CHALMERS

Master of Science Thesis in the Master Degree Programme, Secure and Dependable Computer Systems

Using honeypots to study skill level of attackers based on the exploited vulnerabilities in the network

Vusal Aliyev

Department of Computer Science and Engineering
Division of Computer Security
CHALMERS UNIVERSITY OF TECHNOLOGY
Göteborg, Sweden, 2010

Abstract

Malware in the form of computer viruses, worms, trojan horses, rootkits, and spyware acts as a major threat to the security of networks and creates significant security risks to the organizations. In order to protect the networked systems against these kinds of threats and try to find methods to stop at least some part of them, we must learn more about their behavior, and also methods and tactics of the attackers, which attack our networks.

This thesis makes a practical analysis of observed attacks and exploited vulnerabilities using honeypots in an organization network. Based on this,we study the attackers' behavior and in particular the skill level of the attackers once they gain access to the honeypot systems. The first part of the work describes: i) the honeypot architecture as well as implementation details so that we can observe the attackers behavior and ii) proposed hybrid honeypot solution which will be used in the future work. The second part presents: i) the detailed analysis and classification of the attacks and vulnerabilities, which are used by the attackers and ii) the attackers' skill level based on the exploited vulnerabilities.

Acknowledgments

This master thesis wouldn't have been a success without the help and support of a great number of people.
First and foremost I would like to thank my supervisor Tomas Olsson to have it made possible for me to work on my master thesis at the Swedish Institute of Computer Science (SICS) and his understanding and availability as well as his thoughtful guidance when things were turning complicated. Without his energy, and patience this work would not be possible. Then**,** my gratitude also goes to the members of Industrial Applications and Methods Lab (IAM), especially to Anders Holst and Björn Bjurling for proposing this project. I want to thank System and Network Administration team, especially to Alex Bustamante and John Morrison for their support for network problems.
I also want to thank all the teachers of my master programme entitled Secure and Dependable Computer Systems at Chalmers University of Technology who efficiently prepared me for this master's thesis, in particular my examiner professor Erland Jonsson for sharing his insights and knowledge about security.
Big thanks to my friend Aslan Askarov who motivated me to study the master program within IT Security. Finally, I would like to thank my family for supporting me from Home all the time I have been in Sweden.

Contents

1 Introduction **3**

2 Honeypots **7**
2.1 Concept of honeypots 7
2.2 Types of honeypots 7
2.3 Honeynet 8

3 Related work **11**

4 Implemented architecture and proposed hybrid honeypot framework **13**
4.1 Introduction 13
4.2 Implemented Architecture 13
4.2.1 Virtual Honeynet Architecture 14
4.2.2 Honeywall 17
4.3 Proposed Hybrid honeypot framework 17
4.3.1 Introduction 17
4.3.2 Hybrid Honeypot Architecture 18

5 The honeypot architecture Implementation **23**
5.1 Honeyd 24
5.1.1 Installing, configuring and running. 26
5.1.2 Logging 27
5.2 Virtual honeynet 28
5.2.1 Virtualbox Honeypoting 29
5.2.2 Honeywall Roo 31
5.2.3 Snort IDS and Snort-inline IPS 34
5.2.4 Sebek Data Capture 34
5.2.5 Setting up virtual High-Interaction honeypots 35
5.2.5.1 Honeypot1-Ubuntu 36
5.2.5.2 Honeypot2 -Windows 37

6 Experimental results **39**
6.1 Forensics Analysis 39
6.2 Observed attacks 40
6.3 Statistical analysis 47

7 Classification and skill level of attackers **53**
7.1 Classification of attacks and exploited vulnerabilities 53
7.2 Skill level of attackers 58

8 Conclusion **61**

References **63**

Chapter 1

Introduction

During the last decade, a number of tools have been developed to defend against the attacks that organizations are facing. Firewalls, for example, help to protect these organizations and prevent attackers from performing their activities. Intrusion Detection Systems (IDS) are another example of such tools allowing companies to detect and identify attacks, and provide reaction mechanisms against them, or at least reduce their effects. But these tools sometimes lack functionality of detecting new threats and collection of more information about the attacker's activities, methods and skills. For example, signature based IDS's are not capable of detecting new unknown attacks, because they do not have the signatures of the new attacks in their signature database. Thus, they are only able to detect already known attacks. Nevertheless, in order to better protect your organization and build efficient security systems, the developers should gain knowledge of vulnerabilities, attacks and activities of attackers. Today many non-profit research organizations and educational institutions research and analyze methods and tactics of the so-called blackhat community, which acts against their networks. These organizations usually use honeypots to analyze attacks and vulnerabilities, and learn more about the techniques, tactics, intention, and motivations of the attackers [7].

Another important advantage of using honeypots is that they allow us to analyze how the attackers act for exploiting of the system's vulnerabilities. Thus, this analysis provides valuable information to security experts to study the skill level of the attackers. This is what our work will focus on. The goal of our project is to study the skill level of the attackers based on the exploited vulnerabilities in the honeypot environment. In this work, we provide the vulnerable systems for the attackers which are built and set up in order to be hacked. These systems are monitored closely, and the attackers' skills are studied based on the gathered data.

In order to react properly against detected attacks, the observed skill and knowledge of the attackers should be taken into account when the countermeasure process is activated by the security system designers. Therefore, the experimental studies of the attacker's skill level would be very useful to design proper and efficient reaction model against the malwares and blackhat community in the organization's computer network.

The work presented in this thesis creates the following main contributions to help learning the attacker's skill level:

- Implementing the virtual honeypot architecture and proposing an improved hybrid honeypot framework;
- Analysis and classification of the observed attacks and vulnerabilities;

The rest of the report is organized as follows. Chapter 2 provides the necessary background information about honeypots. Chapter 3 presents the review of research related to our work. In Chapter 4, we introduce the honeypot architecture which is implemented in Chapter 5, and propose an improved hybrid honeypot framework for future honeypot deployment. Chapter 6 is dedicated to the forensic and statistical analysis of attacks gathered from the honeypots. The corresponding activities by the attackers are also analyzed in this chapter. The classification of attacks and exploited vulnerabilities and the evaluation of the skill level of the attackers based on the lessons learned from the analysis and statistics of the collected data are showed in Chapter 7. Finally, Chapter 8 concludes the report.

Chapter 2

Honeypots

This chapter contains an overview of honeypot concepts and types which are used in our honeypot deployment, as well as an introduction to honeynet.

2.1 Concept of honeypots

The concept of honeypots was first proposed in Clifford Stoll's book "The Cuckoo's Egg", and Bill Cheswick's paper "An Evening with Berferd"[8]. "The Cuckoo's Egg" [8] is a true-story look at trailing hackers which have been done by the author. There are many definitions of honeypot since the first time appearance by Clifford Stoll, but the most common definition was given by the leading authority on honeypots - Lance Spitzner in [6]: *"A Honeypot is an information system resource whose value lies in unauthorized or illicit use of that resource."* Honeypots as an easy target for the attackers can simulate many vulnerable hosts in the network and provide us with valuable information of blackhat community. Honeypots are not the solution to the network security, they are tools which are implemented for discovering unwanted activities on a network. They are not intrusion detectors, but they teach us how to improve our network security or more importantly, teach us what to look for.
According to the definition we can note that honeypot is a system which is built and set up in order to be hacked. Except for this, honeypot is also a trap system for the attackers which is deployed to counteract the resources of the attacker and slow him down, thus he wastes his time on the honeypot instead of attacking the production systems.

2.2 Types of honeypots

Honeypots are classified into three types [6]. The first classification is according to the use of honeypots, in other word for what purpose they are used: production or research purpose. The second classification is based on the level of interactivity that they provide the attackers: low or high interaction honeypots. The last one is the classification of honeypots according to their implementation: physical and virtual honeypots.

Production honeypots

Production honeypots are basically used by organizations to mitigate risks [6]. By using them, organizations can capture only limited information about blackhat community compared to research honeypots. They are placed in the production networks to help improving the security. Most of the production honeypots are low-interaction honeypots which are easy to deploy.

