A Collaborative Approach for Distributed Detection of Infected Virtual Machines in the Cloud

(Position paper)

Mohammad Reza Memarian
University of Padua
Padua, Italy
memarian.m.reza@gmail.com

Mauro Conti
University of Padua
Padua, Italy
conti@math.unipd.it *

Ville Leppanen
University of Turku
Turku, Finland
ville.leppanen@utu.fi

ABSTRACT

Leveraging cloud services, companies and organizations can significantly improve their efficiency, as well as building novel business opportunities. Unfortunately, some of the cloud's essential characteristics are also the main source of threats. Among these threats, a relevant one is the case of infected Virtual Machines (VMs) 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 the cloud, while using cloud resources to run the attack. In order to mitigate threats like DDoS ones, researchers developed Intrusion Detection Systems (IDSs) for cloud. Unfortunately, still the solutions for detection of botnets in the cloud are not mature yet; in fact DDoS attacks using botnets are on the rise. The main reason for this limitation is that current IDS solutions are designed with limited view over a single host while having no collaboration with detection modules in other hosts.

In this paper, we present a novel IDS approach that has a broad view (i.e., over several virtual instances in the cloud) to address the above concern. In particular, in our design we place the detection module in a higher level than host level in the cloud. Detection module coordinates and analyses information gathered in an agent-less manner from VMs using Virtual Machine Introspection (VMI). The system gathers wide VM's system state information. Such information enables the 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. We believe the approach proposed in this paper is promising, and we expect this work to open discussion and a new vein of research.

*Mauro Conti was supported by Marie Curie Fellowship PCIG11-GA-2012-321980, funded by the European Commission for the PRISM-CODE project. This work has been partially supported by the TENACE PRIN Project 20103P34XC funded by the Italian MIUR, and by the Project "Tackling Mobile Malware with Innovative Machine Learning Techniques" funded by the University of Padua.

1. INTRODUCTION

As users and companies are moving their data to clouds, attackers are redirecting their attacks to cloud as well. Cloud computing provides suitable infrastructure for both honest and malicious usages such as botnets. Botnet is a group of infected computer systems that enables a controller to have control over the group. The infected entities are called zombies. Examples of botnet functions are: DDoS attacks, spamming, search engine optimization poisoning, pay-per-click fraud, financial fraud, bitcoin mining and information stealing [6], [12]. Botmasters specifically benefit from the provided infrastructure as cloud computing offers them various possibilities to conduct their desired attacks [13]. Meanwhile, cyber criminals are moving toward creating more sophisticated botnets that can scape from being detected. For instance, botmasters integrate multiple backup forms of command and control. As described in [8], botnets are becoming more resilient and responding faster to countermeasures. Based on [10], DDoS attacks, which are one of the consequences of botnet formation tend to change their attack vector during a single attack. This matter can help attack sources to remain hidden while increasing attack impact.

As abuse of cloud services is a threat to cloud computing [7], it is vital for Cloud Service Providers (CSPs) to detect botnets in their cloud. Also, it is suitable for CSPs to prevent the probable related attacks and traceback to the botmaster as botmasters may use from cloud services to host their command and control servers [28]. Cloud resources can be misused to conduct attacks toward targets either inside or outside of the cloud. Fraudulent Resource Consumption (FRC) is an example of attacks by botnets toward a service hosted in the cloud [21], [15], [20]. On the other hand, design of network protocols forms a portion of the problem since some of them were not designed with security in mind.

As mentioned earlier, conducting DDoS attack is one of the intentions of botmasters and a prevalent threat to cloud computing [7]. Various recent DDoS attacks which are on the rise and threaten cloud security are Network Time Protocol (NTP), Domain Name System (DNS) and Simple Ser-

vice Discovery Protocol (SSDP) amplification attacks, which are result of vulnerabilities in network protocols [10], [11]. Detection of these kind of attacks is not simple based on the network traffic analysis. As amplification attacks are on the rise and have relatively high impact on the victim, researches such in [17] are demanding to confront the attacks.

Security problems are one of the largest obstacles in adopting cloud computing by companies. There are some cloud-specific vulnerabilities that are sources of the security problems in the cloud. 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 [14]. Hence, increasing security in virtual environment, increases security in cloud environment. VMI is one of the benefits that is obtained in the cloud through virtualization. VMI is a technique, that enables collection of system level information of a monitored VM from a secure point like Virtual Machine Monitor (VMM) by reading information from memory of monitored VM. Introspection through VMI is done in an agent-less manner. Introspection can take place through Forensic Virtual Machines (FVM). As depicted in Figure 1, FVMs are Small introspecting VMs looking for symptoms of malicious behaviour in other VMs by getting help of VMI.

