BotCloud Detection System

Master thesis
University of Turku
Department of Information Technology
Information Security and Cryptography
2015
Mohammad Reza Memarian

Supervisors:
Ville Leppänen
Mauro conti

UNIVERSITY OF TURKU
Department of Information Technology

MOHAMMAD REZA MEMARIAN: BotCloud Detection System

Master thesis, 51 p., 0 app. p.
Information Security and Cryptography
August 2015

Leveraging cloud services, companies and organizations can significantly improve their efficiency, as well as building novel business opportunities. A significant research effort has been put in protecting cloud tenants against external attacks. However, attacks that are originated from elastic, on-demand and legitimate cloud resources should still be considered seriously. The cloud-based botnet or botcloud is one of the prevalent cases of cloud resource misuses. Unfortunately, some of the cloud's essential characteristics enable criminals to form reliable and low cost botclouds in a short time. In this paper, we present a system that helps to detect distributed infected Virtual Machines (VMs) acting as elements of botclouds. Based on a set of botnet related system level symptoms, our system groups VMs. Grouping VMs helps to separate infected VMs from others and narrows down the target group under inspection. Our system takes advantages of Virtual Machine Introspection (VMI) and data mining techniques.

Keywords: Botcloud; cloud computing; misuse detection; virtual machine introspection.

Contents

List of Figures

List of Tables

Chapter 1

INTRODUCTION

Cloud computing is a powerful technology for delivering on-demand hosted services in a multi-tenant environment. It enables consumers to have access to a vast amount of computing resources at any desired time. Cloud computing provides suitable infrastructure, for both honest and malicious purposes. On demand self service resources and rapid elasticity are characteristics of cloud computing that create opportunities for cyber criminals to get advantage of cloud resources. As discussed in [2], abuse of cloud services is a prevalent threat to cloud computing. This matter raises the importance of detection and prevention of abuse uses of cloud services for Cloud Service Provider (CSP). A distinctive example of these kinds of anomalistic usages of cloud services is formation of botnet in the cloud.

Botnet is a group of infected computer systems that enables a controller to have control over the group. Examples of botnet functions are: DDoS attack, spamming, search engine optimization poisoning, pay-per-click fraud, financial fraud, bitcoin mining and information stealing [3], [4]. Botmasters specifically benefit from the cloud's provided infrastructure as cloud computing offers them various possibilities to conduct their desired attacks. In cloud computing, fairly cheap, scalable and reliable resources are always ready for use. So attackers have ready-to-use infrastructure for their malicious activities.

Forming a reliable botclould is an easy and low cost procedure while anonymity of criminals can be preserved [5].

Botnets are moving toward being more elusive which are able to escape detection. Botmasters integrate multiple backup forms of command and control to hide commanding servers. Particularly, DDoS attacks as a consequence of botnets tend to change their attack vectors during a single attack. As described in [6], botnets are becoming more resilient and responding faster to countermeasures, while getting larger in attack volume and frequency of occurrence [7].

There are various issues that pave the path for criminals to form a botcloud much easier compared to forming a botnet in conventional environments. Some of these issues are: trendiness of large scale diverse attacks [8], cloud specific vulnerabilities [9] and some of cloud essential characteristics [10]. Specific features of cloud computing such as image sharing and interest for homogeneity among cloud systems can further ease the formation of botclouds.

Main botnet detection methods are: honeypot and intrusion detection system (IDS). Using honeypot and applying intrusion detection system on each machine in the cloud are not feasible solutions [5] [11]. On the other hand, distributed and specific structure of cloud computing requires detection solutions to meet various cloud specific considerations. In order to detect cloud resource misuses, researchers proposed cloud specific detection systems while mostly overlooking prevention capability in their designed systems. Unfortunately, the solutions for detection of botnets in the cloud are not mature yet; in fact attacks using cloud resources are on rise.

In response to challenges above, a BotCloud Detection System (BCDS) needs to: 1)

have a broad view over all nodes in the cloud in a cloud-wide manner; 2) be based on cloud core technologies; 3) be reliable for large cloud environments while being effective.

Contribution. In this document we propose a botcloud detection system that tackles the problem of detecting when infected VMs in a cloud form botclouds. Our system adopts new approaches toward data gathering, information correlation and decision making steps. We propose the main correlation module to be placed in an outer level of cloud platform which is the cloud controller. This helps the system to be more practical and have a wider view over the cloud nodes. In data gathering phase, our approach takes advantage of virtualization as one of the cloud core technologies [9]. Our system has various detectors in each host that makes the design more reliable. Our result shows that based on the selected attributes, infected VMs that are potential security threats, can be separated from other VMs.

Organization. The rest of this document is organized as follows. Chapter 2 explains cloud computing concept, related threats and risk. Chapter 3 gives background information. Chapter 4 explains existing approaches for botcloud detection and malware analysis using VMI in the cloud. In Chapter 5 we introduce our novel design for detection of infected systems which can be elements of botclouds, while in Chapter 6 we discuss implementation details. In Chapter 7 we discuss and evaluate our results, and in Chapter 8 we explain our conclusion and future research direction.

Chapter 2

CLOUD COMPUTING, THREATS AND RISKS

Cloud computing is an advanced computing model which enables access to a pool of shared, configurable computing resources. Cloud computing provides various opportunities specifically for small to medium level enterprises (SMEs). In other word, cloud computing offers hosted services over the internet to the consumers. Depending on consumer needs, various costs can be saved by getting benefits from cloud computing.

Cloud computing enables service providers to offer variety of cloud based services to the customers. The cloud services vary from storage, networking and computation to applications. For example, a consumer can save his private data on the cloud storage services. In this way the consumer does not invest any more in storage devices. Also, the responsibility of providing high availability services will be offloaded to the service provider. Another example is a consumer which needs high computing resources only in some specific times. Instead of investing in computing resources, he can consume cloud resources whenever he needs the service.

Cloud computing enables consumers to use computing resources as utilities. So consumers pay only for the service that they used. There are different ways to calculate consumer's usages. Basically calculation of usage varies from provider to provider and depends on used service. In other word, consumers pay for the metered service and pay only for the exact usage. By using the word consumer, we refer to individual users in addition to enterprises.

Cloud computing has various advantages and benefits for consumers. Some of the advantages are in the following:

- Implementation speed reduces as customers do not need to invest in time for research, training and implementation matters related to new systems.
- Consumers can rapidly scale their service to meet their needs. This point addresses consumer's needs that arise based on unpredictable events.
- Consumers can have access to their anywhere using a standard internet connection. So basically access anywhere model can be reached.
- Consumers can have access to their anywhere using a standard internet connection. So basically access anywhere model can be reached.
- Consumers offload responsibilities such as software update, patches, licences and hardware maintenance to the service provider. So IT staff of companies can have more productivities by not getting engaged with the mentioned concerns.

Cloud computing has some powerful and specific characteristics which make the cloud very powerful. Based on [10], cloud computing has five essential characteristics. These characteristics are:

- On-demand self-service

- Broad network access
- Resource pooling
- Rapid elasticity
- Measured services

All mentioned characteristics are vital and specific to the cloud computing. Each of them has some benefits for the consumers. On-demand self-service allows consumers to provision resources as much as required. Broad network access means that cloud resources are available for access through several devices (Laptop, smart phones and etc.) and from various locations. Resource pooling refers to cloud service provider resources which are pooled. The resources serve various users in a multi-tenant environment. As addressed earlier, some of the cloud resources are storage, processing and network bandwidth. Rapid elasticity refers to the ability of consumer which can rapidly occupy and release the computing resources. Measured services refers to the fact that cloud services are measured based on some metrics. These metrics are either time or amount of usage or any other suitable metric.

From implementation point of view, there are four deployment models in addition to three service models for the cloud [10]. We discuss deployment and service models in the following parts.

2.1 Deployment models

Private cloud. Private cloud is a type of cloud infrastructure that is operated for only one organization. This type of infrastructure can either be operated by the using enterprise itself or a third party. The private cloud infrastructure can be hosted either inside

or outside the organization. In other word, an organization can request from a service provider for dedicated a infrastructure. So the specific consumer uses the dedicated infrastructure solely. This is the case that private cloud is operated by the third part outside of the company. However, companies can set up their own private cloud and operate it in the physical barriers of the company. So, private cloud model provide computing power to specific users by providing pool of physical resources.