Research honeypots

These honeypots are mostly deployed and used by non-profit research organizations or educational institutes. Security researchers by using this type of honeypots get more information about attacks and the methods of attackers and it helps them to design better security tools. For example, research honeypots can be used in strengthening existing intrusion detection systems and firewalls.

Low/High Interactivity

Low-interaction honeypots simulate only some parts of the system, for example, the network stack [10] Although the low-interactions don't implement real services as high-interaction, they are useful to collect information at higher level e.g., learn about network probes or worm activity. They don't offer attackers to realize operations. In other words, low-interaction honeypots simulate only services that can't be used by an attacker to get full access to the honeypot and shouldn't be able to control the system. Typical use of these honeypots includes identification of port scans and, generation of attack signature and malware collection. Popular examples of this kind of honeypots are Honeyd [26] and Nepenthes [12]. They are also used for analyzing spammers or detecting worms.
A high-interaction honeypot is a conventional computer system or a fully functional VM (Virtual Machine), a router or a switch. Here attackers can interact with a real system where almost nothing is restricted and thus, it's more risky compared to low-interaction honeypots. Therefore this kind of honeypot is normally placed behind a firewall to lessen the risk. They are not easily deployable compared to low-interaction honeypots, but by using them we can learn more about the attackers' behavior and find out new vulnerabilities which we didn't already know of in our network or machine. We can also use them to analyze the behavior of non automatic attacks managed by human beings. These kinds of honeypots are usually used as research honeypots. There are some examples of high-interaction honeypots which are presented in more detail in [3]. We can mention e.g, Argos, Potemkin and Honeybow.

Hybrid honeypots

As we mentioned in our previous discussions, low-interaction honeypots are not so powerful, but they are more secure and easily deployable than high-interaction honeypots. In contrast, high-interaction honeypots are too expensive, they run on real services, and create higher risk. But by combining the advantages of both honeypots, we can use low-interaction honeypots as a proxy which filters and redirects incoming traffic to the high-interaction honeypots. This kind of honeypot combination is called a hybrid. The main goal of using hybrid honeypots is to extend the scalability of low-interaction honeypots and fidelity of high-interaction honeypots in order to collect detailed attacker activities over a large IP subnet, successfully analyze and evaluate a new attack. Table 1 shows the comparison between low, high interaction honeypots and hybrid honeypots.

2.3 Honeynet

A honeynet is a network of high-interaction honeypots. It is used in large networks in which one honeypot is not sufficient to monitor all kind of system and network activities. Therefore many honeypots are placed within this network. This network provides real systems which attackers can interact with, and it is intended to help system administrators to know about vulnerabilities and compromises within the network. It has a network architecture which provides Data Control, Data Capture and Data Collection. Data Control is defined as management and controlling how traffic can flow in and out the honeynet. Data Capture provides the monitoring and logging of the attacker's activities within the honeynet. The purpose of the data collection is to collect the data from the honeypots and combine them in a central location.

High-interaction honeypot	Low-interaction honeypot		Hybrid honeypot
- Slow	+ Fast		+Fast
+ Able to detect unknown attacks + 0 False positive	- Unable to detect unknown attacks	→	+ Able to detect unknown attacks + 0 False positive
- Unable to deal with time bombs and user interaction	+ Able to deal with time bombs and user interaction		+ Able to deal with time bombs and user interaction
- Expensive	+ Cheap		+ Better RoI
- Difficult to setup and operate	+ Easy to setup and operate		- Difficult to setup and operate

Table 1 Comparison between low, high and hybrid honeypots [21]

In the deployment of a honeynet architecture, a honeywall is a key element. It is a transparent gateway to the high-interaction honeypots within the honeynet, and undetectable by attackers. Its purpose is logging and to restrict the incoming and outgoing traffic to/from the honeypots. From the attackers point of view, the honeywall acts like a bridge and can't be detected by attackers. In our experiment, we use a virtual honeynet solution by running high-interaction honeypots on a single physical machine. By using virtualization software, it is possible to run multiple operating systems at the same time in a single hardware, and it allows us to reduce cost and easily control the honeypots. We will talk more about honeywall and virtual honeynet in further chapters.

Chapter 3

Related work

This chapter introduces previous work and the concepts related to our project.
Based on honeypot techniques researchers have developed many methods and tools for the collection of malicious software. The book [3] and the honeynet project [7], as main sources of our work, provide useful guidelines for the implementation of honeypots and practically experimental tools which have been used in different honeypot projects. Among them there are some honeypot projects which are related to our work. One of the main references which we used often was research outcomes of Leurrecom honeypot project [18]. The Leurrecom project has been created by the Eurocom Institute in 2003. The main goal of this project was to deploy low-interaction honeypots across the internet to collect data and learn more about the attacks which were gathered by their platforms in over 20 countries all over the world. Also we benefited from the research papers of LAAS (The Laboratory of Analysis and Architecture of Systems) [19, 20] for deployment of high-interaction honeypots and precise analysis of the observed attacks, attackers skills and exploited vulnerabilities.
The first time the hybrid honeypot framework has been published in the research paper by Hasan Artail. He proposed this framework [24] in order to improve intrusion detection systems and extend the scalability and flexibility of the honeypots. This approach was helpful when we designed our own Hybrid Honeypot architecture which will be proposed as a future work.
There are two important taxonomies on attack processes: Howard's computer and network security taxonomy [33] and Alvarez's Web attacks taxonomy [43]. Howard's taxonomy classifies the whole attack process of an attacker. The other taxonomy also focus on the attack process, thus it is based on the attack life cycle in analysis of Web attacks. There is also a taxonomy proposed by Hansman and Hunt's [36] which has a four unique dimensional taxonomy that provide a classification covering network and computer attacks.
The paper of Wael Kanoun et al. [44] describes the assessment of skill and knowledge level of the attackers from a defensive point of view.
Tomas Olsson's work [45] discusses the required exploitation skill-level of the vulnerability and the exploitation skill of the attacker which are used to calculate a probability estimation of a successful attack. The statistical model created by him is useful in order to incorporate real-time monitor data from a honeypot in assessing security risks. He also classifies exploitation skill-levels into Low, MediumLow, MediumHigh, and High levels.

.

Chapter 4

Implemented architecture and proposed hybrid honeypot framework

4.1 Introduction

We decided to deploy both low and high-interaction honeypots in our experiment. This permitted us to provide comprehensive statistics about the threats, collect high-level information about the attacks, and monitor the activities carried out by different kind attackers (human beings, automated tools). In this chapter we present the whole architecture used in our work and propose a hybrid honeypot framework that will be implemented in the future.
In the hybrid honeypot system, low-interaction honeypots play the role of a gateway to high-interaction honeypots. Low-interaction honeypots filter out incoming traffic and provide the forwarding of selected connections. In other words, a low-interaction honeypot works as proxy between attacker and the high-interaction honeypot. Hybrid systems include scalability of low interaction honeypots and fidelity of high interaction honeypots [24]. In order to achieve this, low interaction honeypots must be able to collect all of the attacks while unknown attacks should be redirected to high-interaction honeypots. Attackers without any restrictions can get access to high-interaction honeypots which have high fidelity. By using a hybrid architecture, we can reduce the cost of deploying honeypots. But due to lack of time we did not implement the proposed hybrid honeypot architecture.

