Topic of Presentation

Security and Isolation in Operating System

Presented By:

BP BHATTA (242551)

MSC, CS

Contents

5.1. Kernel Security Mechanisms: SELinux, AppArmor.

5.5. Case Studies: Linux Security Module, Android security Architecture, macOS Security Architecture.

Kernel security mechanisms

- Kernel security mechanisms are critical components of an operating system (OS) that ensure the integrity, confidentiality, and availability of system resources.
- The kernel, being the core of the OS, has unrestricted access to hardware and manages system operations, making it a prime target for attacks.

Key kernel security mechanisms:

1. Kernel Hardening:

- Kernel hardening involves modifying the kernel's default behavior to reduce its attack surface and make it more resilient to attacks.

Methods:

- **Restricting access to kernel logs and pointers:** Limiting who can access sensitive information and preventing unauthorized manipulation of kernel memory.
- **Kernel Lockdown:** Prevents unauthorized modifications to the kernel image and access to security and cryptographic data in kernel memory.
- **Seccomp (Secure Computing Mode):** Restricts the system calls a process can make, reducing its attack surface and limiting the potential damage from malicious code.
- **BPF hardening:** BPF (Berkeley Packet Filter) is a system used to load and execute bytecode within the kernel dynamically during runtime and is used in a number of Linux kernel subsystems.
- **Kernel Control Flow Integrity (KCFI):** Hardens the kernel against attacks that modify kernel control flow.

2. Security Patches:

- Regularly applying kernel security patches is crucial for mitigating known vulnerabilities and preventing malicious exploitation.
- The Linux kernel is frequently patched to address newly discovered security flaws, making it vital to stay up-to-date with the latest updates.

3. Access Control:

- Kernel-level access control mechanisms ensure that only authorized entities can access critical system components.

Methods:

- **Mandatory Access Control (MAC) systems:** SELinux (Security-Enhanced Linux) is a prominent example that enforces access control policies.
- **Role-Based Access Control (RBAC)**: Grants permissions based on user roles.
- **Discretionary Access Control (DAC)**: Allows resource owners to set access permissions (e.g., file permissions in Unix-like systems).

4. **Kernel Self-Protection:**

- Protects the kernel itself from vulnerabilities and attacks, including preventing privilege escalation.

Methods:

- **Attack surface reduction:** Minimizing the number of potential attack vectors.
- **Memory integrity:** Ensuring the integrity of kernel memory and preventing corruption.
- **Probabilistic defences:** Employing techniques that make it harder for attackers to predict and exploit vulnerabilities.
- **Information exposure prevention:** Preventing sensitive information from being leaked.

SELinux (Security-Enhanced Linux)

- SELinux is a special security system built into Linux computers.
- It helps keep your computer safe and secure.
- With SELinux, different programs and users on the computer have limited permissions. This means each program or user can only access certain files and do certain actions that they are allowed to do.
- SELinux is a special security system built into Linux computers.
- The main purpose of the SELinux is to control what different programs and users are allowed to access on the computer.
- It does this by setting the strict rules.
- Without SELinux, if a program gets a virus or hacker access, that program could access all files and data on the computer. This is bad. With SELinux, each program and user is limited in what they can see and do.
- For example, a web browser can connect to websites but cannot read your private documents. This prevents viruses and hackers from causing full damage if they get into one program.

