Virtual Machine Introspection Based Rootkit Detection in Virtualized Environments

Tongwook Hwang, Youngsang Shin, Kyungho Son, Haeryong Park

Korea Internet & Security Agency, Seoul, Korea
twhwang@kisa.or.kr

Abstract: Cloud computing and the underlying virtualization technology is becoming more and more popular, and valuable information stored inside cloud computing environments are increasing at an alarming rate. As a result, APT attacks that target the information are also increasing. Because the key element of APT attacks is the rootkit that provides stealth, rootkit detection is an effective defensive measure against APT attacks. In this paper, we discuss how to apply VMI (Virtual Machine Introspection) techniques to detecting rootkits in virtualized environments, and use the insights gained to design an effective and efficient rootkit detection system.
[Hwang T, Shin Y, Son K, Park H. **Virtual Machine Introspection Based Rootkit Detection in Virtualized Environments**. *Life Sci J* 2014;11(7):803-808] (ISSN:1097-8135). http://www.lifesciencesite.com. 116

Keywords: Cloud security; virtualization security; rootkit detection; hypervisor

1. Introduction

Cloud computing is rapidly gaining popularity [1][2] as servers, computing platforms, storage, and more. This is because of the cost benefits that cloud computing offers. This popularity indicates that more valuable information is accumulating in cloud computing environments. The downside of this popularity is that hackers are motivated to target cloud computing systems for the valuable information. This states the necessity of protecting cloud computing systems from dangerous attacks, such as the infamous APT (Advanced Persistent Threat) attacks [3]. APT attacks relies heavily on rootkits, which are a type of malware that provides stealth for itself and malware payloads against system administrators. Therefore, rootkit detection is an effective and efficient defensive measure against APT attacks.

Rootkits are a type of malware that focuses on stealthy control. This allows rootkits to hide itself and its payloads from the system administrators, while extracting information from the system and executing remote commands from attackers. Rootkits operate by modifying the OS system call results to serve the rootkits' purpose. The various techniques utilized to implement this objective lead to the different types of rootkits.

There are two major methodology to detect rootkits; in-the-box methodology and out-of-the-box methodology. In-the-box methodology installs a rootkit detection agent inside the system to be monitored. Out-of-the-box methodology utilizes an external rootkit detection agent that monitors the target system from the outside, usually by applying virtualization and VMI (Virtual Machine Introspection) techniques.

In this paper, we build upon our preliminary works [4][5][6] to design an effective, efficient rootkit detection system. We discuss that out-of-the-box rootkit detection methodology [7] is efficient for virtualized environments. We present a non-comprehensive list of rootkit types and the corresponding detection methods [5]. We also present how to apply the detection methods efficiently in a virtualized environment [5], utilizing VMI techniques [8]. We discuss the design requirements that a rootkit detection system in cloud computing environments should satisfy [4]. Finally, we design a rootkit detection system that takes into account the presented design requirements and application techniques for VMI. We utilize the vIPS platform [6] to better incorporate the design requirements.

2. Rootkits and Detection Methods

In Table 1, we summarized the different types of rootkits and the corresponding detection algorithms [9][10]. The detection algorithms listed in Table 1 dictates where to check for rootkits, but not how to detect rootkits.

In terms of how to check for rootkits, there are two classes of detection methodology, in-the-box rootkit detection and out-of-the-box rootkit detection. We will discuss how the two methodologies apply in virtualized environments, and see that out-of-the-box methodology is superior in virtualized environments.

2.1. In-the-box methodology

In-the-box rootkit detection methodology is similar to the one used by the existing virus vaccine. For this methodology, the detection program is installed inside the system to be inspected, and the signature of the known rootkit or contradiction of the OS call result values will be searched while examining the inside of the target system. There are some problems when the in-the-box rootkit detection methodology is applied to a cloud computing environment.

The first problem is the possibility of A/V storms. In enterprise environments, each system

installs the same host intrusion detection/prevention software and is scheduled to perform inspection at a prearranged time. Therefore, the detection program of all hosts performs inspection at the same time. If this methodology is used in a cloud computing virtualized environment, however, resources of the host server such as CPU and disk bandwidth and storage will be saturated by the considerable load. This phenomenon is named as an A/V storm. Since A/V storms cause a bottleneck, it leads to the significant deterioration of the overall system performance.

