Computers and Mathematics with Applications 60 (2010) 256–266

Contents lists available at ScienceDirect

Computers and Mathematics with Applications

journal homepage: www.elsevier.com/locate/camwa

A guest-transparent file integrity monitoring method in virtualization environment

Hai Jin*, Guofu Xiang, Deqing Zou, Feng Zhao, Min Li, Chen Yu

Services Computing Technology and System Lab, Cluster and Grid Computing Lab, School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan, China

ARTICLE INFO

Keywords:
File integrity protection
Transparent monitoring
Real time
Xen

ABSTRACT

The file system becomes the usual target of malicious attacks because it contains lots of sensitive data, such as executable programs, configuration and authorization information. File integrity monitoring is an effective approach to discover aggressive behavior by detecting modification actions on these sensitive files. Traditional real-time integrity monitoring tools, which insert hooks into the OS kernel, are easily controlled and disabled by malicious software. Such existing methods, which insert kernel module into OS, are hard to be compatible because of the diversity of OS. In this paper, we present a non-intrusive real-time file integrity monitoring method in virtual machine-based computing environment, which is transparent to the monitored system. The monitor is isolated from the monitored system, since it observes the state of the monitored system from the outside. This method brings two benefits: detecting file operations in real time and being invisible to malicious attackers in the monitored system. Furthermore, a kind of file classification algorithm based on file security level is proposed to improve efficiency in this paper. The proposed file integrity monitoring method is implemented in the full-virtualization mode supported by the *Xen* platform. The experimental results show that the method is effective with acceptable overhead.

1. Introduction

One of the fundamental goals of computer security is to ensure the integrity of system resources [1]. A large number of safe accidents result from critical files modified in the system. The attackers intrude into the computer system, and hide their traces by tampering critical files, such as executable files and system logs. File Integrity Monitoring (FIM) is one of the most popular approaches to observe hostile behaviors, such as modifying system log, appending backdoors, inserting Trojan horses.

In order to avoid serious damage to the system, it is necessary to observe malicious behaviors in the system. Current FIM can be classified into two types: Periodic FIM (PFIM) and Real-time FIM (RFIM). PFIM tools compare current attributes of the files with previously gathered, for example, the owner, the content, and the last modification time. *Tripwire* [1] gives a snapshot of the protected files firstly, and the administrator can verify their integrity periodically. Trusted Platform Module (TPM) provides integrity authentication for a trusted path (BIOS, boot sector, OS kernel, applications) during the starting procedure of the system. It is difficult to be compromised by attackers because the trust chain is extended from the hardware. PFIM verifies the integrity of the system at one "point", rather than during the whole "process". RFIM tool works in the OS kernel to intercept file operations during the execution process of the system. *XenFIT* [2] is a real-time integrity monitor

* Corresponding author.
E-mail address: hjin@hust.edu.cn (H. Jin).

0898-1221/$ – see front matter © 2010 Elsevier Ltd. All rights reserved.
doi:10.1016/j.camwa.2010.01.007

on the well-known *Xen* virtualization platform [3–5]. It puts several hooks into the kernel of the monitored system, and intercepts system calls related with these file operations. In order to monitor file operation in real time, a kernel module must be inserted in the monitored system. However, this module is easily attacked or masked by rootkits.

So, perfect FIM tool should satisfy the following requirements:

(1) *Attack resistant*: FIM tool should continue working even if the monitored system is completely controlled by malicious software (malware). In *XenFIT*, the system call interception module runs in the monitored system, it may be disabled by malware.
(2) *Real time*: FIM tool should get enough information in real time when configuration or executable files are modified by malicious intruders. *Tripwire* can check the integrity of files periodically, and TPM can measure any file during the system boots, but they cannot ensure the file security during the entire runtime of the system. Furthermore, the administrator does not know when the system begins to be unreliable, and which users tamper critical files.
(3) *Transparence*: FIM tool should neither modify the monitored system, nor insert any kernel module. *XenFIT* inserts multiple breakpoints in the monitored system, which restricts the practicability of the monitor. There are various monitored systems, for example, different types of OSes, diverse versions of the same class.

