RESEARCHERID
THOMSON REUTERS

[ICRTCET-2018]

ISSN 2348 – 8034
Impact Factor- 5.070

GLOBAL JOURNAL OF ENGINEERING SCIENCE AND RESEARCHES

MONITORING VIRTUAL MACHINES IN CLOUD

Indumathy M
Research Scholar, Department of Computer Science, Visvesvaraya Technological University, Belgavi, India

ABSTRACT

cloud customers need assurances regarding the security of their virtual machines operating within an Infrastructure as a service in cloud system. This paper presents an architecture that can monitor a virtual machine's security and identify the threats by combing the in vm and out Vm monitoring methods, take the appropriate actions in parallel without any delay.

Keywords: *security, hypervisor, cloud computing, virtualization, virtual machine Monitor*

I. INTRODUCTION

Cloud computing mainly relies on virtualization technology due to decrease in hardware cost and advancement of computing power of the hardware. Virtualization allows customers to have single machine with multiple OS and applications. The cloud provider maintains the cloud server and provides the VM for each client and allocates the requested amount of resources like processors memory, networks and disk. In an infrastructure-as-a-service, a user can request to launch a virtual machine (VM) in cloud. Monitoring the virtualization is a major consideration in the field of cloud computing. There are various monitoring methods developed recently, it can be classified mainly as performance monitoring (CPU, memory, network and device drivers) and security monitoring. In this paper we consider the security monitoring.

II. OVERVIEW OF VIRTUALIZATION.

As depicted in Figure 2.1 a virtualized environment, has a Virtual Machine monitor also called hypervisor, Interface between each VM and the physical hardware. VMM are classified into two types Type1 and Type2. In a type 1 VMM runs directly on the hardware, Type1 VMM are like VMware ESX, Xen, Microsoft Hyper-V. In a type 2 VMM, runs on the host operating system. It uses OS to interact with physical hardware. VMware Workstation, QEMU, KVM are the examples of Type 2 VMM [1].

A. Working of Virtualization.

In order to determine the working of virtualization we'll take some event sequence which happens when a process tries to access the memory address in its virtual address space. Each process assumes that they access the memory address directly, however the host OS has a role in providing the memory to the virtual machines, this details is hidden from the each process. Generally a process access the page table to map logical address with the physical address, but in case of virtual machines it adds another level of complexity. Virtual Machine monitor is an abstraction between the VMs and the host operating systems. VMMs translates the Page frame number requested by the VM's process to frame number of the physical hardware. Since all the process under every VMs are running on the same platform VMs must be separated carefully. VMM has the major role in all the activities done by every VMs, it has highest privileges to access pages of all VMs present in the cloud platform [2].

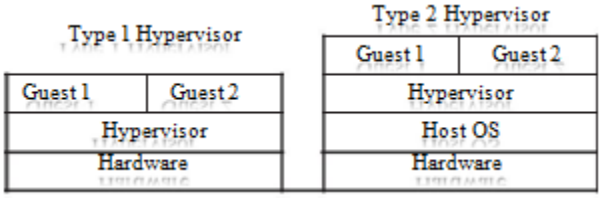

Figure 2.1 Type1 and Type2 Hypervisors

Recently, cloud providers provide APIs for customers to monitor internal status (Resources utilized) of their VMs [3], but customers cannot monitor the security part of their virtual machines. Monitoring the security for VM are great challenges in cloud. Many researchers proposed methods to monitor the internal states of guest OS, these methods are having limitations: 1) few methods required monitor code to be implemented within OS kernel. If the Host OS is compromised attackers take the control of all the VMs running in the same environment 2) some methods requires separate hardware to monitor which is difficult for wide deployment. 3) Some methods make use of the Virtualization technology 4) some methods are measures the static attributes of the Virtual machines and other collects the status of the virtual machine and analyses for attacks serially.

5) Some methods are based on In VM monitoring some methods are Out VM monitoring.[4] To address the limitations we have proposed a framework which combines both In VM an Out VM monitoring methods helps to monitor the virtual machines concurrently.

B. The Xen Architecture

Xen is Type I virtualization system it is sandwich between operating system and the hardware. In Xen, virtualization is a software also called hypervisor, which provides virtualization which communicates with the hardware acts like a Host OS. Inter VM communications are very crucial. According to X86 architecture, there are four defined rings naming from ring 0, highest level to ring 3 lowest level. The virtual machine runs in the ring 0, user applications are running in ring 3. Similar to the system calls, hyper Calls transfer the control from user applications (ring 3) to kernel Layer(ring 0). Shared memory technique is used for inter-VM communication. To share memory page with others VM, Virtual machine checks with the grant table for the permissions. If the domains needs access to memory it makes a hypercall and hypervisor takes decision by looking to the grant table. Dom 0 is having the highest privilege, The Dom 0 can map the domains to the address space called as foreign mapping.

