EyeCloud: A Framework for Cloud-wide Detection of Infected Virtual Machines

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. So offering secure cloud services to users is vital. 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 may occurs either against a system inside the cloud or a system outside the cloud while misusing cloud resources. In order to mitigate DDoS and relevant threats in the cloud, researchers developed Intrusion Detection and Prevention System (IDPS) for cloud while mostly overlooking prevention capability. 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.

In this paper, we present EyeCloud, an IDS system that helps to prevent DDoS attacks by early identification of infected entities. EyeCloud has a broad view (i.e., over several virtual instances in the cloud) to address the above concern. In particular, data collection is done using dedicated entities per VM while correlation takes place in a central point at a level higher than host level. The Data collection modules gather Information in an agent-less manner from VMs using Virtual Machine Introspection (VMI). Main correlation module uses data mining technique to cluster VMs in two infected and uninfected groups. Clustering of infected VMs takes place regardless of type of malware that infected them. This is done based on general symptoms that is common among botnet intended malware. By identifying infected VMs, various botnets elusive techniques are inutiled.

*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

Cloud computing is a powerful definition for delivering on-demand hosted services in a multi-tenant environment. Cloud computing has some unique characteristics like being deployed on virtual infrastructure that apart from advantages, it opens some new venue of vulnerabilities against the cloud. As users and companies are moving their data to clouds, attackers are redirecting their attacks to cloud as well. Security problems are one of the largest obstacles in adopting cloud computing by companies. Cloud computing provides suitable infrastructure for both honest and malicious. As discussed in [7], abuse of cloud services is a prevalent threat to cloud computing. This matter raises the importance of detection and prevention of abuse uses from cloud services for Cloud Service Provider (CSP) at the right time. As mentioned, apart from detection, applying prevention techniques are important too. Most of researchers have ignored prevention capabilities in their proposed systems [27]. Cloud resources can be misused to conduct attacks toward targets either inside (i.e Fraudulent Resource Consumption (FRC) [20], [17], [19]) or outside of the cloud using cloud resources. A distinctive example of these kind of anomalistic usages from cloud services is creation of botnet.

Botnet is a group of infected computer systems that enables a controller to have control over the group. Examples of botnet functions are: DDoS attacks, spamming, search engine optimization poisoning, pay-per-click fraud, financial fraud, bitcoin mining and information stealing [6], [13]. Botmasters specifically benefit from the provided infrastructure as cloud computing offers them various possibilities to conduct their desired attacks [15]. Meanwhile, cyber criminals are moving toward creating more sophisticated botnets that can result in elusive malware escaping detection. 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 while getting larger in attack volume up to 400 Gbps in 2014 [9].

Conducting DDoS attacks is one of the intentions of botmasters and a prevalent threat to cloud computing [7]. DDoS attacks tend to change their attack vector during a single attack. This matter can help attack sources to remain hidden while increasing attack impact [10]. Diverse attacks are trendy techniques toward conducting large-scale attacks

like DDoS against cloud resources [25].

Design of network protocols forms a portion of the problem too since some of them were not designed with security in mind. Various recent DDoS attacks (amplification attack) which are on the rise, are result of existing vulnerabilities in protocols such as Network Time Protocol (NTP), Domain Name System (DNS) and Simple Service Discovery Protocol (SSDP) [10], [11]. Detection of these kind of attacks is not straightforward based on the conventional network traffic analysis. In the first eight months of 2014, DNS amplification attack increased by 183 percent [9].

Cloud-specific vulnerabilities form another portion of the problem as they are some sources of threats in the cloud [16]. Vulnerabilities that exist in one of the core technologies of cloud computing, is applicable to cloud too. Examples of the cloud core technologies are virtualization and cryptography. Hence, increasing security in virtual environment, increases security in cloud environment.

Issue such as image sharing between users and interest for homogeneity among cloud systems, pave the path for malicious users to distribute their infected image and malware in the cloud. So creating botnet in the cloud is easier compared to conventional environments. In the cloud (specifically in the case of image sharing), bots depend much less on victim's system software stack for exploitation.

