2015

Privacy Protection on Cloud Computing

Min Li
lim4@vcu.edu

PRIVACY PROTECTION ON CLOUD COMPUTING

A dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy at Virginia Commonwealth University.

by

MIN LI

M.S., Virginia Commonwealth University, USA - May 2011 to Aug 2012

B.S., Nankai University, China - Sep 2006 to May 2010

Director: Dr. Meng Yu,

Assoc. Professor, Department of Computer Science

Virginia Commonwewalth University

Richmond, Virginia

May, 2015

Acknowledgements

On the way to pursue a Ph.d. degree, I have received various invaluable help from a lot of people. I would never finish this work without the guidance of my committee members and support of my friends.

I would like to deeply appreciate my supervisor, Prof. Meng Yu for the continuous support of my Ph.D study and research. Without his excellent guidance, patience and caring, I can not complete my research in such an excellent atmosphere. Also I would like to express my gratitude to my co-advisor, Prof. Xubin He, who has been always there to give me advice. I really appreciate for his time and efforts to provide me valuable advice in my research and dissertation.

My thanks also goes to other committee members for their comments and feedback on my dissertation. They are Prof. Wanyu Zang, Prof. Wei Cheng and Prof. Thang Dinh in the Department of Computer Science; Prof Wei Zhang in the Department of Electric and Computer Engineer. I really appreciate the time they spend on reviewing my dissertations and giving me very helpful suggestions for improving the work.

I am grateful to all my coauthors for their helpful discussions and collaborations. To name but a few: Prof. Peng Liu from Pennsylvania State University and Dr. Kun Bai from IBM T.J.Watson Research Center. I thank all the members in Cyber Security Lab of Department of Computer Science for many discussions and valuable suggestions they offered.

Fianlly, I deeply appreciate the support from my wife, Meimei Meng. She is always cheering me up through good and bad times. Without her continuous support, none of this would have been possible.

My proposed work is supported by he National Science Foundation under Grant

No. NSF CNS-1100221, NSF CNS-1422355 and NSF IIP-1342664. I would like to thank all reviewers for their insightful comments on my publications.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Abstract

PRIVACY PROTECTION ON CLOUD COMPUTING

By Min Li

A dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy at Virginia Commonwealth University.

Virginia Commonwealth University, 2015.

Director: Dr. Meng Yu,

Assoc. Professor, Department of Computer Science

Cloud is becoming the most popular computing infrastructure because it can attract more and more traditional companies due to flexibility and cost-effectiveness. However, privacy concern is the major issues that prevent users from deploying on public clouds.

The root cause of privacy problem is current cloud privilege design that gives too much power to cloud providers. Once the control virtual machine (installed by cloud providers) is compromised, external adversaries will breach users' privacy. Malicious cloud administrators are also possible to disclose user's privacy by abusing the privilege of cloud providers. In this dissertation, I propose two cloud architectures – MyCloud and MyCloud SEP to protect user's privacy based on hardware virtualization technology. I eliminate the privilege of cloud providers by moving the control virtual machine (control VM) to the processor's non-root mode and only keep the privacy protection and performance crucial components in the Trust Computing Base (TCB). In addition, the new cloud platform can provide rich functionalities on resource management and allocation without greatly increasing the TCB size.

In MyCloud and MyCloud SEP, the hypervisor maintains an access control matrix to record users' configured policy for privacy protection. All resource accesses (e.g. memory and disks) will be checked by the hypervisor against access control matrix. I implement a prototype on x86 architecture and the performance evaluation results show acceptable overheads.

Besides the attacks to control VM, many external adversaries will compromise one guest VM or directly install a malicious guest VM, then target other legitimate guest VMs based on the connections. Thus, collocating with vulnerable virtual machines, or "bad neighbours" on the same physical server introduces additional security risks. I develop a migration based scenario that quantifies the security risk of each VM and generates virtual machine placement to minimize the security risks considering the connections among virtual machines. According to the experiment, our approach can improve the survivability of most VMs.

CHAPTER 1

INTRODUCTION

1.1 Introduction to Cloud Computing

Cloud computing is a comprehensive technology relying on Internet, hardware and software technology. It can offer users cheap, scalable and effective computing services. The cloud users buy shared cloud infrastructures and pay cloud providers as how much they use. Cloud computing allows companies focus on project and services that differentiate their businesses rather than infrastructures management.

In terms of service models, the cloud computing services can be categorized as infrastructure as a service (IaaS), platform as a service (PaaS) and software as a service (Saas). The IaaS cloud providers offer physical resources like storage, memory and network etc to cloud users. In IaaS cloud, users can scale services up and down according their demand. Cloud users can install operating system and application software on the cloud infrastructure. The cloud providers should manage and protect the OS and software deployed by users. In PaaS cloud, cloud providers can offer a computing platform including existed operating system, database, webserver etc. The cloud users can directly run their software or applications without purchasing the physical resources and deploying operating systems. In SaaS cloud, cloud providers have already installed software and applications. Users should buy the access to the "on-demand software" which simplifies the maintenance and support of cloud users to implement their business. In this dissertation, I will focus on solving the privacy issues on IaaS cloud model.

IaaS cloud providers will install a Hypervisor (e.g. Xen [1], KVM [2], VMware

ESXi [3]) and Microsoft [4] to support large numbers of guest virtual machines (VMs). The hypervisor is responsible to provide virtualization, VMs scheduling and system initialization. Cloud providers will also deploy a privileged virtual machine (Dom 0 or Host OS) to manage and allocate the cloud resources. In current Iaas cloud, both hypervisor and the privileged VM are running in the privileged mode because cloud providers should protect cloud resources from external adversaries and migrate the guest VMs in order to improve the performance of cloud resources.

Privacy concerns prevent many users especially financial institutions from deploying their business in cloud computing environment. External attackers are able to compromise one guest VM and target next guest VM connected with the compromised VM. In order to protect users privacy from this attack, I propose a migration based solution. My approach will predict the possibility of being attacked for each guest VM then generate a new migration plan to place the most "dangerous" VM in a dedicated physical machine. However, this approach only improves the possibility of guest VM's survivability in an attack rather than stop the attack from happening. In addition, this approach cannot protect user's privacy from inside attacks. Based on hardware technology, I propose two scenarios on privacy protection – MyCloud and MyCloud SEP . The cloud users only need to trust the hardware (e.g. CPU, motherboard etc,.) and verifiable hypervisor. In summary, the contributions in my dissertation are:

1. Propose a systematic approach to evaluate the security of VMs in cloud platform and generate a migration plan in order to improve the survival possibility of most guest VMs.

2. Design a new cloud architecture to protect users' privacy from both internal and external attacks.

3. Greatly reduce the size of TCB of the new cloud architecture by separating the resource management and security protection.

4. Allow cloud users participate in privacy protection in order to solve mutual distrust between cloud providers and cloud users.

The rest of this dissertation is organized as follows. Related work is in chapter 2. In chapter 3, we will discuss the migration based approach to increase the survivability of VMs. In chapter 4 and chapter 5, we will present hardware based security strategy - MyCloud and MyCloud SEP.

1.2 Motivation and Problem

The privacy issue in cloud computing is the biggest pain points for cloud users. By 2016, the user's concern on privacy will make cloud market lose 35 billion dollars [5]. In my research most threats on users' privacy can be categorized as external attacks and inside attacks.

External attackers can compromise the cloud platform via vulnerabilities of the control VM or the cloud hypervisor [6] [7] [8] [9] [10]. After taking over the privilege of cloud providers, the adversary is able to disclose user's privacy. MyCloud and MyCloud SEP can fight against this kind of attack by reducing the attack surface and moving the control VM to non-privileged mode. Another type of external attacker will deploy a malicious guest VM or compromise a legal guest VM in cloud platform, then try to compromise other guest VMs connected to the malicious/compromised VM via the vulnerabilities in the applications and services of guest VMs [11]. The migration based approach can place other legal VMs to a safe physical server before the external adversary completes an attack.

Inside attackers refer to the malicious cloud administrators who may misuse the

privilege of cloud providers and disclose users' privacy. Both MyCloud and MyCloud SEP can eliminate the privilege of cloud administrators. When cloud providers need to access users' privacy, they should grant the agreement of the owners of the privacy. Cloud users can manage their privacy throughout the interface provided by cloud hypervisor. Cloud hypervisor is responsible to intercept and check the permission of all resource access. The cloud providers in new architecture can only manage and allocate the cloud resource.

In general, there are many business model in cloud. For example, platform as a service (PaaS) and software as a service (SaaS). However, my work only focus on infrastructure as a service (IaaS). In IaaS cloud, users will buy the hardware resources and deploy the service by themselves. Cloud providers only guarantee the security and availability of cloud resources. Therefore, version control of users' data is out of scope of this work. In this work, our contribution is to provide the protection mechanism rather than design protection policy for guest VMs. Our goal is to allow cloud users manage privacy by themselves.

1.3 Intel Virtualization Technology

1.3.1 CPU Virtualization – Vt-x

Intel Vt-x technology (aka. VMX) can support processor virtualization for cloud design. The Vt-x technology can divide all operations in cloud as VMX non-root operation and VMX root operation. When the physical server is booted, the processors will stay in the VMX root mode. Cloud hypervisor can enable the VMX technology then jump to the non-root mode by VMLAUNCH. In MyCloud and MyCloud SEP, only the cloud hypervisor can execute root operations. All VMs including the control VM are running in the non-root mode. If any VMs execute one privileged operation

in non-root mode, A transition (VMEXIT) between VMX root mode and VMX non-root mode will happen. After the cloud hypervisor handles the VMEXIT transition, it can execute VMRESUME to reload the guest VM and return back to the VMX non-root mode.

Except the VMX operations (e.g. VMLAUNCH, VMPTRLD etc), other privileged CPU instructions are executed as they are in the bare-metal machine. The cloud hypervisor should initialize the whole system, manage physical memory and handle all interrupts, exceptions and VMEXITs in root mode. In order to provide a virtualized machine status and control the VMX transitions, Vt-x technology proposes a data structure called virtual-machine control structure (VMCS).

VMCS is composed by a 4KB physical page. The address of VMCS is referred to VMCS pointer. When the hypervisor tries to launch a guest VM, it should send VMCS pointer to a physical core. In Intel Vt-x technology, each physical core can be assigned only one active VMCS. The hypervisor should take responsibility to schedule the physical cores to execute different VMCS. The hypervisor can configure the VMCS through VMREAD, VMWRITE and VMCLEAR instructions. In VMCS the hypervisor can set up the initial register value for guest VM (e.g. CR0, CR3 etc,.) Also, the VMCS can indicate which privileged instruction of guest VM will cause the VMEXIT. For example, the hypervisor can claim any MOV CR instructions should be trapped in VMCS. When guest VM tries to modify the value of CR registers by MOV instructions, a VMEXIT will cause a transition from none-root mode to root mode. The hypervisor can also trap modifying other system register (e.g. MSR, GDTR, LDTR, IDTR etc,). Similarly, the hypervisor can also trap the I/O port read/write, interrupt and exception in configuring the VMCS. The basic layout of VMCS is as follows.

1. **Guest-state area**: The initial system state will be loaded into guest VM after VMLAUNCH or VMRESUME

2. **Host-state area**: The machine state when VMEXIT happens

3. **VM-execution control field**: The restriction of operations in guest VM

4. **VM-exit control field**: Control VM exit

5. **VM-entry control field**: Control VM entry

6. **VM-exit information field**: Restore the information which can describe the cause of VMEXIT.

Usually, the hypervisor uses VM-execution control field, VM-exit control field and VM-entry control field to control the instruction in guest VMs.

Figure 1 shows an overview of cloud architecture made by VMX technology. There are two guest VMs running in the non-root mode. Cloud hypervisor creates two different VMCS for each guest VM. In the opinion of cloud users, guest VMs are assigned two physical cores. The cloud hypervisor should initialize and launch the guest VM by VMLAUNCH and handle the VMEXIT in guest VMs.

We can also improve the cloud security by the Vt-x technology, because the Vt-x build a new privileged architecutre in cloud. The hypervisor could program the VMCS and CPU will return to the hypervisor when VMs execute privileged instructions. Therefore, the malicious VM is not able to compromise the hypervisor or other VMs if the hypervisor is capable to detect the malicious activities. The cloud developers can execute all privileges softwares, malware detection engine in the root mode. Other VMs including the privileged VM are moved to the non-root mode. Then each privileged instructions will be trapped by the CPU and analyzed by the

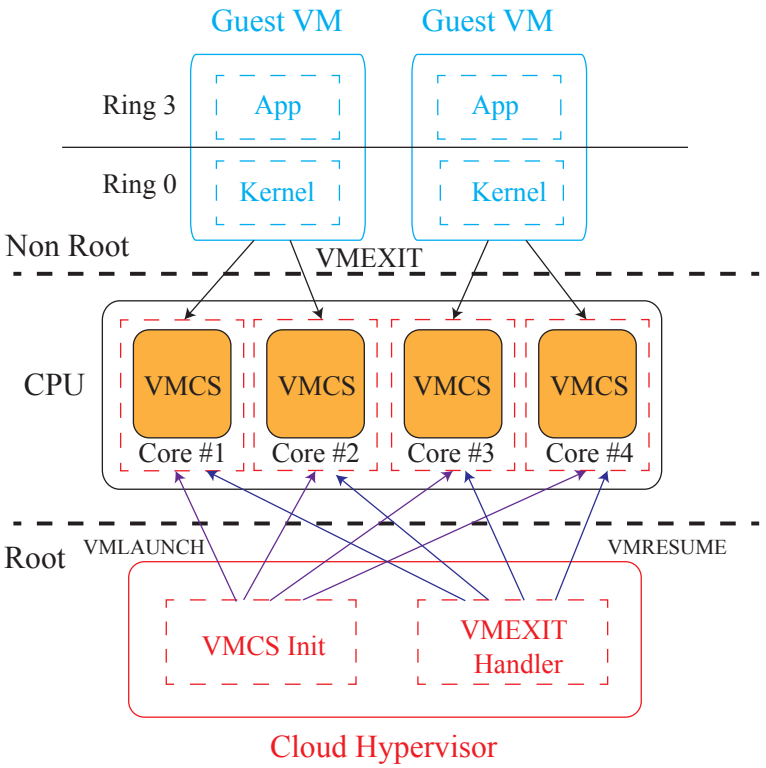


Fig. 1.: Intel VMX Overview [12]

hypervisor. Vt-x is a CPU feature, so we can assume the Vt-x can always trap the instructions as we program.

1.3.2 Memory Virtualization – EPT

The architecture of VMX operation supports the extended page-table mechanism (EPT) in order to implement memory virtualization and address translation. When EPT is in use, the addresses which would normally be translated as physical address are instead treated as guest physical address (GPA). The memory management unit (MMU) will further translate the guest physical address to host physical address (HPA) in EPT. In non-root mode, the operating system is responsible to translate the guest virtual address (GVA) to guest physical address (GPA).

In guest VM, memory access using guest physical address may cause VM exit due to *EPT misconfiguration* and *EPT violations*. An EPT misconfiguration will occur if any of the following cases is identified while MMU translates a guest physical address.

1. The EPT page is either write-only or write/execute only.
2. The EPT page is execute only but the processor does not support this capability.
3. The EPT page is present but the reserved bit is set.

An EPT violation will occur when there is no EPT misconfiguration but the EPT page disallows an access using the guest physical address due to the following reasons.

1. The EPT page is not present.
2. The access is data read but the EPT page does not allow read operation.
3. The access is data write but the EPT page does not allow write operation.

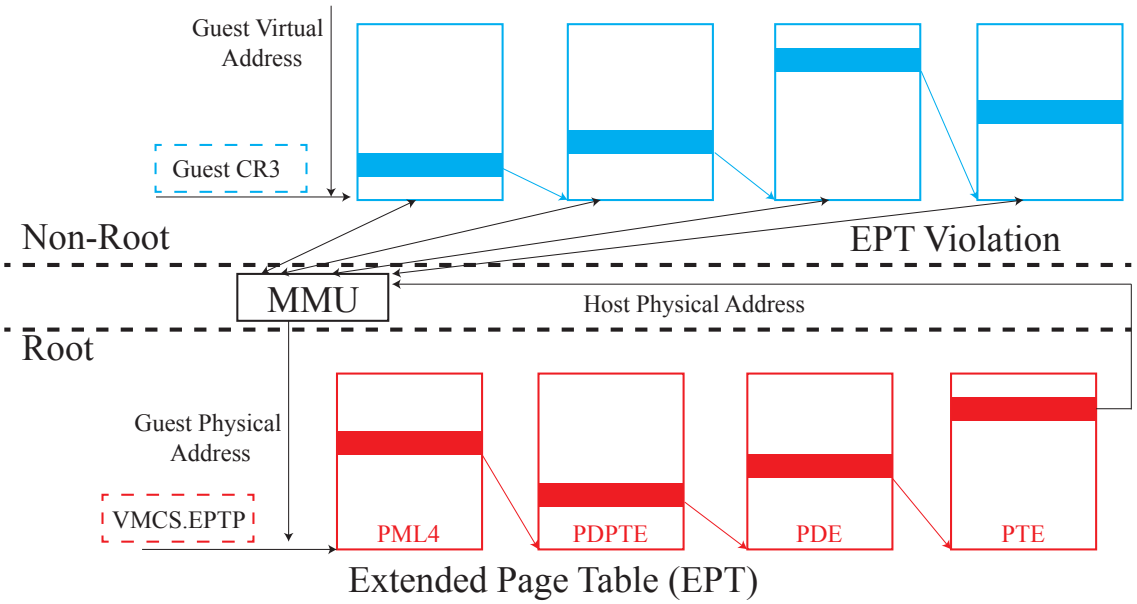


Fig. 2.: Memory Translation in EPT

4. The access is an instruction fetch but the EPT page does not allow execute operation.

As shown in Figure 2, the memory access made by guest VM will cause an EPT violation. MyCloud/MyCloud SEP hypervisor can receive the VMEXIT and check the permission to access the physical address. In MyCloud and MyCloud SEP, the cloud hypervisor will create the 4-step EPT table in order to trap 4-step address translation in guest VM. The hypervisor should claim the start address of EPT in *VMCS.EPTP* and configure the CR3 register for guest VM. The guest VM will translate a GVA to GHA using the page table pointed by CR3 register. In each step of address translation, EPT violation will happen because the hypervisor has marked each physical page as read-only. In the VMEXIT handler, the hypervisor can read the GPA and HPA from VMCS and check if this guest VM can access the intercepted memory space.

With the help of EPT technology, MyCloud and MyCloud SEP can provide memory isolation between VMs. Any memory access will be checked by the cloud hypervisor. Therefore, malicious memory access made by cloud administrators can be prohibited by cloud hypervisor.

With EPT memory virtualization, cloud hypervisor can create 2 or more views of the same machine memory region with different permission. In that case, the code/data of different VMs are isolated. The EPT technology can provide page-granular protections. The hypervisor can build the EPT page table for each VM. When the page table in guest VM try to retrieve the real memory address, CPU will firstly check the EPT table. If a page missing or page fault happens, the CPU will call the hypervisor to handle it. EPT is also a hardware feature and the EPT page table is controlled by the hypervisor. Therefore, the code and data in each guest VM is protected and isolated. If there is a malicious access to guser's memory, the hypervisor will receive the violation from CPU. If the hypervisor is capable to detect this malisiouc access, the attack can be prohibited.

