A Hypervisor for Protecting Information of Public Cloud's User on Memory and on Storage from Malicious Operators

パブリッククラウドユーザのメモリ上・ストレージ上情報を
悪意あるオペレータから保護するハイパーバイザ

48-126454
村上 航規

Supervisors
Professor Shuichi Sakai
Associate Professor Masahiro Goshima

A thesis submitted to
the Information Engineering Cource
of the University of Tokyo
for the degree of Master of Engineering

February 2014

ABSTRACT

This paper introduces a novel cloud computing architecture that en- sure s privacy for guest' s information and computation. In conventional cloud architecture, a security policy proposed by a provider only ensured the protection of guest' s infor- mation. This enabled malicious operators to steal or modify guest' s information. Our architecture protects guest' s infor- mation with novel memory management function of hyper- visor from malicious operators. Cloud computing generally relies on virtualization, and VMM or hypervisor maintains page table for interfering VM' s memory accesses, which is called shadow page table. Our hypervisor regulates memory accesses by management VM by adding a authority bit to shadow page table entry. Our architecture also prohibits a theft of guest' s information when it is stored in storage by encrypting data when they leave memory.

概要

本稿は、IaaS 型のパブリッククク ラウド環境において、ユーザが所有する情報を悪意あるオペレータによる窃盗・改変から保護するハイパーバイザを提案する。本手法では、既存のハイパーバイザのページテーブル機構に新たな情報を付加する改変を加える。またデータのスワップアウト時にストレージを暗号化することで、ゲスト情報を悪意あるオペレータから保護する。

CONTENTS

ABSTRACT i

概要 ii

Chapter.1 Introduction 1

Chapter.2 Motivation and Related Work 3
- 2.1 Motivation 3
- 2.2 Related Work 4

Chapter.3 Memory Management Mechanism and Hypervisor Architecture 6
- 3.1 Virtualization 7
- 3.2 Virtualization supported by Hardware 11

Chapter.4 Design of Hypervisor 14
- 4.1 Assumptions 14
- 4.2 Overview 14
- 4.3 Memory Protection Based on Shadow Page Table Entry 15
- 4.4 Memory Protection Based on Address Space Randomization . . 17
- 4.5 Storage Encryption 19
- 4.6 Key Management 19
- 4.7 Ensuring the Integrity of data on Storage 20
- 4.8 Compatibility with Hardware Supported Virtualization 21

Chapter.5 Discussion 22
- 5.1 Availability of Cloud Service Operation 22

5.2 The safety of Cache 23
5.3 Protection of Memory 23
5.4 Protection of Storage 24
5.5 Comparision of Two Techniques of Memory Protection 24

Chapter.6 Conclusion 26

Bibliography 27

Publications by the Author 29

Acknowledgement 30

Chapter.1

Introduction

Cloud computing is a model for enabling on-demand assignment of computer resources and application based on the Internet. This technology is one of the fastest growing segments of the IT industry. Cloud commputing allows its users to use functionalities or applications with minimal management cost. NIST defines three kinds of service model of cloud computing, Software as a Service (SaaS), Platform as a Service (PaaS) and Infrastructure as a Service (IaaS)[11].

Owe to cloud computing, companies as well as individuals have become able to gain fast access to variable applications or boost their IT infrastructures. On the other hand, clients, especially companies concern security to store critical information in cloud. According to Intel's survey on IT professionals[8], 28 percent of professionals have experienced a public cloud-related security breach. For heavily regulated industries such as banking, finance and healthcare, 83 percent of professionals concern the estrangement of legal responsibilities between legal-sensitive companies and cloud providers.

Cloud providers guarantee with their security policies that there is no leakage of clients' data nor computations. If a operator of provider or data center is malicious, however, they may steal or modify the data or computations of clients with their authority. Security policy cannot prevent these operators from illegal behavior. Needless to say, it will be more dangerous when management account of hypervisor is stolen by adversaries.

This paper proposes a novel cloud computing architecture of IaaS cloud environment. Our architecture ensures the privacy of data owned by clients on memory and storage even if provider's operator is malicious.

Cloud computing normally bases on virtualization. Virtual Machine Monitor(VMM) or Hypervisor has special page table for converting true physical memory space to virtual machine's virtual memory space and showing it to Virtual Machine(VM) as physical memory space.

Two kinds of memory data protection methods of proposing architecture are described in this paper. First one is based on additional privilege information to entries of shadow page table, which is owned by hypervisor. Second one is based on address space layout randomization (ASLR). These methods are compared qualitatively with their privacy, integrity and availability.

Mechanisms of storage encryption and key management are deployed for storage data protection. They are able to applied to both memory data protection methods this paper shows.

Our hypervisor prohibits accesses executed by malicious operators to memory which is assigned to clients. The data swapped out to storage is always encrypted and cannot be read by attackers. As far as our architecture is adopted in cloud provider's infrastructure, the privacy of clients' data is ensured.

The rest of the paper is organized as follows. Chapter 2 shows the motivation of this study and related works. Chapter 3 discuss our assumptions and presents popular designs of hypervisor. Chapter 4 reveals proposing architecture. A qualitative security discussion is done in chapter 5. Finally, we conclude in Section 6.

Chapter.2

Motivation and Related Work

2.1 Motivation

Our goal is to protect the information of cloud's user from malicious operator of cloud service provider. Conventional hypervisor gives a privilege for management functionality to a OS of a operator. This vulnerability does not only mean a malicious operator can steal and modify user's information but also make an influence of takeover of the operator's OS by adversaries. The method of encryption of data on cloud storage service is already put to practical use, on the other hand, plaintext data is stored on memory and malicious operator can steal it. In addition, IaaS cloud services cannot adopt the storage encryption method because some modifications of hypervisor architecture-level is required.

However, fabricating a hypervisor which prohibits illegal behavior of malicious operators is a challenging problem. There are two reasons why hypervisor's manufacturer gives a privilege to a OS of a operator. Firstly, security-critical operations are really necessary for operators to handle some situations such as generation of virtual machines(VMs), migration and troubles. The second reason is hardware device driver. Hypervisors called "microkernel" do not have device driver and every accesses to hardware by users' VMs are actually performed by a OS of a operator. This means that no mechanism on hypervisor limits unfettered hardware accesses of its OS.

