Detangling Resource Management Functions from the TCB in Privacy-Preserving Virtualization

Min Li[1], Zili Zha[1], Wanyu Zang[1], Meng Yu[1], Peng Liu[2], Kun Bai[3]

Virginia Commonwealth University[1]
Pennsylvania State University, University Park[2]
IBM T.J. Watson Research Center[3]
{lim4, zhaz, wzang, myu}@vcu.edu[1], pliu@ist.psu.edu[2], kunbai@us.ibm.com[3]

Abstract. Recent research has developed virtualization architectures to protect the privacy of guest virtual machines. The key technology is to include an access control matrix in the hypervisor. However, existing approaches have either limited functionalities in the hypervisor or a Trusted Computing Base (TCB) which is too large to secure. In this paper, we propose a new architecture, MyCloud SEP, to separate resource allocation and management from the hypervisor in order to reduce the TCB size while supporting privacy protection. In our design, the hypervisor checks all resource accesses against an access control matrix in the hypervisor. While providing flexibility of plugging-in resource management modules, the size of TCB is significantly reduced compared with commercial hypervisors. Using virtual disk manager as an example, we implement a prototype on x86 architecture. The performance evaluation results also show acceptable overheads.

Key words: Cloud Computing, Privacy Protection, TCB Minimization, Decomposition, Isolation

1 Introduction

While more and more companies deploy their service in clouds that provide scalable and effective computing resources, privacy concerns may lead to cloud market loss up to $35 billion by 2016 [1]. The primary cause of security and privacy concerns is the privilege design in existing cloud platforms. On current cloud platforms, such as Xen [2], KVM [3], and Amazon EC2 [4], the control Virtual Machine (VM) has administrative privileges for resource management. Consequently, both the hypervisor and the control VM are running in the processor's root mode that has the most privileges. Unfortunately, such architecture design gives no chance to the cloud clients to protect their privacy. Furthermore, 1) it enables insider attacks from the cloud administrators; 2) the control domain can evade detection of malicious behaviors; and 3) the Trusted Computing Base (TCB) includes both the control domain and the hypervisor, which is too large to secure.

In order to solve the privacy protection problems, recent research such as Self-Service Cloud (SSC) [5] proposed to divide the privileges of Dom0 (control VM) into smaller domains including MTSD domains and user domains. The smaller domains are running in the same processor privilege as legacy Dom0. The TCB size of such design is still very large because SSC does not move the third-part drivers and control VM to a non-privileged mode. Our previous work MyCloud [6] achieves a verifiable TCB size with only 6K LOCs by removing the control VM from the processor root mode. We create a user configurable Access Control Matrix (ACM) in the hypervisor to protect the privacy of guest VMs. However, the functionalities of the hypervisor in MyCloud are very limited.

In this paper, we propose an innovative structure, MyCloud SEP (SEP for separation), to solve the separation of functionality and security check. In our design, we put resource allocator and management outside the hypervisor. Security checks are included in the hypervisor. Such design enables the flexibility of resource management. In this paper, we use virtual disk management as an example to explain our technology. The same approach can be applied to other types of resource management in virtualization platforms.

In MyCloud SEP, since the control VM and resource managers are moved to the processor's non-root mode, the new structure reduces the TCB by an order of magnitude (the size is similar to that of MyCloud) compared with commercial hypervisors. Compared with our previous work, the new architecture supports better functionalities without significantly increasing the TCB size. In summary, our new contributions are: 1) To the best of our knowledge, this is the first effort to separate resource allocation from security checks in order to reduce the hypervisor size; 2) The proposed architecture enables privacy protection and full functionality of a hypervisor without significantly increasing the TCB size; and 3) Our performance evaluations show acceptable overheads.

The rest of the paper is organized as follows. Section 2 discusses related work. Section 3 clarifies assumptions and threat model, and describes our proposed architecture. Section 4 describes the detailed implementations. The experimental results are presented in Section 5 . Section 6 discusses how different threats are handled. Finally, Section 7 concludes the paper.

2 Related Work

In traditional cloud platforms, the cloud provider owns full privileges over the VMM and users' VMs, providing no way for the cloud users to protect their own privacy. To address the threats from the administrative domain, previous research has been focused on shrinking the TCB either by disaggregation of privileges functionality of the control domain [5,7] or by splitting VMM into smaller components based on nested virtualization[8]. Self-Service Cloud computing (SSC) [5] allows client VMs to execute some management of privileges, which used to be provided in administrative domain. SplitVisor [8] splits VMM into a smaller part as the minimized TCB to enforce isolation and a larger part to provide rich service functionality. Nevertheless, this design is not compatible with current cloud computing schemes because the cloud users are required to upload a specialized guest VMM.

Similar to SplitVisor, some approaches investigate the use of nested virtualization to disaggregate some host VMM components to the guest VMM [9,10,11]. CloudVisor [9] introduces a small security monitor underneath the VMM to enforce strict isolation among the VMM and the host VMs using nested virtualization. According to our understanding, CloudVisor's late launch includes the host operating system of KVM as part of the TCB, though it is not explicitly stated. Hence, the TCB is still too large due to the large code base of the whole operating system. Moreover, to deploy nested virtualization on x86 hardware imposes tremendous performance penalties that increase exponentially with nesting depth [12].