An issue which paves the way for malicious users to distribute their infected image in the cloud is image sharing. Image sharing which is common among cloud users enables users to publish and share their VM images with other users in the cloud. Shared images can be published to a specific group of users with the aim of users using a homogeneous or pre-configured image. Homogeneous systems are desired on cloud core services as well. Infection of one of the homogeneous systems may reveal vulnerabilities in other systems too. Adoption of infected shared images multiple times by users across the cloud can lead to cloud-wide infected image usage. Shared image usage may have risk for various entities involved [27], as CSPs do not have strong controls over security of shared images. Infection of various homogeneous images cloud-wide strongly requires a system that looks for similar malicious symptoms among various hosts holding VMs.

According to matters discussed above, creating network of collaborative machines or in other words botnet formation is easier in the cloud compared to conventional environments. When a bot enters a system, it should look for some distinctive vulnerabilities. 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 user groups and CSPs are factors that accelerate malware propagation in the cloud.

Contribution. As presented in [18], it is possible to determine location of cloud instances. So attackers can truly distribute and choose infected machines involved in a DDoS attack to conduct a strong distributed attack. Refer to the mentioned concern and with refer to the fact that DDoS attacks and in general botnet related attacks are on the rise in the cloud [9], presence of a system that has a wide look over the cloud and be able to correlate all the information cloud-wide is vital. In this paper, we review previous works done on cloud-botnet detection. Then we present an approach that adopts a decentralized approach for botnet detection in the cloud. Our approach takes advantage from virtualization as one of the cloud core technologies. Unlike previous works, goal of this research is to present a system to broadly detect malicious collaborative images in cloud-wide manner using VMI. Our design looks for symptoms leading to botnet instead of botnet behaviour itself in the VMs. Our aim is to detect and mitigate the threat from the sources which are the infected VMs. We expect our approach to lead to an increase detection rate while signature database size decreases.

Organization. The remaining of the paper is structured as follows. Section 2 introduces and discusses the main existing approaches for cloud-botnet detection and FVM. In Section 3 we introduce our novel design for detection of a set of infected systems, while in Section 4 we discuss implementation details and show preliminary experimental results. Finally, in Section 5 we draw our conclusions, and discuss possible future research directions.

2. RELATED WORK

In this section, we review previous works done on botnet detection in the context of cloud and also discuss FVMs.

Cloud-Botnet detection. Researchers in [19] designed a passive and an active malware detection module in VMM to actively look for information in VMs 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 behaviours. 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 [22] 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 behaviour based on the network traffic movement.

Beidermann and Katzenbeisser in [25] presented a research work to detect computer worms in the cloud based on the spreading behaviour. The solution looks for inconsistency in behaviour of machines. Presented 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 process which is not is the white list 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 mon-

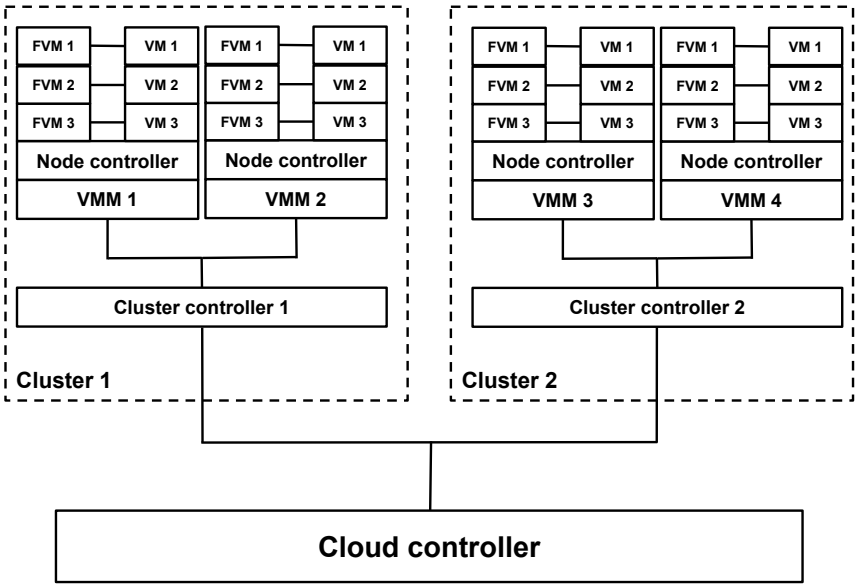

Figure 1: Typical cloud platform design with FVM incorporated inside it

itor that compares the lists. Although it is claimed 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 [23] presented a distributed detection system that combats malware in multi-server cloud. The research proposed having agents in each VMM 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], used method of detecting malware by identifying the symptoms of malicious behaviour as opposed to looking for malware itself. To accomplish the task, they use FVMs. In their design, each FVM looks for existence of only one symptom. FVMs choose the introspected machine randomly using mobility algorithm embedded in them. So one VM may not be introspected by any FVM at a given time, while other VMs are under introspection by several FVMs.

