Intrusion Detection in Virtualized Environment

Nilambari Joshi[1,] Varshapriya J N[2]
[1]MTech II yr (NIMS), [2]Asst. Professor
[2,]Dept. of Computer Engineering and IT Veermata Jijabai Technological Institute, Mumbai

ABSTRACT

Server Virtualization is one of the concepts driving current enterprise infrastructure design and setup. It ensures business continuity and agility through on demand resource sharing and optimum resource utilization. At the same time it has raised concerns over new trends of security breaches and attacks on virtual machines. Traditional Intrusion Detection systems have their own limitations and they are less effective when used in virtual environment. Concept of Virtual Machine Introspection can be considered as an effective mechanism to do intrusion detection in virtual machine. This paper studies approaches of intrusion detection in virtual machines from higher privileged level than virtual machine operating system and presents a framework to detect network based as well as host based attacks on virtual machine.

KEYWORDS: *Hypervisor, Intrusion Detection, Security, Virtual Machine Introspection, Virtualization.*

I. INTRODUCTION

Server Virtualization deals with consolidation of server based applications running on different physical servers on a single host machine using virtual machine (VM). Virtual Machines can run different operating systems and consider that they own and have complete control over the hardware dedicated to them. In reality physical host hardware is shared by virtual machines running on the host and this resource allocation and sharing is monitored and managed by another layer called Virtual Machine Monitor (VMM) or hypervisor. Figure1 shows difference between traditional model and virtualized model of application deployment.

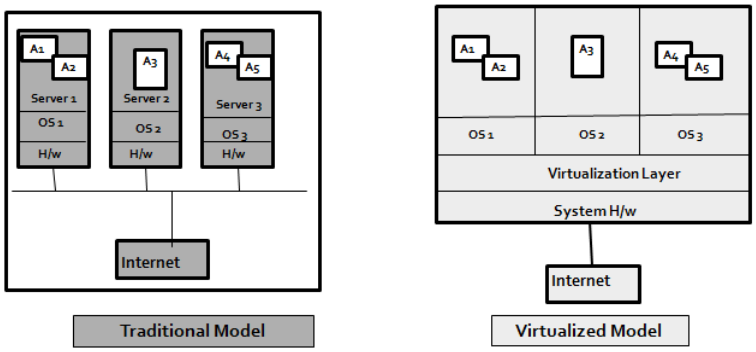

Figure.1 Server Virtualization Model vs. Traditional Model

1.1 Security Concerns in Virtualized Environment

1. As can be seen from Figure.1, Physical host hardware resources (CPU, clock, network port, memory, hard disk) are shared between virtual machines without their knowledge. Though Hypervisor is responsible to ensure isolation among virtual machines, certain business needs demand inter VM communication, achieved by means of clipboard sharing, sharing memory etc. This gives more attacking surface for an intruder to compromise a virtual machine from another virtual machine.

2. If the virtual machines are connected in bridged mode, they are easily accessible on internet for external clients. So all networking related attacks are applicable to virtual machines.

3. Furthermore kernel rootkits which are more hazardous since they acquire root privileges and directly attack Operating System, are also a big threat like any other host system

It is difficult to detect these attacks with traditional intrusion detection systems since virtual machine deals with virtual resources and thus information gathered by IDS deployed on such machines might be inconsistent.

II. MOTIVATION

Intrusion Detection System (IDS) mainly monitors network and system activities for any abnormal behavior or policy violations and produces reports.
Intrusion Detection Systems are mainly classified as

- Host Based IDS – Host Based IDS captures information from system log files, OS data structures, file system details and checks for any unintended, untimely changes in these objects.
- Network IDS – Network based IDS is mainly deployed on routers, gateways. It mainly monitors network traffic and checks for any malicious activity by analyzing packet data with reference to network protocols and identifies any misbehavior.

There are certain Pitfalls of traditional IDS in virtualized environment

1. Host based IDS mainly depends on information extracted from Operating System components and system log files. In virtualized environment virtual machine runs on emulated hardware resources. So information available with guest OS about the resources like memory addresses is not realistic.