To reduce the size of the TCB even further, NOVA [13,14] constructs a microkernel based VMM with 9K LOCs. Nonetheless, Its TCB is not markedly decreased since the microhypervisor is still in charge of complex management tasks, such as address space allocation, interrupt and exception handling. Therefore, the thin TCB is still difficult to verify dynamically. Compared with this, NoHype [15,16] narrows down the attack surface of the hypervisor by dynamically eliminating VMM layer. However, the number of VMs that can run simultaneously on the physical platform are restricted since it requires one-VM-per-core on multi-core processors and pre-allocated nested page table. Flicker [17] is considered as a privacy protection solution based on the hardware features provided by the hardware vendors, like Intel and AMD. It significantly enhances the security and reliability of the code while at the same time inducing large performance overhead. Other than that, it only offers application level protection and is not a general solution for VMs in cloud.

Besides above architectural improvement attempts, many research efforts focus on protecting the privacy of user application against untrusted operating system using a VMM-based approach [18,19,20,21,22]. 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.

Our previous work, MyCloud, achieves a verifiable TCB size (6K LOCs) by removing the control VM from the processor root mode. It also has a flexible privacy protection mechanism based on a user configured ACM. MyCloud isolates the memory space among guest VMs, physical devices and the hypervisor. However, the functionalities of the hypervisor are limited, e.g., needs device level support of virtualization. To remove the restrictions and better support physical devices, we propose a design that launches resource managers in the non-root mode. The procedure and results of resource management can be monitored by the hypervisor in the root mode. Through this design, MyCloud SEP provides better functionalities without significantly increasing the hypervisor size.

3 MyCloud SEP Architecture

3.1 Threat Model and Assumptions

We take *insider attacks* into consideration but we must distinguish the *cloud administrators* from the *cloud providers*. Generally, the famous cloud providers such as Amazon [4], Microsoft [23] and Hewlett-Packard [24] have strong motivation to protect users' privacy rather than reveal customers' privacy. Protecting users' privacy will increase the reputation of cloud enterprises to a large extent and bring more economic benefits. On the contrary, the cloud administrators employed by the cloud providers may be motivated to disclose cloud tenants' privacy to pursue monetary benefits. Moreover, any mistakes they make by accident may breach users' privacy or help external attackers to compromise guest VMs. Therefore, we consider the cloud administrators malicious.

Due to many vulnerabilities from the device drivers, device emulation and software components in the control VM [25,26,27], the external adversary can compromise the control VM and obtain the administrative privilege of the cloud platform. Afterwards, the external adversary will exploit cloud tenants' private data. Meanwhile, the external adversary can also breach the cloud users' privacy relying on the vulnerabilities found in current virtual machine monitors (VMM) design [28,29,30,31,32]. Furthermore, the console interface provided by the cloud provider is also vulnerable to many *external attacks* [33,34].

In MyCloud SEP design, we take both insider and external attacks into consideration. But the physical attack [35] is out of the scope of this paper. The cloud provider can solve the physical attack by deploying more protection mechanisms on the server side such as secure door control system.

In this paper, we assume that the cloud providers can utilize Intel Trusted Execution Technology (TXT) [36] and chip-based Trusted Platform Module (TPM) [37] to measure the integrity of the hypervisor execution environment before MyCloud SEP is loaded. This is not a strong assumption since now all servers are using the technology or similar ones. Similarly, we assume that the System Management Range Register (SMRR) is properly configured in order to protect the processor System Management Mode (SMM) from attacks [38].

We will not discuss how to make a mutually agreed access control policy between the cloud providers and cloud tenants in this paper. It is up to the cloud providers and cloud users to decide which part of memory can be accessed. MyCloud SEP just provides isolated execution environment and mechanisms to implement the access control policies.

3.2 Virtualization Architecture

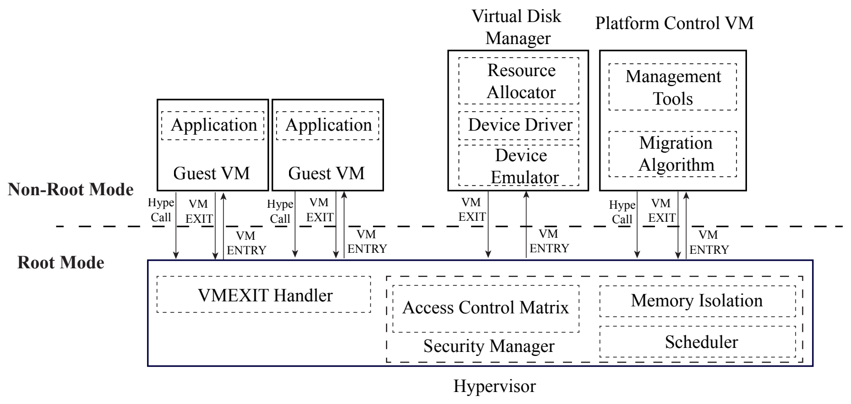


Fig. 1: MyCloud SEP architecture design.