The second problem is that it becomes difficult to maintain the security status of virtual machines at a homogeneous state. The security of the entire virtualized system can be maintained only when the rootkit detection software installed in the individual virtual machines maintains the same version. However, it is difficult to confirm whether all virtual machines have the same version of rootkit detection software in a virtualized environment, because the virtual machine can be migrated, duplicated, deleted, or otherwise modified at any time.

The third problem is that using the OS call return value altered by the rootkit is unavoidable, because the rootkit detection program is installed outside of each virtual machine. Due to this limitation, the rootkit is able to deceive the rootkit detection program [11].

2.2. Out-of-the-box methodology

Out-of-the-box rootkit detection methodology detects the rootkit by examining from the outside of the system to be inspected. For this purpose, the target system is virtualized by installing a hypervisor. Afterwards, the internals of the target system is examined from the outside.

The out-of-the-box methodology has many advantages in a cloud computing environment, since both the cloud computing environment and out-of-the-box rootkit detection methodology are based on virtualization technology.

First, there is no need to install and virtualize the hypervisor in the system to be inspected, because all systems for inspection are already virtualized in a cloud environment. This characteristic provides the benefit of simplified installation. It also invalidates some rootkit implementations that detect the change in virtualization status to foil out-of-the-box methodology.

Second, it is extremely difficult for the rootkit inside the virtual machine to recognize the observer who is outside of the virtual machine and to deceive the observer, because the technique used by the rootkit for hiding, altering the OS return value, becomes futile.

Lastly, it is easy to avoid A/V storms. By having only one copy of the rootkit detection software inside any virtualized system, it is possible to achieve flexibility in inspection scheduling such as putting into consideration the load on the entire system.

Table 1. Types of rootkits and corresponding detection algorithms

Level	Type	Detection Algorithm	Search Device	Search Location
User Level	API Hooking	Function Table Analysis	Virtual Memory	IAT
	Inline Hooking	Hash Value Comparison	Virtual Memory	DLL Code Section
	Trojan Horse	Hash Value Comparison	Virtual Disk	Executable File
Kernel Level	SSDT Hooking	Function Table Analysis	Virtual Memory	SSDT
	IDT/MSR Hooking	Function Table Analysis	Virtual Memory Virtual CPU Register	IDT/MSR
	Code Patching	Hash Value Comparison	Virtual Memory	Kernel Code Section
Boot Level	Bootloader Substitution	Hash Value Comparison	Virtual Disk	Boot Sector/Record Bootloader File

Table 2. Types of VMI libraries and corresponding detection algorithms

VMI Library Name	Supported Hypervisor	Supported Device	Technology Employed	Application to Rootkit Detection
LibVMI	Xen, KVM	Memory, CPU	XenControl Library	IAT, DLL Code Section, SSDT, IDT, MSR, Kernel Code Section
LibGuestFS	Xen, KVM, VMWare	Disk	QEMU emulation	Executable Files, Boot Sectors, Boot Records, Bootloader Files
VMSafe	VMWare	Memory, CPU, Disk	Integrated in VMWare	All

3. Rootkit Detection in Virtualized Environments

To utilize out-of-the-box methodology in virtualized environments, a technique that can retrieve information within a virtual machine from the hypervisor is needed. This technique is called VMI (Virtual Machine Introspection). VMI allows access to a number of information within a virtual machine, such as CPU registers, memory, disk, network devices, etc. There are several libraries that implement VMI functionality for various hypervisors, such as LibVMI [12][13] for Xen [14] and KVM [15], VMSafe [16] for VMWare [17], LibGuestFS [18] for most types of hypervisors, etc. In this chapter, we present how to apply the detection algorithms listed in Table 1 as out-of-the-box methodology by utilizing VMI technology. We use LibVMI and LibGuestFS as examples.

LibVMI reads virtual memory and CPU registers. LibVMI uses hypervisor tools, such as the XenControl Library [14], to access critical information such as page tables, then provides access to the requested information. To utilize LibVMI for rootkit detection, we must read related information from the target virtual machines. Table 1 indicates that we need to read the following information from the virtual machines using LibVMI : IAT, DLL Code Section, SSDT, IDT, MSR and Kernel Code Section. Some of this information such as kernel code section can be accessed with simple kernel symbol references. Other information such as IAT require that the corresponding address be calculated from other kernel data structures and be accessed with memory addresses. One of the listed information, namely MSR, requires access to the CPU registers.