1.3.3 Device Memory Isolation – Vt-d

In order to assign the physical device to guest VM and separate the memory between device and guest VM, MyCloud and MyCloud SEP enable Intel Virtualization Technology for Directed I/O (Vt-d). The general functionality of Vt-d is to isolate and restrict device accesses to the resources belong to guest VMs. Intel Vt for directed I/O technology can provide the hypervisor with the following capabilities:

1.3.3.1 DMA Remapping

DMA remapping can provide a hardware support for isolation between device memory and VM memory. Also, DMA remapping can assign a device to a specific

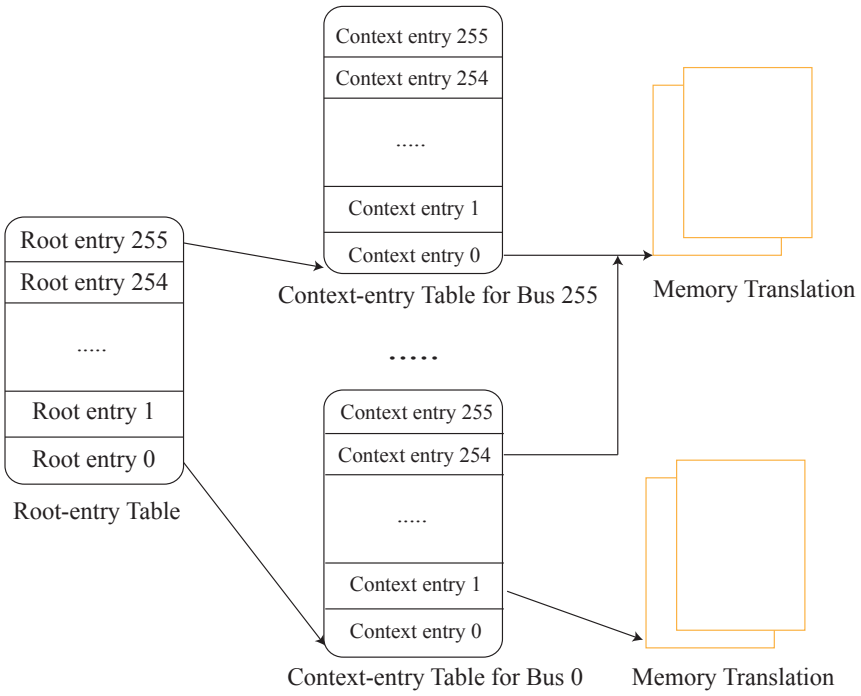


Fig. 3.: Assign A Device to Guest VM

VM through a distinct set of I/O page tables. The DMA remapping hardware can intercept device memory access and determine whether the access can be permitted. The cloud hypervisor can isolate the device DMA access to memory by programming the I/O page table. The DMA remapping can be programmed independently for each device. Similar to the guest physical address (GPA) DMA remapping will treat the address in a DMA request as DMA-virtual address (DVA). DMA remapping will be responsible to transform the DVA to its corresponding host physical address (HPA).

Figure 3 explains how to map a device to a specific memory space. The hypervisor can find the address of Root-entry Table in register *Root-entry table address register*. After the DMA remapping hardware intercepts the DMA request, it can acquire the bus number from DMA transaction's source-id field. The bus number will be used to index into the root-entry structure. The Context-entry maps a specific I/O device on

a bus to the assigned guest VM. The context-entry will store the address of a multi-level page table in ASR field. After the address translating in multi-level page table, device virtual address will be transformed into host physical address. The process of memory translation is the same as general memory management in operating system. Due to the DMA remapping, the memory space where a device can access will be restricted into a specific region.

1.3.3.2 Interrupt Remapping

The interrupt remapping architecture allows the cloud hypervisor to control the external interrupt generated by devices. The interrupt remapping hardware uses an interrupt remapping table specified through the Interrupt Remapping Table Address register. When the hardware traps the interrupt, the interrupt index can be used to search appropriate IRTE in interrupt remapping table. By programming the interrupt remapping table, the hardware will send the interrupt to the corresponding guest VM. However, the general interrupt cannot be remapped. The cloud hypervisor needs to program the Redirection Table Entries (RTE) in I/O APIC and MSI address and data register to support remappable MSI and PCI interrupt.

This technology is designed to isolate the device and VMs. Since device memory is accessed by DMA, the CPU is not able to intercept it. The hypervisor should program the DMA remapping table in order to make sure the device will not access VM's memory.

CHAPTER 2

RELATED WORK

Many existing migration approaches have been used to improve the performance of cloud platform and reduce the resource consumption. For example, [13] and [14] propose a virtual machine placement strategy in order to optimize the poper cost. Our proposed work is the first effort to deploy the placement to enhance cloud security.

In term of the hardware-based approach to protect users privacy, the most similar work to ours is Self-Service Cloud computing (SSC) [15]. SSC isolates the host OS into small components and SSC allows client VMs to execute some management of privileges, which used to be provided in administrative domain such as to access VM's memory, execute CPUID instruction, etc,. Similarly, NoHype [16] and [17] assign physical resources to VMs and dynamically eliminates VMM layer in order to narrow the hypervisor attack surface. Recent work also investigates the uses of nested virtualization to disaggregate some host VMM components to the guest VMM like CloudVisor [18]

2.1 Migration Based Privacy Protection Approach

An effective Virtual Machine placement strategy can greatly improve the performance of cloud platform. For example, the VMs with shared memory pages can be optimized by placing them in the same physical servers [19]. Those VMs can deliver data through shared memory instead of network.

To improve the efficiency of cloud, Many work dynamically clusters VMs and distributes resources to different clusters [14, 20, 21]. [13] developed a generic algo-

rithm to create a placement plan to reduce Estimated Total Execution Time (ETET). Work [22] provided a scheduling model to optimize virtual cluster placement through cloud offers. The cloud prices and user demand have been considered in the model. The experimental results on the real data show that dynamic placement plan can bring more benefits on reducing users' costs than the fixed one.

Our previous work [23] proposed to periodically migrating VMs based on game theory, making it much harder for adversaries to locate the target VMs in terms of survivability measurement. However, our previous work did not discuss how to evaluate the security of a cloud placement and how to generate a placement plan to improve the cloud security.

Unfortunately, none of the above work considered the privacy security issue. I will propose an innovative and effective migration based approach to protect user's privacy.

2.2 Hypervisor Based Privacy Protection Approach

Since the hypervisor is running with the highest privilege in cloud, previous work tried to protect user's privacy by utilizing the privilege of hypervisor. MAVMM [24] and Trustvisor [25] are light-weight hypervisors that can intercept and record the activities of guest VMs. The malware analysis tools running with MAVMM hypervisor can detect the adversary by analyzing those information.

NOVA [26] is a micro-kernel based hypervisor in order to improve the security of cloud platform by reducing the size of Trust Computing Base (TCB) [27]. However, NOVA can not monitor the malicious activities of the privileged VM and external device drivers. NOVA should rely on the host operating system to provide the virtualized environment for guest VMs, but masses of vulnerabilities in host OS will allow the adversary compromise the hypervisor easily.

XMHF [28] and Terra [29] is a tiny hypervisor which can supports other hypervisor development and protection. XMHF provide an extendable hypervisor design platform which also supports modular developments of hypervisor. Also XMHF deploy model checking to verify the hypervisor code. However, the design goal of xmhf is not to protect users privacy from inside and external attack.

Many researchers have focus on disaggregating the functionality of privileged VM into smaller components [30] [15] [31]. Bitvisor is established to reduce the attack surface of privileged VM (Dom0). Bitvisor can remove the untrusted device drivers to other unprivileged VMs. But the hypervisor still takes charge of many complicated management work like resource assignment, device virtualization etc,. Similar to Bitvisor, Self-service cloud (SSC) split Dom 0 into system-wide domain and user administrative domains, service domains and mutually trusted service domains . However those domains are still running in the privileged mode, thus the TCB size is not reduced. DeHype is also designed to separate KVM into deprivileged section and privileged section. However it relies on pinned memory blocks in linux kernel and mapping them to user space. The untrusted host OS is possible to fake the memory mapping. Xen-Disaggregation [32] can disaggregate the management virtual machine (Dom0) by moving the domain builder, the privileged component into a trusted compartment. However these compartments are still running in the privileged mode.

Nested virtualization technology [33] is introduced by Turtle [34] and Xen-blanket [35] which divide the hypervisor as host hypervisor and guest hypervisor in order to allow cloud users execute configured work in guest hypervisor. Many researchers focus on reducing the attack surface by separating the components of hypervisor. Our previsour work Splitvisor [36] [37] can divide the legacy hypervisor into the privileged section and unprivileged section. Only privileged section can be launched in the root

mode and responsible for guest VMs management. The unprivileged section will be in charge of presenting virtualized environment. Besides, Cloudvisor [18] also classify the hypervisor as host hypervisor and guest hypervisor. The host hypervisor executes privileged instruction like VMEXIT handler, while the guest hypervisor will provide rich functionalities like VMs management. However, CloudVisor is launched on an untrusted platform, but it can not provide an authenticated boot process for late launch. Furthermore, the host operating system is running in the same CPU privilege level as hypervisor. The adversary is highly possible to compromise the hypervisor via untrusted host OS. Neither Splitvisor nor Cloudvisor can detect the malicious activity of external device and privileged VM. Moreover, to deploy nested virtualization on x86 hardware imposes tremendous performance penalties which increases exponentially with nesting depth [33].

HyperLock [38] build an isolated memory space and develop Hyperlock controller to restrict the malicious access to privileged resource. However this approach only focuses on protecting privacy on memory and require the support of host OS.

Some previous work measures the hypervisor integrity on the basis of System Management Mode (SMM) [39] hardware feature. For example, Hypersentry [40] and HyperCheck [41] can launch SMI single to CPU and switch to SMM mode. The CPU will store the current machine status in SMRAM, then the network card will deliver the register values and memory to remote verification machine. SICE [42] and SecureSwitch [43] utilize x86 SMM to isolate the TCB. The security of isolated environment is guaranteed by the TCB including hardware, BIOS and SMM program of $\sim$ 300 LOCs. However, SICE only supports one VM so it will not be compatible with any cloud platform. Flicker [44] is also considered a privacy protection solution based on CPU features [45] [46]. Unfortunately, it only offers application level protection and is not a general solution for VMs in cloud.

In order to eliminate the security threat from hypervisor and privileged VMs, NoHype [16] [47] removes the virtualization layer. However, each guest VM must be assigned dedicated physical CPU core and nested page table. This design restricts the number of VMs on cloud platform.

Besides privacy protection in cloud computing, many research efforts focus on protecting the privacy of user application against untrusted operating system using a VMM-based approach [48, 49, 50, 37, 51]. The goal of our work is different from that of above research. We aim to protect privacy of guest VMs (including the hosted user applications) against the untrusted cloud administrators, rather than protecting the user applications' privacy against the untrusted OS.

MyCloud and MyCloudSEP reduce the attack surface of privileged VMs by removing the Dom0 or hostOS from privileged mode to non-privileged mode. We also move the resource management, device driver and virtualized device from the hypervisor to a resource allocator in order to reduce the possibility of hypervisor to be compromised. In MyCloud and MyCloud SEP, we design user-configured access control scenario so that the mutually distrust can solved. The hypervisor will be responsible for monitoring the malicious resource access of cloud provider and external device. [52]

CHAPTER 3

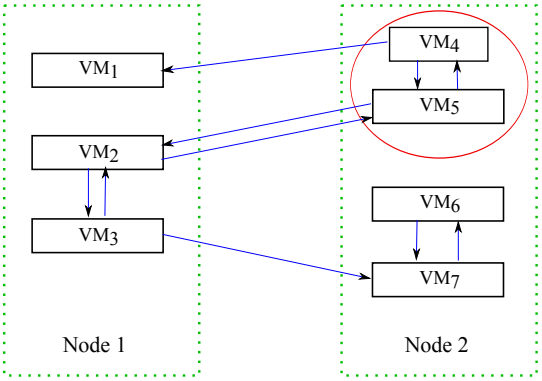
MIGRATION BASED PRIVACY PROTECTION - VM PLACEMENT

3.1 Introduction

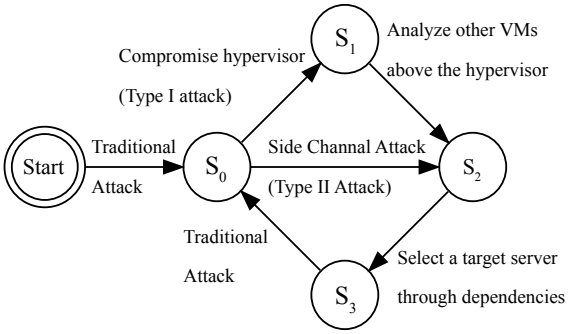
In current public cloud, VMs are installed in the same physical machines. Some of VMs working in the same subnet or physical server may collaborate in order to complete a service. Collaborating with vulnerable VMs or running in the physical server with malicious VMs will increase the security risk. The connections between VMs via network or shared physical resources will introduce attacks. The external adversary can compromise a vulnerable VM, then find next target via network connection. Also he can deploy a malicious VM and attack other VMs in the same physical server. In order to improve the security of the entire cloud, we design a VM placement strategy which will migrate the legitimate VMs to a secure physical servers or network.

We have already found many attacks against vulnerabilities in cloud hypervisor and the control VM. For example, some adversaries exploit the vulnerabilities of cloud hypervisor (e.g. CVE-2007-4993,2007). Once they compromise the hypervisor (e.g. KVM and Xen), the users' VMs will be taken over. Some other attackers will place a malicious VM in the public cloud and compromise the VMs running in the same physical server by side channel attack [11]. Additionally, the adversary can find the next target by analyzing the network connections of compromised VMs.

In order to protect users privacy in public cloud, I propose a migration-based approach which generates the VM placements strategy based on security evaluation of each VM. To evaluate the security of each VM, I deploy District Time Markov Chain (DTMC) and predict the possibility of each VM being compromised. Then a place-



(a) Cloud Placement Example.



(b) The State Transition Graph of Attacks.

ment strategy will be produced based on security evaluation. After VMs migration, users' VMs will survive before the attack completes. Meanwhile, my approach also considers the performance overhead because the closer connected VMs are placed, the better performance the cloud platform will acquire.

To the best of our knowledge, this approach is the first effort to develop the following mechanisms and techniques to enhance cloud security through changing cloud placement. The contribution of my migration-based placement strategy is as below:

1. I present a systematic approach to predict the possibility of VM on each attack step, then move away the VMs before attack succeeds.

2. I propose an algorithm to generate a secure placement plan which also takes performance cost into consideration.

3. The evaluation results in a real public cloud show my approach can greatly improve the security of entire cloud platform.

3.2 Assumption and Design Goals

An example of cloud placement is shown in Figure 4a. Each VMs on every node (physical server) may execute different services and some of VMs are dependent

on others. Node is a physical machine where runs a few of VMs with the limit of hardware resources. The cloud provider is the owner of a public cloud and sell the cloud resources to cloud users. The cloud provider also takes charges of managing the cloud resources and protecting users privacy. Therefore, cloud provider will design and deploy VMs placement strategy in a public cloud.

Generally, the adversaries should take several steps to compromise a VM. As shown in Figure 4b. The adversary starts from compromising one VM on a new server (S0). After hypervisor is compromised (S1), the adversary will collect dependency information of compromised VMs (S2). Finally, new target server will be selected (S3).

We make the following assumptions on cloud adversaries who want to compromise the VMs and users privacy.

1. The cloud adversaries can detect the vulnerabilities of both cloud platform and virtual machines.

2. The cloud adversaries will follow the attack transition graph (Figure 4b) to compromise a VM step by step.

3. The cloud adversaries always choose the easiest target in term of the vulnerabilities.

4. The attacker has no global view of the cloud at the beginning of the attacks. However, the attacker may acquire more knowledge after compromising more nodes in the cloud.

If the cloud provider can migrate VMs to a safe node before the node is compromised, migrated VMs will survive this attack. We set up the following goals when design the placement algorithm.

1. Reduce the number of compromised VMs.

2. Increase the survivability of services.

3. The placement algorithm is also compatible with performance requirements.

In order to verify the improvement of survivability after migration, we define the survivability of a service in Theorem 1. Then we need to evaluate the survivability of a node as shown in Theorem 2.

Theorem 1 (Survivability of a Service)**.** *Given a service S_i (VM chain) including some related $S_i = \{VM_a, VM_b, ..., VM_n\}$ and node set $N = \{N_1, N_2, ..., N_m\}$, If the survivability in specific attack step for the Nodes which hold the VM belong to S_i is $\{PN_1, PN_2, ..., PN_m\}$, Then survivability (PS) for service S_i is below:*

$$PS_i = \prod_{j=1}^{m} PN_j. \tag{3.1}$$

Theorem 2 (Survivability of a Node)**.** *Given a node N and a set of $VMs = \{VM_a, VM_b, ...VM_m\}$ which locate at node N, and the compromised probability for these VMs are $\{P_a, P_b, ..., P_m\}$, the survivability (PN) for Node N is below:*

$$PN_N = \prod_{j=1}^{m} (1 - P_j) \tag{3.2}$$

3.3 Approach Overview

The structure of migration-based approach is shown in Figure 4. In order to accomplish the design goal, my approach includes three components: security evaluation, strategy generation, and performance evaluation. First of all, the dependency

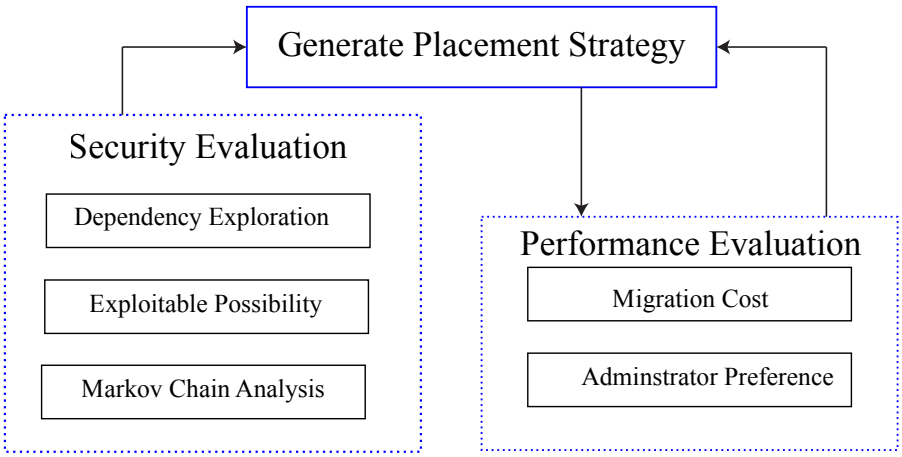


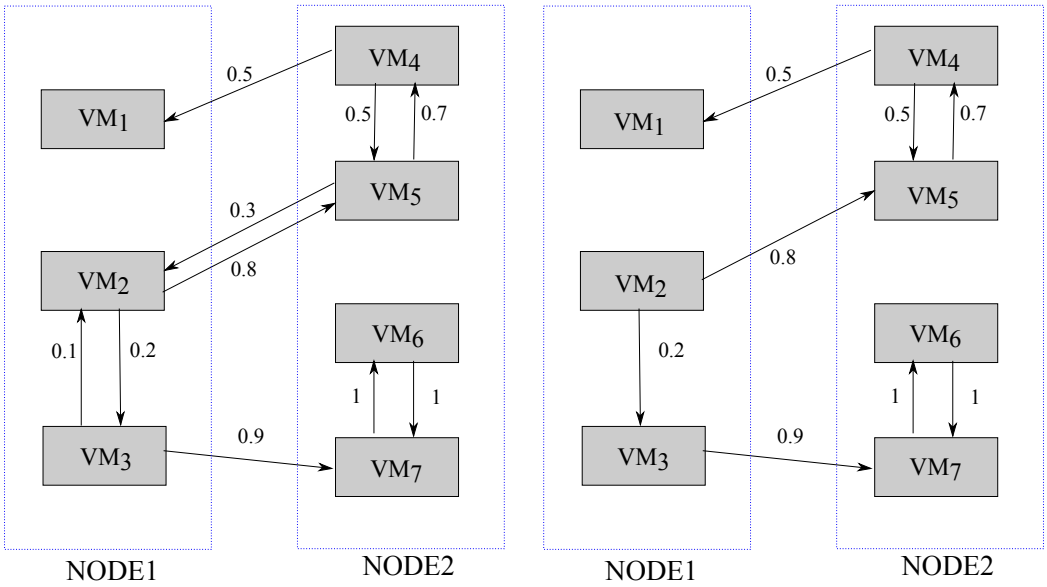
Fig. 4.: Architecture

exploration mechanism detects the service dependencies among VMs through network connections.We evaluate each VM's security level according to the vulnerabilities found in VM's operating system. Afterwards, we utilize Discrete Time Markov Chain Analysis (DTMC) to predict the possibility of successful attacks in each attack step. Finally, we design an algorithm to create the placement plan which takes security and performance into consideration.

3.3.1 Security Evaluation

In order to quantify the vulnerability, we firstly scan the guest VMs and detect the vulnerabilities matched in National Vulnerability Database (NVD) [53]. Afterwards, Discrete Time Markov Chain Analysis (DTMC) will predict the possibility of an successful attacks in each step.

