Kernel Security Mechanisms & Isolation Technique

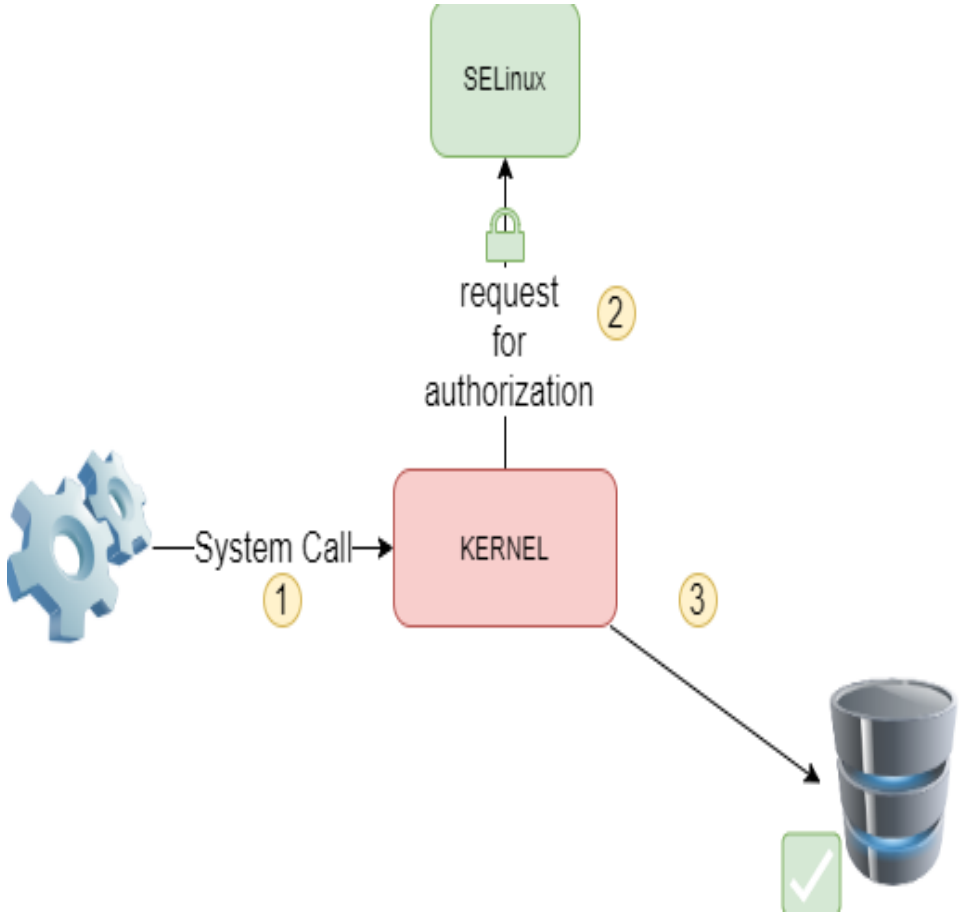
Ayush Chaudhary

M.E. Computer

SELinux

Introduction

- SELinux (Security-Enhanced Linux) is a security feature that:
  - Controls which programs can access files and resources.
  - Prevents unauthorized users or software from making changes.
  - Was originally developed by the NSA (National Security Agency).
- SELinux is a Mandatory Access Control system which is :
  - Strict Security System → Controls who can access what.
  - Different from Traditional Linux → Not based on normal file permissions (read, write, execute).
  - Root User Restrictions → Even root must follow security rules.
  - Prevents Unauthorized Access → Hackers can't bypass policies, even with root access.
  - Enhances Security → Reduces risk of data leaks and unauthorized changes.
- With each system call, the kernel queries SELinux to see if it allows the action to be performed.

