A Novel VENOM Attack Identification Mechanism in Cloud Virtualization Environment

Cheick Abdoul-Kader [1], Shih-Hao Chang [2*]
[1, 2]Department of Computer Science and Information Engineering, Tamkang University
New Taipei City, Taiwan 25137
[1]kader.oued@hotmail.fr、[2]shhchang@mail.tku.edu.tw

ABSTRACT

This paper investigates the security issue of virtualization in the cloud computing. We focus on how to identify the VENOM attack in the cloud-computing environment, and how to protect the hypervisor from this VENOM attack. Firstly, we have implemented VENOM vulnerability in the environment of QEMU/KVM and tried to identify its behaviors (action) in the cloud. Secondly, we also tried to protect the hypervisor, which is the most vulnerability part for virtualization environment. The proposed mechanism provides identification of the VENOM attack and lock the FDC port (0x3f5), which is responsible to send I/O command to the hypervisor.

Keywords: VENOM, QEMU, Virtualization, I/O command, Malware Attack

1. Introduction

Cloud computing is the delivery of on-demand computing resources – servers, storage, databases, networking, software, analytics and more over the Internet. It was inspired by the cloud symbol, which represents the general term for anything that involves delivering hosted services over the Internet. Cloud Computing exploits many existing technologies such as web services, web browsers, and virtualization, which contributes to the evolution of cloud environments [3]. Virtualization is technology that allows you to create multiple simulated environments or dedicated resources from a single, physical hardware system. Software called a hypervisor connects directly to that hardware and allows you to split one system into separate, distinct, and secure environments known as virtual machines (VMs). These VMs rely on the hypervisor's ability to separate the machine's resources from the hardware and distribute them appropriately.

Number of open sources virtualization tools or hypervisors available today, such as Xen,

* Corresponding author.

KVM [9], QEMU [4], have announced in the market. It's typically incorporates infrastructure as a service (IaaS), platform as a service (PaaS), and software as a service (SaaS). The cloud computing has five essential characteristics:

- *On-demand self-service*: A consumer armed with an appropriate delegation of rights (permission) can unilaterally provision computing capabilities, such as server time and network storage. As its automatically feature that do not require human interaction with each service's provider.
- *Broad network access*: Capabilities are available over the network and accessed through standard mechanisms that promote use by heterogeneous thin or thick client platforms (e.g., mobile phones, laptops, PDAs...).
- *Resource pooling*: The provider's computing resources are pooled to serve multiple consumers using a multi-tenant model, with different physical and virtual resources dynamically assigned and reassigned according to consumer demand. There is a sense of location independence in that the customer generally has no control or knowledge over the exact location of the provided resources but may be able to specify location at a higher level of abstraction (e.g., country, state, or datacenter). Examples of separately allocable resources include storage, processing, memory, network bandwidth, and virtual machines.
- *Rapid elasticity*: Capabilities can be rapidly and elastically provisioned, in some cases automatically, to scale out quickly and then rapidly released to scale in quickly. To the consumer, the capabilities available for provisioning often appear to be unlimited and can be purchased in any quantity at any time.
- *Measured service*: Cloud systems automatically control and optimize resource use by leveraging a metering capability at some level of abstraction appropriate to the type of service (e.g., storage, processing, bandwidth, and active user accounts). Resource usage can be monitored, controlled, and reported, providing transparency for both the provider and consumer of the utilized service.

VMware is a commercial leader but it is also based on open source. The only inconvenient is located in security of Cloud computing (Security of Data) because you share the same server in the cloud, thus it is possible that your data be shared somewhere. The virtualization network is the most critical threat in Cloud Computing and in particular hypervisor it the most vulnerability part for virtual network. The hypervisor or called virtual machine manager (VMM) is the main component of virtualization, it divided the operating system (OS) from the hardware by taking the responsibility of allowing each running OS time with the underlying hardware. It acts as a traffic cop to allow time to use the CPU, memory, GPU, and other hardware. Thus, the

security of hypervisors is very crucial as the whole system could be compromised even one vulnerability is exploited. It's imperative to ensure that this component is as secure as possible due to any defect in hypervisor can be used by attacker to do what they want in the server. The hypervisor sits between the guest and host, operating as the bridge (as show in the Figure 1), and connect both sides to communicate each other; any defects in hypervisor security may impact both the host and guest machine. Hypervisor is basically a software program, so it has all the traditional software bugs and the security vulnerabilities as any software have.

