2012 Second International Conference on Instrumentation & Measurement, Computer, Communication and Control

A File Integrity Monitoring System Based on Virtual Machine

Zhu Wang
Graduate School of Arts and Science
New York University, New York, NY, USA 10012
zw401@nyu.edu

Tao Huang, Sha Wen
Department of Computer Science and Technology
Huazhong University of Science and Technology Wuhan,
Hubei, China 430074

***Abstract*—This paper describes the design and implementation of a file integrity monitoring system, named FSGuard, based on the virtualization software Xen. Monitored system (DomU) runs in full virtualized mode on Xen, therefore it is unable to perceive the existence of the underlying VMM, but its system calls related to file operations are recorded in real time. User mode programs in DomU provide configuration and management interface, so that the administrator can assign a certain DomU to specify the access control policy and a list of files that need to be protected. These characters make FSGuard possible to monitor file operations in real time, and get feedback through the user mode program in DomU.**

Keywords—file integrity; virtualization; intrusion detection

I. INTRODUCTION

There are two methods to monitor file integrity currently: periodical monitor and real-time monitor [1]. Periodical monitoring tools compare a file's current properties (such as contents, owner, and last modified time) to the prior collected information. Tripwire takes snapshots first for all the files that need to be protected. The system administrator can detect if they've been tampered with. For real-time monitor, TPM (Trusted Platform Module) ensures the integrity of system's boot processes (BIOS, Boot Sector, Kernel, and Application Program). This mechanism establishes a trust chain from hardware, so system is difficult to be modified by malicious software. However, these two methods both have their own shortcomings. Periodical monitor only runs at some specific points, instead of detecting persistently during the whole process. And the real-time monitor only works during the system boot.

There are several inadequacies of current file integrity monitoring methods:

1) Unreliability: Monitoring tools will be no longer valid if they were modified or disabled by malicious software.

2) Cannot meet the requirement for real-time monitoring: Apparently periodical monitor cannot meet the requirement for real-time. Real-time monitor can notice aggressive behaviors as soon as possible, and perform defense whenever needed.

3) Insufficient information. When the critical files in the monitored system has been modified, the system administrator may need to obtain all the information related to the attack, such as time, method, etc.

The system in this paper implemented the integrity protection based on Xen virtualization technology, achieved integrity protection for file system in DomU on VMM layer, strived to overcome the deficiencies of the traditional ways.

II. IMPLEMENTATION PROPOSAL

A. Full Virtualization Technology

In x86 architecture, the CPU supports 4 privilege levels ranging from 0 (highest priority), to 3 (lowest priority). It assigns level 0 to kernel and level 3 to application. From OS/2 to the present, all well-known operating systems based on x86 have not yet constructed on level 1 and 2. Applications in level 3 cannot execute some privilege instructions, such as LGDT, LIDT; only the operating system kernel can invoke them. Generally, level 0 instructions run on level 3 may lead to exceptions, however, it is also possible to get different results executing some instructions both on level 0 and 3 without any exceptions [2].

With the help of virtualization technology, Xen directly manages physical hardware, and provides abstract interface to its upper operating systems. Xen runs in level 0, it provides an environment to make the monitored systems, which are actually running on higher levels, believe they are running on level 0. In full virtualization mode, we cannot make any changes to operating system, normally there are two methods to apply the changes: a. hardware assistance; b. execute privileged instructions through instruction translation.

With hardware assistance, Xen does not need to modify the guest operating systems in full virtualization mode, because Intel and AMD are using different hardware methods to do the similar operations, such as Intel VT and AMD SVM. Take Intel VT as an example, CPU supports instructions about virtualization (like VMXON, VMXLAUNCH, etc.), and provides two kinds of processor modes: root mode and non-root mode. When specific events (such as privileged instructions, external interrupt, page fault, etc.) occur, it switches CPU from non-root mode to root mode. This process is called VMEXIT. In this process, the states of guest OS (registers and contexts) are saved to VMCS region managed by Xen, which will judge the reason why VMEXIT occurs and perform appropriate actions later. After the events processed, CPU re-executes the instruction stream from guest OS, meanwhile, the processor switches from root mode to non-root

978-0-7695-4935-4/12 $26.00 © 2012 IEEE
DOI

mode. This process is called VMENTRY. The following figure 1 illustrates the switch between the two modes.

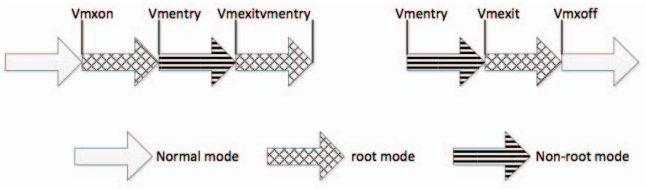

Figure 1. Switch between root mode and non-root mode

B. System Call

Operating system kernel has higher priority than applications. Applications cannot directly call processes provided by kernel. To call these processes, applications must use the interface provided by kernel, i.e. system call [3].

In Linux, there are two different system call implementations: (1) soft interrupt. When user mode processes invoke system call, kernel checks its privilege and input data. However, the context switch from level 3 to level 0 in this procedure is quite time-consuming; (2) fast system call (sysenter). It provides a fast switch from user mode to kernel mode, it has higher execution efficiency than soft interrupt since there's no privilege check or push operation on stack. Linux 2.6 kernel or later supports fast system calls. SYSENTER_EIP_MSR must be filled before executing "sysenter", because it specifies the entry address and kernel mode stack.

We need to intercept fast system calls from DomU via Xen, analyze the data from the system calls and determine whether the files being modified are the ones that administrator wants to protect. Actually, there is a tool XenFit can monitor real-time file integrity, which inserts hooks into monitored system, and intercepts system calls related to file operations. However, if these hooks encounter malicious attacks, the obtained information may not be reliable and correct. In order to overcome this flaw, we designed FSGuard to intercept system calls in another way.