Common Vulnerability Scoring System (CVSS) [54] provides a framework to scan the guest VM and score security of guest VMs based on vulnerabilities. There are three metrics group in CVSS system: base, temporal, and environment. Each of them can represents different characters of vulnerabilities.



(a) Step One. (b) Step Two.

Fig. 5.: An example based on Markov Chain Analysis.

3.3.2 Markov Chain Analysis

In order to represent the attack path and possibility of successful attack in each step, we will use attack dependency graph (ADG) as shown in Figure 5. The procedure of DTMC prediction is explained as follows:

1. For n nodes in ADG graph, assume the initial probability distribution on each node is $\underline{\pi}(0) = (1,\underbrace{0,0,\ldots,0}_{n-1})$. The initial $\underline{\pi}(0)$ is determined by attacker's first choice.

2. In attack step 1, the possibility distribution of attacker can compromise connected VM's is $\underline{\pi}(1) = \underline{\pi}(0)P^1$

3. After k^{th} step attack, the possibility distribution will become $\underline{\pi}(n) = \underline{\pi}(0)P^n$ where $\mathbb{P}$ is the state-transition probability matrix of DTMC and $\mathbb{P} = \underbrace{\mathbb{P}\cdot\mathbb{P}\cdots\mathbb{P}}_{n}$.

For example, Figure 5 is used as an example to explain how DTMC predict the attack possibility in each step. I assume that the first compromised VM should be VM_2, so $\underline{\pi}(0) = \{0\ 1\ 0\ 0\ 0\ 0\ 0\}$. Hence we may obtain the attack possibility from above assumption. For example, the edge from VM_3 to VM_7 is 0.9, which indicates that after VM_3 is compromised, VM_7 will have 90% chance to be taken over by the attacker. The corresponding state-transition probability matrix $\mathbb{P}$ is as follows.

$$\mathbb{P} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.2 & 0 & 0.8 & 0 & 0 \\ 0 & 0.1 & 0 & 0 & 0 & 0 & 0.9 \\ 0.5 & 0 & 0 & 0 & 0.5 & 0 & 0 \\ 0 & 0.3 & 0 & 0.7 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix} \tag{3.3}$$

For Step 1, we get the possibility of being compromised for each VM is $\underline{\pi}(1) = \{0\ 0\ 0.2\ 0\ 0.8\ 0\ 0\}$. Based on the result, VM_5 is the most dangerous VM, then we remove the edges which point to VM_5 because the attacker will not compromise VM_5 again in the following attack path. Hence, the result of matrix $\mathbb{P}$ after step one should be as follows and the DAG should be:

$$\mathbb{P} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0.1 & 0 & 0 & 0 & 0 & 0.9 \\ 0.5 & 0 & 0 & 0 & 0.5 & 0 & 0 \\ 0 & 0.3 & 0 & 0.7 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix} \tag{3.4}$$

Markov Chain analysis will end if the step reach the longest path in an ADG. Hence, The probability distribution for the initial state and the first 6 steps are as follows.

$$\begin{pmatrix} \underline{\pi}(0) \\ \underline{\pi}(1) \\ \underline{\pi}(2) \\ \underline{\pi}(3) \\ \underline{\pi}(4) \\ \underline{\pi}(5) \end{pmatrix} = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.2 & 0 & 0 & 0.8 & 0 \\ 0 & 0.26 & 0 & 0.56 & 0 & 0 & 0.18 \\ 0.56 & 0 & 0.26 & 0 & 0 & 0.18 & 0 \\ 0 & 0.026 & 0 & 0 & 0 & 0 & 0.414 \\ 0 & 0 & 0.026 & 0 & 0 & 0.414 & 0 \end{pmatrix} \tag{3.5}$$

where $\underline{\pi}(k), 0 \leq k \leq 5$ is the probability distribution in step k. In the above example, according to $\underline{\pi}(4)$, in the 4^{th} step, the probability that VM_2 is compromised will be 2.6% and the probability that VM_7 is compromised will be 41.4%.

3.3.3 Placement Generation

Based on the above discussion, we have acquired the possibility of being attacked for each VM. Next, we will design a placement strategy to reallocate guest VMs before the attack succeeds. The principle of new strategy is isolating the VMs with high security risks from VMs with low security. In order to reduce the performance overhead, connected VMs with similar security risk will be assigned in the same node. When design the placement algorithm, we assume the node will have enough resources capacity (e.g. CPU, memory and disks etc.) to hold all guest VMs. In each attack step, DTMC and CVSS will predict the attack possibility for each VM. The algorithm will sort the possibility and find the most "dangerous" VM which is most likely to be compromised. The algorithm will assign the most dangerous VM to a dedicated node and allocate other VMs to different node. Therefore, even if the most dangerous VM is compromised, other VM will not be exploited by the attacker.

Algorithm 1 Placement Strategy Generation Algorithm

Require:

- Virtual machine set $V = \{V_1, V_2, ...V_n\}$
- Dependent VMs set for each VM set *DependentVMs*, where $DependentVM_i$ is the set which represents the dependent VMs for VM_i. (i ≤ n)
- The Physical machine (Node) set $N = \{N_1, N_2, ...N_k\}$
- The compromised possibility for each VM is P_i(i ≤ n)

Ensure: Placement Strategy

1: Sort VMs in ascending order of attack possibility **Sort(**$V = \{V_1, V_2, ...V_n\}$**)**
2: *dangerousVM* = **findMostDangerous()** will find the most dangerous VM index by comparing compromised possibility of VMs in set V
3: *dangerousNode* = **findRandomNode()** will find a random node to store the dangerous VM.
4: **Map(***dangerousVM***,** *dangerousNode***)** will assign *dangerousVM* to *dangerousNode*
5: **while** !*V*.empty() **do**
6: *node* = **findRandomNode()** will return a new node
7: *newVM* = *V***.pop()** find a safe VM
8: **Map(***NewVM***,** *node***)** assign new VM to safe node
9: **Map(***DependentNewVM***,** *node***)** assign the connected VMs into same node.
10: update *V*
11: **end while**

Hardware
CPU: Intel Xeon x5650 2.66GHz × 2
RAM: 8GB DDR3 1333MHz ×
Ethernet: Broadcom NetXtreme II BCM570

Table 1.: Platform Specifications of DTMC Calculation.

3.4 Evaluation

3.4.1 Case Study

Thanks to [ANONYMIZED COMPANY NAME] offering us a real cloud data set. The ADG and VMs relations are based on this data set. The data set and our placement conditions are explained as follows.

- 81 VMs and 10 physical nodes.
- The capacity for 10 nodes are 20, 15, 10, 10, 10, 5, 5, 5, 5, and 5.
- 81 services are running on these VMs.

3.4.1.1 Migration Overhead

When migrating a VM, the VM is usually shut off first, hence, migration time is one of the most significant factor we should consider in order to improve the system performance. In order to test the overhead on VMs migration, we design an evaluation on the platform specified as Table 1.

Figure 6 presents a migration delay of Web server on our platform.

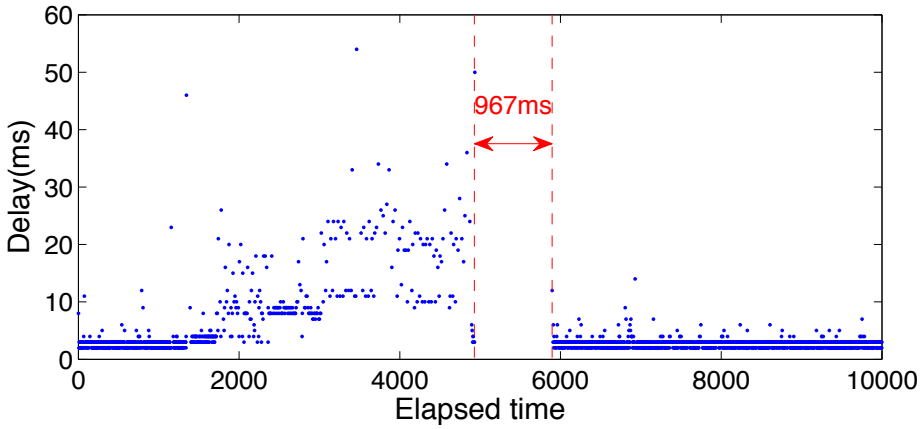


Fig. 6.: Migration impact on response delay of web server. As illustrated in the graph, the web service downtime due to migration is 967ms.

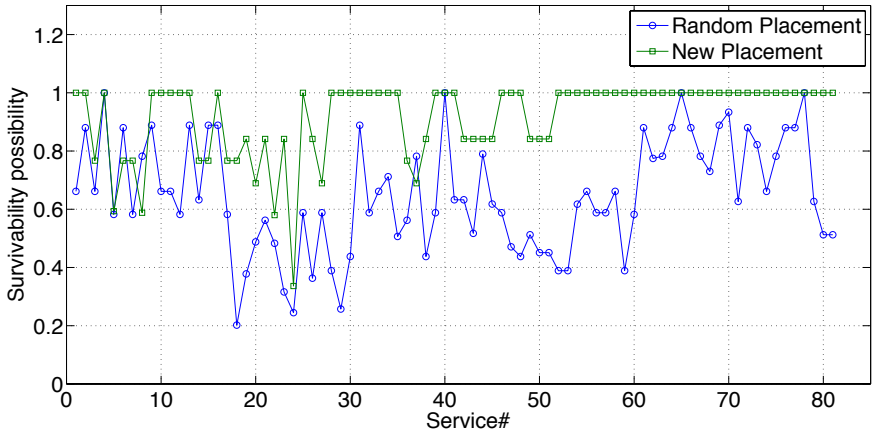


Fig. 7.: Comparison of Survivability.

3.4.1.2 Security Improvement

According to our experimental results shown in Figure 7, the 91.3% services obtained improved survivability. The maximum survivability enhancement is 74.28% and the average improvement of survivability possibility is 27.15%.

In our experiment, we can find there are 20 VMs will be compromised at attack step 1 in random placement plan. However, in our new placement plan, the number of compromised VMs is only 4. Moreover, according to our statistics, the average compromised VM number is 4 in our plan, But in random placement, this average number of compromised VMs is 11.

3.5 Summary

Cloud computing is quickly becoming more and more important in computing infrastructures. In the migration-based privacy protection approach, we demonstrated that the placement of VMs can make huge difference in terms of security levels.

Based on survivability evaluation of VMs and DTMC analysis, we developed an algorithm to generate a safe placement plan that moves the guest VMs before attack succeeds. The experimental results show that our algorithm can significantly improve the survivability of VMs in the cloud and reduce the number of compromised VMs in case of attacks. The overheads of our proposed approach are also practical.

In many cases, same vulnerabilities can run on different VMs which will cause adversary to compromise these different VMs simultaneously. However, our scenario doesn't take this problem into account. Also, my approach does not help cloud providers decide how often to migrate the guest VMs. According to the performance evaluation part, if the cloud providers migrate new VMs too frequently, the VMs will response slowly. On the contrary, if the cloud providers keep a migration plan for

long time, the attackers will have enough time to complete the attack.

CHAPTER 4

USER CONFIGURED CLOUD PLATFORM WITH MYCLOUD

4.1 Introduction

Privacy security is still the major concern for cloud users. The migration based protection approach cannot fundamentally solve the privacy issues because the cloud provider is still in charge of VMs migration. The cause of privacy issues is that current cloud design may enable the adversary abuse the privilege of cloud provider to disclose users' privacy. Once the cloud provider is compromised or the malicious cloud administrator acquire the management of cloud provider, users' privacy will be breached. In addition, in current cloud platform, only cloud provider takes charge of cloud management. Cloud users do not realize whether their private information is securely protected, since they are not engaged in data management. In this project, I propose a new scenario - MyCloud [55] which replies on hardware virtualization technology and eliminates the privilege of cloud provider. In MyCloud, only hypervisor is executed in CPU root mode and responsible for protecting users' privacy from malicious inspection. The cloud users can design the policy of privacy protection and deploy user-configured privacy through HyperCall.

In current cloud platform like KVM [2], Xen [1] and VMware [3] etc, cloud provider is endowed with the privilege to manage the whole cloud resources including users' data. Cloud providers allocate specific resources to guest VMs, scan virus on guest VMs and migrate guest VMs in order to improve the performance of cloud platform. Cloud users cannot manage the private data stored in current cloud platform. For example, the users of Amazon EC2 [56] cannot inspect their owned memory

because cloud providers will not allow such privileged operations.

Mutual distrust between cloud users and the cloud providers is another challenge on privacy protection in cloud computing. On one hand, the cloud providers are motivated to monitor client virtual machines in order to make sure that users will not use cloud resources to launch illegal services. For example, a lot of spam emails are reported to sent from IP addresses belong Amazon EC2 [57]. On the other hand, user's concerns about privilege misuse by the cloud providers are greatly increasing because a malicious administrator can easily disclose their private data [58] .

According to my understanding, current cloud privilege design is the root cause of privacy issues in cloud computing. Current trust computing base (TCB) of cloud platform includes a privileged control VM (Dom 0 or host VM) and cloud hypervisor. Usually, this control VM is a complete commercial operating system and executes multiple third-party device drivers. Therefore, it is quite difficult to verify the integrity of the control VM. The external adversaries can compromise the current cloud TCB via the vulnerabilities on both control VM [6] [7] [8] and cloud hypervisor [9] [10]. After the attack succeeds, the adversary will disclose user's privacy by using the administrator privilege of cloud platform. In current cloud design, the cloud users can only deploy VMs in non-root mode. The inside attackers (malicious cloud administrators) can disclose user's privacy by using the privilege of cloud providers, while the cloud users are not aware. Additionally, current cloud users cannot enrol in the privacy management. Hence, cloud providers are difficult to convince users that their private data are well protected.

One possible solution is homomorphic cryptography [59] which protects users' private data by encrypting before users submit them to cloud providers. The homomorphic cryptography technology also allows CPU operates arbitrary computing on the encrypted data. However homomorphic cryptography cannot offer practi-

cal performance at current state [60]. Another solution to protect users' privacy is completely removing the privilege of cloud provider. For example, NoHype [16] removes the privileged VM (e.g. dom0 or hostOS) assigned by cloud providers and only keeps significant management in cloud platform (e.g. VM initialization and migration). However, such design disables the cloud provider's ability to manage and protect cloud resources. Cloud providers will not deploy Virtual Machine Inspector (VMI) [61] to detect virus or spam software over VM's side.

Although homomorphic encryption [60] can solve the above privacy issues, it does not offer practical performance solution yet. Another scenario to solve cloud privacy issues is Self-Service Cloud computing (SSC) [15] which separates the privileges of Dom0 into multiple domains, user domains and an MTSD domain. The MTSD domain checks regulatory compliances mutually agreed upon the cloud provider and the clients. However, the TCB size of SSC is still too large and the separated small domains are still running in the root mode.

We propose a new cloud architecture - MyCloud which eliminates the privilege of cloud providers to fight against inside adversaries and reduces the TCB size to block external adversaries. In MyCloud, due to the small TCB size, it becomes practical for the cloud providers to verify the integrity of hypervisor. Meanwhile the cloud users are able to manage their private data by themselves and set up the protection policy via hypercall offered by MyCloud hypervisor. The contribution of MyCloud is as follows:

- We propose a new virtualization architecture, MyCloud, to support user-configured privacy protection. MyCloud eliminate the privilege of cloud providers so that the malicious administrators do not have privileges to breach users' privacy.
- We minimize the TCB of MyCloud by removing the control VM from the root

mode of the processor. Thus, the external adversaries are difficult to find vulnerabilities in MyCloud platform.

- We implement a prototype of MyCloud on x86 platform, which has acceptable performance overhead but much stronger security protections.

4.2 Design

4.2.1 Threat Mode and Assumptions

The most popular threat is external attack which will compromise the control VM or cloud provider through vulnerabilities [6] [7] [8] [9] [10]. Then the external adversaries can disclose users' privacy by using the privilege of cloud platform.

Inside attack is another threat on users' privacy. We distinguish malicious cloud administrators from cloud providers. Usually, well-known enterprises such as Google and Amazon are motivated to protect users' privacy, which can improve their reputation and attract more customers. On the contrary, the cloud administrators employed by cloud providers, are very likely to breach users' privacy for pursuing monetary benefits. Even if the cloud administrator is benign, he may make mistakes by accident and cause privacy breach. Therefore, only cloud administrators are considered adversarial in this project.

Physical attacks or other hardware attacks are not considered. The attack launched from System Management Mode (SMM) [62] of processors is not included either, because a proper configuration of the System Range Register (SMRR) is required to ensure this assumption.

In MyCloud design, we assume that the hardware is equipped with Trusted Execution Technology (TXT). Therefore, the integrity of crucial components in TCB can be measured. Additionally, we assume the physical device in cloud environment

is able to support Intel Vt-c technology [63] which can offer 255 virtualized devices to cloud users. MyCloud will directly assign these devices to cloud users.

In this project, we assume that the swap area of the guest VMs is turned off or the swapped pages are encrypted in order to protect the guest VM space.

4.2.2 Design Goals

The principle of MyCloud design is to provide privacy protection mechanism but not the policy. The primary goal of MyCloud is to enable configurable privacy protection and isolate users privacy from other VMs including the control VM. In addition, the MyCloud design eliminates the privilege of cloud providers to stop inside attacks and reduce the TCB size to protect privacy from external attack. The detailed design considerations are as follows.

1. Offer Users-Configured Privacy Protection In MyCloud, an Access Control Matrices (ACM) is maintained in hypervisor memory in order to record user's configurations on privacy protection. The cloud users are able to set up ACM via the hypercall offered by cloud hypervisor. The control VM has no access permissions to any of the guest VM unless the guest VM grants the permission. By default, no access permission is granted to the cloud provider.

2. Minimize TCB Size The TCB size of the cloud architecture with MyCloud should be as small as possible. Hence, the external adversaries cannot detect the vulnerabilities easily. Also the approach to measure the integrity of MyCloud implementation has restrictions on the TCB size. For example the recent successful report of formal verification shows the capability of a general-purpose kernel with 8.7K LOCs [64]. Therefore, we need to control the TCB size by including only security related or crucial functionalities.

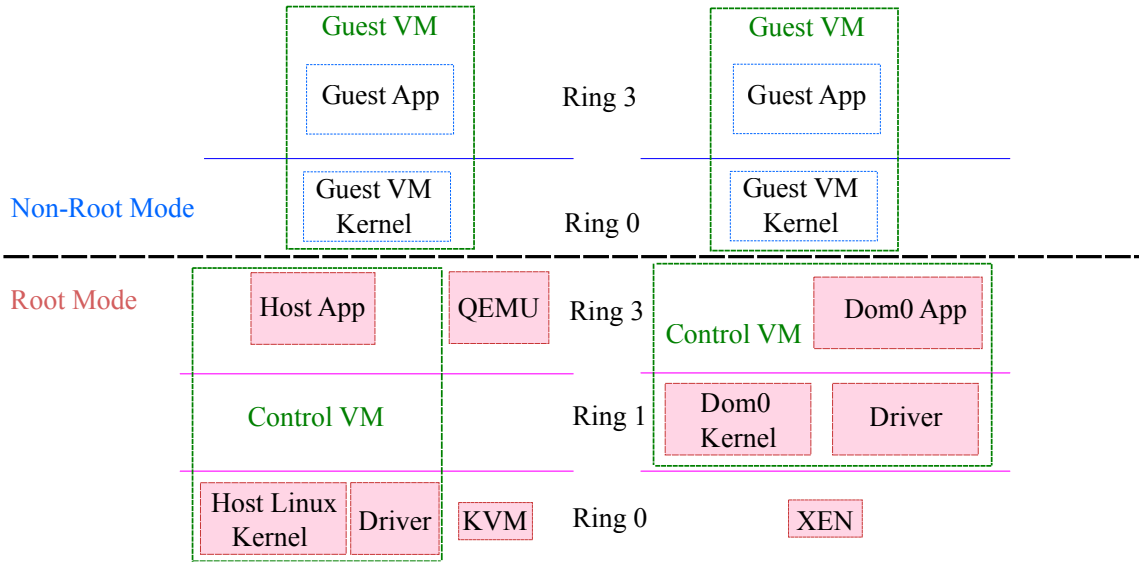


Fig. 8.: Type 1(Xen) and Type 2(KVM) cloud architectures

3. Isolate Cloud Users Space in VM Level The isolation granularity is the VM level in Iaas Cloud, because VM is a simple encapsulation of privacy for each cloud user. Protecting privacy at the VM level is more likely to preserve backward-compatibility, without the need of modifying OS kernels and applications.