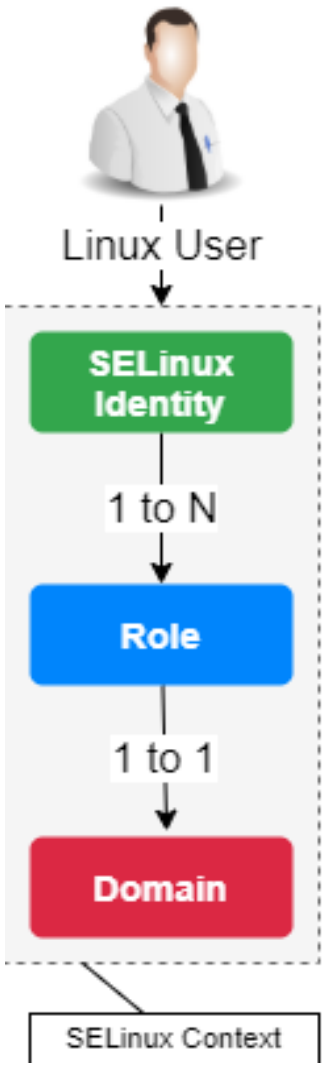


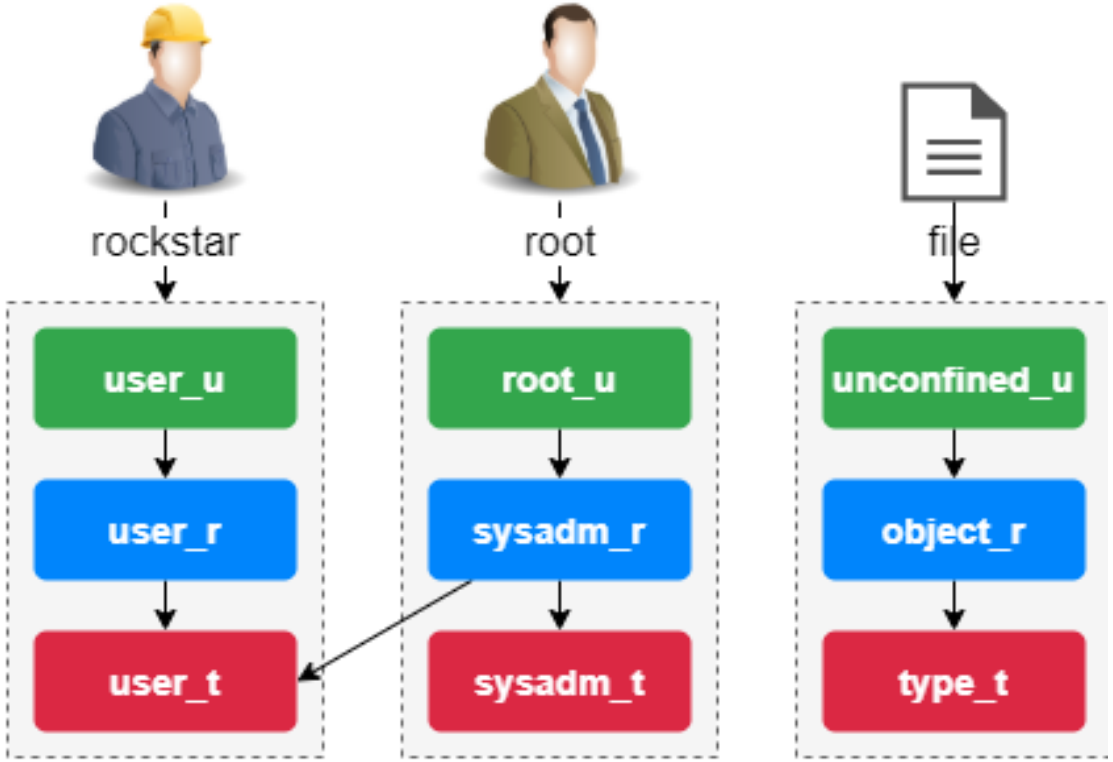
How SELinux work?

- **Defining security policies**: These policies dictate how processes can interact with each other and with system resources (files, devices, etc.).
- **Labeling**: Every file, process, and resource in SELinux has an associated security context, which includes information about the identity and the allowed actions. This context includes user, role, type, and level.
- **Access control**: When a process tries to access a resource, SELinux checks its security context against the resource's security context to determine if the action is allowed.