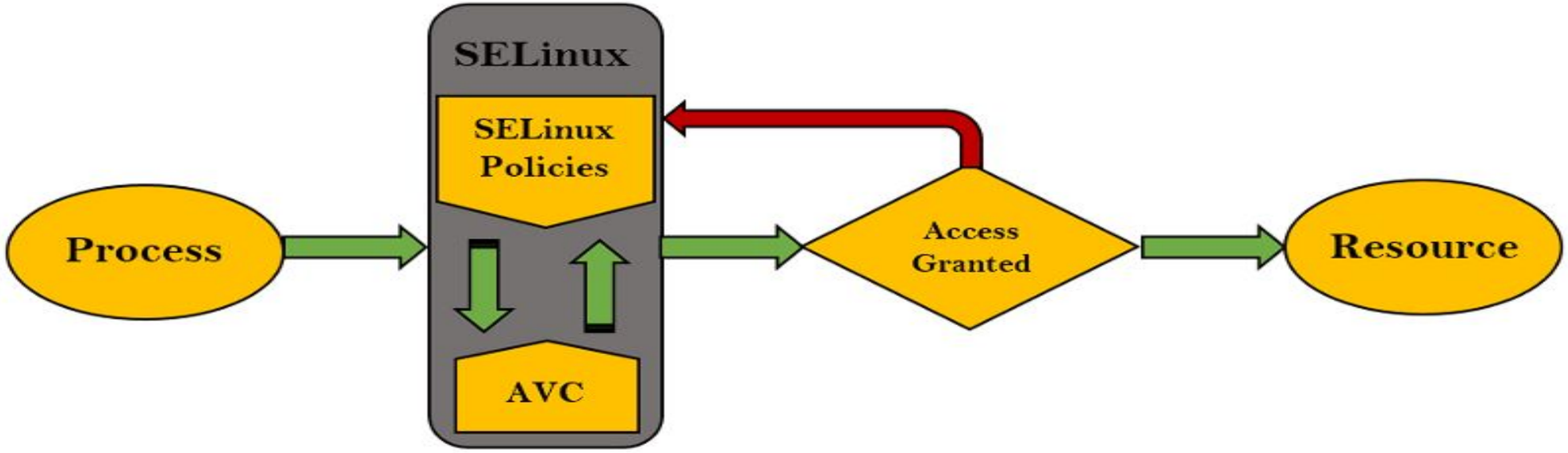
How Security-Enhanced Linux Works?

- SELinux works by implementing mandatory access controls (MAC).
- With MAC, sysadmins define which users and processes have access to specific resources rather than relying on less secure broadly-defined permissions.
- To accomplish this, SELinux uses security policies.

SELinux Policies

- SELinux blocks all applications and users by default, allowing access only to those specified in the security policies.
- Security policies are a set of simple rules that tell SELinux who is allowed to access which parts of the system.
- These rules set the permissions for each user, program, and resource.
- SELinux keeps track of every decision (allow or block access) in the Access Vector Cache (AVC).
- This makes checking permissions faster.

- When a program tries to access something, SELinux first checks AVC to see if a decision has already been made. If so, it follows that decision quickly.
- If not, SELinux looks at the policy rules and makes a new decision.



- One key feature is that SELinux can give different permissions to different programs.
- For example, a web server program might be allowed to read and write files, but other programs cannot. SELinux can also check conditions before allowing access. Maybe a web server can only read/write if the request comes from a trusted address.

SELinux Labels and Type Enforcement

- SELinux uses labels with the policy rules to decide what actions to allow for each resource.
- Admins assign labels to every process, network port, file, etc.

Labels include:

- User - The Linux user mapped to a SELinux user
- Role - The user's authorized role for that system
- Type - This determines the permissions that are enforced
- Level (optional) - A security clearance level

Label format :