4. Eliminate the Privilege of Cloud Providers. In order to protect users' privacy from inside attack, MyCloud should alleviate the privilege of cloud providers. The cloud providers need to request user's permission before perform privileged work (e.g. spam inspection). However, it can normally perform cloud resource management (allocation and migration).

4.2.3 MyCloud Architecture

Intel virtualization extension VMX [39] and AMD SVM [65] will divide the CPU privilege mode into *root* and *non-root*. In each mode, there are four privileged levels from ring 0 to ring 3 where ring 0 has the highest privileged level. In the operating

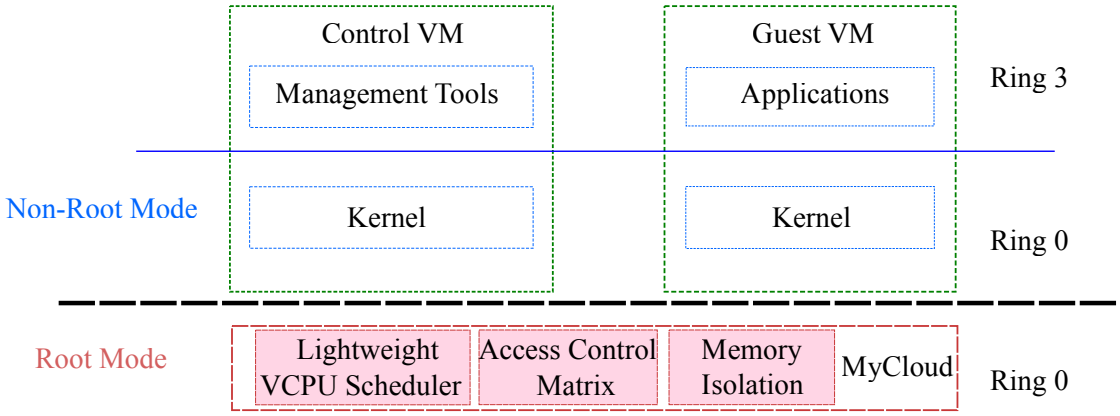


Fig. 9.: MyCloud architecture

system, the OS kernel is running in ring 0 and applications are executed in ring 3.

Figure 8 shows the architectures of both KVM [2] and Xen [1] In both architecture, the control VM created by cloud providers does not have a separate VMCS for virtual machine context switching so it runs in the root mode. The management tools and device drivers provided by cloud providers are running in the control VM at root mode. QEMU is also executed in the root mode to handle exceptions from guest VM. Moreover, since the control VM is in the root mode, the control VM is able to manipulate the VMCS structures of all guest VMs and the page tables of all other VMs. The guest VM is placed in non-root mode under the monitoring of the control VM. It is impossible to protect any guest VMs from the malicious cloud administrators.

Compared with existing cloud design, figure 9 shows the architecture of MyCloud. Only the cloud hypervisor runs in the root mode and maintains security related components in the TCB. In our design, the scheduler is a timer triggered preemptive scheduler against DoS attacks from any VM running on the platform. Memory iso-

Table 2.: Access Control Matrix of MyCloud.

(A-Allocation, M-Migration, D-Deallocation, H-Hyper Calls, R-Read, W-Write)

Components	VMM	Control VM	VM_i	VM_j	ACM_i	ACM_j
VMM	Full	Full	Full	Full	Full	Full
Control VM	H	Full	A/M/D/ACM_i	A/M/D/ACM_j	R	R
VM_i	H		Full	ACM_j	R/W	
VM_j	H		ACM_i	Full		R/W

lation is also enforced by the MyCloud hypervisor. I will describe these components implementation in the Section 4.3. In MyCloud design, there is no operating system running in the processor's root mode. Therefore, no VM, including the control VM, is more privileged than others, or can manipulate any others. The access permissions are specified by an Access Control Matrix (ACM) in the cloud hypervisor. According to the ACM, the control VM can access a guest VM's space if and only if the guest VM explicitly grants the permission.

4.3 Implementation

4.3.1 User-Configured Access Control

As shown in Figure 9, MyCloud design removes the privileges of the control VM and enables cloud users to configure the privacy protection on Access Control Matrix (ACM) via the hypercall provided by MyCloud hypervisor. ACM is maintained in hypervisor's memory at root mode. Any access to modify ACM will be trapped and checked by MyCloud hypervisor.

The ACM allows a cloud user to choose which part of private memory in the user's VM space can be accessed by the cloud provider or other guest VMs. As shown in Table 2, ACM_i is the Access Control Matrix of VM_i. The access permissions design of MyCloud architecture is completely different from any of the existing cloud

platforms. Most existing designs assign full privileges to the control VM, which causes security problems once the control VM is compromised. Even worse, users have no privacy if the control VM has full privileges.

In ACM, only the hypervisor owns the full privilege of the whole cloud platform such as accessing physical memory belong to cloud users and cloud providers as well as modifying the ACM table as the request of cloud users. Each cloud user can modify the access permissions to the user's space. By default, all accesses by other users including the control VM are prohibited. However, cloud users can grant access permissions to other users, or the cloud provider to enable information sharing or virus-scan.

In MyCloud, $\boldsymbol{ACM_i}$ of a guest VM_i is implemented by a Access Control List (ACL) that specifies memory regions, VM identifiers, and access permissions. When a guest VM or the control VM wants to access a memory region of other VMs for executing privileged operations, e.g., doing Virtual Machine Introspection [61] for virus scan, the VM initiates a HyperCall to request the operation. The hypervisor will check the requests against the ACL of the visited VM. If the access is permitted, the hypervisor will conduct the operation on behalf of the requesting VM. Otherwise, the access will be denied.

Figure 10 shows a sequence of machine instruction level operations on how the ACM is set and how one VM checks the access request against the ACM. In the figure, by utilizing a HyperCall, $\mathrm{VM}_{\boldsymbol{A}}$ can initialize $\mathrm{ACM}_{\boldsymbol{A}}$ on ACM. If $\mathrm{VM}_{\boldsymbol{B}}$ wants to access the memory of $\mathrm{VM}_{\boldsymbol{A}}$, $\mathrm{VM}_{\boldsymbol{A}}$ should grant $\mathrm{VM}_{\boldsymbol{B}}$ the permissions by sending MyCloud hypervisor a hypercall to modify $\mathrm{ACM}_{\boldsymbol{A}}$. After that, When $\mathrm{VM}_{\boldsymbol{B}}$ is scheduled and try to access $\mathrm{VM}_{\boldsymbol{A}}$'s memory, MyCloud hypervisor can intercept this activity by EPT violation. Then MyCloud hypervisor can check the access permission on ACM and allow $\mathrm{VM}_{\boldsymbol{B}}$ complete the privileged work e.g., reading $\mathrm{VM}_{\boldsymbol{A}}$'s kernel data structures.

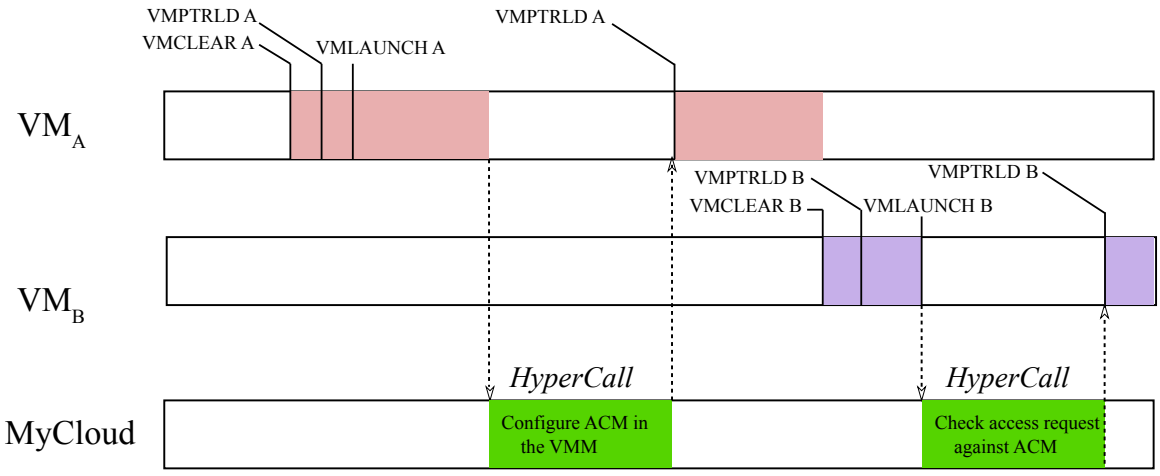


Fig. 10.: The procedure for users to modify the ACM

Note that we do not need security keys for VMs to implement the HyperCall. The hypervisor assign each guest VM a different identifier (VPID) in VMCSs when cloud users create guest VMs. It is impossible for one VM_i to set up ACM_j if $i \neq j$.

4.3.2 Memory and Device Isolation

Intel hardware virtualization provides extended page table (EPT) to translate guest physical address (GPA) to host physical address (HPA). MyCloud will use EPT table to verify the access permission to users memory space. In EPT table, we can mark the host page table as read only so that any address translation on EPT will be trapped by MyCloud hypervisor due to EPT violation. MyCloud hypervisor will acquire the address space from VMEXIT data structure and check the access permission on ACM.

In guest VM, the guest page table (gPt) specified by CR3 register of guest VM is responsible for translating guest virtual address (GVA) to guest physical address (GPA). EPT table controlled by the hypervisor is used to translate GPA to HPA. The address of EPT is specified by a VMCS filed (VMCS.EPTP). As shown in Figure 11,

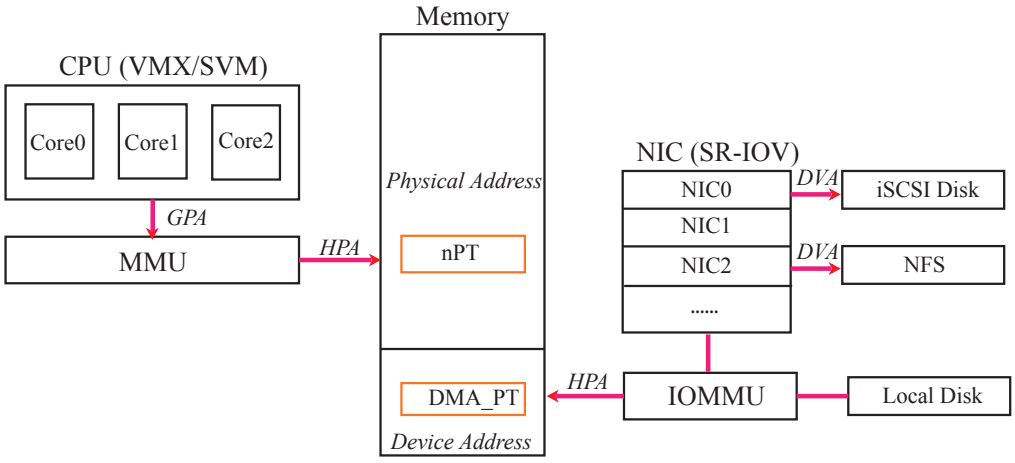


Fig. 11.: Memory and I/O management in MyCloud.

MyCloud sets up the EPT and MMU will automatically translate guest physical address to machine address. Once the EPT table is set up, memory translation will be processed by MMU if there is no EPT violation or EPT misconfiguration happens. The hypervisor will update the EPT table or check the access permission when EPT violation happens.

Since the control VM has its own EPT and VMCS, it cannot access any other guest VMs either. The control VM has can configure a resource table about the current memory allocations. When creating a new VM, the control VM initiates a hyper call to allocate memory for the guest VM. The hypervisor handles the boot process of the new guest VM. By default, no access permissions are granted to the control VM to access the new guest VM space once the memory is allocated.

I/O management is another important issue to consider when MyCloud remove the control VM from root-mode. Usually, device drivers must directly communicate with external devices in order to complete the I/O operation. Because including the

data structure for multiple device drivers in MyCloud hypervisor will increase the attack surface, MyCloud hypervisor will allocate each guest VM a dedicated external device. However, the number of external device in cloud platform is limited. In order to solve the above problems, MyCloud hypervisor relies on Intel Vt-c [63] technology to generate maximal 255 physical devices in cloud platform. As shown in Figure 11,the MyCloud hypervisor is responsible for configuring the I/O interrupt remapping and DMA remapping page table in Input/Output Memory Management Unit (IOMMU) to assign device to guest VM.

In Intel Vt-c technology, peripheral devices start to support SR-IOV [66] to enable a Single Root Functions to be prepared as multiple separate physical devices, called virtual functions (VFs). Therefore, in the opinion of cloud users, there are a lot of physical devices in the cloud platform supporting SR-IOV. The hypervisor can program IOMMU and assign any of these devices to a guest VM.

Like MMU that translates virtual address to physical address, the IOMMU takes care of mapping device virtual address to physical address. With the help of IOMMU, devices can be directly assigned to VMs. This kind of direct assignment of devices also provides very fast I/O and eliminates device drivers from root mode. The guest VMs should prepare specific device drivers for the assigned device and protect private data sent to the external device by themselves.

4.3.3 Cloud Management and Scheduling

To simplify the system design, MyCloud currently supports two scheduling algorithms, round-robin and simple fair-sharing. In the case of round-robin, every VMCS is set to have a fixed amount of timer expiration time before the VMENTRY. Timer expiration will trigger a VMEXIT. In current round-robin method, we only consider scheduling another VM when the timer expires. The drawback of this method is

that it lowers the overall CPU utilization if the VM does not have a lot of things to do. We also implement an algorithm close to fair-sharing that evaluates more often on whether scheduling another VM to use the CPU upon the number of VMEXITs, which will improve the CPU utilization.

In traditional cloud architecture, cloud providers will set up the control VM in the root mode. Cloud providers can control the privileges over cloud users. On the contrary, MyCloud only allows cloud providers deploy unprivileged control VM which is responsible for resource management. The management work is indirect and should be done through the interface offered by MyCloud hypervisor. Any resource allocation change requested by the control VM will be handled by the hypervisor.

Key management is out of the scope of MyCloud contribution, but a key system is necessary to ensure authentication and cloud platform verification. In order to protect the integrity of the platform, DRTM such as Intel TXT/MLE technology can be used during the boot procedure. In order to allow remote users to attest the integrity of the platform, MyCloud implements a simple key management mechanism like CloudVisor [18]. When users create a new VM, they encrypt the VM key (K_{VM}) and VM image by a public key of TPM ($K_{AIK}\{K_{VM} \mid \text{VM imange}\}$) so that only MyCloud can decrypt and verify the VM key. If the VM key is approved, MyCloud will store it in hypervisor' memory space in order to ensure that cloud provider cannot modify the VM key. Then, MyCloud will send the encrypted hash value of VM image by using the VM key ($K_{VM}\{Hash(\text{VM image})\}$to remote users. Hence, the remote users can authenticate the integrity of cloud platform.

When cloud users create a VM, MyCloud will allocate the resource under the request from the control VM. The cloud user will remotely attest the platform and negotiate a session key with MyCloud. Then, the cloud users can submit an image with the hash value encrypted by the session key to MyCloud platform. If MyCloud

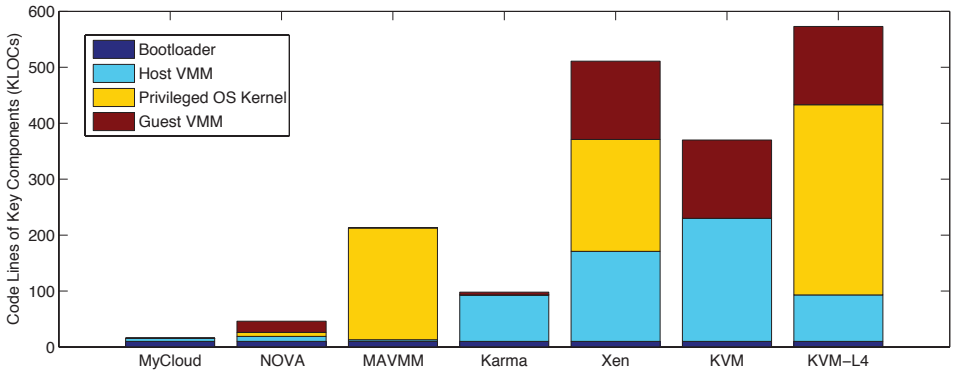


Fig. 12.: TCB size comparison of some virtualization architectures.

can successfully verify the image, it will launch the VM until users delete this VM. If the resources allocated to a guest VM are expired or no longer needed, MyCloud will destroy the data first then mark the resources as free space to the control VM. Because the control VM's memory access is restricted by the ACM and any privileged CPU or I/O instructions can be captured and check by the MyCloud hypervisor. The malicious cloud administrator is impossible to breach user's privacy in MyCloud platform.

4.4 Evaluation

The TCB size of MyCloud prototype is around 5.8K LOCs including a trusted bootloader (e.g. tboot [67]) and a verifiable cloud hypervisor. The comparison of the TCB size with other virtualization techniques is shown in Figure 12. According to my statistics, MyCloud has the smallest TCB.

Our prototype is built on a hardware platform that has an Intel i7 2600 processor (with both Vt-x and Vt-d) running at 3.3Ghz, an Intel DQ67SW Motherboard (Chip: Q67), 4 GB RAM, a 1 TB SATA HDD, and an Intel e1000 ethernet controller. We use Ubuntu 10.04 LTS with linux kernel 2.6.32 for the VM. We deploy

two bench mark tools on MyCloud prototype – lmbench [68] and compilebench [69]. In order to simplify the prototype implementation, I disable caches and symmetric multiprocessing processors (SMP) in MyCloud.

4.4.1 Performance Analysis

In order to evaluate the overheads of our platform, we compared the following five configurations.

1. Run an OS on a bare metal machine, labelled as "No_virt" in the figures.
2. Run MyCloud with only one VM, labelled as "One VM" in the figures.
3. Run MyCloud with two VMs. The light-weight Round-Robin scheduler will be triggered by VMX CPU timer and the scheduling interval is 10ms, labelled as "10ms" in the figures.
4. Run MyCloud with two VMs. The light-weight Round-Robin scheduler will be triggered by VMX CPU timer and the scheduling interval is 20ms, labelled as "20ms" in the figures.
5. Run MyCloud with two VMs. The scheduling algorithm will allow a busy VM to take more CPU time (95% CPU time) and assign an idle VM less CPU time (around 5% CPU time), labelled as "Fair Share" in the figures.

Figure 13 indicates the overhead of CPU computing performance on MyCloud. The lmbench will evaluate the CPU operations on 32 bit integers, 64 bit integers, float numbers and double. According to the result shown in figure 13, the enabling of two VMs slows down the performance by 2%, but the frequency of VM context switching does not impact the performance very much. Figure 13 also shows the

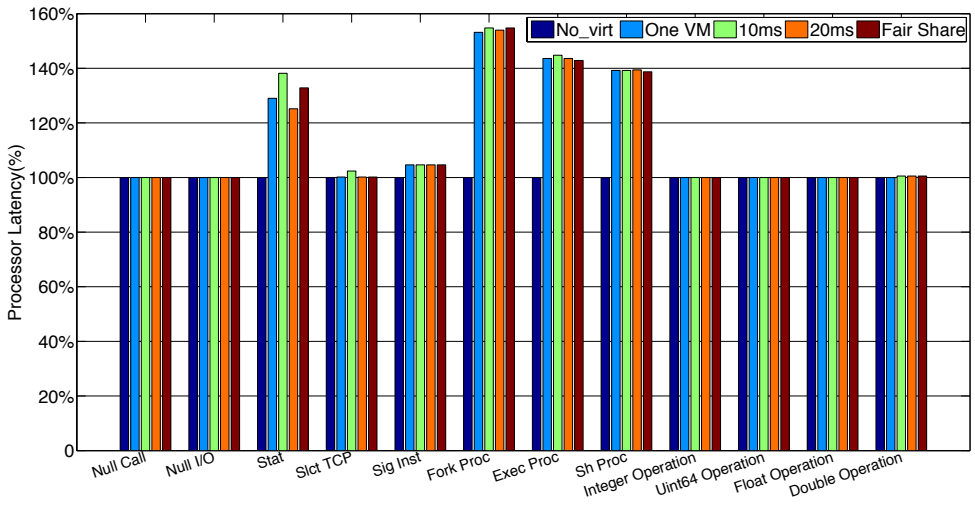


Fig. 13.: CPU latency measurements, measured by *lmbench*.

performance for popular processes like fork, exec and sh. The lmbench contains lots of context switches which have to be executed in VMX root mode. The frequent Non-root/Root mode transitions cause the performance reduction of fork and exec processes. However, in the real world, the applications in the guest VMs do not have so many context switches. Thus, the real performance of guest VMs in MyCloud should be better than what we have in the experiments.

