# MyCloud – Supporting User-Configured Privacy Protection in Cloud Computing

Min Li
Virginia Commonwealth University
lim4@vcu.edu

Wanyu Zang
Virginia Commonwealth University
wzang@vcu.edu

Kun Bai
IBM T.J. Watson Research Center
kbai@us.ibm.com

Meng Yu
Virginia Commonwealth University
myu@vcu.edu

Peng Liu
Pennsylvania State University, University Park
pliu@ist.psu.edu

## ABSTRACT

Privacy concern is still one of the major issues that prevent users from moving to public clouds. The root cause of the privacy problem is that the cloud provider has more privileges than it is necessary, which leaves no options for the cloud users to protect their privacy. Due to the same problem, once the control virtual machine or the cloud platform is compromised, all user's privacy will be breached. Many cryptographic solutions have been developed to protect sensitive data in the cloud. However, arbitrary processing is usually prohibited once cryptography is used. Homomorphic cryptography is considered promising but it does not offer practical performance at the current stage.

Instead of cryptographic solutions, in this paper, we propose a new cloud architecture - MyCloud to solve the problem. MyCloud removes the control virtual machine (control VM) from the processor's root mode and only keeps security and performance crucial components in the TCB. MyCloud achieves the following security goals. First, MyCloud de-privileges the cloud provider such that the cloud provider cannot inspect users' memory through the control virtual machine. Second, MyCloud enables user configured privacy protection. Third, the reduced TCB size also minimizes the attack surface of the cloud platform. We implemented a prototype system on x86 platform and the prototype has 5.8K LOCs. According to our experimental results, our platform shows acceptable overhead while providing significantly enhanced security and privacy protection that can be configured by users.

## Categories and Subject Descriptors

D.4.6 [**Operating Systems**]: Security and Protection; D.4.7 [**Operating Systems**]: Organization and Design; D.4.8 [**Operating Systems**]: Performance

## General Terms

Design, Security, Performance


*ACSAC* '13 New Orleans, Louisiana USA


## Keywords

Virtualization, TCB Minimization, Decomposition, Isolation

## 1. INTRODUCTION

Cloud computing attracts more and more traditional services due to its flexibility and cost-effectiveness. However, privacy concerns are preventing many users, especially government and military users, from moving to cloud computing environments. In current cloud computing platforms like Xen [59], KVM [38], and VMware [55], a cloud provider owns the Virtual Machine Monitor (VMM) and an administrative domain Dom0 (the control virtual machine, control VM) to do cloud management. Thus, the cloud provider has full privileges over the whole platform.

Unfortunately, under such an architecture, the cloud users have no control over their privacy protection. Even worse, they cannot even access the contents they own under some circumstance. For example, cloud users of Amazon EC2 [21] cannot dump VM memory because the operation is not allowed according to Amazon's policy.

While cryptography can be used to protect confidentiality in cloud, arbitrary processing will be disabled at the same time. Currently, only homomorphic encryption [11] allows arbitrary processing but it does not offer practical performance yet.

A possible solution to the privacy protection problem is to completely remove the service provider's privileges accessing users' space and only keep necessary privileges for cloud management, such as creation and migration of VMs, to the cloud provider. However, such design will also disable the cloud provider's capability of Virtual Machine Introspection (VMI) and its applications [24, 20]. Consequently, the service provider has no way to monitor any part of the users' VMs. In many situations, the service provider needs to run anti-virus or anti-spamware software over user's active VMs.

The challenging privacy protection problem is caused not only by the difficulties of cloud architecture design, but also the mutual distrust and conflict of interest between the service provider and clients [47]. On one hand, the cloud providers are motivated to monitor client VMs in order to make sure that clients will not use cloud resources to launch illegal services or breach privacy of other clients [7] that damages the cloud providers' reputation [39]. For example, a lot of spam emails are discovered to sent from IP addresses belonging to Amazon's EC2 service [37]. On the other hand, client users are more and more concerned about privilege misuse by cloud administrators. For example, the employees in cloud provider may disclose the privacy information of clients in order to seek more benefits [36].

As part of the efforts to solve the above problems, recent work Self-

Service Cloud computing (SSC) [12] separates the privileges of Dom0 into multiple domains. In SSC design, privileges in current privileged domain, Dom0, are assigned to user domains and an MTSD domain. The MTSD domain checks regulatory compliances mutually agreed between cloud provider and the clients. In SSC design, clients have privileges to protect their privacy from service providers. However, the Trusted Computing Base (TCB) of SSC is too large to be secure.

Current virtualization architecture includes a privileged domain, Dom0, in the cloud TCB. Usually, a complete operating system runs in Dom0 in order to support user level software. Dom0 maintains a lot of complex hardware drivers which are developed by different vendors. It is currently not affordable to formally verify the security of all these software components. Vulnerabilities are discovered to compromise the TCB in many different ways. For example, an adversary can exploit client privacy information via vulnerability from OS, drivers, or other software components [15, 19, 17]. Furthermore, current VMM designs are vulnerable to many known attacks [22, 35, 57, 48, 49]. The attackers can also utilize console interface provided by cloud provider to compromise the cloud platform [18, 16]. An effective approach to solve the problem is to significantly reduce the TCB size and reduce the attack surface.

In this paper, we propose a new architecture design, MyCloud, to address the problems. MyCloud allows clients to configure privacy protection and prevents service provider from tampering users' privacy settings. Moreover, in MyCloud design, we reduce the cloud TCB by an order of magnitude (our prototype has $\sim$ 5.8K LOCs). In MyCloud design, we remove the control VM from the TCB and the control VM is no more privileged than any other guest VMs. **Hence, it is certain that deploying MyCloud will provide the following benefits for both cloud providers and cloud tenants. Cloud providers can reduce the budgets on verifying the integrity of hypervisor due to the small TCB size in MyCloud. Meanwhile, cloud users are assigned the privileges to manage cloud services and protect their privacy from revealing by cloud providers.** The main contributions of this paper are as follows.

- We propose a new virtualization architecture, MyCloud, to support user-configured privacy protection. MyCloud uses the separation of privileges design where cloud provider does 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.
- We implement a prototype of MyCloud on x86 platform which has acceptable performance overhead but much stronger security protections.
- To the best of our knowledge, so far MyCloud has the smallest TCB that supports user configurable privacy protection in cloud computing.

