Whitelist-Based Security Software for Linux

Minoru Suzuki,
Yasuhiro Otake

Keywords Security, Whitelist, Anti-virus measures

Abstract

More than 20 years have passed since security in computer systems became a concern.

WhiteShield, which makes a security-focused Operating System (secure OS) module inside the Windows environment, was released more than 10 years ago, and continues to be used today. WhiteShield employs a method called the Role-Based Access Control (RBAC), which defines a pre-approved operations role, making it possible to deal with unknown security threats. This role is called "Whitelist".

As the importance of security increases, we also adopted the same method for Linux. This makes a secure OS module inside Linux environment.

Our industrial computer products are embedded products that perform control and monitoring. The whitelist method is suitable because it does not require software updates such as security patches and puts less load on the Central Processing Unit (CPU).

1 Preface

By around 2010, the Linux free software community had developed a function to make a security-focused operating system (secure OS)※1 module inside the Linux environment.

The basic functions are standardly built into Linux as secure OS module. The whitelist required for actual operation is, however, not prepared in advance. This is because it is necessary to give appropriate permissions to all software that runs under Linux that is necessary to respond according to the application.

This time, we built and commercialized an application whitelist suitable for the operation of our products. This paper introduces the whitelist and its method.

2 Applicable Computer Security Measure

Measures for computer security have a broad meaning. The subject is a computer virus. This refers to a measure against any "malicious software". "Malicious software" that causes disadvantages to computer users is classified and has many names, but the general term is "virus".

In addition, the word "vulnerability" related to computer security. This is a risk that a virus will be introduced into the software of the computer and will be activated.

2.1 Vulnerabilities and Malicious Software

An example of the process flow until a virus is introduced and activated is as follows.

(1) A "stepping stone" is created to send the main body of the virus using the vulnerability. Fig. 1 shows the use of vulnerabilities.

(2) A "stepping stone" is used to create the main body of the virus and a device that automatically executes the virus. The main body of the virus is an executable file on disk. Auto-execution devices are also created by altering the contents of existing files. Fig. 2 shows the use of a stepping stone.

(3) Viruses are executed by automatic execution devices, and malicious acts are performed. Fig. 3 shows the virus works.

2.2 Whitelist Role

Specific software normally does not create virus executable files or alter existing files. The application whitelist role is to register normal opera-

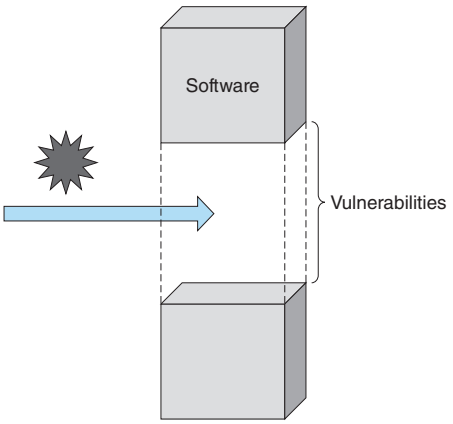

Fig. 1 Use of Vulnerabilities

With bad use of Vulnerabilities, a “stepping stone” is created to deliver the virus.

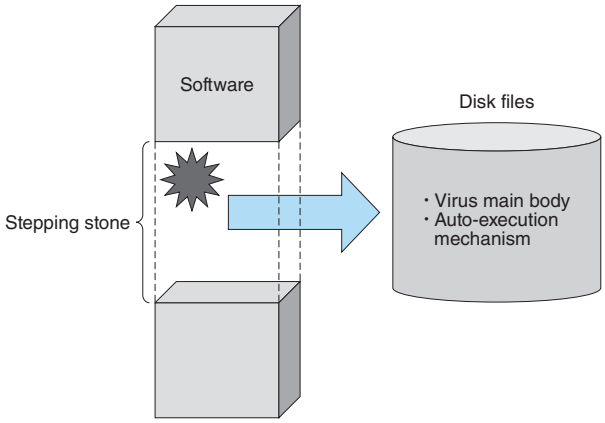

Fig. 2 Use of Stepping Stone

With an adverse use of a stepping stone, the virus main body is delivered into the disk file.

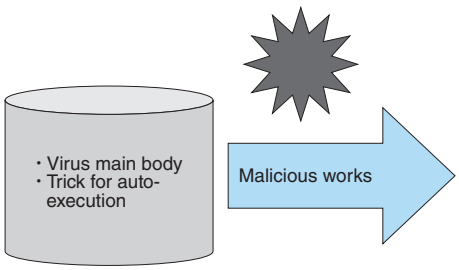

Fig. 3 Virus Works

Virus works are executed by the auto-execution mechanism.

tions as a “whitelist role data” and block operations that are not on it “whitelist role data”. **Fig. 4** shows an overview of the whitelist. **Table 1** shows a comparison of the whitelist and the blacklist used in the usual security software. The advantage of the application whitelist is that there is no need to update the

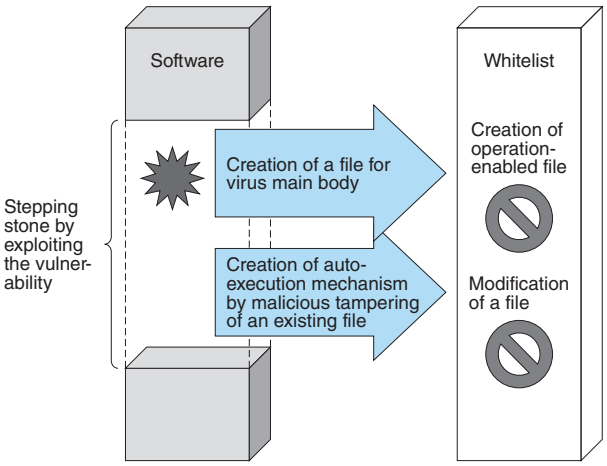

Fig. 4 Overview of Whitelist

Any operation not registered in the whitelist is blocked.