4.2 Implemented Architecture

For our experiment, we implemented a honeypot architecture which combines the both low and high interaction honeypots. For the low-interaction part we used Honeyd [2] and for the high-interaction part we implemented a virtual honeynet architecture based on the Virtualbox virtualization software [13]. The low- and high-interaction honeypots are deployed separately, and the backup of the collected attack data on each host machine of the low and high-interaction honeypots is stored in a common database on a remote machine. The implemented architecture is illustrated in Figure 4.1. In our implementation, we used only two physical machines which contain the virtual honeypots and a remote management machine to remotely control the collection of attack data and to monitor the activities and processes on the honeypots. All of the honeypots are deployed and configured on the virtual machines. We describe virtual honeynet architecture in more detailed in next section. There is a disadvantage of the implemented architecture in that it has no filter mechanism for the incoming traffic. That means that, no port-scan attempts or connections to closed ports will be filtered, and high interaction honeypots can become overwhelmed by receiving a large volume of such traffic. It can also receive uninterested traffic such as unestablished TCP connections that have been received many times before. That is why we propose an improved hybrid honeypot architecture as a future work.

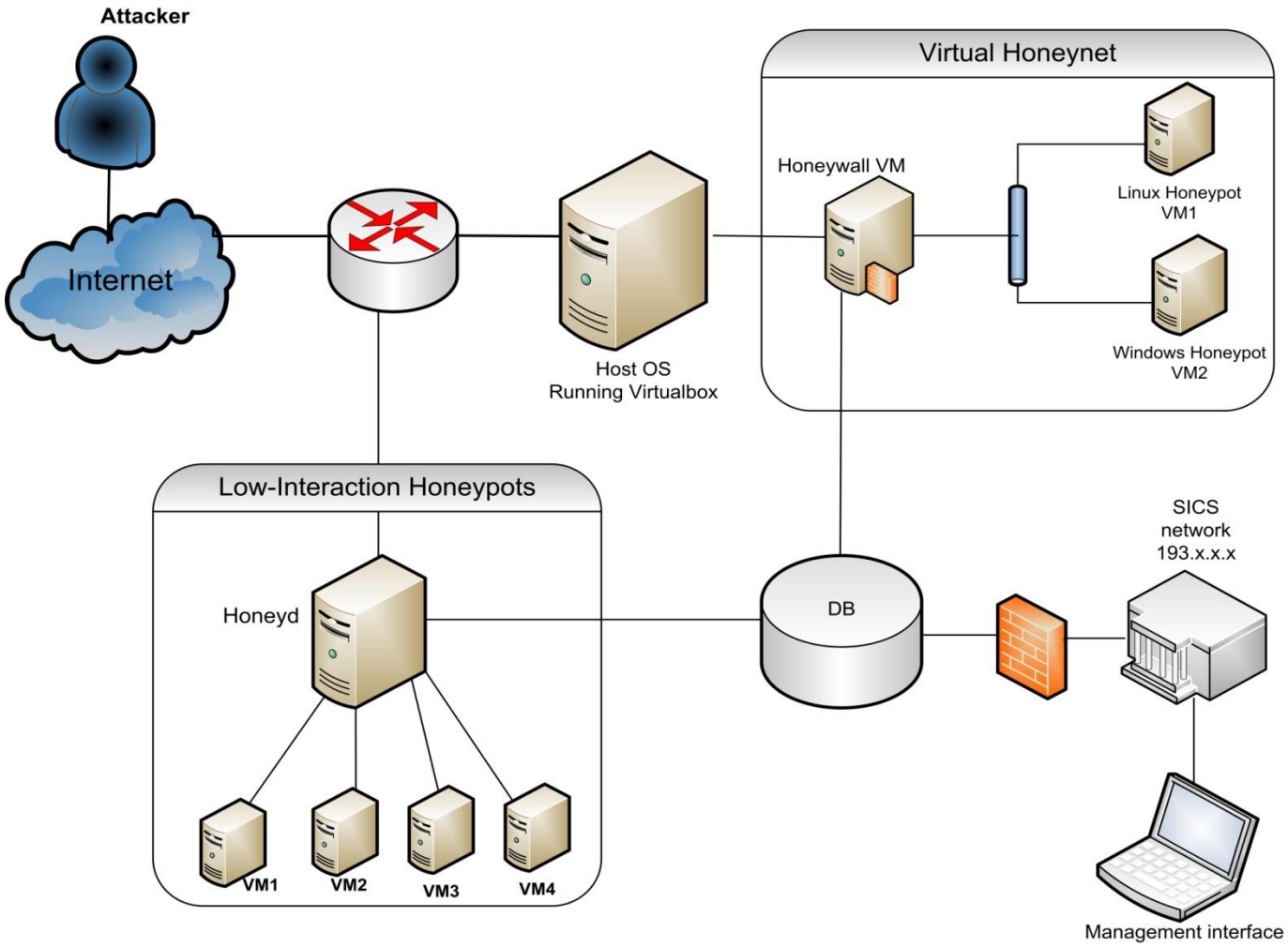


Figure 4.1 Implemented honeypot architecture

4.2.1 Virtual Honeynet architecture

There are three honeynet architectures which have been developed by the Honeynet alliance [7]

- GEN I
- GEN II
- GEN III

GEN I was the first developed architecture and had limited functionality in Data Capture and Data Control. It was effective to detect entry level attacks against targets, but the architecture was simple. Thus, the limitation in outbound connections can allow the attackers to detect the existence of the honeynet, which is called fingerprinting. In 2002, GEN II Honeynets were developed in order to address the issues with GEN I Honeynets, and after two years, GEN III was released. Changes in the last two generations of architectures make it possible to handle data capture and data control mechanisms of the honeynet in a single device where is called the Honeywall. And the fingerprinting issue is also solved in the new architecture where attackers can't detect the honeynet anymore. GEN II

and GENIII honeynets have the same architecture. The only difference between them is the addition of a Sebek server [25] installed in the honeywall within GEN III architecture. SEBEK is a data capture tool which is installed and hidden on the honeypots which are connected to Honeywall. It logs the attackers activities according to the system read and write calls. It's very useful capturing tool which is based on the client-server architecture. In Figure 4.2, you can see an example of a GEN III Honeynet architecture.

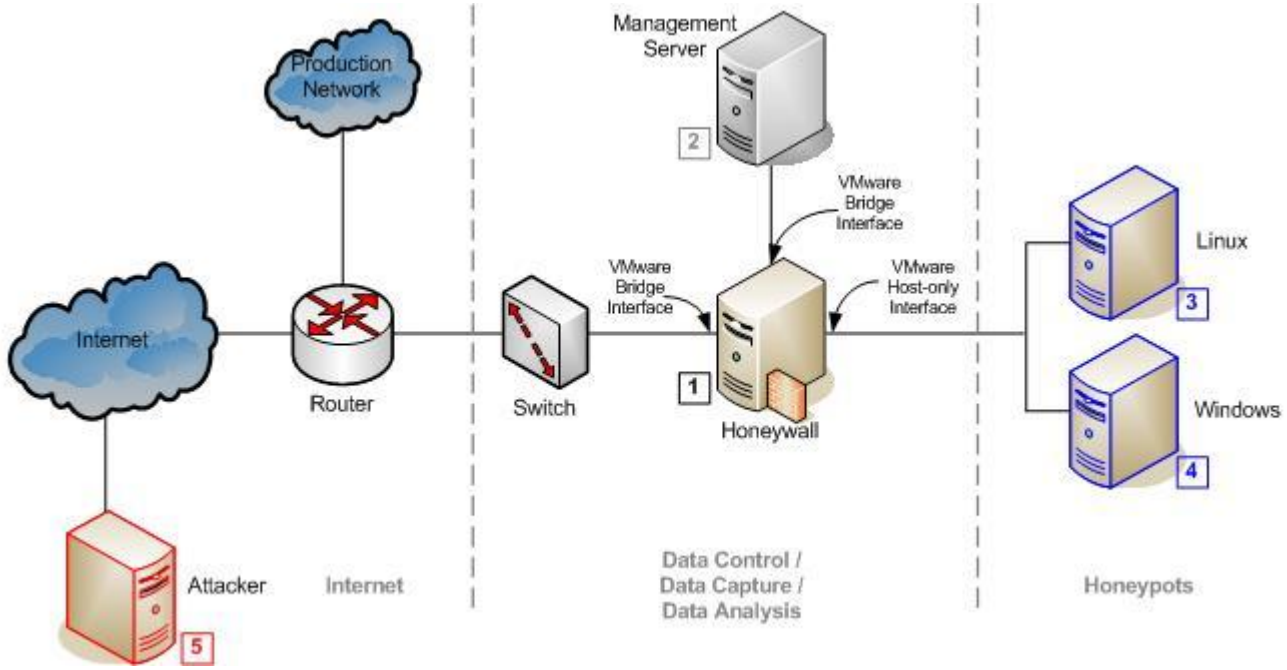
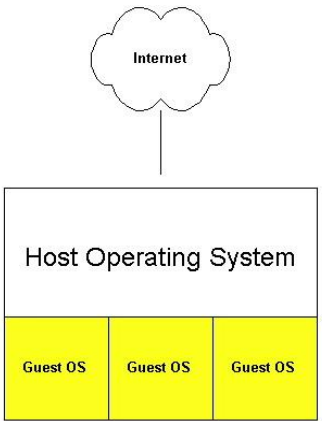


