An Intelligent Virtual Machine Monitoring System Using KVM for Reliable And Secure Environment in Cloud

Mrs. Swarupa Mahesh Deshpande
Department of Computer Engineering
MIT College of Engineering, Pune
swarupa.kul@gmail.com

Prof. Mrs. Bharati Ainapure
Department of Computer Engineering
MIT College of Engineering, Pune
bharati.ainapure@mitcoe.edu.in

***Abstract*—The KVM (Kernel-based Virtual Machine Manager) hypervisor manages many VMs (Virtual Machine) on single host. The virtualization environment for cloud systems in IaaS (Infrastructure as a Service) layer supports heterogeneous operating systems in homogeneous environment. An Intelligent Virtualization Monitoring System Using KVM monitors the virtual machines under KVM hypervisor. The KVM is a type II hypervisor that is Linux kernel based virtual machine manager which comes with Linux OS. The VMs are the target for the malicious or abnormal attacks. To protect the VMs from the attack, the proposed system uses the VM monitoring script to get the status of the VMs. An intelligent virtualization monitoring system is incorporated as a part of KVM by sending the status of each VM running on hypervisor. The intelligent system categorizes attack patterns and actively rectify the attack. The necessary action taken will be communicated through the cloud API.**

Keywords— KVM , Intelligent Virtualization Monitoring System, Cloud Computing, Virtual Machine, Hypervisor, Virtualization.

I. INTRODUCTION

Now a days the cloud environment has emerged as the most promising computation platforms . Public users and System owners share resources in the cloud environment. Security threats may create serious damage to the cloud environment than individual PCs. Virtual Machines are running in the cloud. One successful attack on the host machine in the cloud may cause the damage of all Virtual machines. It is necessary to have the promising mechanisms to increase the cloud security .

The open-source solution to this problem is to adopt KVM as a VM manager to manage each Virtual Machine in the cloud environment. Virtualization is the basic role of KVM. It can virtualize the devices, like processor, storage , RAM, mother boards, network interface cards, system timers etc.
Virtual Machine managers must be able to protect host machines from the attacks that are likely to happen, and detect VM that has been compromised. It also must be able to provide a security service to guest VMs. The cloud computing environment may have attacks like : a) The hypervisor attacks: attackers may acquire the control of host privileges through software loopholes and vulnerabilities . b) attacks on guest Virtual Machines from another Virtual Machine: attackers can damage another virtual machine in the same cloud platform, and c) attacks on Virtual Machines from the virtual machines that are not part of the same cloud environment.

In supporting the Infrastructure as a Service(IaaS) model, the virtualization plays an important role. The hardware resources can be shared among multiple clients to achieve multitenancy. Inexpensive rental and cost free trials of cloud services may be the cause that inspires attackers to add their defective code on cloud environment. Due to cloud scalability and availability, the attacker can take advantage to attack on victim [1]. Such type of cloud can bring down the business services through denial of service attacks and flooding [1]. The secure environment to client is the responsibility of cloud provider. Generally, cloud provider monitors the infrastructure of cloud by introducing an inside agent for each VM. But, there is possibility that an agent itself can get compromised by an intruder. This kind of traditional in-guest security is not reliable as one VM that has been compromised can be used to compromise other VMs.

The new attack surface layer has been added due to virtualization for the attacker. It is a challenge to avoid attacks at the virtualization layer. The hypervisor manages VMs in the cloud. The hypervisor itself can be a central point of attack to get the control of whole system. The hypervisor vulnerability exploits like malware and rootkit can get into physical host through VMs. The virtualization attack like VM escape can disturb the isolation given by hypervisor and can get into physical host. The vulnerability like CVE-2008-0923 in the VMware hypervisor has proved that VM escape is possible [2]. CVE-2012-0217 in Xen 4.1.2 hypervisor is a similar kind of vulnerability.

