Isolated Mini-domain for Trusted Cloud Computing

Jaewon Choi
Computer Science
KAIST
Daejeon, Korea
jwchoi@calab.kaist.ac.kr

Jongse Park
Computer Science
KAIST
Daejeon, Korea
jspark@calab.kaist.ac.kr

Jinho Seol
Computer Science
KAIST
Daejeon, Korea
jhseol@calab.kaist.ac.kr

Seungryoul Maeng
Computer Science
KAIST
Daejeon, Korea
maeng@calab.kaist.ac.kr

***Abstract*—On the cloud system, guest domains for cloud customers can be attacked by one of administrators with privilege or remote hackers who can compromise management tools. Therefore, the customers need a guarantee that their domains run on the secure environment with a protection against them. In this paper, we examine the security issues incurred by I/O model of hypervisors with a management domain, and propose an isolated mini-domain to protect the guest domains under the untrustworthy environment by addressing those issues.**

***Keywords*-cloud; trusted computing; virtualization**

I. Introduction

Recently, many people are using cloud services to utilize computing resources due to a plethora of advantages such as low costs, scalability, and flexibility. However, some people are still reluctant to use the cloud service since there is no guarantee that cloud computing environment is trusted. One of possible threats towards the guest domains for cloud customers is an attack from one of administrators with privilege or remote hackers who can compromise management tools. Since they can have the highest privilege in the system, they are able to access and leak the guest domains' data.

In general, the providers use virtualization technique to facilitate resource management, and the technique allows a malicious provider or remote hackers to leak sensitive data more easily than unvirtualized systems. In the virtualized systems, a hypervisor creates, destroys, manages all the guest domains, and it can access memory spaces and stored images of guest domains. Thus, the guest domains can be compromised in the environment where there is no guarantee that the hypervisor is trusted.

When a conventional hypervisor is used, the administrators can attain memory contents of the guest domains [3]. Therefore, earlier studies proposed the trusted hypervisor capable of protecting confidentiality and integrity of guest domains from the administrators [1]. However, these works cannot be directly applied to the cloud system since the administrators are able to run guest domains on untrustworthy hypervisors when launching or migrating them.

To address the problem, Trusted Cloud Computing Platform (TCCP) leverages Trusted Third Party (TTP), called Certificate Authority (CA), to authenticate trusted hypervisors which guarantee the confidentiality and integrity of guest domains [2]. To communicate securely between cloud providers and the CA, TCCP uses Public Key Infrastructure (PKI), and both the providers and CA exchange their public keys via hardware devices.

Most of currently used hypervisors have a special domain which is in charge of the management of devices. As hypervisors rely on the domain for device drivers, the hypervisors don't have to directly access devices. This model makes hypervisors as small as possible. Thus, modern hypervisors such as Hyper-V and Xen adopt the model. In this paper, we solve the problem incurred by the I/O model of those hypervisors.

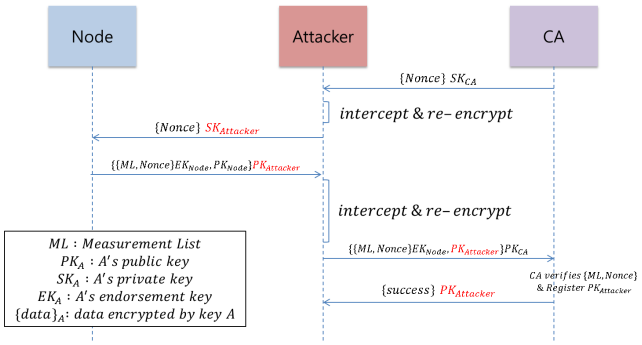

Figure 1. Man-in-the-middle-attack between the node and certificate authority(CA). The message between the node and CA could be intercepted by the attacker.

II. Motivation

In most hypervisors currently used, a management domain processes all I/O operations instead of hypervisors. To perform the node authentication protocol securely, the management domain need to be included in Trusted Computing Base (TCB), but it is too risky since one of cloud administrators is able to manipulate the management domain. Therefore, the hypervisor cannot trust the CA's public key derived from device drivers of the management domain.

For a trusted computing environment on cloud, Trusted Cloud Computing Platform (TCCP) [2] proposed an authentication protocol that all nodes need to be registered to the CA first to create or migrate guest domains. The CA authenticates each node using the remote attestation mechanism of Trusted Platform Module (TPM) [4]. When the CA finishes to authenticate each node successfully, the node would store the private key identifying the trusted hypervisor, and the CA would store the node's public key. However, a malicious administrator is able to replace the CA's public key with his one since he can control the management domain processing all I/O operations. Once he replaces the CA's public key with his one, he can intercept all messages exchanged between the node and CA, and it is called *man-in-the-middle attack*.