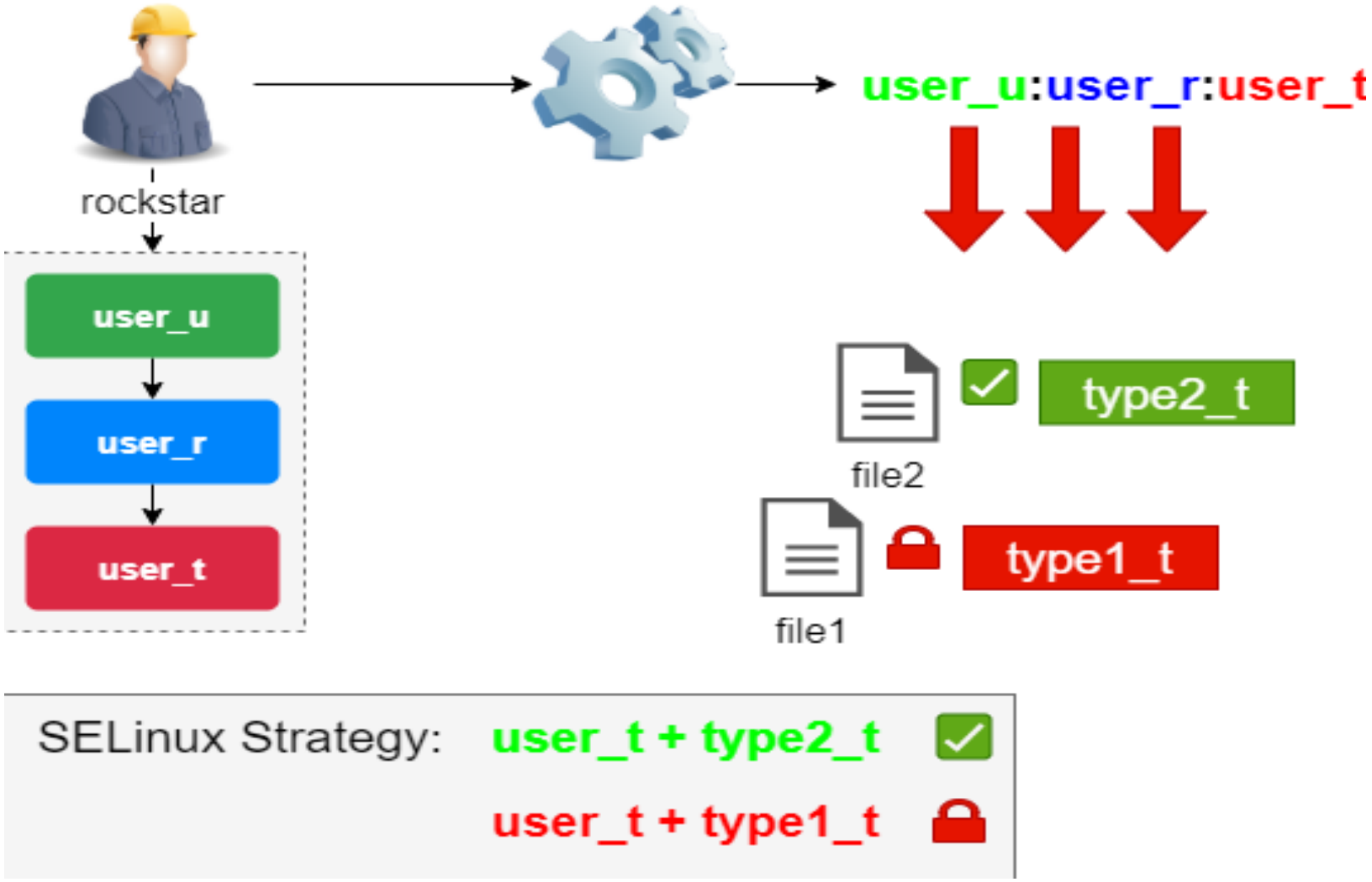
The SELinux context

- The SELinux security context is defined by the trio identity+role+domain.
- The terms "domain" and "type" are similar. Typically "domain" refers to a process, while "type" refers to an object.
- The naming convention is: user_u:role_r:type_t.





How SELinux controls access to files based on user roles and file types



SELinux Modes

- SELinux can run in three different modes:

Mode	Description
Enforcing	SELinux is fully active and blocks unauthorized actions.
Permissive	SELinux does not block but only logs actions for monitoring.
Disabled	SELinux is turned off completely (not recommended).

Scenario: A Web Server is Running

- You have a **web server** on your Linux system, hosting a website at **/var/www/html/**.
- **Problem:**
  What if a hacker uploads a **malicious script** that tries to access **/etc/passwd** (a critical system file that stores user information)?
- **How SELinux Protects the System**
  - **Step 1**: File and Process Labels
    - SELinux labels each file and process to determine who can access what.
      - The web server (Apache, Nginx) runs as httpd_t (type label).
      - The website files are labeled httpd_sys_content_t.
      - The password file is labeled etc_t.
  - **Step 2**: Access Rules
    - SELinux has rules that say:
      - A web server can read files labeled httpd_sys_content_t.
      - A web server CANNOT read files labeled etc_t.
- So, even if a hacker uploads a script that tries to read /etc/passwd, SELinux will block it because:
  - httpd_t → CANNOT access → etc_t

Pros and Cons of SELinux

- **Pros:**
  Strong security against hackers and malware.
  Prevents unauthorized file access.
  Protects system processes from being misused.
- **Cons:**
  Can be complex for beginners.
  Some applications may not work without proper configuration.

AppArmor

Introduction

- AppArmor is a Linux Security Module (LSM) that restricts what programs can access on your system.
- It works by creating profiles for applications. Each profile defines what files, directories, and resources the application can access.
- For example, if you have a web browser, AppArmor can limit it to only accessing your downloads folder and not your entire file system.

