Secure Communication Mechanism in Xen Hypervisor

Thant Zin Tun
University of Computer Studies, Yangon
thantzintunster@gmail.com

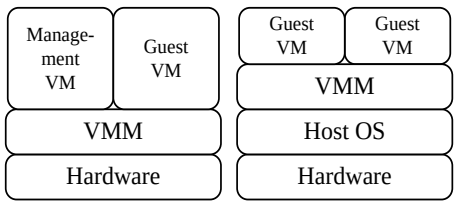
Abstract

In virtualization architectures, a small trusted computing base (TCB) minimizes the attacks which could be dangerous the security of the entire system. The TCB for an application includes the hardware, the virtual machine monitor (VMM) and the whole management operating system (OS). The management OS contains not only the device drivers but also virtual machine (VM) management functionality. This management OS is not acceptable to trust for many applications due to its high privilege level. In this paper, the secure virtual machine execution mechanism under the assumption of untrusted management OS (Dom0 in Xen) is proposed. This mechanism provides the confidentiality and integrity of the guest VM (DomUs in Xen). A secure communication channel is also proposed to exchange the secure information between Dom0 and the DomUs by authenticating and avoiding replay attacks.

1. Introduction

Virtualization is a rapidly evolving technology that can be used to provide a range of benefits to computing systems, including improved resource utilization, software portability and reliability. This abstracts the physical resources of a computing platform into many separate logical resources or computing environments. The virtualization environment allows users to create, copy, save, read, modify, share, migrate and roll back the execution state of VMs, which trims administrative overhead and makes system administration and management easier. However, the easier management also gives rise to security concerns. If the management environment is compromised, all the VMs can be easily copied and modified.

There are two basic types of virtualization architectures, as shown in figure 1. In Type I virtualization architectures, the virtual machine monitor (VMM) is just above the hardware and intercepts all the communications between the VMs and the hardware. There is a management VM on top of the VMM, which manages other guest VMs, and is responsible for most communications with the hardware. A popular instance of this type of virtualization architecture is the Xen system [10]. In Type II virtualization architectures, such as VMware Player, the VMM runs as an application within the host operating system (OS). The host OS is responsible for providing I/O drivers and managing the guest VMs.



(a) Type I (b) Type II

Figure 1. Types of Virtualization Architecture

From a security point of view, both architectures raise the question. The VM execution environment, which may be either malicious, or susceptible to vulnerability exploits. Type I virtualization architecture is mainly focused on in this paper and Xen hypervisor is used for demonstration to ensure the data confidentiality and integrity of a VMs execution.

The Xen architecture is composed of the hypervisor, the virtual machines called unprivileged domains (DomU), and a privileged virtual machine called Domain 0 (Dom0). The Xen hypervisor controls the physical resource accesses and handles the I/O operations performed by the domains [8]. Since Dom0 is a driver domain, it stores all physical devices. In addition, Dom0 is also the management interface between the administrator and the hypervisor to create virtual machines, modify Xen parameters and manage Xen operation. Attacks form the management domain can easily bypass the security mechanisms present in guest VMs due to the higher privilege level of the management VM.

The rest of this paper is organized as follows. In the next section, the related work is discussed. In section 3, this paper classifies the security threats of virtual machines under an untrusted Dom0. And then the detail description of the secure execution mechanism and secure communication channel for Xen hypervisor is described in section 4. It will be followed by conclusion in section 5.

2. Related Work

The security of virtualization itself is a significant concern. Various virtualization architectures propose different mechanisms to provide the TCB, which is everything in the system that provides a secure environment.

In [13], security challenges in virtual environments are summarized by T. Garfinkel et al. Virtualization can be utilized to enhance security. A lot of research studies [14] [1] [9], [5] utilize virtualization to implement introspection from the secure domain to the target domain.

Terra [12] is a virtualization architecture that allows many VM that have different security requirements to run independently without the threat of interference from each other.

D. G. Murray [2] et al. introduced a disaggregate domain builder in a Xen-based system. They illustrated how this approach may be used to implement “trusted virtualization” and improve the security of virtual TPM implementations.

Fagui LIU et al. proposed a new method of building a bridge firewall based on Xen and developing independent extension of firewall for special purpose and to supervise the host and guest system to enhance the computer system security [3].

VSITE, a scalable and secure architecture for seamless L2 enterprise extension in the cloud was proposed by Li Erran Li and Thomas Woo [6]. VISITE achieves abstraction through the use of VPN technologies, the assignment of different VLANs to different enterprises and the encoding of enterprise IDs in MAC addresses.