Lmbench is also used to measure the overheads in multi-process context switching. Figure 14 shows the result of latencies when running multiple processes in guest VMs. When the number of processes increases to 16 and the data size increases to 64K, we can see the context switching efficiency based on the results in Figure 14. Since the simple scheduler algorithm in MyCloud is very simple and We disable symmetric multiprocessing and cache in our platform, the performance overhead reaches 40% in some cases. However, the real performance overhead on context switching should be much less than the results in Figure 14.

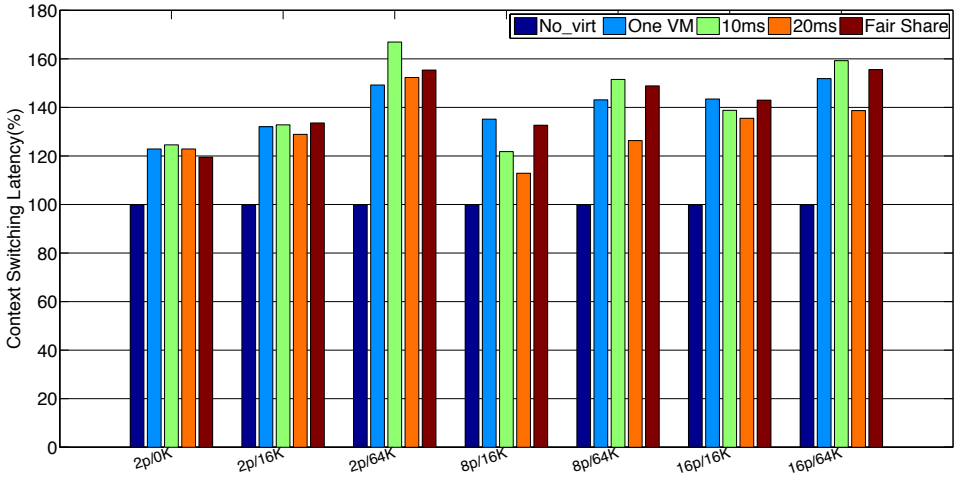


Fig. 14.: Context switch latencies measurements, measured by *lmbench*.

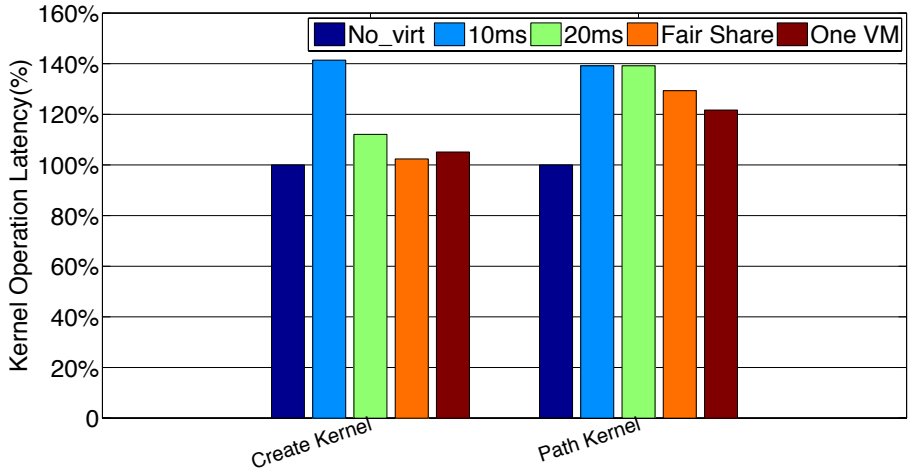


Fig. 15.: Kernel Operation latencies measurements, measured by *compilebench*.

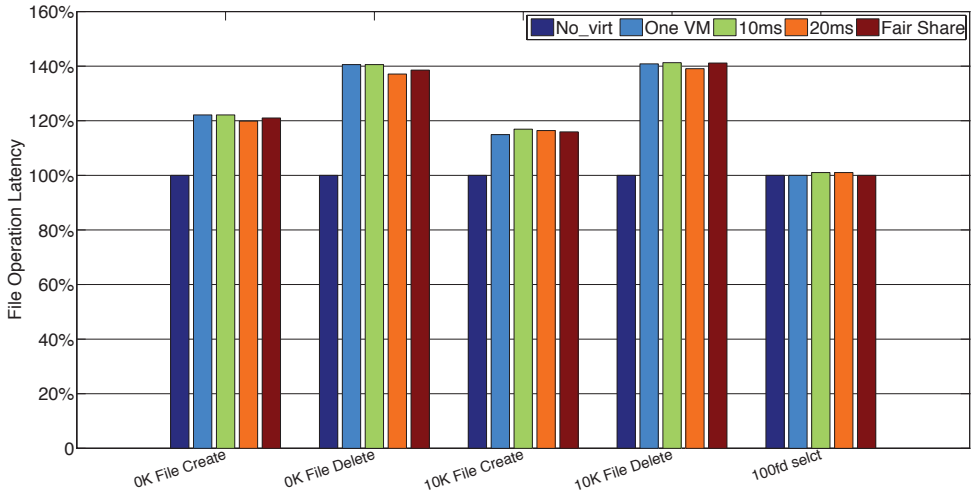


Fig. 16.: File and virtual memory latencies

Figure 15 shows the kernel operation performance measured by compilebench. The result indicates how greatly the scheduling algorithm imparts the performance. When there is only one VM, the performance loss is around 21% compared with the operating system running on bare metal machine. Kernel operation performance measurement includes operations of computing and memory read/write. Disabling caches and SMP is still the main cause of overheads.

Figure 16 and Figure 17 show the results of a comprehensive measurement of system bandwidth and latencies, including file creation/deletion and virtual memory latencies as well as local communication bandwidth. From the results we can conclude that local physical memory access (R/W), file operation and I/O operation do not have much influence on the guest VM. In addition, the VM scheduling algorithm has little contribution to the performance loss.

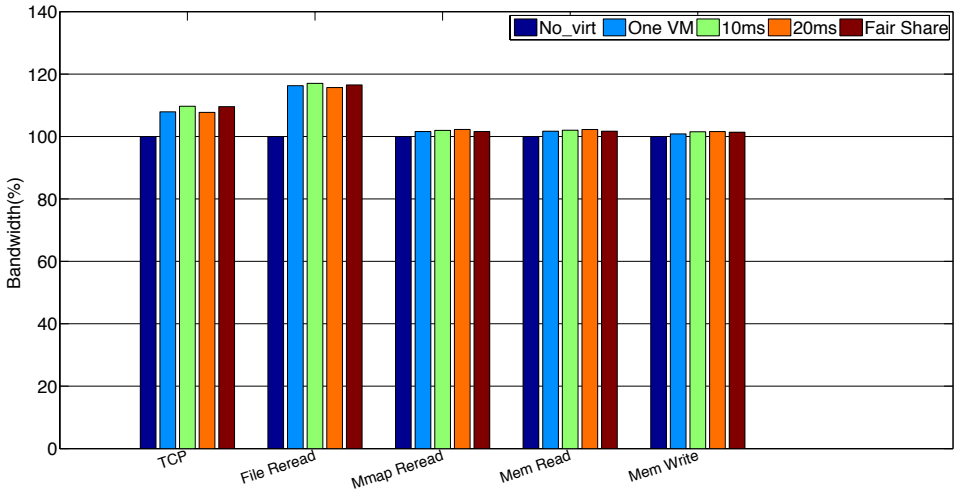


Fig. 17.: Bandwidth latencies

4.4.2 Security Analysis

Since malicious system administrators are deprived of the privileges to access users' privacy, they may hijack the hypercall and change the users' ACM when guest VMs are modifying the ACM. MyCloud can defeat against this kind of attacks because the hypervisor will manage and check all hypercall with assigned VPIDs. In MyCloud, access control specified by Table 2 is precisely and strictly followed.

The TCB size of MyCloud is greatly reduced by excluding the external drivers, management tools and complex OS kernels. As long as the TCB is secure, the privacy protection is guaranteed. Therefore, external adversaries cannot detect the vulnerabilities easily and take over the privilege of cloud provider after compromising the control VM.

The control VM is put in the non-root mode. If a malicious administrator attempts to access memory page that belong to cloud users, this activity will be intercepted through an EPT violation and handled by the hypervisor. The hypervisor will

check whether the access is authorized in the ACM table. The only interface to access memory of guest VMs is to acquire the permissions from cloud users. Therefore, the privacy breaching from malicious cloud administrators can be prevented.

Some may concern that if a VM can launch VM-to-VM Deny-of-Service (DoS) attacks by causing a lot of unauthorized memory accesses. This attack forces the hypervisor to process VMEXITs frequently, and takes CPU time slices away from the other VMs. Due to this concern, we provide a simple timer based round-robin algorithm to protect against DoS attacks. The availability is always guaranteed by round-robin since time slices are fixed for each VM.

In MyCloud, the cloud provider can only manage cloud resource allocation through the interface provided by the hypervisor. Any resource allocation requested by the control VM will be checked and handled by the hypervisor. In this way, the cloud provider cannot stealthily manipulate the users' secrets. Moreover, the control VM is not more privileged than any guest VM. Even if the control VM is compromised or exploited by inside attackers or malicious codes, the access towards the resources allocated to guest VMs will be intercepted by MyCloud hypervisor.

Intel Vt-c technology will allow physical device offer 255 virtualized interface to multiple domains. The MyCloud hypervisor will assign dedicated physical device to guest VMs by Intel Vt-d technology. In MyCloud, the cloud users should provide device drivers and protect their privacy from external device by themselves. Since a VM is usually attached to virtual or physical disks, anything stored in those disks can be accessed without the control of the VM. Thus, it is the user's responsibility to encrypt sensitive data when the data needs to be stored into any storage devices. A VM's network traffic should be treated in the same way. Since the cloud provider can always inspect user's traffic through an intrusion detection system or network management software, the users should protect their network traffic through encryption

if they have privacy concerns.

In MyCloud design, any type of physical attacks including SMM attack is not taken into consideration. SMRAM and SMM registers are assumed to be protected and set up properly. However, in order to tamper with the SMM-based attacks, we are designing a specific BIOS for MyCloud based on SeaBIOS and CoreBoot. The new BIOS can not only load hypervisor correctly, but also lock the SMRAM by setting the *D_LOCK* bit on chipset. Additionally, we remove the redundant codes for booting and initializing process, further reducing the size of TCB.

4.5 Summary

We propose a new cloud architecture – MyCloud. MyCloud can protect users' privacy from inside attack because we eliminate the privilege of cloud providers. MyCloud can also fight against external attack since we remove the control VM, device driver and management tools from the TCB of the cloud platform. In MyCloud, users can set up the protection policy in ACM, while cloud providers still manage cloud resources and access users' memory with permissions. In that case, cloud users may participate in cloud management and build mutual trust with cloud providers. Cloud providers are still able to execute privileged work on user's VM (inspect spam). We have built a prototype system of MyCloud on the x86 platform with acceptable overheads.

However, there are still some flaws in MyCloud architecture. First, many cloud users are motivated to utilize virtualized device rather than provide device drivers and manage the dedicated physical device. Assign a physical device also reduce the flexibility of platform, because users have to change the device drivers if a new physical device is added. Second, most cloud users do not have enough technique and budgets to protect their privacy from malicious device drivers. Cloud providers or hypervisor

should provide security features to protect users privacy. Hence, we propose a new architecture MyCloud SEP to solve the problems in MyCloud.

CHAPTER 5

DETANGLING RESOURCE MANAGEMENT FROM CLOUD PLATFORM WITH MYCLOUD SEP

5.1 Introduction

Recent research including MyCloud has developed new cloud architectures to protect cloud users‘ privacy. However, current cloud architecture either has limited functionalities in the hypervisor or the TCB size is too large to be protected. For example, Self Service Cloud (SSC) [15] divided the privileges of Dom0 into smaller domain. However, the smaller domains are still running in the same privilege mode as legacy Dom0. The TCB size of SSC is not reduced because SSC does not move the functionalities of Dom0 (control VM) to non-privilege mode. Our previous work MyCloud [55] can achieve a verifiable TCB size. MyCloud eliminate the privilege of the control VM and create a user configurable Access Control Matrix (ACM) in the hypervisor. However, the functionality of MyCloud hypervisor is very limited. Cloud users have to manage the external device and install drivers by themselves. This architecture design decreases the flexibility of cloud platform. Usually, cloud users do not have enough technique and budget to update cloud drivers and protect privacy from external devices.

I propose another cloud architecture – MyCloud SEP (SEP for separation) to protect users' privacy and provide device management for cloud users. In MyCloud SEP, cloud users do not need to manage real physical devices and install device driver. Instead, MyCloud SEP will include a resource allocator and real device drivers in non-root mode. MyCloud SEP hypervisor can trap the device/driver instructions based

on AHCI protocol. Like MyCloud, the cloud users can configure the policy of privacy protection on assigned device resource. If there is any malicious access on their device resource, the hypervisor will prohibit it. Such design increase the flexibility of cloud device management. In this project, we use disk management as an exmple to explain our technology. The similar scenario can be applied to other types of resource management.

In MyCloud SEP, since the component of resource management is moved to the non-root mode, the TCB size of MyCloud SEP is not greatly reduced. Compared with MyCloud, MyCloud SEP can support better functionalities without significantly increasing the TCB size. The major contribution of MyCloud SEP is as follows:

1. Protect cloud users‘ privacy from inside attack and external attack as well as provide full functionality of a hypervisor without increasing the TCB size too much.

2. Separate the resource management from privacy protection component in order to reduce the TCB size.

3. We implement a prototype MyCloud SEP and the performance evaluation shows an acceptable overheads.

5.2 Design

5.2.1 Threat Mode and Assumptions

Similar to MyCloud, the MyCloud SEP should protect user's privacy from inside and external attacks. The malicious cloud administrators may breach user's privacy by abusing the privilege of cloud providers. In addition, any mistakes they made by accident may breach user's privacy or help external adversaries to compromise

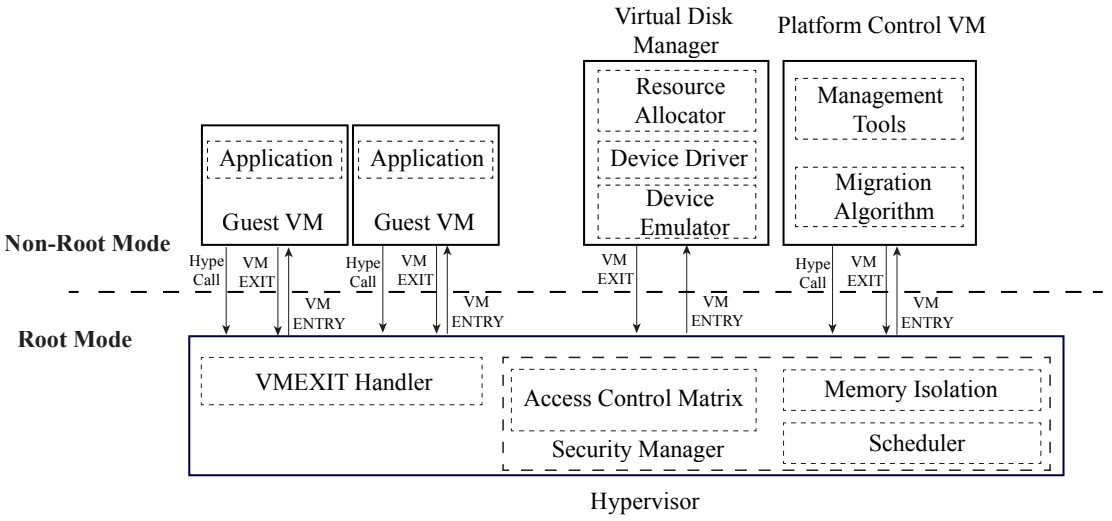


Fig. 18.: MyCloud SEP architecture Design

the cloud platform. Usually, the external adversaries can take over the cloud via the vulnerabilities found in the device drivers, device emulation and software components in the control VM.

Physical attack is also out of the scope of this paper. In this project, cloud providers can utilize Intel Trusted Execution Technology (TXT) [45] and chip-based Trusted Platform Module (TPM) [70] to verify the integrity of MyCloud SEP. Currently, many server with Intel chipset can support this measurement features. Similarly, the System Management Range Register is properly configured in order to prohibit the attack from System Management Mode (SMM).

5.2.2 Architecture Overview

Figure 18 explains the architecture of MyCloud SEP. The cloud hypervisor runs in the root mode, while other VMs or management components are running in non-root mode. After MyCloud SEP is booted, the logical processor stays in the root mode. The CPU can complete the privilege mode transition by executing specific

VMX instructions. When the guest VM executes the privileged instructions, the processor will automatically transmit to the root mode and trigger the hypervisor handler via VMEXITs.

In Figure 18, virtual machines and virtual disk manager (VDM) are launched in non-root mode. Unlike the existing techniques, VDM is not part of the TCB because all physical disks accesses made by VDM are examined by the MyCloud SEP hypervisor. In MyCloud SEP design, only the hypervisor and hardware are in the TCB. All guest VMs, control VM, device drivers and device emulator are running in the non-root mode. The hypervisor can intercept all privileged instructions executed in non-root mode. Compared with other Type 1 cloud platforms (e.g. Xen), the TCB size is remarkably reduced.

As shown in Figure 18, the hypervisor is responsible to isolate users' privacy from malicious cloud administrators and device drivers. Additionally, the hypervisor will check permission of resource access including physical memory and disks of guest VMs. The guest VMs are running as it is in bare-mental machine. The cloud users can design the policy of privacy protection on their memory and disks. Unlike MyCloud, the cloud users do not need to manage the assigned external devices and install drivers by themselves. The virtual disk manager (VDM) is used to support device virtualization and manage device drivers to implement I/O operations. The hypervisor can invoke the VDM by injecting specific interrupt for I/O operations. The control VM in MyCloud SEP can design the migration strategy and check the platform resource via hyperCall. The functionality of each component in MyCloud SEP will be explained the following sections.

5.2.2.1 MyCloud SEP Hypervisor

The hypervisor is the only component running in the root mode and owning the privilege to access all cloud resources. Before the hypervisor is launched, a trustable boot loader should verify the integrity of the execution environment using Intel TXT technology. The initialization process of hypervisor needs to complete the following tasks.

1. Detect E820 map and allocate the available physical memory for each component.
2. Detect all PCI devices installed in cloud platform.
3. Configure EPT in order to isolate guest VM' memory from the control VM in order to prohibit the access made by malicious cloud administrators.
4. Configure DMA Remapping Page Table in order to isolate the memory access space of external device. Thus, the memory space assigned to cloud users can be protected from malicious devices.
5. Copy the hypervisor into specific memory space.

After the initialization process, the hypervisor should complete the following tasks in order to launch the guest VMs, the control VM and virtual disk manager.

1. Create VMCS structure for the control VM, guest VMs and Virtual Disk Manager.
2. Create Access Control Matrix and Resource Allocation Recorder.
3. Allocate resources (e.g. memory and disk) to guest VMs.
4. Schedule the guest VMs.