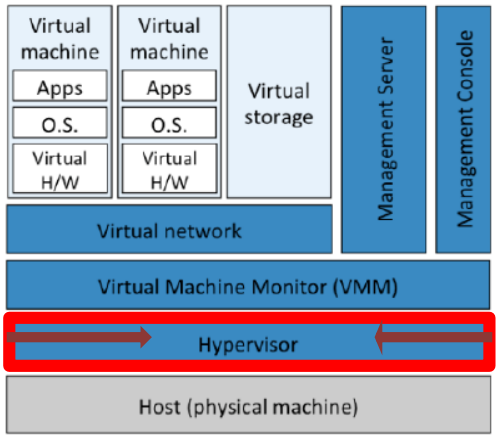

Figure1: Components of Virtualization

2. Literature Review

There have been a few works which discuss about the security issue of hypervisor in the cloud computing. Tsafrir et al. in [14] called theft-of-service attack as "cheat attack" where an ordinary process hijacks some desirable percentage of CPU without the notice of an administrator. The attacker, during theft-of-service attack targets the hypervisor of VM, which can be categorized, further, as concealment action.

In [5] rootkits modify critical system code and/or data structures in OS behavior for this reason; A possible rootkit detection method is to check if all critical components of an operating system are in their expected state. In [12][15] authors design a plugin specifically to malware detection and analysis by use information extracted from window swap file in the windows kernel memory pools.

There are several case studies such as [6] that provided the comparative analysis of KVM with respect to other hypervisors in term of different vulnerabilities. The main goal of this thesis is to build a novel mechanism to identify VENOM attack in the cloud computing, but also to

provide a solution to protect the hypervisor for such kind of attack in the cloud. However, the difference between the thesis and related work [6] is the source code of VENOM Vulnerability used by an attacker from the guest machine. Moreover, this thesis presents a new problem in the "FDC" which are outdated for most of virtualization products, but can be exploitable by an attacker to run a malicious code.

In 2011, Jason Geffner, CrowdStrike [8] Senior Security Researcher, discovered the vulnerability while performing a security review of virtual machine hypervisors and this vulnerability namely is VENOM. VENOM Vulnerability is the acronym for "Virtual Environment Neglected Operations Manipulation". Also named "CVE-2015-3456", exploits a flaw in the virtual disk driver code in virtual machines to allow an attacker to break out of the guest operating system and interact with the host. The interesting fact about VENOM is that it applies to a wide range of virtualization platforms (using the default configurations) and it allows for arbitrary code execution. The vulnerability specifically affects open source hypervisor called Quick Emulator (QEMU). The bug is in QEMU' s virtual Floppy Disk Controller (FDC), which is used in a number of common virtualization products (XEN, KVM, and Virtual Box...).

As the fact that the vulnerability exists in the hypervisor's codebase, it affects all host and guest OS. With this performance, which can make possible an attacker to break out of protected guest environments and take full control of the operating system hosting them. As VENOM attack exploits virtual disk driver codes and flaws personal data in the virtual machine, it seen like one of the most of the dangerous threats in cloud computing especially for VM.

3. VENOM ATTACK BEHAVIORS

This paper proposed an algorithm to identify VENOM vulnerability in the cloud. This research also suggests a method to protect the hypervisor for such kind of attack. Our experiment will be conducted with 2 different VMs in the cloud, with limited processors cores and memories:

- The host machine, worked as consists of Ubuntu 14.4 with kernel 4.1.0 and QEMU emulation software on Inter processor with limited RAM
- The guest machines consist of one virtual CPU and 1GB of RAM.

VM1 is a standard user, VM2 is an attacker that uses machine to launch a malicious program (VENOM attack), through FDC I/O port to affects all host and guest operating systems, as show in Figure 2.

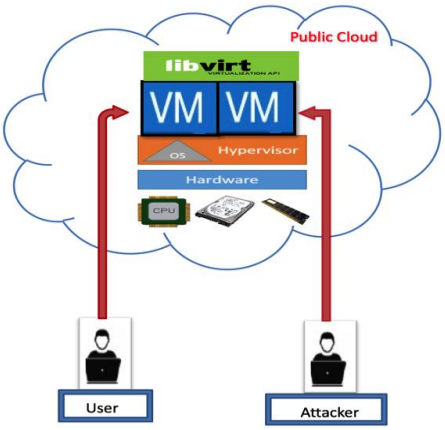

Figure2: Vulnerable Cloud Infrastructure

For our experiences, we also utilize Libvirt [7] due to its a toolkit to manage virtualization guest operating systems running on a host. The level of VENOM's vulnerability depends to the attacker. VENOM attacks access to device I/O ports from GM to get inside the Hypervisor as show in Figure 3.