The organization of this paper is as follows. Related work is discussed in Section 2. In Section 3, we describe the threat model, discuss the design goals, and provide an overview of our approach. In Section 4 and Section 5, we present the implementation and performance evaluations. We provide security analysis and discussions in Section 6 and 7 respectively. Finally, we conclude the paper in Section 8.

## 2. RELATED WORK

When the cloud provider has full privileges over the VMM and users' VMs, there is no way that cloud users can protect their privacy. The first step to enable privacy protection in virtualization architecture design is to design the separation of privileges.

To address the threats from the administrative domain, previous work try to disaggregate privileges functionality of Dom0 into separated client VMs [12, 44] or split VMM into two components [46]. In [44], the domain builder, which is part of the important privileged components is moved out of the service domain into a small TCB. Conceptually, the most similar work to ours is Self-Service Cloud computing (SSC) [12]. SSC allows client VMs to execute some privileges management which used to be provided in administrative domain such as, to access VM's memory, execute CPUID instruction, etc,.

However, in these work, the TCBs are not minimized because all of them consider a full functional hypervisor as part of the TCB. SplitVisor [46] splits VMM into two parts according to their functions: a smaller one regarded as minimized TCB in order to enforce isolation and a larger one to provide rich service functionality. Nevertheless, in SplitVisor design, clients have to upload a specialized guest VMM. Hence, the design is not compatible with current cloud computing scheme.

Similar to SplitVisor, recent work also investigates the uses of nested virtualization to disaggregate some host VMM components to the guest VMM [60, 56, 8]. CloudVisor [60] uses nested virutalization to separate hypervisor from the computing software stacks. As the host VMM, CloudVisor simply forwards all operation and data flows between the guest VMM and the VMs. CloudVisor has to allow guest VMM to operate with VM's sensitive components. For this reason, guest VMM should be considered as part of TCB. Moreover, to deploy nested virtualization on x86 hardware will impose tremendous performance penalties. Performance loss will increase exponentially with nesting depth [31]. Xen-Blanket [56] allows users to build their own business by launching VMs on a guest VMM. However, the service provider still have full control over the host VMM. In such case, a service provider has the ability to breach the client's privacy. Thus, clients cannot prevent administrators of service providers from accessing the client's privacy.

Besides the above architectural improvement attempts, there are many other research on encrypting client's sensitive data and the encryption algorithms do not rely on the offer of service providers, such as work [13]. Any employee from the service provider cannot decrypt data without a user's private key. However, during computation encrypted data should be decrypted into plain text in memory. In other words, these work is not suitable for supporting arbitrary computing such as a full guest VM. Also, encryption and decryption will consume a lot computing resources.

Given the above efforts in separation of privileges design, there are also a lot of efforts in minimizing the TCB of cloud platform, which is also one of our design goals. In order to minimize the size of TCB (VMM and dom 0), NOVA [52, 25] constructs a microkernel based VMM including $\sim$ 9K LOCs. Despite its thin TCB compared with commodity hypervisors, the complexity of TCB is not markedly decreased since the microhypervisor still needs to manage complex duties, such as address space allocation, interrupt and exception handling. Therefore, the thin TCB is still difficult to be verified dynamically. Although seL4 [33] proposes a technique to verify a microkernel with 8.7K KOC, it sacrifices functionality and usability.

At the same time, many researches focus on removing the unnecessary virtualization component. MAVMM [45] and Trustvisor [40] are specialized VMM with minimized TCB. Although the size of TCB is quite thin (MVMM contains $\sim$ 3.2K LOCs and Trustvisor has $\sim$ 2K LOCs for core functions), they can only handle specific scheme and neither of them support multiple VMs. In conclusion, these architectures are not suitable for serving commodity VMs.

Hardware vendors, like Intel and AMD, are willing to provide hardware features to minimize TCB size. These features include system

management mode (SMM) [29] and Trusted Execution Technology (TXT) [26]. SICE [6] utilize x86 SMM to isolate TCB. The security of isolated environment is guaranteed by TCB including hardware, BIOS and SMM program of $\sim$ 300 LOCs. However, SICE only supports one VM so that it will not be compatible with any cloud platform. Flicker [41] is also considered a privacy protection solution based on CPU features [26]. Unfortunately, it only offers application level protection and is not a general solution for VMs in cloud.

NoHype [32, 53] dynamically eliminate VMM layer in order to narrow the hypervisor attack surface. Nonetheless, the disadvantage of NoHype is that it require one-VM-per-core on multi-core processors and pre-allocated nested page table. The two requirements restrict the number of VMs that can be simultaneously hosted on the physical platform and decrease the elasticity. Coreboot [14, 9] is a promising way for TCB minimization. It replaces the BIOS firmware with lightweight system designed to perform minimum of initializing tasks and directly boot ELF image in ROM. However, coorboot has no virtualization component designed in mind.

Compared with the above work, MyCloud design reduces the TCB size by removing the control VM from the TCB. In MyCloud design, a cloud provider can only launch a de-privileged VM in order to do cloud management rather than having full privileges over the whole cloud platform. Thus, both separation of privileges and minimizing TCB are achieved.

**Unlike some researchers who reduce the privileges of cloud providers by extending the architecture of cloud operating system (e.g. OpenStack) [10], MyCloud minimizes cloud provider's privileges on cloud hypervisor which is located lower than cloud operating system. Therefore, MyCloud is more secure because the TCB size is much smaller. Besides, cloud operating system work based on the support of cloud hypervisors (e.g. Xen and KVM) which may contains a lot of vulnerabilities.**

**Data-Protection-as-a-service (DPaaS) model [51] is verifiable platform which can protect data integrity, control users access and allow users audit on the strength of data encryption, secure execution isolation and trusted platform module (TPM). MyCloud can manage users access and protect users data by relying on hardware (CPU and mainboard) security features and TPM.**

# 3. OVERVIEW

## 3.1 Threat Model and Assumptions

Similar to the adversary assumed in SSC [12], we distinguish *service providers* from *system administrators*. Normally, well-known enterprises such as Amazon and Microsoft are interested in protecting client's privacy rather than revealing or snooping on users' privacy, which protects their reputation of running cloud business. Thus, we assume that a service provider has no motivation or intention to breach users' privacy in the cloud or launch any physical attacks like memory bus tapping. Thus, physical attacks such as [5] or protection against other hardware attacks are out of the scope of this paper. The physical attack requires special tools and definitely leaves some evidence physically.

