VM Migration for Secure Out-of-band Remote Management with Nested Virtualization

Tomoya Unoki
Kyushu Institute of Technology
unoki@ksl.ci.kyutech.ac.jp

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

***Abstract*—Infrastructure-as-a-Service clouds provide out-of-band remote management of the systems in virtual machines (VMs). This management method enables users to manage their systems even on several types of failures inside VMs. In this method, users access virtual devices of their VMs, but virtual devices are not sufficiently protected against untrusted cloud operators. For secure out-of-band remote management, previous work securely runs *shadow devices* outside an untrusted virtualized system using *nested virtualization*. However, the states of shadow devices are lost during VM migration. In this paper, we propose *USShadow* for continuing secure out-of-band remote management after VM migration. USShadow enables the migration manager inside the virtualized system to transparently and securely save and restore the states of shadow devices outside it. We have implemented USShadow, which supports Xen and KVM as virtualized systems. Then, we confirmed that USShadow could continue virtual serial console and that the migration overhead was negligible.**

1. Introduction

In Infrastructure-as-a-Service (IaaS) clouds, users can use the necessary number of virtual machines (VMs) to construct large systems. To manage their systems in VMs provided by clouds, users access VMs from remote hosts by running remote management servers such as SSH inside target VMs. In addition to such a direct management method, clouds provide another method called *out-of-band* remote management. This method enables users to indirectly access VMs via their virtual devices located outside target VMs. One advantage of this method is that users can continue to manage the systems in VMs even on failures of their virtual networks and remote management servers. On the other hand, virtual devices are not sufficiently protected in current clouds. They are easily accessible to cloud operators, who perform daily cloud management. Unfortunately, cloud operators may be untrusted and even malicious [1]–[4].

To prevent the leakage of inputs and outputs of remote management from virtual devices, a system called VSBypass [5] has been proposed. VSBypass runs the entire virtualized system including VMs and their virtual devices in an outer VM using *nested virtualization* [6]. It intercepts access to virtual devices used for out-of-band remote management and forwards it to *shadow devices* running outside the virtualized system. Since all the inputs and outputs are securely handled in shadow devices, VSBypass can achieve secure out-of-band remote management against cloud operators, who are confined in the virtualized system. However, users cannot continue this secure out-of-band remote management with shadow devices after their VMs are migrated to other hosts. Since shadow devices run outside the virtualized system, the migration manager running inside it cannot save or restore their states. As a result, the states are lost during VM migration and cannot be restored until the migrated VM is rebooted.

This paper proposes *USShadow* for continuing secure out-of-band remote management with shadow devices after VM migration. USShadow enables the migration manager to transparently and securely transfer the states of shadow devices as well as the traditional ones of a VM. To avoid any modifications to the migration manager, USShadow runs *pseudo devices* in the virtualized system, each of which corresponds to a shadow device. The migration manager can transparently save and restore the state of a shadow device as that of a pseudo device. A pseudo device securely and efficiently communicates with the corresponding shadow device by completely bypassing the virtualized system. To prevent information leakage from the saved states, USShadow provides end-to-end encryption between shadow devices, which means that shadow devices themselves encrypt and decrypt their states.

We have implemented USShadow on top of VSBypass, which was developed on the basis of Xen 4.8. Currently, USShadow supports Xen and KVM as virtualized systems for running users' VMs. As shadow devices, it supports a virtual serial device used for virtual serial console. Using USShadow, we confirmed that secure out-of-band remote management could be continued after VM migration. In addition, our experimental results show that the increases in migration time and downtime were negligible.

The organization of this paper is as follows. Section 2 describes VSBypass and an issue in VM migration using it. Section 3 proposes USShadow, which enables out-of-band remote management with shadow devices after VM migration. Section 4 explains its implementation and Section 5 shows the performance of USShadow. Section 6 describes related work and Section 7 concludes this paper.