Figure 4.2 GEN III Honeynet architecture Example [26]

Using virtualization, we can run multiple virtual machines on a single physical machine. An operating systems running inside of physical and virtual machines are referred to the *host and guest system (guest virtual machine)*. In order to achieve virtualization, the host machine should share the CPU and memory resources with the guest virtual machines.
The term virtual is used because all different operating systems have the 'appearance' to be running as an independent computer [27]. Virtualization software plays an important role in implementing virtual honeynets.
Below you can see some specific advantages and disadvantages of the virtual honeynets compare to traditional honeynets.



Virtual Honeynet	
Advantages	**Disadvantages**
▪ reduced cost ▪ easier maintenance ▪ Portable	▪ Fingerprinting risk ▪ Limitation of OS according to hardware ,and Virtualization software

In our experiment we deploy and built Virtual Honeynet as a high-interaction part of our implemented architecture. We used a single physical machine to install a complete honeynet. More details about the implementation of virtual honeynet are described in Chapter 5.

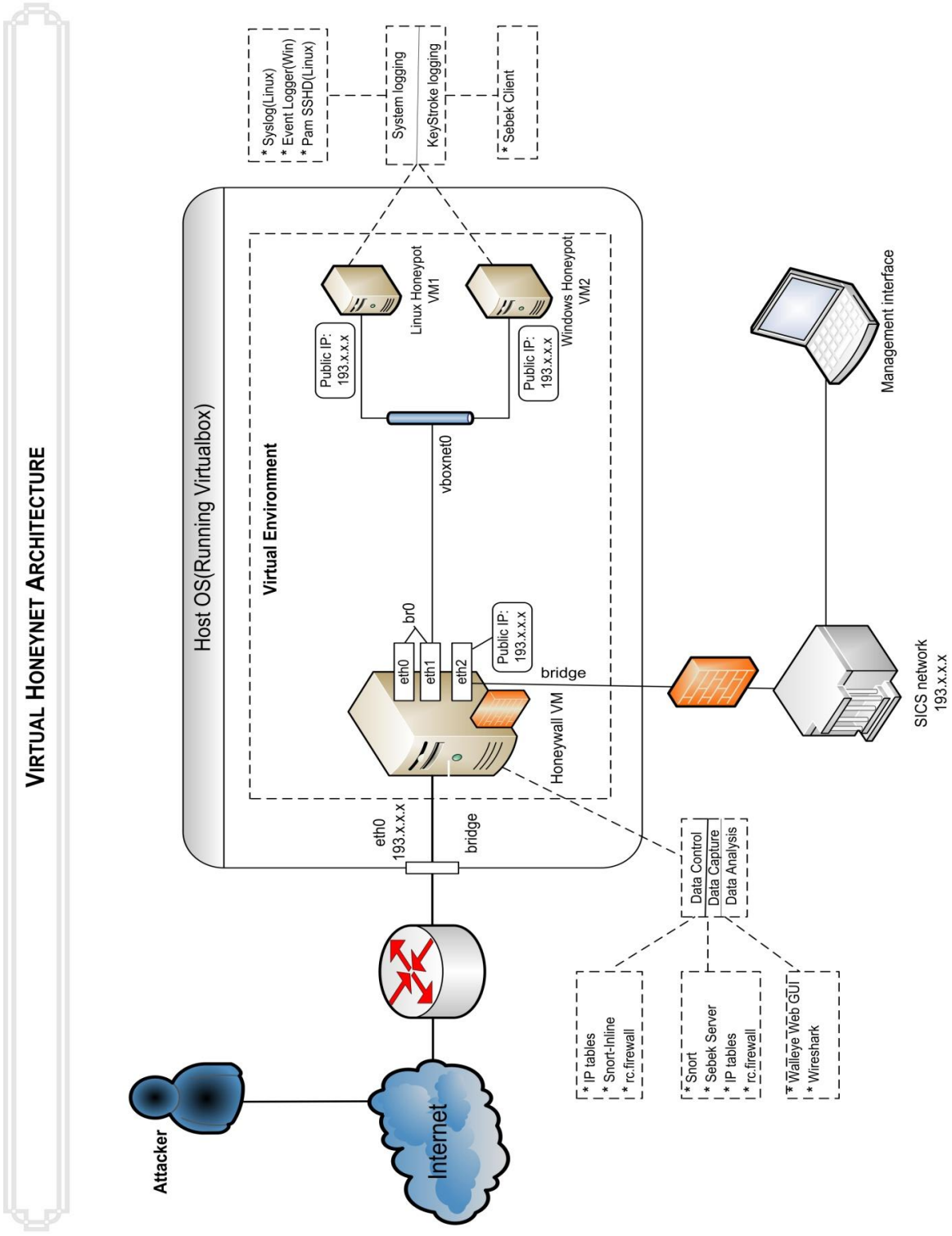


Figure 4.3: Virtual Honeynet Architecture

4.2.2 Honeywall

A Honeywall is used to safeguard honeypots in the network from malware. It can translate and route packets by changing IP and MAC addresses. In our experimental honeypot architecture we use a honeywall as a gateway to the high-interaction honeypots. A Honeywall has three main goals [3]:

- Data Capture: All activities of the attacker within the honeynet and the information that enters and leaves the honeynet should be captured without attackers knowing they are monitored;
- Data Control: To control suspicious traffic entering or leaving the honeynet. Moreover, this mechanism must ensure that once a honeypot within the honeynet is compromised, all malicious activities must be contained within the honeynet;
- Data Analysis: To help the operator of a honeynet to simplify the analysis of all captured data and help in computer and network forensics;

Installation of the honeywall is very easy, because the "Honeynet project" provides the Honeywall CD-ROM which can setup within a few minutes. All the captured data in the Honeywall will be written to a database.
Data Capture collects data of activities of the attacker. The simplest mechanism for Data Capture is to capture all incoming and outgoing traffic. This can be done by tools like tcpdump or Wireshark. These tools simply log all packets passing through the network interfaces and write them into a database for later analysis. Moreover, all events that are logged by an installed IDS or IPS are also logged into a database. Writing the malicious activities into a database and using IDS helps us to identify the attack types (unknown or known) easily. A detailed description of the Honeywall is presented in next chapter.

4.3 Proposed Hybrid honeypot framework

4.3.1 Introduction

As a future work we propose an improved hybrid honeypot framework. We already mentioned above that, the first time hybrid honeypot framework has been proposed by Hasan Artail [24]. His hybrid honeypot framework is shown in Figure 4.4. It consists of one single common gateway for external traffic and three different internet zones. Production server and clients are in the first zone. The second zone consists of Honeyd server. The Honeyd server has three different services. The first one is for collecting incoming traffic, and stores them in the Honeyd database. The second service generates honeypots based on the statistics provided by the database [24] and the third service provides redirection between low and high interaction honeypots. The last zone consists of an array of high-interaction honeypots running on Physical Machines. As we can see, by default, all the connections are directed into the second zone. And the redirection can happen where the low interaction honeypot filters the traffic to a high interaction honeypot in the third zone. This kind of method can prevent attackers from identifying the existence of the honeypot environment, and provides better configuration to monitor attacks in detail.