On the contrary, *system administrators* who are employed by services providers, may have opportunities and motivation to filch user's data for pursuing monetary benefits. Even if system administrators are benign, they could make mistakes by accident and cause privacy breach. Therefore, system administrators are considered adversarial.

In MyCloud design, we assume that we can utilize the Trusted Platform Module (TPM) equipped in hardware to measure the integrity of crucial components in TCB. Also, since the System Management Mode (SMM) of processors has already been protected from cache-poisoning attacks [58], we do not consider this type of attacks either. Similarly, on Intel platform, a proper configuration of the System Management Range Register (SMRR) is required to ensure this assumption.

Note that in this paper, we do not try to derive which part of memory should be protected in order to protect a data item at high abstraction level, such as a social security number. This is out of the scope of this paper. We provide protection mechanisms at the VMM level but it is up to the guest VM which regions of memory should be protected.

In this paper, we assume that the swap area of the guest VMs are turned off or the swapped pages are encrypted in order to protect the guest VM space. For example, this can be achieved by using `swapctl`, `swapoff`, and/or encryption tools in Linux. Thus, the guest VM's privacy will not be compromised when part of the memory is swapped out to the disks.

## 3.2 TCB Integrity Measurement

The Root of Trust Measurement (RTM) mechanism is commonly used to ensure the integrity of TCB, which mainly relies on the Trusted Platform Module (TPM) chip. Specified by the Trusted Computing Group (TCG), the TPM chip can be used to authenticate hardware devices [54]. It can be found on almost all the motherboards of servers and high-end PCs. A unique and secret RSA Endorsement Key (EK) is generated for each TPM at the time of manufacture and will be permanently sealed inside the chip, and other sensitive data will be stored into shielded memory. A Privacy CA (Certificate Agency) can authenticate a TPM according to its public Endorsement Key. The main role of TPM chips in trusted computing is to act as the Core Root of Trust for Measurement (CRTM), which measures the integrity metrics of modules, holds them in Platform Configuration Registers (PCRs) and reports them in an authenticated way in remote attestation. For privacy concerns, EK is not allowed to be used as platform identity directly. Instead, Application Identity Keys (AIKs) are created to sign these PCR values. A detailed example to establish TCB with TPM can be found in Terra model [23]. Note that RTM can be either Static or Dynamic (SRTM and DRTM, respectively) [26].

## 3.3 Design Goals

One principle of MyCloud design is to provide privacy protection *mechanism* but *not the policy* themselves. More specifically, the primary goal of MyCloud is to enable configurable privacy protection by 1) allowing users to build their own Access Control Matrices (ACM) in the TCB; and 2) reducing the TCB size to be more secure. The detailed design considerations are listed below.

### *Users-Configured Privacy Protection.*

By default, the control VM has no access permission to any of the guest VM unless the guest VM grants the permission. In other words, once the cloud provider allocates the memory resource to run a user's VM, the cloud provide will lose access permissions to the user's VM space, unless the user explicitly authorizes accesses. By default, no access permission is granted to the cloud provider.

An interesting argument will be whether the cloud provider should have right to get the resource back from a particular VM in order to protect the platform against DoS attacks. Either solution (allow or not) can be supported by our design through the configuration of the ACM in the VMM, depending on the cloud provider's Service Level Agreement (SLA).

### *TCB Minimization.*

The large size and high complexity of security-sensitive applications and systems software is a primary cause of poor testability and high vulnerability [50]. Hence the TCB size of the cloud architecture with MyCloud should be as small as possible. However, a small/simple

TCB is not sufficient. There should be a strong protection mechanism to enforce the security of the TCB. Formal analysis [34] is usually used to verify the TCB correctness and security properties, and software model checking [30] can be utilized to verify the implementation. There are also approaches utilizing hardware based dynamic measurement to secure TCB, like TrustVisor [40] and Flicker [41] . All the above protection mechanisms, however, have 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 [34]. Therefore, we need to control the TCB size by including only security related or crucial functionalities.

*Weakening the Cloud Provider's Power.*

In order to alleviate the concerns of privacy leakage to cloud provider's internal employees, the over-powerfulness of cloud providers should be dealt with. In the current cloud architecture, Xen [59] for example, the cloud provider occupies the most privileged domain and handles all the operations with the authority to look into users' data and computation. In MyCloud based architecture, the power of cloud provider should be limited, as long as it can normally perform cloud resources management (allocation, revoking and migration).

*VM Level Isolation.*

We choose the isolation granularity at the VM level. First, most of the current commercial public clouds provide the service in the IaaS fashion (e.g. Amazon EC2 [21]). Second, VM is a native and simple abstraction/encapsulation of privacy for each cloud user. Unlike protecting processes, protecting VMs does not require handling the complex and subtle semantic gaps. And third, protection at the VM level is more likely to preserve backward-compatibility, without the need of modifying OS kernels and applications.

## 3.4 Architecture

Intel processors support virtualization extension through VMX [29] and AMD processors use SVM [4]. Once the virtualization extension is enabled by processor, CPU will have two modes - *root* and *non-root*. In each mode, there are four privileged levels from ring 0 to ring 3 where ring 0 has the highest privilege level. A standalone operating system usually runs the kernel in ring 0 and applications in ring 3.

The VMM, running in the `root` mode, can specify what should be trapped and handled by VMM, such as EPT exceptions, page faults, timers, etc. When those privileged operations are run by a guest operating system in `non-root` mode, it will trigger mode transition to the `root` mode with a VMEXIT. After those priveleged operations are handled, VMM will enter guest operating system again through a VMENTRY. Note that the handling of those operations in VMM is transparent to guest VMs. There is a VMCS structure, including both the host status and the guest status, for each VM when VMEXIT (VMENTRY) happens to record (recover) the running context of the VM.

Figure 1 shows the architectures of both KVM and Xen. In both architectural designs, the host operating system (control VM) does not have a separate VMCS for virtual machine context switching so it runs in the root mode. As the results, the host operating system has all privileges that the processor's `root` mode has. Therefore, the TCB of both designs includes the whole control VM (the host operating system).

Moreover, since the control VM is in the `root` mode, the control VM is able to manipulate the VMCS structures of all guest VMs, also the page tables of all other VMs. Under such designs, it is impossible to protect any guest VM from the control VM. Virtual machine introspection, e.g., using XenAccess [2], can be done in the control VM. Thus, no privacy protection can be achieved.