The virtual machine introspection can be a good solution in order to monitor the malicious activities in VMs from outside. Virtual Machine Introspection tool monitors and analyzes the virtual machine status i.e. inside view of processor and general purpose registers, VM main memory, system calls, and hard disk contents of VM that is currently running . This VM status can be used for identification and avoidance of malicious program at Virtual machine Manager(VMM).

In this project, a new intelligent virtualization monitoring system (VMS) for cloud environment continuously checks status of guest VMs static and dynamic both to identify and prevent the cloud environment from serious vulnerable attacks. VMS is, dynamic and real-time mechanism . The aim is to achieve reasonable efficiency and security.

The Intelligent Virtual machine security manager on cloud System mainly aimed at providing the security to the virtual machines on the cloud. The main objectives of the system are:

- To protect the virtual machines from malicious attacks.
- To add intelligence in virtualization introspection system through pattern recognition algorithm.
- To improve the efficiency of hypervisor and virtual machines on cloud.

Section II focuses on the related work done in this area. In section III the system architecture has been explored. Section IV deals with the proposed work and the modules involved in the system. In section V the attack model has been covered. Section VI deals with the algorithms. In section VII , the system performance is evaluated.

II. RELATED WORK

The paper "Survey of the Virtual Machine Introspection in Cloud Computing". Virtualization gives the capacity to have numerous visitor working frameworks on a solitary physical host framework in paravirtualized or completely virtualized modes, and speaks to a characteristic development for the prerequisite of multi-tenure of distributed computing. Virtualization lets clients have "one machine, numerous working frameworks, various applications" and switch between them voluntarily[2].

The paper "Evasion Resistant Intrusion Detection Framework at Hypervisor Layer in Cloud" proposed a security mechanism using Virtual Machine Introspection technique, that detects the suspicious VM tasks from VM from outside. It works at VMM layer and checks hardware status of VM to analyze working activities of the VM[3].

In "Securing KVM-based Cloud Systems via Virtualization Introspection" paper the author proposed that the running VM utilizes the resource that is virtualized and one of the VMs on the host and it is a procedure executed by KVM, and this utilize another physical host as a sandbox host . Thus they have chosen strace and qemu monitor to gather the runtime information from VM and analyze the VM's behavior[4].

- *VMI (Virtual Machine Introspection) Mechanism*:

The proposed system uses Nitro, a system call tracing tool that monitors the status of the virtual machines from outside. It actually traces and works on all the three kinds of system call tracing mechanisms like Interrupt based , syscall based and sysenter based system calls[5].

Each system call tracing mechanism is explained in brief below:

A. Interrupt-System Calls:

The copy of guest IDT is virtualized into the hypervisor. The manipulations to the IDTR is done to monitor and trap the write requests to it.

B. SYSCALL- System Calls :

The SYSCALL instruction and SYSRET are the instructions used to handle system calls. These instructions are model specific. A system can be interrupted by unsetting the SCE (System Call Extension) flag and setting the VMM to trap exception called invalid opcode exceptions. Nitro must be capable of handling exceptions generated by the SYSRET and SYSCALL instruction as well as emulating these instruction. Nitro must also be able to collect the value returned by the system call required by the application.

C. SYSENTER- System Calls:

The SYSENTER and SYSEXIT instructions depends on a set of model specific instructions. After calling SYSENTER ,the values in each of the Model Specific Registers are copied to specific system registers. The value of the SYSENTER Code Segment MSR is copied into the CS register during the execution of SYSENTER and the CS register contents are made null. The general protection fault will be generated as a result. Thereby causing a system interrupt which requires to save the current value of the SYSENTER CS Model Specific Register in the VMM and load it with a null. This can be trapped by the hypervisor.

III. SYSTEM ARCHITECTURE

Problem Statement:

To design Intelligent Virtualization monitoring System mainly aimed at providing the security to the virtual machines on the cloud. The main objectives of the system are:

• To protect the virtual machines from malicious attacks.

• To add intelligence in virtualization introspection system through pattern recognition algorithm.

