Operating System Security

CyberSecPro

PRESENTATION:
STYLIANOS KARAGIANNIS (PDMFC, PORTUGAL) PHD @ IONIAN UNIVERSITY

Introduction to OS Security

Definition: Operating System Security refers to the measures and techniques employed to protect the OS and its resources from unauthorized access, misuse, or compromise.

Key Concepts:

- Threats: Malware, rootkits, privilege escalation, data theft.
- Goals: Confidentiality, Integrity, and Availability (CIA Triad).
- Security Models: Mandatory Access Control (MAC), Discretionary Access Control (DAC), Role-Based Access Control (RBAC).
- Importance of updates and patches.

What is an Operating System?

- Acts as an intermediary between user and hardware.
- Interface for executing programs conveniently and efficiently.
- Manages computer hardware and software.

Key Responsibilities:

- Allocation of resources (memory, processors, I/O devices).
- Ensures correct and secure system operation.

Purpose of an Operating System

- Provides an environment for efficient program execution.
- Core program running on the computer (kernel).
- Coordinates hardware usage for system and application programs.

History of Operating Systems

Era	Key Developments	Examples
1950s	First OS: GM-NAA I/O (1956)	GM-NAA I/O
1960s	Time-sharing systems	OS/360, DOS/360, TSS/360
1970s	Simplicity, multitasking, personal computers	Unix, CP/M
1980s	GUI-based OS, networking	Apple Macintosh, Windows
1990s	Open-source Linux, advanced GUIs	Linux, Windows 95
2000s-Present	Mobile OS, cloud, virtualization	iOS, Android

Characteristics of Operating Systems

- **Device Management**: Tracks and allocates devices.
- **File Management**: Allocates/deallocates resources for files.
- **Memory Management**: Allocates primary memory efficiently.
- **Processor Management**: Assigns processors to tasks.
- **Security**: Protects data from unauthorized access.
- **Error Detection**: Debugging tools and error messages.
- **Efficiency**: Maximizes resource utilization.

Common Operating Systems

OS	Developer	Key Features	Use Cases
Windows	Microsoft	User-friendly, gaming support	Personal, business, gaming
macOS	Apple	Sleek design, strong security	Creative industries, personal use
Linux	Community-driven	Open-source, customizable	Servers, development, tech enthusiasts
Unix	AT&T Bell Labs	Multiuser, multitasking, secure	Research, academic, servers

Functionalities of an Operating System

- **Resource Management**: Manages hardware for users.
- **Process Management**: Schedules and terminates processes.
- **Storage Management**: Manages file systems and storage.
- **Memory Management**: Tracks and allocates memory.
- **Security/Privacy**: Prevents unauthorized access.

Components of an Operating System

- **Kernel**: Core component managing essential services.
- **Shell**: Outer layer managing user interactions.
- **I/O System Management**: Tracks and manages devices.

Layered Design of an OS

- **Layered Structure**: Simplifies OS coding and testing.
  - Extended machine layer.
  - Operating system layer.
- **Advantages**: Abstraction improves reliability and modularity.

Key Takeaways:

- OS is the backbone of a computer system.
- Provides an interface between user and hardware.
- Manages resources, processes, and security.
- Evolved from simple systems to advanced mobile and cloud platforms.

File Permissions and Ownership

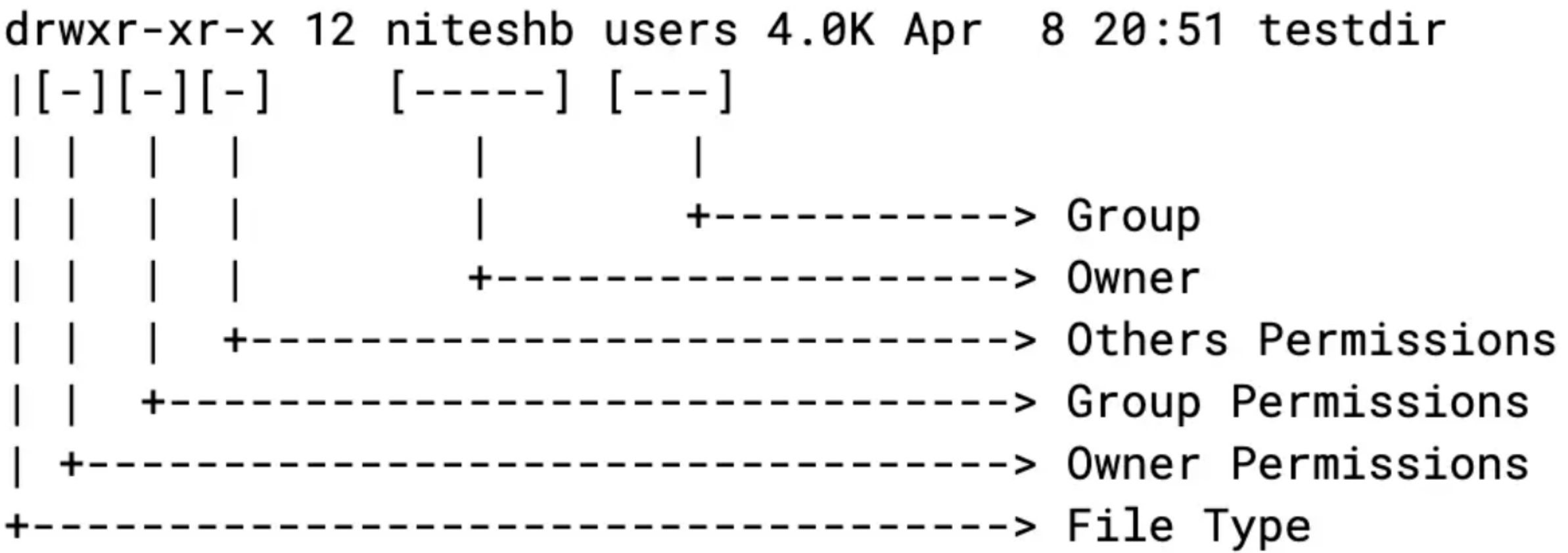
Linux File System Basics:

- **Ownership:** Every file/directory is associated with a user (owner) and group.
- **Permissions:** Controls who can read, write, or execute files.
  - **Types:** Read (r), Write (w), Execute (x).
  - **Levels:** User (u), Group (g), Others (o).

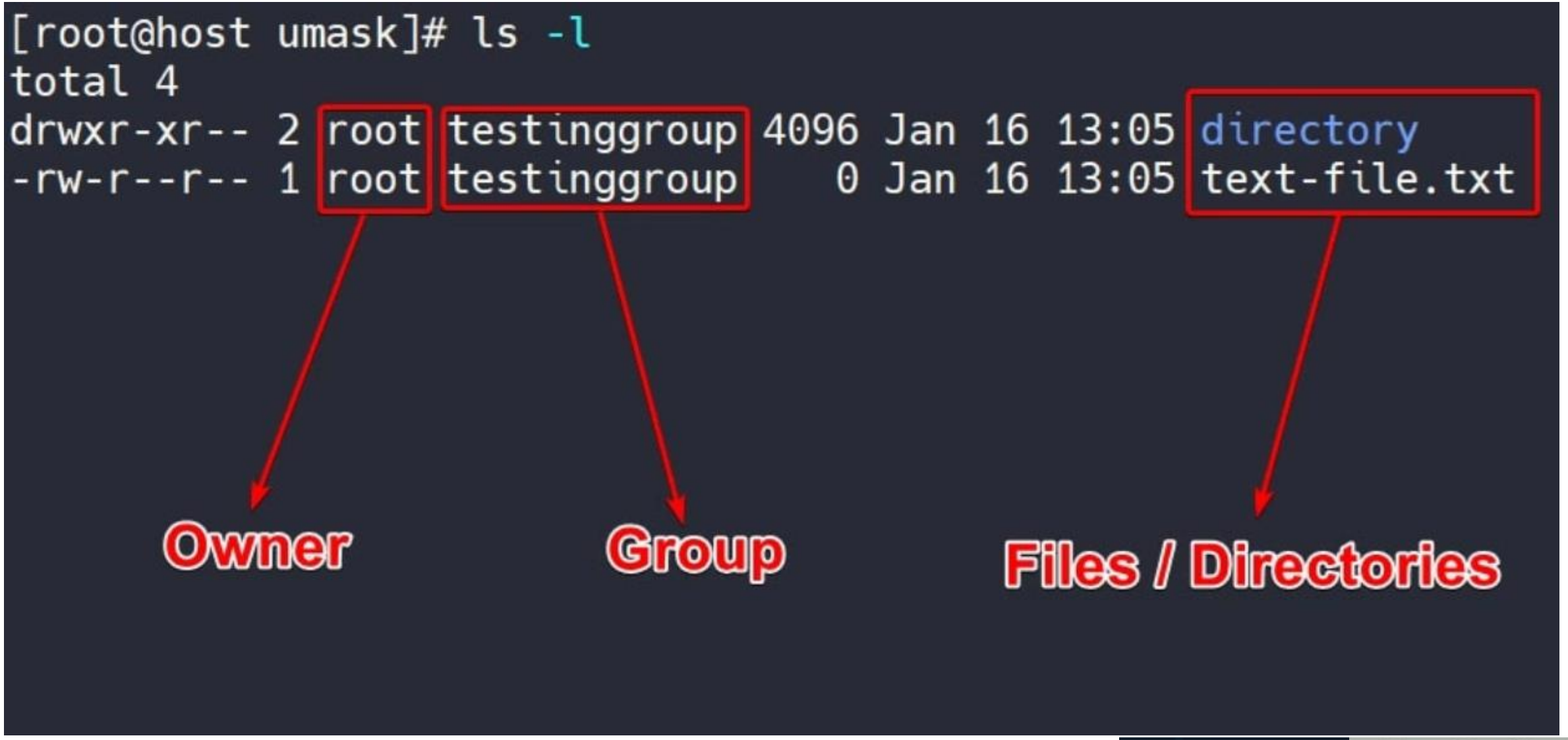
```
-rwxr-xr-- 1 user group size date file
```

