TSINGHUA SCIENCE AND TECHNOLOGY
ISSN 1007-0214 03/10 pp469-477
Volume 22, Number 5, October 2017

Trusted Attestation Architecture on an Infrastructure-as-a-Service

Xin Jin, Xingshu Chen*, Cheng Zhao, and Dandan Zhao

Abstract: Trusted attestation is the main obstruction preventing large-scale promotion of cloud computing. How to extend a trusted relationship from a single physical node to an Infrastructure-as-a-Service (IaaS) platform is a problem that must be solved. The IaaS platform provides the Virtual Machine (VM), and the Trusted VM, equipped with a virtual Trusted Platform Module (vTPM), is the foundation of the trusted IaaS platform. We propose a multi-dimensional trusted attestation architecture that can collect and verify trusted attestation information from the computing nodes, and manage the information centrally on a cloud management platform. The architecture verifies the IaaS's trusted attestation by apprising the VM, Hypervisor, and host Operating System's (OS) trusted status. The theory and the technology roadmap were introduced, and the key technologies were analyzed. The key technologies include dynamic measurement of the Hypervisor at the process level, the protection of vTPM instances, the reinforcement of Hypervisor security, and the verification of the IaaS trusted attestation. A prototype was deployed to verify the feasibility of the system. The advantages of the prototype system were compared with the Open CIT (Intel Cloud attestation solution). A performance analysis experiment was performed on computing nodes and the results show that the performance loss is within an acceptable range.

Key words: dynamic measurement; trusted cloud; vTPM; trusted attestation

1 Introduction

The promotion and effective use of cloud services depend on the credibility of cloud computing[1, 2]. The root of the trust problem in cloud computing is that users lose the ability to directly control the data and the operating environment. The users data may be threatened in two ways, on the one hand it may suffer theft and destruction by the cloud providers[3], on the other hand it may be threatened by other cloud users because of cloud computing shared resources[4]. User identity and other mandatory access control measures can mitigate the risk of data leakage to a certain extent[5], but cannot fundamentally fix the cloud platform credibility problem.

Virtualization technology is one of the key technologies in cloud computing, where the traditional host is replaced by a Virtual Machine (VM)[6]. It effectively improves the use of resources, reducing the cost of management, however it also presents a growing number of security risks such as improper isolation among VMs, VM escape, VM hijacking, and dynamic changes in VM network topology. These security factors make the VM security issue more complex than in a traditional host. The VENOM vulnerability appeared in 2015 enabling VM escape from theory to reality[7].

As a type of system security technology, Trusted Computing[8] has played a large role in solving the security problems of traditional computer systems. Using Trusted Computing technology to solve the credibility problems of the cloud has become a trend in Trusted Cloud development. We allocated a virtual trusted root (a virtual Trusted Platform Module,

• Xin Jin, Xingshu Chen, Cheng Zhao, and Dandan Zhao are with the College of Computer Science, Sichuan University, Chengdu 610065, China. E-mail: xinjin_cn@163.com; chenxsh@scu.edu.cn; zcscuit@foxmail.com; zddzhaodandan@qq.com.

∗ To whom correspondence should be addressed.
Manuscript received: 2016-10-01; accepted: 2016-10-21

vTPM[9]) to a VM, using virtual trusted computing technology to solve the credibility problem of VM. However, how to protect a running Hypervisor has now become an urgent problem.

This paper presents a trusted attestation architecture for cloud platform. In this architecture, the platform administrator can monitor the realtime trusted status of the host Operating System (OS), the Hypervisor, and specific VMs in trusted computing nodes. To ensure the security of the cloud platform, the administrator can take appropriate action to deal with any alerts.

The rest of the paper is organized as follows. Section 2 introduces related background knowledge. In Section 3, we demonstrate the theory of trusted attestation architecture. In Section 4, we analyze the key technologies involved in the architecture. Section 5 shows analysis of our design and Intel Open CIT technology[10]. In Section 6, we analyze the application prospects for the architecture.

2 Background and Related Work

A Trusted Platform Module (TPM) is a small tamper proof hardware chip embedded in most recent motherboards and offers a trusted hardware root bound to a single, standalone device. The TPM is equipped with encryption keys that never leave it, which reduces the possibility of those keys being compromised. The module has the virtue of remote authentication and interacts with a symmetric key, which can be used for various cryptographic purposes, from the protection of network communications to data encryption. In the Infrastructure-as-a-Service (IaaS)[11] context, it ensures that only a remote resource, in which the user is communicating using the TCG protocol, can interact with the ciphered data.

