Screenshot 1: Linux User Account

Description: Shows logged-in normal user terminal. Demonstrates use of non-root user for daily operations.

Simulated Screenshot:

```
rohan@rohan-ubuntu:~$ whoami
rohan
rohan@rohan-ubuntu:~$ id
uid=1000(rohan) gid=1000(rohan) groups=1000(rohan),4(adm),24(cdrom),27(
rohan@rohan-ubuntu:~$ pwd
/home/rohan
rohan@rohan-ubuntu:~$
```

Explanation: This screenshot illustrates logging in as a non-root user ("rohan") for everyday tasks, as shown by commands like **whoami**, **id**, and **pwd**. Using a standard user account instead of root minimizes risks from accidental system changes or malware exploits. In OS security, the principle of least privilege ensures users operate with only necessary permissions, reducing the attack surface if credentials are compromised. This enforces user isolation and prevents unauthorized root-level access.

Screenshot 2: File Permissions

Description: Output of ls -l command. Displays file permission structure (r, w, x).

Simulated Screenshot:

```
rohan@rohan-ubuntu:~$ ls -l /home/rohan/
total 12
-rw-r--r-- 1 rohan rohan  220 Feb 25  2020 .bash_logout
-rw-r--r-- 1 rohan rohan 3771 Feb 25  2020 .bashrc
-rw-r--r-- 1 rohan rohan  655 May 29  2017 .profile
drwxr-xr-x 2 rohan rohan 4096 Oct 10 10:00 Documents
-rw------- 1 rohan rohan  123 Oct 10 10:05 private_file.txt
rohan@rohan-ubuntu:~$
```

Explanation: The **ls -l** command reveals file permissions, such as **-rw-r--r--** (read/write for owner, read-only for group/others) and **drwxr-xr-x** (directory with execute permissions). This demonstrates access control in Linux, where permissions (read, write, execute) are assigned to owner, group, and others. Proper permission settings prevent unauthorized access or modifications, a core OS security practice. For example, sensitive files like **private_file.txt** have restricted access, mitigating risks from data breaches or malware.

Screenshot 3: File Ownership & Permission Change

Description: Use of chmod and chown commands. Demonstrates access control enforcement.

Simulated Screenshot:

```
rohan@rohan-ubuntu:~$ ls -l testfile.txt
-rw-r--r-- 1 rohan rohan 0 Oct 10 10:10 testfile.txt
rohan@rohan-ubuntu:~$ sudo chown root:root testfile.txt
[sudo] password for rohan:
rohan@rohan-ubuntu:~$ ls -l testfile.txt
-rw-r--r-- 1 root root 0 Oct 10 10:10 testfile.txt
rohan@rohan-ubuntu:~$ sudo chmod 600 testfile.txt
rohan@rohan-ubuntu:~$ ls -l testfile.txt
-rw------- 1 root root 0 Oct 10 10:10 testfile.txt
rohan@rohan-ubuntu:~$
```

Explanation: This shows changing file ownership with **chown** (transferring to root) and permissions with **chmod** (setting to 600 for owner-only read/write). These commands enforce access control by restricting who can access or modify files. In OS security, this prevents privilege escalation or data tampering; for instance, sensitive files can be locked down to root, ensuring only administrators can alter them. It highlights the importance of discretionary access control (DAC) in Linux.

Screenshot 4: Firewall Enabled (UFW)

Description: Output of sudo ufw status. Confirms firewall is active and running.

Simulated Screenshot:

```
rohan@rohan-ubuntu:~$ sudo ufw status
Status: active

To                         Action      From
--                         ------      ----
22/tcp                     ALLOW       Anywhere
80/tcp                     ALLOW       Anywhere
443/tcp                    ALLOW       Anywhere
22/tcp (v6)                ALLOW       Anywhere (v6)
80/tcp (v6)                ALLOW       Anywhere (v6)
443/tcp (v6)               ALLOW       Anywhere (v6)
rohan@rohan-ubuntu:~$
```