Despite the vulnerability of conventional hypervisors, a social demand of introducing cloud services to companies and organization is increasing. Ones handle sensitive security information such as medical institutions and financial ones also require the importation of cloud services. As a result of washed away of servers of hospitals with electronic medical records in the 2011 Tohoku earthquake and tsunami, using cloud computing for records as a measures to deal with a natural disaster is advocated[13].

The proposing architecture achieves both limitation of operator's privilege for protecting user's information and ensuring the necessary management interface of operator's works. A threat of malicious operators' OS, which is one of problems those have prevented companies and institutions from introducing cloud computing, is resolved by the proposing hypervisor.

2.2 Related Work

In virtualized environment, a malicious guest may attack the vulnerability of hypervisor and gain a privilege. Once the malicious guest does that, it may run the illegal processes or steal their secret information. There are many studies[1, 14, 17] for preventing it. Those studies focused on the limitation of malicious guests' processes, thus the malicious operator is not limited on his/her privilege.

Dealing with a malicious operator's illegal behavior is also studied. CertiKOS[6] is an architecture for preventing information leakage on cloud environment by extending the general method of certification. CertiKOS hides the function of resource assignment in hypervisor and does not give its function to management VM. Every resource assigment mechanisms are delegated to the hypervisor kernel. Assigned resources are tagged and ownership record is owned by hypervisor. Resources with these tags are isolated in order to prevent certain VM (including management VM) from encroaching another VM's information. The granularity of CPU allocation on CertiKOS is a CPU core, since tagging resource allocation method. Time slicing allocation, however, cannot be performed.

NoHype[9] protects a guest operating system by extending CPU architecture and removing hypervisor. The address space conversion, which is normally performed by hypervisor, is implemented in special CPU. This means that NoHype cannot be applied on current commercial CPUs.

Challenging problems of protecting user's information from malicious operators in a cloud computing environment originate in a contrudiction of protecting the data and the process of lower-privileged subjects from higher-privileged ones. The reason why specific processes are commisioned to dominate other ones is privileged processes have mechanisms of critical functionalities such as hardware devices and memory management. On this point, studies of secure processor and ones based on them resembles to this study. Proxos is a hypervisor-based trust partitioning system. Overshadow[4] has multi-shadowing mechanisms to protect the memory allocated to the application that presents defferent views of the physical memory depending on whether the data is accessed by the application or by the OS. Flicker[10] proposed by McCune relies on a secure coprocessor like TPM[16] to provide a secure execution environment. Suh et al. proposed AEGIS[15] and it performs isolation between processes based on ID that is stored as tags for processor registers. SecureME[5] consists of secure processor substrate and hypervisor. The data of the application on memory is encrypted by the mechanism of processor and hypervisor-based ID assignment isolates process's memory from other ones.

Chapter.3

Memory Management Mechanism and Hypervisor Architecture

Paging is a mechanism of memory allocation performed by OS. OS divides physical address space in a fixed size (normally 4KiB) called **pages**, and assign them to each processes. The physical memory space is partitioned into the same size, and their fragments are called **frames**. Processes refer their virtual address space and OS converts it to physical address space with a structure named **page table**. The information of occupancy of frames are stored in **frame table**. Paging mechanism and memory address virtualization prevent memory area of certain process from overlapping one of another and relieve application programer from the load of memory management implementation.

Fig. 3.1 shows a mechanism of virtual address conversion of general OSes. A specific register of processor (called CR3 in x86 processors) stores the head address of **page directory table** (PDT). Entries of PDT are links to heads of page table. Each processes has its unique ID (PID) and it is correlated to a CR3 value. This relationship isolates virtual memory area of each processes. Conversion from virtual address to physical address is performed in the following manner; firstly, the front part of a virtual address is offset from the head of page table and a PDT entry is specified; secondly, a page table is linked with the PDT entry and the middle part of one is offset from the head of PT; finally, a page connencted to the page table entry(PTE) is referred and an absolute physical address is determined by the bottom part of the virtual address.

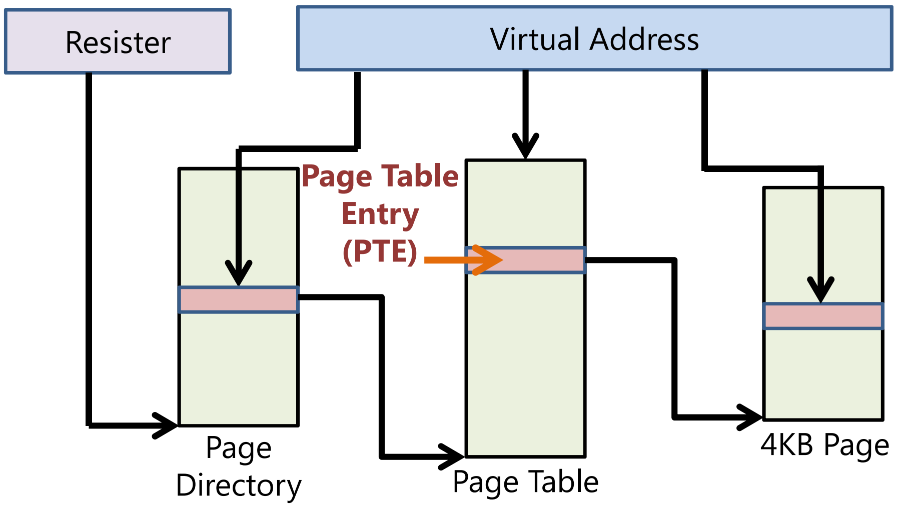


Fig.3.1 A mechanism of virtual address conversion based on paging.

This conversion is one of reasons of overhead. Most CPUs relieve overhead by caching its conversion information in its memory management unit(MMU). Some of these kinds of CPUs defines the structure of **page table entry**(PTE) having access permission bit and regulates accesses of processes. User processes, which have lower authority than kernel processes, are prohibited accessing physical address with PTEs whose bit is not set available for user processes.

Once the conversion is performed, the result is stored in a specific cache named translation lookaside buffer (TLB). This makes the conversion of second or later faster.