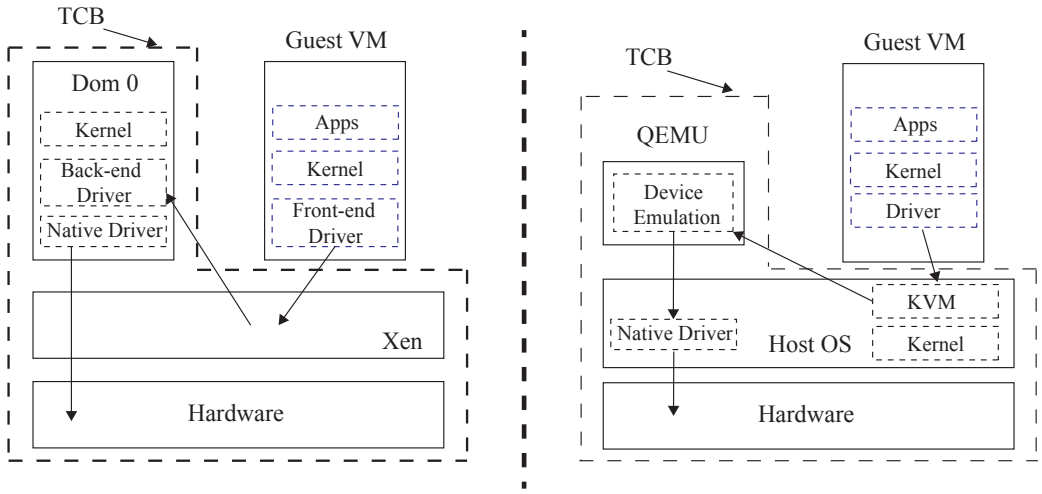


Fig. 19.: Device management in KVM and Xen

5.2.2.2 Virtual Disk Manager

As shown in Figure 19, both KVM and Xen rely on real device drivers installed in the privileged VM to communicate with physical disks. The hypervisor will intercept the device operations in guest VMs and forward this activity to the privileged VM. Xen deploys a split-driver mechanism to deliver the I/O operation, but KVM will rely on the existing linux system call. After the privileged VM completes I/O operation, the hypervisor will inject the result to the guest VM. Since the privileged VM is running in the root-mode, neither the hypervisor nore guest VMs can monitor how the device drivers complete the I/O operation.

I implement a MyCloud SEP prototype to explain how to separate disk management from security management. The virtual disk structure in MyCloud SEP is illustrated in Figure 20. Each virtual machine including the control VM only have access to limited number of disks in the virtual disk pool. The virtual disk manager (VDM) manages the disk resources.

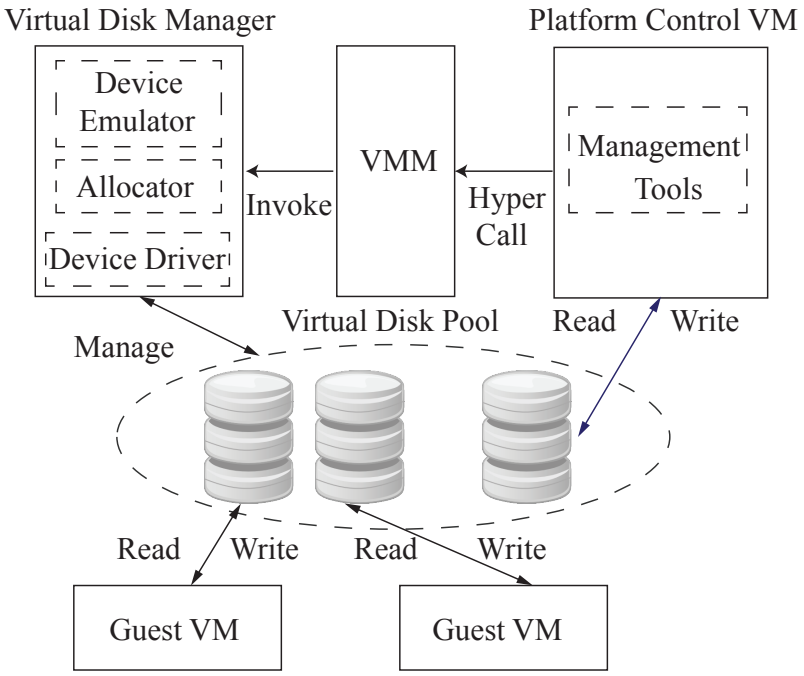


Fig. 20.: Virtual Disk Management

All disk access will be checked by the hypervisor against the ACM. Since, the device drivers and resource allocator work in non-root mode, MyCloud SEP will grant an access if the access is permitted in the ACM. During the initialization process of a VM, the device drivers need a lot of information such as manufacture ID, etc,. MyCloud SEP will intercept these instructions and provide an emulated device to guest VMs.

Since the device drivers may breach user's privacy, MyCloud SEP needs to monitor I/O operations from device drivers. In MyCloud SEP, the VDM is just a piece of codes which provides Intel AHCI [71] emulation and communicates with local SATA disks. The design reduce the attack surface by running the VDM in non-root mode. In order to monitor the activity of disk drivers, the hypervisor will also create a VMCS structure and configure which instructions should be intercepted.

5.2.2.3 Control VM

In MyCloud SEP, the control VM is launched in non-root mode. The hypervisor will create VMCS for VMCS so that any memory access not in its EPT table will be trapped by the hypervisor. If the guest VM does not grant cloud providers access permissions, the hypervisor will prohibit the memory access of the control VM.

MyCloud SEP allows the control VM manage resources allocation and check resource utilization through HyperCall API. The control VM can migrate VMs as long as it follows resource allocation procedures and the migration plan does not violate policies specified in ACM.

5.2.2.4 Guest VM

Although guest VMs are running the non-root mode, cloud users can implement privileged work such as memory introspection. Also, the cloud users can modify the privacy protection policy via HyperCall provided by the hypervisor. Normally, the guest VMs are running as the same way in physical machine. The hypervisor will trap all privileged instructions of guest VMs and resume VMs after completing the security check.

5.3 Implementation

5.3.1 Access Control on I/O operations

The privilege design of MyCloud SEP is different from existing cloud platform, since the control VM does not have privileges over the user's privacy. In MyCloud SEP design, the control VM is removed from the root mode and all access permissions are set up in the ACM by cloud users. MyCloud SEP hypervisor relies on Intel Extended Page Table (EPT) technology to intercept any memory accesses. Also, MyCloud

Table 3.: Access Control Matrix in MyCloud SEP (VDM-Virtual Disk Manager, CVM-Control Virtual Machine, H-Hyper Calls, R-Read, W-Write, P- Permission Required)

Components	$Hypervisor$	CVM	VDM	$ResourceRegion_i$	$ResourceRegion_j$
$Hypervisor$	Full	Full	Full	Full	Full
CVM	H	Full		P	P
VDM	H		Full		
VM_i	H				Full
VM_j	H			Full	

SEP use Intel VT-d technology [72] [73] [74] to monitor all I/O operations. Once a resource access is invoked by guest VM, Virtual Disk Manager or the control VM, the hypervisor will verify the permission over ACM.

Like MyCloud, the MyCloud SEP still maintains an Access Control Matrix which is configurable by users. Table 3 shows the details of ACM in MyCloud SEP. The ACM stores access permissions for each VM and resource regions. Cloud users are assigned special HyperCall to set up the ACM. In ACM, we use Virtual Disk Manager (VDM) as an example of resource manager which can only access to the allocated resource after the hypervisor verify the permission.

As shown in Talbe 3, only the hypervisor has full access rights to all resources in MyCloud SEP. The control VM is assigned the same privilege level as guest VMs. Therefore, the cloud administrator can only access resources shared by cloud users. If the cloud administrator needs to access user resources, it has to acquire the permission from cloud users. If the cloud users allow cloud providers access their privacy, they can change the ACM by a series of hyperCalls. Besides, VDM is responsible to provide device emulator and complete I/O instructions of guest VMs. All activities of VDM are under the control of hypervisor which can verify the access permission of VDM

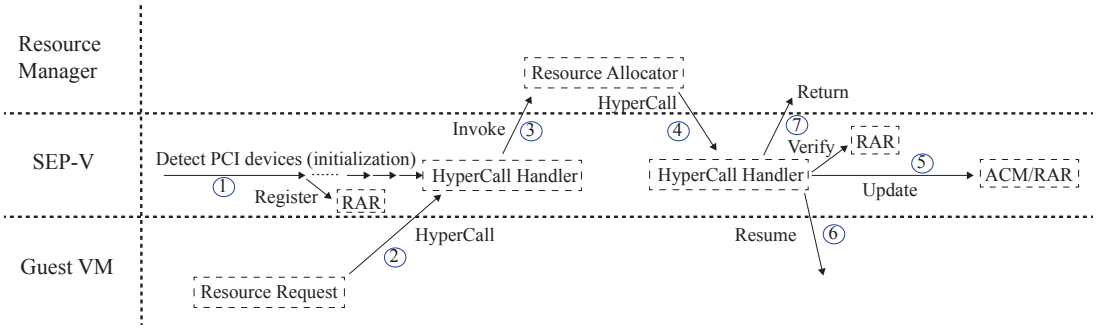


Fig. 21.: The Workflow of Updating ACM .

against ACM.

In MyCloud SEP, the resources are managed in the unit of a "resource region" as shown in Table 3. A resource region is specified by `{start address, end address}`. A region is not necessary to be the full address space for a VM. For example, a VM can have a disk block $ResourceRegion_i$ `{(track #100, head #0, sector #15), (track #500, head #0, sector #15)}`.

Figure 21 introduces the procedure of how hypervisor assign a free block of resource to the guest VM. In step 1, when the hypervisor initializes the hardware, it sends I/O commands to port `0xcf8` and `0xcfc` in order to obtain the configuration of each PCI device. The acquired information is packaged in PCI device structures including base address (BAR), specified command and I/O ports etc,. The hypervisor will allocate the memory space for each device and register these allocation information in a data structure – Resource Access Recorder (RAR).

In step 2, the guest VM sends a HyperCall to the hypervisor in order to apply new resource region. To improve the compatibility for different resource allocators and reduce the TCB size, MyCloud SEP allows multiple resource allocators in the non-root mode. The HyperCall handler will send VMLAUNCH instructions to invoke the resource allocators in step 3. The resource allocator will generate the allocation

plan and inform the hypervisor by another HyperCall in step 4. Since the resource allocator is not trusted, the hypervisor will verify the allocation plan by checking the RAR table. The hypervisor should guarantee the allocator only assign free resources to the guest VM. If the plan is approved, the hypervisor will update the RAR and ACM table in step 5. In step 6, the hypervisor will resume the guest VM with a new allocated resource region. Finally, the hypervisor resume the execution of resource manager in step 7.

The process to free a resource region is similar. First of all, a guest VM sends the request to the hypervisor by HyperCall. The hypervisor invokes the resource allocator, then verifies the security of new resource allocation plan by searching the RAR table and checking ACM. Finally, the hypervisor will resume the guest VM and resource manager after updating the ACM table.

5.3.2 Resource Management

Figure 20 explain the resource management in MyCloud SEP by using disks management. The control VM is constrained in the non-root mode, thus has to access virtual disks as the same way as guest VMs. During the boot of guest VMs, OS will request device information such as *device ID, mentor ID etc,*. These requests will be trapped into the hypervisor then handled by a device emulator. In this stage, the device emulator will offer virtualized device information to support the boot of guest VM.

In order to manage the disk allocation, the MyCloud SEP implement a linear mapping from a logical disk space a physical disk space. Figure 22 explains how the physical disk blocks are mapped to virtual disks. The hypervisor track each physical block by three parameters: cylinder, sector and head. When users try to expand the size of virtual disks, the hypervisor should allocate free physical disks to virtual disks

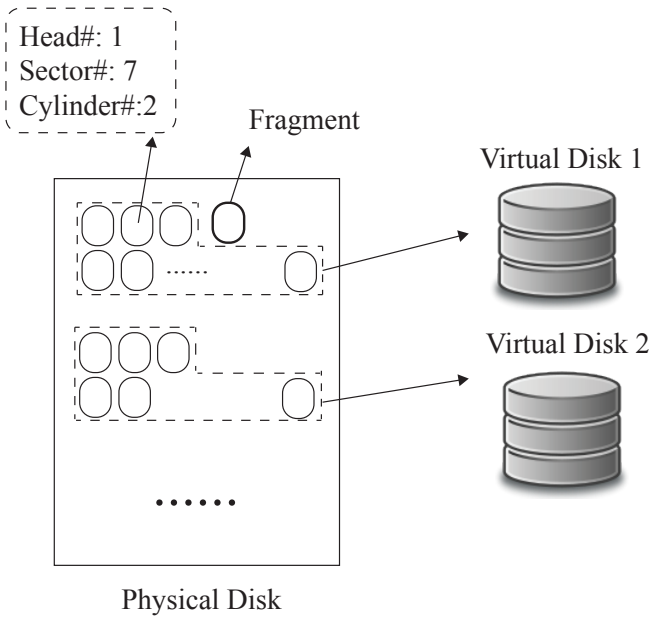


Fig. 22.: Physical disk assignment.

and update the Resource Access Recorder (RAR).

MyCloud SEP should not only verify the security of resource allocation plan made by VDM, but also monitor activity of drivers when completing I/O operations. In MyCloud SEP, we rely on AHCI protocol which is widely used in communication with Intel SATA disks. The hypervisor can understand the Advanced Host and Controller Interface (AHCI) [71] information throughout PCI configuration space (0xcf8 and 0xcfc). The RAR table will be used to store these allocation information such as base address, AHCI specific I/O port and registers etc,. When the users request new disk spaces, the VDM will decide which part of physical disks can be assigned. Then, the hypervisor can update the ACM and RAR table. Since the device drivers complete I/O operations based on AHCI and MyCloud SEP understand the AHCI 1.3 specification, each I/O command sent from drivers will be trapped and traversed by the hypervisor. Afterwards, the hypervisor will verify the access permission against

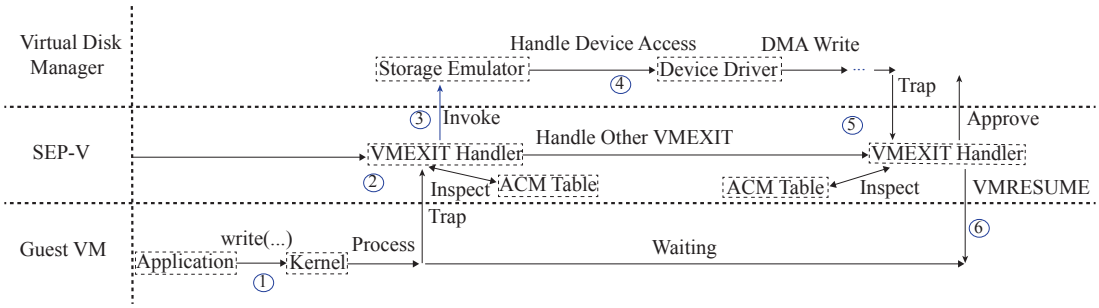


Fig. 23.: Workflow of I/O operation.

ACM and only execute the approved I/O instructions.

In nature, the AHCI encompasses a PCI device and the AHCI bus adapter is composed of a PCI header and PCI Capabilities. In the booting stage of guest VM, the OS will try to acquire the PCI configuration by sending I/O command to port *0xcf8* and *0xcfc*. The hypervisor can set up the VMCS and ask CPU to intercept any I/O commands sent to both ports. Therefore, the hypervisor can handle the I/O commands and implement device emulation.

Figure 23 explains how MyCloud SEP verify the access permission of I/O commands. We use I/O write as an example. According to the AHCI specification, When an application in the guest VM sends a disk write request to OS kernel, the kernel will issue a series of I/O commands to configure the specific I/O ports and transfer data with AHCI HBA. Because the hypervisor can configure the VMCS and trap these I/O commands. Then, The hypervisor will check if the trapped I/O commands has permission to the resource in ACM table. . After approved, the hypervisor will trigger the VDM and deliver the command to the device emulator. The VDM handles the commands and calls physical disk drivers to execute the I/O write operation.

In order to transfer data from memory to disk, the untrustworthy device drivers in VDM has to access physical memory and I/O ports. The hypervisor will also verify

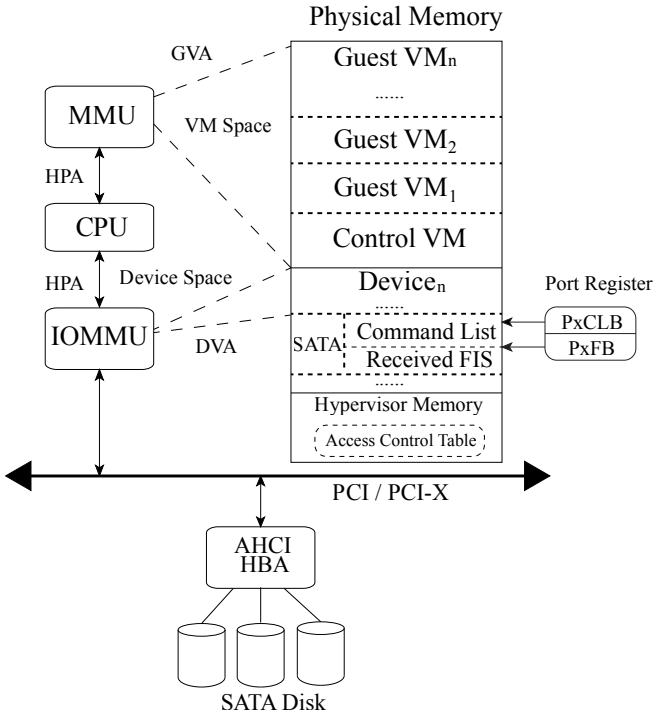


Fig. 24.: Device and VM isolation.

the permission of resource access made by device drivers. If the trapped I/O command indicates the disk is ready to transfer data, the hypervisor will assign the physical disk to the VDM using DMA remapping technology. MyCloud SEP will configure the remapping table in order to prohibit the drivers access resources of other VMs.

To prevent VDM drivers from reconfiguring the device via I/O command, the hypervisor records the boundary of each VM in resource region when users send I/O commands to prepare disk operations. If the access is out of the scope of users-specified resourced region, VMEXIT will be caused and the hypervisor will block the command. After VDM finishes the write operation, hypervisor can resume the guest VM.

5.3.3 Memory Isolation

5.3.3.1 Memory Access Isolation

MyCloud SEP should isolate the memory of each guest VM from the control VM relying on the Intel Extended Page Table (EPT) technology. In the memory of hypervisor, MyCloud SEP will build a 4-layer EPT table for each VM before users create the guest VM. The *EPT base pointer* in VMCS is configured by the hypervisor to record the entry address of EPT table. When a memory translation is made by the kernel of guest VM, Memory Management Unit (MMU) will traverse the EPT table and translate the Guest Virtual Address (GVA) to Host Physical Address (HPA). Since there is no overlapped host physical memory space in EPT table, any guest VM cannot access the memory space assigned to other VMs. If a page fault happens in the guest VM, it can be trapped by the hypervisor through VMEXIT. The hypervisor can update the EPT by adding a free page into the EPT table then resume the execution of the guest VM.

5.3.3.2 Device Access Isolation

Most of devices rely on IOMMU to translate Device Virtual Address (DVA) to Host Physical Address (HPA) and use DMA to deliver data between devices and memory. In order to protect users' privacy from malicious devices and drivers, MyCloud SEP implements Intel Virtualization Technology for Directed I/O and trap the I/O commands by configuring the VMCS. Before devices execute DMA access, MyCloud SEP will elaborately build Context-Entry Table (CET) in IOMMU to implement DMA Remapping for each device. IOMMU users $\langle PCIbus, deviceandfunction \rangle$ to index the CET table and find the Multi-Level Page Table to translate the DVA. Although the CPU cannot control the DMA access, the hypervisor still receives VMEX-

ITs if the device access the memory unmapped in Multi-Level Page Table. In our prototype, we implement the IOMMU access isolation for SATA disks.

5.3.3.3 RAR Isolation

Figure 24 explains how to implement the isolation in MyCloud SEP. Besides the users' privacy, MyCloud SEP also protects Memory Mapped I/O space, PCI device configuration space and MSR mapped space. For each PCI device, the data and I/O command lists are stored in Command List and Received FIS as shown in Figure 24. The entry point is specified at chipset register PxCLB and PxFB. The hypervisor will set up these registers and assign EPT-mapped memory to the devices. In order to protect PCI configuration space, the hypervisor will monitor all I/O commands related to 0xcfc and 0xcf8. Therefore, the hypervisor can prohibit the malicious drivers from compromising these I/O ports.