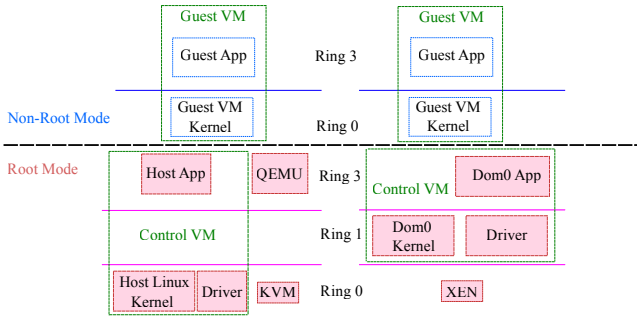



Figure 1: KVM (left) and Xen (right) architectures. Components of the TCB of each architecture are shown in pink color (or shadow).

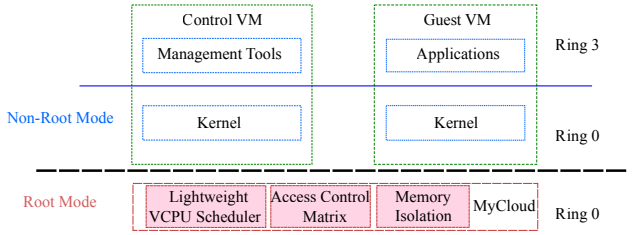



Figure 2: MyCloud architecture. Components of the TCB are shown in pink color (or shadow).

Figure 2 shows the architecture of MyCloud. In MyCloud, only the VMM runs in the *root* mode and VMM keeps security related functionalities in the TCB. In our design, the scheduler is a timer triggered pre-emptive scheduler against DoS attacks from any VM running on the platform. Memory isolation is also enforced by the VMM. In Section 4.2, we describe how memory isolation and devices are handled in MyCloud architecture.

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 is specified by an Access Control Matrix (ACM) in the VMM, following a separation of privilege design as described in Section 4.1. 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. IMPLEMENTATION

## 4.1 VM isolation and user-configured access control

MyCould uses EPT tables to isolate VMs including the control VM. If a VM tries to access any memory location out of its space, it will trigger an `EPT violation` that causes a VMEXIT. Since the control VM has its own EPT and VMCS, it cannot access any other guest VMs either. The control VM, has access to a resource table in the VMM 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 VMM 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.

In MyCloud architecture as shown in Figure 2, the cloud provider uses the control VM for cloud management. Our design completely removes the control VM from the root mode and the cloud provider's

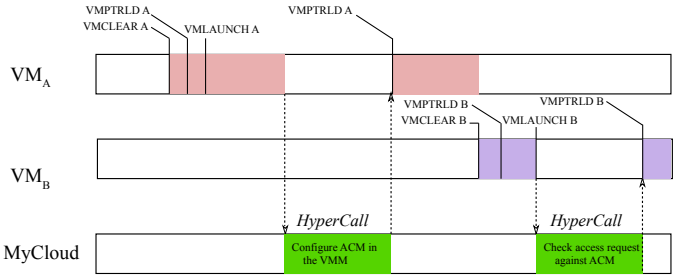



Figure 3: The procedure for users to modify the ACM

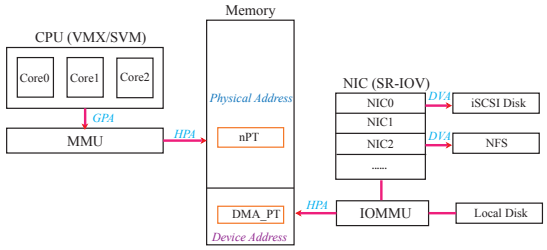



Figure 4: Memory and I/O management in MyCloud.

privileges are specified in the ACM in the bottom layer in root mode. Therefore, the platform has a de-priviledged domain for the cloud provider.

The ACM, shown in Table 1, enables a user to choose what information in the user's VM space can be accessed by the cloud provider or other VM domains. In the table, $ACM_i$ is the Access Control Matrix of $VM_i$. Note that the access permissions of our proposed architecture are completely different from any of the existing cloud platforms, as shown in the second row of the table. Actually by filling different values into the second row of Table 1, we can get a full spectrum of possible hypervisor designs. 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.

According to the ACM, each user can modify the access permissions to the user's space. By default, all accesses by other users including the service provider are prohibited. However, if the user likes, the user can grant access permissions to other users, or the cloud provider to enable information sharing or virus-scan. Our access control mechanism protects a user's sensitive information in the user's space.

Furthermore, all exiting technologies support page-level access control since violations are captured through page access related exceptions. In such design, a sensitive data item will be over-protected since accesses to all other data on the same page will be trapped and prohibited. To protect sensitive data items more accurately, we create additional data structures to draw the boundary between sensitive data and non-sensitive data.

In MyCloud, $ACM_i$ of a guest $VM_i$ is implemented by a Access Control List (ACL) that specifies memory regions, VM identifiers, and access permissions. If a VM wants to access a memory region of other VMs, e.g., doing Virtual Machine Introspection for virus scan, the VM initiates a hyper call to request the operation. VMM will check the request against the ACL of the VM being visited. If the access is permitted, the VMM will conduct the operation on behalf of the requesting VM. Otherwise, the access will be denied.

Since the memory region can be specified at the byte level, our protection provides byte-level access control. Also, because the protection is enforced in the procedure of a hypercall, it does not rely on paging mechanisms or exception handling either. Thus, it does not add overhead to EPT based protections.

Figure 3 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, through a hypercall, $VM_A$ modifies $ACM_A$ about whether it wants to share anything with other VMs. After $VM_B$ is scheduled, $VM_B$ sends a request through another hypercall to access $VM_A$'s memory space, e.g., reading $VM_A$'s kernel data structures. The request will be checked against $ACM_A$ by VMM to determine if it should be granted.

Note that we do not need security keys for VMs to do the hypercall. The VMM manages VM identifiers and VMCSs for all VMs. It is impossible for one $VM_i$ to set up $ACM_j$ if $i \neq j$.