The private cloud model is closer to conventional organization's networks than other deployment models. The private cloud deployment offer various kind of advantages. Consumers of private cloud architecture compared to other models (in particular public model) can benefit from advantages such as: higher security and privacy, cloud bursting, more efficiency and reliability.

Public cloud. Public cloud is referred to a kind of a cloud deployment model which is ready to use for all public consumers. Various consumers should use shared underlying infrastructure. there are variety of enterprises that offer public cloud services. Some of the most important ones are: amazon, rackspace and google. Basically the advantage of this deployment model is in multi-tenancy. Various consumers may use and share same resources. So service provider shares the cost among them. In addition to that, devices such as servers, run with more load. So servers resources do not get wasted as in the case of organizations. Normally in the organizations, servers and computing resources do not run with full load.

As shown in Figure 2.1, in public cloud deployment model various consumers use shared resources. Although multi-tenancy opens up opportunities, but it is a major threat toward security and privacy of users data.

Public cloud deployment model offers some advantages to consumers compared to

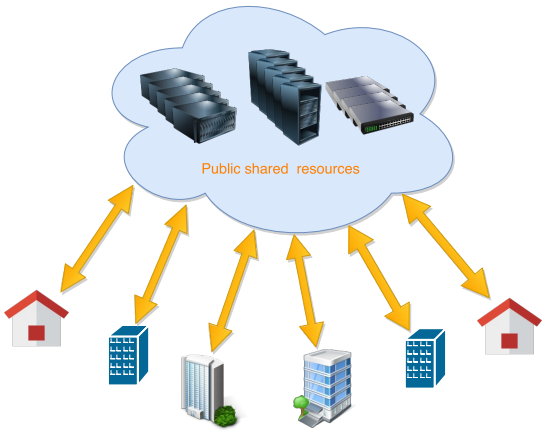

Figure 2.1: A high level public cloud architecture

other deployment models. The benefits of public cloud deployment models are: ultimate scalability and reliability; location independence and utility model metering and payment.

Hybrid cloud. Hybrid model is a kind of combination model between other deployment models (mostly public and private deployment models). Computing needs can move back and forth between public and private cloud of consumers. For example, an enterprise can have critical data and computation running on own private cloud while the less critical and private data and computation to take place on a reputable public cloud. Indeed, one of the advantages of private cloud deployment model which we addresses was cloud bursting. Cloud bursting has a high similarity with definition of hybrid cloud. Hybrid model benefits from characteristics of all other 3 deployment models while depending on various service providers. So there are not clear boundaries between service providers services in this model as private and public models.

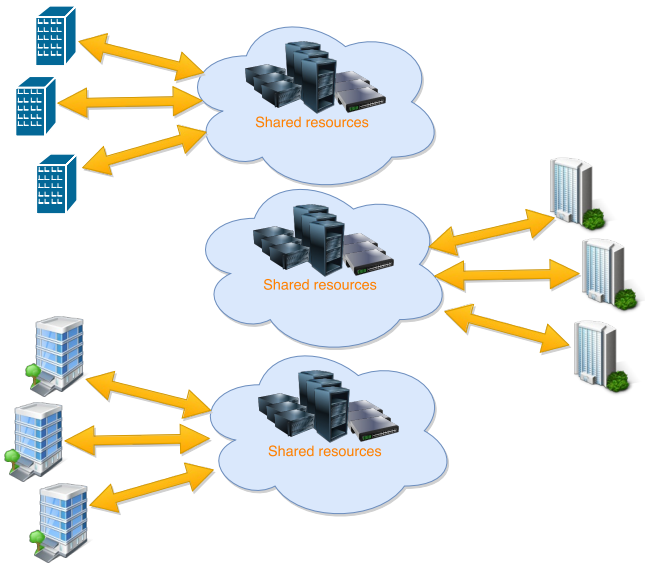

Figure 2.2: A high level community cloud architecture

As mentioned earlier, the core idea at back of the hybrid cloud deployment model is getting benefit of various deployment models. So a prevalent use case is when consumers use their own private cloud and in time of need, they consume the resources available in public or community cloud.

Community Cloud. Community cloud is a type of cloud deployment model that several organizations with same concern, policy and need share a cloud infrastructure. As a result, costs are shared among all the consumers. Community cloud as other deployment models (in particular private model) can either be deployed and hosted internally or externally. Also community cloud consumers can either control and maintain the service by themself or a third party can handle maintenance. The third party can either be a CSP that offers cloud services to other enterprises too or it can be a company that maintains an specific community cloud. As depicted in Figure 2.2, organizations with same concern and policies may share same cloud resources.

2.2 Service models

Software as a Service (SaaS). Consumer can use provider's application which is running on a cloud infrastructure. As mentioned earlier, consumers can access provider's application through various devices and from various locations. Basically, consumer need a device with a web browser or provider's user application installed on it, and an internet connection. Consumers using SaaS, do not manage or control the underlying infrastructure of the application such as networking and storage. Service provider should handle all the responsibilities of this kind. So, consumers do not have control over the core settings of the application.

Platform as a Service(PaaS). PaaS model is a very useful model mainly for developers. In this model, consumers develop and deploy their applications by using development tools which are provided by the service provider. This model is very suitable for enterprises which are not wishing to invest in development tools. Tools like programming languages, libraries, service and related licences are provided and supported by the service provider. As in the case of SaaS, Consumers do not have control over cloud's underlying infrastructure such as networking and storage. In PaaS, consumers have control over setting of their own developed applications deployed in the cloud.

Infrastructure as a Service (IaaS). This service model enables consumers to provision infrastructure related resources such as storage, network and computing resources. Basically, consumers have the capability to run any desirable software on the assigned systems. The full control of operating system of the assigned machines are with the consumer. As previous models, consumer do not have control over the underlying cloud infrastructure.

2.3 Threats associated with cloud computing

Cloud computing has transformed service computing and businesses. The way that cloud computing delivers computing, has opened various new business opportunities. Shifting to cloud computing is fast but at the same time it should be with caution. As there are various benefits in the cloud for consumers, more and more consumers are moving their data, processes and computations to the cloud. At the same time malicious parties are specifically focusing to the cloud too. So this movement of attacks and attentions toward the cloud, reveal several threats that threaten security and privacy of users data.

Some of threats toward the cloud either arise because of cloud's essential characteristics (mentioned in previous sections) or cloud's core technologies. Basically cloud computing is comprised of several core technologies. Three cloud main technologies are: cryptography, virtualization and web applications and services [9]. To a large extend, cloud computing depends mainly on the mentioned technologies. Specifically, some of service models which will be discussed later in this Chapter, are deliverable through web applications and services. On the other hand ,the whole cloud is built on virtualization technology, particularly its infrastructure service. Third technology which is cryptography, is the solution to many cloud computing security and privacy problems.

A threat can be related to the cloud if, it's root cause is in one of the cloud's characteristics mentioned earlier or the threat is prevalent in one of the cloud's core technologies [9]. In other word, a vulnerability that threaten security of the virtualization, definitely threaten security of the cloud too.

Based on [2], nine top threats to the cloud which should be seriously addressed specifically by researchers are in the following:

- Data breaches: It is very important that sensitive data of any information remain se-

cret and not to be known by non authorized parties. Multi-tenancy (specifically in public cloud deployment model) opens a venue for this threat to take place. Vulnerabilities in shared applications can put data of all consumers of that specific shared application in danger. One of the prevalent solutions to mitigate against this threat is using encryption technologies. On the other hand, encryption technologies are also prone to attacks.

- Data loss: This point refers to actions that need to be taken by both service user and provider to prevent loss of data. Examples of these actions are back up and disaster recovery policies. Data in the cloud can be lost due to reasons other than malicious activities. An example of those reasons can be natural disasters. Generally, consumers should be responsible for having backup of their own data while service providers should be responsible for taking preventive actions to reduce probable losses.

- Account or service hijacking: This point is not cloud specific as it happens in conventional environments too. The difference is by gaining access to specific cloud accounts, attacker may have access to vast amount of private data and infrastructure.

- Insecure APIs: Cloud's consumers use from several cloud services through APIs. These APIs are provided to them by service providers. Consumers use APIs to manage their resources. APIs should be protected no to be misused by malicious users to access to critical data of users.