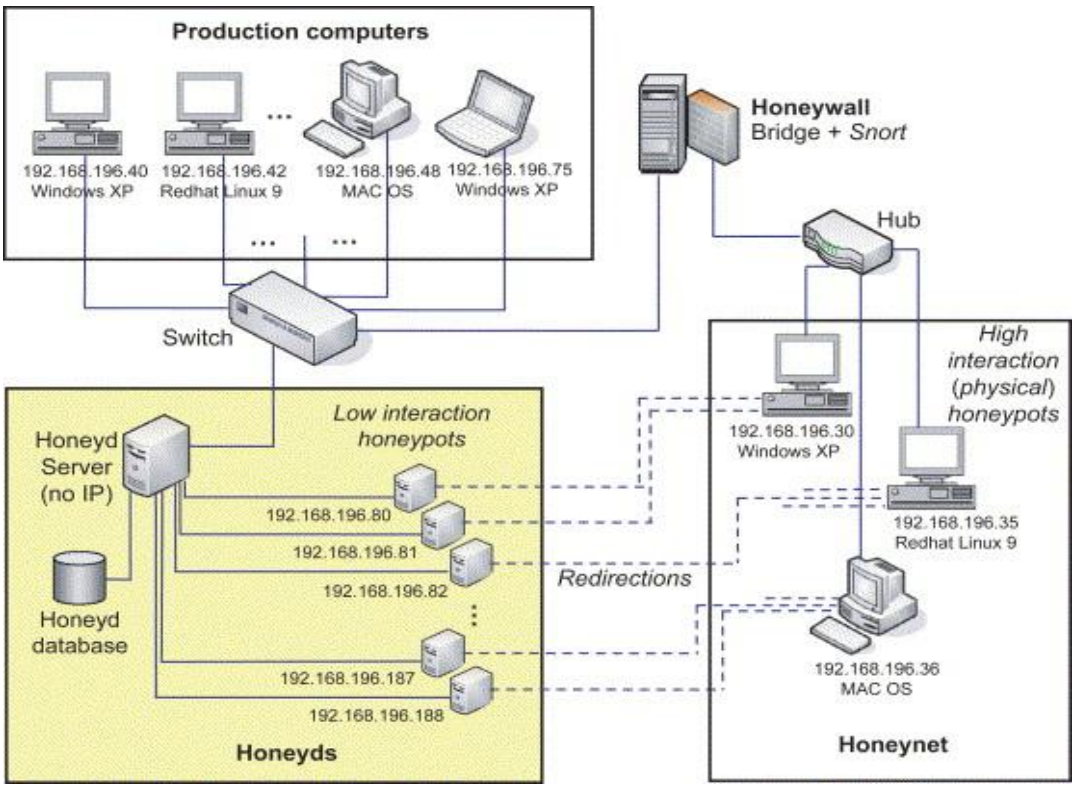


Figure 4.4 Typical hybrid honeypot deployment [24]

4.3.2 Hybrid Honeypot Architecture

In designing our proposed hybrid honeypot solution, we have benefited from the relevant research into hybrid honeypot systems. We are going to discuss the ideas which can be achieved by using the combination of existing honeypot solutions, and give brief overview of the current research systems that use different approaches to deploying hybrid honeypots. These research systems are distinguished according to their functionality and they are grouped as follows:

-solutions to increase the resource management on high-interaction honeypots;

-architectures that use an intelligent gateway to filter unnecessary incoming traffic;

There are three main research systems which are used to deploy honeypots to large number of IP addresses. Two of them - Collapsar, Potemkin [3] are the projects which use virtual high-interaction honeypots. Their goal is to enhance the scalability, and increase resource management –minimize the cost and risk of deploying high-interaction honeypots.

Collapsar is VM based honeypot architecture for collecting traffic from different remote large networks [3]. Collapsar system provides a central management, distributed presence, and convenient attack correlation and data mining methods in order to achieve a highly scalable system. Figure 4.5 shows the high-level Collapsar architecture which consists of five components: traffic redirectors, a transparent firewall-like front end, virtual-machine-based honeypots, a management station and a correlation engine. The traffic redirectors are installed in different production networks that you can monitor, while the rest of the infrastructure is centralized.

The redirectors forward addresses in the production networks to the front end. The traffic is redirected via GRE tunnels. The front end acts as a firewall gate for Collapsar Center and filters the GRE tunneled packets and forward them to high-interaction honeypots. Collapsar has three modules:

the logging module, the trapping module and the correlation module. The trapping module is used to reduce the rate of outgoing TCP-SYN packets and examine outgoing traffic based on signatures of known attacks. The correlation module detects DDOS attacks, worm propagation and hidden overlay networks.

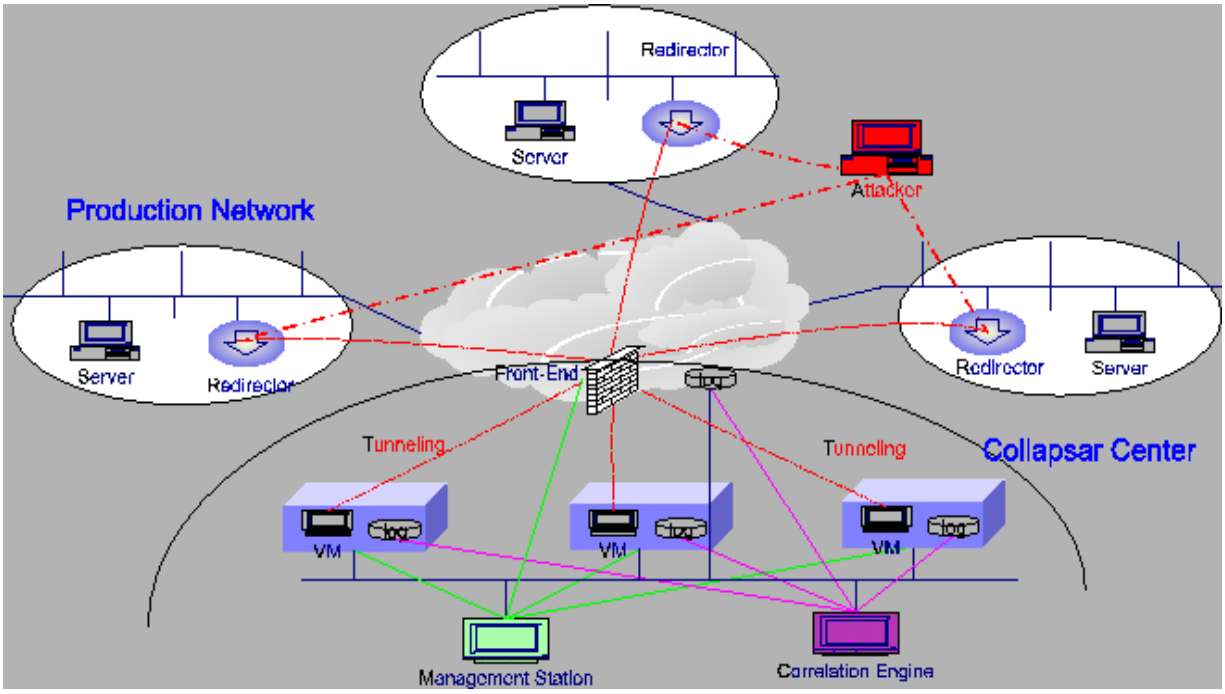


Figure 4.5 Collapsar architecture [3]

Another big project called Potemkin [3] has been built to gather traffic from large IP spaces by only using high-interaction honeypots. The main idea behind this project is to emulate vulnerable services. It is such a honeyfarm architecture that generates virtual machines on demand when there are incoming connections.

