Dynamic Partitioning of Physical Memory Among Virtual Machines

[ASMI:Architectural Support for Memory Isolation]

Jithin R[*]
Dept. of Computer Science and Engineering
NIT Calicut, India
jithinr550@gmail.com

Priya Chandran[†]
Dept. of Computer Science and Engineering
NIT Calicut, India
priya@nitc.ac.in

ABSTRACT

It is an open challenge for virtualization technology architects to provide security to Virtual Machine (VM), in the presence of an infected hypervisor, without much compromise on performance. A few hardware modifications have been introduced by manufactures like Intel and AMD to provide a secure VM environment with low performance degradation. These solutions are unable to provide VM isolation in the presence of an infected hypervisor. In this paper we propose a novel memory architecture model, that can achieve a secure physical memory region to each VM without performance degradation.

CCS Concepts

•**Security and privacy → Virtualization and security;**
•**Computer systems organization →** *Cloud computing;*

Keywords

Virtualization Security; VM Isolation; Secure VM; Secure Cloud; Hardware-assisted Virtualization

1. INTRODUCTION

Protection of user data at different levels of architecture like CPU, memory and I/O devices has to be provided, proved and assured to convince the users of the credibility of the system [4] [5] [6]. This paper concentrates on the memory level protection.

We review literature in the area of memory virtualization to identify open challenges that prevent hypervisors from providing a secure physical memory region to VMs without compromising much on performance. A memory architecture model, ASMI (Architectural Support for Memory Isolation) has been proposed in this paper, aimed at solving those challenges.

The necessary features for any technology designed for improving VMs in terms of security and performance are identified in Section 2. Description of the design of ASMI is given in the Section 3. Section 4 concludes the article with future directions of research.

[*]Research Scholar
[†]Professor

SAC 2016,April 04-08, 2016, Pisa, Italy
©2016 ACM. ISBN 978-1-4503-3739-7/16/04…$15.00
DOI: http://dx.doi.org/10.1145/2851613.2851870

2. SECURITY THREATS

Memory protection in the presence of infected hypervisors is an open research problem [8]. In this context, protection of memory from the hypervisor level to the hardware level are desirable [8].

HyperWall [8], an extension to the IOMMU [3] hardware unit provides hardware for protecting guest VMs from malicious hypervisors. HyperWall architecture aims to protect only *confidentiality* and *integrity* of VMs data and not *availability.*

Hyperwall allows the VM user to set the memory page with a protection level which denies access to DMA and hypervisor. Hence, any malicious VM can use this feature to create memory starvation for other VMs by locking several pages at a time. A secure memory architecture should provide the assurance of a minimum and fair amount of memory to all VMs at all point of their lifetime.

Extension of our survey on the other availability challenges shows that a huge volume of unutilized memory is allocated to VMs [1] [2]. In the current architecture, the requested physical memory is divided and allocated to VMs when they are created. Existing solutions [1] [2] to the underutilization issue are susceptible to covert channel based attacks by colluding VMs on the same server.

Clearly the need of the day is to have a memory virtualization infrastructure that supports the following properties even in the presence of a malicious hypervisors.

- Isolated physical memory region to each VM
- Fair allocation of physical memory among VMs

3. ARCHITECTURAL SUPPORT FOR MEMORY ISOLATION

We propose *ASMI, Architectural Support for Memory Isolation*, that can satisfy the above requirements. Figure 1 illustrates the proposed architecture.

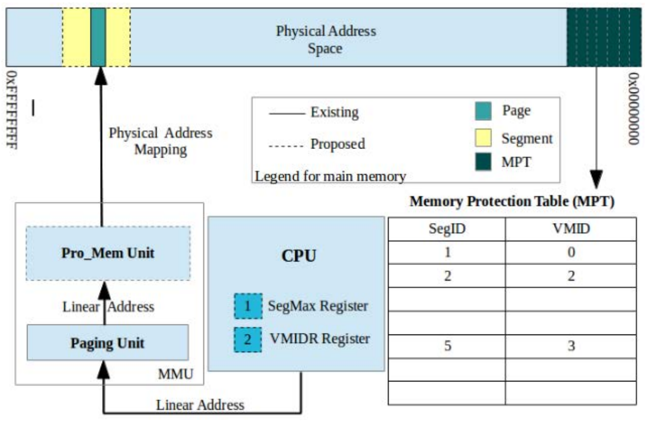

Figure 1: ASMI: Architectural Support for Memory Isolation

We propose some modifications in the current Intel-VT architecture [3] to incorporate ASMI architecture. The modifications to the memory management unit, the CPU (Central Processing Unit) and ISA (Instruction Set Architecture) are described in the rest of this section.

