International Journal for Research in Applied Science & Engineering Technology (IJRASET)

Securing Guest Virtual Machine By Using Software And Hardware Processor

Shanthi.S[1], Sowmiya.N.D[2]
[1]Assistant Professor, [2]P.G Student , Department of Computer Science and Engineering
Valliammai Engineering College

Abstract:With rapid development on cloud computing, securing guest virtual machines (VMs) from malicious user has become critical to provide secure services. Today's cloud security model is based up on the software based virtualization with protecting of software hypervisor and its ethical administrator with root permission. Proposed system defines combined process of hardware and software based approach for protecting virtual machine even under cynical hypervisor. Hardware-assisted (H-SVM) used between hypervisor and memory by blocking direct modifications. Nested page tables (hardware) for VMs are stored in the protected memory region, which can be processed only by the H-SVM hardware. For any switch in memory allocation for a VM, the hypervisor, at the honored stage, makes a request to H-SVM to update the memory hardware for the virtual machine. If the hypervisor is compromised, it can try to allotment a physical memory page already defined to a virtual machine to the address space of the hypervisor or a malicious virtual machine.

Keywords: Guest virtual machine, cloud computing, cloud security, H-SVM, virtualization security.

I. INTRODUCTION

Virtualization is a risky part of cloud computing. Virtualization provides an vital layer of abstraction from physical hardware, processing the resource pooling and degree of system commonly associated with cloud computing. Virtualized operating systems is the important factor of Infrastructure as a Service (IaaS). In a virtualized system, hypervisor, software layer creation and management of virtual machines, is responsible for the isolation of illegal access to the virtual machine. In the today's memory virtualization techniques model, at the secured opportunity level, hypervisors can control both factors of memory virtualization, memory allocation, and memory isolation through address translation. A hypervisor define a set of memory pages to be allocated for a virtual machine, and maintains a mapping table from guest-physical to machine address for each virtual machine.We propose the connecting of memory isolation from memory allocation, both of which are now performed by the hypervisor. With decoupling, the role of a hypervisor is to reduced the memory resource allocation to use the physical memory properly.The hardware processor is responsible for Update of page mapping and the pointer on the frame, nested page table to plan a virtual machine on a core. Every time modified nested page tables for a virtual machine, checks the update is valid. The decoupling of confidence Computing Base for reduced storage insulation hardware system hardware combination and hypervisor in conventional mechanisms.

In this proposed system, we focus on the protection of guest virtual machines when the hardware is securely protected in the data center of a cloud provider. With the restricted threat model, our goal is to minimize necessary changes in the current processor architectures and hypervisors as much as possible.by using software hypervisor and H-SVM.

This paper proposes a practical design for the hardware based virtual machine isolation, called hardware-assisted secure virtual machine (H-SVM) architecture, minimizing the changes from the todays technological available architectural supports for virtualization. Our approach still supports 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, we have implemented a prototype system using Universal Synchronous Asynchronous Receiver Transmitter (USART) available in current commercial systems. Device which are used to translate data between parallel and serial forms. USARTS are commonly used in conjunction with communication. Our prototype adds security functions to the USART layer, which the hypervisor cannot modify or change. However, the currently used USART implementation is designed for the management of systems, not for their security, limiting the functionality and performance of our prototype.

II. BACKGROUND

A. Software Based Virtualization

In the software-based CPU virtualization, the guest virtual application code runs directly on the processor, while the guest fortunate code is translated and the translated code executes on processor. The translated code is slightly more than and usually executes more

slowly than the native natural version. As a result, guest programs, which have a small fortunate code component, run with speeds very close to native natural version. Programs with a significant fortunate code component, such as traps, system calls, or page table updates can run slower in the virtualized system environment.