LibGuestFS reads virtual disks. LibGuestFS mounts the virtual disk in question into an emulator named QEMU [19], and then communicates with this emulator to provide access to the virtual disk. Table 1 indicates that we need to read the following information from the virtual machines using LibGuestFS: Executable Files, Boot Sectors, Boot Records and Bootloader Files. Two different types of access modes are required. First, executable files and bootloader files are part of the system partition, so we need to be able to access files inside the virtual disk. Second, boot sectors and boot records does not exist as files, but have predetermined sector position relative to disk start or partition start. Therefore we need to be able to access sectors with predetermined offset from disk start or partition start.

4. Design Requirements of a Rootkit Detection System in Virtualized Environments

Detecting rootkits in cloud computing environments did not receive much attention in the past. Therefore, to design effective and efficient rootkit detection systems in cloud computing environments, we need to specify the design requirements. We reiterate the four design requirements from our previous work [4]; agentless virtual security appliances, hypervisor independence, performance and usability.

Agentless virtual security appliance means that the rootkit detection system should avoid using agents inside the target virtual machines, and instead rely solely upon VMI techniques to observe the target virtual machines from an isolated VSA (Virtual Security Appliance). This is to minimize the effect of the various detection evasion techniques that rootkits employ.

Hypervisor independence means that the rootkit detection system should support various different kinds of hypervisors in order to be practically applicable.

Performance means that the rootkit detection system should take measures to not hamper the performance of the cloud computing system in question. This is to compensate for the fact that cloud computing tends to use the hardware resources more efficiently, thereby removing slack resources that can be harmlessly diverted to security operations.

Usability means that the rootkit detection system should provide integration with other system security monitoring tools, such as SIEM (Security Information and Event Management) systems, to provide fast and intelligible alert when an infection incident occurs. This takes into account that a system infected with rootkits usually cannot be recovered without formatting and reinstalling the operating system, which cannot and should not be performed automatically by machine discretion.

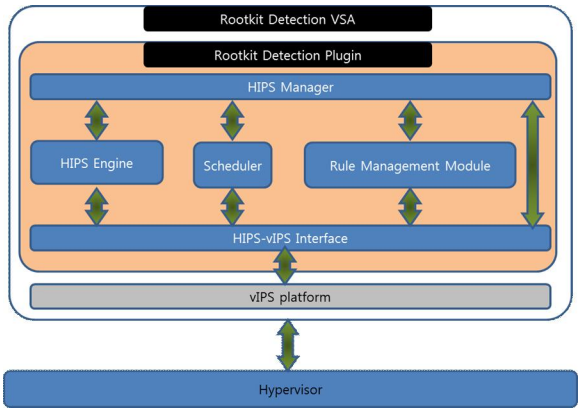

Figure 1. Proposed Rootkit Detection System Architecture

5. Design of a Rootkit Detection System in Virtualized Environments

In this section, we present the architecture for the rootkit detection system. We take into

consideration the design requirement that we discussed in Section 4. We also apply the VMI techniques that we discussed in Section 3. In order to better uphold the requirements, we utilize vIPS platform [6] from our previous work.

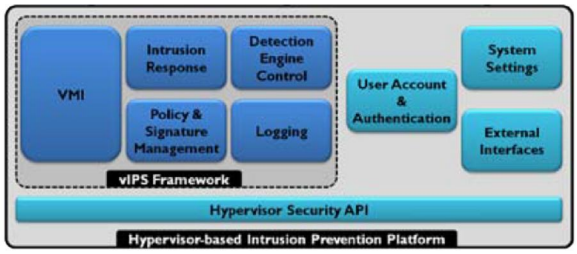

Figure 2. vIPS Platform Architecture [6]

The architecture of the proposed rootkit detection system is presented in Figure 1. The architecture of vIPS platform is revisited in Figure 2. The rootkit detection system is composed as a VSA, providing isolation from other virtual machines. The rootkit detection functionality is implemented as a HIPS engine plugin for vIPS platform.

The modules that compose the rootkit detection system is explained below.

vIPS platform [6] is a hypervisor-independent virtual host/network IPS platform that is designed to help in developing flexible and effective VSA. It implements a hypervisor neutral API for accessing VM information, including VMI functionality. It also provides integration with SIEMs, therefore providing support for easy management of the numerous virtual machines across multiple physical hosts.

