Operating System Security Fundamentals (Linux & Windows)

Operating System Security means protecting the computer's operating system so that:

- Unauthorized users cannot access it
- Malware cannot damage it
- Data remains safe, private, and unchanged

Both Windows and Linux use security mechanisms to control:

- Who can access the system
- What actions users and programs can perform

Why OS Security Is Important

If the operating system is not secure:

- Hackers can steal data
- Viruses can control your system
- Files can be deleted or modified
- Personal information can leak

Example:

If anyone can log in to your laptop without a password, they can access your files, photos, or banking information.

User Accounts and Authentication

The OS checks **who you are** before allowing access.

Windows

- Uses **username + password**
- Supports **PIN, fingerprint, face recognition**
- Admin and Standard user accounts

Linux

- Uses **username + password**
- Root user has full control
- Normal users have limited access

Example

- **Admin user**: Can install software
- **Normal user**: Can only use allowed applications

If malware runs as a **normal user**, damage is limited.

Authorization and Permissions

After login, the OS decides **what you are allowed to do**.

Linux Permissions

- Read (r)
- Write (w)
- Execute (x)

Permissions apply to:

- Owner
- Group

- Others

Example (Linux):

- File readable only by the owner
- Others cannot open or modify it

Windows Permissions

- Read
- Write
- Modify
- Full control

Example (Windows):

- Only HR department can access salary files

Administrator / Root Protection

Linux

- **Root user** has complete system control
- Uses sudo command for admin tasks
- Prevents accidental system damage

Windows

- Uses **Administrator account**
- User Account Control (UAC) asks permission before system changes

Example

When installing software:

- Windows asks: “Do you want to allow this app?”
- Linux asks for sudo password

This prevents malware from installing silently.

File System Security

Protecting files and folders from unauthorized access.

Linux

- Strong permission-based file system
- System files protected from normal users

Windows

- NTFS file system
- Access Control Lists (ACL)

Example

System files cannot be deleted by normal users, preventing OS damage.

Malware Protection

Windows

- Windows Defender (built-in antivirus)
- Real-time protection
- Automatic updates

Linux

- Fewer viruses due to permission model

- Antivirus used mainly for servers
- Firewalls and permissions reduce risk

Example

A virus downloaded in Linux cannot modify system files without root access.

Updates and Patch Management

Fixing security holes by updating the OS.

Windows

- Automatic updates
- Security patches released monthly

Linux

- Updates via package managers (apt, yum)
- Faster patch availability

Example

If a hacker finds a flaw in Windows, Microsoft releases a patch. If you don't update, attackers can exploit it.

Firewall Protection

Controls incoming and outgoing network traffic.

Windows

- Windows Firewall
- Blocks unauthorized network access

Linux

- iptables, ufw, firewalld

Example

Firewall blocks unknown devices from accessing your computer over the network.

Logging and Monitoring

Recording system activities to detect suspicious behavior.

Windows

- Event Viewer logs
- Login attempts
- System changes

Linux

- Log files (/var/log)
- Tracks login attempts and system actions

Example

If someone tries to log in multiple times with wrong passwords, logs help detect attacks.

Encryption

Protecting data so others cannot read it.

Windows

- BitLocker (disk encryption)

Linux

- LUKS (disk encryption)

Example
If your laptop is stolen, encrypted data cannot be accessed without the password.

Simple Real-Life Scenario
Scenario: Malware Download

1. User downloads a malicious file
2. OS asks for admin/root permission
3. User denies permission
4. Malware cannot modify system files
5. System remains safe

This shows **OS security working correctly**.

---

1. What is Linux?
Linux is an operating system, just like Windows.

- Windows → paid, closed source
- Linux → free, open source, secure

Linux is widely used in:

- Cyber security
- Servers
- Ethical hacking
- Cloud computing

Simple example:

- Windows = Your main house
- Linux = A secure lab where you experiment safely

---

2. What is a Virtual Machine (VM)?
A **Virtual Machine** is a **computer inside your computer**.

- It runs like a real PC
- Has its own OS (Linux)
- Cannot harm your main Windows system

Simple example:
Think of a VM like:

- A **game running in a window**
- When you close it, everything inside stops

---

3. What will a VM actually do?
A VM allows you to:

- Run Linux inside Windows
- Test tools safely
- Practice cybersecurity tasks
- Learn Linux commands
- Avoid damaging your real system

Example:
If you install hacking or malware-analysis tools:

- VM gets affected
- Windows stays safe

---

4. Why do we need to install Linux?
Main reasons:

1. **Security practice**
2. **Linux is more secure**
3. Most cybersecurity tools work on Linux
4. Real-world servers use Linux
5. You can safely test attacks and defenses

Example:
You should not test security tools on your real Windows OS.
Linux VM = Safe environment.

Linux Security Commands

1. sudo : Run Commands as Superuser
Use **sudo** to execute commands with root privileges. This is crucial for administrative tasks like installing software or editing system files.
sudo apt update
Tip: Always use sudo sparingly to avoid accidental system-wide changes.
2. ssh : ***Secure Remote Login***
SSH allows you to securely connect to remote Linux servers.
ssh user@192.168.1.10
Use SSH key authentication for enhanced security instead of passwords.
3. scp : ***Secure Copy Between Machines***
Use scp to securely transfer files between your machine and a remote server.
scp file.txt user@192.168.1.10:/home/user/
4. sftp : ***Secure FTP Transfers***
SFTP is like FTP, but over an encrypted SSH connection.
sftp user@192.168.1.10
5. iptables : ***Configure Advanced Firewall Rules***
iptables gives you granular control over network traffic.
sudo iptables -A INPUT -p tcp --dport 22 -j ACCEPT
6. ufw : ***User-Friendly Firewall***
ufw is a simplified interface for managing iptables.
sudo ufw enable
sudo ufw allow 22/tcp
7. fail2ban : ***Protect Against Brute Force Attacks***
Monitors logs and bans IPs showing malicious behavior.
sudo fail2ban-client status sshd