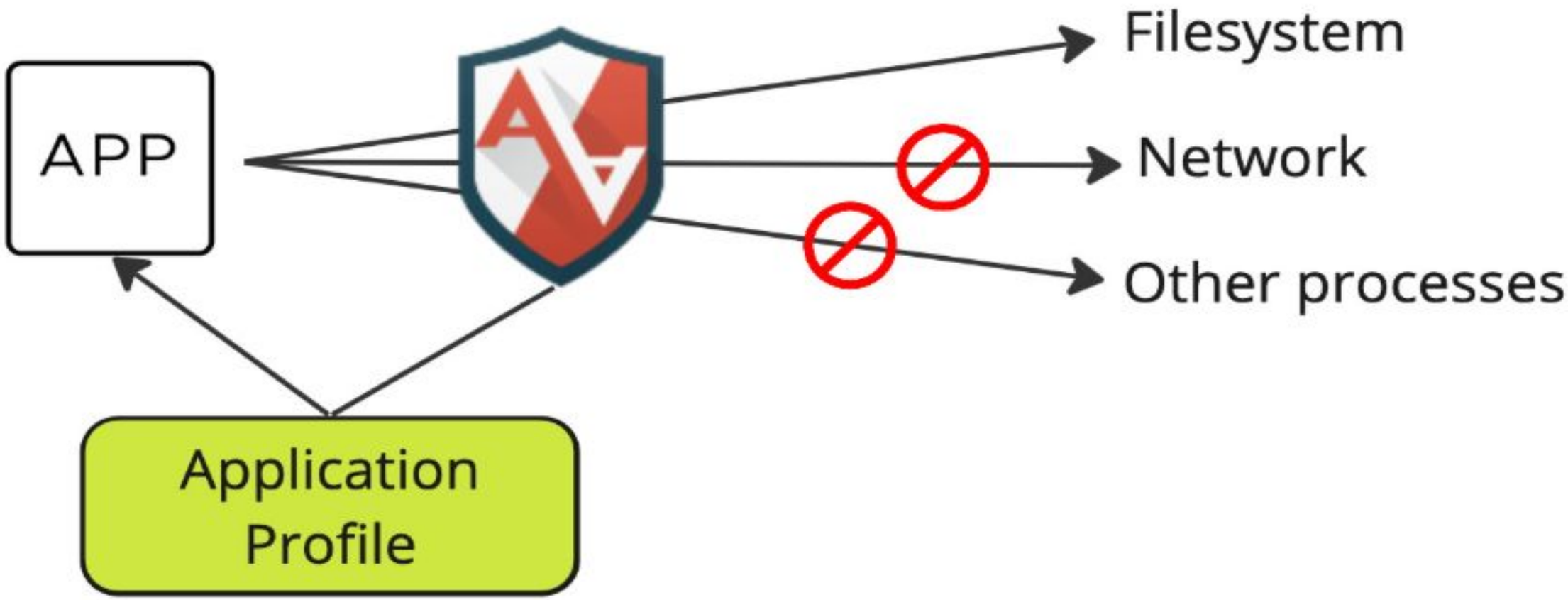
user:role:type:level

SELinux Modes

1. **Enforcing mode :** This is default and most secure. SELinux actively enforces the policy rules, denying any unauthorized access attempts. Blocked attempts are logged.
2. **Permissive mode :** Less secure but still monitors access. SELinux just logs what would be blocked by policies, but doesn't actually block it. Useful for testing.
3. **Disabled mode :** SELinux is completely turned off removing all the access protection. This mode is Only for the troubleshooting.

AppArmor

- AppArmor is a Linux kernel security module that restricts the capabilities of programs by applying security profiles, allowing administrators to control what applications can access and do.
- AppArmor is an easy-to-use Linux Security Module implementation that restricts applications' capabilities and permissions with **profiles** that are set per-program.
- It provides mandatory access control (MAC) to supplement the more traditional UNIX model of discretionary access control (DAC).
- AppArmor is a security module included in the Linux kernel since version 2.6.36.
- It allows or forbids access to system resources, such as network, filesystem, or other processes, based on application profiles.
- This prevents application flaws from being exploited.


APP
Filesystem
Network
Other processes
Application Profile

AppArmor Profiles

- AppArmor profiles are simple text files that define mandatory access control (MAC) policies for our applications and are stored under **/etc/apparmor.d/**.
- We can control access to the following system resources, capabilities, and processes:
- Files
- Linux capabilities
- Network
- Mount, remount, and umount
- pivot_root
- ptrace
- signal
- DBus
- Unix domain sockets

- AppArmor profiles can be put in three different modes:
- **Enforce** = Process cannot escape
- **Complain** = Process can escape but will be logged
- **Unconfined** = Process can escape

An empty profile looks like the following example:

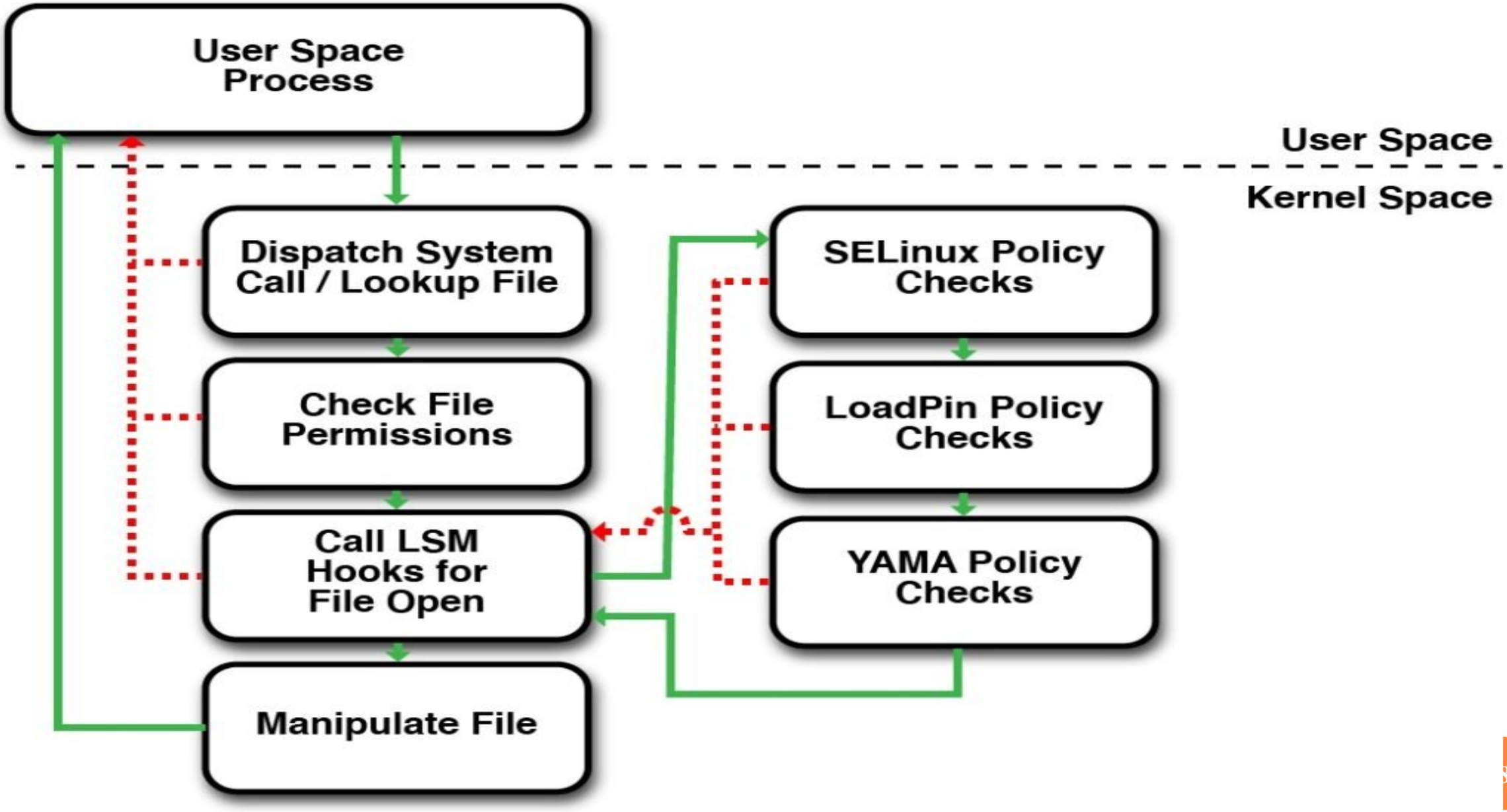
```
# AppArmor policy for appxyz
# ###Author and copyright###
# ###Comments###
#include <tunables/global>
# No template variables specified
"/usr/bin/appxyz" {
#include <abstractions/base>
# No abstractions specified

# No policy groups specified
```

AppArmor vs. SELinux

- AppArmor and SELinux both protect the systems from the bad software and the unauthorized access but they work differently.
- AppArmor uses rule files for each program to allow or block actions. SELinux uses policies that admins make to give exact permissions for users, programs, and resources.
- AppArmor is mainly used on SUSE and Ubuntu Linux distributions.
- AppArmor is easier to setup and manage, but has less control. SELinux is more complex but gives admins very detailed control over permissions.

Linux Security Module

- Linux Security Modules (LSM) is a framework that provides a mechanism for various security models to be implemented as loadable kernel modules in the Linux kernel.
- The LSM framework allows these modules to mediate access to internal kernel objects, such as files, tasks, and network sockets, thereby enforcing security policies.
- The Linux security module (LSM) framework, which allows for security extensions to be plugged into the kernel, has been used to implement MAC on Linux.
- LSM hooks in Linux Kernel mediates access to internal kernel objects such as inodes, tasks, files, devices, and IPC.
- LSMs, in general, refer to these generic hooks added in the core kernel code.
- Further, security modules could make use of these generic hooks to implement enhanced access control as independent kernel modules.
- AppArmor, SELinux, Smack, TOMOYO are examples of such independent kernel security modules.