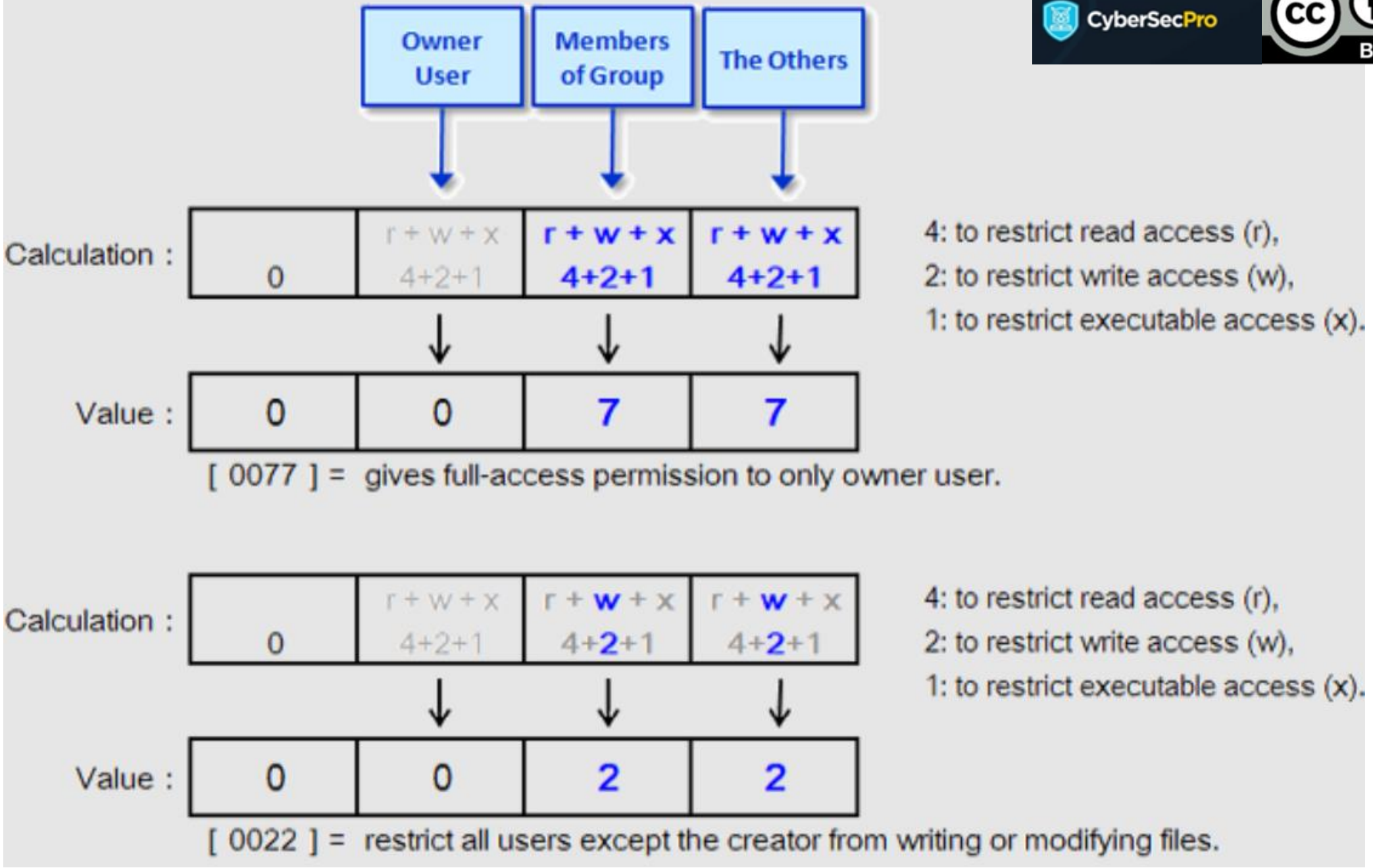
Command Summary:

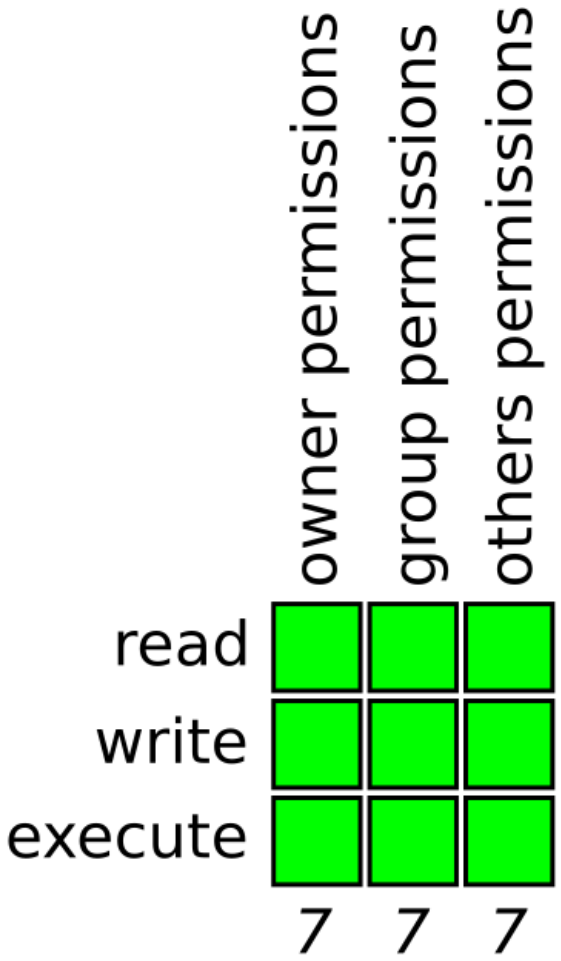
ls -l: View file permissions.
chmod: Modify file permissions.
chown: Change file ownership.
umask: Default permission setting for newly created files.



