IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 5, Issue 1, January 2016

Secure Virtualized Multi Tenancy Architecture in Cloud Computing using H-SVM

Sowmiya.N.D[1], Shanthi.S[2]

P.G Student, Department of Computer Science and Engineering, Valliammai Engineering College[1]

Assistant Professor, Department of Computer Science and Engineering, Valliammai Engineering College[2]

Abstract: Multilateral Security concept to multi-tenancy cloud platform. It is difficult to analyse policies defined by consumers in the same virtualization platform in order to guarantee configuration stability given that policies may have conflicts leading to unpredictable effects. sharing means that malicious activities carried out by one tenant may affect the reputation of another tenant .Multilateral security is just about allowing each of consumers to express their security requirements and actually use their chosen level of security. Multilateral Security Requirements Analysis (MSRA) consider the security and privacy interests or needs of all stakeholders related to the system.

Keywords: MSRA, H-SVM, SMM, TLB.

1. INTRODUCTION

In cloud computing, user run their programs in virtualized system and virtual machines from different users may share the same physical system. However, such cloud computing based on virtualization poses a difficult challenge to securely isolate co-tenants sharing a physical system. Security and privacy protection is more important take in cloud computing and virtualization security is more important element of process in cloud computing.

In virtualized system, hypervisor, software layer creating and managing virtual machine, is responsible for isolating any illegal accesses across virtual machine boundaries. The hypervisors, have increasing their better performance and more features in code size, verify their secure execution become too complex. If hypervisors cannot be trusted, even a trustworthy cloud provider cannot guarantee the protection of a virtual machine from a malicious person or user.The secure execution of guest virtual machine, and to guarantee the privacy of user information, memory protection and data protection across virtual machines is a critical component.

In the day today memory virtualization techniques, at the highest secure, hypervisors can control both element of memory virtualization, memory allocation, and memory isolation by address translation. set of memory pages to be allocated for a VM and maintains by using hypervisor and mapping from guest-physical to machine address table is done by using nested page table (NPT) for each virtual machineMapping transformation of memory isolation from memory allocation, both of which are performed by hypervisor. the role of a hypervisor is reduce the utilization of the memory resource allocation to the physical Memory efficiently and effectively. Updating the page mapping and setting the pointer to the nested page table to schedule a virtual machine on a core. Whenever the nested page table for virtual machine is changed, valid update is checked by hardware. Prototype used from implementation are hardware assist and secure virtual machine (H-SVM).prototype implementation are done by using system management mode (SMM).Memory isolation is reduced by using trusted computing base that hardware system is combined in the form of hardware and hypervisor.

In this paper, we pretend on the protection of guest virtual machine is based upon the securely protecting the datacentre by using hardware processor of cloud provider. With the restricted threat model, our goal is to minimize memory, data and processor from the malicious user by using hypervisor. New extensions of available hardware-assisted virtualization in commercial processors can lead to a significantly improved memory protection under an untrusted hypervisor, as long as the hardware is securely protected in the server room of the cloud provider.

Based virtual machine isolation, called hardware-assisted secure virtual machine (H-SVM) architecture, H-SVM is used to minimize the changes from the new available architectural supports for virtualization. Our approach still supports both hardware and the flexibility of software hypervisors, but the memory protection mechanism is decoupled from the hypervisors, and moved to the hardware processor.

In addition to the H-SVM design, implementation is done using the prototype system using System Management Mode (SMM) available in new commercial systems. SMM is designed to provide accesses to systems for remote management and virtual machine. Using SMM, our prototype adds security functions to the SMM layer, which the hypervisor cannot modify or access. However, the current SMM implementation is designed for the management of systems processing, not for their security, limiting the performance and functionality of our prototype system**.**

2. BACKGROUND

2.1 Hardware-Assisted Virtualization

Memory isolation in the new virtualization technology is based on the support for virtual memory with hardware address translation and page tables. In processors, with

translation look-aside buffers (TLBs) a virtual address is translated to a physical address. If the corresponding entry does not exist in the TLBs, either a HW or SW-based page table walker fetches the entry from the page table of the current address space. we assume a hardware walker - based page table walker in our architecture, as the proposed architecture aims to move the improve the responsibility of page table management from hypervisors to HW processors. For each context switch between address spaces, the page table register, which points to the top-level page table entry, must be properly set, so that the walker can traverse the correct page table.