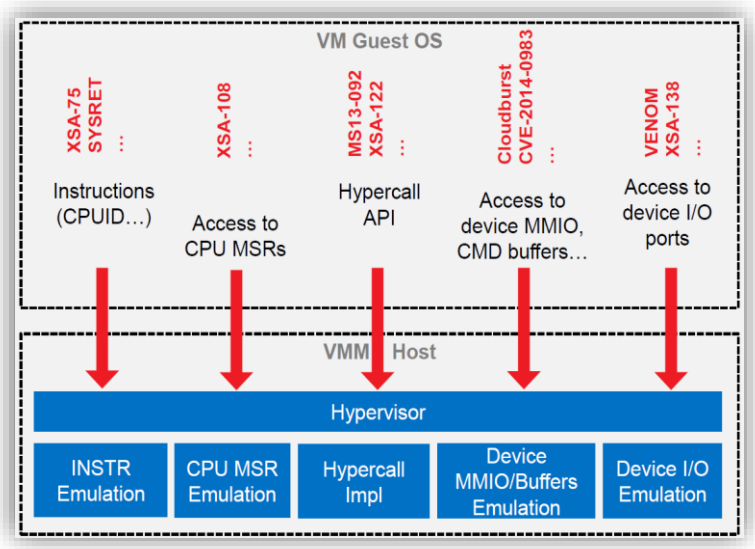

Figure 3: Attacking Hypervisor Emulation of Hardware Devices

The problem resides in FDC (Floppy-Disk-Controller). The guest machine communicates with the FDC (I/O ports) by sending commands such as (Read, Write, format...) to the device input/output emulation in the hypervisor. After every command, the FDC set each variable to 0 for next command. An attacker can send these commands to the FDC to overflow the data buffer

and execute arbitrary code in hypervisor process. Successful exploitation of the VENOM vulnerability can expose access to corporate intellectual property, sensitive and personally identifiable information which can impact organizations and end users that rely on affected VMs for the allocation of shared computing resources, connectivity, storage, security and privacy.

Therefore, this paper proposed an algorithm, which can identify VENOM vulnerability and protect the hypervisor for such kind of attack. The proposed algorithm will base on two steps:

1. The mechanism to identify VENOM vulnerability: in this step, we launched VENOM vulnerability from one of a guest machines. The source code of VENOM, which is provided by Jason Geffner CrowdStrike [8], we analyzed the behavior of the host machine through the Libvirt API [7] in KVM environment.
2. The mechanism to protect the hypervisor: in this step, we built a program to protect the hypervisor for such kind of attack. We programed a code to lock the FDC controller port in the hypervisor. The main purpose of this code is to lock the 0x3F5, which is mapped to Read/Write the I/O command from the guest machine.

4. Evaluation

In this section, we present our VENOM identification study of the proposed framework. According to our literature review, the attackers can trigger the VENOM vulnerability by sending commands and specially crafted parameter data from the guest system to the vulnerable Floppy Disk Controller to cause the data buffer overflow and execute arbitrary code in the context of the host's hypervisor process. The flaw is very dangerous because attackers could exploit it against a wide array of virtual machines, it is trigger-able on default configurations, and would allow the arbitrary code execution. We consider VENOM different from other vulnerabilities in the past that effect virtualized environments, since it exists in the hypervisor's codebase it is independent from the specific host operating system (Linux, Windows, Mac OS, etc.).

Therefore, we investigated KVM-Hypervisor with LibVirt API to monitor the VM. User1 (U1) and User2 (VENOM) applied both Ubuntu 14 based on KVM hypervisor. As mentioned that there is non-application has been launched on these VMs, that is the main reason why the CPU of the hypervisor it 0%, and the memory is 2048MB of 7860MB (that mean each VM have respectively the same memory 1024MB. With LibVirt toolkit managed all of my VMs (VENOM and U1) running in my host machine. The figures 4 and figure 5 below show how

LibVirt work and how to identify VENOM vulnerability.