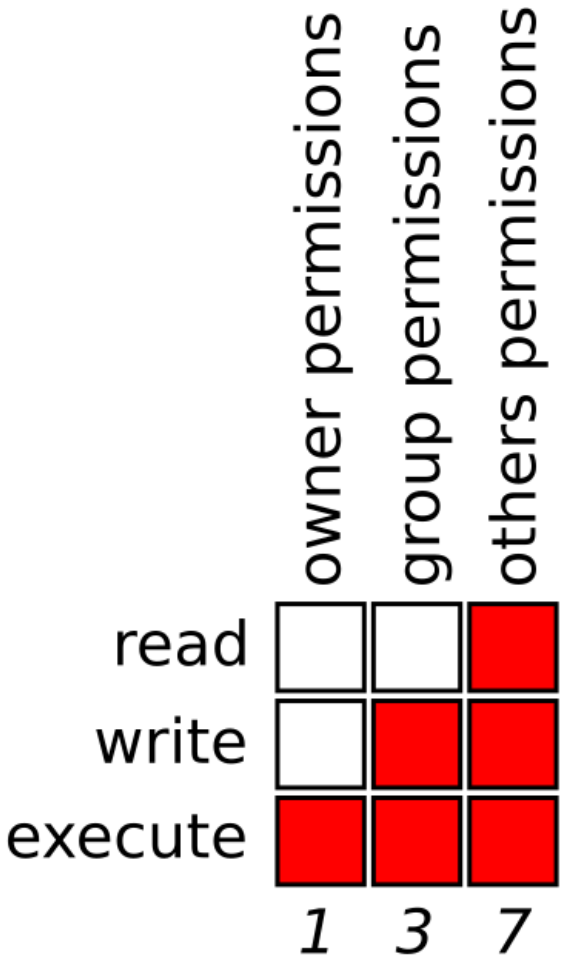
umask Value Octal (xyz)	Default File Permissions	666 - xyz	Default Directory Permissions	777 - xyz
000	rw-rw-rw	666	rwxrwxrwx	777
002	rw-rw-r--	664	rwxrwxr-x	775
022	rw-r--r--	644	rwxr-xr-x	755
026	rw-r-----	640	rwxr-x--x	751
046	rw--w----	620	rwx-wx--x	731
062	rw----r--	604	rwx--xr-x	715
066	rw-------	600	rwx--x--x	711
222	r--r--r--	444	r-xr-xr-x	555
600	---rw-rw-	066	--xrwxrwx	177
666	---------	000	--x--x--x	111
777	---------	000	---------	000



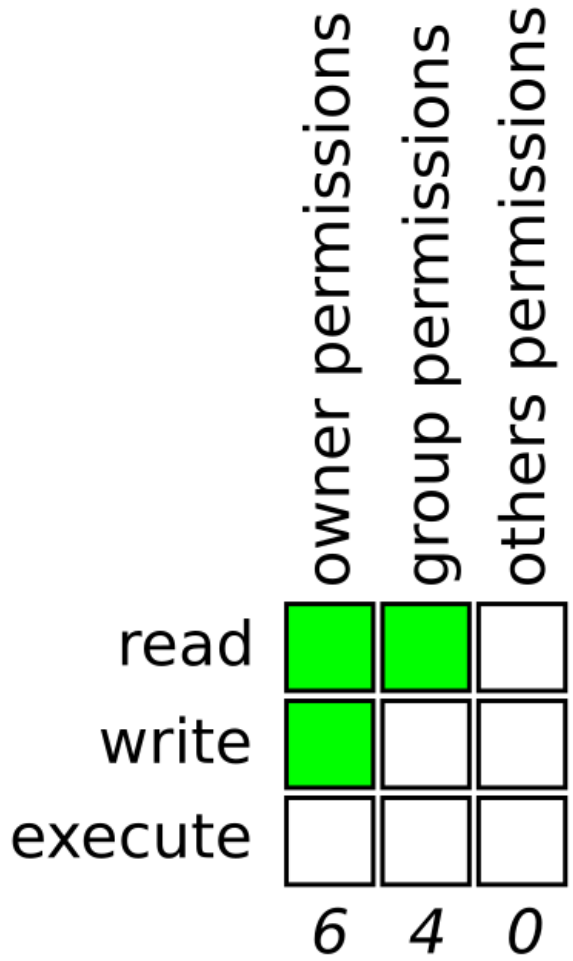




permissions of a new file before applying umask



umask



permissions of a new file after applying umask

```
[student@rhel9-ws test]$ umask 035

[student@rhel9-ws test]$ touch file

[student@rhel9-ws test]$ mkdir directory

[student@rhel9-ws test]$ ls -l
total 0
drwxr---w-. 2 student student 6 Feb 10 08:56 directory
-rw-r---w-. 1 student student 0 Feb 10 08:56 file

[student@rhel9-ws test]$ umask
0035
```

File Ownership and Permissions in Linux

User (Owner): The individual who owns the file.
Group: A set of users who share access to the file.
Others: Everyone else who is not the owner or in the group.

Common Permissions

r (read): Permission to view the file's content.
w (write): Permission to modify the file.
x (execute): Permission to run the file as a program or script.

Operating System Security: Operating system security is essential to safeguard the integrity, confidentiality, and availability of a system's resources. It includes mechanisms that enforce restrictions and monitor access to resources to prevent malicious or unauthorized activities.

Key Concepts

Confidentiality: Ensuring that information is only accessible to those who are authorized.
Integrity: Ensuring that data is not altered by unauthorized users or malicious software.
Availability: Ensuring that resources are accessible and functional when needed.

Core Elements of OS Security

Authentication: Verifying the identity of users or systems accessing the OS.
Authorization: Granting or denying access to system resources based on user identity and roles.
Auditing: Tracking activities in the system to detect security violations or misuse.

Types of Security Mechanisms

Access Control Lists (ACLs): A more granular method of setting permissions for files, directories, or resources.
Mandatory Access Control (MAC): Policies that restrict the access of users to resources, based on security labels and classifications (e.g., SELinux, AppArmor).
Discretionary Access Control (DAC): Users have control over their own resources and can grant or revoke access to others (e.g., Linux file permissions).

File Permission Format

Permissions are displayed in a 10-character string: -rwxr-xr--

- The first character is the file type (e.g., - for regular files, d for directories).
- The next three characters represent the owner's permissions.
- The following three characters represent the group's permissions.
- The last three characters represent others' permissions.

Changing File Permissions

chmod (Change Mode): Modify file permissions.
    Example: chmod u+x file.txt (adds execute permission for the owner).
chown (Change Owner): Change file owner or group.
    Example: chown user:group file.txt
umask (User Mask): Default permission settings for new files and directories.
    Example: umask 022 (sets permissions to 755 for directories and 644 for files).

Basic Hardening Practices for Linux

- Remove unnecessary services and software.
- Regularly update and patch the system.
- Enable and configure a firewall (iptables, ufw).
- Enforce strong password policies (/etc/security/pwquality.conf).
- Disable root login and use sudo for administrative tasks.
- Log and monitor activities (/var/log).
- Secure critical system files using advanced tools (e.g., chattr).

Key Hardening Practices

- **File System Security:** Ensure proper file permissions are set for all files, especially critical system files.
- **Minimizing Installed Software:** Only install software necessary for system operation to limit potential vulnerabilities.
- **Patching and Updates:** Regularly apply updates and security patches to the operating system and applications.
- **Secure System Configurations:** Disable unused services and ensure that security configurations (like ***firewalls***, ***SELinux***, or ***AppArmor***) are properly set up.
- **Access Control Policies:** Use access control mechanisms to limit the actions users can perform and what they can access.
- **Audit Logging:** Keep logs of system activity and monitor for suspicious actions.

Questions

1. What is a process and process table?
2. What are the different states of the process?
3. What is a Thread?
4. What are the differences between process and thread?
5. What are the benefits of multithreaded programming?
6. What is Thrashing?
7. What is Buffer?
8. What is virtual memory?
9. Explain the main purpose of an operating system?
10. What is demand paging?
11. What is a kernel?
12. What are the different scheduling algorithms?
13. Describe the objective of multi-programming?
14. What is the time-sharing system?
15. What problem we face in computer system without OS?
16. Give some benefits of multithreaded programming?
17. Briefly explain FCFS?
18. What is the RoundRobin scheduling algorithm?
19. Enumerate the different RAID levels?
20. What is Banker's algorithm?
21. State the main difference between logical and physical address space?
22. How does dynamic loading aid in better memory space utilization?
23. What are overlays?

Questions (2)

24. What is fragmentation?
25. What is the basic function of paging?
26. How does swapping result in better memory management?
27. Write a name of classic synchronization problems?
28. What is the Direct Access Method?
29. When does thrashing occur?
30. What is the best page size when designing an operating system?
31. What is multitasking?
32. What is caching?
33. What is spooling?
34. What is the functionality of an Assembler?

Linux Filesystem

The Linux filesystem follows a hierarchical structure, starting from the root directory (/). Below is a list of common Linux directories and their purposes:

1. / (Root Directory)

Purpose: The top-level directory containing all other files and directories.

2. /bin (Binaries)

Purpose: Essential command binaries required for basic system operation.
Examples: `ls`, `cp`, `rm`, `echo`.

3. /boot (Boot Loader Files)

Purpose: Files required for the system boot process.
Examples: Kernel (`vmlinuz`), bootloader configuration files (`grub`).

8. /lib64 (64-bit Libraries)

Purpose: Similar to `/lib`, but specifically contains 64-bit libraries for systems running a 64-bit architecture.
Examples: `libc.so` (64-bit C library).

9. /media (Removable Media)

Purpose: Mount point for removable media such as CDs, USB drives.
Examples: `/media/usb`, `/media/cdrom`.

10. /mnt (Temporary Mount Point)

Purpose: Temporary mount point for filesystems.
Examples: `/mnt/external_drive`.