To mitigate the system and network level anomalies in the cloud, various IDPS solutions are developed. Traditional IDS solutions are inefficient for cloud environments [12] [27]. In other word, Distributed and specific structure of cloud computing makes Implementing IDS solutions in the cloud to require various considerations.

In response to challenges above, the cloud-specific IDPS needs to 1) have a broad view over all nodes in a cloud region, 2) Monitor entities in a wide-spread and distributed manner to best mitigate problems like botnet, 3) The solution should be based on cloud core technologies to be cloud specific, 4) As cloud is large, IDS solutions should be efficient, optimised and effective.

Contribution. EyeCloud tackles the problem of detecting when infected VMs in a cloud form botnets to launch DDoS attacks. Our system adopts new approaches toward data gathering, information correlation and decision making steps. We proposes the main correlation system to be placed in an outer level of infrastructure, which is cloud controller. This helps the system to be more practical and have wide view over the cloud instances that satisfying the first requirement mentioned above. In the data gathering phase, Our approach takes advantage of virtualization as one of the cloud core technologies satisfying the third point mentioned in the requirement list above. By having various detectors in each host, our approach follows second requirement mentioned above. Looking at experiment results while adopting unsupervised algorithm of data mining fourth requirement is satisfied.

In this work we concentrate to extract a diverse set of system level symptoms as opposed to botnet behaviour (behaviour analysis) to detect entities involved in a botnet. This approach aims to detect the infected VMs. Searching for symptoms leads to generalization of anomalies. This helps detection rate increment while signature database size decreases.

Organization. The rest of this paper is organized as follows. Section 2 discusses the main existing approaches for cloud-botnet detection. 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 experimental results. Finally, in Section 5 we draw our conclusions, and discuss possible future research directions.

2. RELATED WORK

In this section, we discuss previous works done on Collaborative IDS and malware detection in the cloud.

Researchers in [25] proposed NIMBUS, a cloud-scale attack detection and mitigation approach. The work presents a design which gets benefit of rapid elasticity characteristics of cloud computing. It tries to resist against large-scale attacks by spreading the high volume work load of detection among various instances in the time of need.

In [24] researchers presented a collaborative IDS framework for the cloud. Their proposed design is a system that makes IDSs in different cloud regions to cooperate and exchange information together. The presented system detects coordinated attacks such as DDoS by correlating network traffics based on either source or destination address.

Researchers in [14], proposed a cooperative IDS to reduce impact of attacks like DDoS in the cloud. In this work IDSs in different regions exchange their alerts with each other. So Other IDSs can prevent from the same attack happening in other regions.

Researchers in [18] 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 infected 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 [21] 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 [28] presented a research work to detect computer worms in the cloud based on the spreading behaviour. The solution looks for inconsistencies in behaviour of machines. Presented system is

limited to only looking for two kinds of information. Start of a new process or a loaded module which is not listed in the predetermined white list or its presence is not normal are the focused inconsistencies.

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.

3. EYECLOUD DESIGN

In this section, we present our approach toward EyeCloud design, in particular monitoring, detection and reaction methods.

3.1 Monitoring method

Data collectors gather relevant information in wide-spread manner. In our design, we use concept of FVM for describing data gathering entities that was introduced in [29]. As depicted in Figure 1, we assign one FVM per VM to extract system level information from each VM's memory page at any given time. Each FVM is dedicated to one VM and looks for existence of all the required symptoms in the VM assigned to it. Assigning one FVM per VM as opposed to one FVM per symptom [29], has various benefits. Introspected VMs should trust only to 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. In time of any 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. Figure 1 depicts a high level design of EyeCloud with FVMs incorporated in it. FVMs collect data in wide-spread manner and transfer them to central correlation module. By locating EyeCloud and a typical cloud platform design together, cloud controllers can be hosting units for central correlation modules. The reason in because cloud controllers are outermost entity in cloud platform design.