The architecture of MyCloud SEP is shown in Figure 1. Using Intel virtualization technology [?], the software stack of MyCloud SEP is divided into *root* mode and *non-root* mode. Each mode in MyCloud SEP has the same ring privilege structure from ring 0 to ring 3. As shown in Figure 1, the hypervisor runs in the root mode, while other components run in non-root mode. When the hypervisor is booted, MyCloud SEP will stay in the root mode. The CPU will enter the non-root mode, when the hypervisor executes VMRESUME/VMLAUNCH instruction. If the guest VMs execute the privileged instructions, CPU will automatically transfer to the root-mode and trigger hypervisor handlers via VMEXITs. After the hypervisor handles the privileged instruction, the guest VM can be resumed.

In Figure 1, the Platform Control VM is moved to non-root mode and a Virtual Disk Manager (VDM) launched in non-root mode will drive physical disks. Different from existing techniques, VDM is not part of the TCB and the access to the physical disks will be examined by the hypervisor against an ACM in the hypervisor. In MyCloud SEP design, only the hypervisor and platform hardware are in the TCB. The TCB size is remarkably reduced because there is no operating system, physical device drivers and device emulator running in the privileged mode and the hypervisor will intercept all privileged instruction executed by the components in non-root mode.

Note that our architecture is different from Xen since the control VM is moved out of the processor's root mode. Also, different from other designs, we are not trying to put device management in a separate domain. Instead, our design goal is to put resource management outside the TCB. In the figure, we only show virtual disk management since in cloud environment, we usually need much less device support than a desktop computer does.

Device Management In this paper, we use virtual disks as an example to explain how to separate resource management from security management in the hypervisor. The virtual disk structure in MyCloud SEP is illustrated in Figure 2. As shown in the figure, each virtual machine, including the Platform Control VM, only has access to limited number of disks in the virtual disk pool. The Virtual Disk Manager manages the disk resources and has access privileges to the physical disks.

Note that all accesses to the physical disks will be checked by the hypervisor against the ACM in it. Although the device drivers and resource allocator work in non-root mode, MyCloud SEP will grant an access if and only if the access is permitted in the ACM. In the initialization process of a VM, the device drivers need a lot of device information such as manufacturer ID, etc.. MyCloud SEP intercepts the guest VM initialization operations and provides a device emulator to guest VMs. The device drivers in guest VMs may be malicious, thus, MyCloud SEP needs to monitor I/O from the device drivers in the guest VMs.

Since the resource allocator is out of the TCB, MyCloud SEP hypervisor will verify whether the results of resource allocation and allocation procedure (described in Section 4) are secure. For example, allocating the same disk block to multiple VMs is prohibited. The allocation of disks space should have no overlaps either.

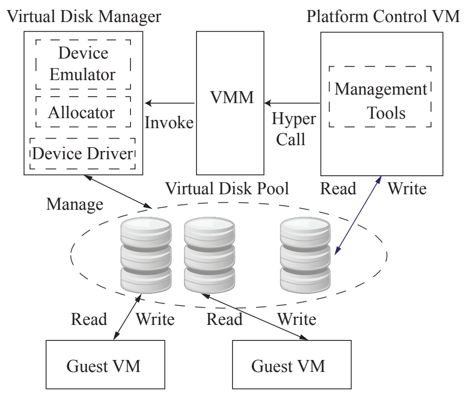


Fig. 2: Virtual disk structure.

The Virtual Disk Manager launched in non-root mode includes device emulators for guest VMs and physical device drivers for disks. In MyCloud SEP implementation, the Virtual Disk Manager is just a piece of codes which provides Intel AHCI [39] emulation and communicates with local SATA disks. The new design reduces the attack surface of Virtual Disk Manager. In order to monitor the activity of disk drivers, the hypervisor will also create a VMCS structure and configure which instructions should be intercepted.

MyCloud SEP Hypervisor The hypervisor is the only component running in the root mode. Before the hypervisor is initialized, the boot loader of MyCloud SEP will verify the integrity of the hypervisor execution environment using Intel TXT technology. If the environment is secure, the hypervisor will be initialized. The initialization process of hypervisor completes the following tasks.

– Detect E820 map and isolate the physical memory for other component.
– Detect all PCI devices installed in cloud platform.
– Configure IOMMU in order to isolate device memory and guest VM's memory.
– Copy the hypervisor into specific memory address.

After the initialization process is finished, the hypervisor will be able to perform the following tasks

– Create VMCS structure for the control VM, guest VMs and Virtual Disk Manager. Specify what should be trapped in each VMCS structure.
– Create Access Control Matrix.
– Handle interrupts and exceptions happened in the guest VMs and devices while checking those operations against ACM.
– Deliver the device access operations from guest VMs to device emulator.
– Schedule the guest VMs.

The Platform Control VM The hypervisor creates a VMCS for the Platform Control VM and launches it in non-root mode. In MyCloud SEP, the hypervisor will set VMCS for the control VM so that any memory access not in its EPT will be trapped by CPU. Therefore, even the Platform Control VM cannot access the memory of a guest VM without its explicit permissions. The guest VM can grant access permissions to its own memory space through a hypercall that modifies the ACM in hypervisor.