Virtualization adds an extra translation layer. A virtual address in guest virtual machines must be translated into a guest-physical address (virtual physical address) like non-virtualized systems, and the guest-physical address is translated to a machine address in the real physical memory. The guest OS maintains the virtual to guest-physical address mapping in per-process page tables, and the hypervisor maintains per- virtual machine guest-physical to machine address mapping tables, called nested page tables. When the processor supports only single-page table walks designed for traditional native operating systems, the hypervisor maintains direct translation tables, called shadow page tables, to map virtual addresses to machine addresses directly. With the hardware-assisted virtualization, the hypervisor has its own address space, but not like guest virtual machines, the hypervisor address space uses a single translation without nested paging.

The context of each defined in a control block virtual machine (VMCB) .Hardware-assisted virtualization also facilitates World switch between a virtual machine and hypervisor environments. For example, in AMD-V architecture, the context of each virtual machine is defined in a control block of the virtual machine (VMCB). The hypervisor in the host mode, run the VMrun instruction to switch to a guest VM context. The hardware processor, by implementing micro-coded routines, hypervisor saves the current context to a specific area and restores the VM guest VMCB context of the processor. The VM context contains state registration included the pointer to the table of nested pages. If an event, you should be handled by the hypervisor, occurs, the hardware saves the guest VM context in VMCB, and restores the hypervisor context.

To isolate the memory of each virtual machine, a hypervisor must protect nested page tables illegal modifications to guest virtual machines. Guest virtual machine cannot read or change the nested page tables. In addition, for each context switch between virtual machines in a core, must change the embedded hypervisor the table on page pointer directly or by running the VMrun instruction. The hypervisor handles memory allocation Memory tracker applications of virtual machines, and can allocate and deallocate pages for a virtual machine. For such mapping changes, the hypervisor can change the nested page table the virtual machine. Note that the hypervisor also has access to its own memory space via the address translation mechanism. Since the hypervisor has full control on change nested table pages, a compromise may hypervisor read or modify the physical memory allocated for anyone virtual machine. A malicious user can allocate physical memory pages already allocated for other virtual machines to their address spaces own hypervisor or virtual machine.

2.2 Threat Model

To protect the memory of the virtual machines, even under threat hypervisor, the proposed mechanism allows only the hardware (H-SVM) to validate and update nested page tables, reducing the trusted computing base in the system hardware. The proposed mechanism can be vulnerable to hardware attacks, such as probing external buses or reading DRAM power off after. However, it is assumed that the cloud provider is trustworthy and does not intentionally attempt to compromise the system hardware. Reliable cloud provider protects its servers with physical security measures such as the supplier not only has a legal obligation not access the customer data without explicit permission, but also has a strong commercial interest in protecting his reputation.

TCB to protect memory function of virtual machines in the proposal sent system does not include the hypervisor. We assume hypervisors that are vulnerable to attack by malicious remote guest virtual machines. The threat model assumes an opponent with the permission of the hypervisor root they can attempt to access the memory of the guest virtual machines. The proposal memory system can protect the guest VM always the opponent cannot commit physical servers directly.

With appropriate security measures in the server room, have access to the server room and physically compromised systems are much more difficult to obtain root permission for remote attacks. Not sanctity hardware support, simplified requirements for H-SVM significantly. H-SVM moves the minimum functionality in traditional hypervisors for updating the page tables for nested hardware processor. The protection mechanism of memory in this work deals isolation between virtual machines or from the hypervisor. No security improves guest operating systems and applications themselves.

3. ARCHITECTURE

H-SVM improve the isolation of memory between virtual machines by blocking direct modifications of nested page tables for hypervisor. Nested page tables for the virtual machines are stored in the protected memory region that can be accessed by the H-SVM equipment. Any change in the memory allocation for a virtual machine, Hypervisor privilege level, applied to H-SVM to update the nested page table for the virtual machine.