11. /opt (Optional Software)

Purpose: Optional or third-party software packages.
Examples: `/opt/myapp`.

12. /proc (Process Filesystem)

Purpose: Virtual filesystem providing information about system processes and the kernel.
Examples: `/proc/cpuinfo`, `/proc/meminfo`.

4. /dev (Device Files)

Purpose: Special device files representing hardware devices (e.g., disks, terminals).
Examples: `/dev/sda` (disk), `/dev/tty` (terminal).

5. /etc (Configuration Files)

Purpose: System-wide configuration files.
Examples: `/etc/passwd`, `/etc/fstab`, `/etc/network/interfaces`.

6. /home (User Home Directories)

Purpose: Personal directories for individual users.
Examples: `/home/user1`, `/home/user2`.

7. /lib (Libraries)

Purpose: Shared libraries required by binaries in `/bin` and `/sbin`.
Examples: `libc.so.6`.

13. /root (Root User Directory)

Purpose: Home directory for the root user.
Examples: `/root/.bashrc`.

14. /run (Runtime Data)

Purpose: Information about the running system since the last boot.
Examples: `/run/lock`, `/run/user`.

15. /sbin (System Binaries)

Purpose: Essential system administration binaries.
Examples: `ifconfig`, `fdisk`, `fsck`.

16. /srv (Service Data)

Purpose: Data served by the system, such as web or FTP server content.
Examples: `/srv/www`, `/srv/ftp`.

17. /sys (System Information)

Purpose: Virtual filesystem for interacting with the kernel and hardware devices.
Examples: `/sys/class`, `/sys/block`.

18. /tmp (Temporary Files)

Purpose: Temporary files created by the system or users.
Examples: Automatically cleared on reboot.

19. /usr (User Programs)

Purpose: Secondary hierarchy for user applications and utilities.
Examples:

`/usr/bin` (User binaries)

`/usr/sbin` (System administration binaries)

`/usr/lib` (Libraries)

`/usr/local` (Locally installed programs)

20. /var (Variable Files)

Purpose: Files that are expected to grow, such as logs, caches, and spool files.
Examples:

`/var/log` (Log files)

`/var/spool` (Spool files, e.g., mail)

`/var/tmp` (Temporary files)

21. /lost+found (Recovered Files)

Purpose: Directory used by the filesystem to store recovered files after a crash.
Examples: Found on each partition formatted with ext-based filesystems.

Linux vs. Windows Filesystem

While both Linux and Windows have their file systems, they differ in structure and functionality:

Linux Filesystem: (i) Hierarchical structure with a single root (/). (ii) Case-sensitive filenames (e.g., `file.txt` and `File.txt` are different). (iii) Supports special files such as device files (`/dev`), symbolic links, and named pipes. (iv) Common filesystem types include EXT4, XFS, and Btrfs.

Windows Filesystem: (i) Uses drive letters (e.g., C:, D:). (ii) Case-insensitive filenames (e.g., `File.txt` and `file.txt` are the same). (iii) Relies on extensions like `.exe`, `.txt`, etc., to represent file types. (iv) Common filesystem types include NTFS and FAT32.

Windows Registry vs Linux Configuration Files

Windows Registry: (i) Centralized hierarchical database for system and application settings. (ii) Organized into keys and values.
Linux Configuration Files: (i) Plain-text files in directories like `/etc/`. (ii) Easy to read and edit with key-value pairs.

Topic 1: Auditing with Lynis

Objective: Learn how to use Lynis to audit the system's security posture. (i) Students will understand the theory behind security audits with Lynis, (ii) They will develop hands-on skills in auditing, interpreting results, and applying fixes to harden the system's security posture.

Outcome: By the end of this module, participants will: (i) Understand how to use Lynis for system security auditing. (ii) Learn to interpret audit results and prioritize fixes. (iii) Gain hands-on experience in applying real-world hardening techniques.

1.1 Overview of Lynis Security Auditing Tool

What is Lynis? Open-source, command-line-based security auditing tool for Linux, macOS, and Unix-based systems. (i) It is designed to assess system security posture, identify vulnerabilities, and recommend hardening steps.

Use Cases: (i) System compliance checks, (ii) Vulnerability assessments, (iii) Hardening servers (e.g., web, database, application servers).

Key Features: (i) Modular approach for various system components (e.g., kernel, file systems, networking), (ii) Generates detailed security reports with actionable recommendations, (iii) Lightweight, with no agent installation required.

1.2 Configuring and Running Security Scans

Installing Lynis (i) Available as a package in most distributions or directly downloadable from its website, (ii) Requires root or sudo privileges for a comprehensive audit.

```
# Perform a full system audit.
lynis audit system
# Focus on security-specific checks.
lynis audit security

# Display configuration profiles.
lynis show profiles
```

Customization Options: (i) Use profiles to configure the scope of the audit, (ii) Exclude specific tests if needed (e.g., for performance reasons).

Pre-scan Recommendations: (i) Backup critical data before running audits, (ii) Run Lynis in a non-production environment first to test configurations.

1.3 Interpreting Lynis Output and Reports

Report Overview: (i) Categories in the Output:

General system information.

Kernel and hardware security checks.

File permissions and ownership.

Installed software vulnerabilities.

Network and firewall configurations. (ii) Risk Levels: (i) Warning: Requires immediate attention, (ii) Suggestion: Recommended for best practices, (iii) Hardening Index: (i) Scored value indicating the system's overall security posture (e.g., "Hardening Index: 65/100").

Action Plan: (i) Focus on the most critical warnings first, (ii) Implement suggestions incrementally and retest after changes.

Documentation: (i) Log findings for compliance and auditing purposes.

Topic 1: Lab Exercises

1. Install and Configure Lynis

Step 1: Update your system:

```
sudo apt update && sudo apt upgrade -y    # For Debian/Ubuntu-based systems
sudo yum update -y                        # For CentOS/RHEL-based systems
```

Step 2: Install Lynis:

```
sudo apt install lynis  # On Debian/Ubuntu
sudo yum install lynis  # On CentOS/RHEL
```

Step 3: Verify installation:

```
lynis --version
```

Alternate Installation: For systems without package managers:

```
wget https://downloads.cisofy.com/lynis/lynis-latest.tar.gz
tar -xvzf lynis-latest.tar.gz
cd lynis
sudo ./lynis audit system
```

2. Run a System Audit

Step 1: Run the full system audit:

```
sudo lynis audit system
```

Step 2: Examine the output: (i) Look for "Warnings" and "Suggestions" in the report, (ii) Note the Hardening Index for your system.

Understanding the Output:

After running `sudo lynis audit system`, you'll see a report with categories like: (i) **Warnings:** Critical security issues (e.g., no firewall, weak passwords). (ii) **Suggestions:** Recommendations for improving system security (e.g., disable unused services). (iii) **Hardening Index:** A score between 0 and 100 indicating system security strength.

Example Output:

```
Hardening Index: 65 [Security level: Moderate]
Warnings:
1. Firewall not detected [test:FW-7526].
2. No password policy configured for user accounts [test:AUTH-9267].

Suggestions:
- Install and configure a firewall (e.g., UFW or iptables).
- Set up a password policy (e.g., PAM module for password complexity).
```

Report Location:

(i) Log file: /var/log/lynis.log. (ii) Report file: /var/log/lynis-report.dat.

3. Analyze and Apply Security Recommendations

Step 1: Locate the detailed audit log:

```
cat /var/log/lynis.log
```

Step 2: Identify key warnings. Example warning: Warning: No firewall detected [test:FW-7526] Recommendation: Install and configure a firewall like UFW or iptables.

Example 0: Install and Configure a Firewall

If Lynis warns about no firewall:

```
sudo apt install ufw
sudo ufw enable
sudo ufw allow ssh
```

Verify firewall status:

```
sudo ufw status
```

Example 1: File permissions

```
chmod 640 /etc/ssh/sshd_config
```

Example 2: Disable unnecessary services

```
sudo systemctl disable apache2.service
```

Example 3: Strengthen Password Policies If Lynis recommends configuring password policies, edit '/etc/security/pwquality.conf':

minlen = 12 minclass = 3 retry = 3

Test the configuration with 'passwd' command.

