Towards a user-friendly reference monitor for files using tags and namespaces

FENG, Zeyuan[1] Yasushi Shinjo[1]

1. Introduction

Nowadays, various application programs emerge one after another, and this extremely enriches and facilitates people's work and daily life. Ever-evolving applications are creating security concerns. A typical computer user has a PC, run various applications and uses it for multiple purposes at the same time, such as private projects, work projects, etc. Because the accidental sharing of files between these different projects often causes troubles, we need an isolation mechanism between different projects in a PC. In addition, it is important to protect users from malware attacks. The most common method to achieve these objectives is access control, which means restricting what users can do and what programs executed by users can do [1]. The module that performs access control is called a reference monitor.

At present, the common access control methods include Discretionary Access Control (DAC), Mandatory Access Control (MAC), Role-based access control (RBAC), etc. In Linux, SElinux and Apparmor are the most commonly used MAC kernel modules. In MAC, an administrator edits and manages the access control policy for general users. The advantage of this method is that, the administrator is an expert, has a lot of knowledge, and can enforce more secure access control policies on behalf of general users.

Most MAC mechanisms have poor usability for general users. For instance, SELinux is disabled by default in many Linux distributions, and it is only enabled in enterprises that require a higher security level.

Firewalls are reference monitors for network traffic. Typical firewalls can only be used by professional administrators. However, there is a kind of firewall which is called an application-level firewall. Application-level firewalls run on general users' PCs, and filter traffic based on rules. Users can interactively edit rules for each application. The representation product is Little Snitch[*1].

The objective of this research is to build a reference monitor for files similar to application-level firewalls, in which users can easily isolate applications according to their purpose. Users can edit policies dynamically and interactively without special knowledge. This reference monitor must have a GUI for high usability.

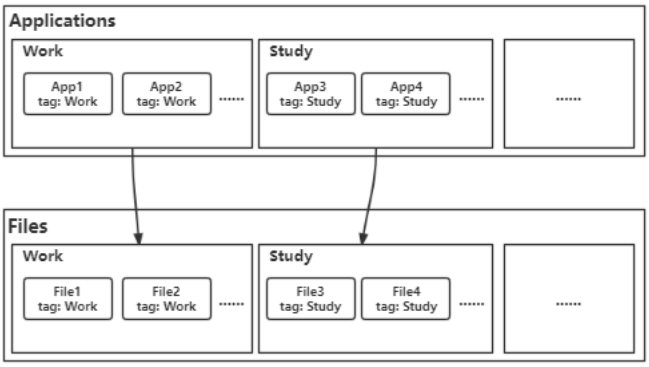

Fig. 1 Tagging application programs and files for access control.

2. Acces Control using tags and namespaces

In our reference monitor, we use tags and namespaces to isolate applications and files according to their purpose. **Fig. 1** shows this mechanism. Users can add tags to applications and files, and group them according to projects, such as work, study, etc. We choose tagging because general users can do it. For example, in macOS, general uses can perform tagging with Finder, the file browser.

We call one same-purpose set of applications and files *a zone*. For convenience, when an application creates a file, the tag of application is automatically added to the file by default.

When users open an application, it runs in a namespace which is only used for the application and files in the same zone. When an application with a tag runs, files with the same tag are automatically become accessible in the namespace. Sometimes, a user needs to run an application in another zone temporarily. For instance, a user regularly uses text editor Emacs[*2] for processing files in the zone "Study". In addition, the user has to edit files in the zone "Work". Frequently changing application tags may cause problems such as accidentally giving a wrong tag to a new file. We must provide support for temporarily changing an application's zone in a user-friendly way.