Shen and Tong[12] addressed cloud computing security challenges by proposing a solution called the Trusted Computing Platform (TCP). A trusted cloud computing system is built using the TCP as the hardware for cloud computing, and it ensures privacy and trust. By design, the TPM offers a trusted hardware root bound to a single, standalone device. vTPM is the virtualization of the hardware TPM, and here we allocate a vTPM to a VM and used it as the trusted root.

Kernel Virtual Machine (KVM)[13] is a Linux kernel module that allows a user space program to utilize the hardware virtualization features of the processors. Based on the QEMU-KVM project[14], we used KVM technology to virtualize the CPU and memory, and used QEMU technology to virtualize the I/O device. The best performance and addition of certain features were achieved using KVM with QEMU on x86 platform. We use KVM+QEMU+TPM to build a Trusted Cloud base environment.

The concept of Trusted Cloud is defined in Ref. [15]. The trusted attestation for a cloud platform should include three aspects: VM, Hypervisor, and host. However, to date most cloud platform, security research has focused on VM data safety or communication policy safety. The latest research results may be found in Refs. [16, 17].

Dynamic measurement, as opposed to static measurement, can guarantee the credibility of a particular application process. It is difficult to detect the operating behavior of a particular process. There are a variety of dynamic measurement methods to solve this problem. The two research directions of dynamic measurement are software behavior attestation[18] and property-based attestation[19]. These methods did not exhaust all the operation modes and there were undetected behaviors. Periodic attestation has been used in many research works[20], but it is not aware of attack behavior if the attack finishes between testing cycles.

To solve these problems, this study applies a dynamic measurement mechanism based on the switching process of the Hypervisor. When switching to the QEMU process, it measures the integrity of the KVM. When the integrity of the KVM has been destroyed, it immediately writes a system alert log for the system administrator to deal with.

3 Trusted Attestation Architecture on IaaS

3.1 Goals of the architecture

The goals of the trusted attestation architecture presented here are: (1) to provide a flexible distributed cloud architecture that can attest to the trusted credibility of the VM, Hypervisor, and host in the cloud, (2) to provide a trusted attestation mechanism for different roles (Platform Manager, Tenant Manager, and Terminal User) at different levels, and (3) to provide a process-level dynamic measurement mechanism that can seamlessly monitor the Hypervisor.

3.2 Architecture overview

Figure 1 shows an overview of the trusted attestation

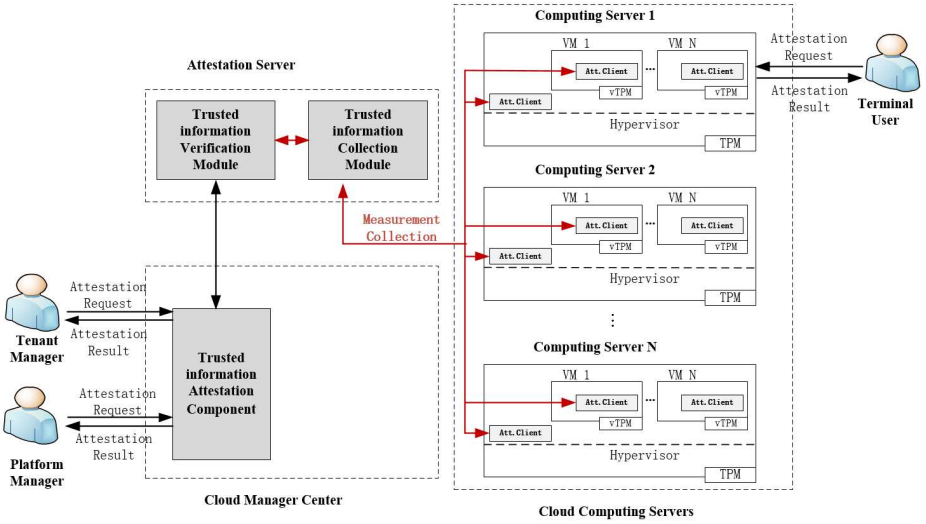

Fig. 1 Trusted attestation architecture on IaaS.

architecture. It comprises six entities: (1) Platform Manager, (2) Tenant Manager, (3) Terminal User, (4) Cloud Manager Center, (5) Attestation Server, and (6) Cloud Computing Servers.

3.2.1 Platform Manager

This is the administrator of the cloud platform. He can monitor the running status of the Hypervisors and Hosts. He can invoke an attestation request at any time, and can view the trusted information on the Hypervisors and Hosts collected by the attestation client.

3.2.2 Tenant Managers

These are the administrators of the tenants. There is one platform Manager and many Tenant Managers. They can monitor the running status of the VM for the tenant. The VM trusted attestation includes the VM trusted status and the trusted status of the infrastructure that the VM is running on. He can invoke an attestation request at any time to the VM the tenant holds, and he can also view the tenant VMs trusted information collected by the attestation client.