2. Network based IDS if deployed on external router cannot monitor inter VM communication if virtual machines are connected in user mode (based on NAT). Also in case of VLAN there can be multiple software bridges / switches in same host connecting virtual machines, which all should be monitored to provide secure network communication.

3. Another shortcoming of HIDS is it has to be deployed on each system which needs to be protected. So in case of virtualized system, its an additional effort and consumption of resources to deploy HIDS on each VM

4. Virtual Machines should own resources sufficient to run the services deployed in them so it is not effective utilization of resources if they are being used by security system.

5. HIDS being part of system under attack, it itself is susceptible to get compromised by the attacker.

Therefore it is required to design a framework that can provide a secure mechanism to detect intrusion into virtual machines running on a host system.

III. VIRTUAL MACHINE INTROSPECTION

Virtual Machine Introspection is a way of getting Virtual Machine state information by monitoring it from a privilege level higher than that of guest Operating System. The concept was first introduced by Tal Garfinkel in 2004[1]. And can be used effectively to monitor virtual machine from outside and detect any intruding activity. A virtual machine state mainly involves CPU registers, volatile and non volatile memory and I/O data. Virtual Machine Introspection relies on the fact that Virtual Machine Monitor (VMM) runs at a privileged level higher than guest operating system and thus has realistic information about guest system and has more control over the guest. Therefore VMM APIs can be used to extract details about guest. The main complexity of this approach lies in the fact that, the data provided by VMM is raw data i.e. in the form of string of 0s and 1s (binary data) Its up to the VMI application to use knowledge about guest operating system and decode this data. This discrepancy is called semantic gap. Virtual Machine Introspection still provides advantage in terms of keeping monitoring system out of guest machine and thus out off insight of attacker. This method is passive control and thus data is pulled by the monitoring system rather than pushed by the monitored system (as in case of active monitoring).

IV. SOLUTION DESIGN

The work done to design and implement an Intrusion Detection System for virtual machines was based on following consideration.

- Reliability – The system should be able to do its designed activities (like data extraction, analysis) under normal workload of the host machine and VM.
- Attack-safe –If the virtual machine is compromised, it should not adversely affect the detection system.
- Not interfering – The system should not intervene with normal operations of the service providing VM.
- Provide security to all Virtual Machines running on the same physical host.
- Detect intrusion initiated from within vlan.

While designing the solution main assumption considered is Hypervisor as a trusted component base. Since virtualized environment is a new target for attackers, it is decided to go ahead with Anomaly based intrusion detection, in which normal system behaviour is first monitored and any deviation from such behaviour is detected and reported. Signature based intrusion detection which particularly checks for a known attack is not advisable due to newly emerging attacking mechanisms.In virtualized environment since guest OS runs in ring 3 or user application level it does not provide realistic hardware state information to the intrusion detection system. Therefore concept of virtual machine introspection as discussed in section III is used to get the state information of guest system from hypervisor. The approach of active monitoring, which includes placing hooks into guest OS [8] to capture events, or placing monitoring code in guest system [9] can put the system at risk and has additional task of changing OS of each guest. Considering the pros and cons of both the approaches, the solution proposed considers two types of interfaces with the ID system, the prototype named as VM-ID (Virtual Machine Intrusion Detection). Interfacing with the hypervisor to get external view of the system and interfacing with the guest to get internal view of the system.

