A novel approach for distributed detection of infected machines in the cloud

Position paper

Mohammad Reza Memarian
Department of Information Technology
University of Turku
moreme@utu.fi

Mauro Conti
Department of Mathematics
University of Padua
conti@math.unipd.it

Ville Leppanen
Department of Information Technology
University of Turku
ville.leppanen@utu.fi

ABSTRACT

Abusing cloud services for cyber-crime intentions is a prevalent threat to cloud computing technology. Some of the cloud's essential characteristics are main origin of the threats for the cloud. One example of these kind of malicious usages is infected virtual machines forming botnets to conduct cyber-attacks such as Distributed Denial of Service (DDOS). DDOS can happen either against a system inside the cloud or a system outside of the cloud using cloud resources. While various works have been done to develop Intrusion Detection Systems (IDS) for cloud to mitigate threats like DDOS, Unfortunatly still the solutions for detection of botnets in the cloud are not mature. Still DDOS attacks using botnets are on rise. In fact most solutions are designed for being used on a single host. In this paper we present an aproach which has a broad view over all virtual instances in the cloud. In our design we place the detecting module in a higher level than host level in the cloud. Detection module coordinates and analyzes information gathered in an agent-less manner from Virtual machines (VMs) using Virtual Machine Introspection (VMI). The system gathers wide VM's system state information. The generic information gathered enables detection module to have a broad view over the entire cloud live images and detect malicious collaborative entities. As VMs in cloud are distributed over various clusters, cloud monitoring systems need to be distributed too.

1. INTRODUCTION

Cyber criminals are moving toward creating more sophisticated botnet intended malware to form undetectable botnets. Based on [12], DDOS attacks which are one of the consequences of botnet formation, tend to change their attack vector for remaining undetectable instead of relying on single vector in one attack. On the other hand, design of network protocols forms a portion of the problem as some of them were not designed by security in mind manner. From time to time, vulnerabilities in network protocols get exploited which may result in massive attacks and damage systems. Only one type of misuse from cloud services can be botnet formation. As mentioned earlier, conducting DDOS attack is one of the intentions of botnet controllers. There are various DDOS attacks which threaten cloud security. One of the DDOS attacks that is on rise is UDP reflection attack which is result of vulnerabilities in network protocols such as Network Time Protocol (NTP), Domain Name System (DNS) and Simple Service Discovery Protocol (SSDP) [12], [13]. Normally reflection attacks are hard to be detected as the preliminary request is a legitimate request with spoofed IP address of the victim. In these attacks, reflector which can be a vulnerable DNS or NTP server, has a much larger response to a query sent by a client. The amount of responded traffic from reflector to victim is measured by amplification factor. NTP amplification factor can be up to 556.9 meaning that the attacker can generate 556.9 times more traffic than the request that is sent [3]. Detection of these attacks are not easy as the request looks legitimate to the reflector. As users and companies are moving their data to clouds, attackers are redirecting their attack focus to cloud too. As described in [10], botnets are becoming more resilient and responding faster to countermeasures. They integrate multiple backup forms of command and control. On the other hand, cloud computing provides suitable infrastructure for both acceptable and malicious usages.

Based on [7], five essential cloud characteristics are on-demand self-service, broad network access, resource pooling, rapid elasticity and measured service. From the mentioned characteristics, two which empower malicious activities are rapid elasticity and on-demand self-service. On demand self-service refers to ability for users to provision cloud services without human interaction. Rapid elasticity refers to ability of users to shape the cloud resources consumption based on their need. Using online DDOS services does not need sophisticated knowledge and also they can be rented by as little cost as 5$ [11]. As a result, even small companies or individuals can have access to vast amount of computing resources in a short period of time with low financial requirements. Cloud Security Alliance (CSA) has addressed top nine cloud security threats in year 2013 [9]. Abuse of cloud services is one of the threats placed in the mentioned list as well as DDOS attacks.