## 4.2 Memory and Device Virtualization

Both Intel and AMD have extended two-layer address translation to three-layer address translation (nested paging). The guest page table (gPT) specified by CR3 register in guest VM is responsible for translating guest virtual addresses (GVA) to guest physical addresses (GPA). A new table called Extended Page Table (EPT) controlled by the VMM is responsible for translating GPA to machine frame number (HPA). The address of EPT is specified by a VMCS field. As shown in Figure 4, MyCloud maintains the EPT and MMU will automatically translate guest physical address to machine address. Once set up, the memory translation process will be automatically done by MMU and no interaction with the virtualization software is necessary. The VMM will only be called for EPT updating when a EPT violation exception happens.

Besides memory translation, I/O management is another important issue to consider. MyCloud fully makes use of the hardware extensions for virtualization, including IOMMU (VTd [27] for Intel and AMD-Vi [3] for AMD) and SR-IOV [28], to minimize the TCB size. The input/output memory management unit (IOMMU) connects a DMA-capable I/O bus to the main memory. Like a traditional MMU, which translates CPU-visible virtual addresses to physical addresses, the IOMMU takes care of mapping device-visible virtual addresses (also called device addresses or I/O addresses in this context) to physical addresses. With the help of IOMMU, devices can be directly assigned to VMs. This kind of direct assignment of devices provides very fast I/O and eliminates drivers from VMM. However, it prevents the sharing of I/O devices. To solve this problem, peripheral devices start to support SR-IOV to enable a Single Root Function to be appeared as multiple separate physical devices, called virtual functions (VFs). As shown in Figure 4, the Ethernet adaptor on MyCloud platform is configured to appear in the PCI configuration space as multiple functions. The slave layer can assign different VFs to different VMs.

For a device that does not support virtualization, like hard drives, there are two solutions. First, cloud users can mount iSCSI disks [43]. Second, MyCloud can redirect the disk I/O requests to the control VM, who controls the local disk. However, this solution exposes users' data to cloud provider, so I/O encryption is required for it.

## 4.3 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

Table 1: 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 |

have a lot of things to do.

We also implement an algorithm close to fire-sharing that evaluates more often about whether scheduling another VM to use the CPU upon many VMEXITs, thus other VMs have more chances to use the CPU. The experimental results are discussed in Section 5.

## 4.4 Cloud Management

Unlike the traditional architectures, the cloud provider only controls an unprivileged control VM in MyCloud design. The control VM is responsible for resource management. The management work is indirect and should be done through the interface provided by the VMM. Any resource allocation change requested by the control VM will be handled by the VMM.

Cloud users' key management is out of the scope of this paper while a key system is necessary to ensure authentication and cloud platform verification. We provide the following examples to explain that the cloud management is possible with a pre-configured public key system.

To create a VM, MyCloud will allocate the resources according to the request from the control VM. The cloud user can remotely attest the platform, and negotiate a session key with MyCloud. Afterwards the cloud user can upload an image along with the hash value encrypted by the session key. If MyCloud can successfully verify the image, it will launch the VM until a destroy request is received.

If the resources allocated to a cloud user are expired or no longer needed, MyCloud will destroy the data first, and then mark them as free space to the control VM. There can be an argument upon whether the control VM should be able to forcibly re-collect guest VMs memory pages. If this is allowed, a compromised control VM may be a huge threat to the whole platform. Although MyCloud can clean all the content in the re-collected memory pages, the effects due to missing pages would still leak side-channel information to the control VM. However, if the forcible re-collection is not allowed, cloud providers may have a lot of troubles if they are not willing to continue providing service to some VMs. In such situations, if we want to satisfy the users' desire of privacy, we must sacrifice the power of cloud providers. For example, we can disable forcible re-collection of memory but keep charging the users who do not give up the resources.

We do not try to find out an ultimate solution to this argument. How to make such decision is business policy related and it is simply out of the scope of this paper. Our design simply support either way of decisions. Simply put, we provide security mechanisms but not policies.

Since the control VM's memory access is also restricted by the ACM and any privileged CPU or I/O instructions will be captured and checked by the VMM, it is impossible for cloud provider's internal employees to launch insider-attacks.

# 5. EVALUATION

In our prototype implementation, the TCB size of MyCloud is around 5.8K LOCs. The comparison of TCB size with other virtualization techniques is shown in Figure 5. In the figure, 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

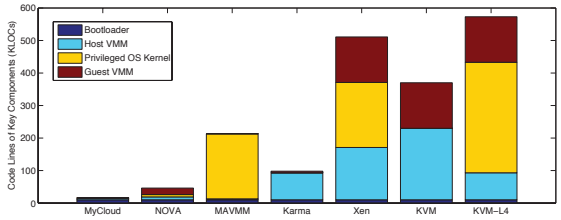



Figure 5: TCB size comparison of some virtualization architectures.

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.

In order to evaluate the overheads of our platform, we compared the following five configurations of experiments. **Since we just implement a simple Round-Robin schedule algorithm which can exert a tremendous influence on guest VM performance, we don't compare MyCloud with other commercial hypervisor (Xen and KVM) with sophisticated schedule scenario**.

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 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.

### *CPU Computing Performance.*

The result of CPU operations, 32 bit integers, 64 bit integers, float numbers, and doubles are shown in Figure 6. 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 6 also shows the performance for popular processes like fork, exec and sh. **According to our understanding and experiment observation that MyCloud reports a large number of VMexits for context switches, we believe that complex processes in *lmbench* involves many context switches which have to be executed in VMX Root Mode. Hence, the frequent Non-Root/Root Mode transitions contribute to performance reduction for 'fork' and 'exec' processes. Fortunately, in the real word, applications and computing in the guest VMs will not generate such a lot of context switches. Consequently, the performance of guest VMs in MyCloud is still acceptable.**

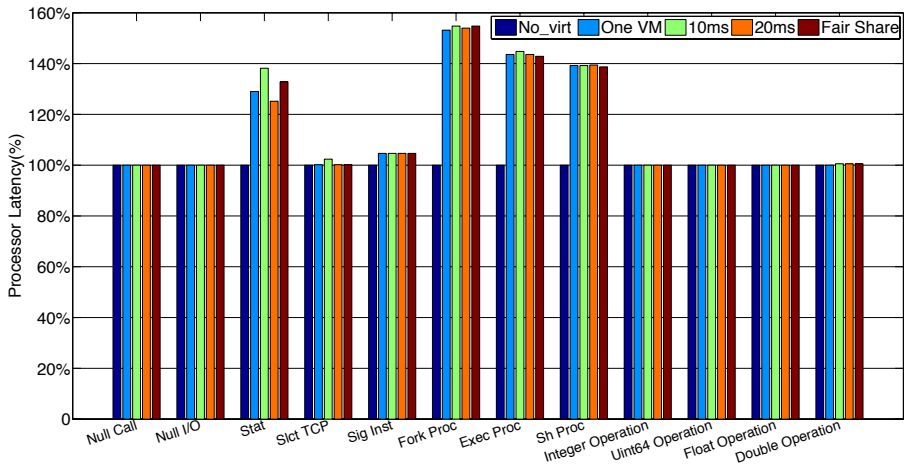



Figure 6: CPU latency measurements, measured by *lmbench*.