- Denial of service: As users of cloud resources primarily have access to services through standard internet connection, availability of the services is vital. So, any obstacle that prevents network connections between consumers and online resources, prevents users to access their data through other means like local access in conventional environments. DDoS attack toward CSPs can prevent consumers to have

access to their data. This matter can cause financial losses to both CSPs and consumers.

- Malicious insiders: System administrators and CSP's employees that have access to cloud's resources may misuse their privileged access. Adequate access control methods should be applied by CSPs to prevent probable misuses.

- Abuse of cloud service: Cloud computing offers a great amount of on demand and scalable resources to all kind of consumers. In some cases, malicious consumers can misuse the vast amount of resources for variety of malicious activities. It is vital for CSPs to detect misuses from their resources and prevent them from happening.

- insufficient due diligence: This point is a less technical threat compared to other threats. It refers to lack of consumer's knowledge about the cloud environment in which they are willing to move to. The environment refers to CSP, application or a cloud service.

- Shared technology issues: CSPs deliver their services through sharing underlying resources. As mentioned earlier, vulnerabilities in shared resources can threaten security of all consumers using the same shared resource.

2.4 Risk associated with cloud computing

While some consumers move their data to the cloud, other consumers are in doubt. The doubt arises from risks associated with cloud services. That is the reason that cloud risk assessment studies are developing. Based on [12], cloud computing offers various opportunities (specifically for small to Medium Enterprises) and advantages. Opportunities that are discussed here are mostly network and information security opportunities. Some of these opportunities are: Backups; Certification and compliance; standard formats and interfaces and physical security. There are some risks associated with these opportunities.

Some of these risks are consequences of the cloud related threats. In [12], eleven risks are mentioned for the cloud. The risks associated with cloud are in the following:

- Software security vulnerabilities
- Network attacks
- Social engineering attacks
- Management GUI and API compromise
- Device theft/loss
- Physical hazards
- Overloads
- Unexpected costs
- Vendor lock-in
- Administrator or legal outage
- Foreign jurisdiction issues

Chapter 3

PRELIMINARY INFORMATION

In this Chapter, we give some preliminary information to make understanding of rest of this document smoother. In particular, we explain preliminary knowledge about cloud computing (Section 2.1), virtualization (Section 2.2), botcloud (Section 2.3), VMM (Section 2.4) and VMI (Section 2.5).

3.1 Virtualization

Virtualization is a software technology which enables multiple operating systems and applications to run on a same server (host) at the same time. Virtualization is an effective way to reduce IT related expenses and to increase efficiency. It is effective for both large and small enterprises. Virtualization is some sense is the heart of cloud computing. Virtualization refers to virtual forms of various physical devices. Virtualization applies to computer hardware, storage devices, operating system, applications and networks too. Virtual computer systems are called the Virtual Machine (VM). Each VM consist an operating system with various applications. VMs are independent from each other. Using virtualizaiton has various benefits. Some examples of benefits of virtualization consumption are in the following:

- Reduction of operational costs dramatically
- Increase productivity and efficiency
- High availability for applications and services through provisioning
- Enable a central management through simple systems

A layer which is placed between VMs and the host called hypervisor is responsible for operation of VMs and allocation of physical resources to them. VMM or hypervisor is a software that runs VMs and manages resource allocation to them. The system on which VMM runs is called host. VMMs are either bare-metal or hosted. In case of bare-metal, VMM runs directly on the underlying hardware. In case of hosted, VMM operates on an operating system. There are a variety of commercial and open-source versions of VMMs.

One of the open-source VMMs that we also used for our implementation is Xen. Xen [13] is an open source and widely used VMM. 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. VMs running on Xen should run either on Hardware Virtualization (HVM) or Para Virtualization (PV) mode. 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 a virtual platform.

3.2 Botnet and BotCloud

Botnet refers to a group of computer systems that are infected with a piece of malicious software. The malware allows the controller (bot master) to take control of various parts of infected systems and take specific arbitrary action. Victims are not aware of the actions which their system take in favour of the bot master. Infected machines are called zom-

bie. The program that is responsible for infecting victim's system is called bot. As bots tend to remain hidden, rootkits are used for their activities. There are various methods that bots can infect victim machines. Spreading pirated software is one of the prevalent ways. Malware authors try to hide their malicious software inside pirated software. At the time which victims install the pirated software, bots leave their effects on the systems too. Another way to spread the malicious bot software is through Email and spamming.

At the time of installation, Bot software installs a backdoor on the system of the victim. The backdoor enables botmaster to communicate with victim's machine. Through the backdoor, botmaster is able to send commands to the zombies and even send updates and newer versions of the malicious code. As mentioned earlier, some of activities that botnets are able to do are: DDoS attack, spamming, search engine optimization poisoning, pay-per-click fraud, financial fraud, bitcoin mining and information stealing [3].

Botclouds are botnets that are formed based on a CSP's resources. Botcloud misuses cloud's legitimate resources to conduct malicious activities. Some of the cloud's characteristics allow attackers to form botclouds much easier than forming botnets in conventional environments. On the other hand, CSPs allow consumers to have access to a vast amount of resources with very low prices. So botmasters tend to form botclouds compared to forming botnets.

3.3 Virtual Machine Introspection (VMI)

VMI is a technique used in virtualization to inspect memory space of a VM in real time. Inspection takes place from a secure point which can be another VM (i.e., Dom0) outside of the inspected VM. In this way, there is no need for installation of local agents on each introspected machine. Agentless manner has the benefit of increment in security. In agent

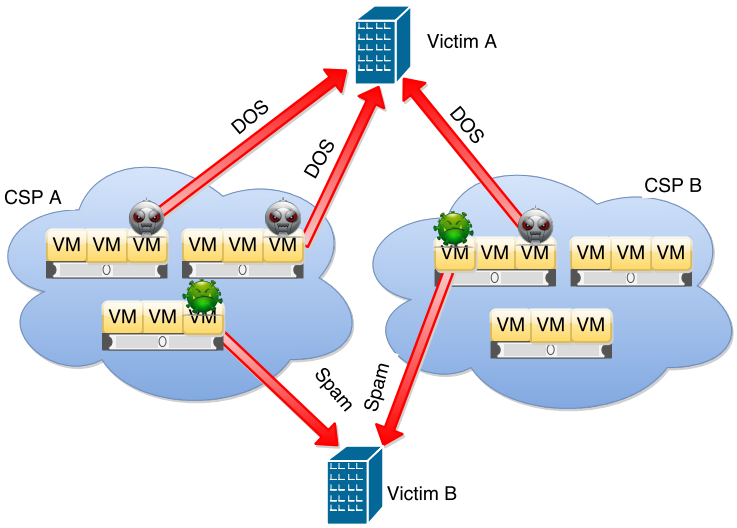

Figure 3.1: A botcloud figure

oriented systems, malicious code might disable the security software or the related agent in order to escape detection. Since VMI takes place through an agent less procedure, this threat is mitigated. VMI technique is applicable to security monitoring systems which operate within virtual environments.

To conduct introspection, an appropriate Application Programming Interface (API) to use is LibVMI [14]. LibVMI which currently is usable with XEN and KVM, enables introspecting VM to read from and write to memory of introspected VM. Introspection through VMI is done in an agent-less manner. LibVMI is well integrated with Volatility [15]. 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. It happens by adding the LibVMI address space to the address space list of Volatility.