User Space Process
User Space
Kernel Space
Dispatch System Call / Lookup File
Check File Permissions
Call LSM Hooks for File Open
Manipulate File
SELinux Policy Checks
LoadPin Policy Checks
YAMA Policy Checks

Key Features of LSM:

- **Modularity**: LSM allows different security models to be implemented as kernel modules, which can be loaded and unloaded dynamically.
- **Flexibility**: Multiple security modules can be loaded simultaneously, although typically only one will enforce access control.
- **Hooks**: LSM provides a set of hooks throughout the Linux kernel code. These hooks are placed at critical points where security decisions need to be made, such as file access, task creation, and network operations.

Popular LSM Implementations:

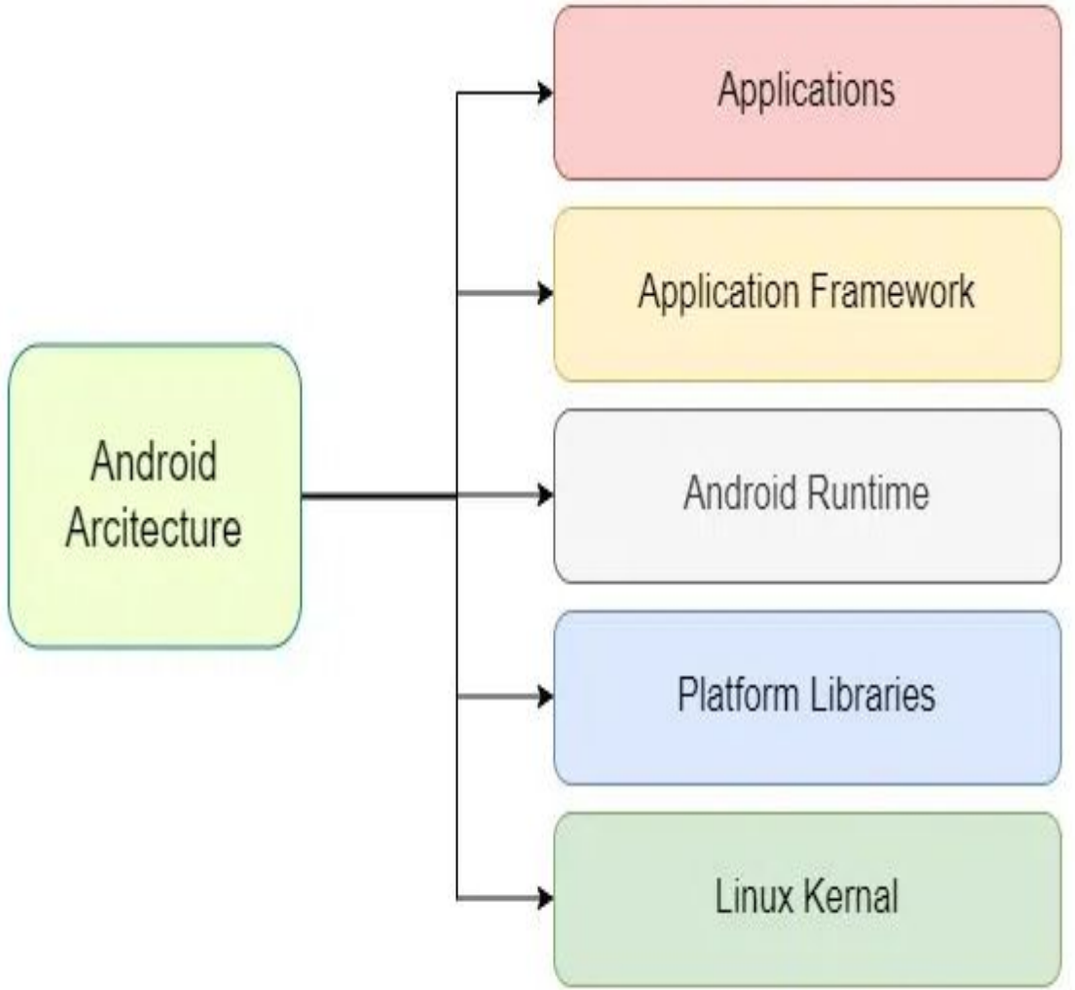
- **SELinux (Security-Enhanced Linux)**:
  - Developed by the NSA.
  - Implements Mandatory Access Control (MAC) using security contexts and policies.
  - Provides fine-grained control over processes and files.
- **AppArmor**:
  - Developed by Canonical.
  - Implements MAC using path-based profiles.
  - Easier to configure and manage compared to SELinux.
- **Smack (Simplified Mandatory Access Control Kernel)**:
  - Designed for simplicity and ease of use.
  - Suitable for embedded systems and IoT devices.
- **TOMOYO Linux**:
  - Focuses on behavior analysis and policy generation.
  - Uses pathname-based access control.
- **Yama**:
  - Focuses on restricting process operations, such as ptrace.
  - Provides additional security for user-space processes.