Example 4: Disable Unnecessary Services Lynis might detect unnecessary services running (e.g., FTP server):

```
sudo systemctl stop vsftpd
sudo systemctl disable vsftpd
```

Example 5: Kernel Parameter Hardening Recommendation: Harden kernel parameters for network security. Add the following to '/etc/sysctl.conf':

net.ipv4.tcp$_{s}yncookies = 1net.ipv4.conf.all.rp_{f}ilter = 1net.ipv4.icmp_{e}cho_{i}gnore_{b}roadcasts = 1$ Apply changes:

```
sudo sysctl —p
```

Example 6: Update Outdated Software Lynis often highlights outdated software. Update all packages:

```
sudo apt update && sudo apt upgrade —y
```

Example 7: Secure SSH Configuration Lynis might recommend hardening SSH. Edit '/etc/ssh/sshd$_{c}onfig$' : PermitRootLogin no PasswordAuthentication no Protocol 2
Restart SSH:

```
sudo systemctl restart sshd
```

Re-audit the System After applying changes, run another audit to verify improvements:

```
sudo lynis audit system
```

Additional Real-World Examples

(i) **Enterprise Server Auditing:** A Lynis scan revealed multiple vulnerabilities, including open ports for deprecated services like Telnet. After recommendations were applied (e.g., disabling Telnet, installing a firewall, and patching software), the hardening index improved from 55 to 85.

(ii) **Public Cloud Hardening:**

A public cloud instance running Apache was audited. Lynis detected: (a) Weak SSL configuration. (b) Lack of HTTP headers (e.g., Content-Security-Policy). (c) Missing web application firewall (WAF).

Fixes: (a) Updated SSL/TLS protocols in `httpd.conf`. (b) Enabled security headers using `mod_headers`. (c) Deployed a WAF like ModSecurity.

(iii) **Small Business Office:**

Lynis flagged SMB (Samba) sharing services as vulnerable due to outdated versions. The system admin disabled SMB v1, upgraded Samba, and limited access via firewall rules.

Real-World Examples

1. Healthcare Systems:

(i) Lynis is used to ensure compliance with regulations like HIPAA by auditing server configurations and network setups.

2. Corporate Servers:

(i) Conducting quarterly audits to identify open ports, services, and misconfigurations in enterprise environments.

3. Web Hosting Providers:

(i) Auditing web servers (Apache, Nginx) to check for outdated software or weak SSL configurations.

Other Cases

(i) **Case 1: Financial Institutions** – Auditing payment gateway servers to ensure compliance with PCI DSS and detecting misconfigured SSL/TLS settings. (ii) **Case 2: Government Systems** – Periodic audits of defense servers to identify weak configurations. (iii) **Case 3: Small Businesses** – Hardening cloud-based systems to reduce attack surfaces.

Topic 2: Auditing using Auditctl

Objective: Learn how to audit Linux systems using the Linux Audit Framework (LAF) with auditctl for real-time auditing and ausearch for analyzing audit logs.

Outcome: By the end of this module, participants will: (i) Understand the Linux Audit Framework and its key components. (ii) Create and manage real-time audit rules using `auditctl`. (iii) Query and analyze audit logs with `ausearch`. (iv) Gain experience in applying audit techniques for real-world security scenarios.

Section 1: Lecture Topics

1.1 Linux Audit Framework (LAF) and Configuration

What is the Linux Audit Framework? The Linux Audit Framework (LAF) provides tools and features for monitoring and logging security-relevant events on a Linux system: (i) Records user actions, system changes, and other activities for compliance and security. (ii) Helps administrators track unauthorized access or suspicious activities.

Key Components of LAF: (i) `auditctl`: Command-line tool for managing audit rules in real time. (ii) `auditd`: The audit daemon that logs events. (iii) `ausearch`: Tool for querying and analyzing audit logs. (iv) Audit rules: Rules that define what activities should be monitored.

Audit Log Location: Audit logs are stored in `/var/log/audit/audit.log`. Example log entry:

```
type=USER_CMD msg=audit(1673020200.786:324): pid=1524 uid=0 auid=1000 ses=2 msg='cwd="/home↩
    /user" cmd="ls"'
```

Why Use Linux Auditing? (i) Compliance (e.g., PCI DSS, HIPAA). (ii) Detecting unauthorized file changes. (iii) Monitoring privileged command execution.

1.2 Creating Audit Rules with auditctl

Auditctl Command Overview: (i) Add, delete, and list audit rules in real-time. (ii) Audit rules can monitor files, commands, directories, and system calls.

Audit Rule Types: (i) File Audit Rules: Monitor access to specific files or directories. (ii) System Call Rules: Track low-level system operations. (iii) User Audit Rules: Log user actions (e.g., sudo commands).

Audit Rule Examples:

Monitor access to `/etc/passwd`:

```
auditctl -w /etc/passwd -p rwxa -k passwd_monitor
```

Explanation: (i) `-w`: Watch file or directory. (ii) `-p rwxa`: Monitor read, write, execute, and attribute changes. (iii) `-k passwd_monitor`: Add a custom key for easy search.

Audit all sudo commands:

```
auditctl -a always,exit -F arch=b64 -S execve -F euid=0 -k sudo_commands
```

Explanation: (i) `-a always,exit`: Audit system calls on exit. (ii) `-F arch=b64`: Target 64-bit architecture. (iii) `-S execve`: System call for executing commands. (iv) `-F euid=0`: Target actions by root user. (v) `-k sudo_commands`: Add a custom key.

1.3 Searching Audit Logs Using ausearch

What is ausearch? Ausearch is used to query and analyze audit logs based on different criteria such as event types, users, and file paths.

Common ausearch Options: (i) Search by event type:

```
ausearch -m USER_CMD
```

(ii) Search by key:

```
ausearch -k sudo_commands
```

(iii) Search by time range:

```
ausearch -ts today
```

Real-World Use Cases: (i) Identify all users who accessed a sensitive file. (ii) Track the execution of critical administrative commands (e.g., sudo). (iii) Investigate suspicious activity (e.g., unauthorized file modification).

Section 2: Lab Exercises

2.1 Set Up Basic Audit Rules (e.g., Monitor sudo Commands)

Ensure Audit Framework is Installed:

```
sudo apt install auditd audispd-plugins
sudo systemctl start auditd
sudo systemctl enable auditd
```

Add a Rule to Monitor sudo Commands:

```
sudo auditctl -a always,exit -F arch=b64 -S execve -F euid=0 -k sudo_commands
```

Test the Rule: (i) Run a sudo command:

```
sudo ls /root
```

(ii) Check the audit log:

```
sudo ausearch -k sudo_commands
```

2.2 Use auditctl to Add Rules for File Access

Monitor Access to `/etc/shadow`:

```
sudo auditctl -w /etc/shadow -p rwa -k shadow_monitor
```

Test the Rule: (i) Try reading `/etc/shadow` (as root):

```
sudo cat /etc/shadow
```

(ii) Check the audit log:

```
sudo ausearch -k shadow_monitor
```

Monitor a Custom Directory: (i) Create a directory:

```
mkdir ~/important_files
```

(ii) Add a rule to monitor it:

```
sudo auditctl -w ~/important_files -p rwxa -k important_files_monitor
```

2.3 Use ausearch to Query Audit Logs

Search Logs by Key:

```
sudo ausearch -k sudo_commands
```

Filter by User:

```
sudo ausearch -ua $(id -u john)
```

Search by File Access:

```
sudo ausearch -k shadow_monitor
```

Search by Time Range:

```
sudo ausearch -ts recent
```

Additional Advanced Examples

Example 1: Detect Unauthorized File Modifications

```
sudo auditctl -w /etc/ssh/sshd_config -p wa -k ssh_config_monitor
```

Modify the file and search logs:

```
sudo nano /etc/ssh/sshd_config
sudo ausearch -k ssh_config_monitor
```

Example 2: Monitor Network Configuration Changes

Add a rule to track changes to `/etc/network/interfaces`:

```
sudo auditctl -w /etc/network/interfaces -p wa -k network_config
```

Search for changes:

```
sudo ausearch -k network_config
```

Topic 3: Security Hardenning

Objective

Learn how to implement system hardening practices to secure a Linux environment, focusing on configuring security settings, limiting user access, and disabling unnecessary services.

1. Configuring Security Settings (sysctl Hardening)

What is sysctl?
sysctl is a utility for examining and modifying kernel parameters at runtime. It is commonly used to adjust kernel settings related to networking, system performance, and security.