J. Oberheide et al. defined and investigated three classes of threats to virtual machine migration [4]. They also showed how a malicious party using these attack strategies can exploit the latest versions of the popular Xen and VMware virtual machine monitors and present a tool to automate the manipulation of a guest operating system’s memory during a live virtual machine migration.

3. Security Threats of Virtual Machines under an Untursted Dom0

The Xen hypervisor sits between the OS and the hardware. The hypervisor, OS kernel and user applications are three software layers in a Xen virtualization system. The mechanism used for inter-VM communication is shared memory, which can be established through either the grant table or foreign mapping.

The management OS (Dom0 in the Xen architecture) can directly map memory pages from other domains into its own address space, which is called foreign mapping. This mapping can only be made by Dom0. During several management operations, such as domain building, saving and restoring, this mapping mechanism is used.

The most important task performed by Dom0 is to handle hardware devices. The device drivers are normally located in Dom0. Another role played but Dom0 in the Xen architecture is the task of VM management. However, in an untrusted Dom0, these tasks must be supervised

in order to ensure the integrity and confidentiality of the guest VM [7].

Suppose a client is running a guest VM on the remote virtualized computing platform provided by a cloud computing company. The computation in the VM is security-critical, and involves confidential data of an enterprise and/or personal sensitive information. The untrusted management domain, i.e., Dom0 in Xen, is capable of undermining the confidentiality, integrity and the availability of a DomU.

- Confidentiality: Dom0 may access any memory page of DomU and read its contents. Also, Dom0 contains the device drivers for I/O devices such as the network card and hard disk, which endangers the privacy of the data transmitted through the network and the data stored on the hard disk.
- Integrity: For the same reason, Dom0 may access any memory page of Dom U and change its contents, as well as modify the data transmitted through the network and the data stored on the hard disk.
- Availability: Dom0 has the privilege to start and shutdown the other domains, and thus controls the availability of all guest VMs and the applications that execute within them.

Even if the administrator in Dom0 is not malicious, some malicious software installed in Dom0, which has root privileges, or a hacker who exploits some vulnerabilities in the management OS, or even someone who, by chance, has the image file of the memory contents of DomUs, can easily break into DomUs and extract all the secret information from the disk.

DomUs can also affect each other or Dom0, due to malicious actions and system faults. There may be replay attacks in which the opponent domain repeats old control messages to spoil information in the data plane of the attacked domain.

4. Secure Communication Mechanism

As mention in the previous section, for a remote computing VM under an untrusted Dom0, DomU should have the following four aspects to obtain secure execution environment.

- A secure network interface between the client and the server
- A secure secondary storage
- A secure run-time mechanism and
- A secure communication channel

Although there are solutions for secure network interface and secure secondary storage, a secure run-time mechanism is the most fundamental among the above aspects.

The secure-run time mechanism includes secure vCPU context and secure memory, ensuring both confidentiality and integrity. Dom0 must not be allowed to access the sensitive information in the vCPU registers and the memory of the security-critical VM. However, the management of these resources by Dom0 is also necessary. This paper mainly focuses on creating the mechanism for Dom0 to manage the DomUs without knowing their contents. Moreover, a secure communication channel between Dom0 and DomUs is designed, providing authentication and privacy in data transfer.

4.1. Domain Save and Restore process in Xen

The Xen hypervisor is requested by the guest domain (DomU) by invoking hypercalls [11]. During the domain execution, Dom0 uses the hypercalls to communicate with DomUs as shown in figure 2. The proposed architecture is focused on those hypercalls that are potentially harmful to the confidentiality and integrity of DomU memory content and vCPU context.

Some hypercalls that are used for foreign mapping and getting or setting vCPU context, can harm DomU. For those hypercalls, Dom0 mainly uses the domain save or restore operations.

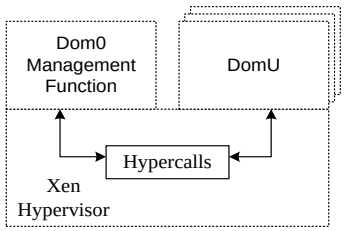


Figure 2. Hypercall between Dom0 and DomU

In the domain save and restore process, machine address and physical address are defined first. Machine address is the real host address, which can be understood by the physical processor. Physical address is for each VM. There is a physical-to-machine (p2m) mapping table stored in each VM, and a machine-to-physical (m2p) mapping table stored in the hypervisor layer.