FSGuard analyze the information not directly from system calls, but from page fault exceptions. Under normal circumstances, guest OS jumps to interrupt handler when interruption occurs, and page fault exception may not occur in this period. However, FSGuard will automatically set SYSENTER_EIP_MSR to generate page fault exceptions `intentionally`. Thus, page fault exception will follow every system call before DomU jumps to interrupt handler. By this method, we can get all relevant information about registers and stacks from page fault exceptions; besides, this procedure is completely free of malicious intrusion.

C. Implementation

After intercepting system calls, the next step we need to do is data analysis, that is to get the target data by analyzing the registers. When Linux invokes system calls, system call numbers are stored in the register EAX, and the corresponding parameters are stored in other registers respectively. We need to screen out others and focus on some particular data in EAX since we are only interested in the system calls related to read and write of the files. `In the same way`, we can get file path and related information of file operations by analyzing the contents of each register. The file properties that we need to monitor in the user interface of Dom0 can be defined in user interface, which include file path that need to be protected, and the protection operations (read protection, write protection, etc.). FSGuard sends this property information to shared pages of Dom0 to offer Xen access to read it when needed.

Xen reads records about protection policy and the list of files that need to be protected from the shared pages. When DomU invokes fast system calls, Xen process the properties (including the system call number, file name, absolute path) that just intercepted, compares the file properties to the records in shared pages. When there is a filename in the file list, system will check its operation information and protection policies. If the file's user has the `permission` to process the system calls, system will `dispose it in normal process`. Otherwise, system operates the stacks of guest OS to make it return to the original state that before the system call invoked without impacting any other DomU.

III. SYSTEM ARCHITECTURE

The FSGuard system consists of two parts, the kernel mode and the user mode. Figure 2 illustrates the architecture of the system. To implement Kernel mode, we need to patch on the source code of Xen, then compile and install it. The kernel mode program is used for intercepting and analyzing fast system calls in DomU, building up communications about protection policy between Xen and DomU. The user mode program is mainly used to provide an user-friendly interface, and to implement user interaction. Through the user mode program running on DomU, the users with administrator privilege can define file integration policies in DomU.

Kernel mode needs to implement these functions: listen and intercept system calls, screen out the system calls we may be interested in, analyze file path by entry parameters, combine policy information to determine if the operations are permitted. User mode should complete the following functions: interpret protection policy, monitor shared pages, and obtain feedback from kernel mode.

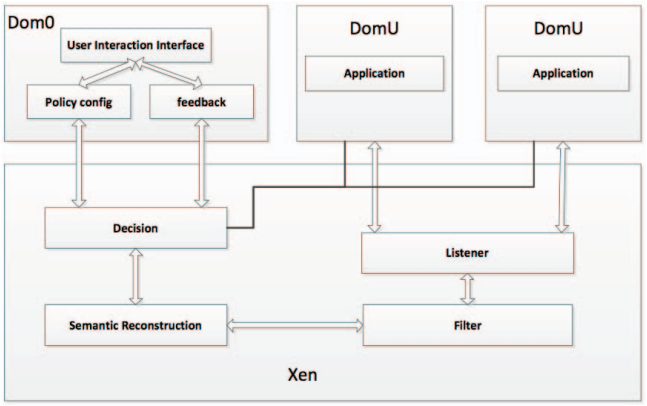

Figure 2. System Architecture

The function of each module is described below.

1) Listener Module: Listener module is in charge of monitoring. It monitors whether DomU generated fast system calls.

2) Filter Module: When intercepting system calls, we get all information of registers at that time. At this point, Dom U is paused (because of VMEXIT), since system call numbers are stored in EAX, we need to check protection policy to select the concerning system calls.

3) Semantic Reconstruction Module: This module is responsible for converting lower level information in registers to meaningful higher-level semantic information.

4) Decision Module: This module is for determining if the received files are that we want to protect.

5) Policy Configuration Module: It has two functions. One is to select policy file, the other is to choose protection mode.

6) Monitoring Information Feedback Module: Through decision module, determine if the running system call operation is legal or not, then process system calls according to the normal processing procedure.

7) User Interaction Interface Module: If there's information from kernel mode needs to be displayed on the client; this module would be notified and then display the message on client.

IV. CONCLUSION

Virtualization would be a main trend in the future due to its high resource utilization and energy-saving characters. Xen is an elite in the virtualization world. FSGuard is built on Xen and it provides an efficient and secured mechanism to protect file integrity. It depends on analysis of returned information from page fault exceptions after system calls intercepted. According to the analysis, we will know whether the file has been modified and whether the integrity is secured. The main features of FSGuard are:

1) High Security: It has file integrity monitoring part and monitored system in isolated virtual machine, so the monitoring tools cannot be attacked by malicious software, which ensures the high security of monitoring tools.

2) Real-time: Through the system calls intercepted from VMM layer, achieved real-time integrity monitoring to the key files, and invaded behavior can be found in a timely manner.

3) Transparency: Monitored system cannot perceive that it's running on VMM. So monitoring tools are transparent to it, and wouldn't affect any activities of monitored system.

REFERENCES

[1] Qiaoyun Gu, Anxin Li, The Design and Implementation of File Integrity Monitoring System Based on Windows [J], Computer Engineering, 2004, 30(12)

[2] [2] Da Su, Virtualization: The Key of High Effeciency IT [J], Computer Users of China, 2009(9)

[3] [3] Daniel P B. Marco Cesat. Understanding the Linux Kernel [M], 3rd Edition, Sebastopol: O'Reilly, 2005