Researchers in [26] implemented FVMs using MiniOS. MiniOS is a small operating systems distributed by Xen source code. Although MiniOS lacks variety of a normal OS functionalities, but they have advantage of being very small in size.

3. OUR PROPOSED APPROACH

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

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. Examples of informations obtained by VMI are: 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 FVM in our design is dedicated to one VM and looks for existence of all symptoms in the VM assigned to it.

Assigning one FVM per VM as opposed to [26], has various benefits. Introspected VMs trust only one FVM, which results in monitored VM's Trusted Computing Base (TCB) reduction. In addition, all 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. Indeed Figure 1 depicts a high level architecture design of cloud platform with FVMs incorporated in it. Cloud controller is outermost entity in cloud platform.

On gathered data. The list of checked 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 nine botnet intended malware do to victim systems after infecting them. Under each malware's name the year that malware was discovered has been mentioned. For example, all mal-

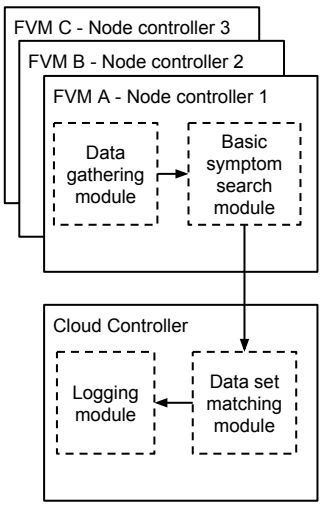

Figure 2: Architecture of detection system

ware listed in Table 1 create registry entry after infecting victim. So it can be resulted 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 given in [24], a sample set of inconsistencies that can be considered prevalent among botnet intended malicious codes can be: File/Folder creation, registry key creation, existence of hidden (unlinked) processes, existence of hidden (unlinked) DLL and existence of abnormal loaded service in memory.

Categorization of obtained data. 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 categorization of obtained data, as shown in Figure 2, data gathering module in FVM gathers the data and 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 transferred 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 such as quarantining the suspicious VMs. As detective module is placed in cloud controller level while VMM does not have any role in analysing data. This results in offloading analytic task from VMM.

4. IMPLEMENTATION

In this section, we describe an overview of the configured system. In particular, in Section 4.1 we describe Virtualization technology we used, in section 4.2 Introspection tools we used to conduct VMI, in section 4.3 the experiment that we did.

4.1 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 CSPs like Amazon use Xen as the VMM for virtualization infrastructure [5]. 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 as privileged domain. Other guest VMs run as unprivileged domains. By modifying Xen access control, Guest VMs in addition to Dom0 will be able to introspect memory of other VMs.

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 they are running in a virtualized environment. Whereas in PV mode, VMs experience some modification and are aware that they are running on virtual platform. Closed source operating systems like Microsoft Windows, must run is HVM mode as modification of them is not possible. We run a guest operating system running in HVM mode while having Microsoft Windows 7 as operating system.

4.2 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 VM to read from and write to memory of introspected VM [3]. LibVMI supports VMs running on Xen and KVM.

In order for introspection to take place using LibVMI, introspecting VM should have access to kernel symbols of introspected VM. 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 symbols. Then through several mappings, correct data page is found and returned to LibVMI. LibVMI returns the requested data to the requesting application.

LibVMI is well integrated with volatility [4]. 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 by adding LibVMI address space to address space list of volatility.

4.3 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

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]

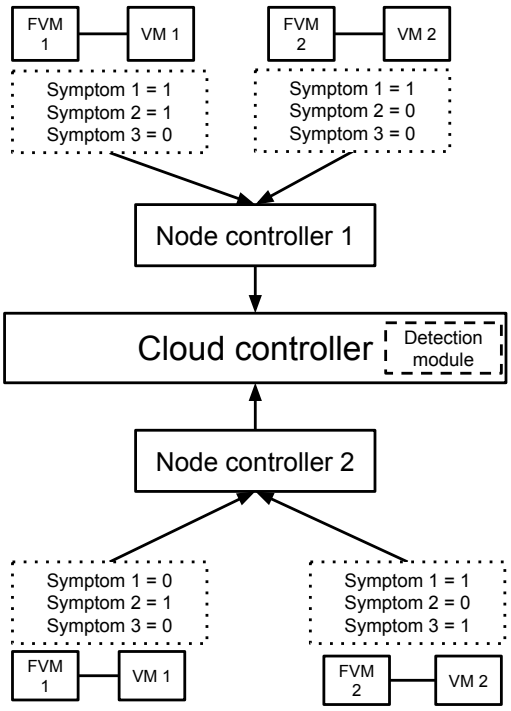

Figure 3: Transfer of gathered data to cloud controller

an instance of Koobface [1]. LibVMI and volatility are configured to gather information from Dom0 level. We used from `psscan` plug-in of volatility [4] to gather all running processes including hidden (unlinked) processes that are not shown in the task list of the operating system. Also using the same plug-in, terminated processes which with random introspection technique is not detectable are obtainable.