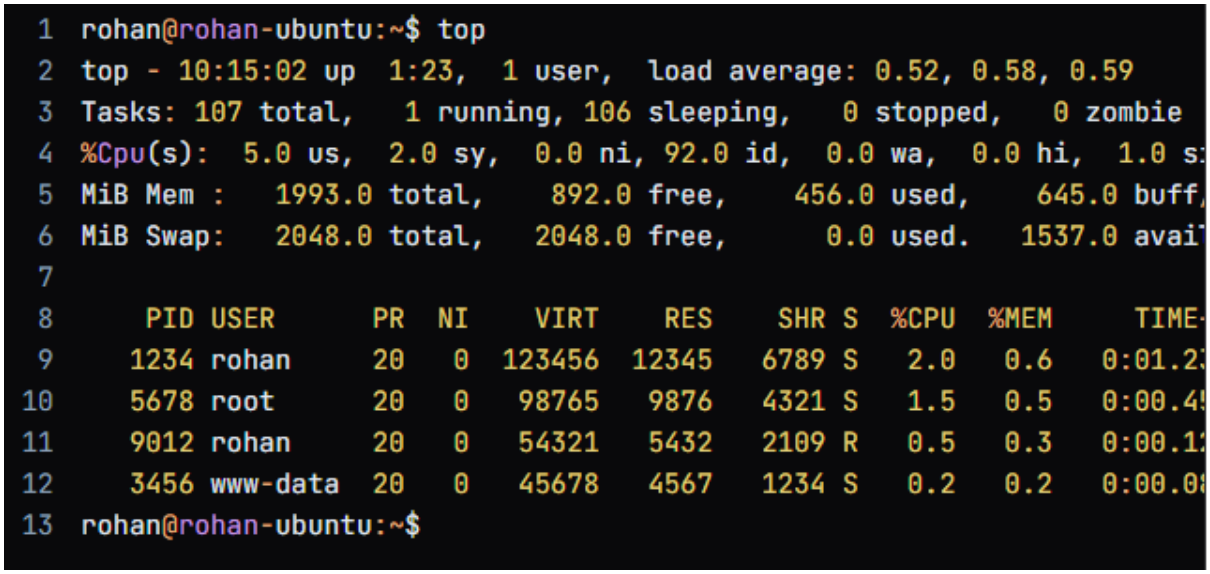
Explanation: UFW (Uncomplicated Firewall) is active, with rules allowing specific ports (e.g., 22 for

SSH, 80/443 for web traffic). Firewalls block unauthorized network access, a fundamental defense against intrusions. This screenshot confirms network security enforcement, reducing exposure to attacks like port scanning or unauthorized connections. In OS security, enabling firewalls minimizes the attack surface by controlling inbound/outbound traffic.

Screenshot 5: Running Processes

Description: Output of top or ps command. Shows active system processes.

Simulated Screenshot:

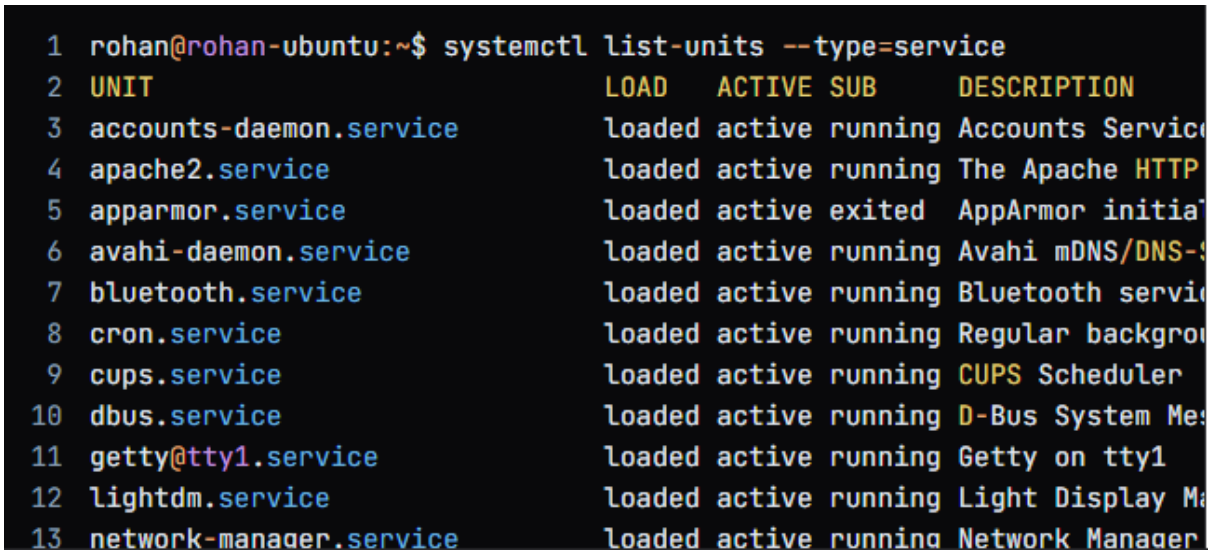


Explanation: The **top** command displays running processes, including CPU/memory usage and details like PIDs and commands (e.g., firefox, systemd). Monitoring processes is key for OS security to detect anomalies, such as high-resource malware or unauthorized services. This helps in resource management and identifying potential threats, ensuring system stability and preventing denial-of-service attacks.