In the domain save process, Dom0 suspends the VM, and maps the p2m table into its own memory space. An image file to store the memory and vCPU state of DomU is then created. After saving the p2m table in this image file, Dom0 repeatedly maps and saves each of the memory pages of the VM. Dom0 unmaps every memory page after saving its contents in the image file, then makes a hypercall to get the vCPU context and saves it in the same image file. Finally, the original VM resumes execution.

In the domain restore process, Dom0 is responsible for loading the image file and allocating the new memory area (through the use of memory allocation hypercalls). Then, Dom0 maps each page of the newly allocated memory, reads the contents of the image file, and writes every page back to memory. After loading and setting the vCPU context, Dom0 is now ready to launch the new VM.

4.2. Design of Secure Run-time Mechanism

DomU initiates the granting and Dom0 asks for access through hypercalls. During the execution of functions in which foreign mapping has to be used, the memory page mappings are monitored and controlled by the hypervisor layer.

During the domain save process, the memory page and vCPU context of DomU are totally exposed to Dom0 in plain text. Therefore, in the case of untrusted Dom0, secure mechanism must be taken to protect the confidentiality and integrity of DomU.

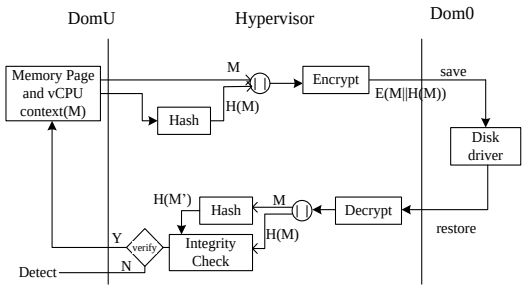


Figure 3. Secure Run-time of Domain Save and Restore Process

In the secure VMs execution design, the encryption and hash check is added in the domain save process as shown in figure 3. Encryption uses AES-192 in CBC mode, and hashing uses SHA-512. Before the memory pages and vCPU context are handed over to Dom0, the hypervisor calculates hash and encrypts them plus concatenated hash code, then save in the Dom0. During the domain restore process the hypervisor decrypts the entire pages plus hash code, then calculate the hash of the pages and verify the hash.

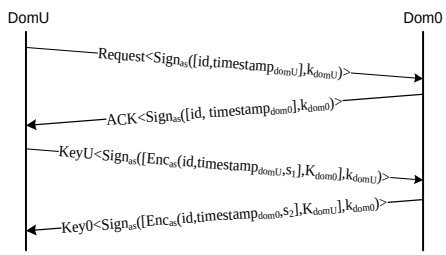
The Dom0 can only see the encrypted view of the memory pages and vCPU context and any malicious modification can be detected by the DomU. Confidentiality is also provided as encryption is applied to the entire pages plus hash code. Using this mechanism, DomU is protected from the untrusted domain Dom0, while Dom0 can still carry out the normal domain administrative tasks, such as domain build, domain save and domain restore.

By using the encryption and decryption process and hash check, the overhead may seem significant, however, note that domain build only occurs once in the whole life cycle of DomU and

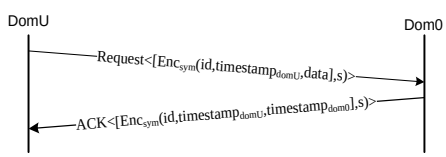
domain save/restore occur only when Dom0 needs to back up the state of DomU. These events may have a frequency of once an hour or several hours, even once a day. Hence, the overall overhead for the proposed protection architecture may be quite acceptable.

4.3. Design of Secure Communication Module

Secure communication module creates a secure communication channel between the Dom0 and the DomUs, providing mutual authentication and privacy in data transfer. Mutual authentication is required to ensure that no opponent domain can forge the identity of a common domain or of Dom0 to generate spoiled information in the data plane that corresponds to the attacked domain.



(a) Establishment of a session key



(b) Message exchange mechanism

Figure 4. Creating the Secure Channel between Dom0 and DomU

The secure communication module is composed of two mechanisms: one based on asymmetric cryptography for exchanging session keys, as described in figure. 4(a), and other based on symmetric cryptography for securely transmitting data between a DomU and Dom0, as described in figure. 4(b). The symbols used in the Figures are listed in table 1.

Table 1. Symbolic Description

Symbol	Description
k	Private key
K	Public key
Enc_{sym}	Encryption for symmetric cryptography
Enc_{as}	Encryption for asymmetric cryptography
$Sign_{as}$	Signature for asymmetric cryptography
id	Source node identity
s_1,s_2,s	Session keys

During the session key exchanging, DomU sends a request to initiate the establishment. It sends the id and timestamp by the signature with its private key k_{domU}. Dom0 sends the acknowledgement for the request by its signature. Then DomU sends the KeyU, which is the establishment of session key by encrypting with Dom0's public key using the asymmetric key cryptography. In this case, the encrypted session key is also signature by the DomU's private key. The Dom0 also sends its session key as the same mechanism of the DomU.

