

### Screenshot 1: Linux User Account

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

**Simulated Screenshot:**

```
1 rohan@rohan-ubuntu:~$ whoami
2 rohan
3 rohan@rohan-ubuntu:~$ id
4 uid=1000(rohan) gid=1000(rohan) groups=1000(rohan),4(adm),24(cdrom),270
5 rohan@rohan-ubuntu:~$ pwd
6 /home/rohan
7 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:**

```
1 rohan@rohan-ubuntu:~$ ls -l /home/rohan/
2 total 12
3 -rw-r--r-- 1 rohan rohan 220 Feb 25 2020 .bash_logout
4 -rw-r--r-- 1 rohan rohan 3771 Feb 25 2020 .bashrc
5 -rw-r--r-- 1 rohan rohan 655 May 29 2017 .profile
6 drwxr-xr-x 2 rohan rohan 4096 Oct 10 10:00 Documents
7 -rw----- 1 rohan rohan 123 Oct 10 10:05 private_file.txt
8 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:

```
1 rohan@rohan-ubuntu:~$ ls -l testfile.txt
2 -rw-r--r-- 1 rohan rohan 0 Oct 10 10:10 testfile.txt
3 rohan@rohan-ubuntu:~$ sudo chown root:root testfile.txt
4 [sudo] password for rohan:
5 rohan@rohan-ubuntu:~$ ls -l testfile.txt
6 -rw-r--r-- 1 root root 0 Oct 10 10:10 testfile.txt
7 rohan@rohan-ubuntu:~$ sudo chmod 600 testfile.txt
8 rohan@rohan-ubuntu:~$ ls -l testfile.txt
9 -rw----- 1 root root 0 Oct 10 10:10 testfile.txt
10 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:

```
1 rohan@rohan-ubuntu:~$ sudo ufw status
2 Status: active
3
4 To                         Action      From
5 --                         -----      -----
6 22/tcp                      ALLOW       Anywhere
7 80/tcp                      ALLOW       Anywhere
8 443/tcp                     ALLOW       Anywhere
9 22/tcp (v6)                 ALLOW       Anywhere (v6)
10 80/tcp (v6)                ALLOW       Anywhere (v6)
11 443/tcp (v6)               ALLOW       Anywhere (v6)
12 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:**

```
1 rohan@rohan-ubuntu:~$ top
2 top - 10:15:02 up 1:23, 1 user, load average: 0.52, 0.58, 0.59
3 Tasks: 107 total, 1 running, 106 sleeping, 0 stopped, 0 zombie
4 %Cpu(s): 5.0 us, 2.0 sy, 0.0 ni, 92.0 id, 0.0 wa, 0.0 hi, 1.0 s:
5 MiB Mem : 1993.0 total, 892.0 free, 456.0 used, 645.0 buff,
6 MiB Swap: 2048.0 total, 2048.0 free, 0.0 used. 1537.0 avai
7
8      PID USER      PR  NI      VIRT      RES      SHR S %CPU %MEM     TIME-
9      1234 rohan    20   0  123456  12345  6789 S  2.0  0.6  0:01.23
10     5678 root     20   0  98765  9876  4321 S  1.5  0.5  0:00.49
11     9012 rohan    20   0  54321  5432  2109 R  0.5  0.3  0:00.12
12    3456 www-data  20   0  45678  4567  1234 S  0.2  0.2  0:00.08
13 rohan@rohan-ubuntu:~$
```

**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:**

```

1 rohan@rohan-ubuntu:~$ systemctl list-units --type=service
2 UNIT                                     LOAD  ACTIVE SUB      DESCRIPTION
3 accounts-daemon.service                 loaded active running Accounts Service
4 apache2.service                         loaded active running The Apache HTTP
5 apparmor.service                        loaded active exited  AppArmor initial
6 avahi-daemon.service                  loaded active running Avahi mDNS/DNS-SD
7 bluetooth.service                      loaded active running Bluetooth service
8 cron.service                           loaded active running Regular background
9 cups.service                            loaded active running CUPS Scheduler
10 dbus.service                           loaded active running D-Bus System Message Bus
11 getty@tty1.service                    loaded active running Getty on tty1
12 lightdm.service                       loaded active running Light Display Manager
13 network-manager.service               loaded active running Network Manager

```

**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:

```

1 rohan@rohan-ubuntu:~$ sudo systemctl stop cups.service
2 rohan@rohan-ubuntu:~$ sudo systemctl disable cups.service
3 Removed /etc/systemd/system/printer.target.wants/cups.service.
4 Removed /etc/systemd/system/sockets.target.wants/cups.socket.
5 rohan@rohan-ubuntu:~$ systemctl status cups.service
6 ● cups.service - CUPS Scheduler
7     Loaded: loaded (/lib/systemd/system/cups.service; disabled; vendor
8     Active: inactive (dead)
9     Docs: man:cupsd(8)
10
11 Oct 10 10:20:00 rohan-ubuntu systemd[1]: Stopping CUPS Scheduler...
12 Oct 10 10:20:00 rohan-ubuntu systemd[1]: cups.service: Succeeded.
13 Oct 10 10:20:00 rohan-ubuntu systemd[1]: Stopped CUPS Scheduler.
14 rohan@rohan-ubuntu:~$

```

**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.