3.1 Virtualization

3.1.1 Hypervisor

Virtualization is a technology for abstraction of physical resources of computer. This enables to manage plural computers as one VM or to run variable VMs in the same time.

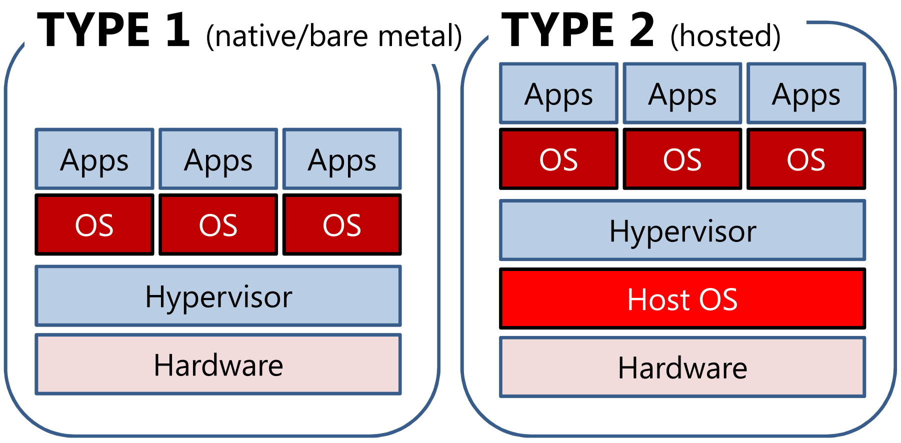


Fig.3.2 A classification of hypervisors.

Native and Hosted

The host software of virtualization, which is called hypervisor or Virtual Machine Monitor(VMM), manages the VMs as guests. Hypervisors are classified under two types (shown in fig3.2), Type 1(bare metal) and Type 2(hosted). Type 1 hypervisor runs directly on hardware and all guest OS do on hypervisor. Type 2 hypervisor runs on host OS as an application.

Type 1 hypervisor interrupts some kinds of critical instructions of guest, and other kinds of ones are computed in the same as physical environment. This means the guest OS runs fast. Type 2 hypervisor, otherwise, simulates VM's hardware. This means the guest OS runs slower than one of Type 1. Due to the faster computation, almost public cloud services consist on Type 1 environment. Our architecture also assumes that a hypervisor is Type 1.

Ring Protection

Ring protection (or hierarchical protection domains) is a layered structure for establishing the plural privilege levels. This protects data and functionalities from system faults and malicious behaviors. The deferent access levels are provided to each resources by OS. A processor defines the privilege state of current execution, and MMU controls accesses to memory confirming the present ring. Fig. 3.3 shows a typical four-level structure of ring protection. A kernel level process runs in ring 0 (the most privileged) and a user level process executes in ring 3.

In a virtualized system with a certain Type 1 hypervisor, a kernel level process of a guest OS works in ring 1 while the hypervisor acts in ring 0. A guest OS cannot run on the environment with this kind of hypervisors since the working ring is defferent from the physical environment. Virtualized environment which require guest OSes to be modified to suit to the environment is called **paravirtualization**. On the other hand, **full virtualization** environment do not demand guest OSes to be modified.

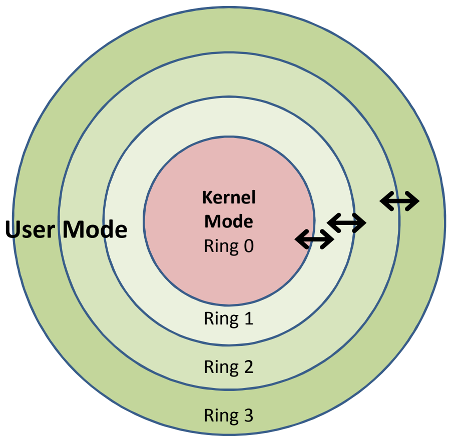


Fig.3.3 Ring protection of general systems.

Management VM

In virtualized environment, an administrator has authority for managing it and can access to management interface. Since there is no host OS in Type 1 hypervisor-environment, management interface is equipped to unique VM. Xen[2] has the same architecture. The first-launched VM of Xen environment is named dom0 and the administrator controls it as management interface. A guest OS on dom0 can directly access all kinds of physical hardware.

Monolithic and Microkernel

Type 1 hypervisors are categorized as **monolithic** (e.g. VMware ESX[7]) or **microkernel**ized (e.g. Xen[2] and Hyper-V[12]). Monolithic hypervisors (fig.

3.4) have their own device drivers. Accesses to hardware with drivers of all VMs (including an administrator's VM) are virtualized and ones of hypervisor perform the actual accesses to real hardware. These hypervisors are sperior in terms of performance due to the intimate cooperation between hypervisors and drivers, on the other hand, it is inconvinient that devices whose drivers are not implemented in monolithic hypervisors is unavailable. Accesses from VMs to hardware on microkernelized hypervisors are performed by drivers of management VM (fig. 3.5). Users' VMs cannot access directly while the management VM can do. Developers of microkernelized hypervisors are not responsible for implementing device drivers.

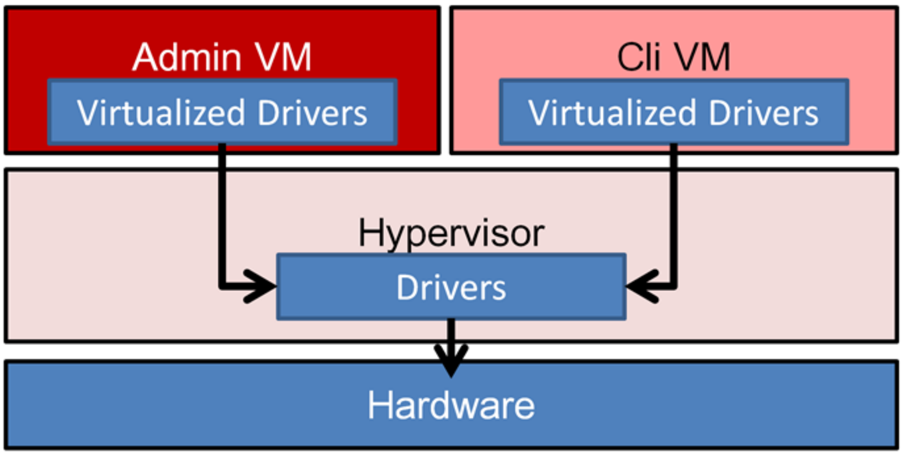


Fig.3.4 Hardware accesses on monolithic hypervisor.