As security is one of the largest obstacles in adopting cloud computing by companies. But the question is, what are the characteristics of cloud-specific vulnerabilities that create cloud-specific threats? A vulnerability is applicable to cloud computing if that vulnerability exists in one of the core technologies of cloud computing [15]. As cloud computing is designed on the idea of reducing cost and increasing efficiency, it is constructed on virtualization technology. So virtualization technology is one of the core technologies in cloud computing architecture. Vulnerabilities that are threats to virtualization are indeed threats to cloud computing too. Tackling vulnerabilities in virtualization leads to increasing security of the cloud as virtualization security has direct relationship with cloud security. Security of images running on virtual machine's structure in the cloud while holding applications, are critical to the cloud security in general as they are the most inner entity in cloud design virtualization circle. Securing VM images and monitoring activities of these small entities, boots up overall security of the cloud as images construct base system of the cloud. VM images must have high integrity as they determine initial state of the virtual machines running in the cloud. Distribution of these images may require some cautions as users in the cloud can use shared third party images. An example of these kind of shared image usage model can be a company advertising its application. The software firms configure the application on a VM image and publish that image for use and test of the application by cloud users. Other users can either publish their specific configured image and share it on the cloud either for free or to sell to others too. Shared images can be published either to a specific group of users or to a public group of users with the aim of users using a homogeneous image. This case can be counted as a case with a good intention. But there can be other cases that sharing an image may have malicious intention. Malicious users can configure and publish an infected image which may have malware embedded in it such as backdoor or a rootkit. Adoption of these images multiple times by users across the cloud can lead to malware propagation cloud-wide. While usage of shared images can have a high risk, cloud service providers do not have strong controls over image sharing as risk of using the shared images should be handled by image users. As an example Amazon, a cloud service provider, stated some security guides to help users to reduce risks of adopting shared images. It depicts that users must handle the risk in this case [1].

According to matters discussed above, creating network of collaborative images or botnet formation are easier in the cloud compared to conventional environments. When a bot enters a computer system, it should look for some distinctive vulnerabilities. But in the cloud (specifically in the case of image sharing), bots depend much less on victim's system software stack for exploitation. Ease of image sharing and interest for employing homogeneous images by groups of users and cloud service providers (CSP) are factors that accelerate malware propagation. Furthermore Infection methods may differ in a cloud botnet to mislead detection systems. While in the previous works, most of the focus has been on one way infection method. In [24], researchers explained risks that administrators, image publishers and image retrievers face in cloud image repositories. An image management system is proposed to control access to images, track source of images, and provide users and administrators with efficient image filters and scanners that detect and repair security violations. The mentioned research depicts the importance of risk reduction of image repositories and risks that can be involved in using shared cloud images. As mentioned earlier cloud is designed on the idea of reducing cost and increasing efficiency which leads cloud to homogenization. Specially in the case of VMs running cloud core services, homogeneous systems are desirable. Infection of one of these machines can reveal the vulnerability that exists in other homogeneous virtual machines too. It can end up in vast infection of homogeneous virtual machines across the cloud. Other scenario regarding this matter can be that one or multiple malicious users rent multiple virtual machines on the cloud. These virtual machines which are controlled from the beginning by specific users can do malicious activities in collaboration with each other.

As DDOS attacks and in general botnet related attacks are on rise in the cloud, presence of a system that has a wide look over the cloud and is able to correlate and coordinate all the information cloud-wide is vital. In this paper we review previous works done on cloud-botnet detection. Then we present a system design that adopts a de-centralized approach for botnet detection in the cloud by getting advantage of virtualization technology as one of the cloud core technologies. The goal of this research is to present a system to broadly detect malicious collaborative images in cloud-wide manner. As our system gets benefit of looking for symptoms leading to botnet instead of botnet behavior itself, detection rate increases while signature database size decreases.

The rest of this research paper is structured as follows: Section 2 provides background to the research area specifically: Botnet, cloud computing and virtual machine introspection technique. Section 3 focuses on related works and their approach in cloud-botnet detection and forensic virtual machines. Section 4 discusses technical approach toward information gathering and detection system design. Section 5 discusses implementation and experiment details and Section 6 gives conclusion and our future work direction.

2. BACKGROUND

This section presents a study about botnet, cloud computing and virtual machine introspection technique.