Apart from method of data collection, determination of what is actually collected is vital. A sample list of symptoms can be defined by determining inconsistencies in an infected VM. We consider windows operating systems and we focus on system level inconsistencies. Based on [2], we prepared a table that determines common modifications that nine botnet intended malware cause to victim's system. Under each malware's name the year that malware was discovered is mentioned. By looking at Table 1, we conclude all the listed malware create registry entry. As a result, creating registry entry is a prevalent action among listed malware. Hence, Creation of registry value under a specific key would be a symptom as opposed to entry value. To reduce false positive, one should look for a set of diverse symptoms. So occurrence of number of the diverse symptoms among several machines would trigger security alert while putting suspicious machines in a different cluster.

Based on information depicted in Table 1 and given in [26] and [22], set of inconsistencies that we consider are: existence of hidden (unlinked) process, existence of hidden (unlinked) DLL, existence of critical terminated process, existence of abnormal loaded service in memory.

The main audit source location, is in particular each VM's memory. We use VMI technique to collect data from memory of target VM. VMI can help security applications by enabling access to system level information of guest VM. Using VM's memory as audit source has benefit of combating against hiding low-level local activities while identifying actual attacking entities.

Time of detect, is in non real-time manner as an infected machine is detected to be infected after the actual infection is occurred and correlation is done. non real-time manner detection leads to less power consumption.

3.2 Detection method

EyeCloud detection logic is based on anomaly detection. Under anomaly detection branch, we use data mining technique. Data mining is an analytic wide-spread technique that has high usage in anomaly detection. There are various branches under data mining technique to be used for anomaly detection. Clustering is one of the notable branches of data mining that is based on unsupervised learning technique. Unsupervised learning is finding structures out of unlabelled data sets. In EyeCloud design, we aim to cluster the VMs into two clusters based on collected data. One cluster would contain infected VMs while the other, contains clean entities. The novelty of the combination of these solutions is that based on general symptoms and regardless of type of malware, VMs are clustered. So by using clustering technique, EyeCloud identifies set of VMs that are infected by different malware.

Using general symptoms as input for anomaly detection algorithm has benefit of reducing signature based technique rules of misuse detection while increasing detection rate. The mentioned technique would increase likelihood of new malware to be detected because system would not need to be updated frequently as signature based systems.

3.3 Reaction method

IDPSs have two reaction methods, passive and active. Passive response does not take correction action against attacking entity. It takes actions such as notifying administrator or terminating session. While active response takes corrective actions such as policy or attacking entity modification.

EyeCloud has both passive and active response type. Looking at Figure 2, passive method is achieved when analysis and clustering module notifies to alarm and report module about it's decision. Administrators are required to analyse the obtained figures and results in this level. Active mode is achieved by the central correlation unit communicating with VMM of the physical machine holding the infected VM. By the command sent to VMM, Infected VM's can be deprived from using system resources by freezing their CPU cycles using the privileged domain.

3.4 Infrastructure

A collaborative IDPS consists of several IDPS where each consist of two main components, detection and correlation

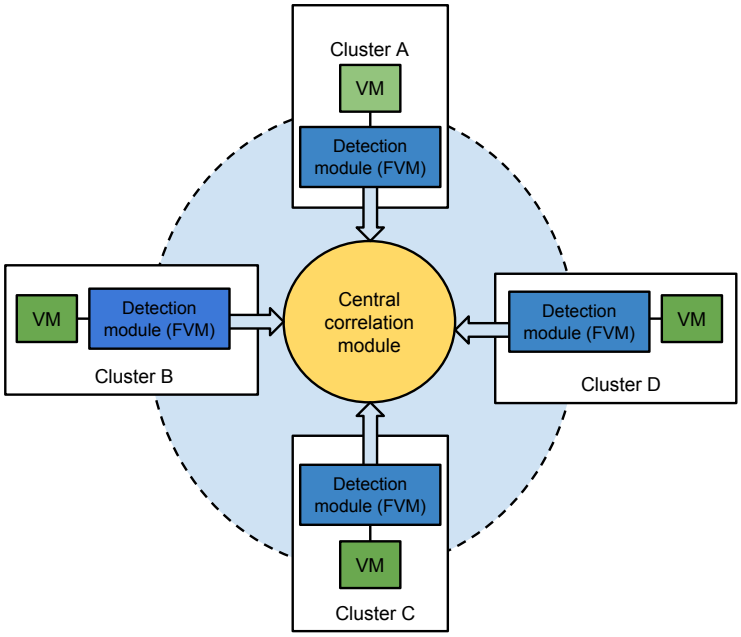

Figure 1: A wide-spread data collection with central correlation system design

modules. By getting idea from Software Defined Networking (SDN), we only hold detection module in the form of FVMs while having a main correlation unit. So EyeCloud can be categorized as a collaborative IDPS.