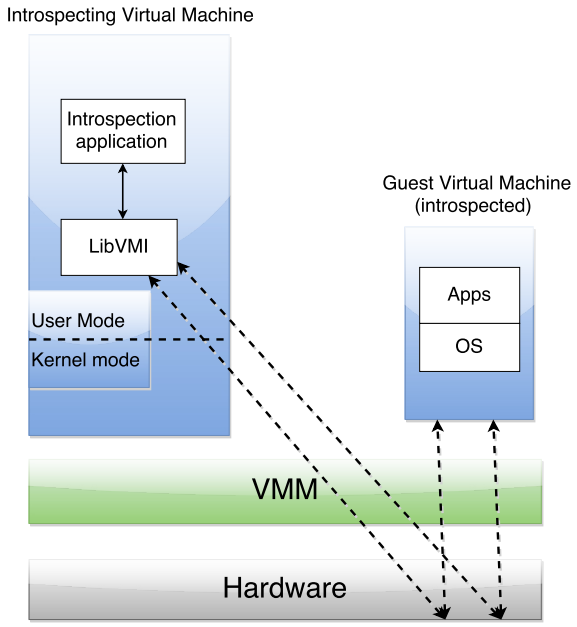

Figure 3.2: Using introspection to obtain internal Vm's data

3.4 Data mining and cluster analysis

Data mining is a wide-spread analytic technique. Indeed, it is one of the interdisciplinary branches of computer science. It is about finding some structure out of large data sets that would make sense. The goal is to find meaningful data for further analysis processes. That is why that data mining is very useful in handling big data problems. Nowadays, data mining techniques are widely used in various type of application areas. Data mining uses statistics.

There are various problem areas in data mining. Examples of related problems are: regression, classification, clustering, association rules and etc. There are various algorithms designed to solve each problem. The right choice of problem and algorithm depends totally on the uses' goals. So there is no single answer or best choice. There are common steps in the process of data mining while algorithms are different. There are two majors learning modes in data mining. Supervised and unsupervised learning modes.

In supervised learning processes, machine learns from the data set provided by the user. In other word, user provides machine with a sample data set. So the machine learns from the provided data set. An example of a problem that lies in supervised learning section is classification. In the classification, the machine learns to predict to which category a data set belongs too. The user provides a data set and the machine learns the characteristics of each category. At the next level when a new data entry is passed to the machine, the machine can predict to which category the new data point belongs too.

Unsupervised learning is finding structures out of unlabelled data sets. So basically the machine will not learn the structure of the data set based on a previous data set. An example of unsupervised learning problem is clustering. This technique is a very useful branch of machine learning that categorizes alike elements in groups. A drawback of this

technique is that the number of groups is user defined. So user has to have basic knowledge about the data. There are evaluation techniques that can help user to better choose number of clusters. We discuss evaluation techniques in Section 6.

Chapter 4

RELATED WORK

In this Chapter, we discuss previous works done on botcloud detection and malware detection in the cloud using VMI.

4.1 Botcloud detection

As mentioned in Chapter 3, botclouds are different from conventional botnets in various ways. So detection of them requires some considerations. That is why we mostly focus on previous works done on detection of botclouds instead of botnets in conventional environments.

Kebande and Venter [16] presented an approach to detec botnet in the cloud environment using Artificial Immune System (AIS). This mechanism uses a negative selection algorithm. Through this proposed process, authors clain to be able to detect the probability of botnet infection based on the process of independent Poisson process and detection based on negative selection. Researchers in [17] proposed the botcloud source detection approach which aims to detect abnormal VMs behaviour. The proposed detection approach is based on principal component analysis and focuses on behaviours that support DDoS flooding attacks.

Researchers in [18] designed a passive and an active detection module in VMM to look for information in VMs using VMI and profiling bots. Their work mainly focuses on functions in one host and detects infected machines, based on trained node about bot behaviours. They make the botnet profile based on just the API calls done by the applications. Francois et al., in [19] address botnets that rely on peer to peer networks for communication. They propose a distributed framework that leverages a host dependency model and an adapted PageRank algorithm.

In [20], researchers present an approach that enables a source-based detection of UDP-flood DDoS attacks. Their system's detection logic is based on a distributed system behaviour analysis. As [17], In this work the detection is based on a principal component analysis.

4.2 Malware detection in the cloud specifically using VMI

Garfinkel and Rosenblum in [21] proposed use of VMI as an enabler for intrusion detection tools. They correctly addressed the problem that IDSs must reside on VMs to have an excellent view of interior of systems. In that way, IDS can be target of attack by the malicious software that infects the system. Furthermore they presented their detection system equipped with VMI. Researchers in [22], propose a multipartition, multichunk ensemble classifier. The multipartition, multichunk ensemble technique helps to reduce classification error.

Watson in [23] presented a distributed detection system design that detects spread of malware in multi-server cloud. This work presents a method of detection based on a

distributed network of detection agents. This work takes place in order to coordinate detection across the whole cloud. This would provide a mechanism for locking down vulnerable VMs in the event that malware is detected in the cloud. Beidermann and Katzenbeisser in [24] presented a research work to detect computer worms in the cloud based on the spreading behaviour. Their solution looks for inconsistencies in behaviour of machines. The claim that their system is able to detect unknown and new appearing computer worms. As mentioned, their proposed system detects spread ing worms based on spreading behaviour.

Harrison et al., in [25] presented a framework for malware detection in the cloud by using symptoms. They present a method of detecting malware by identifying the symptoms of malicious behaviour as opposed to looking for the malware itself. Researchers in [26], presented tiny VMs called Forensic Virtual Machines (FVMs) to gather information via VMI for detection of malicious activities. Each FVM searches for existence of a specific symptom of malicious activity in a VM. After inspection, the FVM chooses a random VM and starts inspecting the newly chosen VM.

Chapter 5

SOLUTION

In this Chapter, we present our approach toward design if system. In particular we focus on monitoring (Section 4.1), detection (Section 4.2) and reaction (Section 4.3) methods in addition to infrastructure architecture (Section 4.4).

5.1 Monitoring method

The method of monitoring in anomaly detection systems is determinative. The true choice of data collection method can increase reliability of the system while rate of false positive would decrease. In cloud environment, data collectors should be able to gather relevant information in wide-spread manner. In our design, we use the concept of the FVM for describing data gathering elements [26]. FVMs are small operating system that are capable of accomplishing specific and small tasks.

As depicted in Figure 1, we assign one FVM per VM to extract system level information from each VM's memory pages. The act of information extraction takes place at any desired time using VMI. The time is user specific. Based on the user's requirements, FVMs can gather information at arbitrary times. Examples of user's requirements are: system performance; sensitivity of the data; number of symptoms. There should be a

balance between frequency of extraction of information and system performance. Users should note this point that VMI interrupts introspected VM while extracting information. So the act of introspection should not be done so frequently.

In our design, each FVM is dedicated to one VM and looks for existence of all the required symptoms in the VM assigned to it. So each FVM receives a list of symptoms that are required to be checked. FVMs extract the information based on the received list until the receive a new list. User defines list of symptoms at the user's console. So from the main correlation center, a single list gets distributed to FVMs.

Basically, introspection should takes place from a point, out of an introspected VM. The VM that does introspection should be able to access to the memory of introsped VM. In other word, introspecting VM should have adequate access for this task. In virtualization, VMM is the first choice for this task as it has adequate access rights. However, VMM can be single point of failure too. So in this case, adequate access should be given to other VMs in order to be able to do introspection. As trend of system design is moving toward distributed systems, so we considered this trend in our system design too.

We considered to have various introspecting VMs. In other word, we have several data collectors that each collector introspects one VM at any moment. Assigning one FVM per VM as opposed to one FVM per symptom [26], has various benefits. Introspected VM should only trust to one FVM, which results in monitored VM's Trusted Computing Base (TCB) reduction. In addition, the symptoms are checked (frequently) all at once; this differs from state of the art approach considering only random checks. FVMs as any VM are not attack resilient. They might get infected too. In case of a FVM infection, other monitored machines can remain immune.

One matter we noticed is about deploying our system in large scale. In this case, overhead and scalability issues arise. In our design, number of FVMs has direct relationship with number of VMs. In start of the art design, number of FVMs has direct relationship with number of symptoms.

There are two scenarios that are considerable. First scenario is, set of servers which have high number of VMs running on them (i.e., more than 50 VMs). So to the number of VMs, there exists FVM. There might be be scalability overhead for the host running all the VMs. Depending on the number of symptoms, start of the art design might have less overhead. Second scenario can be set of servers which run a low number of VMs (i.e., around 20 VMs). In the second scenario, our design is expected to have performance in term of overhead for the host. Deep discussion about these issues are beyond the scope of this document, and we will consider them as future work.

Figure 1 depicts a high level design of our system with FVMs incorporated in it. FVMs collect data and transfer them to central correlation module. By comparing our system design and a typical cloud platform design, cloud controllers can be hosting units for central correlation modules. The reason is because the cloud controller is the outermost entity in a cloud platform design. So, in order to discover distributed attacks on VMs, FVMs do not have to participate in event correlation. This helps to make FVMs lighter.