Botnet. Botnet is a group of infected computer systems that enables a controller to have control over the group. The infected entities are called zombies and in most cases, users of victim machines are not aware that their system is compromised. There are various malicious usages of botnets for controllers. Examples of these usages are conducting DDOS attacks, spamming, search engine optimization poisoning, pay-per-click fraud, financial fraud, bitcoin mining and information stealing [8]. There are various ways that victim systems are infected such as through opening an email with malicious content, installing pirated software and visiting malicious websites. When systems are infected, malware installs a backdoor to enable botnet controller to have consistent communication with zombies (Infected machine in botnet is called zombie). Infected systems may send primary

victim system's information such as IP address, geographical location, operating system type, host name as other information which may help classifying zombies. From time to time malware may download newer version of itself to get updated. Infected machines and bot controller may communicate with each other through various ways such as P2P protocol, HTTP commands and IRC channels. Nowadays, cost and skills of having a botnet is reduced. Some of the botnets source code can be found in underground black market websites with very low cost while they are designed to be easily worked.

Cloud Computing. Based on national institute of standards and technology (NIST), cloud computing is a model for enabling on-demand access to a shared pool of configurable computing resources. These resources are able to quickly be employed with minimal management effort and human interaction [7]. Indeed cloud computing is changing the way that computing services are offered. Cloud services are offered through 3 service models which are: Software as a service (SaaS), Platform as a service (PaaS) and Infrastructure as a service (IaaS). Also cloud computing is deployed based on 4 deployment models. The choice of deployment model depends on the environment in which cloud is going to be deployed on. The 4 cloud deployment models are: private cloud, community cloud, public cloud and hybrid cloud. As mentioned earlier, security issues are one of the biggest obstacles in adoption of cloud through companies.

Virtual Machine Introspection (VMI). VMI is a technique, enabling monitoring VM by collecting system level data from memory. VMI is used for developing wide vareity of security related applications. The prevalent use is in security monitoring applications that VMI enables VM's system state data collection from a secure place outside of the monitored VMs. Virtual machine introspection is done in an agent-less manner meaning that in order to collect monitoring data, there is no need for an agent to be installed in monitored system. Traditional monitoring methods required monitoring agent to be placed in the monitored VMs. One of the disadvantage of agent-oriented monitoring systems is that in case the monitored VMs are compromised, malware on the systems may deactivate the agent on the monitored systems. This results in lack of having accurate and on demand data. VMI tackles the problem of agent-oriented techniques in virtual environments. The monitoring can be done from a secure place like virtual machine monitor (VMM) or any other privileged virtual machine. Garfinkel and Rosenblum in [23], presented an intrusion detection system design based on VMI technique. As any entity in computing can be prone to attacks, so VMI is too. Researchers in [14] showed that VMI can be used to present inaccurate data to the monitoring system outside of monitored VMs.

3. RELATED WORK

In this section we review previous works done on botnet detection in the context of cloud and also discuss on forensic virtual machine which is quite a new term.

Cloud-Botnet detection. Researchers in [17], designed a passive and an active malware detection module in VMM to actively look for information in VMs without installing agent (for using VMI). The solution presented in their work mainly focuses on functions in one host and detects zombie machines, based on trained node about bot behaviors. The solution does not have a wide view over the cloud, and it makes the botnet profile based on just the API calls done by the applications. A research paper by Kebande and Venter [18], proposed botnet detection methods in the cloud environment using Artificial Immune System (AIS). This mechanism uses negative selection algorithm to match whether the botnet belongs to self or non-self pattern. It gets done by training some detectors on identifying malicious activities pattern. In the time of attack (when bots are attacking victims), AIS is trained to detect a malicious activity pattern by observing the behavior based on the network traffic movement. Beidermann and Katzenbeisser in [21] presented a research work to detect computer worms in the cloud based on the spreading behavior. The solution looks for inconsistency in behavior of machines. Their system is limited to only looking for two kinds of information that are start of a new process or a loaded module which is not listed in the predetermined white list or its presence is not normal. In this case, start of a black list process even on several machines can not necessarily indicate malicious activity. In this solution also monitoring stages are mentioned that have a random look into VMs which is not suitable for continues monitoring. The information obtained from each VM using VMI is sent to a central spreading monitor in the VMM layer that compares the lists. Although it is mentioned that the system is able to detect various unknown malware but trojans today do not necessarily start a new process with an undefined name. Watson in [19] presented a distributed detection system that combats malware in multi-server cloud. The research proposed having agents in each hypervisor of servers that communicate with each other and pass obtained information to each other. Without a central decisioning and information processing point, solutions will not be effective.

