) and well-documented to maintain a chain of custody. Adhering to international standards like NIST SP 800-86 ensures reliability, legality, and the evidential value of the acquired data.