When the incoming network packet arrives to a router, it is forwarded to the Potemkin gateway which binds IP addresses to physical honeyfarm servers, as described in the figure 4.6. Here we see that for each IP address the honeyfarm server creates VM. Thus, honeyfarm server allocates the memory for each VM. If IP address is not active, then VMs stop using the CPU and physical memory. Therefore Potemkin can emulate over 64,000 honeypots in live development by using only a few physical machines [3].

There is one more work in high-performance honeypot development which is called RolePlayer.It has a completely different approach compare to Collapsar and Potemkin.

Instead of creating and using many virtual machines and making them more efficient, the creators of RolePlayer implemented a self-learning mechanism which can mimic protocols from the previous already known attacks. This method can help to drop uninteresting traffic. In this case, RolePlayer can filter known attacks and forward unknown, valuable attacks to high-interaction honeypots. Therefore, RolePlayer can easily learn new protocols by observing just a few sessions [3].

All of three above-mentioned hybrid solutions are not open source, and can't be changed. They are implemented for research purposes of the universities, not available for public use. They give us useful insights into designing our own high-performance hybrid honeypot system. It's possible to create such hybrid system ourselves with using NAT, low-interaction honeypots like Honeyd, and high-interaction honeypots. From Collapsar we can learn how to handle the traffic redirection for a hybrid system. From Potemkin we learn how to manage VMs as high interaction honeypots. Potemkin introduces some

interesting methods such as delta virtualization and flash cloning which will be used when implementing high-interaction honeypots.

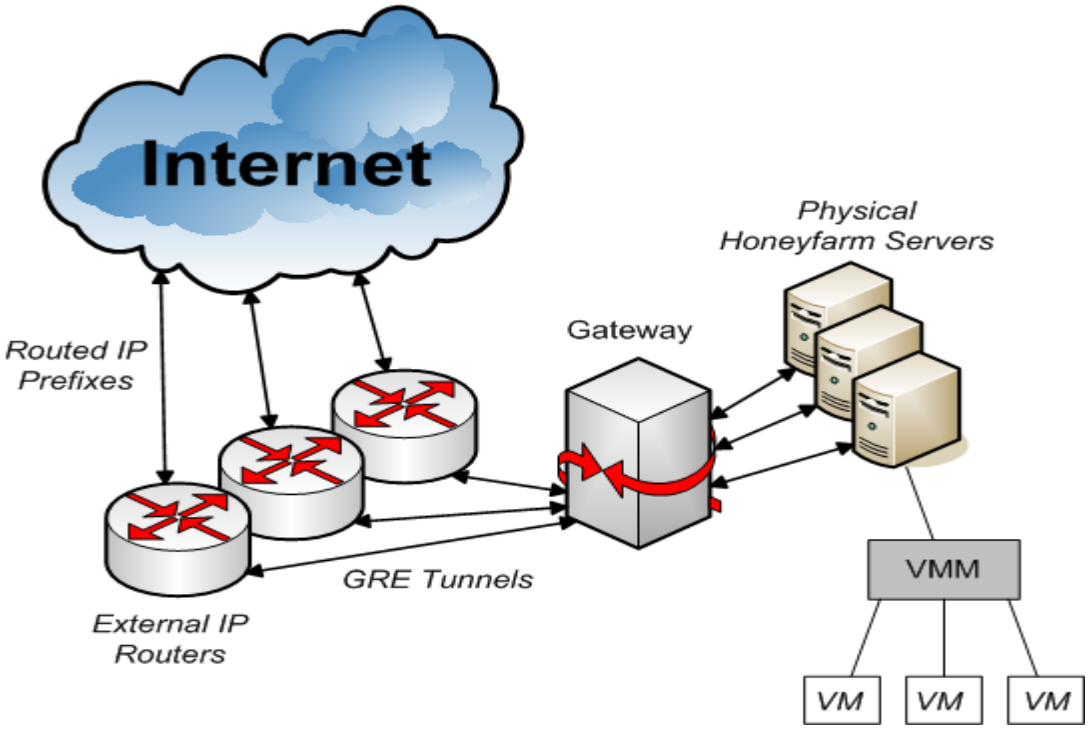


Figure 4.6 Potemkin honeyfarm architecture

By using the ideas from these hybrid solutions and the research paper [24] we propose a framework based on the hybrid honeypot architecture. This architecture can monitor large number of IP addresses and control both incoming and outgoing traffic.

This hybrid honeypot solution combines the advantages of the low and high-interaction honeypots for collection of detailed activities of attackers on large IP spaces. Figure 4.7 shows our proposed Hybrid Honeypot Architecture. It consists of three parts: a Honeywall gateway, a set of low-interaction honeypots and a set of high-interaction honeypots. Compared to our experimental honeypot architecture, Honeywall Gateway is placed outside of the low and high interaction parts of the system, and in addition, the Decision and Redirection Modules should be implemented within this gateway. The Decision Module should perform filtering of incoming network traffic based on the some filtering criteria. By default, all incoming traffic should be sent to the low-interaction honeypots, and among the established network sessions there can be some unknown manual attacks which are more interesting for high-interaction honeypots. Thus, the Decision Module selects these interesting network connections, makes decision in order to route these connections to High-Interaction virtual machines, and stores them in the routing table. Then The Redirection module transparently redirects those attacks to high-interaction honeypots by changing the destination of the selected traffic. In this case high-interaction honeypots offer full interaction to the attackers.

Low and high interaction honeypots interact with the gateway during replying to incoming attack traffic. They connect to gateway through the TCP/IP network. Similar to our experimental honeypot architecture here we also used Honeyd [2] for the low-interaction part and for the high-interaction part by using Virtualbox [13]. The Honeywall Gateway also provides logging and control of network traffic. Thus, all allowed incoming traffic passes through this gateway, but outgoing connections are limited in order to minimize damages by the attackers.