HIPS-vIPS interface module provides API access between vIPS platform and the rest of the rootkit detection system. The centralized management of APIs support encapsulation.

HIPS manager provides supporting functionality for the rootkit detection system, including module control, inter-module communication, environment setting management, logging, etc.

HIPS engine is the main module of the proposed system. HIPS engine follows the schedule set by the Scheduler module to apply rootkit detection rules from Rules management module to target virtual machines.

Scheduler module manages the schedule for HIPS engine. It takes into account the current workload of the cloud computing system and the predefined guidelines in the environment settings.

Rule management module manages the rootkit detection rules. It maintains the whole list of rootkit detection rules, and selectively provides the applicable rules for the current virtual machine in question to HIPS engine. It also performs preliminary optimizations on the rules when applicable.

The presented design satisfies all the design requirements that was discussed in Section 4. Having a separate Scheduler module allows for flexible schedule adjustment according to system load, and Rule management module provides rule optimization. This satisfies the performance design requirement. By adopting the vIPS platform, the other 3 design requirements (agentless virtual security appliance, hypervisor independence, usability) are satisfied as well.

The presented design utilizes VMI techniques to monitor virtual machines from a VSA. vIPS platform provides access to hypervisor-neutral API for underlying VMI libraries. The VMI API provided by vIPS platform supports access to virtual CPU registers, virtual memory and virtual storage. Rule management module provides the instructions for utilizing the provided API to access information related to rootkit detection, and HIPS engine executes the instructions. The system as a whole is allowed to utilize VMI as stated in the discussion in Section 3.

The workflow of the presented system design is as follows. When the VSA is started, the vIPS platform starts and prepares all plugins. In the process, rootkit detection environment information is given to HIPS manager. Also, rootkit detection rules are given to Rule management module, and Scheduler acquires information about the currently active virtual machines. Upon acquiring these information, Scheduler builds a preliminary rootkit detection schedule. It starts receiving system load information to make adjustments to the schedule when needed.

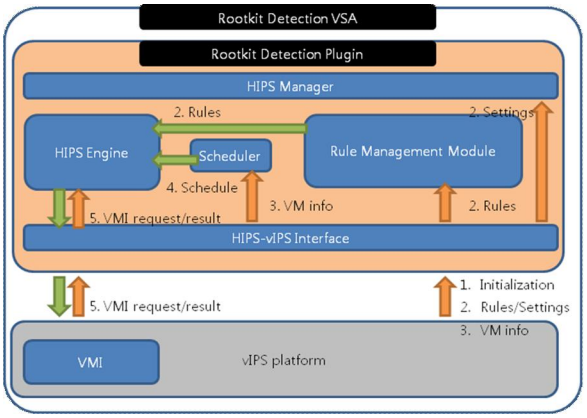

Figure 3. Rootkit Detection System Workflow

When the schedule calls for the test of a virtual machine, HIPS engine starts to operate on the scheduled virtual machine. It receives rule information from Rule management module. It starts

to call VMI functionality provided by vIPS platform through HIPS-vIPS interface. When HIPS engine detects signs of rootkits, the event is reported to HIPS manager, which then passes the report to vIPS platform for further processing, logging and alert generation. This procedure is pictured in Figure 3.

6. Related Works

There are several other works that attempted to apply VMI techniques to security applications. Garfinkel and Rosenblum utilizes callback functions and polling mechanisms to present a VMI-based IDS with a signature-based detection engine targeted for VMware Workstation [8]. Jiang, Wang and Xu present a method to overcome the semantic gap weakness of using VMI techniques to reconstruct the machine state [7].

There are many studies about rootkits and how to detect them. Arnold performed a survey on rootkits and corresponding detection methods [10]. Kim, Park, Lee, You and Yim performed a similar survey from a different angle, focusing on grouping the rootkits by infection area [20].

There are some researches on utilizing VMI for rootkit detection. Ibrahim, Hamlyn-Harris, Grundy and Almorsy suggests reconstructing kernel objects of the target virtual machine inside a separate VSA [21]. Carbone, Cui, Lu and Lee suggests analyzing the operating source code to construct a kernel object type graph that is utilized to build a specialized kernel object reconstruction system [22]. However, they focus on applying the relevant techniques against a single target system by virtualizing the system. Therefore, they do not take into consideration performance and management issues in cloud computing environments where multiple target virtual machines coexist.