5.4 Evaluation

Table 4 shows the specification of evaluation platform. In order to evaluation the overheads of MyCloudSEP on I/O instructions, we test the number of VMexits and time consumption when creating 1GB empty file with 4KB block size and 8KB block size. Besides, we test the overheads on CPU instruction and memory access by benchmark. lmbench [68] [75].

We design four test cases as follows:

- **No_Virt**: Run one OS in the bare mental machine.
- **One_VM**: Run MyCloud SEP with one VM.
- **Round_Robin**: Run MyCloud SEP with two VMs. Scheduling interval is 10ms and the switch is triggered by VMX preemption timer.

Table 4.: Evaluation Platform Specification

CPU	Intel i7 2600
Motherboard	Intel DQ67SW chipset
Memory	4GB
Disk	1TB SATA 7200rpm
Operating System	Ubuntu 10.04
Kernel Version	2.6.32

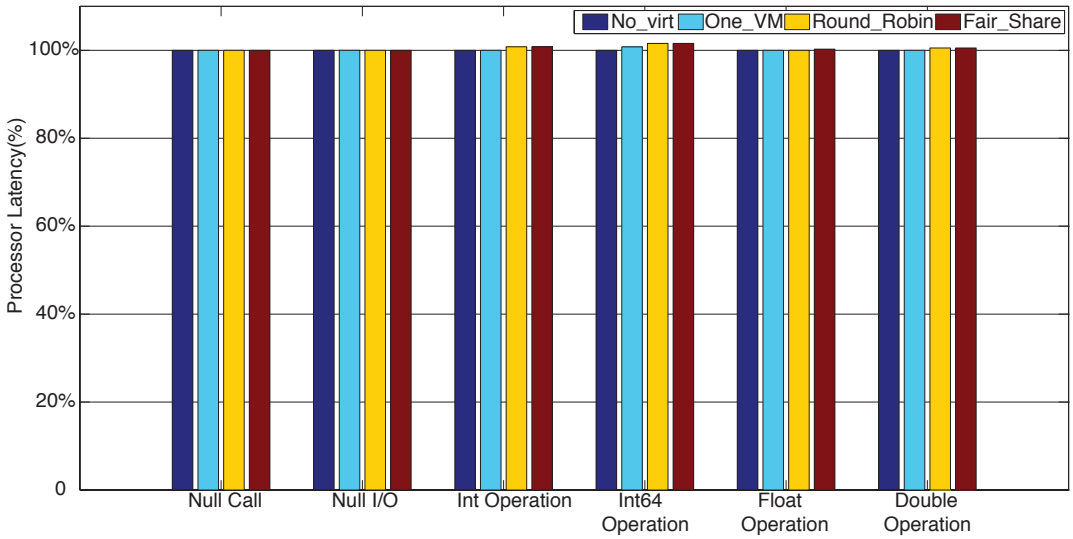


Fig. 25.: The overhead of CPU instructions

- **Fair_Share**: Run MyCloud SEP with two VMs. One busy VM takes over 95% CPU time and the other VM is only assigned 5% CPU time. The scheduler is also triggered by VMX preemption timer.

5.4.1 CPU Instructions

lmbench benchmark is able to test the time consumption when MyCloud SEP executes some popular CPU instructions. For example, CALL, I/O operation, Int operation, Float operation and double operation. According to the result of bench-

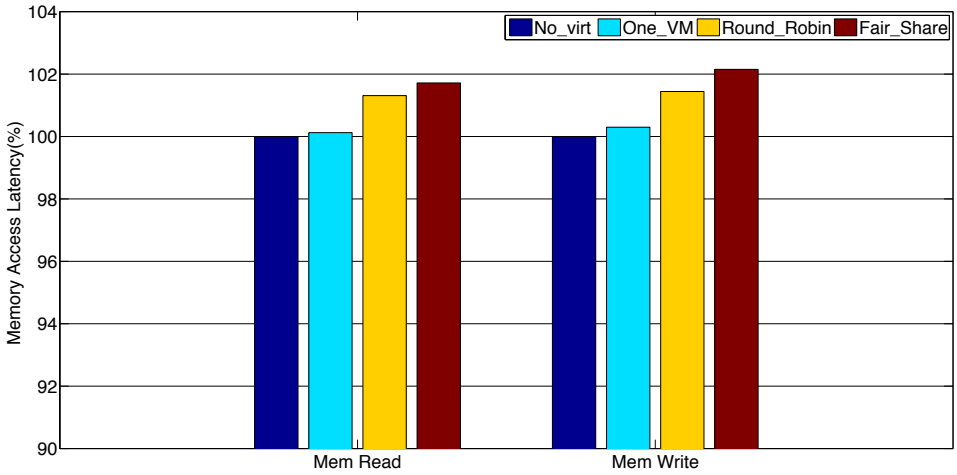


Fig. 26.: The overhead of memory access

mark evaluation shown in Figure 25, MyCloud SEP design can barely increase the overhead when the system executes a CPU instruction. Only when CPU instruct 64bit operations, MyCloud SEP will cause extra overhead.

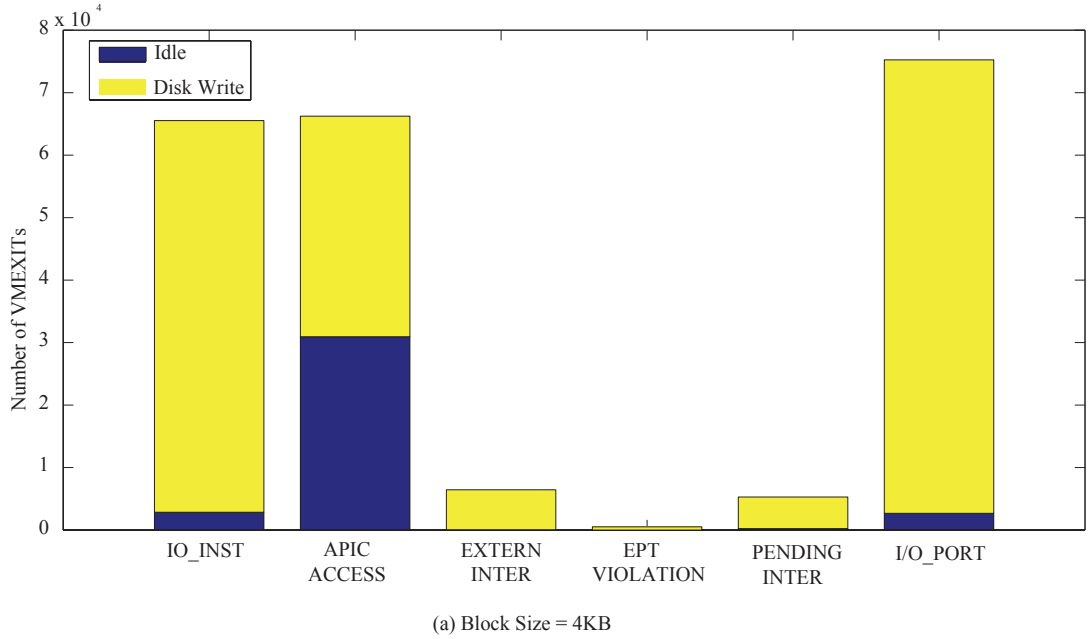
5.4.2 Memory Access

Figure 26 shows the evaluation results on memory access made by lmbench benchmark. When there is one VM running in MyCloud SEP, the performance of memory R/W is not impacted. Because we only deploy simple round-robin and fair-share scheduler algorithm, the performance overhead is 2%.

5.4.3 I/O Operation

Figure 27 shows the type and numbers of VMEXITs when MyCloud SEP creates one 1GB empty file with 4KB block size. The guest VM will introduce 2×10^5 VMEXITS. Most of them are caused by I/O instructions.

Figure 28 introduces the similar evaluation results when MyCloud SEP creates



(a) Block Size = 4KB

Fig. 27.: Number of VMEXITs on creating 1GB file with 4KB bloksize

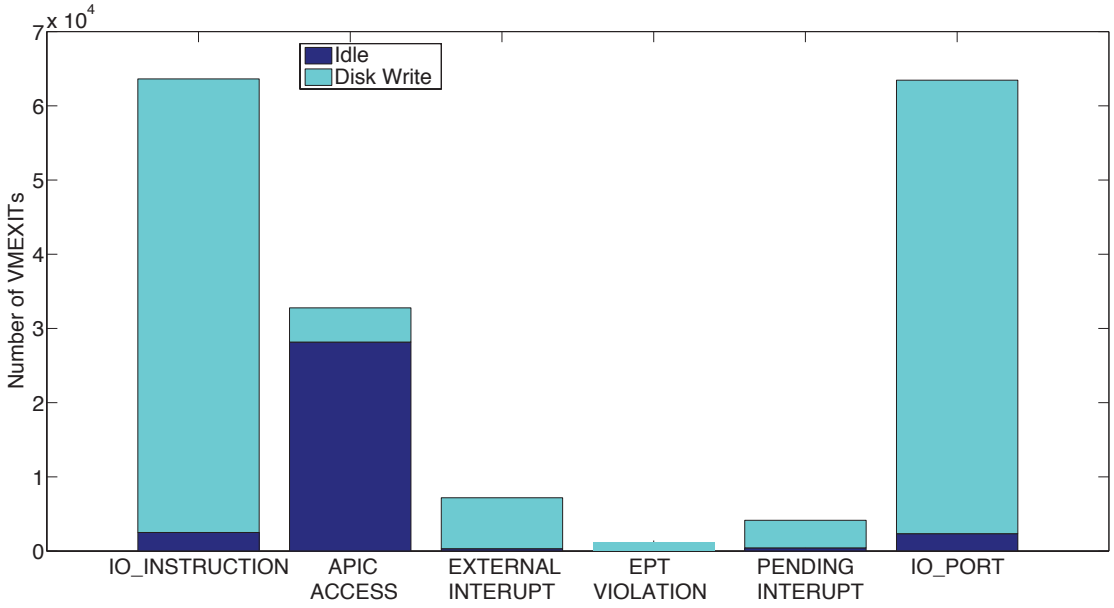


Fig. 28.: Number of VMEXITs on creating 1GB file with 8KB bloksize

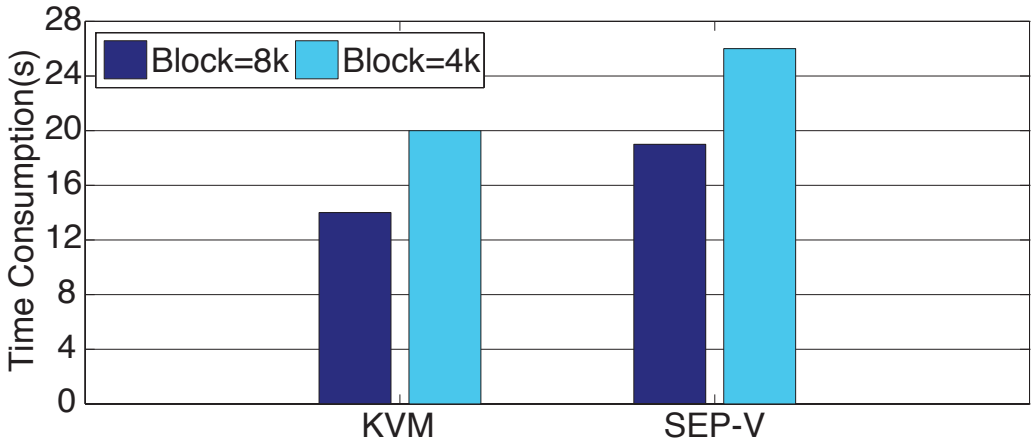


Fig. 29.: Time Consumption for Disk Operations

one 1GB empty file with 8KB block size. Because the block size is bigger, guest VM will generate less VMEXITS (1.38×10^5). Although the number of VMEXITS looks huge, the extra overhead is acceptable.

Figure 29 compares the MyCloud SEP with KVM on time consumption of creating the 1GB file. We also set the block size as 4KB and 8KBM on both of them. MyCloud SEP takes 20% more time than KVM. The overhead is caused because I/O operations will be trapped by hypervisor and examined against ACM. In addition, the evaluation result shows that the bigger block size is, the less VMEXITs will be caused. The time consumption with 8KB block size is less than that of 4KB block size.

5.5 Security Analysis

The goal of MyCloud SEP design is to fully protect the users' privacy from both inside attack and external attack. MyCloud SEP offers guest VMs the ability to configure the resource access permission in ACM and share the resource with others by a series of HyperCalls. In MyCloud SEP, the control VM, guest VMs and virtual

disk manager can only access the assigned resources. Any illegal resource access and attempt ti modify ACM will be detected and prohibited by the hypervisor.

5.5.1 Inside Attack

5.5.1.1 Cloud Administrator

The malicious cloud administrator may abuse the privilege of the control VM to disclose users' privacy [58]. In MyCloud SEP design, the control VM is running in the non-root mode and under the control of hypervisor. If the administrator tries to dump the users' memory, a memory read instruction should be executed. CPU will trap this instruction and the hypervisor will stop the memory read activity because it violates the permission in ACM. Therefore, the malicious cloud administrator cannot access a guest VM space unless the guest VM explicitly grants the access through the ACM.

5.5.1.2 Applications of Guest VMs

In the previous design, the attacker may compromise users' applications and gain the privilege in the guest VM. Cloud users is also possible to install malicious applications in their VMs. For example, attack [76] shows that the vulnerabilitiy in JRE 6 is able to allow context-dependent attackers to gain privileges via malicious application or applet. Afterwards, the attacker will acquire the privileges to read and write local files. In MyCloud SEP, the hypervisor will trap the activities of writing and reading local files. Since the attacker is not approved in the ACM, this attack will not compromise users' privacy.

5.5.1.3 Device Driver

In current cloud design, the malicious drivers can compromise the functionalities of guest VMs. For example, the backend driver in Xen allows malicious guest OS users to cause a denial of service via a kernel thread leak, which prevents the device and guest OS from being shut down or create a zombie domain [77].

MyCloud SEP can protect the cloud users from this attack, because the VMs scheduling is control by the hypervisor. MyCloud SEP replies on a preemption timer in VMCS in order to allow other VMs execute CPU instructions. Since the verifiable hypervisor is running in the root mode, the scheduler can work properly.

5.5.1.4 Management Tools

Due to the vulnerability of management tool in the control VM, the malicious users may access to the management functionalities and cause a DoS attack. For example [78], use-after-free vulnerability in the function of Xen 4.1.x through 4.3.x, when using a multithreaded toolstack, does not properly handle a failure by the xc_cpumap_alloc function, which allows local users with access to management functions to cause a denial of service (heap corruption) and possibly gain privileges via unspecified vectors. [79] shows that cross-site scripting (XSS) vulnerability in webaccess in VMware allows attackers to inject arbitrary web script via vectors related to context data.

In MyCloud SEP, both management tools of the control VM and the guest VMs are running in the non-root mode. Only the hypervisor can call invoke the control VM and call the management function by VMRESUME. If the malicious VM tries to access the functions of the control VM, the hypervisor will intercept and prohibit the *call* instruction from guest VMs, because the guest VM is against the permission

in ACM.

5.5.1.5 Malicious Cloud Users

Due to the cloud business model, each person has a chance to deploy guest VMs on cloud providers. Even the attacker can set up a guest VMs and compromise other guest VMs in the same cloud. For example, [80] [6] vulnerability in the virtual machine display functions in VMware Workstation allows guest OS users to execute arbitrary code on host OS. Therefore the attacker could execute malicious code on host OS and disclose the privacy of other legitimate VMs.

In MyCloud SEP, this attack will not happen because the display functions of host OS are running in the non-root mode. Even if the host OS is compromised, the attacker cannot acquire any privacy information of other VMs. All resources accesses are under the control of ACM in the hypervisor. The attackers in the non-root mode are impossible to detect and compromise the hypervisor.

5.5.2 External Attack

The most of external attacks come from malicious guest VMs, targeting at the hypervisor, innocent cloud users and the control VM. In MyCloud SEP, users' privacy is isolated from other components in the cloud. The TCB component of MyCloud SEP is the light-weighted hypervisor, which can be verified by Intel TXT technology. Only the hypervisor is granted the full privileges to access all resources. The malicious guest VMs are running in the non-root mode, any access to other VMs and the control VM will be intercepted by the hypervisor. Compromising a guest VM or other components out of the hypervisor does not gain access to any other guest VMs since the ACM is maintained and enforced by the hypervisor. The only interface to access other VM's space is acquiring the permission in ACM after approved users modifying the access

ruls by HyperCall.

The same protection goes with disk drivers and device emulator. The disk drivers are in the VDM, the control VM cannot directly send malicious I/O commands or interrupts to access guest VMs. Only the hypervisor can invoke the VDM to handle the trapped I/O commands.

The attackers cannot breach users privacy through PCI devices either. MyCloud SEP configure the IOMMU to translate the VDA, therefore, any malicious DMA access will be prohibited by the hypervisor. The hypervisor first identifies all PCI devices at initialization process. Then, the hypervisor records MMIO and PCI Configuration space in RAR for each device. In that case, MyCloud SEP can prevent the attackers from overlapping the device memory to disclose users' private data.

5.6 Summary

In MyCloud SEP design, we separate resource allocation and management in hypervisor. MyCloud SEP is able to provide the resource management modules, but the TCB size of the hypervisor is significantly reduced. In our design, the hypervisor deploy ACM fro the resource manager, control VM and guest VM in order to identify the resource access. Hence, the privacy in guest VM is protected. In MyCloud SEP, the functionality and privacy protection is also separated. We implement a prototype by using disk management as an example. The performance shows acceptable overheads in MyCloud SEP.

CHAPTER 6

CONCLUSION AND FUTURE WORK

In this dissertation, I have presented two types of approaches to protect users' privacy in cloud. The migration based approach is compatible with current existing cloud and protect user' privacy with the help of cloud providers. According to my evaluation, the migration based approach can improve the security of guest VMs. The cloud providers do not need to update their hypervisor and the management tools do not need to be revised in order to use special HyperCall to communicate with hypervisor.

However, the migration based approach cannot protect users' privacy from inside attacks, such as the malicious cloud administrator, vulnerable management tool and illegal device drivers. The hardware-based approach (MyCloud and MyCloud SEP) proposes a redesigned cloud hypervisor to eliminate the privileges of cloud providers and isolate users' privacy from device and control VM. Any unapproved access to users' privacy will be trapped and prohibited by the hypervisor. Based on the experiment on our prototype, this approach can provide cloud users a secure execution environment. Cloud users only need to trust the verifiable hypervisor and the hardware resources (e.g. CPU, motherboard). Although the cloud providers need to replace their hypervisor with MyCloud or MyCloud SEP, the performance overhead is acceptable. In MyCloud SEP, we even move the disk management to the non-root mode. Therefore, the hypervisor not only monitors the memory resources but also, prohibits any malicious disk access. Cloud users do not need to manage the device drivers by themselves, instead cloud platform can assign them a virtualized device.

The approach in this dissertation has laid a solid foundation towards trustworthy

virtualization systems. It creates various opportunities for future work. In the following, I propose a new direction to protect users' privacy on an untrusted hypervisor. In the future, the hardware mentor will offer more and more security features on CPU and motherboard. CPU will automatically encrypt and decrypt the instructions and users' data. Then, the guest VMs can be executed by an untrusted cloud hypervisor. In current stage, we just implement a prototype with simple scheduler. In the future, we will upgrade the scheduler and support more VMs in MyCloud and MyCloud SEP.