3.2.3 Terminal User

This is a real person who uses the VM. He can use a VM just like a physical machine. He can run a specific business system on a VM. He can invoke a local attestation to the VM when the attestation client was deployed in it.

3.2.4 Cloud Manager Center

This is the interface for the Platform and Tenant Managers to use and manage the cloud service. We deployed a Trusted Information Attestation Component into the Cloud Manager Center. It collects and verifies the attestation information for the cloud.

3.2.5 Attestation Server

This acts as the attestation requester and appraiser. It consists of two essential modules: (1) The Trusted Information Collection Module is responsible for validating measurement requests, interpreting orders, and making attestation decisions. (2) The Trusted Information Verification Module is responsible for verifying the collection of Trusted Information by comparing it with historical trusted attestation data in the library.

3.2.6 Cloud Computing Server

This is the computer that runs the VMs. The attestation client is deployed in the host operating system and VM to collect the Trusted Information. The Trusted Information is the hash value in the Platform Configuration Register (PCR). The PCR values respect the integrity status of the OS and dynamic applications (such as Hypervisor).

3.3 Hypervisor dynamic measurement mechanism

Figure 2 shows the Hypervisor dynamic measurement mechanism. There is a PCR bank in TPM, which was used to save the integrity information of the Cloud Computing server. There are 24 registers in the PCR Bank, each with a different use. PCR1–PCR8 were used to store OS critical bootloader information, for example, BIOS, MBR, and Bootloader. PCR10 was used to store the trusted chain for the OS. We used PCR17 and PCR18 to save the Hypervisor measurement information. These PCRs values were collected by an attestation client to verify the integrity of the

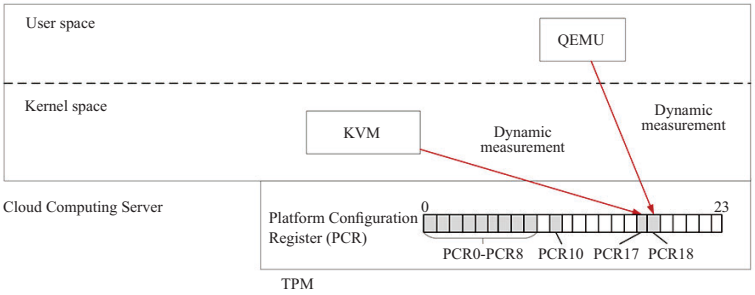

Fig. 2 Hypervisor dynamic measurement mechanism.

Hypervisor and the host OS.

The QEMU-KVM module was used as the Hypervisor in the architecture. The KVM module was deployed in kernel space, while QEMU processes ran in the user space. Therefore, we measured KVM and QEMU using different methods (Fig. 2). The measured value of KVM was extended to PCR17, and the measured value of QEMU was extended to PCR18.

4 Key Technologies

4.1 Dynamic KVM measurement

In a cloud computing server, the Hypervisor (KVM) is located between the VM and the physical hardware. It is responsible for managing the VM and its virtual hardware resources. If malware has destroyed the integrity of the Hypervisor (KVM) by tampering with the code, then it would attack other VMs and steal user data. Therefore, it is necessary to measure Hypervisor (KVM) code integrity.

Figure 3 shows the KVM dynamic measurement mechanism. The roadmap is as follow:

(1) Research the process switch function __schedule() in the Linux kernel source, insert measuring code after completing the memory context and hardware context switch process (①).

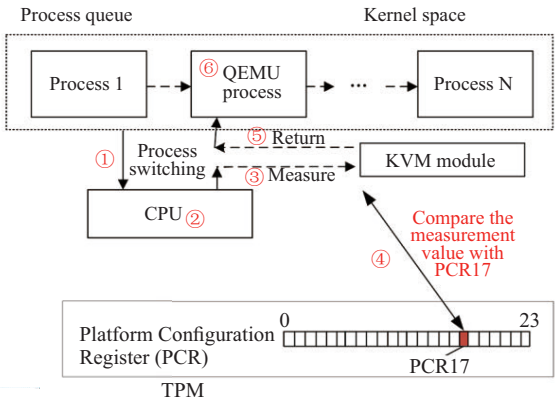

Fig. 3 KVM dynamic measurement mechanism.

(2) At a measuring point, the process queue is analyzed (②), when the current process is QEMU, get the address information of the KVM kernel module core code segment.

(3) According to the address information, the SHA1 algorithm is adopted to compute the value of the Hypervisor (KVM) code measurement (③).

(4) Compare the measurement value with history measurement values extended in PCR17 (baseline values), if they match, then the KVM code has not been damaged during the measuring period (④). If the KVM code segments are destroyed, then extend the new measurement value to the PCR register. After collection and verification of the integrity information, the platform manager receives this information and takes appropriate action.