The Platform Control VM can still allocate resources because the hypervisor will provide resource utilization status through HyperCall API (described in Section 4). Thus, the Platform Control VM can migrate VMs as long as it follows resource allocation procedures and the resource allocation does not violate policies specified in ACM.

Guest VMs Although guest VMs are running in the non-root mode, they can configure the ACM table via interfaces (HyperCalls) provided by the hypervisor. The guest VMs can also implement some privileged work such as memory introspection. The VM image and configuration file are stored in the local storage. Normally, the guest VMs are running as the same way in physical machine, because all of privileged instructions, interrupt and exceptions will be handled by the hypervisor. When a privileged instruction is executed in guest VMs, CPU will automatically switch to the root mode. Consequently, the hypervisor will receive a VMEXIT containing all information about the privileged instruction. After the hypervisor handles the privilege instruction, it will execute VMRESUME to return to the non-root mode. Guest VMs will receive the results generated by the hypervisor and resume.

4 Implementation

4.1 General Resource Management

There are resources on two types of devices - character devices and block devices. Character devices include keyboard, mouse and serial port etc,. Block devices include disks, network card etc,. In MyCloud SEP, block devices are managed in the unit of a "resource region". 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)}`.

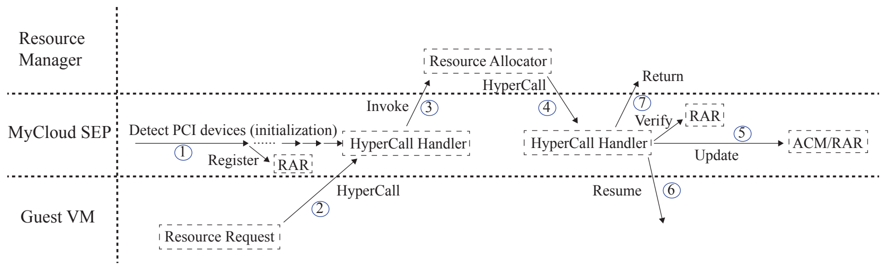


Fig. 3: Workflow of resource allocation

Figure 3 shows the procedure of how guest VMs apply for a block of resource. In step ①, the hypervisor sends I/O commands to port `0xcf8` and `0xcfc` in order to obtain each PCI device configurations. The acquired PCI device structure includes base address (BAR), specified command and I/O ports etc,. The hypervisor will then register the allocation information in a data structure – Resource Access Recorder (RAR).

When a guest VM applies a new resource region, it starts with step ②. The guest VM sends a HyperCall to the hypervisor. In order 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 invokes the resource allocators in step ③ by VMLAUNCH instruction. The resource allocator will return the allocation plan by another HyperCall. Since the resource allocator is not trusted, the hypervisor will verify the allocation plan by checking the RAR table. If the plan is approved, the hypervisor will update the RAR and ACM table. In step ⑥, the hypervisor will resume the guest VM with a new allocated resource region. Finally, the hypervisor returns the responses of the HyperCall sent from resource manager in step ⑦.

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

4.2 Access Authorization Based on ACM

Table 1: 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	

In MyCloud SEP, the hypervisor maintains an Access Control Matrix that is configurable by users, as shown in Table 1. The ACM table stores access permissions for each VM and resource regions. In the table, we use VDM as an example of resource managers. The VDM does not have direct access to any allocated resource regions such as disk blocks.

Note that the privilege design in MyCloud SEP is completely different from any of the existing cloud platform because the control VM does not have full privileges over the platform. In MyCloud SEP, the control VM is removed from the root mode and the privileges are specified in the ACM maintained by the hypervisor.

The hypervisor relies on Intel Extended Page Table (EPT) technology to intercept CPU memory accesses. We use Intel VT-d technology to isolate IOMMU memory accesses. Besides, the hypervisor will also check ACM table when allocating devices.

As shown in Table 1, only the hypervisor has accesses to all resources in the platform. The control VM has the same privilege level as guest VMs. It can only access resources assigned to the cloud administrator. If the cloud administrator needs to access users resources, it needs to be authorized by users through hypercalls of ACM configuration. VDM is responsible to provide device emulator and transfer data between SATA disks and guest VMs. Therefore, VDM has no permissions to access VMs memory. But the hypervisor provides a secure mechanism to verify the activities of VDM. The details will be explained in section 4.4.

4.3 Case Study: Disk Management

Figure 2 shows how to manage virtual disk in MyCloud SEP. The control VM accesses the virtual disks in the same way as guest VMs because it is running in the non-root mode. When the control VM or guest VMs boot, any device initialization in guest VMs or control VM will be trapped into the hypervisor, then handled by a device emulator. In the initialization stage, the guest OS will request device information such as *device ID, mentor ID, Base Address etc,.* The device emulator will offer virtualized device information to enable a guest OS to complete initialization.