3.1 Memory

In the Intel 64 bit architecture, the address translation mechanism disables segmentation and only paging is utilized [3]. In ASMI, segmentation is enabled and used to provide isolation to VMs. The physical memory is partitioned into a number of physical segments having fixed number of pages. A hardware unit *Pro-mem*, controls the entire physical memory.

A data structure in hardware named *Memory Protection Table (MPT)* is proposed, and is managed by *Pro-mem*. *MPT* contains the segment ID (SegID) and its corresponding virtual Machine ID (VMID). Each segment can be assigned only to a single VM at a time. *MPT* is stored in the primary memory as shown in Figure 1. The part of main memory containing *MPT* is designed to be accessible to the *Pro-mem* unit only. At system boot time, *MPT* would be empty.

3.2 Central Processing Unit (CPU)

A CPU register named VMIDR is introduced in this architecture, for each processor in the system, and its purpose is to store the identity of the currently running VM on that processor. A unique ID is assigned and stored in VMIDR register by *Pro-mem* when a new VM is created. Switch between VMs on a processor causes a change in the VMIDR value, done by the *Pro-mem*.

VM or Hypervisor boot up is informed to the *Pro-mem* unit through the modified VMLAUNCH instruction. When the *VM exit* instruction is executed, VMIDR contents get stored in the initial address of the first memory segment of the corresponding VM and the VMIDR is loaded with the initial address of the first memory segment of the hypervisor. Similarly when the *VM entry* instruction executes, the VMIDR value is stored in the initial address of the first segment of hypervisor and the VMIDR value is loaded from the initial address of the first segment of the running VM. These load and store operations are executed by *Pro-mem* as atomic operations to *VM exit* and *VM entry* instructions.

SegMax is a CPU register, whose values are managed by *Pro-mem*. Segmax stores the maximum number of segments(MSEG) that can be assigned to a VM when the entire physical memory is full. The value stored in MSEG is computed as the quotient of the total number of physical segments (TSEG) in primary memory, and TOT, where *TOT* is the sum of total number of VMs and hypervisor running above the hardware. These values are computed by the *Pro-mem* unit upon the creation or deletion of a VM

3.3 Instruction Set Architecture (ISA)

In the proposed architecture, *VM Entry* and *VM Exit* instructions are modified to inform the *Pro-mem* unit about the control transfer among VMs and hypervisor in an atomic manner.

SegMax, MSEG, and TOT values are changed when a new VM is created or when an old one is destroyed. VM creation and deletion are informed to the *Pro-mem* unit by modifying the *VMLAUNCH* and *VMCLEAR* instructions in a similar manner as VM Exit and VM Entry instructions, which are used for managing VMCS entries in Intel-VT architecture [3].

TSEG is fixed at boot time, by the hypervisor, depending on the page size that the hypervisor is designed to work with. Initially at boot time, MSEG and TOT are zero. When a hypervisor is loaded or a VM is created, TOT is incremented by one and MSEG is recomputed according to the new TOT value using the modified *VMLAUNCH* and *VMCLEAR* instructions.

These architectural modifications in CPU, memory and ISA facilitate the required modifications in memory operations. The memory operations in ASMI are described next.

3.4 Memory operations

Memory operations in ASMI are explained with reference to the changes in memory allocation and memory access. We propose *Pro-mem* to be installed in between hardware paging architecture and primary memory, by modifying paging unit to include *Pro-mem* functionality. Since all the communication between *Pro-mem* and VM are done through the paging unit interface, no modification is required in the guest OS.

We proposed a new memory page allocation algorithm for security, as shown in Algorithm 1, that records the allotted VM of each memory segment.

ASMI page allocation algorithm ensure a minimum number of physical segments (physical memory) to each VM when the physical memory need is at its peak. It simultaneously ensures that physical memory is not be left free or unutilized

Algorithm 1 ASMI Page allocation

```
Input: page_struct, virtual_address
Output: physical_address of main memory
 1: Get the list of segments allotted to the VM in MPT
    table with ID in VMIDR register
 2: if A free page_frame in allotted segment exists then
 3:     Copy the data, in secondary memory corresponds to
        the page_struct, to page_frame in main memory
 4:     return physical_address of the page_frame
 5: else if Any VM has more than MSEG segments then
 6:     Free one segment of that VM by requesting the cor-
        responding VM to swap ¹
 7:     go to 1
 8: else
 9:     return memory_full_exception
10: end if
```

[7] when a VM requires it.