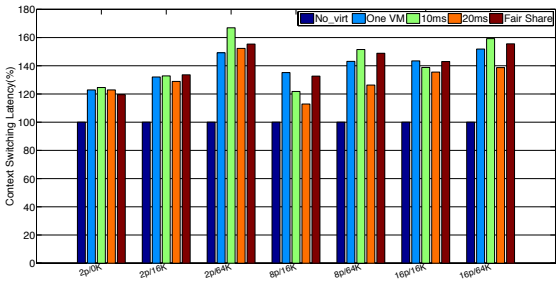



Figure 7: Context switch latencies measurements, measured by *lmbench*.

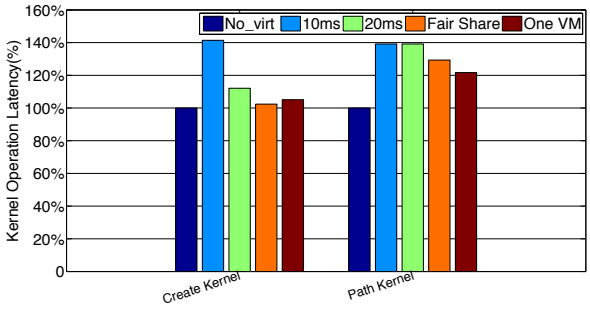



Figure 8: Kernel Operation latencies measurements, measured by *compilebench*.

### *Context Switch Efficiency.*

We use *lmbench* [42] to measure the overheads introduced in multi-process context switching. Figure 7 shows the results of latencies when running multiple processes in guest VMs. When the number of processes increases to 16 and the data size transferred within processes increases to 64K, we can see the context switching efficiency based on the results.

### *Kernel Operation Performance.*

Figure 8 shows the kernel operation performance measured by *Compilebench* [1]. We found that the scheduling algorithm can greatly impact the performance. When there is only one VM, the performance loss is around 21% compared with the operating system running on the bare metal machine. Kernel operation performance measurement includes operations of computing and memory read/write. Each time of VM switching clears up caches and TLBs, which is the main cause of overheads. Since one VM will not cause VM switching, the performance is close to the case without virtualization.

### *System Bandwidth and Latencies.*

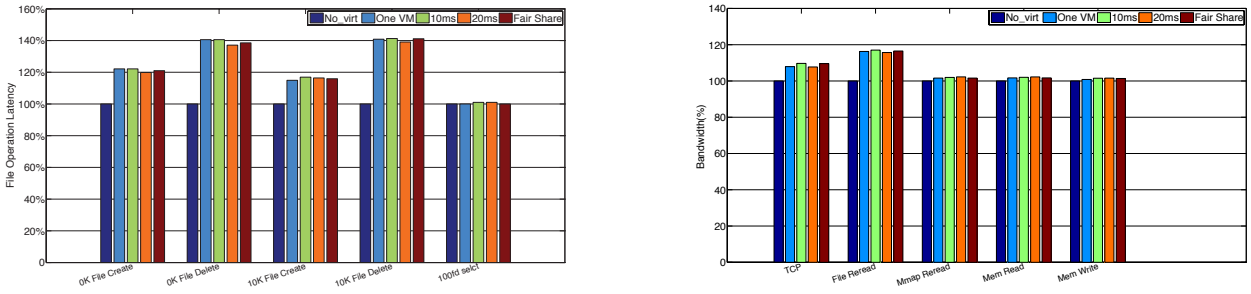
Figure 9a and Figure 9b show the results of a comprehensive measurement of system bandwidth and latencies, including file creation/deletion and virtual memory latencies (Figure 9a) and local communication bandwidth (Figure 9b). From the results we can conclude that local physical memory access (R/W), File operation and I/O operation do not have a lot of influence on the guest OS. Besides, the VM scheduling algorithm has little contribution to the performance loss.

## 6. SECURITY ANALYSIS

### *Privacy Protection Interface.*

Since malicious system administrators are deprived of the privileges to access users' privacy, they may try to hijack the hypercall and change the users' ACM when client VMs are modifying the ACM. MyCloud can defeat against this kind of attacks because VM identifiers are managed and checked by the VMM for each hypercall. Furthermore, since this is the only hypercall changing the ACM, the attack surface is very small. In MyCloud, access control specified by Table 1 is precisely and strictly followed.

### *VM-to-VMM Attack Surface.*




(a) File and virtual memory latencies

(b) Bandwidth latencies

Figure 9: Latency and bandwidth measurements, measured by *lmbench*.

In any virtualization systems, executions should be intercepted if they attempt to perform privileged operations. In MyCloud, VMEXIT happens on a privileged operation or exceptions. Thus, the VMM needs to interact with the VM frequently due to VMEXITs.

In MyCloud design, TCB size is greatly reduced by excluding the complicated drivers, management programs and complex scheduling codes. The control VM is put into the `non-root` mode. Currently, the TCB of MyCloud is quite small (∼5.8K LOCs in our prototype) and can be easily verified (recent work has shown the capability to verify ∼8.7K LOCs VMM [34]). As long as the TCB is secure, the privacy protection is guaranteed.

### *VM-to-VM Attack Surface.*

The security of MyCloud TCB ensues the enforcement of VM isolation. Thus many VM-to-VM attacks are immunized. If a VM attempts to access memory pages that not belong to it, it will be trapped through an EPT violation exception and handled by the VMM. The only interface to access other VM's space is through a hypercall. In such situations, MyCloud will check whether the access is authorized by the pages' owner or not. In this way, privacy breaching through memory access 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 VMM to process VMEXITs frequently, and takes CPU time slices away from the other VMs. Due to this concern, we provide simple round-robin algorithm that is purely timer based to protect against DoS attacks. The availability is always guaranteed by round-robin since time slices are fixed for each VM. Alternatively, MyCloud can defend against it by keeping statistics of VMEXITs (e.g. a large number of unauthorized memory access with a time period) and quarantining such VMs.