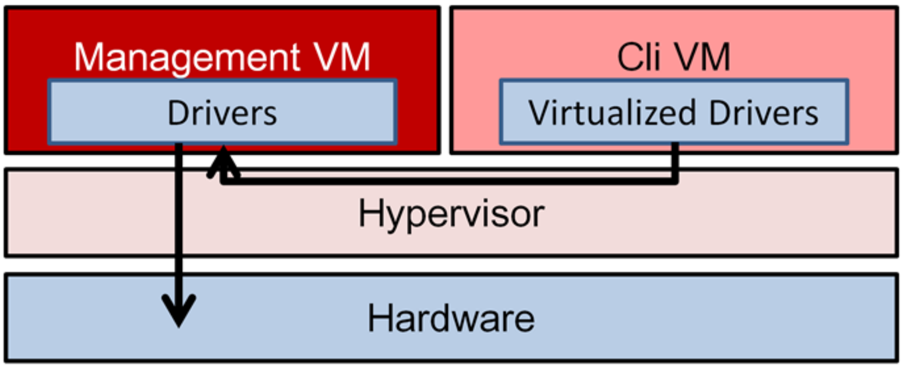


Fig.3.5 Hardware accesses on microkernelized hypervisor.

Shadow Page Table

When a guest OS accesses to memory, ordinary OS directly refers certain physical address. In virtualized environment, however, direct accesses to physical address performed by guest OS makes encroachment of memory assigned to host or other guests. Shadow page table (SPT) avoid this by converting system's physical address to guest's address. Fig. 3.6 is the relationship between address and page table. Hypervisor informs guest OS of virtual address as physical address referring SPT. Since this physical address is actually virtual address, hypervisor can isolate each assigned addresses for guests.

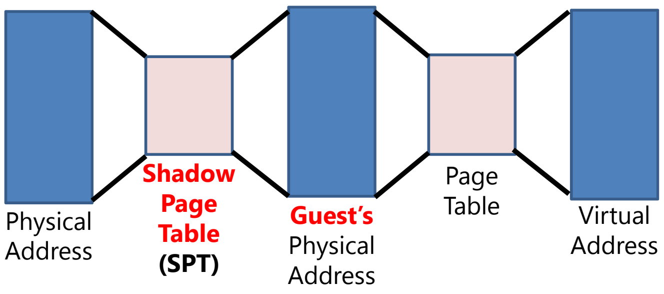


Fig.3.6 Address conversion of shadow page table.

3.2 Virtualization supported by Hardware

In recent years, many products of processors implement the supporting mechanism of virtualization. This is caused by the spread of virtualization among individual users. Hardware support of virtualization resolves some of problems of virtualized environment, especially complexity of implementation and permance overhead.

Ring Protection

In conventional virtualization, Type 1 hypervisor runs in ring 0 and guest OSes does in ring 1. Commercial OSes are generally constructed on the assumption that they execute in ring 0. This mismatch makes some of instructions not to be trapped by CPU and it may causes critical errors. Dealing processes with

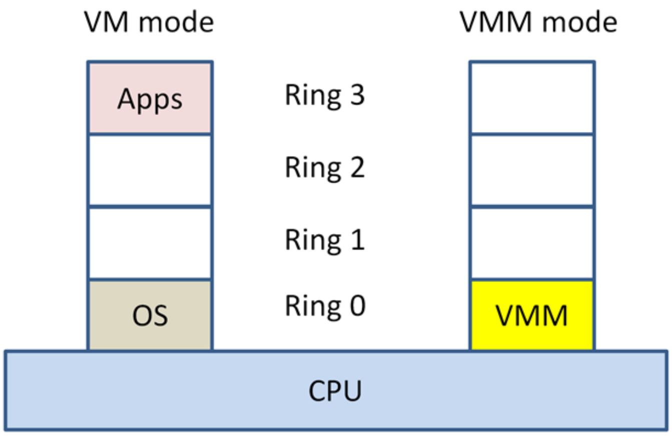


Fig.3.7 The structure of ring protection in the environment of hardware supported virtualization.

the mismatch is one of the large overhead in virtualized environment.

Fig. 3.7 shows the structure of ring protection with hardware supported virtualization. Processors with virtualization support have special modes of execution, VMM mode and VM mode. In such environment, guest OS runs in ring 0 and processors switch the mode on interrupts of hypervisor. Since no instruction handling mechanism is required in this architecture, performance overhead will be smaller.

Shadow Page Table

Shadow page table is a mechanism for avoiding a conflict of memory area of VMs. This makes an additional step to virtual address conversion and it is performed by hypervisor (i.e. not hardware but software), thus it makes large performance overhead. Processors with virtualization support implement the hardware conversion of shadow page table entry. Though entries have the same structure as hardware vendors ristricted, required time for address conversion will be reduced greatly. Many processors have multi TLB for each VMs and makes overhead furthermore smaller.

Processor vendors name this hardware-supported shadow page table while fix the structure of its entry. In the environment of VT-x of Intel, this is named **extended page table**. AMD-V of Advanced Macro Device calls it **nested**

page table.

Chapter.4

Design of Hypervisor

4.1 Assumptions

In the proposing method, a cloud service provider is assumed to offers IaaS cloud and use a data center as a storage. The provider authorizes its operator enough to do his/her operation; otherwise he/she does not obey the security policy which is presented by provider. Even if the operator is innocent, malicious operator might take over the OS of the operator. Malicious management OS might access whole physical memory area and search the critical information of users from it. A client puts its data to assigned storage, which may contain critical information. In addition, malicious operators might modify bytecode of a running application on memory to Operators of the data center might also be malicious and steal the critical information. The provider adopts Type 1 hypervisor.

In general, it is assumed that there is a hypervisor-verification mechanism such as intel TXT. Hence, an attempt to modify the hypervisor before launch will be detected as an integrity failure and users can know it. Users may inquire of the service provider whether security hazard has occured.