Apart from the method that is used for data collection, determination of what is actually collected (attribute) is vital. In other word, choice of attribute directly affects the result of analysis. By correctly choosing effective attributes, rate of false positive can reduce while accuracy increases. A 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.

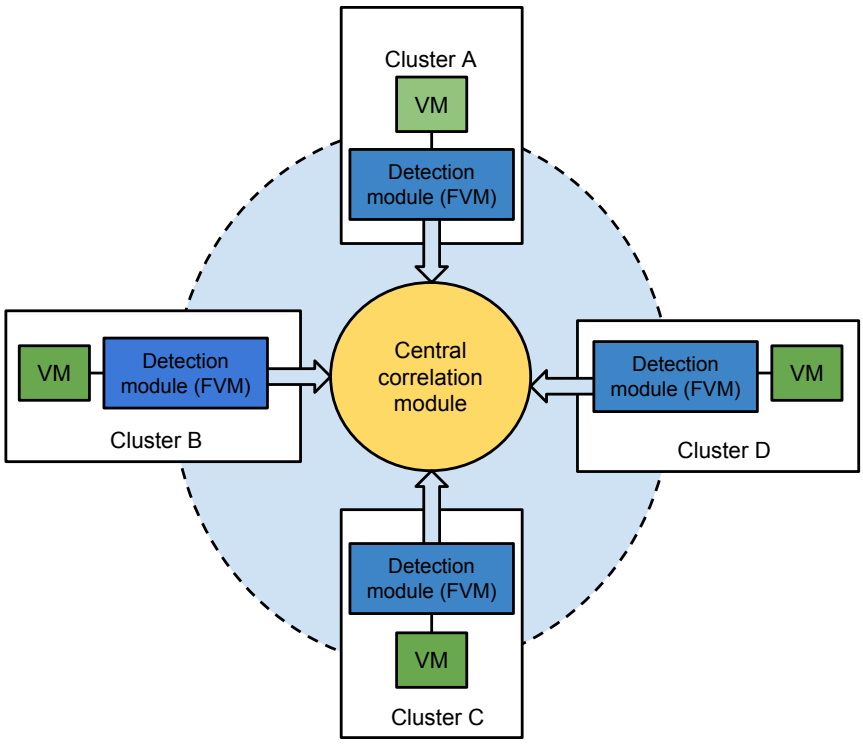

Figure 5.1: A wide-spread data collection with central correlation

Based on [1], we prepared Table 5.1 which depicts common modifications that nine botnet related malwares cause to victim's system. Under each malware's name, the year that malware was discovered is mentioned. By looking at Table 5.1, we conclude all the listed malwares create registry entries. As a result, creating registry entry is a prevalent action among listed malwares. Creating registry key can be done by honest processes too. Hence, creation of registry value under a specific key would be a symptom as opposed to the registry entry value. To reduce false positive rate, one should look for a set of diverse symptoms. So occurrence of number of diverse symptoms among several machines would trigger security alert while revealing differences among VMs. This difference can separate malicious VMs from the crowd of working VMs.

Based on [27], [28], [29], [30] and Table 5.1, a set of inconsistencies that we consider

is the following;

1) **Existence of hidden processes**. Normally, when processes start activities in operating system environment, they are attached to a linked list. Processes appear with a name and Process Identification (PID). Malicious processes may appear with inappropriate process's names in the process list. A technique to detect malicious software is to compare the name of a process against a white list of processes. Advancements in malicious software technologies allow malware writers to make their malicious software be able to scape detection in some cases. So detecting systems should be look from other pieces of information regarding this matter and not just a simple work of looking for process names.

Existence of an abnormal process in the process list is rare by advanced malware as their existence will be revealed easily and fast. There is an alternative option for malware authors to inject the malicious code into legitimate processes. In this way, the malicious process appears as if it in a honest process. So as mentioned, relying solely on the name of running processes in the process list, is not a potent method toward detection of malicious software. Second alternative for rootkits, is to unlink the malicious process from the process list's linked list causing the malicious process not to be present in the list of running processes. A process usually is not unlinked from the linked list of processes, unless there would be possibly malicious activities at the back of it.

It is possible to detect hidden processes by comparing various sources of process listing. A malicious process might be able to unlink and hide from various source of process lists but not from all of them.

2) **Shutting down security mechanisms - Firewall**. Security mechanisms can be target of attack by malicious software. In some cases, malicious software writers choose the

social engineering method for this kind of attack. Their software simply ask from users to click on a button that disables security mechanisms of the system for legitimate looking activity. This is a simple method while effective. In case of Windows operating systems, one of the security mechanisms that can be target of attacks is Windows firewall. Windows firewall plays an important role blocking various kind of attacks. Its absence can make the system vulnerable. Windows firewall can be disabled for legitimate purposes too (i.e., test of an application), but appearance of this symptom beside other symptoms can be a potentially malicious act.

3) **Inappropriate commands in history of the command shell**. Using manual commands is one of the prevalent ways to install a service on a system [30]. So in case the adversary has direct access to the shell, some artifacts might remain in the memory of the system in the shell. As mentioned, Adversary may have direct access to the shell of a system and run some commands directly through the shell.

4) **Existence of hidden Dynamic-Link Library (DLL)**. A running process loads number of DLLs. Loading an inappropriate DLL by normally an inappropriate process can reveal some information about intend of the malicious code. There are various ways to hide DLLs. One way is to unlink the DLL from the Process Environment Block (PEB) linked lists. If DLLs unlink themselves from PEB link list, some information would remain in Virtual Address Descriptor (VAD). So, existence of a hidden DLL as similar to hidden process is another technique that can be used by malicious codes to escape detection. There is possibility that the DLL is injected into other processes. This possibly requires a new view toward malicious DLL detection.

5) **Shutting down security mechanisms - Windows defender**. As we focus on Windows operating system, Windows defender is one of the vital services that should be

checked against getting disabled specifically in Windows 7. The Windows defender, is one of the first lines of defence against unwanted software in various Windows operating systems environment. Windows defender helps to combat malware. On the other hand, some users may decide to shut down this feature of Windows operating system and consume their own desirable security product.

6) **Shutting down security mechanisms - Automatic update**. Updates and patches are necessary for systems to remain secure. Systems which are not updated are vulnerable to the found related vulnerabilities in the wild. So it is one of the important tasks of any administrator to make sure all the system are updated. Automatic updates service may be target of attack in order to be shut down.

One of the primary matters to address in order to more clarify the discussion is to identify the audit source location. In our design, the main audit source location is each VM's memory. Using VM's memory in conjunction with introspection has two advantages. First advantage is that data collection takes place on live and running memory. This helps to speed of the process of data collection while being more effective. Second advantage is revealing of hiding and low-level local activities while identifying actual attacking entities.

In detection systems (i.e., intrusion detection system), time of detection is one of the important parameters. Time of detection is either in real-time or in non real-time manner. In real-time detection, malicious software or attackers are identified while the system in under monitoring for existance of possible malicious software. A non-real time detection system, analyses gathered data with some delay. Time of detection for our system is real-time manner. An infected machine is detected to be infected by the correlation element after an actual infection has occurred. The actual infection may not necessarily result as

an immediate damage.

As our system detects malicious VMs based on system symptoms as opposed to behavioural analysis, it can prevent further damages to systems and networks. So it may happen that a system is infected by a bot, but the malicious code does not immediately start to take offensive activities. Our system would detect those system in order to prevent them from taking arbitrary activities.

5.2 Detection method

Our system's detection logic is based on clustering analysis technique. In our design, we aim to group a set of VMs into subsets such that all elements in each subset group are more similar in comparison to elements in the other group. One group would contain infected VMs while the other would contain clean elements as grouping occurs based on inconsistencies determined earlier. The novelty of the combination of all the solutions is that based on general symptoms and regardless of type of malwares, VMs are grouped. So by using a clustering technique, it would be possible to identify a set of VMs that are infected by different malwares.

Using general symptoms as attributes for clustering 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. On the other hand, there might be false positives too. In our method, likelihood of detection and rate of false positive, heavily depend on the symptoms which are chosen to be checked by detection modules.