### *Insider Attacks.*

In MyCloud design, the cloud provider owns only the control VM and indirectly manages cloud resource allocation through the interface provided by the VMM. Note that cloud management tools also rely on the same set of interfaces provided by the VMM. Any resource allocation change requested by the control VM will be checked and handled by the VMM. In this way, the cloud provider cannot stealthily manipulate users' secrets. Moreover, the control VM is not more privileged than any guest VMs. 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.

## 7. DISCUSSION

### *Secure I/O Workflow.*

Since a VM is usually attached to virtual or physical disks, anything stored to 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. Due to the same reason, a user should disable operating system swap file or uses encrypted swap files against attacks to the data on the 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.

### *Mutually Trusted Policy.*

In MyCloud design, we provided mechanisms for low level access control but we do not try to provide policies, such as which part of VM space should be protected. We believe that the policy should be determined by the cloud provider's SLA and also the clients.

For example, it will benefit the clients if the clients agrees to grant access to the operating system critical data structures to protect the VM from malicious codes. By this way, the cloud provider can periodically scan the VM's critical data structures to make sure it is not compromised and malicious code free. However, such efforts is related to the previous problem - how to determine the exact boundary of protection or how we are certain that we are not over-protecting or under-protecting our sensitive data. We will consider those problems in our future research.

### *SMM Attack.*

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.

### *Secure Boot & Key Management.*

The integrity of MyCloud platform can be protected in several steps. During the boot procedure, SRTM based on TPM can be used. Later on, DRTM such as Intel TXT/MLE technology can be used to verify the integrity of platform. In order to allow remote users attest the integrity of the platform, MyCloud implement a simple key management mechanism like CloudVisor [60]. 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 using the VM key ($K_{VM}\{Hash(\text{VM image})\}$to remote users. Hence, the remote users can authenticate the integrity of cloud platform.

# 8. CONCLUSION

In this paper, we propose a new cloud computing platform - MyCloud. MyCloud de-privileges the control VM and removes the control VM from the TCB of the cloud platform. MyCloud enables users to set up an ACM in the VMM to protect the user‘s space. We have built a prototype system of MyCloud on the x86 platform with acceptable overheads. The Trusted Computing Base (TCB) of MyCloud is only around 5.8K LOCs.

# 9. ACKNOWLEDGEMENT

We thank anonymous reviewers for their insightful comments that helped us to greatly improve the paper. Meng Yu was partially funded by NSF CNS-1100221 and NSF IIP-1342664. Peng Liu was partially funded by NSF CNS-0905131, AFOSR W911NF1210055, and ARO W911NF-09-1-0525 (MURI).

# 10. REFERENCES

[1] compilebench. `https://oss.oracle.com/~mason/compilebench/`.

[2] xenaccess. `http://doc.xenaccess.org/`.

[3] Advanced Micro Devices. AMD I/O Virtualization Technology (IOMMU) Specification, February 2009.

[4] Advanced Micro Devices. AMD64 Architecture Programmer's Manual Volume 2: System Programming, December 2011.

[5] R. Anderson and M. Kuhn. Tamper resistance-a cautionary note. In *Proceedings of the second Usenix workshop on electronic commerce*, volume 2, pages 1–11, 1996.

[6] A. 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*, pages 375–388. ACM, 2011.

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

[8] M. Ben-Yehuda, M. Day, Z. Dubitzky, M. Factor, N. Har'El, A. Gordon, A. Liguori, O. Wasserman, and B. Yassour. The turtles project: Design and implementation of nested virtualization. In *Proceedings of the 9th USENIX conference on Operating systems design and implementation*, pages 1–6. USENIX Association, 2010.

[9] E. Biederman. Kernel korner: About linuxbios. *Linux J.*, 2001(92):7–, Dec. 2001.

[10] S. Bleikertz, A. Kurmus, Z. A. Nagy, and M. Schunter. Secure cloud maintenance: protecting workloads against insider attacks. In *Proceedings of the 7th ACM Symposium on Information, Computer and Communications Security*, ASIACCS '12, pages 83–84, New York, NY, USA, 2012. ACM.

[11] D. Boneh, G. Segev, and B. Waters. Targeted malleability: homomorphic encryption for restricted computations. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, ITCS '12, pages 350–366, New York, NY, USA, 2012. ACM.

[12] S. Butt, H. A. Lagar-Cavilla, A. Srivastava, and V. Ganapathy. Self-service cloud computing. In *Proceedings of the 2012 ACM conference on Computer and communications security*, CCS '12, pages 253–264, New York, NY, USA, 2012. ACM.

[13] I.-H. Chuang, S.-H. Li, K.-C. Huang, and Y.-H. Kuo. An effective privacy protection scheme for cloud computing. In *Advanced Communication Technology (ICACT), 2011 13th International Conference on*, pages 260 –265, feb. 2011.

[14] Coreboot. `http://www.coreboot.org`.

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

[16] CVE-2009-1244. Vulnerability in the virtual machine display function in vmware workstation allows guest os users to execute arbitrary code on host os.

[17] 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.

[18] 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.

[19] 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.

[20] B. Dolan-Gavitt, T. Leek, M. Zhivich, J. Giffin, and W. Lee. Virtuoso: Narrowing the semantic gap in virtual machine introspection. In *Security and Privacy (SP), 2011 IEEE Symposium on*, pages 297 –312, may 2011.

[21] EC2. http://aws.amazon.com/ec2/.

[22] N. Elhage. Virtunoid: Breaking out of kvm, 2011.

[23] T. Garfinkel, B. Pfaff, J. Chow, M. Rosenblum, and D. Boneh. Terra: a virtual machine-based platform for trusted computing. *SIGOPS Oper. Syst. Rev.*, 37(5):193–206, Oct. 2003.

[24] T. Garfinkel, M. Rosenblum, et al. A virtual machine introspection based architecture for intrusion detection. In *Proc. Network and Distributed Systems Security Symposium*, 2003.

[25] G. Heiser, V. Uhlig, and J. LeVasseur. Are virtual-machine monitors microkernels done right? *SIGOPS Oper. Syst. Rev.*, 40(1):95–99, Jan. 2006.

[26] Intel Coperation. Intel® trusted execution technology, 2011.