(5) Return to CPU process queue and continue to run QEMU process (⑤ and ⑥).

When the KVM code segment was measured at the process level, the CPU performance degraded large extent. This was because the process switching frequently and KVM code segment SHA1 process take a longer time. To solve this problem, we used an optimized scheme (Fig. 4). It did not perform the SHA1 process when process switching to QEMU, we just used binary date segment to compare a certain number of times, e. g., 100, between SHA1 operations.

If the current process is QEMU, when the content switching is finished but it's not yet time to carry out the SHA1 operation, we do the following:

(1) Based on the salt value (a random number) that initialized at system setup, skip the salt size bytes in the KVM code address. Then copy a certain number of times, e. g., 20, the binary isometric code segment, e. g., a byte, to form a new data (①).

(2) Compare the new data with the history baseline value (②). If they match, the KVM segment is not

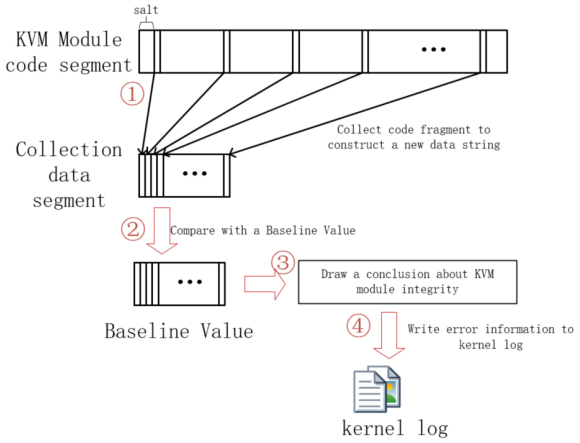

Fig. 4 Code segment data extraction and comparing.

compromised, so continue running the QEMU process (③).

(3) If the data values do not match, the KVM segment has been compromised. Write a log to the kernel log and continue running the QEMU process (④).

4.2 Dynamic QEMU measurement

QEMU runs as an application process in user space and a QEMU process represents a VM instance. When the VM needs to interact with the virtual IO device, the QEMU code segment is measured dynamically. Figure 5 shows QEMU dynamic measurement mechanism:

(1) When VM needs to use the virtual IO device, it will send a request to KVM (①). KVM sends a message to QEMU, request it do the IO job (②).

(2) QEMU invokes a sys_ioctl system call to finish the IO device emulation job (③). The system call is intercepted to measure the QEMU code segment (④ and ⑤).

(3) The measurement value is compared with the history measurement value extended in PCR18 (baseline value). If they match, the QEMU code has not been damaged during the measuring period (⑥). If the QEMU code segment has been destroyed, extend the new measurement value to the PCR register. After collection and verification of the integrity information, the platform manager receives this information and takes appropriate action.

(4) Return the virtual IO device and continue to implement the remaining tasks (⑦, ⑧, ⑨, and ⑩).

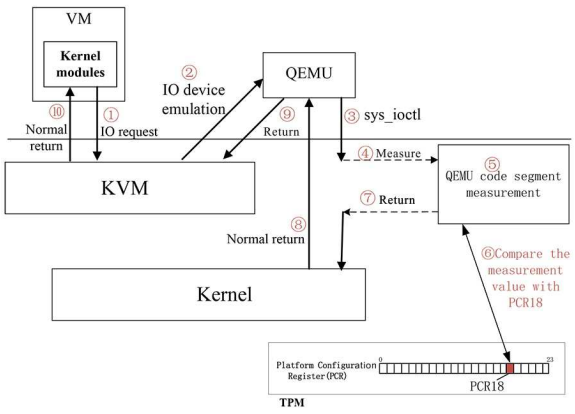

Fig. 5 QEMU dynamic measurement mechanism.

4.3 vTPM protection

4.3.1 vTPM measurement list

When creating a VM instance, the vTPM instance is established accordingly and the one-to-one association of the VM-vTPM instances is maintained by the management center. When starting the VM, the management center checks if the loading vTPM instance is the correct one for that VM association. If the answer is correct, then loading proceeds. If not, then the loading process is stopped and an alert with a warning message is raised. Similarly, when deleting a VM, the vTPM instance associated with the VM will also be deleted.

To guarantee the credibility of the vTPM, a **vTPM measurement list** was established to maintain the association of hardware TPM and vTPM instance. Then, a vTPM instance can gain trusted credibility based on hardware TPM.