Security-Related sysctl Parameters:

(i) Protection against SYN Floods:
Increase the size of the backlog queue to prevent SYN flooding attacks.

```
sudo sysctl -w net.ipv4.tcp_max_syn_backlog=2048
```

(ii) Disable IP Forwarding (Preventing IP Spoofing):
When the system is not supposed to forward packets between networks (e.g., it's not a router).

```
sudo sysctl -w net.ipv4.ip_forward=0
```

(iii) Disable ICMP Redirect Acceptance:
Prevent the system from accepting ICMP redirects, which could be used in man-in-the-middle attacks.

```
sudo sysctl -w net.ipv4.conf.all.accept_redirects=0
```

(iv) Log Suspicious Connections:
Enable logging for abnormal connection attempts.

```
sudo sysctl -w net.ipv4.tcp_rfc1337=1
```

(v) Disable Source Routing:
This prevents IP source routing, a technique often used in network attacks.

```
sudo sysctl -w net.ipv4.conf.all.accept_source_route=0
```

(vi) Enable TCP SYN Cookies (Defend Against DoS Attacks):
SYN cookies help prevent DoS (Denial of Service) attacks.

```
sudo sysctl -w net.ipv4.tcp_syncookies=1
```

Making sysctl Changes Persistent:
Changes made with sysctl are not persistent across reboots by default. To make changes permanent, add them to /etc/sysctl.conf or create a new configuration file in /etc/sysctl.d/:

```
echo "net.ipv4.tcp_syncookies=1" | sudo tee -a /etc/sysctl.conf
sudo sysctl -p
```

2. Limiting User Access with /etc/security/limits.conf

What is /etc/security/limits.conf?

The /etc/security/limits.conf file is used to configure resource limits for users and groups. These limits can help mitigate denial-of-service (DoS) attacks and prevent misuse of system resources.

Common Resource Limits:

(i) Maximum number of file descriptors (nofile):
The maximum number of files a user can have open simultaneously.

```
sudo vim /etc/security/limits.conf
```

Add:

```
user_name soft nofile 1000
user_name hard nofile 2000
```

(ii) Maximum number of processes (nproc):
Limit the number of processes a user can spawn.

```
sudo vim /etc/security/limits.conf
```

Add:

```
user_name soft nproc 500
user_name hard nproc 1000
```

(iii) Example Configuration:

```
# Increase file descriptor limits for user 'john'
john soft nofile 1024
john hard nofile 4096

# Limit processes for user 'john'
john soft nproc 100
john hard nproc 200
```

Best Practices:

(i) Set reasonable limits to avoid users or processes consuming excessive system resources.

(ii) Use soft limits for regular operations and hard limits to enforce strict maximums.

Outcome

By the end of this week, participants will have:

(i) Configured system security settings using sysctl.

(ii) Limited user access and set resource limits using /etc/security/limits.conf.

(iii) Disabled unnecessary services to minimize security risks.

(iv) Applied various system hardening practices that contribute to a more secure Linux environment.

3. Disabling Unnecessary Services

Why Disable Unnecessary Services?

Disabling unnecessary services reduces the attack surface of a system, minimizing the potential entry points for attackers.

Identifying Running Services:

To see a list of active services and their status, use the systemctl command:

```
systemctl list-units --type=service --state=running
```

Disabling Unnecessary Services:

To disable a service that is not needed, you can use the systemctl disable command:

```
sudo systemctl disable telnet.socket
sudo systemctl stop telnet.socket
```

Common services to disable if not needed:

(i) Telnet: Telnet sends data in plaintext and should be replaced with SSH.

(ii) FTP: Use SFTP instead for secure file transfers.

(iii) Avahi-daemon: Multicast DNS and service discovery (useful for local networks, but should be disabled on servers).

Disabling Specific Services: Example: Disable the FTP service (vsftpd):

```
sudo systemctl disable vsftpd
sudo systemctl stop vsftpd
```

Ensure Services Do Not Start After Reboot:
To ensure a service remains disabled after reboot, use:

```
sudo systemctl disable <service-name>
```

Checking for Disabled Services After Reboot:
After reboot, check if a service has been disabled using:

```
systemctl is-enabled <service-name>
```

Lab Exercises

Exercise 1: Apply Basic System Hardening Practices

(i) Disable Root Login via SSH:
Edit the SSH configuration file /etc/ssh/sshd_config to disable root login:

```
sudo vim /etc/ssh/sshd_config
```

Set:

```
PermitRootLogin no
```

(ii) Limit User Logins (e.g., Max Concurrent Sessions):
In /etc/security/limits.conf, set a limit for a specific user (e.g., john) to have a maximum of 3 concurrent sessions:

```
john hard maxlogins 3
```

Exercise 2: Disable Unnecessary Services (Using systemctl)

(i) Disable FTP Service:
Disable and stop the FTP service (vsftpd):

```
sudo systemctl disable vsftpd
sudo systemctl stop vsftpd
```

(ii) Disable Telnet:
Disable and stop the Telnet service:

```
sudo systemctl disable telnet.socket
sudo systemctl stop telnet.socket
```

Exercise 3: Configure System Parameters for Security (sysctl)

(i) Set Kernel Parameters for Security:

```
sudo sysctl -w net.ipv4.ip_forward=0
```

(ii) Make sysctl Changes Persistent:
Add the changes to /etc/sysctl.conf or a file in /etc/sysctl.d/ to persist across reboots:

```
sudo vim /etc/sysctl.conf
```

Add the following lines:

```
net.ipv4.ip_forward=0
net.ipv4.conf.all.accept_redirects=0
```

Topic 4: Log Management

Objective: Learn how to configure and manage logs, detect suspicious activity, and set up basic Intrusion Detection Systems (IDS) for monitoring security breaches.

Outcome: By the end of this week, participants will have:

Configured and managed system logs using Syslog and journald.

Identified suspicious activity in system logs and applied basic analysis tools.

Set up a basic IDS (OSSEC) to monitor and detect potential security breaches.

Configured log rotation to manage log file sizes and ensure continuous logging.

What is Syslog?

Syslog is a standard for logging system messages in Unix-like systems. It facilitates the centralization and management of logs from different system components.

Syslog messages are classified by facility (e.g., kernel messages, authentication, mail) and severity (e.g., emergency, alert, critical).

Syslog messages are typically stored in the /var/log/ directory (e.g., /var/log/auth.log, /var/log/syslog).

Syslog Configuration:

The Syslog daemon (rsyslog on most modern systems) listens for system log messages. It is configured via /etc/rsyslog.conf. To manage logging behavior (e.g., which logs should be stored or forwarded), you can modify this configuration file.

```
auth,authpriv.*    /var/log/auth.log
```

What is journald?

journald is the logging service provided by systemd. It offers structured logging, advanced filtering, and centralization of logs in the /var/log/journal/ directory.

Logs can be viewed using the journalctl command.

Unlike Syslog, journald stores logs in a binary format and offers advanced filtering (e.g., by time range, unit, priority).

You can configure journald using /etc/systemd/journald.conf.

Configuring journald for Log Management:

You can configure how long logs are retained and how much disk space is used by editing /etc/systemd/journald.conf. Example configuration for limiting disk usage:

```
sudo vim /etc/systemd/journald.conf
```

Add or modify the following lines:

```
SystemMaxUse=1G
SystemKeepFree=500M
```

Key Logs to Monitor for Suspicious Activity:

/var/log/auth.log: Authentication attempts and user logins.

/var/log/syslog: General system messages, including kernel and application logs.

/var/log/messages: System-related messages and kernel logs.

/var/log/secure: Security-related events, such as sudo command usage.

/var/log/cron: Cron job execution logs.

Common Suspicious Activity Indicators:

Failed Login Attempts: Multiple failed login attempts in auth.log or secure can indicate brute-force attempts. Example:

```
Jun 25 14:30:00 server sshd[1234]: Failed password for root from 192.168.1.1 port 22 ↩
    ssh2
```

Unexpected User Activity: Watch for unexpected users or root access to critical files. Example:

```
Jun 25 14:40:00 server sudo:   john : TTY=pts/1 ; PWD=/home/john ; USER=root ; COMMAND↩
    =/bin/
```

Abnormal System Behavior: Look for changes in file ownership, system crashes, or unfamiliar process executions.

Using grep and awk to Analyze Logs:

You can use grep to filter log files for suspicious activity. Example command to find failed SSH login attempts:

```
grep "Failed password" /var/log/auth.log
```

(3) Setting Up Intrusion Detection Systems (IDS) like OSSEC

What is OSSEC?

OSSEC is an open-source, host-based Intrusion Detection System (HIDS) that performs log analysis, file integrity checking, and real-time alerting to detect suspicious activity on a system.

OSSEC Features:

Log Monitoring: Automatically analyzes logs from various sources, including Syslog, Apache, SSH, and more.

File Integrity Checking: Monitors critical files and directories for changes.

Real-time Alerting: Sends alerts via email or integrates with SIEM tools for security monitoring.