Android Architecture

- Android architecture contains a different number of components to support any Android device's needs.
- Android software contains an open-source Linux Kernel having a collection of a number of C/C++ libraries which are exposed through application framework services.
- Among all the components Linux Kernel provides the main functionality of operating system functions to smartphones and Dalvik Virtual Machine (DVM) provide a platform for running an Android application.

Components of Android Architecture

- The main components of Android architecture are the following:-
  - Applications
  - Application Framework
  - Android Runtime
  - Platform Libraries
  - Linux Kernel



1. Applications

- Applications is the top layer of android architecture.
- The pre-installed applications like home, contacts, camera, gallery etc and third party applications downloaded from the play store like chat applications, games etc. will be installed on this layer only.
- It runs within the Android run time with the help of the classes and services provided by the application framework.

2. Application framework

- Application Framework provides several important classes which are used to create an Android application.
- It provides a generic abstraction for hardware access and also helps in managing the user interface with application resources.
- Generally, it provides the services with the help of which we can create a particular class and make that class helpful for the Applications creation.
- It includes different types of services activity manager, notification manager, view system, package manager etc. which are helpful for the development of our application according to the prerequisite.

3. Application runtime

- Android Runtime environment is one of the most important part of Android.
- It contains components like core libraries and the Dalvik virtual machine(DVM).
- Mainly, it provides the base for the application framework and powers our application with the help of the core libraries.
- Like Java Virtual Machine (JVM), **Dalvik Virtual Machine (DVM)** is a register-based virtual machine and specially designed and optimized for android to ensure that a device can run multiple instances efficiently.
- It depends on the layer Linux kernel for threading and low-level memory management.
- The core libraries enable us to implement android applications using the standard JAVA or Kotlin programming languages.

4. Platform libraries

- The Platform Libraries includes various C/C++ core libraries and Java based libraries such as Media, Graphics, Surface Manager, OpenGL etc. to provide a support for android development.
- **Media** library provides support to play and record an audio and video formats.
- **Surface manager** responsible for managing access to the display subsystem.
- **SGL** and **OpenGL** both cross-language, cross-platform application program interface (API) are used for 2D and 3D computer graphics.
- **SQLite** provides database support and **FreeType** provides font support.
- **Web-Kit** This open source web browser engine provides all the functionality to display web content and to simplify page loading.
- **SSL (Secure Sockets Layer)** is security technology to establish an encrypted link between a web server and a web browser.