When domU became infected, a process with name `bolivar30.exe` started. Then immediately `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. Indeed creation of registry entry can be done for legitimate purposes too. Existence the symptoms in conjunction with each other in a cloud-wide manner can be a brief indication of infection. For the detection system to be able to distinguish among different infected groups, diverse set of symptoms must be checked.

5. 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 entities of the cloud computing which are VM images and running 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 approach which is able to act as a super system, looking into its subsystems, and resulting in relating events occurring in each cluster and detect collaborative running malicious images.

There are two future directions which we would like to take. First we would like to move more toward depicting detection ratio of our approach. We aim to completely implement the proposed approach, test it under various attack vectors and obtain related benchmark results. Second we would like to enhance privacy of data obtained from guest VMs using VMI as is a very important point. In the re-

searches which VMI is involved, privacy of obtained user's data is generally overlooked in favour 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.

6. REFERENCES

[1] Koobface virus information on Sophos website. https://nakedsecurity.sophos.com/?s=koobface.

[2] Symantec website. www.symantec.com.

[3] VMItools. https://code.google.com/p/vmitools/.

[4] Volatility foundation website. www.volatilityfoundation.org.

[5] XEN project website. www.xenproject.org.

[6] Anatomy of a Botnet. Technical report, Fortinet, 2012.

[7] Nine threats to cloud. Technical report, Cloud Security Aliance (CSA), 2013.

[8] Security threat report 2014. Technical report, Sophos, 2014.

[9] The continued rise of DDoS attacks. Technical report, Symantec, 2014.

[10] Verisign distributed denial of service trends report, 2nd quarter 2014. Technical report, Versign, 2014.

[11] Verisign distributed denial of service trends report, 3rd quarter 2014. Technical report, Versign, 2014.

[12] Ardagna, C.A. and Conti, M. and Leone, M. and Stefa, J. An Anonymous End-to-End Communication Protocol for Mobile Cloud Environments. *IEEE Transactions on Services Computing*, pages 373–386, 2014.

[13] Clark, Kassidy P. and Warnier, Martijn and Brazier, Frances M. T. Botclouds - The Future of Cloud-based Botnets? In *Proceedings of the 1st International Conference on Cloud Computing and Services Science (CLOSER 2011)*, pages 597–603, 2011.

[14] Grobauer, B. and Walloschek, T. and Stocker, E. Understanding Cloud Computing Vulnerabilities. *IEEE Security Privacy*, 9(2):50–57, 2011.

[15] Gruschka, N. and Jensen, M. Attack Surfaces: A Taxonomy for Attacks on Cloud Services. In *Proceedings of 2010 IEEE 3rd International Conference on Cloud Computing (CLOUD)*, pages 276–279, 2010.

[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] Herzberg, Amir and Shulman, Haya. DNS Authentication As a Service: Preventing Amplification Attacks. In *Proceedings of the 30th Annual Computer Security Applications Conference (ACSAC)*, pages 356–365, 2014.

[18] Herzberg, Amir and Shulman, Haya and Ullrich, Johanna and Weippl, Edgar. Cloudoscopy: Services Discovery and Topology Mapping. In *Proceedings of the 2013 ACM Workshop on Cloud Computing Security Workshop (CCSW)*, pages 113–122, 2013.

[19] 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.

[20] Idziorek, Joseph and Tannian, Mark and Jacobson, Doug. Detecting Fraudulent Use of Cloud Resources. In *Proceedings of the 3rd ACM Workshop on Cloud Computing Security Workshop (CCSW)*, pages 61–72, 2011.

[21] Idziorek, Joseph and Tannian, Mark F. and Jacobson, Doug. The Insecurity of Cloud Utility Models. *IT Professional*, pages 22–27, 2013.

[22] 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.

[23] 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.

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

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

[26] 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.

[27] 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.

[28] Wenjie Lin and Lee, D. Traceback Attacks in Cloud – Pebbletrace Botnet. In *Proceeding of 2012 32nd International Conference on Distributed Computing Systems Workshops (ICDCSW)*, pages 417–426, 2012.