COMPSCI 750 – Assignment 7
Gunnar Yonker

(1)

Commercial OSes like Linux or Windows are often considered less secure because they have a larger attack surface due to their widespread use. However, both Linux and Windows have a variety of security features and receive regular security updates. Their potential vulnerabilities or weaknesses can be exploited if not properly managed and configured. This means that more attackers are actively trying to find vulnerabilities in these systems. Additionally, they often come with default configurations that may not prioritize security, and they have a large number of third-party software and drivers that can introduce security vulnerabilities. Finally, the complexity of these operating systems can make it challenging to identify and fix all security issues, leading to potential vulnerabilities.

(2)

Chroot: The Chroot mechanism allows a process to change its root directory to a different location in the file system hierarchy, effectively creating a “chroot jail”. This mechanism restricts the process and its child processes to a specific directory subtree, isolating them from the rest of the file system. It enhances security by limiting the scope of file system access for processes, reducing the risk of unauthorized access to sensitive files. It is worth noting that while chroot provides a level of process isolation, it was not designed primarily for security. In certain scenarios, especially when the process inside the chroot jail has root privileges, there are methods to escape. Therefore, relying solely on chroot for security can be risky.

Capability-based security model: The capability-based security model is a fine-grained access control mechanism that grants processes specific privileges or “capabilities” based on their needs. Each capability represents a particular permission or action that a process can perform. By using capabilities, the system can enforce access control at a more granular level, reducing the risk of privilege escalation and unauthorized actions.

These mechanisms work together to enhance the security of Xenix by isolating processes and controlling their access to system resources. Chroot provides filesystem isolation, while the capability-based model ensures that processes can only perform actions they have explicitly been granted permission for, reducing the potential for unauthorized access and privilege escalation.

(3)

A Linux Security Module (LSM) hook is a function or mechanism in the Linux kernel that allows for security modules to enforce access controls and other security policies. These hooks are used to mediate and control various operations on kernel objects and security fields in kernel data structures.

Differences between LSM hooks and discretionary access control (DAC):

DAC is a traditional access control mechanism in Unix-like operating systems that relies on file ownership and permissions. It allows the owner of a file to set permissions (read, write, execute) for the file, and these permissions can be modified by the file’s owner or the superuser (root).

LSM hooks are a more flexible and extensible way to enforce access controls. LSM allows security modules like SELinux and AppArmor to define fine-grained policies that go beyond traditional DAC. With LSM, you can create policies based on various attributes like process context, file paths, and more. These policies are not tied to ownership and permissions but can be based on any security attribute.

In summary, LSM hooks provide a mechanism for implementing mandatory access controls (MAC) and enforcing security policies beyond what discretionary access control provides.

(4) App Armor

a. Main Features of AppArmor:

- AppArmor is a mandatory access control (MAC) framework for Linux that focuses on application-level security.
- It uses path-based access control, where security policies are defined based on file paths rather than labels.
- Profiles define the allowed behavior for specific applications or processes.
- Profiles are written in human-readable text files, making it more accessible for administrators.
- It supports both enforced and complain modes, allowing users to test policies without full enforcement.
- AppArmor can restrict not only file access but also network operations.
- It provides security against zero-day vulnerabilities and unknown flaws in applications.
- AppArmor aims to strike a balance between security and ease of use.

b. Drawbacks of AppArmor:

- Path-based restrictions can be less granular than label-based systems like SELinux, potentially leading to less precise access control.
- Profiles need to exist for each application or process that requires protection, which can be time-consuming to create and maintain.
- It may not provide as fine-grained control as SELinux for certain security requirements.
- Some argue that it may offer less security than SELinux due to its design.
- AppArmor's protection is limited to applications with profiles, leaving unprotected applications vulnerable to attacks.

Due to the path-based nature of AppArmor, changes in file paths, such as moving or renaming directories or files, can render profiles ineffective until they are updated. This can introduce administrative overhead and potential vulnerabilities if not properly managed.

c. Differences Between AppArmor and SELinux:

- **Access Control Module:** AppArmor uses a path-based access control model, while SELinux uses a label-based access control model.
- **Ease of Use:** AppArmor is often considered easier to set up and maintain than SELinux, making it more accessible for administrators.
- **Integration:** SELinux is deeply integrated into various Linux distributions like Red Hat and CentOS, while AppArmor is primarily associated with Ubuntu and some other distributions.
- **Granularity:** SELinux can provide more fine-grained control over system resources and access, which can be advantageous in high-security environments.
- **Community and Development:** SELinux has been developed and used in government and enterprise environments for a longer time, while AppArmor is more community driven.

- **Learning Curve:** SELinux may have a steeper learning curve due to its complexity compared to AppArmor.
- **Philosophy:** SELinux adopts a “default deny” philosophy where, by default, everything is denied unless explicitly allowed. In contrast, AppArmor generally allows operations unless they are explicitly denied in the profile. This fundamental difference impacts how administrators approach policy creation and enforcement for each system.