5. Linux Kernel

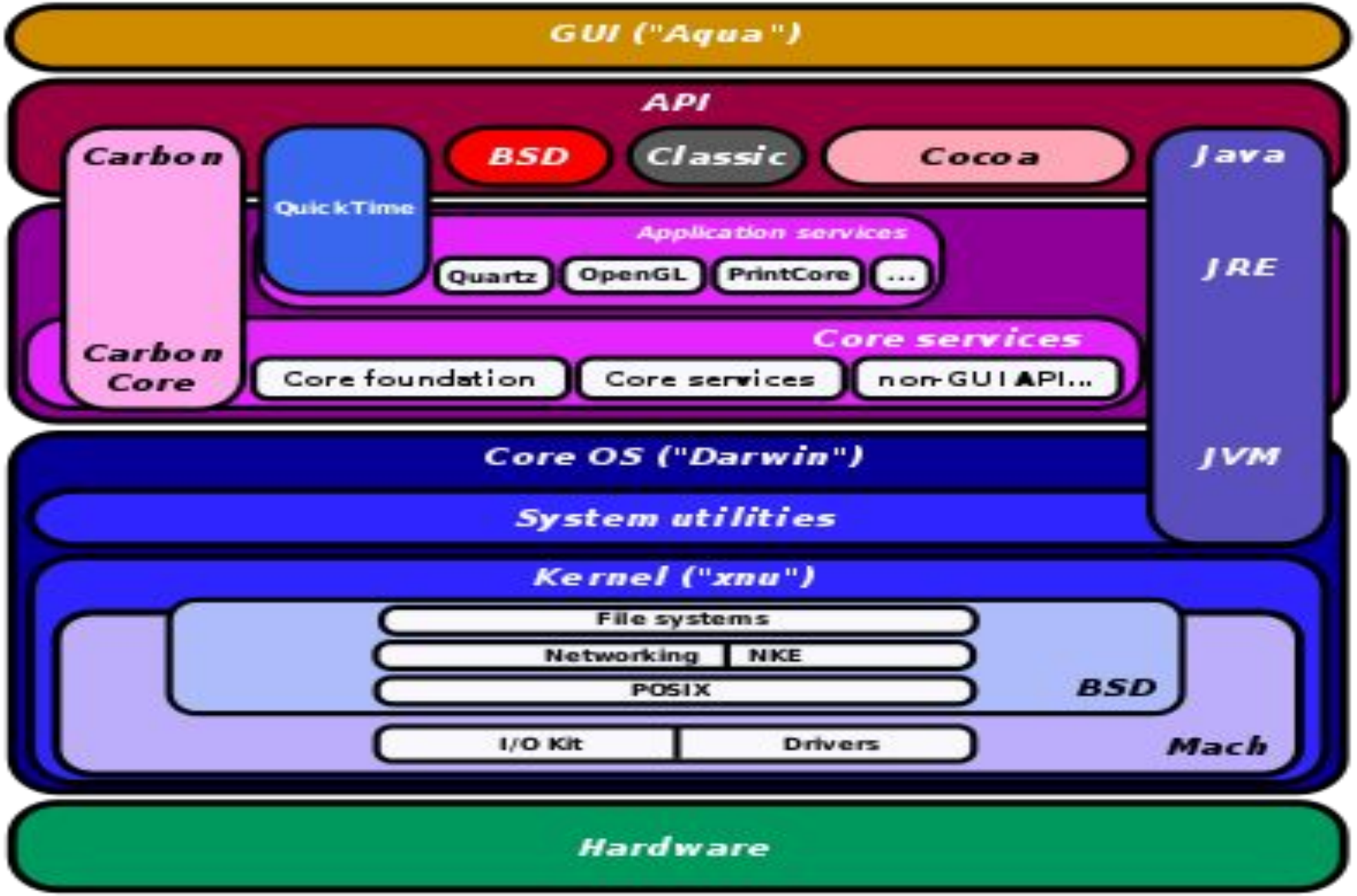
- Linux Kernel is heart of the android architecture.
- It manages all the available drivers such as display drivers, camera drivers, Bluetooth drivers, audio drivers, memory drivers, etc. which are required during the runtime.
- It is responsible for management of memory, power, devices etc.

The features of Linux kernel are:

- **Security:** The Linux kernel handles the security between the application and the system.
- **Memory Management:** It efficiently handles the memory management thereby providing the freedom to develop our apps.
- **Process Management:** It manages the process well, allocates resources to processes whenever they need them.
- **Network Stack:** It effectively handles the network communication.
- **Driver Model:** It ensures that the application works properly on the device and hardware manufacturers responsible for building their drivers into the Linux build.

macOS SECURITY ARCHITECTURE

- The macOS security architecture consists of three discreet levels.
- The bottom layer of the security stack contains the Berkeley Software Distribution (BSD) and Mach.
- BSD is an open source standard and is responsible for the basic file system and network services, and handles access control for users and groups.
- The middle layer of the macOS security architecture is CDSA. Although Apple created its version of CDSA, it stems from an open source standard.
- The main component of the CDSA layer is the Common Security Services Manager (CSSM).
- The top layer of the security architecture consists of the macOS Security Services. This layer contains a collection of macOS security APIs used by the applications.