In order to protect disk allocation information, the hypervisor in MyCloud SEP will employ a linear mapping from a logical disk space to a physical disk space. Figure 4 shows how the physical disk blocks are mapped to virtual disks. The linear mapping function calculates the address of a physical disk block by three parameters: cylinder number, sector number, and head number. We place the virtual disks in similar size into the same physical disk in order to reduce the number of fragments. If the users try to expand the size of virtual disks, the hypervisor can migrate it into other physical disks or servers. The linear mapping is protected in the hypervisor.

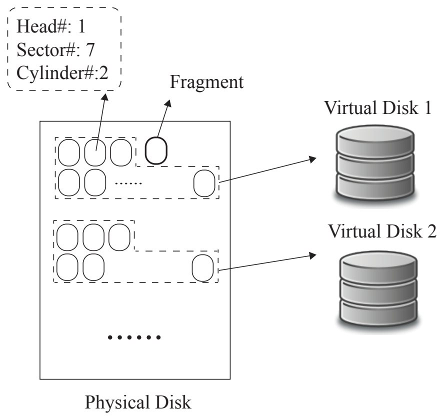


Fig. 4: Physical disk assignment.

According to Intel AHCI 1.3 specification [39], the AHCI works as an interface between OS and SATA disks. The hypervisor can detect AHCI information throughout PCI configuration space (0xcf8 and 0xcfc). Afterwards, the hypervisor will store device allocation information in RAR table such as base address, AHCI specific I/O port and registers. etc,. When a guest VM applies for new virtual disks, the hypervisor will invoke the resource allocator in VDM. The VDM designs which part of physical disk can be used for virtual disk volume. The hypervisor checks the ACM table and verifies if the physical disk blocks have already been allocated. Finally, the hypervisor updates the ACM table.

4.4 Hypervisor Processing of disk I/Os

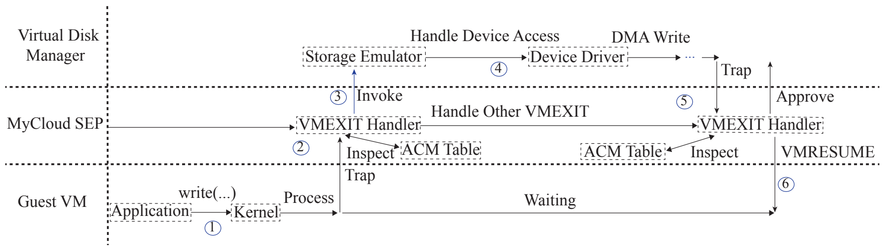


Fig. 5: Workflow of read/write operation.

MyCloud SEP implements disk emulator based on Intel ATA AHCI 1.3 Specification [39]. In essential, the Advanced Host Controller Interface (AHCI) encompasses a PCI device, then the AHCI Host Bus Adapter is constructed by a PCI header and PCI Capabilities. In the initialization step, guest VMs will try to access to PCI Configuration Space by I/O port *0xcf8 and 0xcfc*. As shown in Figure 1, when guest VMs try to detect PCI Configuration Space, a VMEXIT will be triggered and the hypervisor will transfer the I/O command to device emulator in VDM.

Figure 5 shows how a guest VM executes a `write()` function. When an application in the guest VM sends a disk write request to OS kernel, the kernel will process it and issue a series of I/O commands to configure and transfer data with AHCI HBA. The hypervisor can intercept the commands when the guest kernel or driver sends the commands to the I/O ports specified in AHCI 1.3. The hypervisor will verify if the trapped I/O commands meet the requirement of AHCI. The hypervisor will also check the ACM table for permissions. After that, 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.

The VDM needs to access guest memory in order to transfer data from memory to disk. 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 Intel VT-d technology [41]. To prohibit VDM from visiting memory space assigned to other VMs, MyCloud SEP configures IOMMU DMA remapping hardware and specifies the memory space the VDM can access. If the VDM reads/writes other memory space, the hypervisor will receive a VMEXIT.

To prevent VDM drivers from reconfiguring the device via I/O command, the hypervisor stores the resource region information when users send I/O commands to prepare disk operations. If the access is out of the scope of users-specified resourced region, the hypervisor will block the command. After VDM finishes the write operation, hypervisor resumes the guest VM.

4.5 Memory Isolation

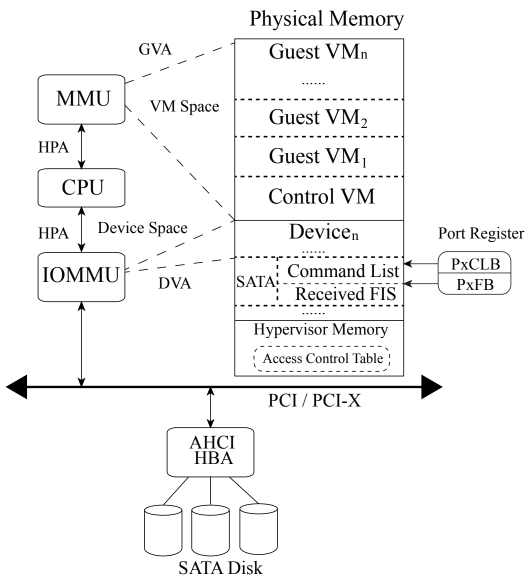


Fig. 6: Device and VM isolation in MyCloud SEP.