V. PROPOSED FRAMEWORK AND IMPLEMENTATION VI.

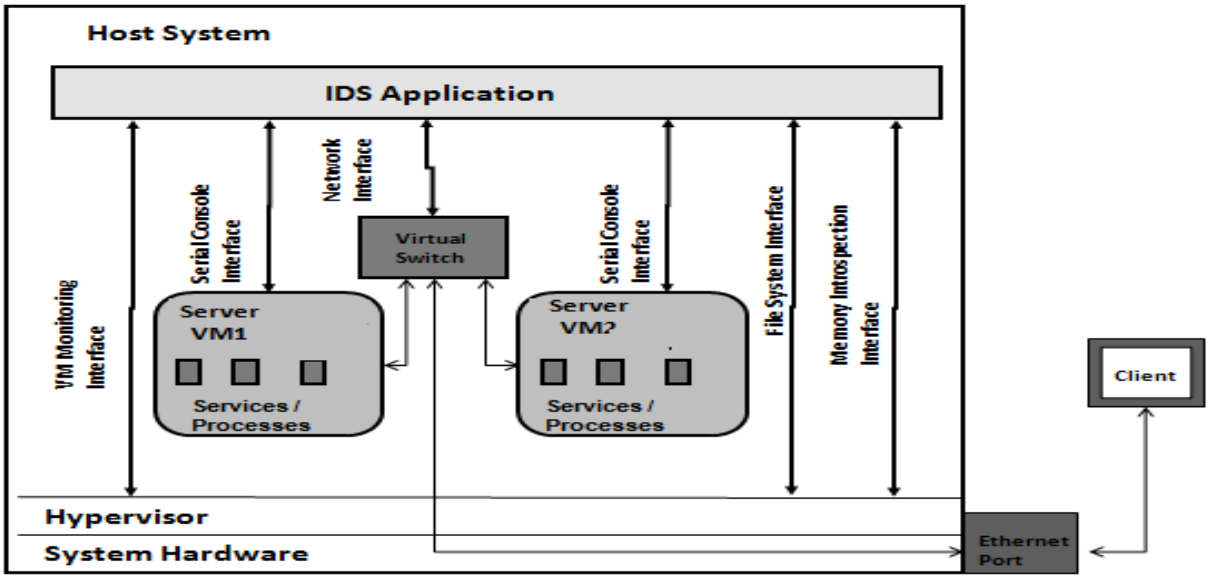

Figure.2 VM-ID system Proposed Framework

As shown in Figure.2 the system consists of following components.

1. Host system - It is high configuration physical machine which hosts different virtual machines running different services,
2. Hypervisor – The intermediate layer supporting server virtualization on the host system
3. Network port – Connecting virtual machines with the network.
4. IDS application (VM-ID) is user level process running on the host system.

The IDS application interfaces with hypervisor in the following ways -

- **Memory Introspection interface** – This interface captures real time memory data from hypervisor. With the help of knowledge about guest OS, it interprets the data and gathers information about important entities like processes, system calls, file system etc. The prototype used libvmi [11] [12] that is data extraction tool by memory introspection developed as extension of xen-access module. It facilitates to reduce semantic gap by configuring system for each Virtual Machine to be introspected.

- **File system mapping interface** – Memory introspection does not give information about static data available on storage disk. The same can be obtained by getting access to the guest file system from outside. Libvirt based guestmount[14] is used to map guest file system in the host in read-only mode using following command

guestmount -o allow_other -d domain_name -i --ro mount_point

- **Virtual Machine Management interface** – The system needs to interact with virtual machines and hence should have control mechanism to start / stop / pause / resume them. Libvirt based virsh [17] tool is used to provide this interface.

The interfaces with the guest are

- **Serial Console** – Serial console is used to extract information from guest system which cannot be easily deciphered from raw data available through introspection interface and which can be cross checked with the information available from introspection interface and can act as lie detector. Serial Console to Virtual Machines is identified using Virsh tool command

  virsh ttyconsole guest_name

5.1 VM-ID Stages

The solution is divided into three stages

- Configuration – To facilitate VM introspection, it is required to configure certain parameters of each virtual machine in the Intrusion Detection application.

- Monitoring – In this stage, original guest OS details are captured and saved for further cross checking.

- Detection – In this stage the guest OS details are re-captured and cross checked with the information collected during monitoring stage to detect any deviation. Detection can be initiated by administrator and it also runs periodically.