8. nmap : ***Network Port Scanner***
nmap scans networks for open ports and potential vulnerabilities.
```
nmap -sV 192.168.1.10
```
9. chmod : ***Change File Permissions***
Control who can read, write, or execute files.
```
chmod 700 secret.txt
```
10. chown : ***Change File Ownership***
Transfer ownership of files or directories.
```
sudo chown user:user /var/www/html
```
11. chkrootkit : ***Rootkit Detection Tool***
Detect rootkits that may be hiding in your system.
```
sudo chkrootkit
```
12. rkhunter : ***Rootkit Scanner***
rkhunter scans for rootkits, backdoors, and local exploits.
```
sudo rkhunter --check
```
13. lynis : ***Full Security Audit***
lynis performs comprehensive security checks on your system.
```
sudo lynis audit system
```
14. openssl : ***Encryption & Certificate Management***
Generate certificates and encrypt data with openssl.
```
openssl enc -aes-256-cbc -salt -in file.txt -out file.enc
```
15. gpg : ***Encrypt & Sign Files***
Encrypt sensitive files or emails using gpg.
```
gpg -c secret.txt
```

User Accounts, Permissions, and Access Control Mechanisms in Operating Systems

1. User Accounts (Who can use the system)
1.1 Types of Users in Linux
Linux separates users to protect the system.
(a) **Root User**

- Username: root
- Has full control
- Can modify system files
- Can break the system if misused

whoami
If output is root, you have full privileges.

(b) **Normal Users**

- Limited permissions

- Cannot change system files
- Used for daily work

Security reason:
If malware runs as a normal user, damage is limited.

(c) **System Users**

- Used by services (web server, database)
- Cannot log in normally

cat /etc/passwd
Shows all users.

---

2. User Authentication (How users prove identity)
Linux uses **username + password**.
Change password
passwd
Security rules:

- Passwords are encrypted
- Stored in /etc/shadow (root only)

sudo cat /etc/shadow

---

3. Groups (Managing multiple users easily)
Groups allow permission control for many users.
View your groups
groups
Add user to a group
sudo usermod -aG sudo username
Security use:
Only trusted users get admin rights.

---

4. File Permissions (Who can access what)
Linux uses Read (r), Write (w), Execute (x).
Check permissions
ls -l
Example output:
-rwxr--r—

---

5. Ownership Control
Every file has:

- Owner
- Group

Change owner

sudo chown user file.txt

Change group

sudo chgrp group file.txt

---

6. Special Permissions (Advanced)

6.1 SUID (Set User ID)

- File runs as owner (often root)

chmod u+s filename

Security risk:

Improper use can allow privilege escalation.

6.2 SGID (Set Group ID)

- File runs with group permissions

chmod g+s filename

6.3 Sticky Bit

- Only file owner can delete files in shared directories

chmod +t /shared_folder

Example:

/tmp uses sticky bit.

---

7. Access Control Lists (ACL)

ACLs give **fine-grained permissions**.

View ACL

getfacl file.txt

Set ACL

setfacl -m u:username:r file.txt

Security benefit:

More precise access control than chmod.

---

8. Sudo Access Control

sudo allows limited admin access.

Edit sudo rules

sudo visudo

Example:

username ALL=(ALL) ALL

Security use:

Avoids direct root login.

---

9. Login Access Control

Limit root login

```
sudo nano /etc/ssh/sshd_config
```

Set:

```
PermitRootLogin no
```

Prevents remote root access.

---

10. Account Locking and Monitoring

Lock a user

```
sudo passwd -l username
```

Unlock a user

```
sudo passwd -u username
```

Check failed logins

```
sudo cat /var/log/auth.log
```

---

11. Real-World Security Scenario

Scenario: Protecting a Confidential File

1. Create file

```
touch secret.txt
```

2. Restrict access

```
chmod 600 secret.txt
```

3. Assign owner

```
sudo chown manager secret.txt
```

Result:

Only the manager can read or modify the file.

Access Permissions: Unix Chmod, Chown and ls -1

#1) chmod: Change file access permissions.

Description: This command is used to change the file permissions. These permissions read, write and execute permission for the owner, group, and others.

Syntax (symbolic mode): ***chmod [ugoa][[+-=][mode]] file***

The first optional parameter indicates who – this can be (u)ser, (g)roup, (o)there, or (a)ll.

The second optional parameter indicates opcode – this can be for adding (+), removing (-), or assigning (=) permission.

The third optional parameter indicates the mode – this can be (r)ead, (w)rite, or e(x)ecute.

Example: Add writes permission for user, group, and others for file1.

$ chmod ugo+w file1

Syntax (numeric mode): *chmod [mode] file*

The mode is a combination of three digits – the first digit indicates the permission of the user, the second digit for the group, and the third digit for others.

Each digit is computed by adding the associated permissions. Read permission is '4', write permission is '2' and execute permission is '1'.

Example: Give read/write/execute permission to the user, read/execute permission to the group, and execute permission to others.

$ chmod 751 file1

#2) chown: Change ownership of the file.

- Description: Only the owner of the file has the right to change the file ownership.
- Syntax: chown [owner] [file]
- Example: Change the owner of file1 to user2 assuming it is currently owned by the current user
    - ***$ chown user2 file1***

#3) ls -l: command in Linux provides a detailed listing of files and directories, including their permissions, ownership, size, and modification date. The permissions section is particularly important for managing access control in a multi-user environment.

- The output of ls -l begins with a string of 10 characters that represent the file type and permissions. For example:
- **-rwxr-xr--**