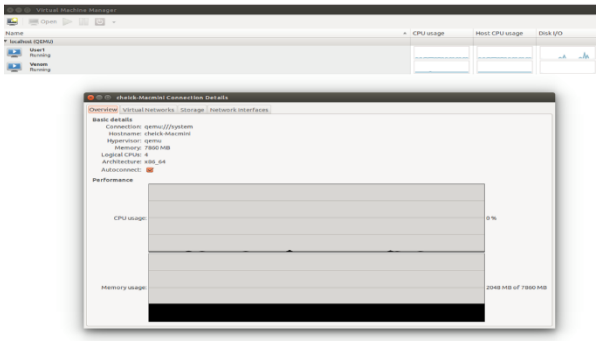

Figure 4: Statistic of KVM hypervisor

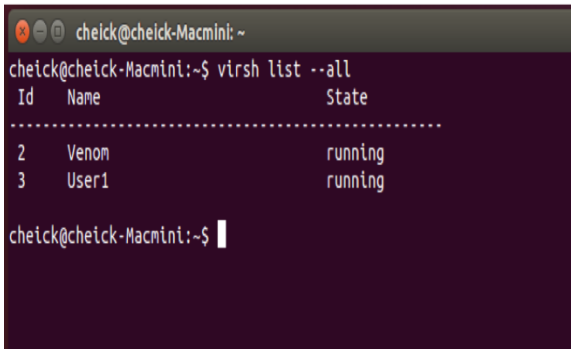

Figure 5: Identify VENOM vulnerability

To prove that the attack was successful in a system with a default set up, VENOM was used to illustrate and simulate a real life attack. Figure 6 displayed the commands of the virtual machines that are run on the hypervisor (VENOM and user1).

Figure 6: General Information of VMs

Figure 7 and Figure 8 show before the attack both from the host using Ubuntu Linux OS.

1. virsh list –all command: provide the states and name of each VM in KVM.
2. virsh dominfo “name of VM” command: provide the general Information of Guest VM. (Ex. ID, name, OS type, memory etc.…)
3. virt-top command: As shown in figure 7, this command can display the CPU static and memory usage of detail of each VM in the Hypervisor.

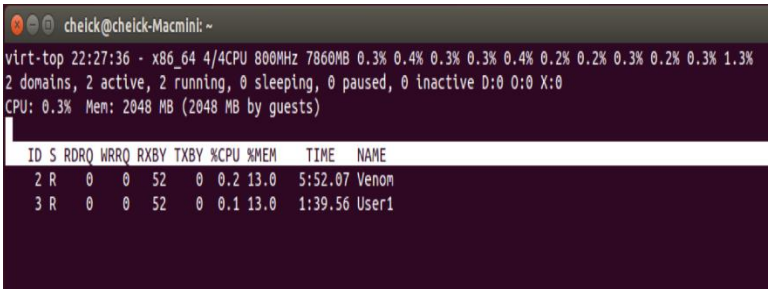

Figure 7: Static of CPU and Memory of VMs

4. free command: this command will display the virtual memory size in the host machine. As shown in Figure 8, it displays the percentage of CPU and memory for VENOM and User 1 are equal, because both of them have not do any operation yet.

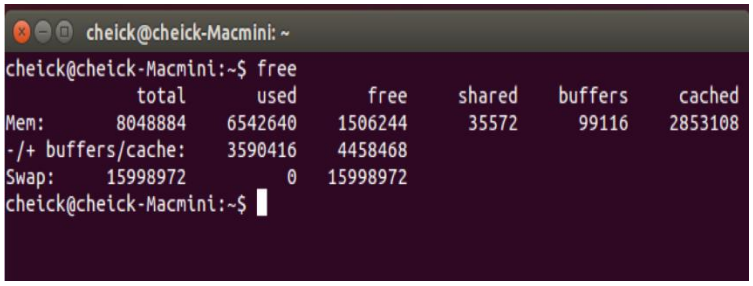

Figure 8: Virtual Memory Size in the Host Machine

5. iostat command: this command will display the OS storage I/O statics for the host hypervisor. As shown Figure 9, it display buffers and the caches will be the most obviously parts to identify VENOM vulnerabilities. Figure 9 also help us to identify VENOM vulnerability overflows the buffer before attack.