- Reporting – If any deviation from normal system behaviour is detected an alert is sent to the administrator and important information is logged for future reference.

5.2 VM-ID Modules

The Intrusion Detection System is designed to be based on anomaly based intrusion detection method involving following modules

1. **Process Validation** – This module is based on principle of lie detection. It gets process details by two ways. One using Memory Introspection APIs (using libvmi) and Second through serial console. It checks for any discrepancy between the two results and accordingly detects intrusion activities. It can be used to detect rootkits like adore-ng, itf, knark [13], which hide processes from administrator. The processes do not appear in serial console view. But they appear through memory introspection. The algorithm is as shown in Figure.3

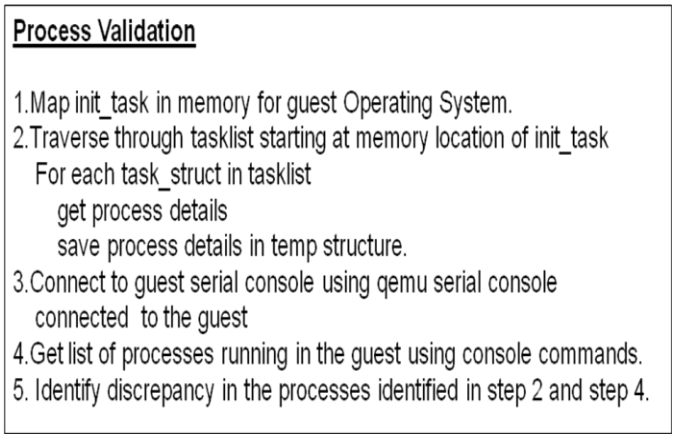

Figure.3 Process Validation

2. **System Call Validation** – This module checks for any stealthy attacks by modifying system call pointers and thus invoking attackers function before proceeding with normal system call behaviour. Using memory introspection it detects the change in system call function pointer. The Algorithm is as shown in Figure.4

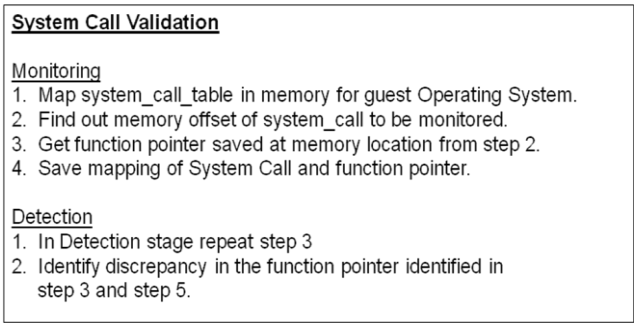

Figure.4 System Call Validation

3. **File system integrity checking** – This module is a two phase process. In the Monitoring phase, it captures SHA hash of important files, as defined in vmid configuration file. In Detection module, it recalculates the hash and checks with the previous one. If there is any change, it alerts the administrator about intrusion detection. Figure.5 gives steps for checking file integrity in guest.

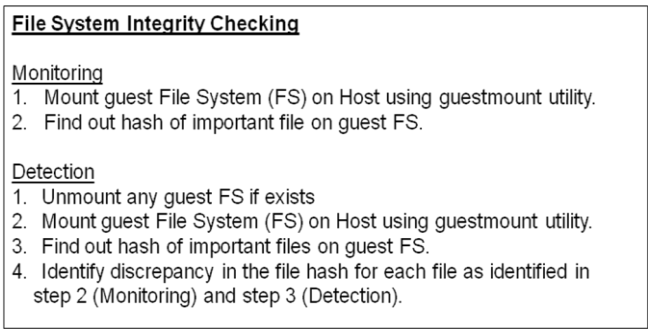

Figure.5 File System Integrity Checking

4. **Network traffic Validation** – When multiple virtual machines are deployed on same physical host as in case of server virtualization, an attacker residing on one VM can infect another VM on same host by manipulating its network traffic with the external world. ARP spoofing, SYN flooding are common threats to co-hosted virtual machines. Validating network traffic by monitoring bridge interface, can detect such kind of attacks. Figure.6 gives steps for ARP Spoofing attack from one VM on another in the same VLAN.