B. Hardware Based Virtualization

Hardware virtualization installs a virtual machine manager or hypervisor which creates an abstraction layer between the software and hardware. Once a hypervisor is in place, software relies upon virtual representations of the computing components such as virtual processors are rather than physical processors. Popular hypervisors include VMware's vSphere and Microsoft's Hyper-VM.H-SVM improves memory isolation among virtual machine by blocking direct modifications of combined page tables by a hypervisor. Combined page tables for virtual machine are stored in the protected memory region, which can be accessible only by the Hardware-SVM hardware. For any changes in memory allocation for a virtual machine, the hypervisor, at the fortunate level, makes a request to H-SVM to update the combined page table for the virtual machine. If the hypervisor is compromised, can try to allocate a physical memory page already defined to a virtual machine to the address space of the hypervisor or a malicious virtual machine. Before updating the combined page table, H-SVM checks whether the request may disrupt memory isolation among virtual machine. If a physical memory page is de-allocated from a virtual machine, H-SVM cleans up the de-allocated page by setting all the bytes to zeros.H-SVM is to implemented either as a separate controller in the processor chip, or can be extra added as microcode routines layer. The combined page tables of all virtual machine are stored in protected memory pages. The protected memory region is just part of physical memory, which is accessible only by Hardware-SVM.H-SVM blocks accesses to the protected memory even from the hypervisor, by call on page mapping requests to the protected pages.

C. Hybrid Based Virtualization

H-SVM maintains more data structures, including virtual machine context information, a page ownership table and combined page tables in the protected memory region. The virtual machine context information data contains various states such as the address of the high-level combined page table, and an encryption key created for the virtual machine. The improved version of computing hardware, virtualization also improves flexibility in application protection and deployment. With a common hypervisor, virtual machines are no longer tied to a single server (the way that a physical application might be combined to a traditional server installation). Instead, a virtual machine on one server can be moved (migrated) to another virtual server in the local data center or servers center in any remote location while the application is still processing or running. This "live-migration" allows virtual machine to be moved as needed to streamline server performance (workload balancing) or relieve a server of its workload is in order to replace or maintain the system, yet not disrupt the applications, which can continue running or processing on other systems. In addition, virtual machine can be protected with point-in-time and backups snapshots, which can both be restored to any virtualization server without regard for the underlying hardware. Although virtual machine ware software virtualizes the CPU, the virtual machine detects the specific model of the processor on which it is running. Processor models might differ in the CPU features they offer, and applications running in the virtual machine can make use of these features. Therefore, it is not possible to use Virtual Motion to migrate virtual machines between systems running on processors with different feature sets. You can avoid this restriction, in some cases, by using Enhanced Virtual Motion Compatibility with processors that support this feature so combing the hardware assisted and software assisted error occurrence will be reduced processing speed will high, performance will be good, more helpful to large industries, IT companies, school, colleges. When error in hardware no data will loss software will maintain the data. Vice versa when any loss or error in the software assisted data will be recollected by using hardware processor.

D. Cloud-Level Support

One of the key issues to secure guest virtual machine with Hardware-SVM is to verify the integrity of newly launched guest virtual machine. An unknown hypervisor can use a wrong virtual machine image to start a new virtual machine, and provide the compromised virtual machine to its user. The cloud architecture must support the integrity checking mechanism and functionality for a virtual machine before it can become accessible to its users. To check the virtual machine image, Hardware-SVM has a mechanism to verify the hash of virtual machine image securely without intervention of the hypervisor. Furthermore, a unique pair of public keys and private is assigned to each H-SVM hardware for secure communication with a cloud managing system either beyond or within the cloud provider boundary. The details of virtual machine information integrity checking mechanism and secure communication are processed by software virtualization that is traditional virtualization technique.

III. ARCHITECTURE

A. Interfacing SVM

Hardware-SVM and software Interfaces in combined process to maintain the secure virtualization and Implementation Hypervisors or virtual machines running special instructions to applications H-SVM. There are four basic interfaces to starts virtual machine context information to update combined page tables, and to schedule a virtual machine.