Appendix A

ABBREVIATIONS

Vt-x	Intel CPU Virtualization Technology
Vt-d	Intel Virtualization Technology for Directed I/O
EPT	Extended Page Table
VM	Virtual Machine
GPA	Guest Physical Address
HPA	Host Physical Address
GVA	Guest Virtual Address
DVA	Device Virtual Address
MMU	Memory Management Unit
IOMMU	IO Memory Management Unit
VMM	Virtual Machine Manager
ACM	Access Control Matrix
RAR	Resource Allocation Recorder
IaaS	Infrastructure as a service
PaaS	Platform as a service
SaaS	Software as a service
Dom 0	Domain 0
OS	Operating System
TCB	Trust Computing Base

VMX	Intel Virtualization Technology
VMCS	Virtual Machine Control Structure
DMA	Direct Memory Access
DTMC	District Time
NVD	National Vulnerability Database
CVSS	Common Vulnerability Scoring System
ADG	Attack Dependency Graph
SMM	System Management Mode
SMRR	System Range Register
TXT	Trusted Execution Technology
LOC	Lines of Codes
VPID	Virtual Processor ID
SMP	Symmetric Multiprocessing Processors
DoS	Denial of Service
TPM	Trusted Platform Module
VDM	Virtual Disk Manager
SATA	Serial ATA
AHCI	Advanced Host Controller Interface
PCI	Peripheral Component Interconnect
CET	Context-Entry Table
MSR	Model-Specific Register

KVM Kernel-based Virtual Machine Module

REFERENCES

[1] Xen. *http://www.xen.org/.*

[2] KVM. *http://www.linux-kvm.org/.*

[3] VMware ESXi. *http://www.vmware.com/products/esxi-and-esx/overview.*

[4] Azure. *http://www.windowsazure.com/en-us/.* Mirosoft Inc.

[5] US Cloud Providers May Lose 35 Billion dollar Due to PRISM. *http://cloudtimes.org/2013/08/cloud-providers-may-lose-35-billion-due-to-prism/.*

[6] CVE-2007-4993. *Xen guest root escape to dom0 via pygrub.*

[7] CVE-2009-1758. *The hypervisor callback function in Xen, as applied to the Linux kernel 2.6.30-rc4 allows guest user applications to cause a denial of service of the guest OS by triggering a segmentation fault in certain address ranges.*

[8] CVE-2010-0431. *QEMU-KVM in RedHat Enterprise Virtualization (RHEV) 2.2 and KVM 83, does not properly validate guest QXL driver pointers, which allows guest OS users to gain privileges via unspecified vectors.*

[9] *Xen multiple vulnerability report.* `http://secunia.com/advisories/44502/`. Secunia.

[10] R Wojtczuk and J Rutkowska. "Xen 0wning trilogy". In: *Black Hat Conference.* 2008.

[11] Thomas Ristenpart et al. "Hey, You, Get off of My Cloud: Exploring Information Leakage in Third-party Compute Clouds". In: *Proceedings of the 16th ACM Conference on Computer and Communications Security.* CCS '09. Chicago, Illi-

nois, USA: ACM, 2009, pp. 199–212. ISBN: 978-1-60558-894-0. DOI: 10.1145/1653662.1653687. URL: http://doi.acm.org/10.1145/1653662.1653687.

[12] *Intel® CPU Virtualization Technology*. http://www.intel.com/content/www/us/en/virtualization/virtualization-technology/intel-virtualization-technology.html. Intel Corporation.

[13] Z.I.M. Yusoh and Maolin Tang. "A penalty-based genetic algorithm for the composite SaaS placement problem in the Cloud". In: *Evolutionary Computation (CEC), 2010 IEEE Congress on*. July 2010, pp. 1–8. DOI: 10.1109/CEC.2010.5586151.

[14] J.S. Chase et al. "Dynamic virtual clusters in a grid site manager". In: *High Performance Distributed Computing, 2003. Proceedings. 12th IEEE International Symposium on*. June 2003, pp. 90–100. DOI: 10.1109/HPDC.2003.1210019.

[15] Shakeel Butt et al. "Self-service Cloud Computing". In: *Proceedings of the 2012 ACM Conference on Computer and Communications Security*. CCS '12. Raleigh, North Carolina, USA: ACM, 2012, pp. 253–264. ISBN: 978-1-4503-1651-4. DOI: 10.1145/2382196.2382226. URL: http://doi.acm.org/10.1145/2382196.2382226.

[16] Eric Keller et al. "NoHype: Virtualized Cloud Infrastructure Without the Virtualization". In: *SIGARCH Comput. Archit. News* 38.3 (June 2010), pp. 350–361. ISSN: 0163-5964. DOI: 10.1145/1816038.1816010. URL: http://doi.acm.org/10.1145/1816038.1816010.

[17] B. Dolan-Gavitt et al. "Virtuoso: Narrowing the Semantic Gap in Virtual Machine Introspection". In: *Security and Privacy (SP), 2011 IEEE Symposium on*. May 2011, pp. 297–312. DOI: 10.1109/SP.2011.11.

[18] Fengzhe Zhang et al. “CloudVisor: Retrofitting Protection of Virtual Machines in Multi-tenant Cloud with Nested Virtualization”. In: *Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles*. SOSP ’11. Cascais, Portugal: ACM, 2011, pp. 203–216. ISBN: 978-1-4503-0977-6. DOI: 10.1145/2043556.2043576. URL: http://doi.acm.org/10.1145/2043556.2043576.

[19] Michael Sindelar, Ramesh K. Sitaraman, and Prashant Shenoy. “Sharing-aware algorithms for virtual machine colocation”. In: *Proceedings of the 23rd ACM symposium on Parallelism in algorithms and architectures*. SPAA ’11. San Jose, California, USA: ACM, 2011, pp. 367–378. ISBN: 978-1-4503-0743-7. DOI: 10.1145/1989493.1989554. URL: http://doi.acm.org/10.1145/1989493.1989554.

[20] L. Grit et al. “Virtual Machine Hosting for Networked Clusters: Building the Foundations for ”Autonomic” Orchestration”. In: *Virtualization Technology in Distributed Computing, 2006. VTDC 2006. First International Workshop on*. Nov. 2006, p. 7. DOI: 10.1109/VTDC.2006.17.

[21] L. Ramakrishnan et al. “Toward a Doctrine of Containment: Grid Hosting with Adaptive Resource Control”. In: *SC 2006 Conference, Proceedings of the ACM/IEEE*. Nov. 2006, p. 20. DOI: 10.1109/SC.2006.64.

[22] J.L. Lucas Simarro et al. “Dynamic placement of virtual machines for cost optimization in multi-cloud environments”. In: *High Performance Computing and Simulation (HPCS), 2011 International Conference on*. July 2011, pp. 1–7. DOI: 10.1109/HPCSim.2011.5999800.

[23] Yulong Zhang et al. “Incentive compatible moving target defense against vm-colocation attacks in clouds”. In: *Information Security and Privacy Research*.

Springer, 2012, pp. 388–399.

[24] Anh M. Nguyen et al. “MAVMM: Lightweight and Purpose Built VMM for Malware Analysis”. In: *Proceedings of the 2009 Annual Computer Security Applications Conference*. ACSAC ’09. Washington, DC, USA: IEEE Computer Society, 2009, pp. 441–450. ISBN: 978-0-7695-3919-5. DOI: 10.1109/ACSAC.2009.48. URL: http://dx.doi.org/10.1109/ACSAC.2009.48.

[25] J.M. McCune et al. “TrustVisor: Efficient TCB reduction and attestation”. In: *Security and Privacy (SP), 2010 IEEE Symposium on*. IEEE. 2010, pp. 143–158.

[26] Udo Steinberg and Bernhard Kauer. “NOVA: A Microhypervisor-based Secure Virtualization Architecture”. In: *Proceedings of the 5th European Conference on Computer Systems*. EuroSys ’10. Paris, France: ACM, 2010, pp. 209–222. ISBN: 978-1-60558-577-2. DOI: 10.1145/1755913.1755935. URL: http://doi.acm.org/10.1145/1755913.1755935.

[27] Lenin Singaravelu et al. “Reducing TCB complexity for security-sensitive applications: three case studies”. In: *SIGOPS Oper. Syst. Rev.* 40.4 (Apr. 2006), pp. 161–174. ISSN: 0163-5980. DOI: 10.1145/1218063.1217951. URL: http://doi.acm.org/10.1145/1218063.1217951.

[28] Amit Vasudevan et al. “Design, Implementation and Verification of an eXtensible and Modular Hypervisor Framework”. In: *Proceedings of the 2013 IEEE Symposium on Security and Privacy*. SP ’13. Washington, DC, USA: IEEE Computer Society, 2013, pp. 430–444. ISBN: 978-0-7695-4977-4. DOI: 10.1109/SP.2013.36. URL: http://dx.doi.org/10.1109/SP.2013.36.

[29] T. Garfinkel et al. “Terra: A virtual machine-based platform for trusted computing”. In: *ACM SIGOPS Operating Systems Review*. Vol. 37. 5. ACM. 2003,

pp. 193–206.

[30] Takahiro Shinagawa et al. “BitVisor: A Thin Hypervisor for Enforcing I/O Device Security”. In: *Proceedings of the 2009 ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments*. VEE ’09. Washington, DC, USA: ACM, 2009, pp. 121–130. ISBN: 978-1-60558-375-4. DOI: 10.1145/1508293.1508311. URL: http://doi.acm.org/10.1145/1508293.1508311.

[31] Yu-Yuan Chen, Pramod A. Jamkhedkar, and Ruby B. Lee. “A Software-hardware Architecture for Self-protecting Data”. In: *Proceedings of the 2012 ACM Conference on Computer and Communications Security*. CCS ’12. Raleigh, North Carolina, USA: ACM, 2012, pp. 14–27. ISBN: 978-1-4503-1651-4. DOI: 10.1145/2382196.2382201. URL: http://doi.acm.org/10.1145/2382196.2382201.

[32] Derek Gordon Murray, Grzegorz Milos, and Steven Hand. “Improving Xen Security Through Disaggregation”. In: *Proceedings of the Fourth ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments*. VEE ’08. Seattle, WA, USA: ACM, 2008, pp. 151–160. ISBN: 978-1-59593-796-4. DOI: 10.1145/1346256.1346278. URL: http://doi.acm.org/10.1145/1346256.1346278.

[33] B. Kauer, P. Verissimo, and A. Bessani. “Recursive virtual machines for advanced security mechanisms”. In: *Dependable Systems and Networks Workshops (DSN-W), 2011 IEEE/IFIP 41st International Conference on*. IEEE. 2011, pp. 117–122.

[34] M. Ben-Yehuda et al. “The Turtles project: Design and implementation of nested virtualization”. In: *Proceedings of the 9th USENIX conference on Operating systems design and implementation*. USENIX Association. 2010, pp. 1–6.

[35] D. Williams, H. Jamjoom, and H. Weatherspoon. “The Xen-Blanket: virtualize once, run everywhere”. In: *ACM EuroSys* (2012).

[36] Wuqiong Pan et al. “Improving Virtualization Security by Splitting Hypervisor into Smaller Components”. In: *Proceedings of the 26th Annual IFIP WG 11.3 Conference on Data and Applications Security and Privacy*. DBSec’12. Paris, France: Springer-Verlag, 2012, pp. 298–313. ISBN: 978-3-642-31539-8. DOI: 10.1007/978-3-642-31540-4_23. URL: http://dx.doi.org/10.1007/978-3-642-31540-4_23.

[37] Richard Ta-Min, Lionel Litty, and David Lie. “Splitting interfaces: making trust between applications and operating system configurable”. In: *Proceedings of the 7th symposium on Operating systems design and implementation*. OSDI ’06. Berkeley, CA, USA: USENIX Association, 2006, pp. 279–292.

[38] Zhi Wang et al. “Isolating Commodity Hosted Hypervisors with HyperLock”. In: *Proceedings of the 7th ACM European Conference on Computer Systems*. EuroSys ’12. Bern, Switzerland: ACM, 2012, pp. 127–140. ISBN: 978-1-4503-1223-3. DOI: 10.1145/2168836.2168850. URL: http://doi.acm.org/10.1145/2168836.2168850.

[39] Intel Inc. *Intel 64 and IA-32 Architectures Software Developer Manuals*. 2009.

[40] Ahmed M. Azab et al. “HyperSentry: Enabling Stealthy In-context Measurement of Hypervisor Integrity”. In: *Proceedings of the 17th ACM Conference on Computer and Communications Security*. CCS ’10. Chicago, Illinois, USA: ACM, 2010, pp. 38–49. ISBN: 978-1-4503-0245-6. DOI: 10.1145/1866307.1866313. URL: http://doi.acm.org/10.1145/1866307.1866313.

[41] Jiang Wang, Angelos Stavrou, and Anup Ghosh. “HyperCheck: A Hardware-assisted Integrity Monitor”. In: *Proceedings of the 13th International Confer-*

ence on Recent Advances in Intrusion Detection. RAID'10. Ottawa, Ontario, Canada: Springer-Verlag, 2010, pp. 158–177. ISBN: 3-642-15511-1, 978-3-642-15511-6. URL: http://dl.acm.org/citation.cfm?id=1894166.1894178.

[42] A.M. Azab, P. Ning, and X. Zhang. "SICE: a hardware-level strongly isolated computing environment for x86 multi-core platforms". In: *Proceedings of the 18th ACM conference on Computer and communications security*. ACM. 2011, pp. 375–388.

[43] K. Sun et al. "Secureswitch: Bios-assisted isolation and switch between trusted and untrusted commodity oses". In: *Proceedings of the 19th Annual Network and Distributed System Security Symposium*. 2012.

[44] Jonathan M. McCune et al. "Flicker: an execution infrastructure for tcb minimization". In: *SIGOPS Oper. Syst. Rev.* 42.4 (Apr. 2008), pp. 315–328. ISSN: 0163-5980. DOI: 10.1145/1357010.1352625. URL: http://doi.acm.org/10.1145/1357010.1352625.

[45] Intel Coperation. *Intel Trusted Execution Technology*. 2011.

[46] M. Price. "The Paradox of Security in Virtual Environments". In: *Computer* 41.11 (Nov. 2008), pp. 22–28. ISSN: 0018-9162. DOI: 10.1109/MC.2008.472.

[47] J. Szefer et al. "Eliminating the hypervisor attack surface for a more secure cloud". In: *Proceedings of the 18th ACM conference on Computer and communications security*. ACM. 2011, pp. 401–412.

[48] Xioaxin Chen et al. "Overshadow: A virtualization-based approach to retrofitting protection in commodity operating systems". In: *In ASPLOS*. May 2008.

[49] Jisoo Yang and Kang G. Shin. "Using hypervisor to provide data secrecy for user applications on a per-page basis". In: *Proceedings of the fourth ACM SIGPLAN/SIGOPS international conference on Virtual execution environments*. VEE '08. New York, NY, USA: ACM, 2008, pp. 71–80.

[50] Owen S. Hofmann et al. "Inktag: secure applications on an untrusted operating system". In: *Proceedings of the eighteenth international conference on Architectural support for programming languages and operating system*. ASPLOS '13. New York, NY, USA: ACM, 2013, pp. 265–278.

[51] Yueqiang Cheng, Xuhua Ding, and Robert H. Deng. "AppShield: Protecting applications against untrusted operating system". In: *Singaport Management University Technical Report*. smu-sis-13-101. 2013.

[52] Lionel Litty, H. Andrés Lagar-Cavilla, and David Lie. "Computer meteorology: monitoring compute clouds". In: *Proceedings of the 12th conference on Hot topics in operating systems*. HotOS'09. Monte Verità, Switzerland: USENIX Association, 2009, pp. 4–4. URL: http://dl.acm.org/citation.cfm?id=1855568.1855572.

[53] NVD. *National Vulnerability Database*. 2012.

[54] CVSS. *Common Vulnerability Scoring System*. 2012.

[55] Min Li et al. "MyCloud: Supporting User-configured Privacy Protection in Cloud Computing". In: *Proceedings of the 29th Annual Computer Security Applications Conference*. ACSAC '13. New Orleans, Louisiana: ACM, 2013, pp. 59–68. ISBN: 978-1-4503-2015-3. DOI: 10.1145/2523649.2523680. URL: http://doi.acm.org/10.1145/2523649.2523680.

[56] EC2. *http://aws.amazon.com/ec2/*. Amazon Inc.

[57] Carl Bagh. *Sony PlayStation Network attack shows Amazon EC2 a hackers' paradise*. http://www.ibtimes.com/articles/146224/20110516/. 2011.

[58] Tom Krazit. *CNET News. Google fired engineer for privacy breach*. http://news.cnet.com/8301-30684_3-20016451-265.html.

[59] Craig Gentry and Shai Halevi. *Implementing Gentry's Fully-Homomorphic Encryption Scheme*. Cryptology ePrint Archive, Report 2010/520. http://eprint.iacr.org/. 2010.

[60] Dan Boneh, Gil Segev, and Brent Waters. "Targeted malleability: homomorphic encryption for restricted computations". In: *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*. ITCS '12. Cambridge, Massachusetts: ACM, 2012, pp. 350–366. ISBN: 978-1-4503-1115-1. DOI: 10.1145/2090236.2090264. URL: http://doi.acm.org/10.1145/2090236.2090264.

[61] Tal Garfinkel, Mendel Rosenblum, et al. "A Virtual Machine Introspection Based Architecture for Intrusion Detection." In: *NDSS*. Vol. 3. 2003, pp. 191–206.

[62] R. Wojtczuk and J. Rutkowska. "Attacking SMM memory via Intel CPU cache poisoning". In: *Invisible Things Lab* (2009).

[63] *Intel® Virtualization Technology Specification for Connectivity*. http://www.intel.com/content/www/us/en/network-adapters/virtualization.html. Intel Corporation.

[64] Gerwin Klein et al. "seL4: formal verification of an OS kernel". In: *Proceedings of the ACM SIGOPS 22nd symposium on Operating systems principles*. SOSP '09. Big Sky, Montana, USA: ACM, 2009, pp. 207–220. ISBN: 978-1-60558-752-3. DOI: 10.1145/1629575.1629596. URL: http://doi.acm.org/10.1145/

1629575.1629596.

[65] Advanced Micro Devices. *AMD64 Architecture Programmer s Manual Volume 2: System Programming*. Dec. 2011.

[66] Intel Corporation. *Intel® PCI-SIG SR-IOV Primer: An Introduction to SR-IOV Technology*. Jan. 2011.

[67] Trusted Boot. *http://sourceforge.net/projects/tboot/*.

[68] Larry McVoy and Carl Staelin. "lmbench: portable tools for performance analysis". In: *Proceedings of the 1996 annual conference on USENIX Annual Technical Conference*. ATEC '96. San Diego, CA: USENIX Association, 1996, pp. 23–23. URL: http://dl.acm.org/citation.cfm?id=1268299.1268322.

[69] compilebench. *https://oss.oracle.com/ mason/compilebench/*.

[70] Intel Coperation. *Intel Trusted Platform Module*. 2003.

[71] Intel Coperation. *Serial ATA Advanced Host Controller Interface*. 2012.

[72] *Intel® Virtualization Technology Specification for Directed I/O Specification*. www.intel.com/technology/vt/. Intel Corporation.

[73] Advanced Micro Devices. *AMD I/O Virtualization Technology (IOMMU) Specification*. Feb. 2009.

[74] AMD Inc. *AMD SB800-Series Southbridges Register Reference Guide*. May 2011.

[75] Larry McVoy and Carl Staelin. "lmbench: portable tools for performance analysis". In: *Proceedings of the 1996 annual conference on USENIX Annual Technical Conference*. ATEC '96. San Diego, CA: USENIX Association, 1996, pp. 23–23. URL: http://dl.acm.org/citation.cfm?id=1268299.1268322.

[76] CVE-2008-3107. *The Virtual Machine in Sun Java Runtime Environment (JRE) in JDK and JRE 6 before Update 7, JDK and JRE 5.0 before Update 16 as well as SDK and JRE 1.4.x allows context-dependent attackers to gain privileges via an untrusted application or applet. This can potentially allow attackers to read, write or execute local programs or files.*

[77] CVE-2010-3699. *he backend driver in Xen 3.x allows guest OS users to cause a denial of service via a kernel thread leak, which prevents the device and guest OS from being shut down or create a zombie domain, causes a hang in zenwatch, or prevents unspecified xm commands from working properly, related to (1) netback, (2) blkback, or (3) blktap.*

[78] CVE-2014-1950. *Use-after-free vulnerability in the function in Xen 4.1.x through 4.3.x, when using a multithreaded toolstack, does not properly handle a failure by the function, which allows local users with access to management functions to cause a denial of service (heap corruption) and possibly gain privileges via unspecified vectors.*

[79] CVE-2009-2277. *Cross-site scripting (XSS) vulnerability in WebAccess in VMware allows attackers to inject arbitrary web script via vectors related to context data.*

[80] CVE-2009-1244. *Vulnerability in the virtual machine display function in VMware Workstation allows guest OS users to execute arbitrary code on host OS.*