As depicted in Figure 2, collected data at each FVM gets in relevant data structure by symptom search module. This module only indicates existence or non existence of symptoms in VM. The data goes to central correlation located in cloud controller through VMM. Indeed by placing detective module in cloud controller level, analytic task gets offloaded from VMM. As FVMs only collect and pass the data, a separate database is not necessary at their level. A global database is placed to be in communication with central correlation unit. Administrators can retrieve previous stages of VMs in clustering groups.

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.

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.

In order for introspection to take place using LibVMI, introspecting VM should have access to kernel symbols of

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]

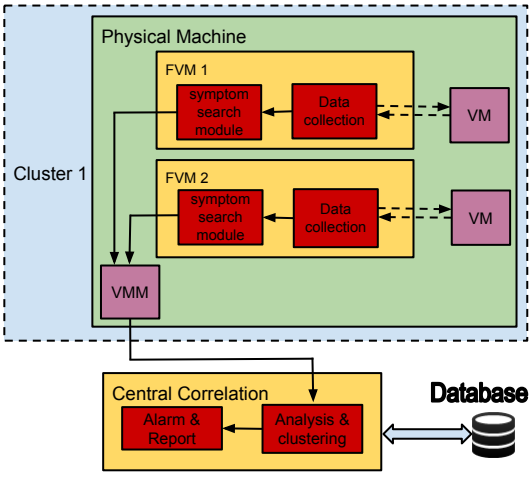

Figure 2: EyeCloud architecture

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 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 effective 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 VMs, the entire security of the cloud can be increased. Having wide-spread monitoring systems that are capable of actively monitor, log and report malicious activities and movements is essential. In this paper, we presented a framework 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 virtual entities.

There are two future directions which we would like to take. As prevention techniques are mostly overlooked, first we would like to move toward a framework that can predict susceptible virtual machines that are capable of getting infected. By identifying talented entities, preventive actions can be taken toward them.

Second we would like to enhance privacy of data obtained from guest VM using VMI. In the researches 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 VM. These information can be target of misuse by malicious insiders.

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] K. B. J. C. J. Ahmed Patel, Mona Taghavi. An intrusion detection and prevention system in cloud computing: A systematic review . *Journal of Network and Computer Applications* , pages 25 – 41, 2013.

[13] 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.

[14] Chi-Chun Lo and Chun-Chieh Huang and Ku, J. A Cooperative Intrusion Detection System Framework for Cloud Computing Networks. In *2010 39th International Conference on Parallel Processing Workshops (ICPPW)*, 2010.

[15] 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.

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

[17] 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.

[18] 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.

[19] 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.

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

[21] 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.

[22] Ligh, Michael Hale and Case, Andrew and Levy, Jamie and Walters, Aaron. *The Art of Memory Forensics: Detecting Malware and Threats in Windows, Linux, and Mac Memory*. 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] Man, NguyenDoan and Huh, Eui-Nam. A Collaborative Intrusion Detection System Framework for Cloud Computing. In *Proceedings of the International Conference on IT Convergence and Security 2011*, pages 91–109.

[25] Miao, Rui and Yu, Minlan and Jain, Navendu. NIMBUS: Cloud-scale Attack Detection and Mitigation. In *Proceedings of the 2014 ACM Conference on SIGCOMM*, SIGCOMM '14, 2014.

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

[27] A. Patel, M. Taghavi, K. Bakhtiyari, and J. C. JÃžnior. An intrusion detection and prevention system in cloud computing: A systematic review. *Journal of Network and Computer Applications*, 36(1):25 – 41, 2013.

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

[29] 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.