We propose a page access algorithm that ensures that VM accesses only the allotted segments in *MPT*. Our proposed page access algorithm is as shown in Algorithm 2

Algorithm 2 ASMI Page access

```
Input: physical_address obtained from page table
Output: non_zero if allowed & zero if not permitted the
access
 1: Calculate the segment_ID of the physical_address
 2: Get vmid of the corresponding segment_ID from MPT
    table
 3: if VMID equal to VMIDR register value then
 4:     return non_zero
 5: else
 6:     return zero
 7: end if
```

Access to segments is validated by *Pro-mem* with the help of VMIDR value and *MPT* table. *Pro-mem* returns zero to paging unit when a VM or hypervisor tries to access the segment that is not allotted to it.

3.5 ASMI deployment

ASMI, an architectural solution that requires hardware modifications, ruling out testing through implementation due to the high implementation costs. To prove that our architecture is *feasible* with the current working environment of hypervisor, we verified it through emulation in the hypervisor kernel.

We chose XEN open source hypervisor as the platform for emulation. While deploying ASMI in kernel level, we assumed the segment size of ASMI architecture as a single page. *Pro-mem* has been emulated using three new files in XEN. Emulation of all the new registers proposed are done through global variables. Hardware signals are emulated using the functions.

Implementation of all these functionalities in the XEN kernel could successfully emulate ASMI on a XEN kernel without modifying the guest OS. Thus, emulation of ASMI on the XEN hypervisor could prove that the design of ASMI is feasible with the current environment of virtualization.

4. CONCLUSIONS AND FUTURE WORK

This article presents security challenges in address translation mechanism of VM technology. Our reviews shows that issues like fair allocation and isolation have to be considered for performance improvement and security of VMs.

As a solution to the above mentioned issues, a memory architecture model named ASMI, has been proposed and described in the paper. ASMI provides an isolated memory region to each VM and the hypervisor. ASMI has been illustrated in this paper on an Intel Platform with hardware enhancements to implement the design. To prove that the model is feasible with the current virtualization environment, a XEN kernel level emulation has been implemented. ASMI's enhancement to the Intel-VT architecture can thus provide confidentiality, integrity and availability to VMs through complete memory isolation, irrespective of hypervisor security without compromising performance.

As a part of our future work for performance comparison, we plan to identify parameters for performance comparison of VMs and explore the possibilities for comparing the performance of VMs on IOMMU and ASMI enabled hypervisors.

5. REFERENCES

[1] O. Agmon Ben-Yehuda, E. Posener, M. Ben-Yehuda, A. Schuster, and A. Mu'alem. Ginseng: market-driven memory allocation. In *Proceedings of the 10th ACM SIGPLAN/SIGOPS international conference on Virtual execution environments*, pages 41–52. ACM, 2014.

[2] J. Hwang, A. Uppal, T. Wood, and H. Huang. Mortar: filling the gaps in data center memory. In *Proceedings of the 10th ACM SIGPLAN/SIGOPS international conference on Virtual execution environments*, pages 53–64. ACM, 2014.

[3] Intel. *Intelő 64 and IA-32 Architectures Software Developers Manual, Combined Volumes: 1, 2A, 2B, 2C, 3A, 3B and 3C*, February 2014.

[4] R. Jithin and P. Chandran. Virtual machine isolation. In *Proceedings of the Second International Conference on Security in Computer Networks and Distributed Systems (SNDS 2014)*, pages 91–102. Springer, March 2014.

[5] M. Pearce, S. Zeadally, and R. Hunt. Virtualization: Issues, security threats, and solutions. *ACM Computing Surveys (CSUR)*, 45(2):17, 2013.

[6] A. Rehman, S. Alqahtani, A. Altameem, and T. Saba. Virtual machine security challenges: case studies. *International Journal of Machine Learning and Cybernetics*, pages 1–14, 2013.

[7] M. Swanson. Security self-assessment guide for information technology systems. Technical report, DTIC Document, 2001.

[8] J. Szefer and R. B. Lee. Architectural support for hypervisor-secure virtualization. *ACM SIGARCH Computer Architecture News*, 40(1):437–450, 2012.