4.2 Overview

The proposing hypervisor protects user's data from malicious operators both from theft and modification wherever it is. Data on cache are isolated depending on processes. Cache-isolation mechanism is implemented on CPU, on the other hand, isolation of data on memory or storage is not. They are two selection of memory peotection on the proposing hypervisor. First one is based on modification of shadow page table entry and second one is base on address space layout randomization. Both techniques have their own advantages and defects which is discussed on chapter 5. Storage protection of proposing hypervisor is based on encryption.

4.3 Memory Protection Based on Shadow Page Table Entry

This section introduces the first technique of memory protection based on modification of shadow page table entry. Since this technique prohibits accesses of management VM to user's memory, this one has no uncertainty.

4.3.1 Additional Bit

Proposing architecture adds a novel authority information to shadow page table entry (SPTE). Certain bit of SPTE presents whether management VM can access the address related to SPTE or not. When hypervisor generates new instance of VM except for management VM, hypervisor assigns demanded amount of physical memory. This physical memory is converted to user's physical memory space with SPTE, and its permission bit is set true.

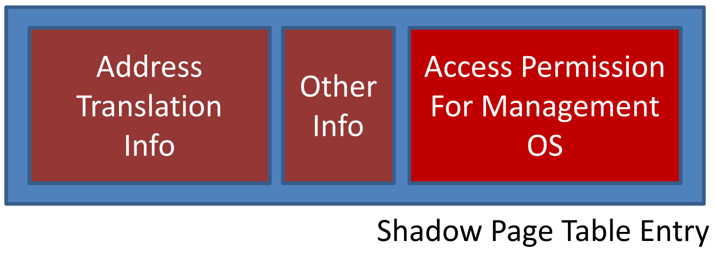


Fig.4.1 Our novel structure of shadow page table entry.

4.3.2 Safe Memory Mapping

Fig. 4.2 is the sequence of memory allocation to the management VM in order to protect user's data on memory. The sequence is as follows;

- When a management OS tries to map certain physical address, hypervisor interrputs it and receive the destination address.
- Hypervisor checks whether the destination address is already allocated to VMs (including the management OS). This check is actually performed by the verification of frame table of hypervisor. If the destination address is not allocated, hypervisor generates a SPT entry corelated to it and informs the management VM of a virtualized address.
- If the destination address is already allocated, hypervisor searches a shadow page table entry connected to it. If the additional bit of the entry is set true, the destination address is owned by user's VM. This causes access privilege fault. If the bit is set false, the access is allowed.
- Allowed access triggers the general memory mapping sequence.
- The result of access is returned to the management VM.

Note that the guest OS cannot distinguish a physical environment and virtualized environment. The novel permission bit does not violate all processes of VMs of clients.

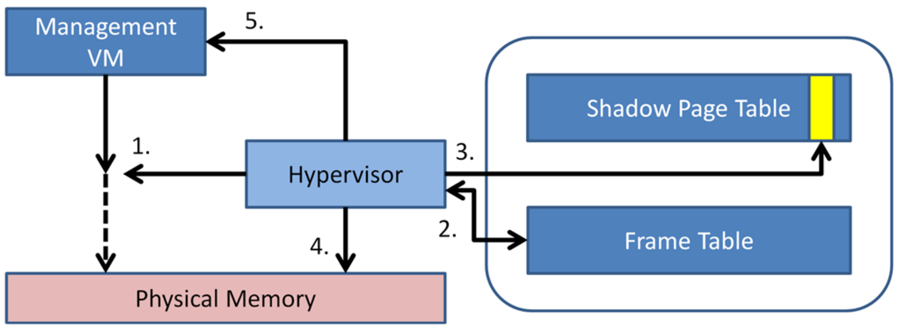


Fig.4.2 The sequence of memory allocation to the management VM.

4.4 Memory Protection Based on Address Space Randomization

Address space layout randomization (ASLR) is originately proposed to deal with a threat of buffer overflows. This technique consists of randomized layout of critical data area such as a foundation of executable file, library, heap and stack on the physical memory area. The layout randomization of such addresses makes an adversarie's expectation of an address of attacking target. Some of commercial OSes (e.g. OpenBSD, Linux, Windows) implement implement ASLR.

4.4.1 Address Space Layout Randomization on Virtualized Environment

The proposing hypervisor adopts ASLR as a concealing mechanism of user's meaningful information. Fig 4.3 shows the structure of proposing ASLR technique. While the technique of memory protection based on shadow page table entry restrics management VM's accesses to user's data on memory, one based on ASLR does not. On an environment with the proposing hypervisor, memory allocation of user's VM is randomized.

The address space whom a VM refers is already virtualized and the VM can deal with the virtualized address space as the consecutive area. This means that no modification of guest OS is required to be accommodated to the proposing hypervisor.

When a malicious operator inspects whole physical memory area, he or she can only see all the data of user's VM randomly deployed. As far as he or she does not know the function and the seed of randomization, he or she cannot construct the meaningful information from the randomized data.