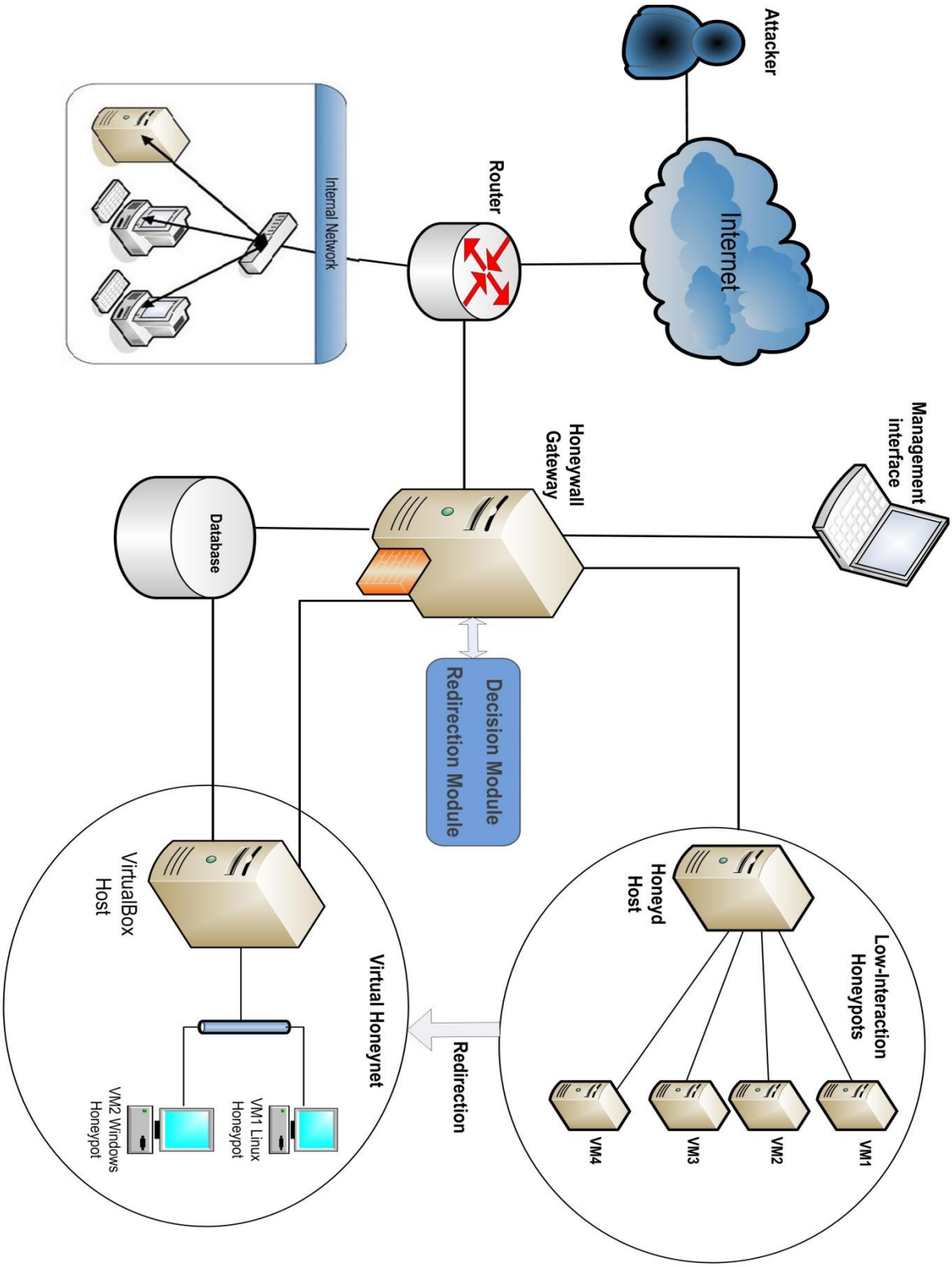


Figure 4.7 Proposed Hybrid Honeypot Architecture

Chapter 5

The honeypot architecture implementation

In this section, we will describe how to set up the honeypots which is proposed in our experimental honeypot architecture. We will also indicate the issues that are involved during the deployment of these honeypots.

Software tools,OS	**Description**	**Specifications**
Linux,Ubuntu 9.10	Honeyd host ,physical machine	Dell OptiPlex GX240; Intel Pentium 4 CPU 1.50GHz; L2 Cache:256KB RAM: 768 MB;Hard disk:18GB; 2 network interface card
Linux,Ubuntu 9.04 (Jaunty Jackalope)	Host machine (virtual honeynet)	Dell Precision WorkStation 370; Intel Pentium 4 CPU 3.00GHz; L2 Cache: 1MB RAM: 2GB ;Hard disk: 500GB; 1 network interface card
Linux, Honeywall Roo (CENTOS 5)	Virtual machine,Honeywall gateway for high-interaction honeypots	Roo 1.4 RAM:512MB ; Hard Disk:40GB Network:3 [2 –bridged(eth0,eth1),1- host-only(eth2)]; eth1 -- > vboxnet0
Linux,Ubuntu 7.10 (Gutsy)	Virtual honeypot	RAM:360MB Hard Disk:20 GB Network: host-only vboxnet0 (public ip)
Windows 2003 Server Standard SP1	Virtual honeypot	RAM:256MB Hard Disk:20 GB Network: host-only vboxnet0 (public ip)
Windows 7 Professional	Remote Management machine	Thinkpad Lenovo T61(laptop) CPU:Intel Core 2 Duo RAM:2GB
Honeyd	Low-interaction open-source honeypot solution	Honeyd 1.5c
Virtualbox	Virtualization software	VirtualBox 3.0.12 for Linux
Snort	IDS	Snort 2.6.1.5
Snort_inline	IPS	Snort_inline 2.6.1.5
Sebek	Data Capture tool based in client-server architecture (Server on Honeywall gw,Clients on virtual honeypots)	Sebek Server 3.0.2(Honeywall) Sebek 3.2.0b client (Linux) Sebek 3.0.5 client (Windows)
Psacct	monitoring user, process activities on Linux honeypot	
Walleye Web Interface	Web based GUI for configuration, administration and data analysis of Honeywall	Walleye-1.2.11

Table2. List of hardware and software resources used in our implementation

In Table 2 you can see the list of hardware and software resources which we have used in the implementation.
As hardware, we use three physical machines: One is used for the Honeyd, and another one is used as Virtualbox host for the virtual honeynet, and the last one is used as the remote management machine.

5.1 Honeyd

Honeyd is a framework for virtual honeypots that simulates virtual computer systems at the network level. It's created and maintained by Niels Provos [10]. This framework allows us to set up and run multiple virtual machines or corresponding network services at the same time on a single physical machine. Thus, Honeyd is a low-interaction honeypot that simulates TCP, UDP and ICMP services, and binds a certain script to a specific port in order to emulate a specific service. For example, we could set up a virtual pop server that seems to run Windows and listens to port 110. According to the following Honeyd configuration template we have a windows virtual honeypot which is running on 193.x.x.x IP address. This "Windows" template presents itself as *Windows 2003 Server Standard Edition* when an attacker wants to fingerprint the honeypot with NMap or XProbe.

```
create windows
set windows personality "Windows 2003 Server Standard Edition"
add windows tcp port 110 "sh scripts/pop3.sh"
bind windows 193.10.x.x
```

When a remote host connects to TCP port 110 of the virtual Windows machine, Honeyd starts to execute the service script *./scripts/pop3.sh.*
With Honeyd, it is even possible to simulate the whole production network. It can create virtual routing topologies which consist of multiple virtual networks with thousands of hosts just in a single machine. Honeyd framework also supports redirection mechanism which can forward connection requests to the services running on a real machine. This redirection feature could be improved and used in building Hybrid honeypot system in order to redirect interesting traffic to high-interaction honeypot machines. Figure 5.1 describes the conceptual overview of the processing network packets by Honeyd. A central Honeyd machine hosts four virtual machines. It receives network traffic sent to the destination IP addresses which belong to the virtual honeypots and replies to their requests.

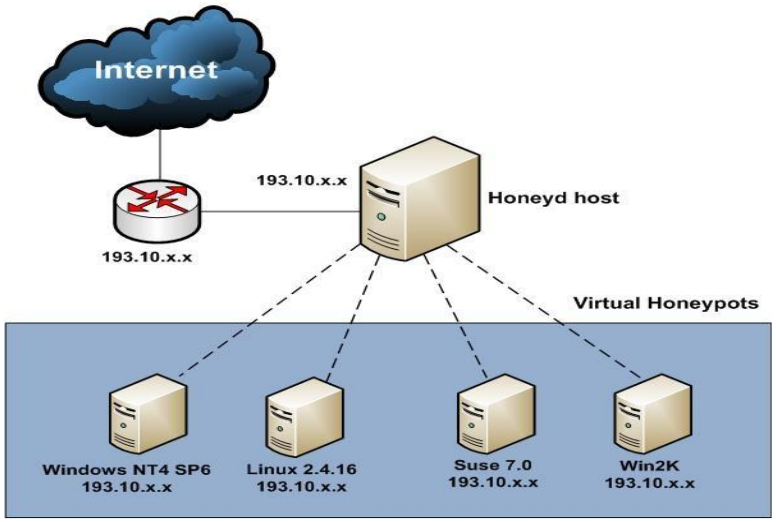


Figure 5.1 Honeyd Framework

Honeyd receives packets to its virtual honeypots via one of the following ways [10]:

1. Configuring network router for the virtual IP addresses that redirect packets to the Honeyd host;
2. Proxy ARP
3. Network tunneling

When the attacker sends a packet to the virtual honeypot, this packet will be passed and forwarded by the network router to the Honeyd host machine. After receiving the packet, the router will check the existence of forwarding address of the virtual honeypot in its routing table. If corresponding route entry exists, the router sends ARP requests for virtual honeypot to determine MAC address of the Honeyd host. This method is called ARP Proxy. If router doesn't find the physical machine to send packets, then ARP requests will be dropped by the router. Tunneling of the network address space to the honeyd host is a more commonly used method in complex networks, especially when you want to deploy a distributed Honeyd network. Honeyd supports the GRE tunneling protocol [10]. When using GRE tunneling, encapsulated packets are sent to the tunneling destination address, and then routed to the Honeyd machine.