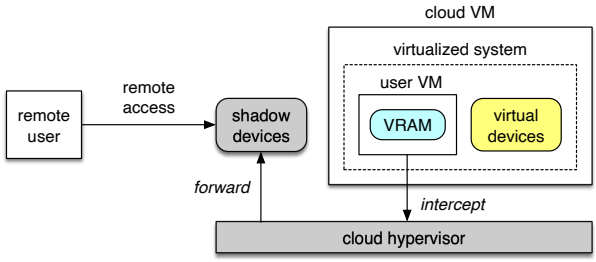


Figure 1: The system architecture of VSBypass.

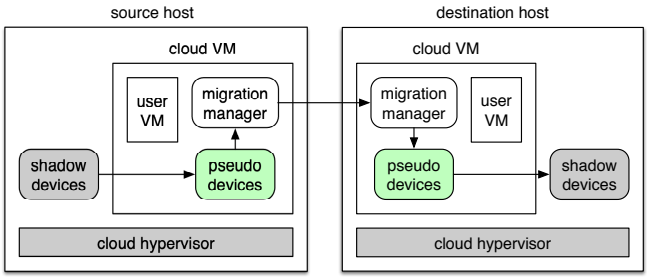


Figure 2: VM migration in USShadow.

2. Secure Out-of-band Remote Management

For secure out-of-band remote management, VSBypass [5] runs special devices called *shadow devices* outside the virtualized system including VMs and their virtual devices. It uses shadow devices for out-of-band remote management, instead of virtual devices. Remote users can access their VMs via shadow devices in a similar manner to the traditional system, while information leakage is prevented because cloud operators confined in the virtualized system cannot access shadow devices. To enable VMs inside the virtualized system to perform I/O using shadow devices outside it, VSBypass provides a mechanism called *transparent passthrough*. This mechanism intercepts I/O issued to virtual devices by VMs and transparently forwards it to shadow devices.

VSBypass assumes that the entire virtualized system is untrusted and that only the outside world is trusted. To implement such a system, it uses *nested virtualization* [6] and runs the virtualized system in a VM. Fig. 1 shows the system architecture of VSBypass. The virtualized system is run in an outer VM called the cloud VM. It runs multiple VMs of users, called user VMs, and their virtual devices. VSBypass runs one shadow device per virtual device outside the cloud VM. The cloud VM and shadow devices run on top of the hypervisor called the cloud hypervisor. When a user VM issues an I/O instruction to a virtual device, the cloud hypervisor intercepts that instruction. It emulates the instruction by accessing the shadow device corresponding to the target virtual device. For an input instruction, it returns an input value sent from a remote user to the shadow device as the result of the instruction execution. For an output one, it writes the specified output value to the shadow device and then the output is sent to a remote user.

However, secure out-of-band remote management with shadow devices stops functioning after VM migration. This is because the migration manager cannot transfer the states of shadow devices. The migration manager runs inside the virtualized system, whereas shadow devices run outside it. Since the migration manager can access only the inside of the virtualized system, it cannot save the states of shadow devices. Therefore, the states cannot be restored at the destination host although the traditional ones of the VM are restored. Due to this inconsistency between the states, neither the migrated VM nor a remote user can access shadow devices correctly.

In this paper, we assume that cloud operators who perform daily management may be untrusted. They have full control over the entire virtualized system. In contrast, we assume that cloud providers are trusted. They maintain the trusted computing base, e.g., hardware, the cloud hypervisor, and shadow devices, outside the virtualized system. This maintenance is done by a few trusted system administrators. We assume that these administrators are rewarded adequately and do not perform malicious activities. A few trusted administrators and many untrusted operators form an administrative hierarchy, which is adopted in many systems. For example, Oracle Database provides two types of privileges: SYSDBA for full administration and SYSOPER for basic operations [7]. IBM Domino restricts privileges to eight types of administrators [8].

3. USShadow

USShadow enables secure out-of-band remote management with shadow devices after VM migration. Unlike VSBypass, the migration manager inside the virtualized system can transfer the states of shadow devices outside it. Fig. 2 shows the overview of VM migration in USShadow. At the source host, the migration manager saves the states of shadow devices outside the virtualized system via special devices called *pseudo devices* inside the virtualized system. Then, it transfers the saved states to the migration manager at the destination host. After that, USShadow stops the shadow devices. At the destination host, USShadow creates new shadow devices at the starting time of VM migration. When the migration manager receives the states of the shadow devices, it restores the states of the new shadow devices using them.