7. Conclusion

In this paper, we have analyzed a number of different types of rootkits and the corresponding detection method. We also discussed the superiority of out-of-the-box methodology in virtualized systems for cloud computing systems. We presented how to apply the available VMI techniques to detecting rootkits, by specifying the information that should be accessed and the required access modes to efficiently support the rootkit detection. We defined the design requirements for rootkit detection system in cloud computing environment. Finally, we presented a rootkit detection system design that satisfies the design requirements and takes into account the application methods for VMI techniques.

As a future work, we plan to implement the presented rootkit detection system and test the system for traits such as rootkit detection rate, performance, etc.

Acknowledgements:

This work was supported by ICT R&D program of MSIP/IITP. [10044938, The Development of Cyber Attacks Detection Technology based on Mass Security Events Analysing and Malicious code Profiling]

Corresponding Author:
Dr. Youngsang Shin
Korea Internet & Security Agency
Seoul, South Korea
E-mail: ysshin@kisa.or.kr

References

1. Forrester Research. The evolution of cloud computing markets, 2010.
2. Cisco. Cisco Global Cloud Index: Forecast and Methodology, 2012–2017. http://www.cisco.com/en/US/solutions/collateral/ns341/ns525/ns537/ns705/ns1175/Cloud_Index_White_Paper.html.
3. Cloud Security and APT defense – Identical Twins?, 2012. http://cloud.trendmicro.com/cloud-security-and-apt-defense-identical-twins/
4. Hwang T, Shin Y, Son K, Park H. Design of a Hypervisor-based Rootkit Detection Method for Virtualized Systems in Cloud Computing Environments. In AASRI Winter International Conference on Engineering and Technology (AASRI-WIET), 2013.
5. Hwang T, Shin Y, Son K, Park H. Virtual Machine Introspection Based Rootkit Detection in Virtualized Environments. In International Conference on Advanced Computing and Services (ACS), 2014.
6. Shin Y, Yoon M, Son K. Design of a Versatile Hypervisor-based Platform for Virtual Network-Host Intrusion Prevention. In International Conference on Information Processing, Management and Intelligent Information Technology (ICIPT), 2013.
7. Jiang X, Wang X, Xu D. Stealthy malware detection through VMM-based “out-of-the-box” semantic view reconstruction. In ACM Conference on Computer and Communications Security (CCS), 2007.
8. Garfinkel T, Rosenblum M, A Virtual Machine Introspection Based Architecture for Intrusion Detection, In Network and Distributed System Security Symposium (NDSS), 2003.
9. Reverend Bill Blunden. The Rootkit Arsenal, Second Edition. Jones&Bartlett Learning, 2013.

10. Arnold TM. A Comparative Analysis of Rootkit Detection Techniques, M.S. Thesis, The University of Houston-Clear Lake, Houston, USA, 2011.
11. Sparks S, Butler J. "Shadow Walker": Raising the Bar for Rootkit Detection, In Blackhat, 2005.
12. Payne D, de Carbone MDP, Lee W. Secure and flexible monitoring of virtual machines. In Annual Computer Security Appliances Conference (ACSAC), 2003.
13. LibVMI project. vmitools. http://code.google.com/p/vmitools/.
14. Citrix. Xen. http://www.xen.org.
15. KVM. Kernel based virtual machine (KVM). http://linux-kvm.org.
16. VMware. VMsafe. http://www.vmware.com/go/vmsafe.
17. VMware, Inc. VMware. http://www.vmware.com.
18. LibGuestFS. http://www.libguestfs.org/.
19. Bellard F. QEMU. http://wiki.qemu.org.
20. Kim S, Park J, Lee K, You I, Yim K. A Brief Survey on Rootkit Techniques in Malicious Codes. Journal of Internet Services and Information Security, 2012;3(4):134-147.
21. AIbrahim AS, Hamlyn-Harris J, Grundy J, Almorsy M. CloudSec: A Security Monitoring Appliance for Virtual Machines in the IaaS Cloud Model. In Internal Conference on Network and System Security (NSS), 2011.
22. Carbone M, Cui W, Lu L, Lee W. Mapping kernel objects to enable systematic integrity checking. In ACM conference on Computer and Communications Security (CCS), 2009.

5/26/2014