B. Interface principle

Create virtual machine. When a hypervisor to create a virtual machine send H-SVM to create a new combined page table for the virtual machine. Hardware-SVM virtual machine starts the context information, and make combined page table for the virtual machine. After data structures are created in the protected or secured memory area, Hardware-SVM returns a virtual machine identifier hypervisor used to designate the virtual machine created for processing interactions with Hardware-SVM. H-SVM also creates an encryption key by virtual machine, which will used for the change or exchange page requested by the hypervisor.

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

Page map. To define a physical page of memory to a virtual machine, request an operation page hypervisor or virtual machine monitor card for Hardware-SVM. A page operation, creation or performance of a page maps map of the machine memory a physical page or actual page table invited by updating an entry in the table of nested pages. The key value is to isolate or divide the memory check ownership of a physical page of each page operation and performance plan H-SVM. Before processing the combined page table entry H-SVM should check the title search page table if the physical page or actual page is owned by another virtual machine. If another virtual machine already has asked the physical page, operation of the card is episodic. When a combined page table the entry is updated to a hypervisor virtual machine on request, the virtual machine becomes the owner of the actual physical page. This verification or authentication mechanism prevents threatened hypervisor the creation of any unlawful distribution of pages already used by other virtual machines.

Page unmap. To entitle a actual physical memory page a virtual machine, the hypervisor is a page request unmap H-SVM. H-SVM changes the consistent nested paget able 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 possessions page table, marking as a free page. With clear, the content of free pages cannot contain customer information virtual machine before.

Context save. When the hypervisor must have a program of virtual machine a core, counting information virtual machinecontext States Register They must be sponsored up and reestablished. The context comprises information the pointer to the table and cataloguing 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 virtual machine identification. While H-SVM saves record says the running kernel virtual machine in the safe memory. So hypervisor cannot touch the context information of virtual machine. This operation is similar to virtual machine exit USART.

Context restore. When a virtual machine is programmed hypervisor context reestablish request H-SVM. Charges H-SVM virtual machine overall information about the basic conditions for inaugurating recording. As the only H-SVM can update the nested page table pointer and save the states, the hypervisor cannot force a shoot virtual machine to use a combined table compromised pages. This operation virtual machine run is similar to PIP.

Volume 4 Issue III, March 2016
ISSN: 2321-9653
International Journal for Research in Applied Science & Engineering Technology (IJRASET)

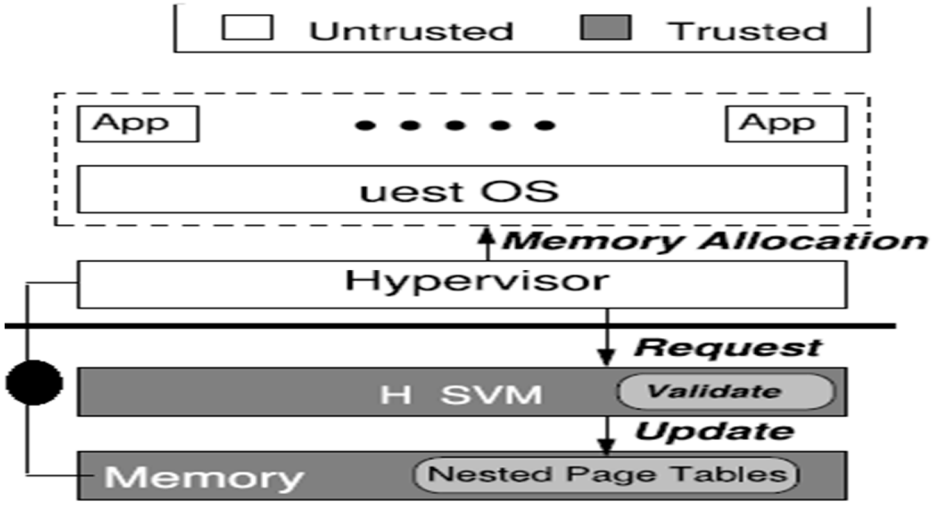

Fig 1. hybrid assisted virtualization.

