Secure IDS Offloading with Nested Virtualization in Untrusted Clouds

Shouhei Miyama
Kyushu Institute of Technology
miyama@ksl.ci.kyutech.ac.jp

Kenichi Kourai
Kyushu Institute of Technology
kourai@ci.kyutech.ac.jp

In Infrastructure-as-a-Service (IaaS) clouds, users run their systems in virtual machines (VMs). Since clouds suffer from attacks more frequently, intrusion detection systems (IDSes) are indispensable. To prevent IDSes from being disabled by intruders into VMs, *IDS offloading* with *VM introspection* has been proposed. This technique runs IDSes outside VMs and introspects the internals of VMs. However, cloud administrators are not always trustworthy. If some of the cloud administrators are malicious, they can easily disable offloaded IDSes and intrude into VMs without detection.

In such untrusted clouds, secure IDS offloading has been achieved by assuming the trusted hypervisor. Even if cloud administrators on the hypervisor attempt to disable offloaded IDSes, their access to the IDSes is prohibited. For example, IDSes can run in special VMs protected by the hypervisor in SSC. Under this assumption, however, average cloud administrators cannot manage the hypervisor because the integrity of the hypervisor has to be maintained. To prevent cloud administrators from disabling security mechanisms provided by the hypervisor, they cannot update the hypervisor. Therefore, it is necessary to give privileges for managing the hypervisor to only a few trusted cloud administrators.

To solve this problem, we propose V-Met, which enables offloading IDSes outside the entire virtualized system. V-Met uses *nested virtualization* to run the traditional virtualized system in a VM, as illustrated in Fig. 1. We call VMs and the hypervisor in an inner virtualized system *guest VMs* and the *guest hypervisor*. In contrast, we call those in an outer one *host VMs* and the *host hypervisor*. Thanks to this architecture, V-Met allows average cloud administrators to completely manage the inner virtualized system including the hypervisor. In addition, it can prevent them from attacking IDSes running in the outer virtualized system.

V-Met directly introspects the memory of guest VMs without relying on the untrusted guest hypervisor. It finds a memory page corresponding to a requested virtual address in a guest VM and provides the page to offloaded IDSes. First, it traverses the page tables in a guest VM to translate virtual into guest physical addresses. The address of the page directory is stored in the CR3 register of a virtual CPU, which is maintained by the guest hypervisor. Since V-Met cannot trust the state of virtual CPUs in the guest hypervisor, it configures the host hypervisor so that VM exit occurs when a guest VM modifies the CR3 register. When the host hypervisor traps that VM exit, it obtains the value that the guest VM attempts to write to the register and saves it.

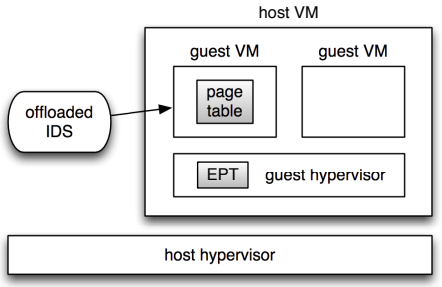

Figure 1. The system architecture of V-Met.

Second, V-Met traverses the extended page tables (EPT) in the guest hypervisor to translate guest physical into host physical addresses. The address of the EPT is stored in the VMCS of a virtual CPU, which is also maintained by the guest hypervisor. Without introspecting the guest hypervisor, the host hypervisor saves the host physical address of the VMCS when a guest VM causes VM exit. Although the VMCS and the EPT are managed by the untrusted guest hypervisor, they can be protected by the host hypervisor, as proposed in CloudVisor.

We have implemented V-Met in Xen 4.4 and ported Transcall [1], which provides an execution environment for offloading legacy IDSes. We measured the execution time of chkrootkit in V-Met and the traditional IDS offloading. Offloaded chkrootkit introspects a guest VM and obtains system information to detect rootkits. The virtual disk of the guest VM was shared with chkrootkit using NFS. According to this experiment, the execution time in V-Met was 25% longer.

References

[1] T. Iida and K. Kourai. Transcall. `http://www.ksl.ci.kyutech.ac.jp/oss/transcall/`.