How it works?

- **AppArmor checks every action**: Whenever an application tries to do something, like opening a file or accessing a network, the Linux kernel (the core of the operating system) asks AppArmor if this action is allowed.
- **AppArmor has rules (called "profiles")**: Each program has its own set of rules called a profile. The profile tells AppArmor what the application is allowed to do.
  - For example, a profile might say, "This program can read files in the /home/user/documents folder but cannot touch files in /etc.
- **Profiles specify what actions are allowed**: The profile describes actions such as:
  - Read: Can the program read a file?
  - Write: Can it change a file?
  - Execute: Can it run a program?

SELinux vs AppArmor

Feature	SELinux	AppArmor
Security Model	Label-based (fine-grained)	Path-based (simple)
Configuration	Complex, requires policy modules	Simple, based on file paths
Profiles	Based on security labels and contexts	Based on file paths and directories
System Support	Red Hat-based (RHEL, CentOS, Fedora)	Debian-based (Ubuntu, SUSE)
Best Use Case	Enterprise, government, high-security environments	General-purpose security, Ubuntu systems

Isolation Techniques: Namespaces, Cgroups, and Sandboxing

Introduction

- Isolation techniques are used in modern operating systems to make sure that different processes (or programs) running on the system don't interfere with each other.
- These techniques ensure that applications run securely, independently, and don’t mess with each other’s data or system resources.
- The three most important isolation techniques:
  - Namespaces
  - Cgroups
  - Sandboxing.

Namespaces

Namespaces

- Namespaces are a way to create isolated environments for processes.
- In simple terms, they allow you to make it look like a program is running on its own, with its own set of resources, even though it's actually sharing the same system with other programs.
- They make sure that different programs or users can’t see or affect each other’s resources (like file systems, network devices, etc.).
- **How it works:** Namespaces give each process its own version of certain system resources (like network, user IDs, files, etc.). This means one process doesn't see or affect the resources of another process.

Type of namespace

- **PID Namespace**: Keeps processes isolated from each other. Programs inside one PID namespace can't see or interact with processes in another PID namespace.
- **Mount Namespace**: Each namespace can have its own view of the file system. For example, one program might only see the /home directory, while another sees /root as its root directory.
- **Network Namespace**: Lets processes have separate networking stacks. This means they can have their own IP addresses, ports, and even network interfaces.
- **User Namespace**: Allows different programs to have their own set of user IDs (UIDs), so one program can think it's running as the root user, while another thinks it's running as a normal user.

Cgroups

Introduction

- Cgroups (short for **Control Groups**) are a way to limit, prioritize, and monitor how much CPU, memory, disk I/O, and other resources a process (or group of processes) can use.
- Cgroups don’t isolate processes from each other like namespaces do, but they control how much resource each process can consume.
- **How it works**: Cgroups allow you to define resource limits for processes, such as:
    - CPU usage: Limit how much CPU power a process can use.
    - Memory usage: Set a maximum memory limit for a process.
    - Disk I/O: Control how much disk access a process can use.
- For example: If you have a web server and a database running on the same machine, you can use cgroups to ensure the web server doesn't consume all the CPU or memory, leaving the database to struggle.

Sandboxing

Introduction

- Sandboxing is a technique where you isolate a program in a restricted environment to limit its access to the rest of the system.
- This is often used to run untrusted or experimental code safely, like in web browsers, mobile apps, or any program that interacts with potentially harmful content.
- Think of a sandbox as a secure "box" where the application can run, but it can’t escape to cause harm to the rest of your system or access sensitive data.
- **How it works**:
  - A program inside a sandbox has restricted access to system resources, files, network, and hardware. It is only allowed to interact with specific resources that are explicitly granted to it.
  - Sandboxes typically use both namespaces and cgroups to ensure that the program is properly isolated and has limited access to the system.

Example

- A web browser runs websites in a sandbox so that if the website contains malware or harmful code, it can't affect the rest of your computer.
- A mobile app may be sandboxed to prevent it from accessing sensitive data, like your photos or contacts, unless it has the right permissions.
- **Summary of Isolation Techniques:**

Technique	What It Does	Key Benefit
Namespaces	Isolates processes so they act like they are in separate environments (e.g., network, user, PID).	Keeps processes separate and secure.
Cgroups	Limits and monitors resource usage (CPU, memory, disk) of processes.	Prevents resource hogging, ensuring fair use.
Sandboxing	Restricts what a program can do, isolating it from the rest of the system.	Enhances security by limiting access to sensitive resources.