Figure 6 shows the vTPM measurement list mechanism and Fig. 7 describes the vTPM measurement list initialization process. The measurement information on the vTPM instance is stored in the vTPM measurement list. The vTPM

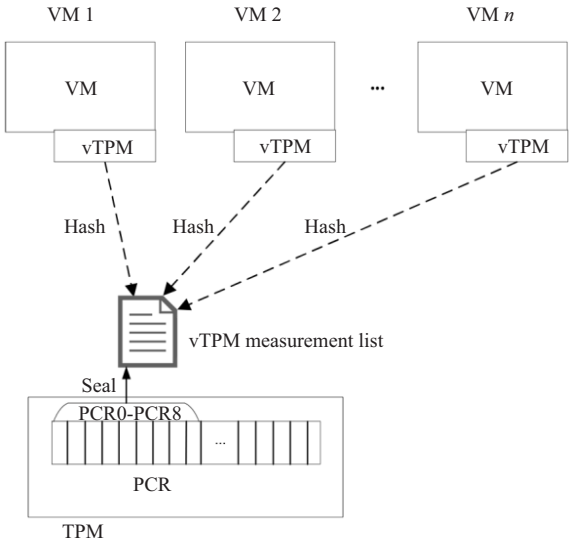

Fig. 6 vTPM measurement list.

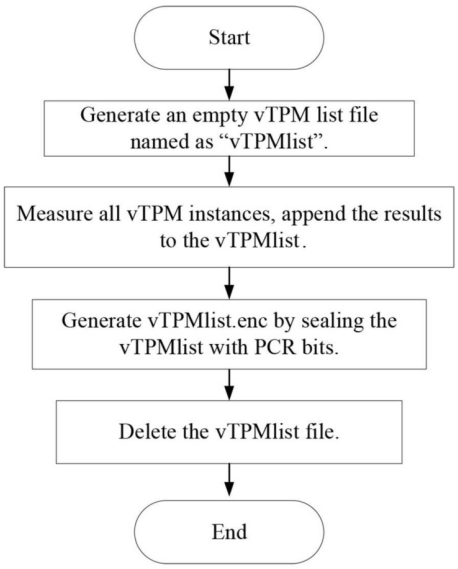

Fig. 7 vTPM measurement list initialization process.

measurement list is usually encrypted and here it was sealed by PCR0–PCR8. Only when the PCR values are as same as the values when sealed the measurement list, can the vTPM measurement list be unsealed and the measurement list data be updated. The unseal operation reflects that the credibility of the host OS is not damaged.

4.3.2 vTPM lifecycle protection

Figure 8 shows the vTPM lifecycle protection mechanism. The following tasks are performed:

(1) Store the vTPM instance in an encrypted status (①).

(2) When loading the trusted VM (②), check the association of the VM-vTPM and measure the integrity of the vTPM instance (③).

(3) If the vTPM instance measurement value matches the value in the vTPM measurement list, continue to load the VM with vTPM (④).

(4) When the VM instance is running, Hypervisor dynamically measures the guest OS (⑤).

(5) When VM exits, update the vTPM integrity information to the vTPM measurement list (⑥). Then reencrypt the vTPM instance to keep its status be encrypted (⑦).

(6) When the VM performed Trusted Migration, the associated vTPM instance was also migrated (⑧).

(7) If the VM is destroyed, the associated vTPM is also destroyed and the vTPM measurement information is also erased (⑨).

4.4 Hypervisor security reinforcement

This section describes the technologies used to enhance the security and trusted credibility of the Hypervisor (KVM). Considering some of the security threats in Hypervisor, it is important to establish a protective mechanism that can make the KVM safe and available. The main research contents include how to hide Hypervisor types, how to monitor the VMX's privileged instructions executed in the guest's OS, and how to prevent guest information being detected.

4.4.1 Hiding hypervisor types

This technology was used to hide a real type of hypervisor. When attackers obtain a wrong hypervisor type, they go the wrong way to attack. It makes attack action more difficulty.