During the message exchange mechanism, DomU sends the encrypted data by using the symmetric key cryptography. The data and its timestamp and id are encrypted by the session key. Then Dom0 also sends back the acknowledgement by encrypting both timestamps and its id with session key.

These mechanisms avoid replay attacks, in which the opponent domain repeats old control messages to spoil information in the data plane of the attacked domain. Hence, Dom0 and each DomU exchange timestamps during the establishment of the session key and then both know the difference between their clocks, T_{d0} and T_{du}, according to equation $\delta_{est} = |T_{d0} - T_{du}|$. Thus, whenever an update message is sent, the source inserts its current timestamp into the message to the destination for checking whether the message is a replay. If so, the timestamp does not satisfy

$$|T_{d0n} - T_{dun}| - e_{max} < \delta_{est} < |T_{d0n} - T_{dun}| + e_{max}$$

where e_{max} is the maximum transmission delay and T_{d0n} and T_{dun} are the current timestamps of Dom0 and DomU. Therefore, the communication between Dom0 and DomU is secure because it checks the authenticity of the data and prevents replay attacks.

5. Conclusion

A virtualization architecture to ensure a secure VMs execution mechanism under an untrusted privileged OS is proposed. By using this mechanism, the secure execution between Dom0 and DomUs is achieved. DomU is protected from the untrusted domain Dom0 by encrypting and using hash function in the domain save and restore process, while Dom0 can still carry out the normal domain administrative tasks. Secure communication module is then designed, which is used the symmetric and asymmetric cryptography to ensure the security of communication channel between Dom0 and DomU. This channel provides authentication and avoids replay attacks in message exchange of Dom0 and DomU.

References

[1] B. Payne, M. Carbone, M. Sharif, and W. Lee, "Lares: An architecture for secure active monitoring using virtualization." In *Proc. IEEE Symp. Security and Privacy*, 2008.

[2] D. G. Murray, G. Milos, S. Hand, "Improving Xen Security through Disaggregation." *ACM*, 2008.

[3] Fagui LIU, Xiang SU, Wenqian LIU, Ming SHI, "The Design and Application of Xen-based Host System Firewall and its Extension." *In International Conference on Electronic Computer Technology*, 2009.

[4] J. Oberheide, E. Cooke, F. Jahanian, " Empirical Exploitation of Live Virtual Machine Migration."

[5] J.Yang and K. G. Shin, "Using hypervisor to provide data secrecy for user applications on a per-page basis," in *Proc. ACM Int. Conf. Virtual Execution Environments*, 2008.

[6] L. Erran Li, T. Woo, "VSITE: a scalable and secure architecture for seamless L2 enterprise extension in the cloud."

[7] L. Singaravelu, C. Pu, H. Hartig, C. Helmuth, "Reducing TCB Complexity for Security-Wensitive Applications: Three Case Studes." *Proc. ACM*, 2008

[8] M. Ben-Yehuda, J. Mason, O. Krieger, J. Xenidis, L. vanDoorn, A. Mallick, J. Nakajima, and E. Wahlig, "Utilizing IOMMUs for virtualization in Linux and Xen," in *Proc. Ottawa Linux Symp*., 2006.

[9] N. L. Petroni, Jr. and M. Hicks, "Automated detection of persistent kernel control-flow attacks." In *Proc. ACM Conf. Computer and Communications Security*, 2007.

[10] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield, " Xen and the art of virtualization," *in Proc, ACM Symp. Operating systems Principles, no.5, Oct.* 2003.

[11] R. Wojtczuk, "Subverting the Xen Hypervisor," in *Proc. BLACK HAT USA*, 2008.

[12] T. Garfinkel, B. Pfaff, J. Chow, M. Rosenblum, and D. Boneh, "Terra: A Virtual Machine-based Platform for Trusted Computing," in *Proc, ACM Symp. Operating Systems Principles*, 2003.

[13] T. Garfinkel and M.Rosenblum, "When virtual is harder than real: Security challenges in virtual machine based computing environment," in *Proc. Conf. Hot Topics in Operating systems*, 2005.

[14] X. Jiang, X. Wang, and D.Xu, "Stealthy malware detection through VMM-based "out-of-the-box" semantic view reconstruction," in *Proc. ACM Conf. Computer and Communications Security*, 2007.