Pseudo devices are used to transparently save and restore the states of shadow devices without any modifications to the migration managers. USShadow additionally runs them inside the virtualized system. When the migration manager attempts to save the state of a pseudo device like a normal virtual device, the pseudo device communicates with the corresponding shadow device and obtains its state. Similarly, when the migration manager attempts to restore the state of a pseudo device, the pseudo device restores the state of the corresponding shadow device. As such, the migration manager can handle the state of a shadow device as that of a pseudo device.

For secure and efficient communication between pseudo and shadow devices, USShadow enables shadow devices to share the memory of pseudo devices. First, a pseudo device invokes the cloud hypervisor using a mechanism called an ultracall [4]. This mechanism completely bypasses the virtualized system and directly transfers the control to the cloud hypervisor. Therefore, it is not affected by the overhead of the virtualized system. Also, it does not require any modifications to the existing virtualized system. After that, the cloud hypervisor invokes the corresponding shadow device. Next, the shadow device shares a buffer in the memory of the pseudo device using the function of the cloud hypervisor. Since it does not use its own buffer for the communication, it can prevent the buffer overflow attack. Using the shared buffer, the shadow and pseudo devices can efficiently exchange data.

In addition, USShadow provides end-to-end encryption between shadow devices in the source and destination hosts. When a shadow device sends and receives its state to and from a pseudo device, it encrypts and decrypts the state by itself, respectively. Since shadow devices run outside the virtualized system, this prevents information leakage from the states of shadow devices to untrusted cloud operators in the virtualized system. For example, the state may include a password typed by a remote user and sensitive information emitted by a user VM. At the source host, the migration manager obtains the encrypted states of shadow devices via pseudo devices and transfers it. At the destination host, when the migration manager passes the received states to shadow devices via pseudo device, the shadow devices decrypt the encrypted states and restore their own states.

4. Implementation

We have implemented USShadow by extending VSBypass, which is based on Xen 4.8.0. USShadow uses the modified hypervisor as the cloud hypervisor and runs the cloud management VM and the cloud VM on top of it. In addition, it creates a proxy VM whenever a user VM is booted. It runs shadow devices in the cloud management VM. In the cloud VM, it runs an existing virtualized system such as Xen and KVM. For Xen, the guest hypervisor, the guest management VM, and user VMs run in the virtualized system. In the guest management VM, the migration manager, virtual devices, and pseudo devices run. For KVM, the guest hypervisor runs inside the Linux kernel of the virtualized system. The migration manager and pseudo devices run on top of the kernel.

Shadow devices are implemented as virtual devices provided for proxy VMs. They are embedded into QEMU for emulating virtual devices of proxy VMs. Proxy VMs are assigned the minimum amounts of resources and are used only for providing shadow devices.

4.1. Pseudo Devices

A pseudo device is a kind of virtual device provided for a user VM. These devices are embedded into QEMU for emulating virtual devices of user VMs. They can be easily developed using several templates of virtual devices. They perform only the registration to QEMU and the forwarding of save and restore requests to the corresponding shadow devices. To save the state of a shadow device, a pseudo device shares a buffer for storing the state with the shadow device using an ultracall. The detailed mechanism of this memory sharing is described in Section 4.2. The shadow device stores its own state to this shared buffer, while the pseudo device receives it. The size of the state is returned as the return value of the ultracall. The migration manager obtains the state by saving that of the pseudo device and transfers it to the destination host.

At the destination host, the migration manager attempts to restore the state of a pseudo device using the received state. At this time, the pseudo device stores the state in a buffer and shares it with the corresponding shadow device using an ultracall. Then, the shadow device restores its state using the data stored in the shared buffer.