Figure 6 shows the isolated memory between VMs, device and hypervisor owned space. The memory isolation is implemented as follows:

MMU Access Isolation In order to isolate the memory space when the applications or kernel in the guest VMs try to access the data or instructions in memory, MyCloud SEP relies on Intel Extended Page Table (EPT) technology. The hypervisor will configure a 4-layer EPT table before users crate a guest VM. *EPT base pointer* in VMCS is set to record the entry address of EPT table. When a memory translation is requested by applications or kernel in the guest VM, Memory Management Unite (MMU) will walk the EPT table and translate the Guest Virtual Address (GVA) to Physical Host Address (PHA). 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 guest VM wish to share memory with the control VM, it should send the request to the hypervisor via a HyperCall. Next, the hyeprvisor will first verify the request, then revise the ACM table and EPT in order to make the memory space "visible" to the control VM.

IOMMU Access Isolation Most of device transmit data via DMA access and IOMMU is responsible for translating device virtual memory address to physical memory address. To isolate the DMA access made by physical device, MyCloud SEP implements Intel Virtualization Technology for Directed I/O [40]. Before disks execute DMA access, the hypervisor will set up Context-Entry Table (CET) in IOMMU to implement DMA Remapping. The CET table is indexed by {*PCI bus, device# and function#*} to find the address of translation table. The hypervisor builds Multi-Level Page Table in hypervisor's memory to translate Device Virtual Address (DVA) to Physical Host Address (PHA). Although the CPU cannot control the DMA access, IOMMU can trap the address translation and report DMA remapping faults if disks access the memory assigned to other devices. In general, the DMA Remapping and IOMMU configuration can also assign other peripheral devices (network card) to guest VMs and control the memory space that the device can visit. In our prototype, we implement the IOMMU access isolation for SATA disks.

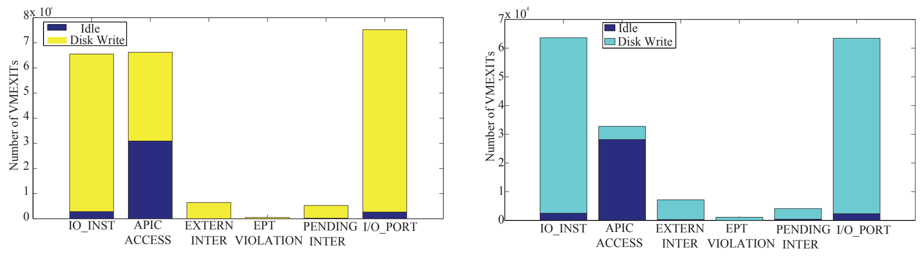
Resource Allocation Recorder Isolation MyCloud SEP also protects I/O related space, such as memory mapped I/O space (MMIO), PCI device configuration space and system register (MSR) mapped space. MMIO space is used to store I/O command and data for each device. The entry address and I/O port assigned for each device are basically specified by device mentor. In MyCloud SEP, we protect the MMIO space for AHCI and SATA disks. Based on AHCI specification 1.3, the most data and I/O commands are stored in two structures: Command List and Received FIS. The entry point for Command List and Received FIS is specified at chipset register PXCLB and PxFB. The hypervisor specifies the memory space for those structures by setting up the port register: PxCLB and PxFB. In order to protect PCI configuration space, the hypervisor will detect base address for each PCI devices via I/O port (0xcfc and 0xcf8), then set those space only "visible" to hypervisor.

To verify the memory and disk access, the hypervisor should store ACM table and a liner mapping that translates 3-dimension logical disk volume to physical disk volume.

5 Evaluation

Our evaluation test is built on a hardware platform that includes an Intel i7 2600 SEPcessor (with both Vt-x and Vt-d) running at 3.3Ghz, an Intel DQ67SW Motherboard, 4 GB RAM and 1 TB SATA HDD. The guest VM is Ubuntu 10.04 LTS with linux kernel 2.6.32.

5.1 Disk Operation Performance



(a) Block Size = 4KB (b) Block Size = 8KB

Fig. 7: Number of VMEXITs for Disk Operations

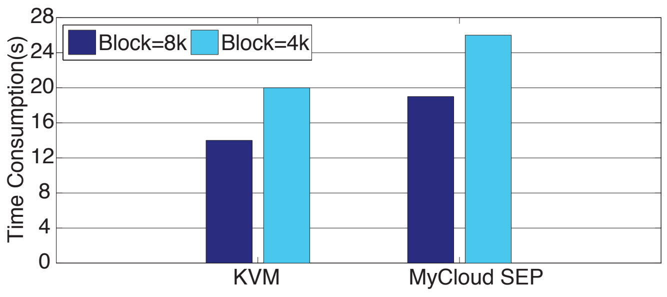


Fig. 8: Time Consumption for Disk Operations

To evaluate the performance of disk I/O operations in MyCloud SEP, we counted the number of VMEXITs and the time used for creating a 1GB blank file in a guest VM.