Installing OSSEC:

OSSEC can be installed from its official repository or using a package manager:

```
sudo apt update
sudo apt install ossec-hids
```

Basic Configuration of OSSEC:

After installation, you can configure OSSEC by editing /var/ossec/etc/ossec.conf:

Define the rules for log analysis and file integrity monitoring.

Configure email alerts for security events.

Example configuration:

```
<global>
  <email_notification>yes</email_notification>
  <email_to>admin@example.com</email_to>
</global>

<syscheck>
  <directories>/etc,/bin,/usr/bin</directories>
</syscheck>
```

Running OSSEC:

Once configured, you can start the OSSEC service:

```
sudo systemctl start ossec
sudo systemctl enable ossec
```

Reviewing OSSEC Alerts:

OSSEC will store alerts in /var/ossec/logs/alerts/. You can review them using the cat or less commands:

```
sudo less /var/ossec/logs/alerts/alerts.log
```

Lab Exercise

Exercise 1: Review and Analyze System Logs

View Authentication Logs (/var/log/auth.log): Use grep to look for suspicious login attempts:

```
    sudo grep "Failed password" /var/log/auth.log
```

Analyze Syslog Logs (/var/log/syslog): Search for unusual system events:

```
    sudo grep "sshd" /var/log/syslog
```

Search for Cron Jobs in /var/log/cron: Look for unexpected cron jobs that could indicate a malicious attempt:

```
sudo grep cron /var/log/cron
```

Exercise 2: Set Up Basic Log Rotation Using Logrotate

Configure Logrotate for a Custom Log File: Open or create a configuration file in /etc/logrotate.d/:

```
sudo vim /etc/logrotate.d/myapp
```

Example configuration for rotating logs:

```
/var/log/myapp.log {
    weekly
    rotate 4
    compress
    missingok
    notifempty
}
```

Test Logrotate: You can force a log rotation with the following command:

```
sudo logrotate /etc/logrotate.conf --debug
```

Exercise 3: Use journalctl to Search Logs

View All Logs:

```
sudo journalctl
```

Filter Logs by Time: To view logs from the last boot:

```
sudo journalctl -b
```

Search for Specific Service Logs: For SSH service logs:

```
sudo journalctl -u ssh
```

Exercise 4: Integrate OSSEC for Log Monitoring

Install and Configure OSSEC: Follow the installation and configuration steps mentioned in the lecture to install OSSEC and configure it to monitor specific logs and directories.

Start OSSEC and Review Alerts:

```
sudo systemctl start ossec
sudo systemctl enable ossec
sudo less /var/ossec/logs/alerts/alerts.log
```

Firewalls: Iptables & UFW Configuration

Objective: Learn how to configure and manage firewalls using iptables and ufw to secure a Linux system by controlling incoming and outgoing network traffic.

Outcome: By the end of this week, participants will: (i) Understand the fundamentals of packet filtering and firewall rules. (ii) Be able to configure and manage firewall rules using both iptables and ufw. (iii) Implement logging to monitor for suspicious activity on the system. (iv) Gain hands-on experience in blocking/allowing services and protecting the system from unauthorized access.

Iptables and Packet Filtering

What is Iptables?

```
iptables is a powerful firewall utility that allows system administrators to configure ↩
    rules for filtering network traffic on a Linux system. It operates at the network layer↩
     (Layer 3) and transport layer (Layer 4) of the OSI model.
```

Packet Filtering Mechanism:

Iptables works by inspecting network packets and comparing them against a set of defined rules.

Each rule specifies a set of conditions (e.g., source IP, destination port, protocol) and an action (e.g., ACCEPT, DROP, REJECT).

Rules are organized into chains (INPUT, OUTPUT, FORWARD), and each chain has its default policy.

Basic Iptables Terminology:

Chain: A list of rules applied to packets. Common chains include:

INPUT: Controls inbound traffic to the system.

OUTPUT: Controls outbound traffic from the system.

FORWARD: Controls traffic that is routed through the system.

Target: The action to take when a packet matches a rule (e.g., ACCEPT, DROP).

Rule: A condition specifying when to take an action (e.g., allow traffic from a specific IP).

Example Iptables Rule:

```
sudo iptables -A INPUT -p tcp --dport 22 -j ACCEPT
```

This rule allows incoming traffic on port 22 (SSH) over TCP.

Viewing Iptables Rules:

```
sudo iptables -L
```

Persisting Iptables Rules: Iptables rules are not persistent by default (they are cleared upon reboot). To save the current rules:

```
sudo iptables-save > /etc/iptables/rules.v4
```

2. **Configuring and Managing Firewalls Using UFW (Uncomplicated Firewall)**

What is UFW? ufw is a front-end for iptables that simplifies firewall management. It is designed to provide an easy-to-use interface for managing firewall rules while still leveraging the underlying iptables system.

Basic UFW Commands:

Enable UFW:

```
sudo ufw enable
```

Disable UFW:

```
sudo ufw disable
```

Check UFW Status:

```
sudo ufw status
```

Allow a Service (e.g., SSH on port 22):

```
sudo ufw allow 22/tcp
```

Deny a Service (e.g., HTTP on port 80):

```
sudo ufw deny 80/tcp
```

Allow a Service by Name (e.g., SSH):

```
sudo ufw allow ssh
```

UFW Profiles: ufw supports predefined service profiles, which are convenient to allow or deny services by name (e.g., ssh, http, etc.).

To see the available profiles:

```
sudo ufw app list
```

To allow a service by profile (e.g., HTTP):

```
sudo ufw allow 'Apache'
```

UFW Logging: To enable logging:

```
sudo ufw logging on
```

The log is stored in /var/log/ufw.log.

3. **Blocking and Allowing Services Using Firewall Rules**

Allowing SSH Access: Allow incoming SSH connections (port 22) using ufw:

```
sudo ufw allow 22/tcp
```

Blocking HTTP Service: Block incoming HTTP traffic on port 80:

```
sudo ufw deny 80/tcp
```

Allowing Specific IP Addresses: Allow only specific IP addresses to access your system (e.g., allowing only 192.168.1.10 to SSH into the server):

```
sudo ufw allow from 192.168.1.10 to any port 22
```

Allowing Specific IP Addresses: Allow only specific IP addresses to access your system (e.g., allowing only 192.168.1.10 to SSH into the server):

```
sudo ufw allow from 192.168.1.10 to any port 22
```

Allowing Subnets or Ranges: Allow a range of IP addresses (e.g., 192.168.0.0/24):

```
sudo ufw allow from 192.168.0.0/24 to any port 22
```

Limiting Connections (Rate Limiting): Protect SSH by limiting the number of connections to prevent brute-force attacks:

```
sudo ufw limit ssh
```

Allowing Services Temporarily: You can set rules for temporary periods. However, UFW does not support this directly, but you can use at or cron for scheduling.

Lab Exercises

Exercise 1: Configure Firewall Rules Using Iptables

(i) Allow Incoming SSH Connections: Add an iptables rule to allow SSH (port 22) from any IP address:

```
sudo iptables –A INPUT –p tcp ––dport 22 –j ACCEPT
```

(ii) Block HTTP Traffic (Port 80): Add an iptables rule to block incoming HTTP traffic:

```
sudo iptables –A INPUT –p tcp ––dport 80 –j DROP
```

(iii) Allow SSH from a Specific IP Address: Allow only IP 192.168.1.100 to access SSH:

```
sudo iptables –A INPUT –p tcp –s 192.168.1.100 ––dport 22 –j ACCEPT
```

(iv) List the Current Iptables Rules:

```
sudo iptables –L
```

(v) Save the Current Rules:

```
sudo iptables–save > /etc/iptables/rules.v4
```

Exercise 2: Set Up UFW to Allow/Deny Specific Services

(i) Allow SSH Connections Using UFW: Enable UFW and allow SSH:

```
sudo ufw enable
sudo ufw allow ssh
```

(ii) Deny HTTP Connections Using UFW: Deny incoming HTTP traffic:

```
sudo ufw deny 80/tcp
```

Exercise 3: Implement Firewall Logging for Suspicious Activity

(i) Enable UFW Logging: Turn on logging to monitor suspicious activity:

```
sudo ufw logging on
```

(ii) Review Firewall Logs: Access the firewall logs stored in /var/log/ufw.log:

```
sudo tail -f /var/log/ufw.log
```

(iii) Analyze Suspicious Activity: Review logs to detect any unauthorized access attempts or anomalous traffic patterns.

SSH Security & Second-Factor Authentication

Objective: Learn how to secure SSH access by configuring SSH with secure keys, disabling password-based authentication, and setting up second-factor authentication (2FA) for SSH using PAM (Pluggable Authentication Module).