4.2. Communication with Shadow Devices

A pseudo device first invokes the cloud hypervisor using an ultracall. An ultracall is implemented by using the vmcall instruction, which is also used for invoking a hypercall to the guest hypervisor. If a pseudo device issues this instruction in the guest management VM, a VM exit directly occurs to the cloud hypervisor, not to the guest hypervisor. If this instruction is used for hypercall invocation, the cloud hypervisor transfers the control to the guest hypervisor. For an ultracall, it handles that instruction by itself. This ultracall takes the virtual address of a buffer for storing the state of a shadow device and its size as parameters. The buffer is an anonymous memory page allocated by the mmap system call and is pinned using the mlock system call to avoid page-out to swap space.

The cloud hypervisor translates the virtual address of this buffer so that a shadow device can access the memory of the pseudo device. First, it obtains the value of the CR3 register from the virtual CPU with which an ultracall is invoked. In the CR3 register, the address of the page directory is stored. Since CPUs are para-virtualized in the guest management VM, the cloud hypervisor can access the virtual CPU of the guest management VM via that of the cloud VM. It does not need to analyze the guest hypervisor to obtain the state of the virtual CPU in the guest management VM.

Next, the cloud hypervisor walks the page tables pointed by the page directory. Then, it translates the virtual address used in the QEMU process that runs the pseudo device into the physical one used in the cloud VM. Since memory is also para-virtualized in the guest management VM, the page tables directly contain physical addresses of the cloud VM. The cloud hypervisor does not need to walk the extended page tables (EPT) used for a fully virtualized VM.

The cloud hypervisor sends a special I/O request to the QEMU process in the cloud management VM and invokes the target shadow device. To distinguish this request from normal I/O requests issued by a proxy VM, it uses dedicated

I/O port numbers for the requests to shadow devices. This I/O request includes a port number, a request type (save or restore), the translated physical address of the buffer in the pseudo device, and the buffer size. The type of a shadow device is determined by the port number.

4.3. Saving and Restoring States

USShadow basically saves and restores the state of a shadow device using the interface for saving and restoring that of a virtual device, which is provided by QEMU. However, the traditional QEMU can save and restore the state only to and from a file or a network socket. This is because that interface is used to save the state of a VM to a disk and to transfer a VM to another host in VM migration. Therefore, our QEMU provides a new interface for saving and restoring the state of a shadow device to and from memory. Using this interface, a shadow device can read and write its own state from and to a buffer in a pseudo device.

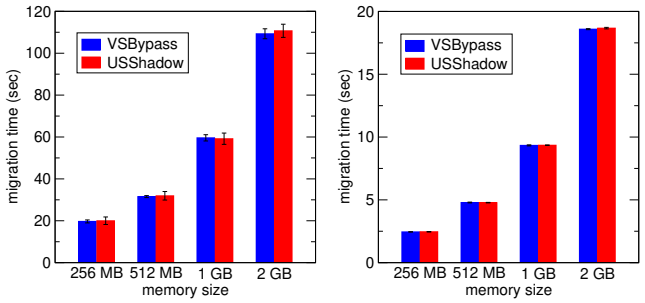
In the current implementation, USShadow saves and restores the state of a shadow serial device, which is used by virtual serial console. To save the state, a shadow device obtains the values of 10 device registers and stores them in the buffer of a pseudo device. At the same time, it encrypts the state using AES every 16 bytes. To restore the state at the destination host, a new shadow device decrypts the data in the buffer and sets the values to its own device registers. The encryption key is shared between the two shadow devices when VM migration is started.

5. Experiments

We conducted several experiments to show the effectiveness of USShadow. For comparison, we used VSBypass, which did not save or restore the states of shadow devices. For the source and destination hosts of VM migration, we used two PCs with an Intel Xeon E3-1226 v3 processor, 8 GB of memory, and Gigabit Ethernet. We assigned two virtual CPUs and 3 GB of memory to the cloud VM. For a proxy VM, we assigned two virtual CPUs and 1 GB of memory. We used Xen 4.4.0 and KVM 2.4.1 as virtualized systems running in the cloud VM. For a user VM, we assigned two virtual CPUs and various sizes of memory. For encryption in shadow devices, we used AES-ECB with a 128-bit key.