The virtualization type query interface of KVM

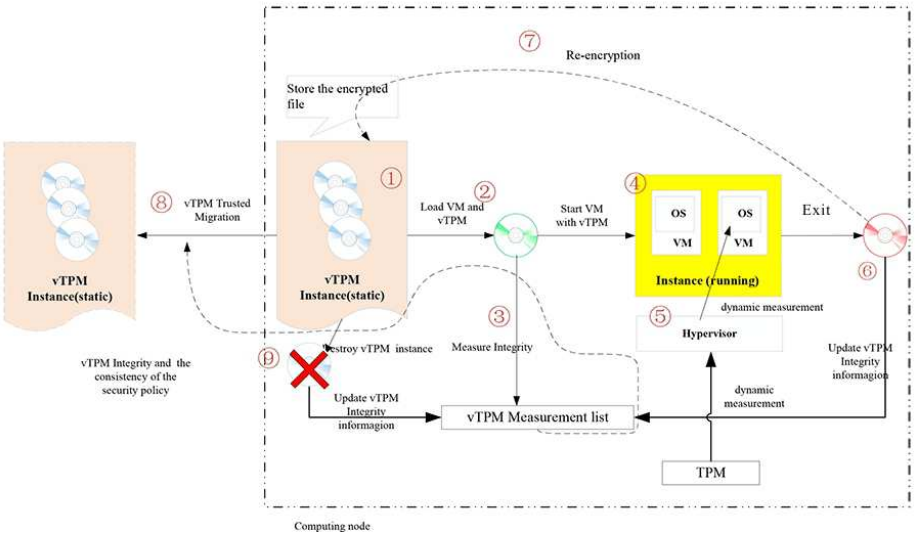

Fig. 8 vTPM lifecycle.

(CPUID instruction) was hooked-up to change the returned results on hypervisor type. It will return XEN when a VM attempts to call this interface to query the hypervisor type, which leads to a wrong type.

4.4.2 Monitoring VMX's privileged instructions

This technology was used to monitor and limit a guest to call on the VMX's privileged instructions provided by KVM. It prevents the Hypervisor's details being detected by guest OS and monitors illegal calls to the VMX's privileged instructions.

By investigating all VMX's privileged instructions and analyzing their call scenarios, invocation paths, and processing routines, we found that this technology could monitor these instructions and trace the source of the invocations. By deploying the monitoring module in KVM, we can detect any illegal VMs which call the VMX's privileged instructions. When unsafe operation occurs, the alert information is written to the KVM log. The platform manager then deals with this.

4.4.3 Protecting the host kernel

Where there is a VM escape, some security threats arise. These include as tampering with the structure of important data in the host's kernel (e.g., tampering with the syscall table), tampering with the KVM source code, and uninstalling the KVM kernel module without permission. By protecting the integrity of the syscall table in the host kernel, measuring the integrity of KVM, and preventing KVM from uninstalling, we can protect the integrity of Hypervisor and detect malicious attacks.

By dynamically checking the syscall table in the Host's kernel, we can protect the integrity of the important data and detect any illegal behavior.

By dynamically measuring the Hypervisor's code segment (see Section 4.1), we can protect its integrity and detect any illegal behavior.

By hiding the kernel module of KVM, the KVM module is protected from being uninstalled without permission.

4.4.4 Protecting the ioctl system call of the host

Here, we check the integrity of the ioctl system call's execution path in the Host while KVM interacts with QEMU. This prevents the execution path from being tampered or hooked by malware. It also aims at preventing the detection of other guests' data from detection by an illegal guest. With the code deployed in the KVM, we dynamically checked the integrity of the important execution paths.

4.5 Verifying the trusted attestation of the IaaS

After the above work, the next step was to collect the TPM and vTPM trusted information, and prepare the data to verify the trusted attestation of the Iaas platform (Fig. 9).

The TPM's PCR information was collected periodically. The Trusted Information Verification Module compares the collected values with the baseline values stored in attestation library. If the values match, the Iaas Platform infrastructure is in a healthy status, else it has been compromised.

The Platform Manager can monitor the trusted status of Hypervisors and Hosts. He can invoke an attestation request at any time, and can view the Trusted Information on the Hypervisors and Hosts collected by the attestation client.

The Tenant Manager can monitor the running status of the VM for the tenant. He can inspect the VM's trusted status and the trusted status of the infrastructure that the VM is running on. When the color column (with respect to the VM trusted status: green is nomal, red is abnomal) turns red, the VM trusted status or the infrastructure trusted status has been destroyed. When the VM is compromised, the Tenant Manager can isolate it and deal with the attacking action. When the infrastructure is compromised, the only work that the Tenant Manager can undertake is to tell the Platform Manager about the accident.

The Terminal User can invoke a local attestation to the VM when the attestation client was deployed.

5 Prototype System Practice and Analysis

We realized prototype system with open source software. Figure 10 shows the deployment of the

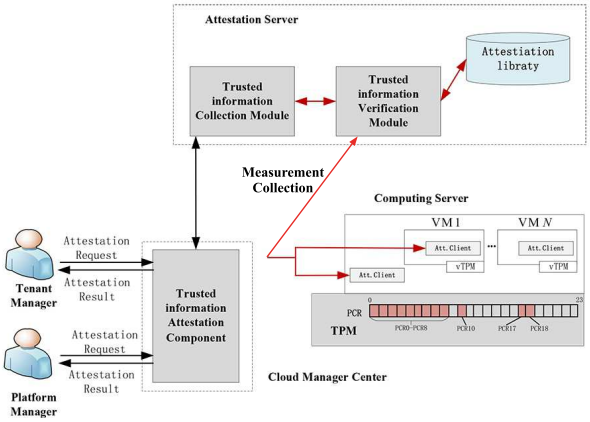

Fig. 9 Verifying the trusted attestation of the IaaS.