• To improve the efficiency of hypervisor and virtual machines on cloud.

The framework architecture demonstrates the virtual machines running in the cloud environment. The hypervisor (KVM), the virtual machine monitor, provides every one of the resources of the host machine to all the VMs through virtualization. The fig. 1 shows the overall system architecture. The proposed framework has following parts:

A. Virtual Machines

Virtual Machines are the set of guest operating systems running in cloud. The hypervisor must be able to keep host machines away from vulnerable attacks, and be able to provide a security mechanism to VM users, by identifying whether the Virtual Machine are attacked or not. The attacks on the cloud computing environment may be.

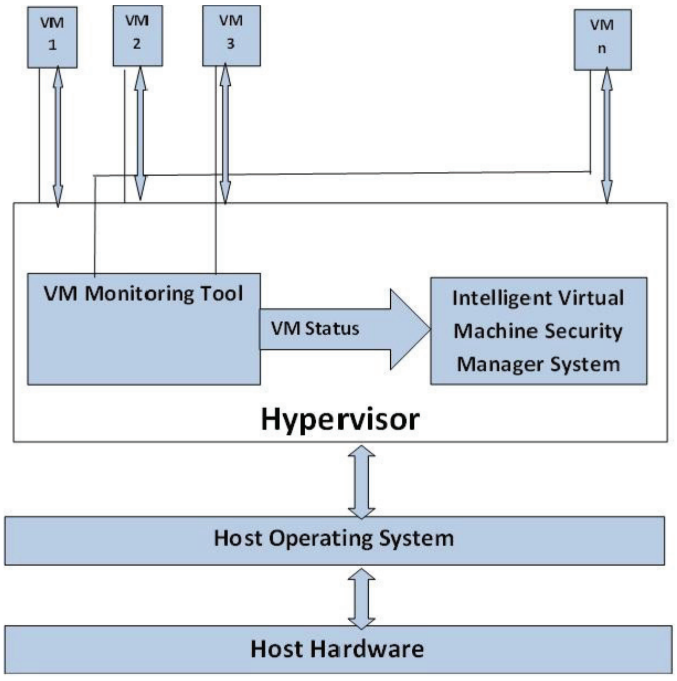


Figure 1: *System Architecture*

- *Hypervisor attacks:* attackers may acquire the control of host privileges through software loopholes and vulnerabilities.
- *Guest Virtual Machine attacks from another Virtual Machine:* attackers can damage another virtual machine in the same cloud infrastructure.
- *Guest Virtual Machine attacks from the virtual machines from outside of the cloud infrastructure.*

B. Hypervisor

The hypervisor is a virtual machine manager. The proposed framework has following parts at hypervisor layer.

- *VM monitoring open source tool (Nitro):* Monitors continuously the status of the VMs from the hypervisor layer.
- *Virtual Machine Security Manager System:* Reads the status from VM monitoring tool and classify the attacks by using the Pattern recognition system and behavior checking and analysis module.
- *Host Operating System:* The host OS is the underlying physical OS on which the VMM is running.
- *Host Hardware:* The host hardware are the physical resources attached to particular machine on cloud infrastructure.

IV. PROPOSED WORK AND MODULES

The various modules in the proposed system are shown in the fig. 2. Monitoring System using KVM for Reliable and Secure Environment in Cloud

Detailed virtual Machine Security manager system is shown in figure 2. It has following components :