5.3 Reaction method

There are two reaction methods that are applicable to detection systems. Reaction methods are: passive and active. A passive response does not take correction action against attacking entities. It takes actions such as notifying an administrator. While active response takes corrective actions such as modification of attacking entity.

Our system has both passive and active response types. Looking at Figure 4.2, passive method is achieved when analysis and clustering module notifies the alarm and report module about its decision. Administrators are required to analyse the obtained figures and results in this level. Active mode is achieved when the central correlation unit communicates with the VMM of the physical machine holding the infected VM. Central correlation unit can issue a command to the VMM to freeze infected VM's CPU cycle. As a result infected VM can not further process and should go under inspection.

5.4 Infrastructure

Basically, detection system's structure is divided into two subgroups. First subgroup is collaborative and second subgroup is individual. A collaborative detection system consists of several detection elements where each consists of two main components: detection and correlation modules. A collaborative system then has three subgroups which are: central, hierarchical and fully distributed. Each type of collaborative detection has a description which is in the following:

- Central: In this model, there exist various local detection elements which can detect and produce alert locally. Then each detective element send the produced results to

a central server. The central server acts as analyser and correlation handler. In this model, the management is central and accurate decision can be made based on data obtained from all detective sources. A point that is noticeable, is that the central correlation unit should handle large amount of data. So the design should be as scalable as possible to tackle possible problems.

- Hierarchical: In this design, the whole system is divided in smaller subgroups. Subgroups are formed based on some features such as: software platforms and administrative control. At the lowest part, there are various detection element which do not have correlation capabilities. Detective modules pass their data to higher detective elements. At the second stage, detective elements have collocation capability. They correlate their own data and the data received from lower stage. Drawback of this design is that the highest level of detection and correlation has an abstract view toward the lowest parts. This reduces capability of the highest level to have a wide view over the whole system.

- Fully distributed: In this design, the management control is distributed. In other word, there is no central coordinator. Each detection element has its own correlation element too. This design has the advantage of high scalability as each detective element should not have information of the network topology. The drawback of this system is that the accuracy of decisions might be reduced as all the information from all elements are not available at the same time.

By getting idea from Software Defined Networking (SDN) and following central design of collaborative category, we only hold detection modules at the host level. Detection modules are in the form of FVMs while having a single main correlation unit at the higher level. So the our system can be categorized as a collaborative detection system with central design.

As depicted in Figure 4.2, the data collected by each FVM gets in a relevant data structure by symptom search modules. This module only indicates existence or non-existence of symptoms in VMs. The data goes to the central correlation module located in the cloud controller through the VMM. Indeed by placing a correlation module in the cloud controller level, analytic task gets offloaded from the VMM.

As FVMs only collect and pass the data, a separate database is not necessary at their level. A global database is considered in order to communicate with the central correlation unit. Administrators can retrieve different stages of clustering results from the central database.

As depicted in Figure 4.2, four main components are in our system design. Each component is described shortly in the following:

- Data collection: Information from VM's memory is collected through data collection component. Data collection takes place based on the symptom list. Data collection module should retrieve the list from central server.

- Symptom search: Collected information should be analysed to determine status of a symptom. If the the symptom exists, then value 1 is returned. If the symptom does not exist, then value 0 is returned. For each VM, a list of 1s and 0s will be created.

- Analysis and clustering: This component analysis the data set using K-Means algorithm of data mining. Based on information available in dataset about VMs, each VM is assigned to one group (cluster). Each VM is assigned to the group which is more similar to the elements of the group compared to the other group.

- Alarm and report: This component makes proper reports based on data received from analyse and clustering component. It will send the reports to the main database.

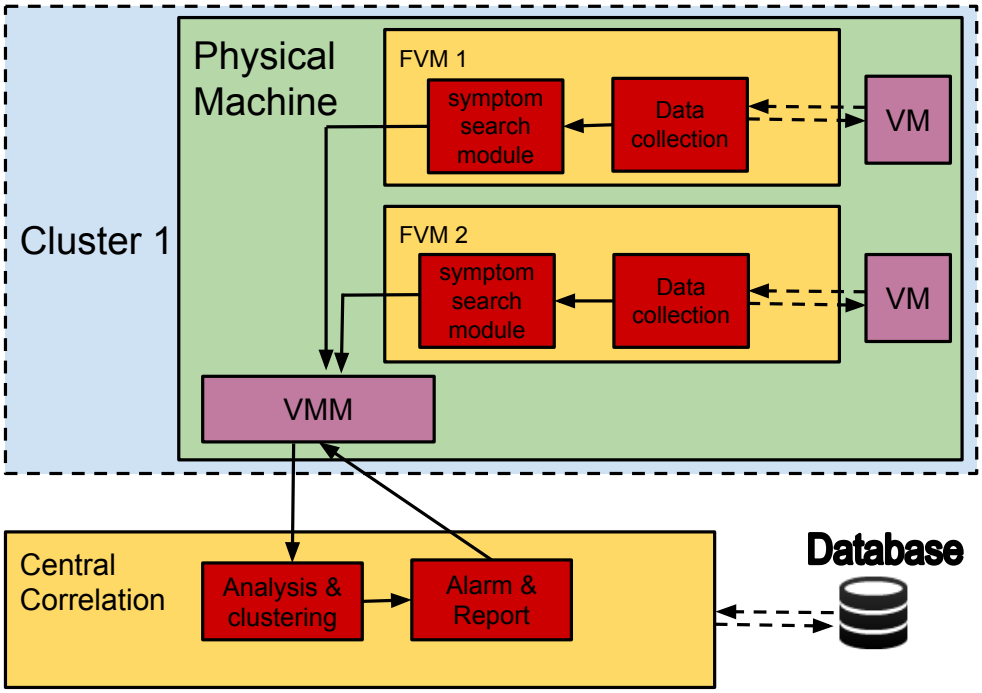

Figure 5.2: EyeCloud architecture

Users can later retrieve the previous reports from the main database. This component also send the proper order to the VMM of the server which holds infected VM. Based on the order, related VMM can freeze the infected VM's CPU cycle.

System modification	Rustock.B (2006)	Virut (2007)	Koobface (2008)	Tidserv (2008)	Qakbot (2009)	Pilleuz (2009)	Zbot (2010)	Carberp.B (2014)
File created (under certain directories)	✓		✓	✓	✓	✓	✓	✓
File Deleted				✓		✓		
File Modified		✓		✓	✓	✓		
Registry entry created (under certain keys)	✓	✓	✓	✓	✓	✓	✓	✓
Kernel modification	✓							
Code injection into process/DLL/App	✓	✓	✓	✓	✓	✓		

Table 5.1: Botnet intended malware functionalities based on [1]

Chapter 6

IMPLEMENTATION AND EXPERIMENT

In this Chapter, we describe our implementation and experiment. In particular, in the first part (Section 5.1) we describe implementation details. In the second part (Section 5.2) we explain our experiment details.

6.1 Implementation

Our proposed solution consists of integration various ideas and technologies. As for testing reliability of the idea which this research is built on it, we set up a test environment. Cloud's environments are very complex and large scale computing environments. Infrastructures used in cloud computing are strong and large. However there are ways to avoid large scale deployment of cloud infrastructure for testing the ideas and developments.

In out implementation, we mainly focus on depicting the possibility of required data collection with desired tools and libraries in order to provide the collected data to clustering algorithm. Furthermore to show that based on the chosen attributes, it is feasible to separate infected VMs which are possible threats from others.

The hardware that we used was a suitable choice for our test. Our focus was not to measure scalability of the solution but reliability of the solution. If scalability was our choice, we had to implement the solution on a much stronger hardware which could run all the test VMs at once. In our experiment, we did not run all the VMs together at once. Only the VMs that were required at any point of time for data collection were up and running.

As the hardware, we used a Dell laptop model Latitude E5440 that consists 8 GB of RAM and an intel core i5 processor. On top of the hardware, we set up a virtualized environment. We used Xen version 4.4 as the VMM. Before setting up Xen, we needed to install an operating system which operates as the Dom0 operating system. Based on experience, Ubuntu 14.04.2 LTS (Linux version 3.16.0-30-generic) was our choice for Dom0. So we set up a Xen 4.4 after we installed an ubuntu 14.04.2. Then we created several VMs running on HVM mode with the Windows 7 professional 32-bit as the operating system. Apart from Dom0, other guest VMs are called DomU.