Table 1 Comparison of Whitelist and Blacklist Used in Usual Security Software

Features of the whitelist method and the blacklist method are shown.

Item	Whitelist method	Blacklist method
List updating	Unnecessary	Necessary
Measures taken against unknown threat	Possible	Impossible
CPU load	Light load	Heavy load for control and monitoring during collation with black list.
Creation of a list	Needed to create by oneself (Auto-creation function available)	Distributed by a security software vendor

list and the load on the Central Processing Unit (CPU) is small.

3 Linux Secure OS Module

SELinux is a typical secure OS module for Linux. TOMOYO Linux was adopted this time. **Table 2** shows a comparison of TOMOYO Linux and SELinux. One of the features of TOMOYO Linux is that it can be used by anyone who is not a security expert.

3.1 Secure OS Modules and Whitelists

When a user program manipulates a file, a set of hook functions, called the Linux Security Module (LSM), is built in, that temporarily hooks the processing calls to the secure OS module. The secure OS module checks the hooked system calls (such as read, write, and fork functions), against the whitelist and determines whether to permit such

Table 2 Comparison of TOMOYO Linux and SELinux

Features of the TOMOYO Linux and the SELinux are shown.

Item	TOMOYO Linux	SELinux
Adoption by Linux Distribution	Nil	Red Hat Enterprise Linux, etc.
Policy to whitelist (policy)	Generation of customize by auto-learning enabled	Reference policy offered to Distribution
General viewpoint to whitelist (policy)	Establishment of specifications to enable non-expert on security to utilize	Difficult to use for non-expert on security
Conformance to built-in Linux	Auto-learning possible for built-in type Linux	Difficult to use because of no provision of reference policy
Conformance to file system	Independent on file system	A function available for dependence on specific file system

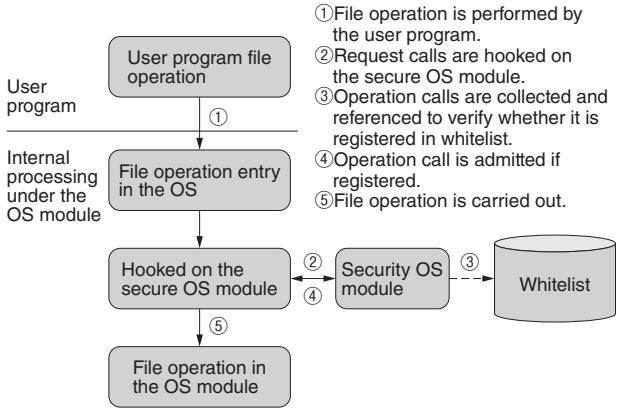

Fig. 5 Overview of Secure OS Module and Whitelist

File operation calls are collected and referenced with the whitelist by the secure OS module.

operation. Only file operations permitted here are performed. Fig. 5 shows an overview of the secure OS module and whitelist.

4 Considerations for Operation

Strong security is desirable, but it must also be compatible with operations such as maintenance. In Linux, management and maintenance by administrators are usually performed using command line operations called shells. This time, we tried to achieve both security and operation by providing two-step authentication for the normal root (administrator) user through shell operations. Fig. 6 shows the idea.

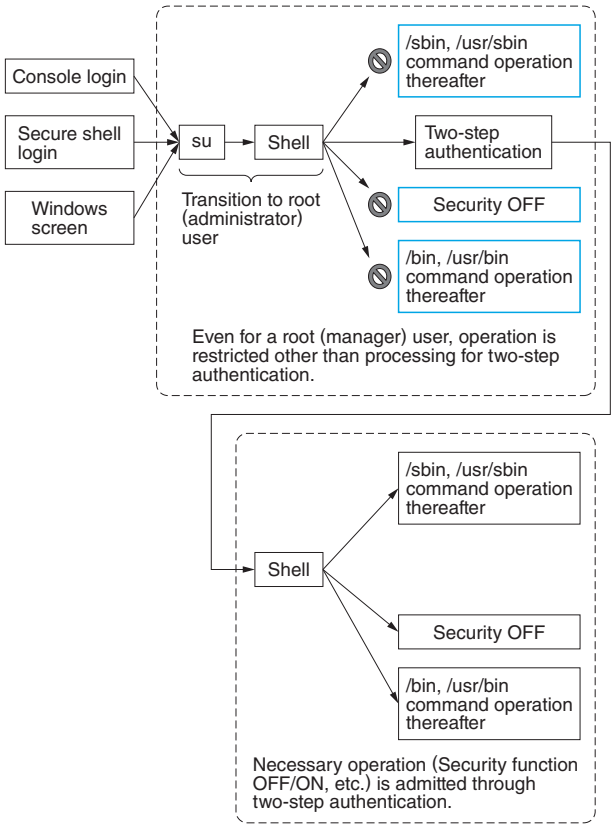

Fig. 6 Idea of Having Both Ways: Security and Execution

Even for a root (Administrator) user, operation is restricted before two-step authentication.

5 Postscript

We introduced whitelist-based security software for Linux. Our products implement security measures not only for Windows but also for Linux (including embedded Linux).

Going forward, we will continue to expand the functions and support so that our customers can use our products with confidence.