Forensic Virtual Machine (FVM). Due to modular design trend of malware, researchers in [16], presented a method of detecting malware by identifying the symptoms of malicious behavior as opposed to looking for malware itself. To accomplish task of symptom detection, they present small introspecting virtual machines. These machines that take advantage of VMI technique to look for existence of desired symptom are called Forensic Virtual Machines (FVM). FVMs are small independent privileged VMs which inspect memory page of other introspected virtual machines. While introspecting VMs, each FVM looks for existence of only one symptom. Findings of FVMs are reported to a central command and control module. FVMs choose the introspected machine randomly using mobility algorithm embedded in them. When an FVM finds existence of a symptoms, other FVMs will be commanded to check existance of other symptoms too on the same system. So one VM may not be introspected by any FVM at a given time while other VMs are under introspection by several FVMs. So as monitoring of symptoms are done randomly, a symptom may appear and disappear before the FVM arrives on the system or it

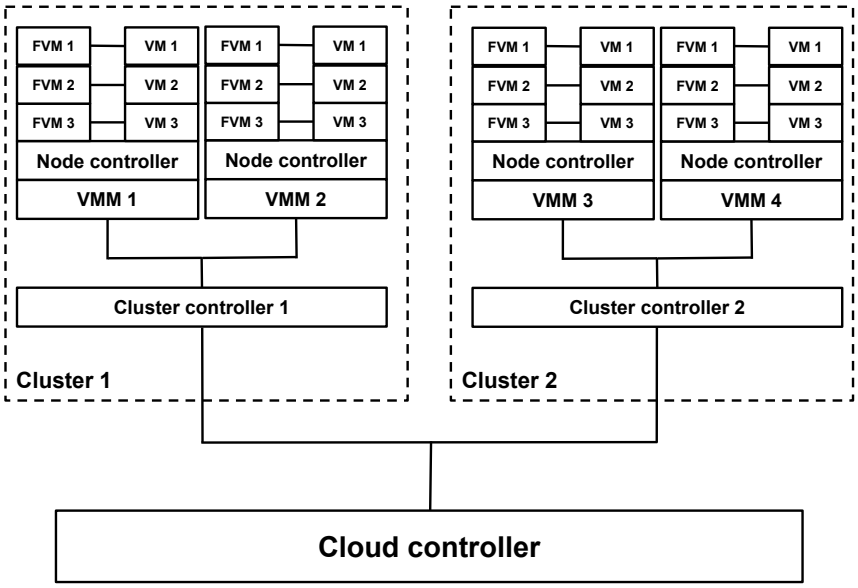

Figure 1: Typical cloud platform design with FVM incorporated inside

may start after FVM introspection is done. As number of FVMs has a direct relationship with number of symptoms the system look for, FVMs should be small system. Researchers in [22], implemented FVMs using MiniOS. MiniOS is a small operating systems distributed by Xen source code and is intended to be used for dom0 disaggregation. Although MiniOS lacks variety of a normal OS functionalities, but they have advantage of being very small in size. As an instance basic networking functionalities can be added to it by using a mini TCP-IP stack module.

4. OUR PROPOSED APPROACH

In this section, we present our approach toward information gathering, analysis and processing of obtained data.

4.1 Data gathering method

As mentioned earlier, VMI can help security applications by enabling access to system level information of guest VMs. So we get advantage of VMI technique to gather needed information from memory of target VMs. Introspection technique can give a set of system-level information at any given point in time about each live virtual machine. Examples of these informations can be list of running processes, loaded modules, opened files and running services. In our design, we use concept of FVM for data gathering entity. As depicted in Figure 1, we assign one FVM per VM to retrieve system level information from each VM's memory page at any given time. Each FVMs in our design is dedicated to one VM and looks for existence of all symptoms only in the VM assigned to it. Assigning one FVM per VM as opposed to [22], has various benefits. Introspected VMs trust to only one FVM which results in monitored VM's Trusted Computing Base (TCB) reduction. In addition, all the symptoms are checked at once and frequently instead of random check which is not feasible in terms of symptom detection. In time of FVM infection, other monitored machines can remain immune from consequences of one of the FVM's infection as FVMs do not randomly check other VMs. In Figure 1, a conceptual design of cloud platform with FVMs added is shown. Cloud controller is outermost entity is conceptual cloud platform design.