III. INTER DOMAIN COMMUNICATION.

Xen uses many methods for inter domain communication each like grant table , buffers and event channels we use these channels to send the event notification to Trusted VM i,e domain 0. has two drivers for network and disk access two modules has to be used one acting like front end and other as a back end driver. Domain users use the ring buffers to transfer the event recorders information between the domains, the Xen use gntab-grant_foriegn_access(), gntab_end_ foriegn_access() and Hypevisor _grant table() to control the access of shared memory [4] . DomU initially establishes a new event channel for notifying the events information Dom0 using the HPERVISOR_event _channel -_op(). The Dom0 the event analyzer module fetches these information by using the Xenbus_scanf(). Dom maps the ring buffers to its address space by using the SOR_grant_table_op(). Dom0 binds the event handler with events received from DomU with the help of **bind_interdomain_evtchn_to_irqhandler** and analyses the event and verifies with the Database rules, if any violations exists i,e notified to system administrator to take the appropriate actions like(suspend , terminate the VMs).

IV. CHALLENGES.

Existing monitoring solutions do the event collection and analyzation in a serial technique [4] .when the VM tries to perform some operation (like executing program, opening a file) the event collector will intercept the operation and check the privileges of the operation if it finds abnormality the information, it is stored in the kernel buffer and is passed to the Trusted VM through the event channel or the ring buffer mechanism. Analyzer present in the Trusted VM analyses the Information and checks with the administrator policies and respond according (e.g suspend or Kill the Process running in the target VM.)

V. FRAME WORK OF OUR MODEL

The basic idea in our framework is to isolate the event recorder and the analyzer into the separate components, which works simultaneously without disturbing the domains and also it increases the time efficiency. Each Dom running in VM will intercept the system events related to security and store in the kernel's buffer and transfer the

status of events to the Trusted VM where the analyzer analyses the events and takes the necessary actions, if it finds the unusual events. Existing methods monitors perform the activities in serial order [4]. The Analyzer component and the event recorder component can be deployed in the user space of the Domains so it is easy to implement. Event recorder component performs the buffer mapping from the user address space to virtual address space

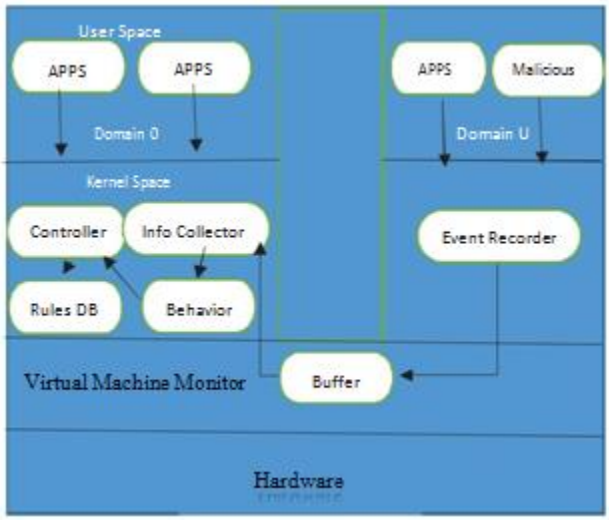

Figure 3.1 Virtual machine Monitor

Event Recorder Module.

To invoke the parallel working of the Recorder and Information analyzer we create the two process on both the sides. When a process in domU performs an event the recorder will capture that event and store in buffer in order to show the implementation of the model we take the powers of Linux kernel file System.map is a symbol table used for mapping the function names with Virtual address in memory. We make use of SYSCAL and SYSENTRY instructions. In x86_32 system call has a sys_read. The sysenter_target entry point writes to a model-specific register in the kernel start function SYSENTER instruction. SYSENTER sets the process to high Privilege and the process running from user space jumps to the kernel space through the Modern ABI through the system call number 3 (3 for read()) into the register EAX and the remaining parameters in registers EBx, ECX and EDx) and then invoke SYSENTER instruction. This causes the code to push into the kernel stack and the function pointer at the EAX, which performs like a wrapper for the real implementation Of SYSC_read().

A. Xen has three layers of memory, virtual, pseudo physical and physical. The pseudophysical memory is intermediate between machine and memory. The system actually uses the vMs.

B. Xen allocates the machine frame number and PFN (physical frame number) to a page of physical memory. The domains

X. Maintains data structure called physical to memory mapping which converts the physical frame number.

Δ. We add the hypercall provided by xen HYPERVISOR_map_by Vaddr_op. To this hyper call two arguments is passed i.e domain id and map_request_t structure which has the parameters for the operation.

The map request is designed like

Struct map_req {

Domid _t Dom_id;

Long Viraddr;

int bytes;

XEN_GUEST_HANDLE (ulong) start1; };
typedef struct map_ req map1;

- Using this hypercall, Domain0 fetches the memory areas of Domain u which conforming to virtual viraddr to viraddr+bytes

- Next the Domain 0 contacts these memory areas in its self-virtual address space,

Event/Info Analyzer Module.
On the other hand the Info Analyzer module receives event related data from the buffer and extracts the behavior of the particular Virtual machine and classify based on the threats. And passes to the pattern recognition system, it checks the Database Rules which contain the security policy. If the threat is detected it takes the appropriate action like terminate the VM, suspend or to migrate.