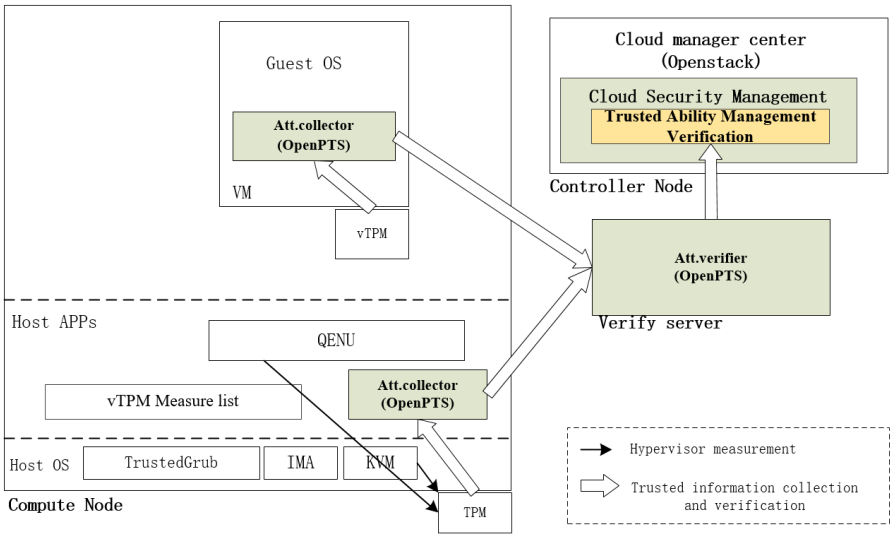

Fig. 10 Prototype system organization.

prototype system. TrustedGRUB and IMA software were used to detect the integrity of the system when it started. OpenPTS software was used as the attestation client in the Host and Guest OSs. These collected the trusted information (system setup, trusted chain, and dynamic measurements) and sent it to the Trusted Ability Management and Verification component in the cloud manager center. The cloud manager center was constructed using openstack software.

Table 1 shows a comparison of the prototype system and Open CIT software. Open CIT is the Intel attestation solution, it provides "Trust" visibility of the cloud infrastructure; If you want to use Open CIT, the hardware and software solutions must comply with Intel's requirements.

A performance analysis experiment was carried out on the computer nodes. Figure 11 shows the memory load latency when deploying the prototype system in computing node. The experimental results show that the performance loss was within an acceptable range.

Table 1 Comparison of prototype system and Open CIT software.

	Prototype system	Open CIT
Open source	Yes	Yes
KVM integrity	Yes	Yes
VM integrity	Yes	No
QEMU integrity	Yes	No
Hardware	Do not bind to Intel corporation	Bind to Intel corporation
Software cluster	TrustedGRUB, IMA, OpenPTS, Openstack	Tboot, Intel txt, Open CIT, Openstack

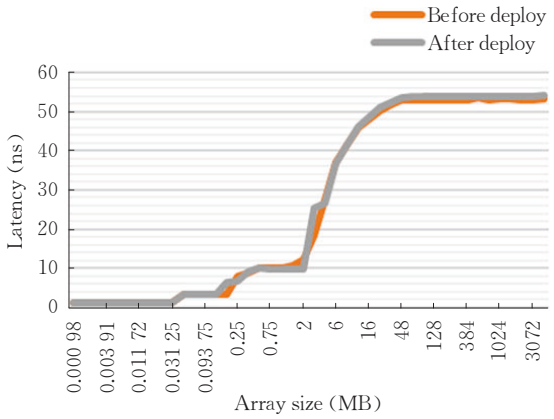

Fig. 11 Memory load latency when deploying the prototype system in computing node.

6 Conclusion and Prospects

The comparative analysis and experimental data show that our architecture has advantages compared with existing cloud platform attestation software. Our architecture is not bound to hardware, which makes it appropriate for more application scenario.

With the development of cloud computing, the Trusted Cloud will be a future development direction and will play an important role in cloud security. Trusted attestation is likely to become a hot topic in trusted cloud research.

Acknowledgment

This work was supported by the National Natural Science Foundation of China (No. 61272447).