5.1. Remote Management after VM Migration

To examine that secure out-of-band remote management could be continued after VM migration, we migrated a user VM while using its virtual serial console. First, we logged in to the cloud management VM using SSH and connected to the virtual serial console using the xl console command in Xen. Then, we logged in to a user VM inside the cloud VM via this virtual serial console. The access to the virtual serial console was redirected to the user VM by transparent passthrough.



(a) Xen (b) KVM

Figure 3: The migration time in USShadow.

Next, we migrated the user VM to the cloud VM at the destination host. When we used Xen as a virtualized system, we executed the xl migrate command in the guest management VM. For KVM, we executed the virsh migrate command. After the VM migration, we logged in to the cloud management VM in the destination host using SSH again. In USShadow, the login session was preserved and we could continue to access the user VM using the virtual serial console. However, we could not continue the access in VSBypass.

5.2. Migration Performance

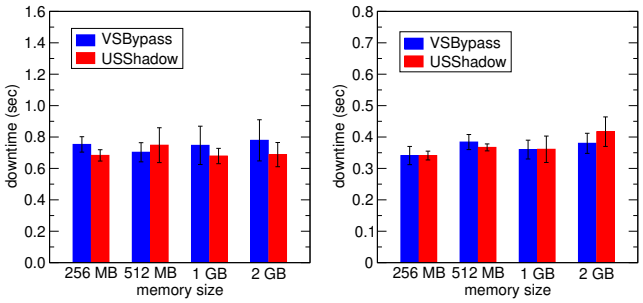
We examined the migration performance of a user VM. For USShadow, a shadow serial device was run and its state was transferred to the destination host. In VSBypass, the state was not transferred. We assigned memory between 256 MB and 2 GB to a user VM. We measured the migration time and the downtime. We executed VM migration 5 times for each memory size and calculated the average and the standard deviation.

Fig. 3(a) and Fig. 3(b) show the migration time when we used Xen and KVM as virtualized systems, respectively. Compared with VSBypass, the average increase in migration time was only 743 ms for Xen and 24 ms for KVM. The overhead was 1.5% at most. The difference between Xen and KVM came from the overhead of nested virtualization. Since the overhead in Xen was much larger than that in KVM, it took a much longer time to migrate a VM in Xen.

Fig. 4(a) and Fig. 4(b) show the downtime when we used Xen and KVM as virtualized systems, respectively. For KVM, the downtime increased only by 6 ms on average, compared with VSBypass. This is due to handling the state of the shadow device in USShadow. For Xen, in contrast, the downtime was reduced by 46 ms on average. This reason is under investigation, but the performance impact is not large in any case.

6. Related Work

A shadow device in USShadow is a kind of passthrough device of VMs. A passthrough device is a virtual device



(a) Xen (b) KVM

Figure 4: The downtime in USShadow.

that can directly access hardware to improve I/O performance. Similarly, a shadow device directly accesses virtual hardware outside the virtualized system using nested virtualization. Since passthrough devices are tightly coupled with hardware, it is difficult to migrate VMs with such devices.

To migrate VMs with passthrough NICs, the bonding drivers are used inside VMs [9]. This migration method bonds passthrough and paravirtual NICs. Since it hot-unplugs the passthrough NIC before VM migration, it can easily migrate the VM only with the paravirtual NIC. After VM migration, the method hot-plugs the passthrough NIC. The network plugin architecture (NPA) [10] can be also used for VM migration with passthrough NICs. Before VM migration, this migration method removes the plugin for a passthrough NIC and adds that for a paravirtual NIC. Then, it switches back to the plugin for a passthrough NIC after VM migration.

Transparent VM migration with passthrough NICs is enabled using shadow drivers [11]. A shadow driver records access to a network driver and redirects requests to a new network driver started at the destination host. CompSC [12] completely restores the values of the device registers of NICs after VM migration. Since this is difficult due to registers with side effects, CompSC records access to NICs at the source host and replays it at the destination host. SRVM [13] does not restore the complete states of NICs but does only the minimum states so that NICs work correctly. It detects memory regions to which NICs write received packets and forwards the packets to the destination host. Since shadow devices in USShadow are virtual devices, their states can be completely saved and restored.