1) Guest VM 2) Monitor 3) Cloud API 4) Databases

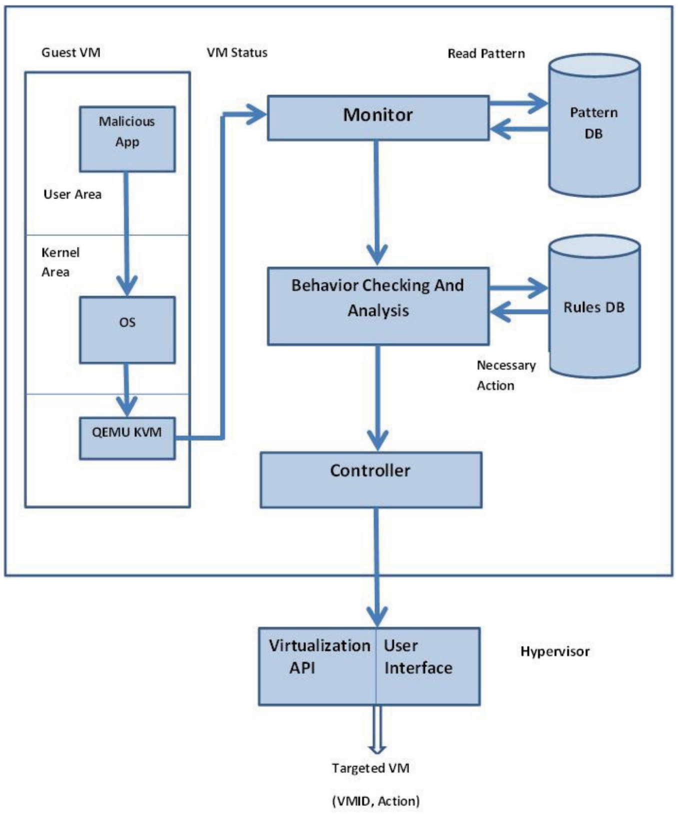


Figure 2: *Detailed Components of Intelligent Virtualization*

1) Guest VM: OS hosts various applications. The VM status will be sent to Monitor module through QEMU KVM. The user application which has been attacked is hosted in user area. Hypervisor QEMU KVM sends VM status periodically to monitor.

*2) Monitor:*Monitor analyzes the behavior of the VM. It checks attack pattern for the current VM in pattern DB. If the pattern matches from the pattern DB it returns corrective action data for identified attack. If the pattern not matches it sends VM id for behavior checking and analysis for rules creating from rules DB .controller holds data for VM to interact with the cloud middleware.

3) Cloud API: Cloud API contains virtualization API and user interface. It returns the VM id with corrective actions.

4) Database :Database stores data for different malicious behaviors

A) Pattern database

Stores data for checking malicious behaviors if the attacked VM works or behaves according to the pattern that is stored in database it returns the matching pattern with appropriate action for the identified attack. If match not found in pattern database it stores pattern as new pattern and VM id sent to rule creation.

B) Rules Database

Rules database stores data for VM for different prototype that built on the behavior of the attacked VM .the prototype and rules are created from attack type, behavior of the VM, nature and severity of the attack.

V. ATTACK MODEL

The main objectives of a VMM is to guarantees the isolation of facilitated VMs. The attacks on guest VM break the isolation of VMs[6][7]. The attacks may be of different categories. The section focuses on the different categories of attacks and their side effects.

i)Denial of service attacks:

A VM utilizes all the resources of the underlying host, causing other VMs from running smoothly and effectively.

1. Zombie attack

An attacker through network sends the unnecessary packets over channel of the targeted VM thereby causing the congestion in the network by sending demands from legitimate hosts in the system. These sorts of VMs are called zombies. In the Cloud, the solicitations for VMs are accessible by every client over network. An aggressor can send huge number of solicitations via zombies causing interruption in the cloud server services.

2. HX-DOS attack

HTTP and XML messages are combined together and are sent intentionally to increase traffic and makes overwhelming channel communication over the cloud.

Semantic standard based methodology is used to identify the malfunctioning in cloud API layer. A deterministic finite state machines are used to deal with malicious characteristics.

ii) *Insecure Interfaces and APIs*

1. Guarantee solid validation of authentication and implementing access control.
2. Investigation of the API interfaces security.
3. Implementation of two level access control mechanism at the API level using the Role Based Access Control Model.

iii) Shared Technology attacks

1. VM Hopping

By acquiring the control of victim VM through the intruder VM that is located on the same host where the victim VM is found. In this kind of attacks the intruder can control, monitor, and manipulate flow of data. The attacker can alter the configuration file and it can disturb the communication.