[1] University of Tsukuba

[*1] https://www.obdev.at/products/littlesnitch/index.html

[*2] https://www.gnu.org/software/emacs/

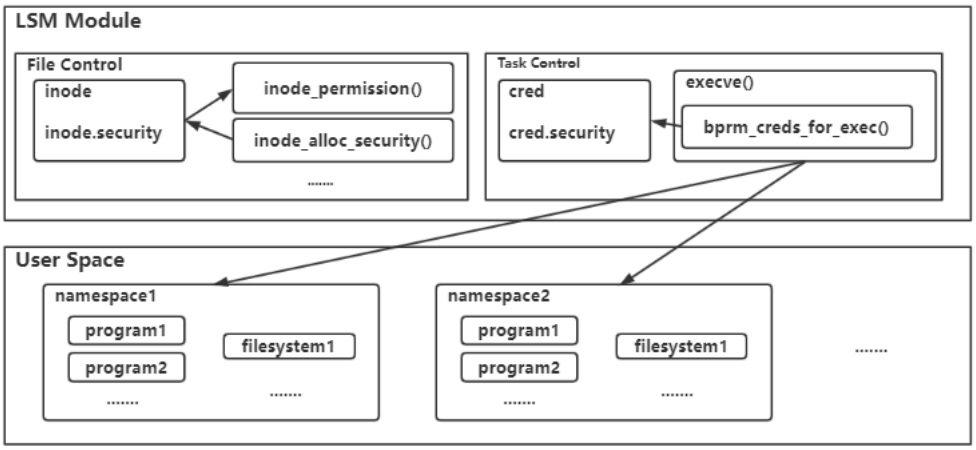

Fig. 2 The implement of the reference monitor with Linux Security Module.

We use Linux Security Modules (LSM) [2] and Linux namespaces to implement zones in the reference monitor. **Fig. 2** shows the structure of the reference monitor. LSM is a generic framework for access control in the Linux kernel. Developers can add LSM hooks into various kinds of kernel objects such as inodes, tasks, credentials, etc. Developers also use security fields in these objects.

We add the following LSM hooks and security fields to implement the reference monitor.

- The hook function of `inode_permissions()`. This is called when a file is accessed. In this hook, we use `inode_alloc_security()` to allocate the security field `inode.security` and store tags.
- The hook function of `bprm_creds_for_exec()`. This is called when `execve()` is called. We use this to create or join in the corresponding namespace according to tags in the security field `cred.security` of the task structure.

When a user executes an application program with the system call `execve()`, the kernel does the following things.

(1) The kernel calls the function `sys_execve()`, which implements `execve()`.

(2) `sys_execve()` calls function `bprm_creds_for_exec()`. This function gets the information of the executable file, such as the access mode and the set-UID bit. In this function, we gets the tag from the security field `inode.security` of the executable file.

(3) We find the existing namespace of the tag. If there is no such namespace:
   - We create a namespace of the tag, and
   - We mount the file system of the tag to the namespace.

(4) We change the namespace of the process to the namespace of the tag, and set the tag to the security field `cred.security`.

3. Related Work

Many researchers agree that it is difficult to use MAC mechanisms such as SELinux, and are working on improving the usability of the mechanisms. The paper [3] proposes a high usability access control model named Functionality-Based Application Confinement (FBAC). They apply the functions of an application to roles in the RBAC model. They abstract and structure the privileges and resources that the application has and can access. This represents functions hierarchically and parametrically. Users can parametrically use the concept of abstract functions to manage the permissions of applications. For example, the concept of "browser" includes http, ftp and other resource access permissions. The parameters may include the paths of the browser modules.

Their work is dedicated to improving the usability of the access control mechanisms so that users without extensive expertise can efficiently edit access control policies. However, the relevant abstraction design is a very complex work. In this paper, we use tags to describe access control policy because general users are familiar with tags.

4. Conclusion

We are implementing a reference monitor of files for a broad range of general users. They can use tags to describe access control policies. We are developing the reference monitor by using LSM and namespaces in Linux. We plan to design a GUI to improve its usability furthermore.

5. Reference

References

[1] Sandhu, R.S. and Samarati, P.: Access control: principle and practice, *IEEE Communications Magazine*, Vol. 32, no. 9, pp. 40-48, Sept. 1994.

[2] Wright, C, Cowan, C, Smalley, S, Morris, J and Kroah-Hartman, G.: Linux security modules: General security support for the linux kernel, *11th USENIX Security Symposium (USENIX Security 02)*, 15 pages, 2002.

[3] Schreuders, Z.C. and Payne, C.: Reusability of Functionality-Based Application Confinement Policy Abstractions, *Springer LNCS*, Vol. 5308, 2008.