Fig. 1 shows an example of the man-in-the-middle attack. At first, the CA sends a random number nonce to the node by signing

it with the CA's private key. The attacker intercepts the message, and sends the same nonce to the node by signing it with the attacker's private key. The node may think that it comes from the valid CA since it matches with the public key received from hardware devices. The node might generate a key pair, and send a hash of software stacks and the node's public key encrypted by the CA's public key (in fact, attacker's one) using *quote* operation of the TPM. Again, the attacker could replace the node's public key with the attacker's one. Finally, the CA verifies the result of *quote*, and if the verification is successful, the CA might store the node's public key (in fact, attacker's one). It means that the CA believes that the attacker is credible.

III. Architecture

A. Isolated Mini-domain

1) Secure I/O: The isolated mini-domain provides the way to securely access hardware devices under a management domain compromised by an administrator. In case of most currently used hypervisors, the management domain should process all I/O operations since it controls hardware devices shared by all guest domains. However, we cannot put the management domain into TCB easily for the following two reasons. At first, it contains the large code base written for device drivers, virtual machine management tools, and user level processes; thus, it could be readily vulnerable to a variety of threats. Moreover, it is possible for one of administrator to leak guest domains' data, using the compromised management domain. Therefore, we cannot guarantee the integrity of the data derived from the device drivers of the management domain. For instance, if the management domain is compromised, the domain could modify the CA's public key; hence, the authentication protocol may malfunction.

To guarantee the integrity of data exchanged between hardware devices and the node authentication protocol, we design an isolate mini-domain that has a privilege to access hardware device directly. Thus, in this system, the management domain cannot intercept the exchanging data.

2) Authentication Protocol: In the TCCP prototype, the trusted hypervisor performs the node authentication protocol. The protocol requires cryptographic functions and other libraries implemented in the hypervisor. It is undesirable that the hypervisor has too many functions because the hypervisor runs with the highest privilege on the system. The separation of the node authentication protocol from the privileged layers can make the virtualization system more secure. Therefore, we separate the node authentication protocol from the hypervisor to the mini-domain which runs with lower privilege than the hypervisor to reduce attack surfaces, bugs, and vulnerabilities of the hypervisor. Fig. 2 shows the architecture of the mini-domain in the cloud system.

Node authentication protocol confines guest domains to be executed only on trusted hypervisors capable of protecting the guest domains from the compromised management domain. Each node in the cloud infrastructure runs the trusted hypervisor as well as the isolated mini-domain performing the node authentication protocol. The mini-domain performs the protocol when the confidentiality and integrity of the guest domains need to be guaranteed.

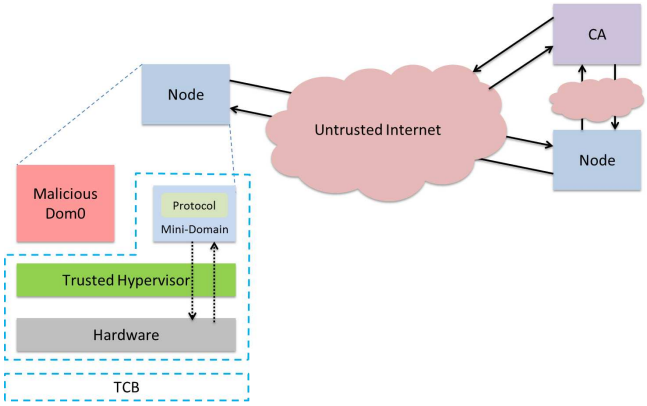

Figure 2. Architecture of the mini-domain and TCB. The TCB of our system excludes the untrusted management domain, and includes the mini-domain and the trusted hypervisor.

B. Integrity of Mini-domain

As the mini-domain is a main component for confidentiality and stores keys, the mini-domain should be included in TCB. Therefore, we need to ensure that the integrity of mini-domain. To protect an initial integrity of the mini-domain, the hypervisor checks the hash of the mini-domain's kernel, and loads it into the separate memory area from the untrustworthy management domain.

IV. Conclusion

We propose an isolated mini-domain which can access hardware devices directly, and perform the node authentication protocol. In our proposed architecture, the management domain including device drivers, domain management tools, and all other user level processes are completely excluded from the TCB. Instead, the TCB of our system consists of the trusted hypervisor and an isolated mini-domain. Therefore, even the management domain is compromised by a malicious administrator, important secrets (e.g. CA's public key) can be protected in the proposed design.

Acknowledgments

This work was supported by the IT R&D Program of MKE/KEIT. [KI002090, Development of Technology Base forTrustworthy Computing]

References

[1] Derek G. Murray, Grzegorz Milos and Steven Hand. Improving Xen Security through Disaggregation. In *Proceedings of the 4th ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments. (VEE)*, pages 151–160, 2008.

[2] Nuno Santos, Krishna P. Gummadi and Rodrigo Rodrigues. Towards Trusted Cloud Computing. In *Workshop On Hot Topics in Cloud Computing. (HotCloud)*, 2009.

[3] B. D. Payne, M. D. de Carbone, and W. Lee. Secure and Flexible Monitoring of Virtual Machines In *Computer Security Applications Conference. (ACSAC)*, pages 385-397, 2007.

[4] "Trusted Platform Module" http://www.trustedcomputinggroup.org/developers/trusted_platform_module