2. VM Escape

The is hypervisor can be itself attacked so that the attacker can get into host directly. An attacker runs malicious code on a VM that causes an Operating System running within it to break out and communicate directly with the VMM thus causing the attacker to get into the host Operating System and other VMs running on the same host.

3. Rootkit in VMM

Virtual device-based rootkit can instantiate a malicious VMM under the authentic one, which takes control over the resources and network channels, allowing attacker VM to introduce an unauthorized code into the framework.

4. Cross-VM side-channel attacks:

Side channel attacks are introduced by creating VM on the same physical host where a targeted VM resides. The vulnerabilities are exploited so as to extract or inferring sensitive information, memory, shared CPU or by monitoring traffic and control signals.

VI. ALGORITHM

Algorithm 1 (Main):

```
1. Start
2. Input: VM={$VM_1$, $VM_2$,$VM_3$, …… , $VM_n$} No. of
   Virtual Machines in cloud.
3. For i=1 to n
   i.    Monitor VM status($VM_i$)
   ii.   Read memory map, CPU status, IO devices
         status
   iii.  Collect data of $VM_i$ during events like VM
         idle state, attack launched by hacker, hacker
         sending keystroke, a hacker taking a
         screenshot
   iv.   Send the VM event behavior and VM ID to
         Virtual Machine Security manager System
4. Go to step 3 for reading next vm status
5. End
```

Algorithm 2: (Virtualization Monitoring System)

```
1. Start
2. Input : {VM event behavior, $VM_i$ ID }
3. Read the event behavior of $VM_i$
4. Virtualization Introspection System(VIS) Identify the
   behavior to classify it according to the threats that are
   to be handled.
5. Sends the category of threat detected by Virtualization
   Introspection System(VIS) and $VM_i$ ID to pattern
   recognition system.
6. End
```

Algorithm 3: (pattern recognition system)

```
1. Start
2. Input: { $VM_i$ ID, threat classified, $VM_i$ event memory
   map}
3. Reads the threat classified for particular $VM_i$ to match
   the rule in pattern database.
```

4. If the pattern is matched apply the corresponding rule to the VM_i .
    - Identify the appropriate action on targeted VM_i
5. Else
    - Define the new pattern for the VM_i
    - Update the pattern database
    - Identify the appropriate action on targeted VM_i
6. Decide to migrate or to terminate or to pause the VM_i
7. Output : {targeted VM_i ID, action: Migrate, Pause, Terminate }
8. End

VII. MATHEMATICAL MODEL

Input :

VM={ VM_1, VM_2,VM_3, …… , VM_n } No. of Virtual Machines in cloud.

Processing:

VMI tool={vm status: System call tracing, Interrupts, Memory map}

Virtualization Introspection Algorithm={VM status categorization}

Pattern Recognition Algorithm = {Patterns of VM attack: Existing, Nonexisting}

Existing= {Read from Pattern DB}

Non Existing= {Update the Pattern DB}

Output:

VMTargeted={ VM_1, VM_2,VM_3, …… , VM_n } No. of Targeted VMs in Cloud

Action = {Migrate, Pause, Terminate} Action to be taken on targeted VMs.

The figure 3 shows the state transition diagram with respect to the above mathematical model. State 1 is the start state in which the request for the VMI tool will be generated by the VM in the set(VM_1, VM_2,VM_3, …… , VM_n) to the hypervisor. If the hypervisor grants the request then in state 3 the VM status will be read, otherwise the application will be terminated. It is an unsuccessful state where the VM do not have the required resources. In state 3 the Nitro VMI will trace the system calls generated by the VM. Depending on the nature of system call (Interrupt, syscall, sysenter), in states 5,6 & 7 respectively status will be generated.