IV. PROPOSED SYSTEM

Proposed system defines combined process of hardware and software based approach to protecting virtual machine even under mocking hypervisor. Hardware-assisted (H-SVM) used between hypervisor and memory by blocking direct modifications. Combined page tables (hardware) for VMs are stored in the protected memory constituency, which can be processed only by the H-SVM hardware. For any switch in memory allocation for a virtual machine, the hypervisor, at the honored stage, makes a request to H-SVM to update the memory hardware for the virtual machine. If the hypervisor is compromised, it can try to share a physical memory page already defined to a virtual machine to the address space of the hypervisor or a malicious virtual machine. When error in hardware no data will loss software will maintain the data. Vice versa when any loss or error in the software assisted data will be recollected by using hardware processor.

V. CONCLUSION

We proposed a hardware-based mechanism called H-SVM to isolate the memory and software to maintain the performance of a virtual machine securely even under a negotiated hypervisor. Error occurrence will be reduced when we combined the both hardware software and hardware assisted with high performance rate. Our prototype adds security functions to the USART layer, which the hypervisor cannot modify or change. H-SVM blocks accesses to the protected memory even from the hypervisor, by call on page mapping requests to the protected pages.

REFERENCES

[1] Z. Wang and X. Jiang, “HyperSafe: A lightweight approach to provide lifetime hypervisor control-flow integrity,” in Proc. IEEE Symp. Secur. Privacy, 2010, pp. 380–395.
[2] Windows Azure Platform. (2010) [Online].Available:http://www.microsoft.com/windowsazure/
[3] 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.
[4] 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.
[5] D. Lie, C. A. Thekkath, and M. Horowitz, “Implementing an untrusted operating system on trusted hardware,” in Proc. 19th ACM Symp. Oper. Syst. Principles, 2003, pp. 178–192.
[6] D. Lie, C. A. Thekkath, M. Mitchell, P. Lincoln, D. Boneh, J. C.Mitchell, and M. Horowitz, “Architectural support for copy and tamper resistant software,” in Proc. 9th Int. Conf. Archit. Support Program. Lang. Oper. Syst., 2000, pp. 168–177.
[7] R. C. Merkle, “Protocols for public key cryptosystems,” in Proc.IEEE Symp. Secur. Privacy, 1980, pp. 122–134.
[8] G. Mi»_os, D. G. Murray, S. Hand, and M. A. Fetterman, “Satori:Enlightened page sharing,” in Proc. Conf. USENIX Annu. Tech.Conf., 2009, pp. 1–14.
[9] D. G. Murray, G. Milos, and S. Hand, “Improving xen security through disaggregation,” in Proc. 8th ACM SIGPLAN/SIGOPS Int. Conf. Virtual Execution Environ., 2008, pp. 151–160.
[10] 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.
[11] D. Gupta, S. Lee, M. Vrable, S. Savage, A. C. Snoeren, G.Varghese, G. M. Voelker, and A. Vahdat, “Difference engine:Harnessing memory redundancy in virtual machines,” in Proc. 8th USENIX Conf. Oper. Syst. Des. Implementation, 2008,pp. 309–322.
[12] J. A. Halderman, S. D. Schoen, N. Heninger, W. Clarkson, W. Paul,J. A. Calandrino, A. J. Feldman, J. Appelbaum, and E. W. Felten,“Lest we remember: Cold-boot attacks on encryption keys,” Commun.ACM, vol. 52, pp. 91–98, May 2009.
[13] Intel, “Intel virtualization technology for directed I/O Architecture Specification,” Number D51397-007, Rev. 2.3, 2014.
[14] 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. Microarchit., 2011, pp. 272–283.
[15] R. B. Lee, P. C. S. Kwan, J. P. McGregor, J. Dwoskin, and Z. Wang,“Architecture for protecting critical secrets in microprocessors,” in Proc. 32nd Annu. Int. Symp. Comput. Archit., 2005, pp. 2–13.
[16] T. Garfinkel, B. Pfaff, J. Chow, M. Rosenblum, and D. Boneh,“Terra: A virtual machine-based platform for trusted computing,” in Proc. 19th ACM Symp. Oper. Syst. Principles, 2003, pp. 193–206.