If the hypervisor is compromised, you can try to assign a physical memory page already allocated to the address space of a virtual machine a malicious hypervisor or virtual machine. Before the update nested table pages, H-SVM checks if the applicant can violate the isolation of memory between VMs. If the physical memory Page assign a virtual machine is canceled, H-SVM clean deallocated page indicating all bytes of zeros.

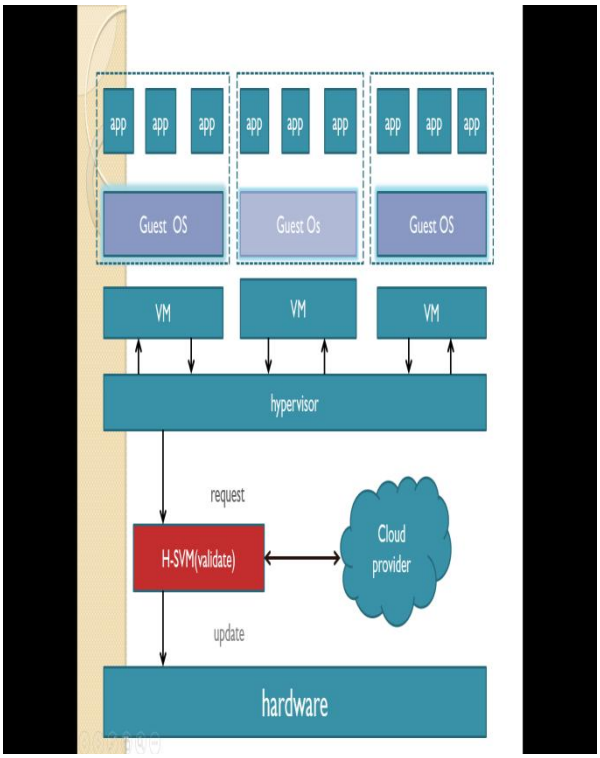
IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 5, Issue 1, January 2016

We presents the overall architecture of H-SVM. H-SVM is implemented either as a separate controller in the processor chip, or may be added subroutines microcode. These microcode routines are commonly used to implement complex features in x86. In the remainder of the paper, suppose an application like microcode. Nested tables of all virtual machines are stored in protected pages memory pages. The protected memory area is only part of physical memory, which is accessible only by H-SVM. H-SVM Blocks access to protected memory, even from hypervisor, refuse requests to the page allocation protected pages.

H-SVM maintaining several data structures, including virtual machine background information, tables of nested pages and a property page table in the protected memory area. The virtual machine context several states contains information such as address nested table top level pages, and created encryption key to the virtual machine. VM context information is similar to VMCB the AMD-V architecture. Property Page Owner-table each physical memory page, and therefore, the number of entries is greater than the number of physical System memory pages. Each entry corresponding a physical page, records the property of the page. There virtual machine hypervisor or self H SVM can be the owner of a page. If H-SVM is the owner of a page, the page is used to the protected memory area. The table on the page is the property used to check if a page request map hypervisor It is valid or not. When control is transferred to the supervisor interruptions, the United H-SVM must keep track of the current virtual CPU (virtual CPU) for VM context information.



After saving the VM state, H-SVM establishes page table pointer to the page table used by the hypervisor. When the hypervisor hours a virtual to a physical processor core, the hypervisor H-SVM requested by the execution of a privileged instruction, a virtual machine for placing a core. To request scheduling, H-SVM restores the state of the VM context information of the virtual machine, including table pointer nested pages. The current x86 to supporting a switching operation as the world VMrun instruction in AMD-V. A major difference H-SVM with the current support VMrun is that H-SVM requires context information should not be virtual machine accessible by the hypervisor. With H-SVM, the leading hypervisor normal memory management operations of the virtual machine. To create a virtual machine, we decided a set of memory pages for the new virtual machine, and made requests to update the nested page table newly created VM. The hypervisor can also dismantle memory pages of a virtual machine, often by the technique of ballooning, but real changes occur to the nested page tables H-SVM. The role of the H-SVM is limited only protected update the page tables and nested before validation any modification nested page tables. Hypervisor yet have control over the management of memory resources allocate virtual machine memory pages.

3.2 H-SVM Interfaces and Implementation

Hypervisors or virtual machines running special instructions to applications H-SVM. There are four basic interfaces to initialize virtual machine context information to update nested page tables, and to schedule a virtual machine.