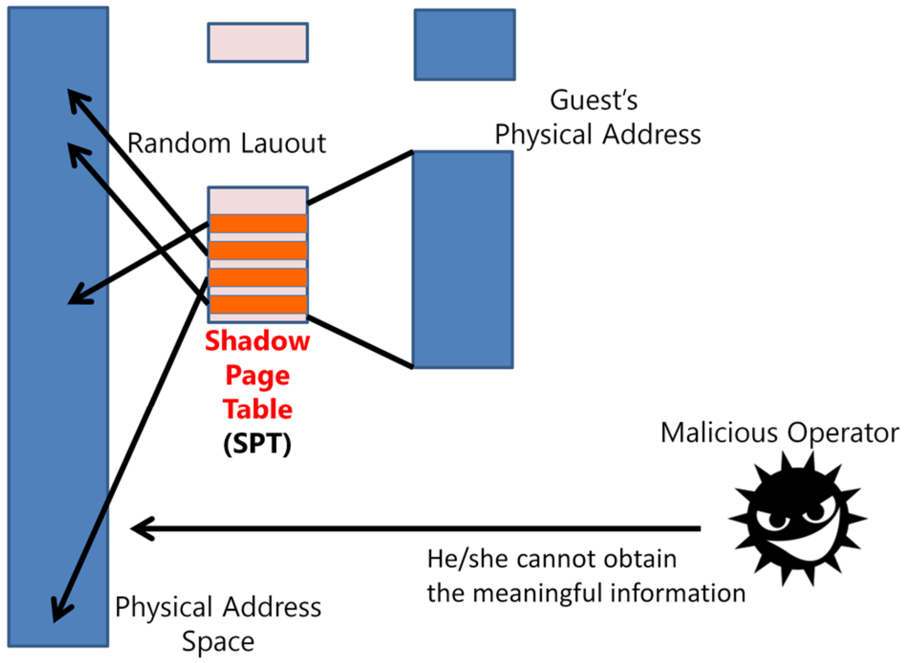


Fig.4.3 Address space layout randomization of proposing hypervisor.

4.4.2 Duplicated Mapping

The memory protection technique based on ASLR does not limit accesses of management VM to physical memory at all. This means that no restriction of modification of a malicious operator is on system. Note that the memory protection technique based on an additional bit of SPTE does not allow a management VM to access the physical address allocated to user's VM and modifications are also not allowed.

Instead of prohibiting accesses of management VM, memory duplication is implemented on proposing hypervisor with ASLR. Duplicated allocation is illustrated in fig. 4.4 Hypervisor correlate VM's physical address to two system's physical address on memory allocation. When a process on VM attempts to access a certain virtual address, an address conversion from VM's physical address to system's physical address is performed after one from virtual address to VM's physical address. Every time address coversion is performed, hypervisor checks data both on two physical addresses. If these fragments of data are

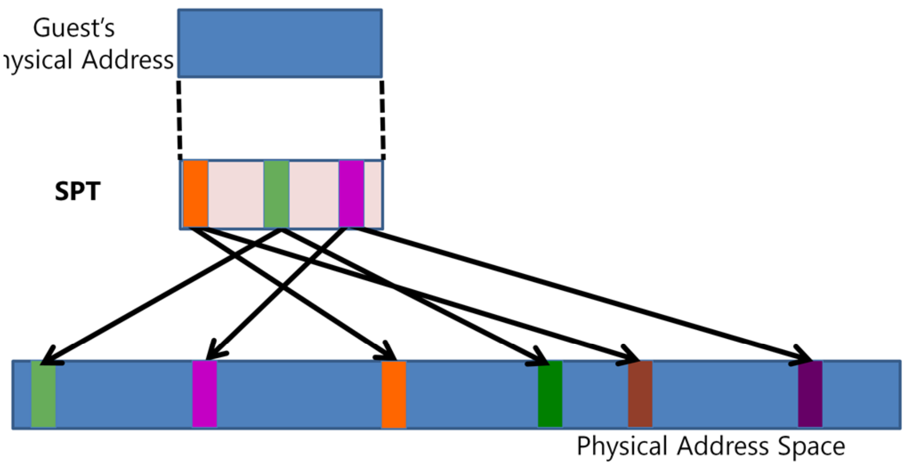


Fig.4.4 Address space layout randomization of proposing hypervisor.

inconsistent, hypervisor issues an integrity fault.

4.5 Storage Encryption

Our architecture protects a clients' information when data exists on memory from attacks of management VM ordered by malicious operators. On the other hand, when clients' information leaves memory and is stored to storage, such as swapping out and suspend, it is easily stolen by malicious operators of data centers. Our architecture encrypts all information when they left memory. Encryption keys are unique to each clients.

Hypervisor knows the relationship between a VM and pieces of physical address assigned to it. The information of this relationship is stored in the SPTE record, which is referred on allocation and release of physical memory of VM.

Guest OSes do not need to be informed of encryption keys. Both encryption on swap out and decryption on swap in are performed on hypervisor layer.

4.6 Key Management

A method of managing cryptograph keys is critical problem. Leaving client a duty of generating private key and public key pair and requiring public key does not solve this problem. This is because VM must handle the plaintext and it needs private keys. Our hypervisor has key-generating function and shares a generated key with client by the private pass such as Virtual Private Network (VPN). On second or later logging on, hypervisor requires a client to send it. Fig. 4.5 illustrates a mechanism of storage protection of proposing hypervisor.

Encryption keys are managed by hypervisor when they are on memory. The area of physical memory where keys are stored as well as SPT is not allowed to be mapped by VMs.

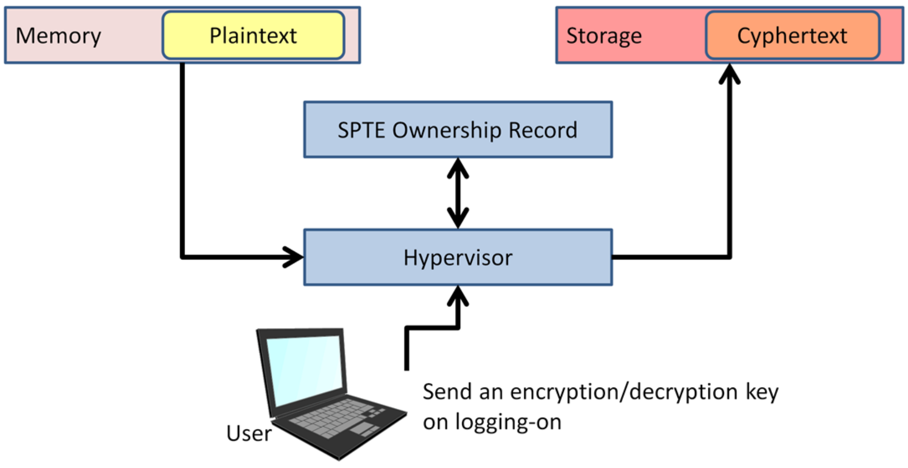


Fig.4.5 A mechanism of storage protection of proposing hypervisor.

4.7 Ensuring the Integrity of data on Storage

Though a malicious operator cannot know a meaningful structure of user's data on storage, he or she is enable to modify it in disorder. In general, modification of data on storage is harder for the user to perceive than that on memory.