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

[28] Intel Corporation. Intel® PCI-SIG SR-IOV Primer: An Introduction to SR-IOV Technology, January 2011.

[29] Intel Inc. Intel® 64 and ia-32 architectures software developer manuals, 2009.

[30] R. Jhala and R. Majumdar. Software model checking. *ACM Comput. Surv.*, 41(4):21:1–21:54, Oct. 2009.

[31] 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*, pages 117–122. IEEE, 2011.

[32] E. Keller, J. Szefer, J. Rexford, and R. B. Lee. Nohype: virtualized cloud infrastructure without the virtualization. In *Proceedings of the 37th annual international symposium on Computer architecture*, ISCA '10, pages 350–361, New York, NY, USA, 2010. ACM.

[33] G. Klein, K. Elphinstone, G. Heiser, J. Andronick, D. Cock,

P. Derrin, D. Elkaduwe, K. Engelhardt, R. Kolanski, M. Norrish, T. Sewell, H. Tuch, and S. Winwood. sel4: formal verification of an os kernel. In *Proceedings of the ACM SIGOPS 22nd symposium on Operating systems principles*, SOSP '09, pages 207–220, New York, NY, USA, 2009. ACM.

[34] G. Klein, K. Elphinstone, G. Heiser, J. Andronick, D. Cock, P. Derrin, D. Elkaduwe, K. Engelhardt, R. Kolanski, M. Norrish, T. Sewell, H. Tuch, and S. Winwood. sel4: formal verification of an os kernel. In *Proceedings of the ACM SIGOPS 22nd symposium on Operating systems principles*, SOSP '09, pages 207–220, New York, NY, USA, 2009. ACM.

[35] K. Kortchinsky. Cloudburst: Hacking 3d (and breaking out of vmware). In *Black Hat Conference*, 2009.

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

[37] B. Krebs. *Amazon: Hey Spammers, Get Off My Cloud!* `http://blog.washingtonpost.com/securityfix/2008/07/amazon\_hey_spammers\_get\_off\_my.html`.

[38] KVM. `http://www.linux-kvm.org/page/Main_Page`.

[39] L. Litty, H. A. Lagar-Cavilla, and D. Lie. Computer meteorology: monitoring compute clouds. In *Proceedings of the 12th conference on Hot topics in operating systems*, HotOS'09, pages 4–4, Berkeley, CA, USA, 2009. USENIX Association.

[40] J. M. McCune, Y. Li, N. Qu, Z. Zhou, A. Datta, V. Gligor, and A. Perrig. Trustvisor: Efficient tcb reduction and attestation. In *Proceedings of the 2010 IEEE Symposium on Security and Privacy*, SP '10, pages 143–158, Washington, DC, USA, 2010. IEEE Computer Society.

[41] J. M. McCune, B. J. Parno, A. Perrig, M. K. Reiter, and H. Isozaki. Flicker: an execution infrastructure for tcb minimization. *SIGOPS Oper. Syst. Rev.*, 42(4):315–328, Apr. 2008.

[42] L. McVoy and C. Staelin. lmbench: portable tools for performance analysis. In *Proceedings of the 1996 annual conference on USENIX Annual Technical Conference*, ATEC '96, pages 23–23, Berkeley, CA, USA, 1996. USENIX Association.

[43] K. Z. Meth and J. Satran. Design of the iscsi protocol. In *Proceedings of the 20 th IEEE/11 th NASA Goddard Conference on Mass Storage Systems and Technologies (MSS'03)*, MSS '03, pages 116–, Washington, DC, USA, 2003. IEEE Computer Society.

[44] D. Murray, G. Milos, and S. Hand. Improving xen security through disaggregation. In *Proceedings of the fourth ACM SIGPLAN/SIGOPS international conference on Virtual execution environments*, pages 151–160. ACM, 2008.

[45] A. Nguyen, N. Schear, H. Jung, A. Godiyal, S. King, and H. Nguyen. Mavmm: Lightweight and purpose built vmm for malware analysis. In *Computer Security Applications Conference, 2009. ACSAC '09. Annual*, pages 441–450, Dec.

[46] W. Pan, Y. Zhang, M. Yu, and J. Jing. Improving virtualization security by splitting hypervisor into smaller components. In N. Cuppens-Boulahia, F. Cuppens, and J. Garcia-Alfaro, editors, *Data and Applications Security and Privacy XXVI*, volume 7371 of *Lecture Notes in Computer Science*, pages 298–313. Springer Berlin Heidelberg, 2012.

[47] M. Price. The paradox of security in virtual environments. *Computer*, 41(11):22 –28, nov. 2008.

[48] Secunia. *Vulnerability report: Vmware esx server 3.x*. `http://secunia.com/advisories/product/10757/`.

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

[50] L. Singaravelu, C. Pu, H. Härtig, and C. Helmuth. Reducing tcb complexity for security-sensitive applications: three case studies. *SIGOPS Oper. Syst. Rev.*, 40(4):161–174, Apr. 2006.

[51] D. Song, E. Shi, I. Fischer, and U. Shankar. Cloud data protection for the masses. *Computer*, 45(1):39–45, 2012.

[52] U. Steinberg and B. Kauer. Nova: a microhypervisor-based secure virtualization architecture. In *Proceedings of the 5th European conference on Computer systems*, EuroSys '10, pages 209–222, New York, NY, USA, 2010. ACM.

[53] J. Szefer, E. Keller, R. Lee, and J. Rexford. Eliminating the hypervisor attack surface for a more secure cloud. In *Proceedings of the 18th ACM conference on Computer and communications security*, pages 401–412. ACM, 2011.

[54] A. Tomlinson. Introduction to the tpm. *Smart Cards, Tokens, Security and Applications*, pages 155–172, 2008.

[55] VMware. http://www.vmware.com/.

[56] D. Williams, H. Jamjoom, and H. Weatherspoon. The xen-blanket: virtualize once, run everywhere. *ACM EuroSys*, 2012.

[57] R. Wojtczuk and J. Rutkowska. Xen 0wning trilogy. In *Black Hat Conference*, 2008.

[58] R. Wojtczuk and J. Rutkowska. Attacking smm memory via intel cpu cache poisoning. *Invisible Things Lab*, 2009.

[59] XEN. `http://www.xen.org`.

[60] F. Zhang, J. Chen, H. Chen, and B. Zang. 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, pages 203–216, New York, NY, USA, 2011. ACM.