We assigned 4 GB of RAM to the Dom0 VM. Rest of the remaining capacity of the hardware was taken by guests VMs that were operating. We assigned 1 core CPU to every VM on our hardware.

After setting up virtualization environment, we configured the required introspection tools. For the introspection tool, we used the LibVMI version 4.4 to conduct introspection. There are various required packages which should be installed before configuration of LibVMI. Indeed LibVMI is an open source project which is available on github too. The LibVMI project is available and retrievable from github.

To verify correctness of installation, several examples are provided by the library at time of set up. We ran them to verify correct installation. By running successfully examples, we made sure that LibVMI is functioning correctly. LibVMI is written in C language. So developers who are interested in developing introspection applications should use from this programming language too. Development of a new introspection application totally depends on the requirements of the user. Indeed it is important that what kind of information does the user need to extract from the memory space of the VM. Based on thhose requirements, application can use from LibVMI API.

LibVMI provides a python wrapper called PyVMI. the provided python wrapper is very useful in two ways. First, developers can use PyVMI to develop introspection tools via python language in addition to C language. Second, PyVMI is required for using Volatility in conjunction with LibVMI.

We configured Volatility framework version 2.4 and connected it to our created live guest VMs. So we used Volatility in conjunction with LibVMI to collect required data by introspecting running VMs. Volatility is designed to work with dump of memory. So using it in conjunction with LibVMI makes this possible to collect data from live VM. For the sake of acquiring the needed data, we operated data collection from the Dom0 for our experiment.

6.2 Experiment

For illustration of reliability of our proposed solution and based on our determined inconsistencies (attributes), we considered to collect data using VMI from 13 VMs. We set up each DomU, did our tests and data collection and we destryed it to create the other DomU.

The reason is because of limitation of our Hardware resources. We depict the result of collected data in Table 6.1. We tried to consider various types of models and scenarios which a VM could take. So some of the VMs are uninfected while some settings changed. Some VMs are infected by malware while settings are either modified or not modified. By doing this, we tried to consider various cases that a guest VM could have.

Looking at Table 6.1, VM1, VM2, VM3, VM4 and VM5 were uninfected entities. In other word, the mentioned VMs were clean and not infected by any kind of malicious piece of code intentionally. As mentioned, various conditions that may happen in non-malicious cases (i.e., shutting down firewall for a legitimate purpose) were considered in settings of these VMs. In this regard, we assumed VM2's user disabled firewall of the VM intentionally for legitimate usage. About VM3, we assumed user disabled Windows defender on purpose. This matter can have various reasons. One of the reasons is that some users might want to use desired anti-virus products. In that case, detection system should be aware of anti-virus processes and check their status too. In VM4, we assumed the user disabled automatic update service. Finally in VM5, we assumed the user disabled all security controls which we considered their absence as inconsistency.

Apart from VM1 to VM5, we infected rest of VMs with a malicious piece of code. We infected VM6 through VM13 with various types of botnet related malwares. We chose a sample of Zbot to infect VM6 and VM7. Then a sample of QAkbot for infection of VM8 and VM9. We chose Koobface for VM10 and VM11 infection. Finally we used Pilleuz malicious software to infect VM12 and VM13. So from VM6 to VM13, are infected each pair of VMs with 1 piece of malware. There is a difference between the first pairs and second pairs. We assumed first pairs have their security controls up and users did not disable them. On the other hand we assumed second pairs have their security controls down. So second pairs are infected while their security controls were down.

We acquired copies of malwares that we used from online resources. Malicious software in the wild might not have the effective functionality as we require. For that reason, We checked accuracy of the obtained malicious files through a legitimate website [31].

We monitored status of each VM with regard to our specified attributes. As mentioned, we put result of checks in Table 6.1. So we analysed the obtained data set with data mining technique. The data mining techniques is for correlation element in the cloud controller. As explained in Chapter 3, clustering is one of data mining techniques which groups elements into user defined number of groups. Elements in each group are more similar to each other. Each problem of data mining has various algorithms. There is no absolute answer to the matter that which algorithm is better. In some research works, researchers conduct cluster analysis with various algorithms and then compare the results with each other to find the optimum algorithm. However in the case of clustering, K-Means algorithm is a well known algorithm which its usage is prevalent. So we used K-Means algorithm for our work too.

To conduct cluster analysis, we tested the collected data set over K-Means algorithm. K-Means algorithm works by primarily choosing centroids of each cluster. A centroid is the mean of elements in each cluster. When the algorithm starts operation, normally centroids are chosen randomly. Random chosen centroids should be within the bounds of all elements. There are other ways of centroid selection too. For example some of the data points can be chosen as primarily position of centroids. Number of centroids has direct relationship with number of clusters. So if we have 10 clusters, we must have 10 centroids.

Each element is a cluster has a distance from the centroid of that cluster. This distance is called error rate. As error rate of elements in a cluster are less, clusters are more com-

pact. In other word small error rate of elements from the centroid indicate elements of a cluster are more alike compared to the situation that error error rate of elements are high. For measuring distance between each data point and centroid in the k-means (error rate), we used the Euclidean distance algorithm.

In cluster analysis, number of clusters is user defined. The user should have preliminary knowledge about the data and his/her own goal for cluster analysis. In our case, our goal in to group elements into two groups of malicious and non malicious. So we need to have 2 clusters. Normally, notation of number of clusters is K and notation of number of data points (elements) is N. So in our case, K=2 which is number of clusters as well as number of centroids and N=13 which is number of data points.

	A.1	A.2	A.3	A.4	A.5	A.6	Cluster index	Error
VM 1	0	0	0	0	0	0	1	0.125
VM 2	0	1	0	0	0	0	1	0.875
VM 3	0	0	0	0	1	0	1	0.875
VM 4	0	0	0	0	0	1	1	0.875
VM 5	0	1	0	0	1	1	2	0.72
VM 6	0	1	0	1	1	1	2	0.52
VM 7	0	0	0	0	0	0	1	0.125
VM 8	1	1	0	1	1	1	2	0.32
VM 9	0	0	0	0	0	0	1	0.125
VM 10	1	1	0	1	1	1	2	0.32
VM 11	1	0	0	0	0	0	1	0.625
VM 12	1	1	0	0	1	1	2	0.52
VM 13	1	0	0	1	0	0	1	1.375

Table 6.1: Collected data from VMs using VMI

Chapter 7

EVALUATION AND DISCUSSION

In this section, we evaluate the obtained result of our cluster analysis. Cluster evaluation is an important part of the result. Results of clustering analysis should be evaluated to measure its goodness. There are various ways to evaluate result of cluster analysis. However good results of cluster evaluation, do not necessarily depict a good result for a specific application. In other word, for an application, result of cluster evaluation might show that good clusters are formed. While in reality, rate of false positive might be high. So there are various ways to evaluate cluster analysis result. We investigated the available methods and chose some with regard to our need.

As mentioned, without an effective cluster evaluation, the result of cluster analysis might remain undefined. Cluster evaluation may address different issues. Some of issues which cluster evaluation might address are in the following:

- Help to determine correct number of clusters: Indeed we mentioned earlier that number of clusters is user defined. Basically user is the best entity which can determine number of clusters. However is the cases that user is not well informed about the data under analysis, then it would be a correct choice to get help of relevant evaluation techniques.

- Compare result of different algorithms: Different algorithms might have relatively outcome and performance. Cluster evaluation is a suitable way to find which algorithm works better for a set of data. In other word, cluster evaluation helps to find the suitable clustering approach.

- Provide evidence whether the data is containing non-random structure.

Basically, the ultimate goal of cluster analysis is to form clusters with high intra-cluster similarity and low inter-cluster similarity. In other word, it is desirable to have clusters which the elements inside it are more similar to each other. Also the elements in one cluster should be more dissimilar to each other. This method is an internal criterion for cluster evaluation. Cluster evaluation has 3 main measures. The three main evaluation measures are: external validity measure, internal validity measure and relative validity measure.