To ensure the integrity of data on storage, HAIL[3] is introduced to proposing hypervisor. HAIL devides a single file to the plural fragments and disposes these fragments and parity blocks in the plural servers. It is difficult to identify the layout of them and to modify suitably. This enables hypervisor to detect an illegal modification.

4.8 Compatibility with Hardware Supported Virtualization

Though processors with virtualization support implement the mechanism of address conversion of shadow page table entry, the technique of memory protection with an additional bit of shadow page table entry can be adopted in the environment with these processors. This is because there are some extra bits in the structure of shadow page table entries and processors do not concern it, and limitation of accesses of management VM is done before address conversion.

In the case of the technique of memory protection with ASLR is more simple. Since this technique is besed only on address layout, hypervisor only need to direct a processor the randomized physical address.

Chapter.5

Discussion

5.1 Availability of Cloud Service Operation

It is a critical question whether an innocent operator of cloud service provider is able to complete his or her regular operation under an environment with proposing hypervisor. Operators are needed to have means of launching or shutting down users' VMs, killing ones and dealing with troubles caused by users.

Obviously, operators do not need to touch the memory area where the hypervisor deal with in launching operation. What operators should determine are how large the physical resource to allocate and when they launch the VM. Where the memory area will be assigned to new VM can be completely hidden in hypervisor mechanism. The case operators are required to shut down user's VM should be very rare if a steady OS is used as user's guest OS. Proposing hypervisor does not limit operators to shut down user's VM. However, shutting down causes flush of memory which is allocated to VM and swapping out. Data swapping out is protected by the technique of encryption and the key is stored until VM completely shuts down. This means no data leakage will be caused by shutting down.

Problems refers to storage is not in a range of operators' responsibility. Resolving possible troubles of users with storage do not requires the help of an administrator. On the other hand, troubles of memory such as long-time of memory leakage are normally impossible to be solved without the strong privilege of operators in system's lifetime. However, there problems can be solved simply by shutting down VM. Operators does not need to know the allocated physical memory address while all of information are hidden in hypervisor.

5.2 The safety of Cache

Most commonly, an OS manages the plural processes by performing context switch. the OS saves the value of registers and cache of the information of address conversion(a general processor retains it in Translation Lookaside Buffer(TLB)) to memory when the OS changes the running processes. Hypervisor, likewise, saves value when it changes a running VM. Accesses on memory performed by posterior VM do not result in accesses to cache TLB is cleared. This means that conventional hypervisor does not allow the leakage of cached data from a VM to another VM.

5.3 Protection of Memory

Memory Protection Based on An Additional Bit

According to section 4.1, all accesses on memory after context switch lead accesses to memory. They are always accompanied by address conversion from guest's physical address to system's physical address. This means that hypervisor can interrupt all memory accesses performed by VMs and validate an additional permission bit. Therefore a malicious operator cannot map the memory domain which is assigned to user and steal contents of it.

Memory Protection Based on ASLR

Some concerns at the memory protection of this technique. Firstly, malicious operator might ractionate the random function and seed. The more simple random function is, the higher risk of disclosing critical values is. For example, linear congruential generators (LCGs) are too simple to ensure the privacy.

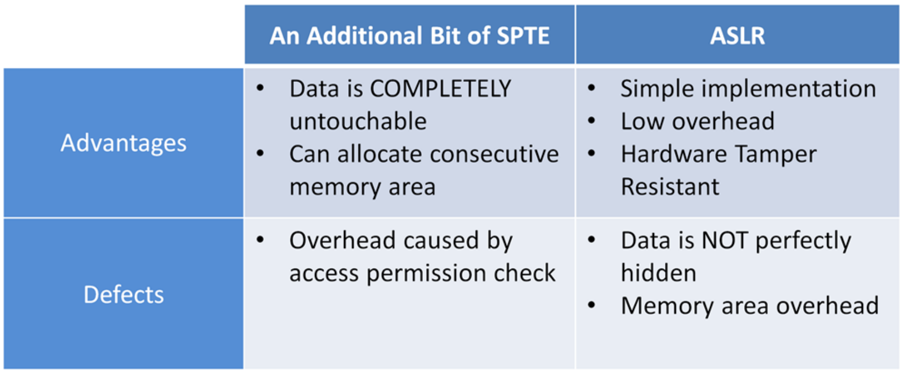
The risk of disclosure depends also on the number of VMs the platform includes. In case that great number of VMs are running and their data is randomly deployed on the physical memory area, inference of cpmsecitove data is very difficult.

The granularity of data randomization is still a critical issue. Ordinaly systems assign processes the memory area devided in 4KB size. Consistent 4KB data might be enough for adversaries to steal the secret infomation. The address-level granularity might cause the size of memory area overhead in return for resolving security issue.

5.4 Protection of Storage

Saving data from memory to storage is always followed by encryption. As far as the encryption key does not be exposed, the privacy of stored information is ensured. Encryption key is protected both on communication path and on memory by VPN and access limitation, respectively.

5.5 Comparision of Two Techniques of Memory Protection



	An Additional Bit of SPTE	ASLR
Advantages	• Data is COMPLETELY untouchable • Can allocate consecutive memory area	• Simple implementation • Low overhead • Hardware Tamper Resistant
Defects	• Overhead caused by access permission check	• Data is NOT perfectly hidden • Memory area overhead

Fig.5.1 Comparision of Two Techniques of Memory Protection.

Fig. 5.1 shows the comparision of two techniques of memory protection. An additional bit of SPTE make user's data completely untouchable from malicious operator while aslr can deal with the data even if the meaningful information is unavailable. ASLR is favorable in terms of implement cost and performance. In case of that the lifetime of VMs is short, ASLR would be adopted due to the risk of complete analysis of randomized address layout is very low. An additional bit suits the security-critical usage such as the platoform for cloud system of medical records.

Chapter.6

Conclusion

In this paper a novel hypervisor architecture for cloud computing was proposed, which ensured the privacy of cloud user's information even if provider's operator was malicious. Proposed hypervisor was focused especially when those kinds of information are in memory, but can protect them when they are in storage.

Two choices of memory protection techniques are available and each of them has its specific advantages and defects. Memory protection based on an additional bit of shadow page table entry enables strict protection but is more complex. On the other hand, memory protection based on address space layout randomization excels at the simple implementation.