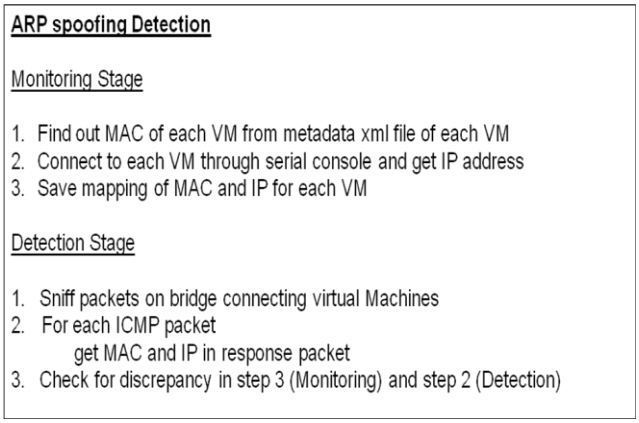

Figure.6 ARP Spoofing Detection

5. **User Interface and Notification** – User interface provides front end to the intrusion detection system and facilitates the administrator to check VM sanity. As a part of prototype it is developed in Java and integrated with other intrusion detection modules through Java Native Interface (JNI) [10]. The system keeps the details logged in log files which can be referred by administrator to decide further actions. Additionally a scheduler runs all validation modules on all configured virtual machines and provides Alert messages in case of any intrusion detected.

VII. EXPERIMENTAL RESULTS

As a part of prototype and testing, the setup considered is as shown in Figure.7 Host is ubuntu 12.04 system (4 GB RAM) with KVM, libvirt and qemu installed. Guest machines have ubuntu 12.04 as guest os with 512 MB RAM each. They are connected to each other and external network through KVM, bridgeutils bridge br0. Attacks are carried out by invoking LKM based rootkits run on VM from client.

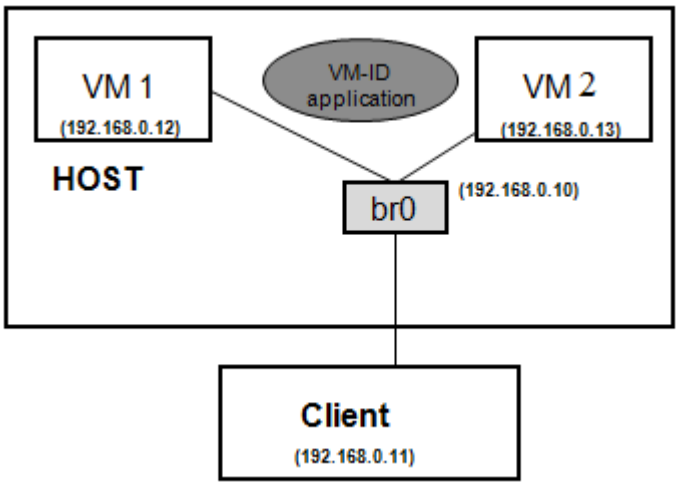

Figure.7 VM-ID Prototype setup

The system is able to detect intrusion through

- Process verification – tested with adore-ng based process hiding rootkit [15]
- System call interception – tested with Linux Kernel Module hijacking system call [16]
- ARP spoofing- tested by running arp poisoning tool arp-sk [18] from one VM targeting another VM on same VLAN.

VIII. RELATED WORK

Security of virtual machines is highly under consideration in academic as well as industry research circle. There are many research projects conducted to address security concerns of virtualized system. The pioneer project on memory introspection was done by Tal Garfinkel and group with Livewire prototype at Stanford University [1]. Kenichi Kourai developed a distributed Intrusion Detection System "Hyperspector" for LAN machines leveraging concept of virtualization [2]. A project by Marcose Laureano [3] traces processes from guest VM to identify suspicious activities. Lares [8] project by Bryan Pane (developer of libvmi) uses a different approach of active monitoring of virtual machines by placing hooks within VM guest OS. The advantage of active monitoring over memory introspection is that we need not bother about semantic gap. And it can give realtime information about activities going on in guest VM. On the other side, it has an overhead of protecting it from an intruder attacking the VM.