In our case, internal evaluation does not determine the correctness of our result so much. So we mostly focus on the external evaluation measures. One of the simple and external evaluation measures for cluster evaluation is purity. Purity is a transparent evaluation measure to show how accurate the clusters are. The maximum value of purity is 1 and the minimum value is 0. Purity result of 1 indicates a perfect clustering. The purity value of 0 indicates undesirable clustering. Purity is a measure which help to indicate number of clusters too. It is easy to obtain higher purity level by increasing number of clusters.

To compute purity, we determined the majority classes of elements in each cluster. then we measured accuracy of this assignment by counting the number of correctly assigned elements. At the next step, we divided the number by the N. As introduced in the previous chapter, N=13 is the total number of data points. In the first cluster we have 4 correct assignments which are uninfected machines. In the second cluster we have 4 correct assignment which are infected machines. So in total we have 8 correct assignment.

To obtain purity, we divided 8 by N which is 13. The purity of cluster analysis for our system is 0.61 which is higher than average.

Another measure for cluster evaluation is Rand Index (RI). RI gives equal weight to False Positives (FP) and False Negatives (FN). In different applications, it might be a good way to assign different weights for FP and FN points. For an instance example, sometimes separating similar elements would have an impact of decisions than putting similar documents together. F-measure is another measurement which is suitable to assign different weighting to FP and FN points.

As depicted in Table 6.1, each data point is assigned to a cluster. Data points which are grouped in cluster 1 are most of uninfected VMs and infected VMs that had security controls up. Data points that are grouped in cluster 2 are infected VMs that had security controls down. VM5 was not infected and is assigned to cluster 2 which is a false positive. Hence infected VMs that have their security controls off are separated as potential points of threat. This helps to reduce the target group under security analysis. As mentioned previously, result of clustering heavily depends on attributes. In our case attributes are symptoms.

Last column of Table 6.1 consists error rates for each data point. Error rate is the distance of a data point from an assigned cluster's centroid. Error rate's occurrences are depicted in Figure 3. As it is shown, 8 data points have error rate of 0.125 or less in this evaluation. Considering false positive points of first group, most of them have high error rate. So they are far from the centroid. High error rate for false positive points indicate that they are close to be fallen in their true respective group. In other word, It is possible to reduce the number of false positive points by changing, removing or adding some of the attributes.

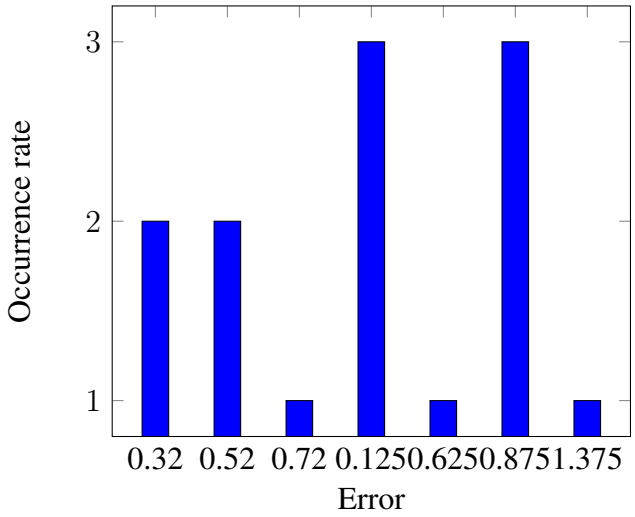

Figure 7.1: Errors of data points in found clusters

Chapter 8

CONCLUSION AND FUTURE WORK

Abusing cloud resources is a prevalent threat toward cloud computing security. This is a threat which addresses mostly service providers than service consumers. Hence, it is vital for service providers to detect service misuses in their network. Detection of such misuses at the right time help to mitigate the threat with suitable countermeasures. 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 are VMs, entire security of the cloud can be increased both for the service provider and the consumer.

In this document we presented a botcloud detection system. Our system detects members of botclouds that abuse cloud resources, among a crowd of VMs. As cloud's environment differ from conventional environments from various aspects, cloud's misuse detection systems should consider some considerations too. A cloud based detection system should be based on one of the cloud's core technologies. Our system is based on Virtualiazation techniques and takes advantage of data mining and virtual machine introspection techniques. Using mentioned technologies, based on some system level attributes and with regard to those attributes, our system groups VMs into two malicious and non-malicious. By identifying elements of botclouds which are grouped separately,

the observable group under inspection gets smaller. As a result, further processes can take place on a smaller target group which consists malicious VMs. In this way, more accurate and reliable analysis can be done on a smaller group of targets which we know that they are malicious. In this regard, the possibility of wrong detection (false positive) in furtherer analysis would be eliminated. However correct grouping of machines in the fist place is very important in this regard.

We would like to further investigate the possibilities of applying classification techniques of data mining on the clustered data (malicious VMs cluster). In other word, we want to furtherer group the malicious VMs either with regard to other attributes. These attributes can be type of attacks they intend to have or type of malware which infected them.

As user's data privacy is always an important matter, the matter of privacy arises here. Although VMI is a very powerful technique for security monitoring tools, at the same time it violates the user's privacy rules too. The information that are obtained by VMI technique, reveals some Our future directions are: expanding the implementation by locating data collection modules in FVMs; testing the proposed solution in a cloud platform and providing methods to protect user's information privacy.

References

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

[2] The Notorious Nine: Cloud Computing Top Threats in 2013. Technical report, Cloud Security Aliance (CSA), 2013.

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

[4] 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.

[5] 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)*.

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

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

[8] 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.

[9] Grobauer, Bernd and Walloschek, Tobias and Stocker, Elmar. Understanding Cloud Computing Vulnerabilities. *IEEE Security and Privacy*, 9(2):50–57, March 2011.

[10] Mell, Peter M. and Grance, Timothy. SP 800-145. The NIST Definition of Cloud Computing. Technical report, Gaithersburg, MD, United States, 2011.

[11] Patel, Ahmed and Taghavi, Mona and Bakhtiyari, Kaveh and Celestino JúNior, Joaquim. Review: An Intrusion Detection and Prevention System in Cloud Computing: A Systematic Review. *J. Netw. Comput. Appl.*, 36(1):25–41, January 2013.

[12] Dr. M.A.C. Dekker, Dimitra Liveri. Cloud Security Guide for SMEs. Technical report, European Union Agency for Network and Information Security (ENISA), 2015.

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

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

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

[16] V.R. Kebande and H.S. 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.

[17] Badis, Hammi and Doyen, Guillaume and Khatoun, Rida. Toward a Source Detection of Botclouds: A PCA-Based Approach. In *Monitoring and Securing Virtualized Networks and Services*, pages 105–117. 2014.

[18] Shun-Wen Hsiao, Yi-Ning Chen, Y.S. Sun, and Meng Chang 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] Francois, J. and Shaonan Wang and Bronzi, W. and State, R. and Engel, T. BotCloud: Detecting botnets using MapReduce. In *Information Forensics and Security (WIFS), 2011 IEEE International Workshop on*, pages 1–6, 2011.

[20] Hammi, B. and Khatoun, R. and Doyen, G. A Factorial Space for a System-Based Detection of Botcloud Activity. In *New Technologies, Mobility and Security (NTMS), 2014 6th International Conference on*, pages 1–5, 2014.

[21] Tal Garfinkel and Mendel Rosenblum. A Virtual Machine Introspection Based Architecture for Intrusion Detection. In *In Proc. Network and Distributed Systems Security Symposium*, pages 191–206, 2003.

[22] Harrington, Peter. *Machine Learning in Action*. Manning Publications Co., Greenwich, CT, USA, 2012.

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

[25] K. Harrison, B. Bordbar, S.T.T. Ali, C.I. 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.

[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] Junghwan Rhee and Riley, R. and Dongyan Xu and Xuxian Jiang. Defeating Dynamic Data Kernel Rootkit Attacks via VMM-Based Guest-Transparent Monitoring. In *International Conference on Availability, Reliability and Security, 2009 (ARES '09)*.

[28] Jianxiong Wang. A rule-based approach for rootkit detection. In *2010 The 2nd IEEE International Conference on Information Management and Engineering (ICIME)*.

[29] Sikorski, Michael and Honig, Andrew. *Practical Malware Analysis: The Hands-On Guide to Dissecting Malicious Software*. No Starch Press, San Francisco, CA, USA, 1st edition, 2012.

[30] 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*. Wiley Publishing, 1st edition, 2014.

[31] Virustotal website.