References

[1] C. Rong, S. T. Nguyen, and M. G. Jaatun, Beyond

lightning: A survey on security challenges in cloud computing, *Computers & Electrical Engineering*, vol. 39, no. 3, pp. 47–54, 2013.
[2] M. D. Ryan, Cloud computing security: The scientific challenge, and a survey of solutions, *Journal of Systems and Software*, vol. 86, no. 9, pp. 2263–2268, 2013.
[3] W. Jansen and G. Timothy, Guidelines on security and privacy in public cloud computing, *NIST Special Publication*, vol. 800, no. 144, pp. 10–11, 2011.
[4] M. S. Inci, B. Glmezoglu, G. I. Apecechea, T. Eisenbarth, and B. Sunar, Seriously, get off my cloud! Cross-VM RSA key recovery in a public cloud, *IACR Cryptology ePrint Archive*, p. 898, 2015.
[5] E. Ghazizadeh, M. Zamani, J. L. Ab Mana, and M. Alizadeh, Trusted computing strengthens cloud authentication, *The Scientific World Journal*, vol. 2014, p. 260187, 2014.
[6] B. Bertholon, S. Varrette, and P. Bouvry, Certicloud: A novel TPM-based approach to ensure cloud IAAS security, in *2011 IEEE International Conference on Cloud Computing (CLOUD)*, 2011.
[7] W. Qiang, K. Zhang, W. Dai, and H. Jin, Secure cryptographic functions via virtualization-based outsourced computing, *Concurrency and Computation: Practice and Experience*, vol. 28, no. 11, pp. 3149–3163, 2015.
[8] N. Santos, K. P. Gummadi, and R. Rodrigues, Towards trusted cloud computing, *HotCloud*, vol. 9, p. 3, 2009.
[9] R. Perez, R. Sailer, and L. van Doorn, vTPM: Virtualizing the trusted platform module, in *Proc. 15th Conf. on USENIX Security Symposium*, 2006.
[10] Opencit, https://github.com/opencit/opencit/wiki/Open-CIT-2.0.7—Server-Product-Guide, 2016.
[11] L. Chen, X. Chen, J. Jiang, X. Yin, and G. Shao, Research and practice of dynamic network security architecture for IaaS platforms, *Tsinghua Science and Technology*, vol. 19, no. 5, pp. 496–507. 2014.
[12] Z. Shen and Q. Tong, The security of cloud computing system enabled by trusted computing technology, in *2010 2nd International Conference on Signal Processing Systems (ICSPS)*, 2010.
[13] A. Kivity, Y. Kamay, D. Laor, U. Lublin, and A. Liguori, KVM: The Linux virtual machine monitor, in *Proceedings of the Linux Symposium*, 2007.
[14] KVM, http://wiki.qemu.org/KVM, 2016.
[15] C. Sethi and S. K. Pradhan, Trusted-Cloud: A cloud security model for Infrastructure-as-a-Service (IaaS), *International Journal of Advanced Research in Computer Science and Software Engineering*, vol. 6, no. 3, pp. 32–46, 2016.
[16] T. Zhang and R. B. Lee, CloudMonatt: An architecture for security health monitoring and attestation of virtual machines in cloud computing, in *2015 ACM/IEEE 42nd Annual International Symposium on Computer Architecture (ISCA)*, 2015, pp. 362–374.
[17] U. Shanmugam and L. Tamilselvan, Dynamic resource monitoring of SaaS with attestation for a trusted cloud environment, *International Journal of Security and Its Applications*, vol.10, no. 4, pp. 41–50, 2016.
[18] A. Van Hoorn, J. Waller, and W. Hasselbring, Kieker: A framework for application performance monitoring and dynamic software analysis, in *Proceedings of the 3rd ACM/SPEC International Conference on Performance Engineering*, 2012, pp. 247–248.
[19] V. Varadharajan and U. Tupakula, Counteracting security attacks in virtual machines in the cloud using property based attestation, *Journal of Network and Computer Applications*, vol. 40, pp. 31–45, 2014.
[20] D. Contractor, D. Patel, and S. Patel, Trusted heartbeat framework for cloud computing, *Journal of Information Security*, vol. 7, no. 3, p.103, 2016.

Xin Jin is a PhD candidate at College of Computer Science of Sichuan University. He got the BE degree from Liaoning Shihua University in 1999 and MS degree from College of Software Engineering of Chongqing University in 2006. His research interests include trusted computing, virtuaization secutity, and cloud computing security.

Xingshu Chen received the PhD degree from Sichuan University in 2004. She is now a professor of the College of Computer Science and Cybersecurity Research Institute of Sichuan University. She is the member of China Information Security Standardization Technical Committee. Her research interests include cloud computing, cloud security, distributed file system, big data processing, network protocol analysis, and new media supervision.

Cheng Zhao is currently a graduate master in College of Computer Science of Sichuan University. He received the bachelor degree from Sichuan University in 2014. His current research focuses on hardware virtualization technology in cloud computing.

Dandan Zhao is currently a graduate master in College of Computer Science of Sichuan University. She received the bachelor degree from Sichuan University in 2014. Her research interests include operation system secutity and virtuaization secutity.