Create VM. When a hypervisor to create a virtual machine requests H-SVM to create a new nested page table for the virtual machine. H-SVM virtual machine initializes the context information, and make nested page table for the virtual machine. After data structures are created in the protected memory area, H-SVM returns a virtual machine identifier hypervisor used to designate the virtual machine created for subsequent interactions with H-SVM. H-SVM also creates an encryption key by virtual machine, which will used for the exchange page requested by the hypervisor.

Delete VM. When a virtual machine hypervisor destroys requests H-SVM to clear the table of existing nested pages for the virtual machine. H-SVM also destroys the virtual machine context information. H-SVM erases the memory of the virtual machine before destroying the nested page table. Virtual machine confidentiality is guaranteed. Resets H-SVM the entry of the page table of the property due to the allocation of virtual machine memory to another. Finally, H-SVM erases the contents of virtual machine as the encryption key by virtual machine and virtual machine identifier.

Page map. To assign a physical page of memory to a virtual machine, request an operation page hypervisor card for H-SVM. A page operation of a page maps map of the machine memory (frame) a physical page invited by updating an entry in the table of nested pages. The key is to isolate the memory check ownership of a physical page of each page operation plan H-SVM. Before turning the nested page table entry H-SVM should check the title search page table if the physical page is owned by another virtual machine. If another virtual machine already has

asked the physical page, operation of the card is interrupted. When a nested page table the entry is updated to a hypervisor virtual machine on request, the virtual machine becomes the owner of the physical page. This verification mechanism prevents endangered hypervisor the creation of any illegal allocation of pages already used by other virtual machines.

Page unmap. To designate a physical memory page a virtual machine, the hypervisor is a page request unmap H-SVM. H-SVM changes the corresponding nested page table entry, and delete the contents of the memory page before the end of the operation. H-SVM also restores owner of the page on the property page table, marking as a free page. With clear, the content of free pages cannot contain customer information VM before.

Context save. When the hypervisor must program a virtual machine a core, including information VMcontext States Register They must be backed up and restored. The context contains information the pointer to the table and registration pages nested page table States. H-SVM and should protect the context information the hypervisor. Calls hypervisor save in the context H-SVM with a VM ID. While H-SVM saves record says the running kernel virtual machine in the protected memory. So hypervisor cannot touch the context information of virtual machine. This operation is similar to VMexit in AMD-V.

Context restore. When a virtual machine is scheduled hypervisor context restore request H-SVM. Charges H-SVM virtual machine general information about the basic conditions for establishing recording. As the only H-SVM can update the nested page table pointer and save the states, the hypervisor cannot force a shoot VM to use a nested table compromised pages. This operation VMrun is similar to AMD-V.

4. PROPOSED SYSTEM

Cloud computing based on virtualization poses a difficult challenge to securely isolate co-tenants sharing a physical system. Even a trust worthy cloud provider cannot guarantee the protection of a virtual machine from malicious co-tenant. With increasing demand on cloud computing, protecting guest virtual machines from malicious attackers has become critical to provide secure service. Practical design for the hardware based VM isolation called H-SVM by using this memory protection mechanism is decoupled from the VM and moved to hardware processor. We implemented a prototype system using system management mode, proving the low complexity and feasibility.

Multilateral security technology in virtualization environment and cloud computing environment. VPMS architecture can allows consumer-defined configuration, conflicts recognition and Negotiation. Multilateral Security Architecture for Virtualization platform (VPMS) to make multilateral security possible and usable for the consumer, with our following efforts:

(1)Consumer-defined configuration of the security features of VMs express security preferences.

(2)When users create, start or run a VM, conflicts can be recognized.

(3) Negotiation can overcome the problems of differing configurations and allow a non-violent resolution of conflicts that will be accepted by all VMs involved.

5. CONCLUSION

In this paper, we proposed a new architecture - the architecture VPMS using H-SVM, which derives from ideal multilateral security and may allow the configuration defined by the consumer, and to the recognition of conflict and negotiation. H-SVM to isolate the memory of the virtual machine virtualization security in present techniques is based on the support for virtual memory with the translation of hardware address and the table on page. Multilateral Security Architecture for virtualization platform (VPMS) allowing the multilateral security for consumers to ensure the configuration given that political stability may have conflicts leading to unpredictable effects.