GUI ("Aqua")
API
Carbon
QuickTime
BSD
Classic
Cocoa
Java
Application services
Quartz
OpenGL
PrintCore
...
JRE
Carbon Core
Core services
Core foundation
Core services
non-GUI API...
Core OS ("Darwin")
JVM
System utilities
Kernel ("xnu")
File systems
Networking
NKE
POSIX
BSD
I/O Kit
Drivers
Mach
Hardware

Below is an overview of the key components and features of macOS security architecture:

1. Hardware Security

- **Apple Silicon (M1, M2, etc.)**: Apple's custom-designed processors include dedicated security features such as the Secure Enclave, which is a coprocessor that provides an additional layer of security for sensitive data.
- **Secure Boot**: Ensures that only trusted software loads during the startup process. It verifies the integrity of the bootloader and kernel before execution.
- **Touch ID and Face ID**: Biometric authentication methods integrated into Apple's hardware for secure and convenient user authentication.

2. System Integrity Protection (SIP)

- **SIP**: Also known as "rootless," SIP restricts the root user account and limits the actions that even the root user can perform on protected parts of the system. This prevents malicious software from modifying critical system files and directories.
- **Protected Directories**: SIP protects directories such as /System, /usr, /bin, /sbin, and others from being modified, even by the root user.

3. File System Security

- **APFS (Apple File System)**: Designed with security in mind, APFS includes features like native encryption, snapshots, and cloning. It supports strong encryption for both data at rest and in transit.
- **FileVault 2**: Provides full-disk encryption using XTS-AES-128 encryption with a 256-bit key. FileVault ensures that data on the disk is encrypted and can only be accessed by an authenticated user.

4. Application Security

- **Gatekeeper**: Ensures that only trusted software runs on macOS. It checks the digital signature of apps and verifies that they are from identified developers or the Mac App Store.
- **Notarization**: Apple requires that all software distributed outside the Mac App Store be notarized. Notarization involves Apple scanning the software for malicious content and ensuring it meets security requirements.
- **Sandboxing**: Restricts apps to their own "sandbox," limiting their access to system resources and user data. This minimizes the potential damage from malicious or compromised apps.

5. User Authentication and Authorization

- **User Accounts**: macOS supports multiple user accounts with different privilege levels. Standard users have limited permissions, while administrators have more control over the system.
- **Authorization Services**: Manages user authentication and authorization for privileged operations. It ensures that only authorized users can perform sensitive tasks.
- **Keychain**: Securely stores passwords, encryption keys, certificates, and other sensitive information. The Keychain is encrypted and protected by the user's login password.

6. Network Security

- **Firewall**: Built-in firewall controls incoming network connections and can be configured to block unauthorized access.
- **VPN and Secure Protocols**: macOS supports various VPN protocols (e.g., IPsec, L2TP, IKEv2) and secure communication protocols (e.g., TLS) to protect data in transit.

7. Privacy Controls

- **Privacy Preferences**: macOS includes privacy settings that allow users to control which apps have access to sensitive data such as location, contacts, calendars, and photos.
- **Transparency, Consent, and Control (TCC)**: Requires apps to request user permission before accessing certain types of data, such as the microphone, camera, or files in specific directories.

8. Security Updates and Patch Management

- **Automatic Updates**: macOS regularly receives security updates and patches from Apple. Users can enable automatic updates to ensure their system is always protected against the latest threats.
- **XProtect**: Apple's built-in malware detection system that uses known malware signatures to detect and block malicious software.

9. Endpoint Security

- **Endpoint Security Framework**: Provides APIs for developers to create security solutions that can monitor and respond to system events, such as file access, process execution, and network activity.
- **Malware Removal Tool (MRT)**: Automatically detects and removes known malware from the system.

10. Secure Enclave

- **Secure Enclave**: A dedicated security subsystem within Apple Silicon that handles cryptographic operations and stores sensitive data such as encryption keys and biometric information. It is isolated from the main processor and operating system, providing an additional layer of protection.

Thank you