Virtualization technology gives people a novel approach to meet these requirements mentioned above. It provides the isolation between FIM tools and malwares, and an advanced FIM design based on such technology is feasible. From the report of IDC [6], virtualization [7–9] becomes an integral part of the IT infrastructure, and the virtualization services market will grow sharply in the future 5 years. Virtualization decouples OS with the underlying hardware, and supports multiple OS instances on the single hardware platform. An OS instance and its upper applications are encapsulated as a Virtual Machine [8](VM). The core component of virtualization is Virtual Machine Monitor [9](VMM), which is a "thin" software layer located under the traditional OS. The VMM owns full domination of the underlying hardware, and it can monitor all events happened in the VM. At present, the design tendency of security systems in virtualization environment is to pull the monitor tool out of the Monitored VM (MVM) and deploy it in another protected VM. Under the virtualization architecture, FIM tools need to observe the file changes on the monitored VM, and prevent possible attacks from the VM at the same time. The monitoring points are moved from the monitored system to the VMM, and cannot be sensed by the monitored system.

In this paper, we present a guest-transparent RFIM method in virtual computing environment. Intercepting file operations is implemented in the VMM, and it is no need to append any module in the monitored system. The RFIM tool locates in another Privileged VM (PVM), and observes file operations in the MVM. We can obtain enough file information, including process identity, file name, file operation, time et al., when one critical file is modified in the MVM. The RFIM tool is isolated from the MVM and transparent to the MVM. All files are classified into three categories based on the file security level. The file classification algorithm is proposed to take different actions on the files which belong to different sets.

The rest of this paper is organized as follows: Section 2 introduces the related work and background. Section 3 proposes the RFIM method in the virtual computing environment and discuss file classification according to the significance level. Section 4 describes the implementation of RFIM in an intrusion prevention system, named *VMFence* [10], based on the *Xen* platform. The experiments on *Xen* are described in Section 5. Conclusions and future work are presented in Section 6.

2. Related work

File integrity monitoring is one function of Host-based Intrusion Detection System [11] (HIDS), and it is used for the administrator to discover malicious behaviors. In this section, we introduce PFIM and RFIM respectively.

Tripwire [1] is a representative example of PFIM. There are four modes in *Tripwire*: *init*, *check*, *update*, and *test*. In *init* mode, the significant files are specified by the administrator, and *Tripwire* gives a snapshot of these files. The administrator can verify whether these files are tampered in *check* mode. *Tripwire* compares the current hash values of files with the previous values stored in a specified database. The problem of *Tripwire* is the selection of inspection cycle. If the cycle is too short, frequent inspection will impact on the system performance, otherwise, it cannot detect the variation in time. In addition, *Tripwire* executes on the monitored system, and is easily disabled by intruders with the administrator privilege. Except for *Tripwire*, *AIDE* and *Samhain* are the similar tools. Pennington et al. [12] proposed a storage-based intrusion detection system which allows the storage systems to watch data modification. This project implements file integrity monitoring when file modification happens on a file server in the distributed environment.

I^3FS [13] intercepts system calls and injects its integrity checking operations in the kernel mode. It performs checksum comparison in the critical path. *XenFIT* is a file integrity monitor which is implemented on *Xen*. The breakpoints are inserted in the monitored system, and intercept these system calls about file operations, for example, *open*, *close*, and *write*. It records the system call log, and sends it to the PVM. However, It is necessary to put an intercepting system call module in the MVM.

With the emergence of multi-core processor, virtualization technology is widely applied to various aspects of computer systems. VM-based security [14,15] becomes important increasingly. The isolation provided by VMM enhances the security of the detector, while brings a new problem, semantic gap simultaneously.

Virtual Machine Introspection [16,17] (VMI) was firstly proposed by Tal Garfinkel and Mendel Rosenblum in the project, *Livewire* [16], which is a new architecture for building an intrusion detection system with the merits of both high resistance and excellent visibility via VMI. VMI is an effective method to enhance system security including intrusion detection [18], malware detection [19,20], and honeypot [21–23]. *VMwatcher* [24] exports the resources in the MVM to the trusted host