Figure 7 shows the types and the corresponding numbers of VMEXITs for creating the file with 4KB and 8KB block size. The figure presents the number of VMEXITs generated when the guest VM is at idle or disk write status. To create a 1GB file, the guest VM will introduce around 2×10^5 VMEXITS with 4KB block, and 1.38×10^5 VMEXITS with 8KB block. Though the number of VMEXITS looks huge, the corresponding extra overhead compared with KVM, such as time consumption (less than 6s more on 4KB block and 5s more on 8KB block, see Figure 8), is acceptable.

Figure 8 shows the time used for creating the 1GB file on KVM and MyCloud SEP platforms. We set the block size as 4KB and 8KB. In either case, MyCloud SEP takes 20% more time than KVM, because the disk I/O operations will be trapped into hypervisor and examined against ACM in it. According to our evaluation, the bigger the block size is, the less VMEXITs will be generated. The time consumption with 8KB block size is less than that of 4 KB block size.

6 Discussion

In MyCloud SEP design, the ACM is fully protected by the hypervisor. The hypervisor identifies any HyperCall that requests to change to or read from the ACM. A VM is allowed to only read or modify its own element in the ACM table. Any attempt to read or modify the ACM other than its own element will be detected and prohibited by the hypervisor.

6.1 External Attacks

The external attacks come from guest VMs, targeting at the hypervisor, through the hypervisor interfaces. In MyCloud SEP, device drivers, device emulator and the control VM are not part of the TCB. Compromising a guest VM or a malicious software component out of the hypervisor does not gain access to any other guest VMs since the ACM is maintained and enforced by the hypervisor. For example, in MyCloud SEP, the control VM is moved to non-root mode and monitored by the hypervisor. The disk space and memory space between guest VMs and the control VM are isolated and protected by the ACM in hypervisor. Any access from the control VM violating the access control rule in ACM will be prohibited by the hypervisor. Therefore, the attacker cannot exploit cloud tenant's private data by comprising the control VM. 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.

The attackers cannot breach users privacy through PCI devices either. MyCloud SEP isolates the device memory from guest memory, 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 for each device in order to prevent the attackers from overlapping the device memory to disclose users' private data.

6.2 Insider Attacks

In MyCloud SEP design, any privileged instructions executed in the control VM or other guest VMs will be trapped into the hypervisor for security check. The memory space of VMs is isolated from each other, so a malicious guest VM cannot access other VMs' space. Also, in MyCloud SEP, a malicious cloud administrator cannot access a guest VM space unless the guest VM explicitly grants the access through the ACM configuration. Thus, a malicious cloud administrator cannot gain control over guest VMs either.

6.3 More about the disk management

In current design, the virtual disk manager in MyCloud SEP does not utilize popular file systems like Linux extfs for higher level management. The fundamental reason is that the disk access information trapped by the hypervisor are physical disks locations indicated by cylinder number, head number, and track number. The hypervisor level information is different from the file system abstraction like `inode` for a file. There is no simple way using affordable size of codes to map from `inode` to disk blocks in the hypervisor.

Therefore, in MyCloud SEP design, we deploy a resource allocation tool, virtual disk manager, in a Linux VM, rather than using Linux file systems directly. The resource allocation tool maps resource regions to device files. Note that malicious resource allocation does not breach user's privacy. For example, allocating the same disk block to multiple VMs are monitored and prohibited by the hypervisor.

7 Conclusion

In this paper, we described a new architecture, MyCloud SEP, to separate resource allocation and management from the hypervisor. While providing flexibility of plugging-in resource management modules, the TCB size of virtualization platform is significantly reduced compared with commercial hypervisors. In our design, the hypervisor runs security check against an ACM for the resource manager, control VM, and guest VMs in the processor non-root mode. As the results, guest VMs' privacy is protected. Functionality and security check are also separated. Using virtual disk manager as an example, we implement a prototype on x86 architecture. The performance evaluation shows acceptable overheads of MyCloud SEP.

Ackknowledgement We thank all reviewers for their insightful comments. Meng Yu was supported by NSF CNS-1100221 and NSF IIP-1342664. Peng Liu was supported by ARO W911NF-09-1-0525, NSF CNS-1223710, ARO W911NF-13-1-0421, and AFOSR W911NF1210055.