In state 8 the behavior checker will analyze the behavior. If it is normal then VM will be again submitted to hypervisor to regenerate status. If the behavior is malicious then appropriate action will be taken to rectify the malicious behavior. It is an successful state.

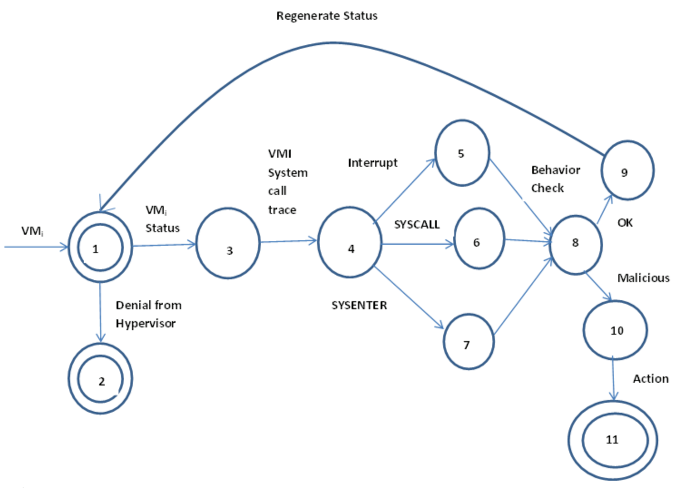


Figure 3: *State Transition Diagram*

VIII. PERFORMANCE AND EVALUATIONS

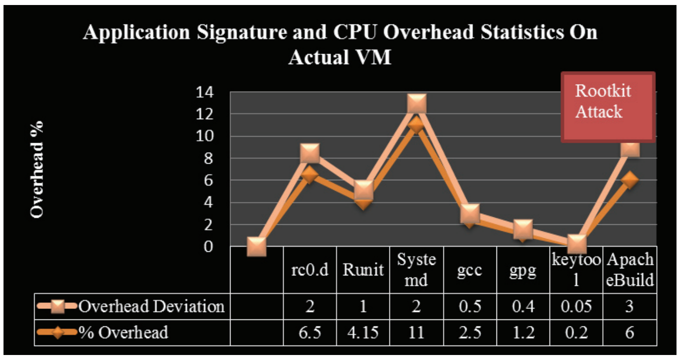
In this section, we have tested the performance for the host OSs like: Ubuntu 14.04 64-bit, and Windows 7 as guest OS etc. The processor used is Intel Core 2 Duo processor at 2.4 GHz with 4 GB of RAM. We used Ubuntu 14.04 host system for the tests. We have used QEMU KVM as the hypervisor.

The test results are evaluated on an Ubuntu 14.04 (64-bit) host OS. The guest Operating system used is Windows 7. The interrupt-based and SYSENTER-based system call instructions are used for testing purpose.

TABLE 1

Binary Name	Binary Type	Application Signature on Normal VM	CPU Overhead Mean	CPU Overhead Deviation	Result	Remark
		Fingerprint Values	% Overhead	Overhead Deviation		
rc0.d	Kernel Daemon	bb9bd2c36d165dc7dae6144ff50de08569fae680	6.5	2	1	Infected
Runit	Kernel Init	49ac54f61fa52ece576c20dba80a8227c37e1b12	4.15	1	0	OK
Systemd	System Daemon	a44a53c54ba77f8a2ae69713f422d9f138899fdb	11	2	0	OK
Gcc	Compiler	fce79b7fe1fee3a977fa1bd4efbd9e9a06c29c14	2.5	0.5	0	OK
Gpg	Signing Tool	9db65548458ceb3702db8e78dfac04a651a6e503	1.2	0.4	0	OK
Keytool	Signing Tool	9868be88cfedf9e7063304fa7fc909879f0d37f9	0.2	0.05	0	OK
ApacheBuild	Server	f89e5be7df3eb771c9bff8bae021893a530fb11a	6	3	1	Infected