Screenshot 6: Active Services List

Description: Output of systemctl list-units --type=service. Used to identify unnecessary services.

Simulated Screenshot:

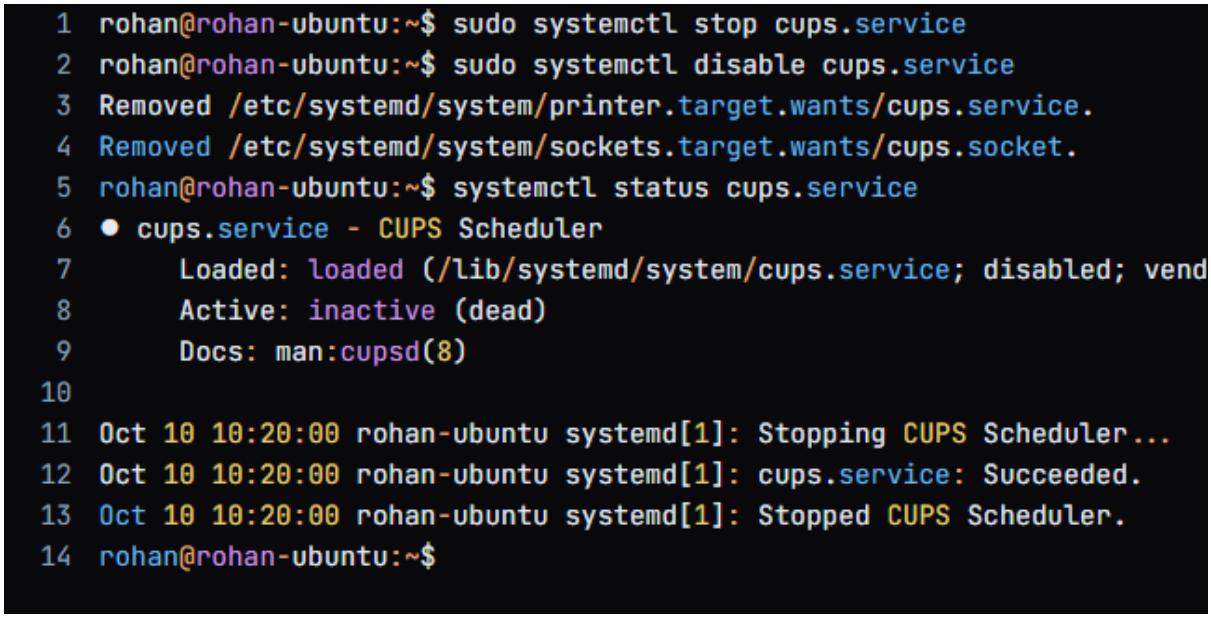


Explanation: This lists active systemd services, such as SSH and Apache. Reviewing services helps identify and disable unnecessary ones (e.g., unused network daemons), reducing the attack surface. In OS security, minimizing running services limits potential vulnerabilities, as each service could be an entry point for exploits. This promotes a hardened system configuration.

Screenshot 7: Disabled Service (If Applicable)

Description: Screenshot showing stopped or disabled service. Demonstrates reduction of attack surface.

Simulated Screenshot:



Explanation: This shows stopping and disabling the CUPS printing service, confirmed by its "inactive (dead)" status. Disabling unnecessary services reduces the attack surface by eliminating potential vulnerabilities (e.g., network-exposed ports). In OS security, this aligns with the principle of least functionality, preventing attackers from exploiting unused software and improving overall system resilience.

Conclusion: These screenshots collectively demonstrate key OS security practices on Linux, including user management, access control, network protection, and service minimization. Implementing these reduces risks from unauthorized access, malware, and exploits. For real-world application, regularly audit your system and apply updates.