Operating System Security

Day 2: Linux File System, Users & Permissions

Linux File System Basics

- **Everything is a File:**
  - In Linux, hardware devices, directories, and documents are all treated as files.

Attackers often target specific system directories like `/etc` or `/var` to gain control, modify configurations, or steal sensitive data. Protecting these areas is paramount.

Core Linux Directories

1

`/` – Root Directory

The absolute top-level directory of the Linux filesystem hierarchy. All other directories and files are contained within it.

Example: `ls /` (lists contents of root), `cd /` (navigates to root).

Purpose: Serves as the starting point for all paths and contains essential system components.

2

`/home` – User Directories

Stores personal directories for each regular user account. These directories typically contain user-specific documents, downloads, and configuration files.

Example: `cd /home/username` (go to user's home), `mkdir /home/username/test` (create a folder).

Purpose: Provides a dedicated space for users to manage their personal files without impacting system stability.

3

`/etc` – Configuration Files

Contains system-wide configuration files for applications and services. These files are usually read during system startup or when a service is initiated.

Example: `cat /etc/passwd` (view user info), `nano /etc/hosts` (edit hosts file).

Purpose: Manages system settings, network configurations, user accounts, and service parameters.

4

`/var` – Logs and Variable Data

Houses variable data files, such as system logs, mail queues, and temporary files that change frequently during system operation.

Example: `ls /var/log` (view log files), `tail -f /var/log/syslog` (monitor live logs).

Purpose: Monitoring system activity, debugging issues, and tracking changes over time.

5

`/bin` – Essential Binaries

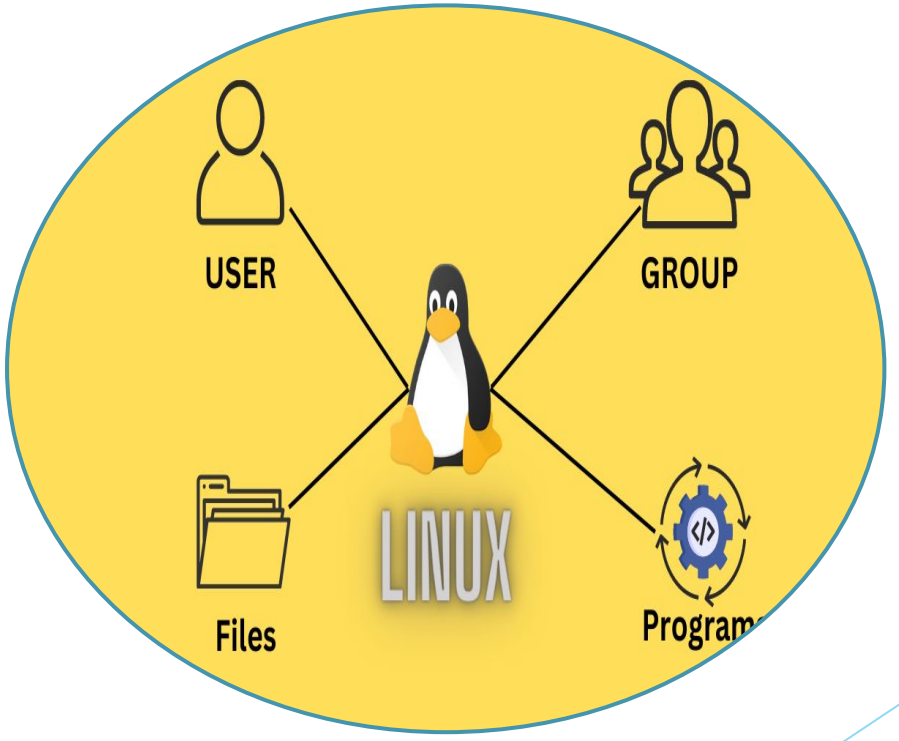
Stores essential command-line programs and utilities that are required for all users and for basic system operations and booting.

Example: `ls /bin` (list commands), `which ls` (shows path of 'ls').

Purpose: Provides core commands for basic file manipulation, navigation, and system interaction.

Users & Groups - Controlling Access

- Linux is a multi-user operating system
  - achieved through user accounts and groups
- A **user account**
  - individual or a service that interacts with the system. Each user has a unique ID (UID) and a home directory.
  - **Groups** simplify permissions by allowing multiple users to share common access rights to files and resources.
- **Root User:** The superuser with unrestricted access to the entire system. Extreme caution is advised when operating as root.
- **Normal Users:** Standard accounts with limited privileges, designed for everyday tasks.