Outcome: (i) Understand how to secure SSH access using SSH keys and disable password-based authentication.
(ii) Be able to configure and test two-factor authentication (2FA) for SSH, adding an additional layer of security to remote access.
(iii) Gain hands-on experience with SSH key management and the integration of PAM for enhanced authentication security.

1. Configuring SSH with Secure Keys

(i) **What is SSH?** Secure Shell (SSH) is a cryptographic network protocol used to securely connect to remote systems over an insecure network. It uses encryption to protect data integrity and confidentiality.
(ii) **Why SSH Keys?** SSH keys are a pair of cryptographic keys that provide a more secure and convenient method of authentication compared to passwords. They are less susceptible to brute-force attacks and man-in-the-middle attacks.
(iii) **Generating SSH Keys:** To create an SSH key pair, use the `ssh-keygen` command. This generates a private and a public key.

Listing 1: Generate SSH Key Pair

```
ssh-keygen -t rsa -b 4096 -C "your_email@example.com"
# -t rsa: Specifies the RSA algorithm.
# -b 4096: Specifies the key length (4096 bits for higher security).
# -C: Adds a comment (typically an email address) to the key.
```

Example of SSH key generation:

Listing 2: Example of SSH Key Generation

```
$ ssh-keygen -t rsa -b 4096 -C "user@example.com"
Generating public/private rsa key pair.
Enter file in which to save the key (/home/user/.ssh/id_rsa): [Press Enter to use default]
Enter passphrase (empty for no passphrase): [Enter passphrase or leave empty]
```

Resulting files: (i) The private key is saved to `~/.ssh/id_rsa` (keep this file secure).
(ii) The public key is saved to `~/.ssh/id_rsa.pub` (this is what you share with remote systems).

2. Disabling Password-Based Authentication

(i) **Why Disable Password-Based Authentication?** Disabling password-based authentication strengthens security because it eliminates the risk of password brute-force attacks. SSH keys are significantly more secure than passwords and harder to crack.
(ii) **Disabling Password Authentication in SSH:** Modify the SSH server configuration to disable password-based login.

Listing 3: Disable Password Authentication in SSH

```
sudo nano /etc/ssh/sshd_config
# Disable Password Authentication:
PasswordAuthentication no
# Enable Public Key Authentication (if not already enabled):
PubkeyAuthentication yes
# Restart the SSH service to apply the changes:
sudo systemctl restart ssh
```

3. Configuring Second-Factor Authentication (2FA) for SSH using PAM

(i) **What is Two-Factor Authentication (2FA)?** 2FA adds an extra layer of security to SSH login by requiring something you know (e.g., SSH key) and something you have (e.g., time-based one-time password from Google Authenticator or a similar service).

(ii) **How PAM Works:** The Pluggable Authentication Module (PAM) system allows for flexible authentication methods, including adding 2FA to SSH.

Steps to Set Up Google Authenticator for SSH 2FA:

Listing 4: Install the Google Authenticator PAM Module

```
sudo apt-get install libpam-google-authenticator
```

1. Run the `google-authenticator` command for the user who wants to enable 2FA. 2. Scan the QR code with the Google Authenticator app. 3. Configure PAM for SSH by editing `/etc/pam.d/sshd`:

Listing 5: Configure PAM for SSH

```
auth required pam_google_authenticator.so
```

4. Enable challenge-response authentication in `/etc/ssh/sshd_config`:

Listing 6: Enable Challenge-Response Authentication

```
ChallengeResponseAuthentication yes
sudo systemctl restart ssh
```

Lab Exercises

Exercise 1: Generate and Deploy SSH Keys

(i) Generate an SSH Key Pair:

```
ssh-keygen -t rsa -b 4096 -C "your_email@example.com"
```

(ii) Deploy the Public Key to the Remote Server:

```
ssh-copy-id user@remote_host
```

(iii) Test Key-Based SSH Login:

```
ssh user@remote_host
```

Exercise 2: Disable Password-Based Login and Enable Key-Based Login

(i) Disable Password Authentication:

```
sudo nano /etc/ssh/sshd_config
PasswordAuthentication no
PubkeyAuthentication yes
```

(ii) Restart SSH Service:

```
sudo systemctl restart ssh
```

(iii) Test Key-Based Login:

```
ssh user@remote_host
```

Exercise 3: Set Up Two-Factor Authentication (2FA) for SSH

(i) Install Google Authenticator PAM Module:

```
sudo apt-get install libpam-google-authenticator
```

(ii) Generate the QR Code for 2FA:

```
google-authenticator
```

(iii) Configure PAM for 2FA:

```
sudo nano /etc/pam.d/sshd
auth required pam_google_authenticator.so
```

(iv) Enable Challenge-Response Authentication:

```
sudo nano /etc/ssh/sshd_config
ChallengeResponseAuthentication yes
sudo systemctl restart ssh
```

(v) Test 2FA Authentication:

```
ssh user@remote_host
```

Privilege Escalation

Objective: Privilege escalation occurs when a user gains unauthorized privileges or higher levels of access than originally granted. In Linux, privilege escalation is typically used by attackers after compromising a low-privileged user account. Common techniques for privilege escalation include:

(i) **Sudo Misconfigurations**: Misconfigured sudo permissions can allow a user to execute commands as root, potentially escalating privileges. (ii) **SUID (Set User ID)**: Files with the SUID bit set run with the privileges of the file owner (often root). An attacker exploiting these can gain root privileges. (iii) **Sticky Bits and World Writable Files**: Files or directories with improper permissions (e.g., world-writable) may be vulnerable to exploitation. (iv) **Kernel Exploits**: Attackers might exploit vulnerabilities in the kernel to elevate their privileges. (v) **Weak or Stale Passwords**: Attackers can escalate privileges by cracking weak or forgotten passwords for privileged accounts. (vi) **Vulnerabilities in Installed Software**: Exploiting known vulnerabilities in installed software (e.g., outdated versions of services) to gain higher access.

2. Understanding and Mitigating Risks Associated with Sudo, SUID, and Sticky Bits

Sudo Risks: - Misconfigured `sudoers` file: A user may be allowed to run certain commands as root without proper restrictions.

Mitigation: Ensure only trusted users have sudo access. Limit commands they can run. For example, avoid the following configuration:

```
user ALL=(ALL) NOPASSWD: ALL
```

SUID Risks: Files with the SUID bit set can be exploited to gain elevated privileges.

Mitigation: Regularly audit files with the SUID bit set:

```
find / -type f -perm -4000 -exec ls -l {} \;
```

Remove the SUID bit from unnecessary files:

```
chmod u-s /path/to/file
```

Sticky Bit Risks: Improper use of sticky bits or world-writable directories can be exploited.
Mitigation: Audit and secure directories like `/tmp`:

```
chmod 1777 /tmp
```

3. Introduction to LinPeas for Privilege Escalation Checks

LinPeas automates the process of checking for privilege escalation vectors. Key features include scanning for SUID files, misconfigured sudoers, kernel vulnerabilities, and outdated packages.

Execution: Download and run LinPeas:

```
wget https://github.com/carlospolop/PEASS-ng/releases/download/v2020.08.04/linpeas.sh
chmod +x linpeas.sh
./linpeas.sh
```

Lab Exercises

Exercise 1: Use LinPeas to Identify Potential Privilege Escalation Vectors (i) Download LinPeas:

```
wget https://github.com/carlospolop/PEASS-ng/releases/download/v2020.08.04/linpeas.sh
chmod +x linpeas.sh
./linpeas.sh
```

(ii) Review LinPeas output for risks such as misconfigured sudo permissions and SUID binaries.

Outcome

By the end of this session, participants will learn to: (i) Identify common privilege escalation vectors using tools like LinPeas. (ii) Secure the system by fixing vulnerabilities. (iii) Apply best practices to mitigate privilege escalation risks.

(ii) Review LinPeas output for risks such as misconfigured sudo permissions and SUID binaries.

Exercise 2: Fix Discovered Privilege Escalation Risks (i) Fix sudo misconfigurations:

```
user1 ALL=(ALL) NOPASSWD: /bin/ls, /bin/cat
```

(ii) Fix SUID vulnerabilities:

```
chmod u-s /path/to/file
```

(iii) Fix writable files and directories:

```
chmod 644 /etc/passwd
chmod 600 /etc/shadow
chmod 1777 /tmp
```

Exercise 3: Analyze SUID and Sticky Bits for Security Vulnerabilities (i) List SUID binaries:

```
find / -type f -perm -4000 -exec ls -l {} \;
```

(ii) Audit sticky bits:

```
find / -type d -perm -1000 -exec ls -ld {} \;
```