References

1. Forbes: PRISM Projected To Cost U.S. Cloud Market $35B. `http://www.forbes.com/sites/louiscolumbus/2013/08/08/prism-projected-to-cost-u-s-cloud-computing-industry-35b`
2. Xen: http://www.xen.org/
3. KVM: http://www.linux-kvm.org/
4. Amazon Inc.: Amazon EC2. `http://aws.amazon.com/ec2/`.
5. Butt, S., Lagar-Cavilla, H.A., Srivastava, A., Ganapathy, V.: Self-service cloud computing. In: Proceedings of the 2012 ACM conference on Computer and communications security. CCS '12, New York, NY, USA, ACM (2012) 253–264
6. Li, M., Zang, W., Bai, K., Yu, M., Liu, P.: Mycloud: Supporting user-configured privacy protection in cloud computing. In: Proceedings of the 29th Annual Computer Security Applications Conference. ACSAC '13, New York, NY, USA, ACM (2013) 59–68
7. Murray, D., Milos, G., Hand, S.: Improving xen security through disaggregation. In: Proceedings of the fourth ACM SIGPLAN/SIGOPS international conference on Virtual execution environments, ACM (2008) 151–160
8. Pan, W., Zhang, Y., Yu, M., Jing, J.: Improving virtualization security by splitting hypervisor into smaller components. In Cuppens-Boulahia, N., Cuppens, F., Garcia-Alfaro, J., eds.: Data and Applications Security and Privacy XXVI. Volume 7371 of Lecture Notes in Computer Science. Springer Berlin Heidelberg (2012) 298–313
9. Zhang, F., Chen, J., Chen, H., Zang, B.: 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, ACM (2011) 203–216
10. Williams, D., Jamjoom, H., Weatherspoon, H.: The xen-blanket: virtualize once, run everywhere. ACM EuroSys (2012)
11. Ben-Yehuda, M., Day, M., Dubitzky, Z., Factor, M., Har'El, N., Gordon, A., Liguori, A., Wasserman, O., Yassour, B.: 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) 1–6
12. Kauer, B., Verissimo, P., Bessani, A.: Recursive virtual machines for advanced security mechanisms. In: Dependable Systems and Networks Workshops (DSN-W), 2011 IEEE/IFIP 41st International Conference on, IEEE (2011) 117–122
13. Steinberg, U., Kauer, B.: Nova: a microhypervisor-based secure virtualization architecture. In: Proceedings of the 5th European conference on Computer systems. EuroSys '10, New York, NY, USA, ACM (2010) 209–222
14. Heiser, G., Uhlig, V., LeVasseur, J.: Are virtual-machine monitors microkernels done right? SIGOPS Oper. Syst. Rev. **40**(1) (January 2006) 95–99
15. Keller, E., Szefer, J., Rexford, J., Lee, R.: Nohype: virtualized cloud infrastructure without the virtualization. In: ACM SIGARCH Computer Architecture News. Volume 38., ACM (2010) 350–361
16. Szefer, J., Keller, E., Lee, R., Rexford, J.: Eliminating the hypervisor attack surface for a more secure cloud. In: Proceedings of the 18th ACM conference on Computer and communications security, ACM (2011) 401–412

17. McCune, J.M., Parno, B.J., Perrig, A., Reiter, M.K., Isozaki, H.: Flicker: an execution infrastructure for tcb minimization. SIGOPS Oper. Syst. Rev. **42**(4) (April 2008) 315–328
18. Chen, X., Garfinkel, T., Lewis, E.C., Subrahmanyam, P., Waldspurger, C.A., Boneh, D., Dwoskin, J., Ports, D.R.K.: Overshadow: A virtualization-based approach to retrofitting protection in commodity operating systems. In: In ASPLOS. (May 2008)
19. Yang, J., Shin, K.G.: 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) 71–80
20. Hofmann, O.S., Kim, S., Dunn, A.M., Lee, M.Z., Witchel, E.: 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) 265–278
21. Ta-Min, R., Litty, L., Lie, D.: 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) 279–292
22. Cheng, Y., Ding, X., Deng, R.H.: Appshield: Protecting applications against untrusted operating system. In: Singaport Management University Technical Report. smu-sis-13-101 (2013)
23. Cloud, M.: http://www.microsoft.com/enterprise/microsoftcloud/
24. Cloud, H.P.: http://www.hpcloud.com/
25. CVE-2007-4993: Xen guest root escape to dom0 via pygrub.
26. 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.
27. 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.
28. Elhage, N.: Virtunoid: Breaking out of kvm (2011)
29. Kortchinsky, K.: Cloudburst: Hacking 3d (and breaking out of vmware). In: Black Hat Conference. (2009)
30. Wojtczuk, R., Rutkowska, J.: Xen 0wning trilogy. In: Black Hat Conference. (2008)
31. Secunia: Vulnerability report: Vmware esx server 3.x. `http://secunia.com/advisories/product/10757/`.
32. Secunia: Xen multiple vulnerability report. `http://secunia.com/advisories/44502/`.
33. 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.
34. CVE-2009-1244: Vulnerability in the virtual machine display function in vmware workstation allows guest os users to execute arbitrary code on host os.
35. Anderson, R., Kuhn, M.: Tamper resistance-a cautionary note. In: Proceedings of the second Usenix workshop on electronic commerce. Volume 2. (1996) 1–11
36. Intel Coperation: Intel® trusted execution technology (2011)
37. Intel Coperation: Intel® trusted platform module (2003)
38. Wojtczuk, R., Rutkowska, J.: Attacking smm memory via intel cpu cache poisoning. Invisible Things Lab (2009)
39. Intel Coperation: Serial ATA Advanced Host Controller Interface (2012)
40. Intel Corporation: Intel® Virtualization Technology Specification for Directed I/O Specification. `www.intel.com/technology/vt/`.