Always adhere to the **Principle of Least Privilege (PoLP)**: grant users only the minimum permissions necessary to perform their tasks.

Managing Users and Groups: Essential Commands

- Effective user and group management involves a set of powerful commands that allow administrators to create, modify, and delete accounts, as well as assign users to specific groups.
- Limiting administrative access is a critical security practice.

`adduser`

Creates a new user account on the system.

`usermod`

Modifies an existing user account's properties.

`groups`

Displays the groups a user belongs to.

`addgroup`

Creates a new group.

Restricting admin/root access to only when absolutely necessary significantly reduces the attack surface and potential damage from accidental errors or malicious activity.

Linux User, Group & Permission Management Command

- Create a New User
  - sudo adduser username
    - Creates a new user with the name username
    - Prompts to set a password and optional user info
    - Automatically creates a home directory /home/username
- Example:
  - sudo adduser alice
    - Creates user alice and /home/alice

Linux User, Group & Permission Management Commands

- Create a New Group
  - sudo groupadd groupname
    - Creates a new group called groupname
    - Groups are used to assign permissions to multiple users
- Example:
  - sudo groupadd developers
    - Creates a group called developers

Linux User, Group & Permission Management Commands

- Assign a User to a Group
  - sudo usermod -aG groupname username
    - Adds an existing user to a group without removing them from other groups
    - -aG = append the group
- Example:
  - sudo usermod -aG developers alice
    - Adds user alice to the developers group

Linux User, Group & Permission Management Commands

- Assign Permissions to Files & Directories
  - Check Current Permissions:
    - ls -l filename
      - Displays permissions in rwx format (owner, group, others)
  - Change Permissions:
    - sudo chmod 755 filename
      - Change Ownership

- **7** = read, write, execute (owner)
- **5** = read, execute (group)
- **5** = read, execute (others)

  - Change Ownership:
    - sudo chown username:groupname filename
      - Changes owner and group of a file or directory
  - Examples:
    - sudo chmod 644 report.txt
    - sudo chown alice:developers report.txt
      - 644 → Owner can read/write, group/others can read
      - File owned by alice in group developers

Linux User, Group & Permission Management Commands

- Test Access as Different Users
- Switch User
  - su - username
    - Switch to another user to test permissions
- Attempt File Access
  - cat filename
    - Check if user can read file
    - Attempt writing to test write permission

Linux User, Group & Permission Management Commands

- Safe User Management Practices
  - Avoid giving **root** access unless necessary
  - Use **groups** to control permissions instead of giving individual file access
  - Regularly review users and groups
- Remove inactive users:
  - sudo deluser username
  - sudo delgroup groupname

File and Directory Permissions

- **Permission Types**
  - **Read (r):** Allows viewing the content of a file or listing the contents of a directory.
  - **Write (w):** Permits modifying a file or creating/deleting files within a directory.
  - **Execute (x):** Enables running a file as a program or entering a directory.
- **Ownership Categories**
  - **Owner:** The user who owns the file or directory.
  - **Group:** The group associated with the file or directory.
  - **Others:** All other users on the system.

Understanding Numeric Permissions

- Permissions can be represented numerically, which is a common and efficient way to set permissions using commands like chmod. Each permission type (read, write, execute) is assigned a numerical value, and these values are summed for each ownership category.
- Linux Numeric Permissions Table

Permission	Symbol	Numeric Value
Read	r	4
Write	w	2
Execute	x	1

How Numeric Permissions Work

- Each file/directory has **three categories**:
  1. Owner
  2. Group
  3. Others
- Sum the values of allowed permissions for each category
- Examples:
  - 7 = 4 + 2 + 1 = rwx → full permissions
  - 6 = 4 + 2 = rw- → read & write
  - 5 = 4 + 1 = r-x → read & execute
  - 4 = 4 = r-- → read only
  - 0 = --- → no permission

Numeric Permission Examples

Numeric	Symbolic	Meaning
777	rwxrwxrwx	Owner, group, others all can read/write/execute
755	rwxr-xr-x	Owner full access, group/others can read & execute
700	rwx------	Owner full access, group/others no access
644	rw-r--r--	Owner can read/write, group/others read only
600	rw-------	Owner can read/write, group/others no access
555	r-xr-xr-x	Everyone can read & execute, no one can write