4.2 On gathered data

As mentioned earlier, FVMs look for symptoms of malicious activities as opposed to malware. These symptoms need to be defined by determining inconsistencies in VMs as our approach focuses on system level inconsistencies. Based on malware analysis information available on [2], we constructed a table depicting common modifications that 9 botnet intended malware do to victim systems after infecting them. Under each malware's name the year that was discovered has been mentioned. For example, all malware listed in Table 1 create registry entry after infecting victim. So it can be resulte that creating registry entry is a prevalent action among malware. Creation of registry value under a specific key would be a symptom as opposed to entry value. Ofcourse creation of registry value happens by legitimate processes too. For this reason, system looks for set of diverse symptoms. Occurrence of number of them among several machines would trigger security alert.

Based on information depicted in Table 1 and [20], a sample set of inconsistencies that can be considered prevalent among botnet intended malicious codes are: File/Folder creation, registry key creation, existence of hidden (unlinked) processes, existence of hidden (unlinked) DLL and existence of unnormal loaded service in memory.

4.3 Categorization of obtained data

System modification	Rustock.B (2006)	Virut (2007)	Koobface (2008)	Tidserv (2008)	Qakbot (2009)	Pilleuz (2009)	Zbot (2010)	Carberp.B (2014)
File/Folder created	✓		✓	✓	✓	✓	✓	✓
Files/Folder deleted				✓		✓		
Files/Folder modified		✓		✓	✓	✓		
Registry entry created	✓	✓	✓	✓	✓	✓	✓	✓
Registry entry deleted								
Registry entry modified								
Kernel Modification	✓							
Code injection into process/DLL/App	✓	✓	✓	✓	✓	✓		

Table 1: Botnet intended malware functionalities based on [2]

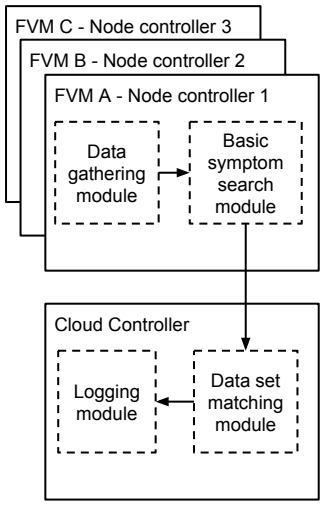

Figure 2: Architecture of detection system

As depicted in Figure 3, information gathered at each host gets in relevant data structure. These information will be sent to cloud controller through node controller and VMM. Digging down in the categorization of obtained data, as shown in Figure 2, after data gathering module in FVM gathers the data, it passes them to a basic symptom search module. This module only indicates existence or non existence of symptoms in VMs. The obtained data structure is transfered to data set matching module in cloud controller. This module may use machine learning techniques such as similarity learning to identify data sets that even do not exactly have same entries in their symptom detection list. In case of accurate matching, one notification will be sent to logging module to log the events. Another command can be sent to privileged domain of each host to take necessary action to stop the threat. As analytic module is placed in cloud controller level, VMM does not have any role in analyzing data which results in offloading this task from VMM.

5. IMPLEMENTATION AND EXPERIMENTS

In this section, we describe an overview of configured system.