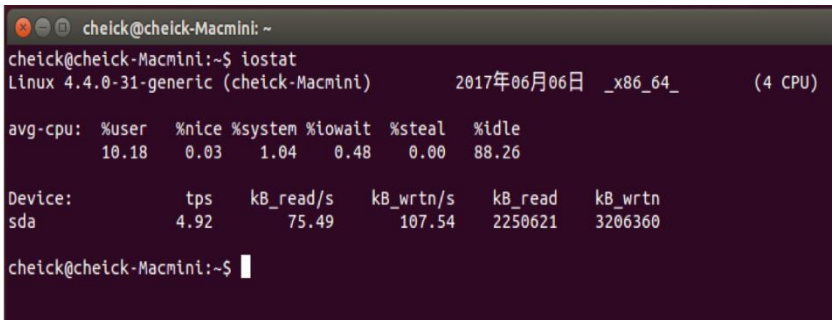

Figure 9: I/O Statics Command of Host Machine

All previous these commands (including CPU, Buffer Size, Virtual Memory and I/O) can identify VENOM attack before the malicious code attack vulnerabilities. Due to normally a virtual floppy drive is added to new virtual machines by default, such as on Xen and QEMU, even if the administrator explicitly disables the virtual floppy drive, an unrelated bug causes the vulnerable FDC code to remain active and exploitable by attackers. Most VM escape vulnerabilities discovered in the past were only exploitable in non-default configurations or in configurations that wouldn't be used in secured environments. As shown in Figure 10, before we implemented the source code of VENOM vulnerability.

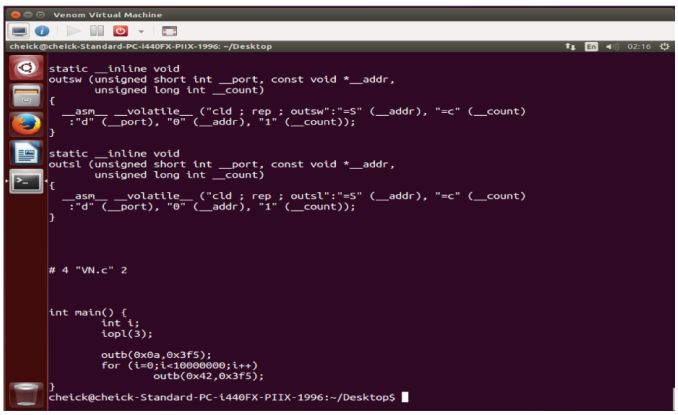

Figure 10: VENOM Vulnerability in the Hypervisor

As shown in Figure 11, after we launched VENOM attack source code in the sandbox, the source code came from Jason Geffner, who discovered this vulnerability while performing a security review of virtual machine hypervisors. Once we launched VENOM vulnerability from the guest machine (U1) by GCC complier (GNU Compiler Collection). The GCC is a compiler system produced by the GNU Project supporting various programming languages. GCC is a key component of the GNU tool chain and the standard compiler for most Unix-like Operating Systems.

Figure 11: Result of VENOM Vulnerability in the Hypervisor

Figure 12 presents the result of VENOM vulnerability sent from the guest machine (U1) to the hypervisor. As the results: "(**CRASH DUMPED**)" show us that the program (VENOM vulnerability) access to a portions of processor data or RAM memory and are erroneously copied to one or more files in the system. In computer, every application runs on a defined memory boundary and if part of the application exceeds the boundary then program crashes or in technical word "**CRASH DUMPED**". In conclusion, this source code (VENOM Vulnerability) exploits the flaw in the virtual disk driver code in virtual machines to break out of the guest operating system and interact with the host

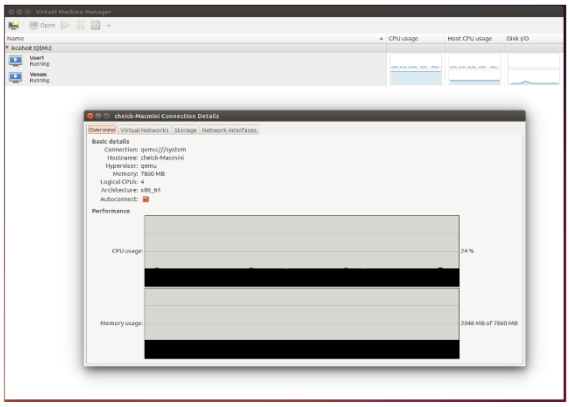

Figure 12: Statistic of hypervisor with VENOM Vulnerability

As shown in Figure 12 above, we can see that the usage of CPU in the Host machine (Hypervisor) is 24%. That means during the attack the guest machine sent I/O data through the FDC port and increased the CPU usage. Therefore, the VENOM vulnerability consumes CPU and also the disk resource in the I/O ports.

After we implemented VENOM attack in the guest machine. It's quite obviously that the problem of VENOM vulnerability resides in FDC (Floppy-Disk-Controller) and it described by this source code.