Storage protection owe to encryption of data on swapping-out from memory to storage and decryption of data on swapping-in. Keys for encryption/decryption are stored in specific memory area where cannot be allocated to VMs for VM's lifetime. Proposed hypervisor requires a user his or her key when the user log on the VM. Trust of conventional cloud services is based only on their privacy policy. Proposed hypervisor does not make trust besed on it but on architecture. This makes more companies and organizations to determine to introduce the cloud services to their operations without security concern.

Bibliography

[1] A. M. Azab, P. Ning, Z. Wang, X. Jiang, X. Zhang, and N. C. Skalsky. Hypersentry: enabling stealthy in-context measurement of hypervisor integrity. In *Proceedings of the 17th ACM conference on Computer and communications security*, CCS '10, pages 38–49, New York, NY, USA, 2010. ACM.

[2] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield. Xen and the art of virtualization. In *Proceedings of the nineteenth ACM symposium on Operating systems principles*, SOSP '03, pages 164–177, New York, NY, USA, 2003. ACM.

[3] K. D. Bowers, A. Juels, and A. Oprea. Hail: A high-availability and integrity layer for cloud storage. In *Proceedings of the 16th ACM Conference on Computer and Communications Security*, CCS '09, pages 187–198, New York, NY, USA, 2009. ACM.

[4] X. Chen, T. Garfinkel, E. C. Lewis, P. Subrahmanyam, C. A. Waldspurger, D. Boneh, J. Dwoskin, and D. R. Ports. Overshadow: A virtualization-based approach to retrofitting protection in commodity operating systems. *SIGPLAN Not.*, 43(3):2–13, Mar. 2008.

[5] S. Chhabra, B. Rogers, Y. Solihin, and M. Prvulovic. Secureme: A hardware-software approach to full system security. In *Proceedings of the International Conference on Supercomputing*, ICS '11, pages 108–119, New York, NY, USA, 2011. ACM.

[6] L. Gu, A. Vaynberg, B. Ford, Z. Shao, and D. Costanzo. Certikos: a certified kernel for secure cloud computing. In *Proceedings of the Second Asia-Pacific Workshop on Systems*, APSys '11, pages 3:1–3:5, New York,

NY, USA, 2011. ACM.

[7] V. Inc. Workstation user’ s manual, 2007.

[8] Intel. What’s holding back the cloud? `http://www.intel.com/content/dam/www/public/us/en/documents/reports/whats-holding-back-the-cloud-peer-research-report2.pdf`.

[9] 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.

[10] 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.

[11] P. Mell and T. Grance. The NIST definition of cloud computing (draft). *NIST special publication*, 800(145):7, 2011.

[12] Microsoft. Hypervisor functional specification, 2008.

[13] M. of Internal Affairs and C. of Japan. White paper information and communications in japan, 2012.

[14] R. Sailer, T. Jaeger, E. Valdez, R. Caceres, R. Perez, S. Berger, J. L. Griffin, and L. van Doorn. Building a MAC-based security architecture for the Xen open-source hypervisor. In *Proceedings of the 2005 Annual Computer Security Applications Conference*, ACS ’05, pages 276–285, 2005.

[15] G. E. Suh, D. Clarke, B. Gassend, M. van Dijk, and S. Devadas. Aegis: Architecture for tamper-evident and tamper-resistant processing. In *Proceedings of the 17th Annual International Conference on Supercomputing*, ICS ’03, pages 160–171, New York, NY, USA, 2003. ACM.

[16] Trusted Computing Group. Tpm main part 1 design principles, 2007.

[17] Z. Wang and X. Jiang. Hypersafe: A lightweight approach to provide lifetime hypervisor control-flow integrity. In *Security and Privacy (SP), 2010 IEEE Symposium on*, SP, ’10, pages 380–395, 2010.

Publications by the Author

[1] K. Murakami, T. Yamada, R. S. Yamaguchi, M. Goshima and S. Sakai. A Cloud Architecture for Protectiong Guest's Information against Attacks of Malicious Operator. Computer Security Symposium, 2013. (In Japanese)

[2] K. Murakami, T. Yamada, R. S. Yamaguchi, M. Goshima and S. Sakai. A Cloud Architecture for Protectiong Guest's Information against Theft and Modification by Malicious Operator. Symposium on Cryptography and Information Security, 2014. (In Japanese)

[3] K. Murakami, T. Yamada, R. S. Yamaguchi, M. Goshima and S. Sakai. A Cloud Architecture for Protecting Guest' s Information from Malicious Operators with Memory Management. ACM Conference on Data and Application Security and Privacy, 2014.

Acknowledgement

This work is done in the Graduate School of Information Science and Technology of the University of Tokyo.

Foremost, I would like to express my special appreciation and thanks to my supervisors, Professor Shuichi Sakai and Associate Professor Masahiro Goshima, you have been tremendous mentors for me. I would like to thank you for encouraging my research and for allowing me to grow as a research scientist. Duscussions with you have been illuminating.

My sincere thanks also goes to Contract Associate Professor Rie Yamaguchi, for her encouragement, insightful comments and detailed coaching as a security reseacher.

I am deeply indebted to Harumi Yagihara and Tamaki Hasebe. Their support were invaluable for acts in laboratory.

The students of security team made enormous contribution to the inspection of research, thanks to Tsuyoshi Yamada, Mizuki Miyanaga, Kenjiro Nakata, Takuya Chida and Takuya Okamoto.

Many thanks to the following students in Sakai Goshima Laboratory: Yaoko Nakagawa, Teppei Hirotsu, Takashi Toyoshima, Naruki Kurata, Junji Yamada, Choi Seoyun, Shuji Yoshida, Soichiro Hirohata, Koske Asami, Akifumi Fujita, Manami Iwata, Suguru Nishikawa, Ushio Jinbo, Takashi Fukuda, Akihito Tsusaka, Masumi Fukuda, Yasutaka Ishikawa, Yuichi Tanigawa, Yoshiro Yamabe, Tatsuya Sugimoto, Yoshitaro Yonamoto.

Last, but not least, I wish to thank my family for their support.