Example of security policy

Security polices present in database rule specify system calls with pattern matching of the syscall arguments.

Default: column specifies the action to be taken.

PolicyFile default

NetworkCond ip(ipaddr) | port(portnum)

Action allow | deny(retnum)

killProc | policyChange(policyfile)
Condition FileCond | NetworkCond | Condition and

Condition | Condition or Condition FileCond

fileEq(argnum,path) | filePrefix(argnum,pathPrefix)

| killProc | policyChange(policyfile)

VI. ALGORITHMS

Algorithm for monitoring the VMs.
- Start.
- Invoke the virtual machines VM1, VM2, VM3 etc depending on the cloud customer's request.
- Perform the operations for all VM's i=1 to n.
- Monitor the event status through system calls. (like request to open a file . read status.
- Record the events and store in the buffer for concurrent access.
- Info analyzer directly maps the memory to the buffer space and receives the events.
- Verifies with rules present in database and performs the necessary action like suspend, Terminate that particular VMs were the suspect ion is present.

- End

ISSN 2348 – 8034
Impact Factor- 5.070

Experimental setup.
The Experiment was carried out on Lenova Thinkpad with dual core Intel with 2.10 GHZ of CPU and made use of Xen to act as Virtualization Monitor we took linux and Ubuntu as a VMs i,e Dom u and Dom 0.

To test our model sample application was written in the Domain U, compiled and run here is the main program

```
Fd =open (“welcome.txt”, O CREAT | O RDWR, \ S IRUSR |S IWUSR);
If (fd) {

Write (fd,”welcome world”!” strlen(“welcome world”));
Close (fd);

}
```

Algorithm 2 (VMM)

5) Start

6) Input : { event behavior , VirtualmachineId}

7) Extract the event behavior of VMi

8) Find and classify based on threats to be handled.

9) Forwards the category of threat detected by VM to pattern recognition system.
10) End.

Algorithm 3: (pattern recognition system)

1. Start

Input: {virtual machine id, threat category, Virtual machine memory map}
3. Extracts the threats classified for specific VMi to match the rule in database.

4. If pattern is matched then apply the respective rule to the VMi

-find the action to be taken on the target virtual machine.

Else

-Define the new pattern for VMi -Store in the database.

-find the action to be taken on the target virtual machine.

5. Take appropriate action like (migrating or terminating, pause);
6. End.

ALGORITHM FOR VMM

1. input

2. VM={ VM1, VM2,VM3, …… , VMn } no of virtual machines present in cloud.

3. For each VM in cloud

4. Track System call tracing, Interrupts, Memory map.

5. Record the events inside VM.

6. Send the event to Dom 0.

7. Categorize the threats.

8. Pass to Analyzer Module.

9. Controller uses Database and search for particular pattern. Using Pattern Recognition Algorithm = {Patterns of VM attack:

10. If Pattern Exist then take appropriate actions

11. Else Update the Pattern Rule DB

12. Output:

13. Action = {Migrate, Pause, Terminate} Action

VII. CONCLUSION

In this paper we present a parallel monitoring of security in virtual environments which is different from traditional serial methods of monitoring , it combines both In VM monitoring and Out VM monitoring methods. The proposed model is time efficient since both the event collector and analyzer are working in concurrently.

REFERENCES

1. *Gabriel Cephas Obasuyi, Arif Sari "Security Challenges of Virtualization Hypervisors in Virtualized Hardware Environment" ,Scientific research publishing , Int. J. Communications, Network and System Sciences, 2015, 260-273*
2. *Kara Nance and Brian Hay University of Alaska, Fairbanks, IEEE security and Privacy "Virtual Machine IntroSpection" September/October 2008 page 32*
3. *Amazon Elastic Compute Cloud [OL]. http://aws.amazon.com/ec2/.*
4. *TIAN Donghai ,JIA Xiaoqi,CHEN Junhua , HU Changzhen " A Concurrent Security Monitoring Method for Virtualization Environments ,Security Schemes and solutions , china communications January 2016.*
5. *An In-VM Measuring Framework for Increasing Virtual Machine Security in Clouds.*
6. *tps://lwn.net/Articles/604515.*
7. *Koichi Onoue , Yoshihiro Oyama, Akinori Yonezawa "Control of System Calls from Outside of Virtual Machines" SAC'08 March 16-20, 2008, Fortaleza, Ceara, Brazil*
8. *M. Sharif et al., "Secure In-VM Monitoring Using Hardware Virtualization," Proc. 16th ACM Conf. Computer and Communications Security, ACM Press, 2009, pp. 477–487.*
9. *B. Payne et al., "Lares: An Architecture for Secure Active Monitoring Using Virtualization," Proc. IEEE Symp. Security and Privacy, IEEE Press, 2008, pp. 233–247.*