5.1 XEN

We used Xen as VMM for our system configuration. Ubuntu 14.04 (Linux kernel 3.13.0-24-generic) is our choice for Dom0 Implementation. The reason for using Xen is that while Xen is a widely used VMM, it is an open source virtualization solution. Large cloud service providers like Amazon use Xen as the VMM for virtualization infrastructure [6]. VMs running on Xen are referred as domains. There are two domain types in Xen, privileged domain and unprivileged domain. A supervisor like VM, called dom0 runs in privileged mode. It is the first VM that boots after VMM starts and holds hardware drivers and control software stack. Other guest VMs run in unprivileged mode. Dom0 is allowed to have access to memory pages of other VMs so VMI is possible from Dom0 level. By modifying Xen access control, other VMs can be granted access to introspect other VMs too. VMs running on Xen should run on one of the two available modes. The two modes are Hardware Virtualization (HVM) and Para Virtualization (PV). In HVM, VMs are not aware that it is running in a virtualized environment. Whereas in PV mode, VMs experience some modification and is aware that is running on virtual platform. VMs running in PV mode experience faster system performance. Closed source operating systems like Microsoft windows, must run is HVM mode because modification of them, is not possible. We run a guest operating system running in HVM mode while having Microsoft windows 7 as operating system.

5.2 Introspection tools

We used LibVMI version 4.4 to conduct VMI for obtaining information from VM's memory. Libvmi is an application programming interface (API) which enables introspecting system to read from and write to memory of monitored system. Libvmi is written in C while offering a python wrapper to users to be able to write the introspection applications in python as well [4]. Libvmi supports Virtual machines running on Xen and KVM VMM. For introspection to take place using Libvmi, introspecting VM should have access to kernel symbols of introspected VMs. So when an application requests to view some data through introspection, kernel symbols of the target system are accessed through accessing to system.map file which should be placed in /boot directory of introspecting machine to receive virtual address (VA) of the symbol. Then through several mapings, correct data page is found and returned to LibVMI and LibVMI returns the requested data to the VMI application. LibVMI is well

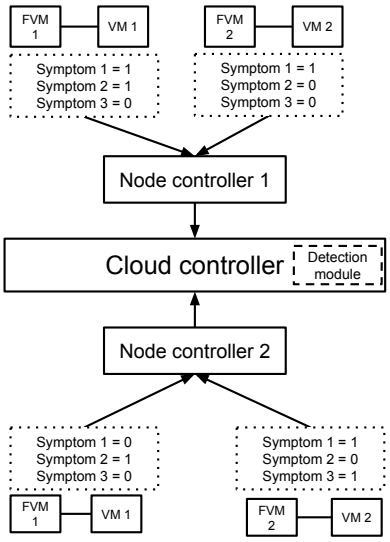

Figure 3: Transfer of gathered data to cloud controller

integrated with Volatility [5]. Volatility is an advanced open source memory forensic framework which offers variety of functions to users for extraction of data from memory snapshots and dumps. As Libvmi has integrity with Volatility, it is possible to use Volatility functions on live VMs while using Libvmi.

5.3 Experiments

We setup an environment to run some preliminary experiments and simulation. We run a guest VM (domU) to infect it with a malware sample. In this test we infect domU with an instance of Koobface. Libvmi and Volatility are configured to gather information from Dom0. We used from psscan plug-in of volatility to gather all running processes including hidden (Unlinked) processes that are not shown in the task list of the operating system and terminated processes which with random introspection technique is not detectable. When domU became infected, a process with name bolivar30.exe started. Then immidiately dllhost.exe and regedit.exe were started and terminate. They were shown by psscan plug-in as terminated process. As a result of regedit.exe start and termination, a registry entry was created to start the malicious process each time the system starts. So in this case our framework looks into the registry key which is prevalent for creation of value to check whether any value is added to the key or not. It does not check what values are added. Of course creation of registry entry can be done for legitimate purposes too but existence of this symptoms in conjunction with others in a cloud-wide manner can be a brief indication of infection. For the system to be able to distinguish among different infected groups, variety of different symptoms must be checked.

6. CONCLUSION AND FUTURE WORK

Abusing cloud services is a prevalent threat toward cloud computing security. The need for having distributed and efficient monitoring systems is vital to empower security of the cloud. By making sure about security of the most inner entity of the cloud computing which are virtual machine images and VMs, the entire security of the cloud can be increased. Having distributed monitoring systems that are capable of widely and actively monitor, log and report malicious activities and movements is essential. In this paper we presented a novel approch which is able to act as a super system looking into its subsystems and strongly resulting in relating events occurring in each cluster and detect collaborative running malicious images.