Table 1*: CPU Overhead Deviation*



		rc0.d	Runit	Systemd	gcc	gpg	keytool	ApacheBuild
Overhead Deviation		2	1	2	0.5	0.4	0.05	3
% Overhead		6.5	4.15	11	2.5	1.2	0.2	6

Figure 4: ***Application Signature and CPU Overhead statistics on Actual VM***

The Table 1(CPU Overhead Deviation) shows system level programs and some user applications that are running on VM. Depending on the finger print values of the executables, the CPU percentage overhead is calculated and the CPU overhead deviation has been observed. Depending on this it has been decided that whether the VM is infected or not.

The graph in Fig. 4 shows the impact of rootkit attack on the performance of the VM. The performance is measured on the basis of the parameters like %overhead and overhead deviation.

CONCLUSION

We propose *An Intelligent Virtual Machine Monitoring System Using KVM for Reliable And Secure Environment in Cloud* for IaaS cloud platforms. This system monitors the static and dynamic VM status using Nitro VM monitoring tool, that analyses the memory dumps by applying intelligence by classifying different attacks to identify which VMs are attacking the VMM, and which VMs are attacking others. This System also detects compromised VMs by applying the intelligence to the system. The system provides protection based on established rules accordingly system will terminate or migrate or pause the targeted VMs.

REFERENCES

[1] *Conference on Advances in Communication, Network, and Computing*. 2014.

[2] Nance, Kara, Matt Bishop, and Brian Hay. "Virtual machine introspection: Observation or interference?." *IEEE Security & Privacy* 5 (2008): 32-37.

[3] Borisaniya, Bhavesh, and Dhiren Patel. "Evasion Resistant Intrusion Detection Framework at Hypervisor Layer in Cloud." *International Conference on*. IEEE, 2014

[4] Lee, Sheng-Wei, and Fang Yu. "Securing KVM-Based Cloud Systems via Virtualization Introspection." *System Sciences (HICSS), 2014 47th Hawaii International Conference on*. IEEE, 2014.

[5] Pfoh, Jonas, Christian Schneider, and Claudia Eckert. "Nitro: Hardware-based system call tracing for virtual machines." *Advances in Information and Computer Security*. Springer Berlin Heidelberg, 2011. 96-112.

[6] Wang, Yulong, and Xianqiang Yang. "Survey of the Virtual Machine Introspection in Cloud Computing." *Advances in Information Sciences and Service Sciences* 5.10 (2013): 295.

[7] Mahajan, Harshal, and Nupur Giri. "Threats to cloud computing security." *VESIT, International Technological Conference-2014 (I-TechCON)*. 2014.

[8] Bahram, Sina, Xuxian Jiang, Zhi Wang, Mike Grace, Jinku Li, Deepa Srinivasan, Junghwan Rhee, and Dongyan Xu. "Dksm: Subverting virtual machine introspection for fun and profit." In *Reliable Distributed Systems, 2010 29th IEEE Symposium on*, pp. 82-91. IEEE, 2010.

[9] More, Asit, and Shashikala Tapaswi. "Virtual machine introspection: towards bridging the semantic gap." *Journal of Cloud Computing* 3.1 (2014): 1-14.

[10] Gonzales, Daniel, Jeremy Kaplan, Evan Saltzman, Zev Winkelman, and Dulani Woods. "Cloud-Trust-a Security Assessment Model for Infrastructure as a Service (IaaS) Clouds."

[11] Payne, Bryan D., Martim Carbone, Monirul Sharif, and Wenke Lee. "Lares: An architecture for secure active monitoring using virtualization." In *Security and Privacy, 2008. SP 2008. IEEE Symposium on*, pp. 233-247. IEEE, 2008.

[12] Dinaburg, Artem, Paul Royal, Monirul Sharif, and Wenke Lee. "Ether: malware analysis via hardware virtualization extensions." In *Proceedings of the 15th ACM conference on Computer and communications security*, pp. 51-62.