We classify the static and dynamic potential conflicts that may arise in the platform virtualization of different behaviors. We proposed a mechanism based on hardware called H-SVM to isolate the memory of a secure virtual machine even under a compromised hypervisor. Unlike before hardware based mechanisms that support or inviolability for an attack equipment, H-SVM simplifies the complexity of the hardware supports significantly by assuming a software-only threat model. We believe this limited threat model is appropriate in the current cloud computing environments where systems can be protected against physical intrusions in a remote data center. Based the H-SVM design, we implemented a prototype system using the system management mode, which proves the low complexity and feasibility of H-SVM.

REFERENCES

1. J. Yang and K. G. Shin, “Using hypervisor to provide data secrecy for user applications on a per-page basis,” in Proc. 4th Int. Conf. Virtual Execution Environ., 2008, pp. 71–80.
2. Y. Xia, Y. Liu, and H. Chen, “Architecture support for guest-transparent VM protection from untrusted hypervisor and physical attacks,” in Proc. IEEE 19th Int. Symp. High Perform. Comput. Archit., 2013, pp. 246–257.
3. Z. Wang and X. Jiang, “Hyper Safe: A lightweight approach to provide lifetime hypervisor control-flow integrity,” in Proc. IEEE Symp. Secure. Privacy, 2010, pp. 380–395.
4. C. A. Waldspurger, “Memory resource management in vmware esx server,” in Proc. 5th Symp. Oper. Syst. Des. Implementation, 2002, pp. 181–194.
5. Survey: Cloud computing “No Hype,” but fear of security and cloud slowing adoption. (2009) [Online]. Available: http://www.circleid.com/posts/20090226_cloud_computing_hype_security
6. R. Ta-Min, L. Litty, and D. Lie, “Splitting interfaces: Making trust between applications and operating systems configurable,” in Proc. 7th Symp. Oper. Syst. Des. Implementation, 2006, pp. 279–292.
7. Trusted platform module. [Online]. Available: http://www.trustedcomputinggroup.org/developers/trusted_platform_module, 2005.
8. G. Neiger, A. Santoni, F. Leung, D. Rodger, andR. Uhlig, “Intel virtualization technology: Hardware support for efficient processor virtualization,” Intel Technol. J., vol. 10, no. 03, pp. 167–178, 2006.
9. D. Lie, C. A. Thekkath, M. Mitchell, P. Lincoln, D. Boneh, J. C. Mitchell, and M. Horowitz, “Architectural support for copy and

DOI 10.17148/IJARCCE.2016.5125

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 5, Issue 1, January 2016

tamper resistant software," in Proc. 9th Int. Conf. Archit. Support Program. Lang. Oper. Syst., 2000, pp. 168–177.

10. S. Jin, J. Ahn, S. Cha, and J. Huh, "Architectural support for secure virtualization under a vulnerable hypervisor," in Proc. 44th Annu. IEEE/ACM Int. Symp. Micro archit., 2011, pp. 272–283.
11. Kai Rannen berg. "Multilateral Security: A concept and examples for balanced security," Proceedings of the 2000 workshop on New security paradigms, Bally cotton, County Cork, Ireland, 2001, 151-162.
12. S. Gilrses, B. Berendt, and Th. Santen, "Multilateral security requirements analysis for preserving privacy in ubiquitous environments," In Proceedings of the Workshop on Ubiquitous Knowledge Discovery for Users at ECMLlPKDD 2006, pages 51-64, Berlin, September 2006.
13. D. Kuhlmann, R. Landfermann, H. Ramasamy, M. Schunter, G. Ramunno, and D. Vernizzi, "An Open Trusted Computing Architecture: Secure virtual machines enabling user-defined policy enforcement," Technical report, OpenTC consortium, 2006.
14. T. Garfinkel, B. Pfaff, J. Chow, M. Rosenblum, D. Boneh, C, "Terra: A virtual machine based platform for trusted computing," In: Proceedings of the 19th ACM Symposium on Operating Systems Principles (SOSP'03), pages 193-206, 2003

DOI 10.17148/IJARCCE.2016.5125