FDC's default:

```
#include <sys/io.h>

#define FIFO 0x3f5

int main() {
        int i;
        iopl(3);

        outb(0x0a,0x3f5); /* READ ID */
        for (i=0;i<10000000;i++)
                outb(0x42,0x3f5); /* push */
}
```

Figure 13: VENOM Vulnerability Source Code

As shown in Figure 13 above, the source code includes the <sys/io.h> library, which will be used by the main code. The code also defines the value of FIFO as: 0x3f5 (1013 in decimal). Thus, the code in the VM guest can write to the FIFO buffer by sending data to the FDC via its FD_REG_FIFO I/O port. Writes are handled by the function below, with each byte sent to the I/O port getting passed to this function as the value parameter.

5. Conclusion

In this paper, we investigate the security issue of VENOM attack in virtualization QENU virtualization environment. First, VENOM attack source code has been implemented in the environment of QEMU/KVM to observe its behaviors in the virtualization environment. Subsequently, the usage of CPU in the host hypervisor is 24%. Then, we tried to protect the hypervisor utilizing the VENOM protection mechanism, which is the most vulnerability part for cloud virtualization environment. The proposed mechanism can identify VENOM attack and try to lock the FDC port (0x3f5), which is responsible to send I/O command to the hypervisor.

References

[1] Ajay Kumara M.A and C.D. Jaidhar, “Hypervisor and virtual machine dependent Intrusion Detection and Prevention System for virtualized cloud environment,” *2015 1st International Conference on Telematics and Future Generation Networks (TAFGEN)*, Malaysia, May 26-28, 2015.

[2] A. Ahmad, N. Nasser and M. Anan, “An identification and prevention of theft-of-service attack on cloud computing,” *2016 International Conference on Selected Topics in Mobile & Wireless Networking (MoWNeT)*, pp. 11-13, April, 2016.

[3] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica and M. Zaharia, “A view of cloud computing,” *ACM Communications*, Vol. 53(4), pp.50-58, 2010.

[4] D. Bartholomew, “Qemu: a multihost, multitarget emulator,” *Linux Journal*, Vol. 2006(145), pp.3, 2006.

[5] S. Behrozinia and R.Azmi, “KLrtD: Kernel Level Rootkit Detection,” *2014 22nd Iranian Conference on Electrical Engineering (ICEE)*, Shahid Beheshti University, Iranian, May

20-22, 2014.

[6] L. Deng, Q. Zeng, W. Wang and Y. Liu, "EqualVisor: Providing Memory Protection in an Untrusted Commodity Hypervisor," *2014 IEEE 13th International Conference on Trust, Security and Privacy in Computing and Communications,* IEEE, pp. 300-309, Beijing.

[7] R. Hat, "Libvirt: The virtualization API," 2012.

[8] Jason Geffner, https://blog.trendmicro.com/understanding-the-venom-vulnerability/

[9] A. Kivity, Y. Kamay, D. Laor, U. Lublin and A. Liguori, "KVM: the Linux virtual machine monitor," *Linux symposium*, Vol. 1, pp. 225-230, 2007.

[10] PacketStorm, "Packetstorm," http://tinyurl.com/qhygrsu, accessed: 29-10-2014.

[11] B. Payne, S. Maresca, T. Lengyel K and A. Saba, "Libvmi," http://www.libvmi.com, accessed: 09- 07-2014.

[12] M. Schmidt, L. Baumgartner, P. Graubner, D. Bock and B. Freisleben, "Malware detection and kernel rootkit prevention in cloud computing environments," *2011 19th International Euromicro Conference on Parallel, Distributed and Network-Based Processing*, IEEE, pp. 603– 610, 2011

[13] M. I. Sharif, W. Lee, W. Cui and A. Lanzi, "Secure in-vm monitoring using hardware virtualization," *16th ACM conference on Computer and communications security*, ACM, pp. 477-487, 2009.

[14] D. Tsafrir, Y. Etsion and D. G. Feitelson, "Secretly Monopolizing the CPU Without Superuser Privileges," *16th USENIX Security Symposium on USENIX Security Symposium*, pp. 17:1–17:18, Berkeley, CA, USA, 2007.

[15] M. R. Watson, N. Shirazi, A. K. Marnerides, A. Mauthe, D. Hutchison, "Malware detection in cloud computing infrastructures," *IEEE Transactions on Dependable and Secure Computing*, pp. 192 –205, 2015.