Linux: Security and permissions

Security and permissions in Linux are fundamental aspects of system administration that ensure the confidentiality, integrity, and availability of system resources. Linux provides a robust set of tools and mechanisms to manage access control, enforce security policies, and protect against unauthorized access.

Below is a detailed explanation of the key sections:

---

1. Setting Advanced Permissions

Linux uses a combination of traditional file permissions and advanced mechanisms like ACLs, SUID, and SGID to provide fine-grained access control.

a. ACLs (Access Control Lists):

ACLs extend the standard file permissions (read, write, execute) to allow more specific access control for users and groups.

Commands:

- ☐ **`setfacl`:** Sets ACLs for files or directories.

  `setfacl -m u:username:rwx file.txt` (grants read, write, and execute permissions to a specific user).

- ☐ **`getfacl`**: Displays ACLs for a file or directory.

  `getfacl file.txt`.

Use Case: Useful when you need to grant permissions to multiple users or groups beyond the standard owner/group/others.

b. SUID (Set User ID):

What it is: When the SUID bit is set on an executable file, the file runs with the permissions of the file owner, not the user executing it.

How to Set:

- `chmod u+s file` (sets SUID bit).
- Example: `chmod u+s /usr/bin/passwd` (allows users to change their passwords by running the `passwd` command with root privileges).

Security Consideration: Use SUID sparingly, as it can introduce security risks if misconfigured.

c. SGID (Set Group ID):

What it is: When the SGID bit is set on a directory, files created within the directory inherit the group ownership of the directory, not the user's primary group.

How to Set:

- `chmod g+s directory` (sets SGID bit).
- Example: `chmod g+s /shared_directory` (ensures all files in `/shared_directory` belong to the same group).

Use Case: Useful for collaborative environments where multiple users need to share files.

---

2. Using Firewalls

Firewalls are essential for controlling incoming and outgoing network traffic. Linux provides tools like `ufw` and `iptables` to configure firewall rules.

a. `ufw` (Uncomplicated Firewall):

What it is: A user-friendly interface for managing `iptables`.

Commands:

- Enable/disable: `sudo ufw enable`, `sudo ufw disable`.
- Allow/deny traffic: `sudo ufw allow 22/tcp` (allow SSH), `sudo ufw deny 80/tcp` (block HTTP).
- Check status: `sudo ufw status`.

Use Case: Ideal for beginners or simple firewall configurations.

b. `iptables`:

What it is: A powerful command-line tool for configuring firewall rules.

Commands:

- Allow traffic: `sudo iptables -A INPUT -p tcp --dport 22 -j ACCEPT` (allow SSH).
- Block traffic: `sudo iptables -A INPUT -p tcp --dport 80 -j DROP` (block HTTP).
- List rules: `sudo iptables -L`.

Use Case: Suitable for advanced users who need granular control over network traffic.

---

3. Basics of SELinux and AppArmor

SELinux and AppArmor are Linux security modules that provide mandatory access control (MAC) to enforce security policies.

a. SELinux (Security-Enhanced Linux):

What it is: A security module developed by the NSA that enforces access control based on policies.

Key Concepts:

- **Modes:** Enforcing (policies are enforced), Permissive (policies are logged but not enforced), Disabled.
- **Contexts:** Files, processes, and users are assigned security contexts (e.g., `user_u:role_r:type_t`).

Commands:

- Check status: `sestatus`.
- Change mode: `sudo setenforce Enforcing` or `sudo setenforce Permissive`.
- View context: `ls -Z` (files), `ps -Z` (processes).

Use Case: Commonly used in enterprise environments for enhanced security.

b. AppArmor:

What it is: A security module that confines programs to a limited set of resources using profiles.

Key Concepts:

- **Profiles:** Each application has a profile defining its allowed actions and resources.
- **Modes:** Enforce (restricts access), Complain (logs violations but does not restrict).

Commands:

- Check status: `sudo apparmor_status`.
- Load/unload profiles: `sudo apparmor_parser -r /etc/apparmor.d/profile_name`.

Use Case: Easier to configure than SELinux and often used in Ubuntu and other distributions.

Summary

- **Advanced Permissions:** Use ACLs, SUID, and SGID for fine-grained access control.
- **Firewalls:** Use `ufw` for simplicity or `iptables` for advanced configurations.
- **SELinux and AppArmor:** Implement mandatory access control to enforce security policies.