To enable traditional out-of-band remote management to be continued after VM migration, D-MORE [14] has been proposed. It runs virtual devices for a VM in a migratable VM dedicated for remote management. It co-migrates that VM together with a user VM and keeps the states of virtual devices. However, if we apply this mechanism to secure out-of-band remote management with shadow devices, we would need a VM dedicated for running shadow devices outside the virtualized system. It is not secure to allow the migration manager inside the virtualized system to migrate that VM outside it with a user VM. In addition, the migration manager needs to be largely modified for the co-migration.

7. Conclusion

This paper proposed USShadow for continuing secure out-of-band remote management with shadow devices after VM migration. USShadow transparently and securely saves and restores the states of shadow devices via pseudo devices running inside the virtualized system. We have implemented USShadow in Xen and confirmed that secure out-of-band remote management was available after migration. Experimental results show that the migration overhead was small.

One of our future work is to support other virtualized systems running in the cloud VM. In the current implementation, USShadow supports Xen and KVM and assumes QEMU as a device emulator. It is necessary to confirm that USShadow is applicable to other types of virtualized systems, e.g., Hyper-V.

Acknowledgment

The research results have been achieved by the "Resilient Edge Cloud Designed Network (19304)," the Commissioned Research of National Institute of Information and Communications Technology (NICT), Japan.

References

[1] N. Santos, K. P. Gummadi, and R. Rodrigues. Towards Trusted Cloud Computing. In *Proc. Workshop Hot Topics in Cloud Computing*, 2009.

[2] F. Zhang, J. Chen, H. Chen, and B. Zang. CloudVisor: Retrofitting Protection of Virtual Machines in Multi-tenant Cloud with Nested Virtualization. In *Proc. Symp. Operating Systems Principles*, pages 203–216, 2011.

[3] S. Butt, H. A. Lagar-Cavilla, A. Srivastava, and V. Ganapathy. Self-service Cloud Computing. In *Proc. Conf. Computer and Communications Security*, pages 253–264, 2012.

[4] S. Miyama and K. Kourai. Secure IDS Offloading with Nested Virtualization and Deep VM Introspection. In *Proc. European Symp. Research in Computer Security*, pages 305–323, 2017.

[5] S. Futagami, T. Unoki, and K. Kourai. Secure Out-of-band Remote Management of Virtual Machines with Transparent Passthrough. In *Proc. Annual Computer Security Applications Conf.*, pages 430–440, 2018.

[6] M. Ben-Yehuda, M. D. Day, Z. Dubitzky, M. Factor, N. Har'El, A. Gordon, A. Liguori, O. Wasserman, and B.-A. Yassour. The Turtles Project: Design and Implementation of Nested Virtualization. In *Proc. Symp. Operating Systems Design and Implementation*, pages 423–436, 2010.

[7] Oracle Corp. Oracle Database 2 Day DBA.

[8] IBM Corp. IBM Domino 10.0.1 Documentation.

[9] E. Zhai, G. Cummings, and Y. Dong. Live Migration with Passthrough Device for Linux VM. In *Proc. Ottawa Linux Symp.*, pages 261–267, 2008.

[10] P. Thakkar. RFC: Network Plugin Architecture (NPA) for vmxnet3. http://lkml.iu.edu/hypermail/linux/kernel/1005.0/01142.html.

[11] A. Kadav and M. Swift. Live Migration of Direct-access Devices. *SIGOPS Oper. Syst. Rev.*, 43(3):95–104, 2009.

[12] Z. Pan, Y. Dong, Y. Chen, L. Zhang, and Z. Zhang. CompSC: Live Migration with Pass-through Devices. In *Proc. Int. Conf. Virtual Execution Environments*, pages 109–120, 2012.

[13] X. Xu and B. Davda. SRVM: Hypervisor Support for Live Migration with Passthrough SR-IOV Network Devices. In *Proc. Int. Conf. Virtual Execution Environments*, pages 65–77, 2016.

[14] S. Kawahara and K. Kourai. The Continuity of Out-of-band Remote Management across Virtual Machine Migration in Clouds. In *Proc. Int. Conf. Utility and Cloud Computing*, pages 176–185, 2014.