There are two future directions which we would like to follow as our future work. First we would like to move more toward depicting detection ratio of our approach by implementing the proposed approach in different cloud scenarios and testing under various attack vectors. Second is enhancing privacy of data obtained from guest VMs using VMI as is a very important point. In the researches which VMI is involved, privacy of user's activities is generally overlooked in favor of secure monitoring. VMI technique can reveal various private information from inside the guest VMs. These information can be target of misuse by malicious insiders. So our second future research direction will be enhancing privacy of user's data in cloud monitoring systems using VMI.

7. REFERENCES

[1] Building AMIs for AWS Marketplace. https://aws.amazon.com/marketplace/help.
[2] Symantec website. www.symantec.com.
[3] United States computer emergency readiness team. www.us-cert.gov.
[4] VMItools. https://code.google.com/p/vmitools/.
[5] Volatility foundation website. www.volatilityfoundation.org.
[6] XEN project website. www.xenproject.org.
[7] Defenition of cloud computing. Technical report, National Institute of Standards and Technology (NIST), 2011.
[8] Anatomy of a Botnet. Technical report, Fortinet, 2012.
[9] Nine threats to cloud. Technical report, Cloud Security Aliance (CSA), 2013.
[10] Security threat report 2014. Technical report, Sophos, 2014.
[11] The continued rise of DDoS attacks. Technical report, Symantec, 2014.
[12] Verisign distributed denial of service trends report, 2nd quarter 2014. Technical report, Versign, 2014.
[13] Verisign distributed denial of service trends report, 3rd quarter 2014. Technical report, Versign, 2014.
[14] Bahram, S. and Xuxian Jiang and Zhi Wang and Grace, M. and Jinku Li and Srinivasan, D. and Junghwan Rhee and Dongyan Xu. DKSM: Subverting Virtual Machine Introspection for Fun and Profit. In *Proceedings of 2010 29th IEEE Symposium on Reliable Distributed Systems (SRDS)*, pages 82–91, 2010.
[15] Grobauer, B. and Walloschek, T. and Stocker, E. Understanding Cloud Computing Vulnerabilities. *IEEE Security Privacy*, 9(2):50–57, 2011.
[16] K. Harrison, B. Bordbar, S. Ali, C. Dalton, and A. Norman. A Framework for Detecting Malware in Cloud by Identifying Symptoms. In *Proceedings of the*

2012 IEEE 16th International Enterprise Distributed Object Computing Conference (EDOC), pages 164–172, 2012.

[17] S.-W. Hsiao, Y.-N. Chen, Y. Sun, and M. C. Chen. A cooperative botnet profiling and detection in virtualized environment. In *Proceedings of 2013 IEEE Conference on Communications and Network Security (CNS)*, pages 154–162, 2013.

[18] V. Kebande and H. Venter. A cognitive approach for botnet detection using Artificial Immune System in the cloud. In *Proceedings of 2014 Third International Conference on Cyber Security, Cyber Warfare and Digital Forensic (CyberSec)*, pages 52–57, 2014.

[19] M. R. Watson. Malware detection in the context of cloud computing. In *Proceedings of 13th annual post graduate symposium on the convergence of telecommunication, networking and broadcasting*, 2012.

[20] A. H. Michael Sikorski. *Practical malware analysis*. 2012.

[21] Sebastian Biedermann, Stefan Katzenbeisser. Detecting Computer Worms in the Cloud. In *Proceedings of 2011 Open Problems in Network Security (iNetSec)*, 2011.

[22] Shaw, A.L. and Bordbar, B. and Saxon, J. and Harrison, K. and Dalton, C.I. Forensic Virtual Machines: Dynamic Defence in the Cloud via Introspection. In *Proceedings of 2014 IEEE International Conference on Cloud Engineering (IC2E)*, pages 303–310, 2014.

[23] Tal Garfinkel, Mendel Rosenblum. A virtual machine introspection based architecture for intrusion detection. In *Proceedings of 2003 Network and Distributed System Security (NDSS) Symposium*, 2003.

[24] Wei, Jinpeng and Zhang, Xiaolan and Ammons, Glenn and Bala, Vasanth and Ning, Peng. Managing Security of Virtual Machine Images in a Cloud Environment. In *Proceedings of the 2009 ACM Workshop on Cloud Computing Security (CCSW)*, pages 91–96, 2009.