Compared to the approaches previously taken, the prototype discussed in this paper differs in following ways

1. It provides a single application to monitor multiple virtual machines.
2. It integrates functionalities of HIDS as well as NIDS.
3. The system is not integrated into the Hypervisor.
4. The system does not need any changes in the guest operating system

IX. CONCLUSION AND FUTURE SCOPE

Virtualization is very promising technology which is deployed to achieve compact and high efficiency data centers. Since it involves multi component stack, security measures have to be incorporated at each level.The current prototype proposed, addresses intrusion detection based on memory introspection, file integrity checking and network traffic sniffing. It leverages functionality of hypervisor to extract data related to virtual machine. It can be further enhanced to integrate with VM active monitoring so that it can provide intrusion prevention functionality as well. The Intrusion detection system can be enhanced further by deploying on different machines which are part of a cluster and provide distributed intrusion detection mechanism.

REFERENCES

[1] A Virtual Machine Introspection Based Architecture for Intrusion Detection", Tal Garfinkel (VMware), Mendel Rosenblum (Stanford) - Network and Distributed System Security Symposium, February 2003.

[2] HyperSpector: Virtual Distributed Monitoring Environments for Secure Intrusion Detection", Kenichi Kourai, Shigeru Chiba, 2005 ACM 1-59593-047-7/05/0006

[3] Intrusion Detection in Virtual Machine Environments", Marcos Laureano, Carlos Maziero, Edgard Jamhour - Proceeding EUROMICRO '04 Proceedings of the 30th EUROMICRO Conference.
[4] Understanding Full Virtualization, Paravirtualization, and Hardware Assist", VMWare white paper
[5] Virtualization -Wiki - http://en.wikipedia.org/wiki/Virtualization
[6] SECURITY CHALLENGES WITH VIRTUALIZATION", J Ramos, Masters thesis, University of Lisboa,2009
[7] HIMA: A Hypervisor-Based Integrity Measurement Agent ", Azab, A.M., Peng Ning ; Sezer, E.C. ; Xiaolan Zhang , Computer Security Applications Conference, 2009. ACSAC '09.
[8] Lares: An Architecture for Secure Active Monitoring Using Virtualization", Payne, B.D., Carbone, M. ; Sharif, M. ; Wenke Lee Security and Privacy, 2008. SP 2008. IEEE Symposium
[9] Process Implanting: A New Active Introspection Framework for Virtualization", Zhongshu Gu, Zhui Deng, Dongyan Xu, Xuxian Jiang, 2011 30th IEEE International Symposium on Reliable Distributed Systems
[10] http://docs.oracle.com/javase/6/docs/technotes/guides/jni/
[11] Simplifying Virtual Machine Introspection Using LibVMI", Bryan Pane, Sandia National Laboratory Report, September 2012.
[12] Libvmi: A Library for Bridging the Semantic Gap between Guest OS and VMM",, Haiquan Xiong et.al Computer and Information Technology (CIT), 2012 IEEE 12th International Conference
[13] Linux Kernel Rootkits" , http://la-samhna.de/library/rootkits/
[14] guestmount" ,http://libguestfs.org/guestmount.1.html
[15] http://commons.oreilly.com/wiki/index.php/Network_Security_Tools/Modifying_and_Hacking_Security_Tools/Fun_with_Linux_Kernel_Modules#hidepid.c
[16] http://sourceforge.net/projects/hijacklinuxsysc/
[17] Virsh", http://linux.die.net/man/1/virsh
[18] Arp swiss knife - http://sid.rstack.org/arp-sk