In our work we chose to use the ARP proxy method, because the first method wasn't provided by the network administrators regard due to the risk of slowing down the performance of the production network, and third method wasn't suitable for our Honeyd architecture.

Now we want to give brief description about the work mechanism of Honeyd. You can see the Honeyd's architecture in Figure 5.2.

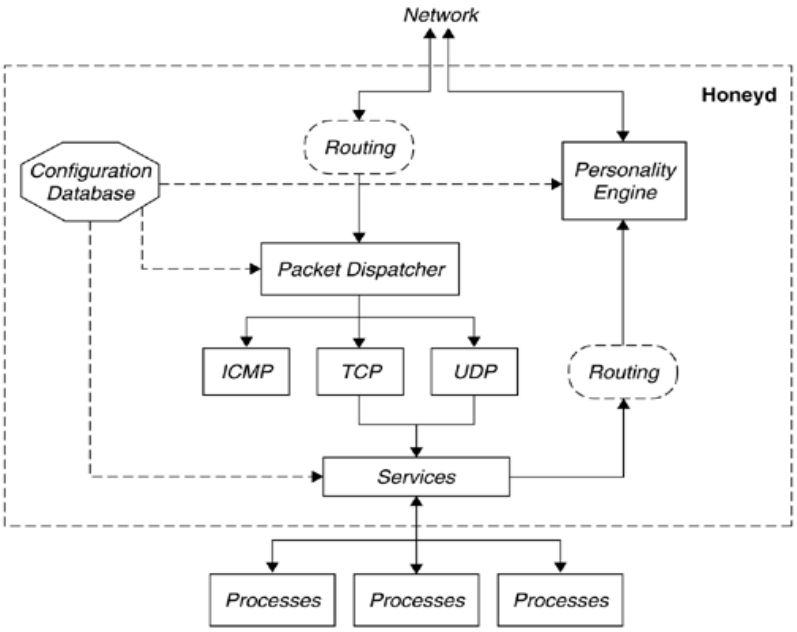


Figure 5.2 Honeyd architecture [3]

Honeyd's architecture consists of components such as a packet dispatcher, a configuration database, protocol handlers, and a personality engine. All incoming packets are processed by packet dispatcher and dispatched to the correct protocol handlers. Before the dispatching process a honeypot configuration will be checked in order to find the destination IP address of the packet. If a configuration found in the configuration database, then the packet will be handled by the protocol handlers, else the packets will be discarded. The services which are configured for TCP and UDP,

create processes to handle the packet. At the last step, all the outgoing packets are processed by the personality engine to match the characteristics of the configured operation system [3]

5.1.1 Installing, configuring and running

Honeyd is powerful, extendable open-source software which is running under different operating systems like Linux, FreeBSD, Mac OS X and Windows. It's more flexible to use Honeyd if you install it under Debian distribution. Thus, we installed the Honeyd 1.5c package on Ubuntu using apt-*get install honeyd [2]*

This package also requires the installation of the following libraries:

• libevent - an asynchronous event library
• libnet – a network library
• libpcap - a packet capture library

It consists of a lot of scripts which can be used to emulate the services that we run in our virtual honeypots.

We have also installed Snort to capture packets and categorize malicious activities for our four virtual honeypots, as well as Mysql database to store all the Snort logs in the database. Before configuring the Honeyd we must be sure that IP forwarding is disabled on the honeyd host. If IP forwarding is enabled, then IP packets which Honeyd receives for the virtual honeypots are forwarded to another computer in the company network. Therefore we used in order to disable the IP forwarding.

echo 0 > /proc/sys/net/ipv4/ip_forward [2]

Before running honeyd we need to ensure that the Honeyd host can answer to all ARP requests which are sent by the router for the IPs of the virtual honeypots. To achieve this we used farpd[2] tool for spoofing the ARP requests. Farpd is made by Niels Povos within the Honeyd project. It listens on the host network interface and responds with the MAC address of the Honeyd for the received ARP requests on the corresponding IP addresses. By using farpd, we could achieve receiving of the incoming traffic through Honeyd's network interface. It allows us to easily monitor, and capture traffic which sent to virtual honeypots. This is achieved by running the following command for four configured IPs

farpd <IP address of virtual honeypot]> -i eth0

where eth0 is the physical network interface of Honeyd machine.

Honeyd uses a simple text-based configuration file where all low-interaction virtual honeypots are specified. It also specifies which IP addresses are used, the open ports and available services of all each virtual machines. Let us start to give a brief overview of the main commands which we have used in our configuration file. Here you can see a part of our configuration file for a virtual honeypot:

```
##### Honeyd Configuration File #####
create linux
set linux personality "Linux 2.4.16 - 2.4.18"
set linux default tcp action reset
set linux default udp action reset
set linux uptime 5184000
add linux tcp port 110 "sh /scripts/unix/general/pop/pop3.sh"
add linux tcp port 25 "sh /scripts/unix/general/smtp.sh"
add linux tcp port 21 "sh /scripts/unix/linux/ftp.sh"
add linux tcp port 22 open
bind A.B.67.128 linux
```

create command creates a template called "linux" and *bind*s the honeypot's IP address to this template. *set* and *add* commands change the configuration of the template. With *set* command, a personality *"Linux 2.4.16 - 2.4.18" from* the Nmap fingerprinting file is assigned to the created template. The *uptime* of the host shows how long time the system has been running. We spoof the uptime to be *5184000* seconds which is equal to 60 days. The *add* command opens the ports on the virtual honeypot, and specifies which service should run on each port. When an attacker establishes connection to port 21, honeyd starts to execute the service script *ftp.sh* [2]. The value of *reset* defines that by default all ports are closed for the tcp and udp protocols. There are also *open* and *block* actions which determine the reaction for each of the protocols. When *open* action is used then all ports will be open by default, but in case of *block* action, all packets will be dropped by default.
Now it's time to run honeyd with our configuration file. Honeyd usually runs as a user *nobody* which is more secure than other users.

honeyd -d -i eth0 -f honeyd.conf -p nmap.prints -x xprobe2.conf -a nmap.assoc -0 pf.os -l /var/log/honeyd/honeyd-packet.log –s /var/log/honeyd/honeyd-service.log <IP1, IP2, IP3, IP4>

At this point, Honeyd starts listening to the *eth0* interface and answering to the packets for the four IP addresses *<IP1, IP2, IP3, IP4>* of the configured virtual honeypots which are from the same subnet on the organization's network.
We used ping, Nmap, telnet and traceroute tools in order to test that the honeyd installation is working correctly and receiving the network traffic.To get more detailed malicious activities, the Honeyd machine is placed outside of the centralized corporate firewall, so it can be exposed to all types of attackers, worms and other threats. Taking into consideration that virtual honeypots are in the same subnet as the corporate computers, then it can be dangerous for the corporate network. Therefore we used IPTables to secure our virtual honeypots. The firewall allows all inbound traffic to the virtual honeypots, but limits the outgoing traffic from the honeypots. We only allowed www, http, and ssh access from the remote management system to the hosting machine in order to monitor processes, analysis and backup logs on hosting machine.

iptables -A INPUT -p tcp -s A.B.66.230 -d A.B.66.129 --dport 80 -mstate --state RELATED,ESTABLISHED -j ACCEPT

iptables -A INPUT -p tcp -s A.B.66.230 -d A.B.66.129 --dport 443 –m state --state RELATED,ESTABLISHED -j ACCEPT

iptables -A INPUT -p tcp --tcp-flags ALL SYN - s A.B.66.230 -d A.B.66.129 --dport 22 -m state --state RELATED,ESTABLISHED -j ACCEPT

By using virtual routing topologies honeyd allows simulating several network subnets which have thousands of computers. But due to the risks for the company network, we used only four IP addresses in order to realize our experiments with Honeyd.

5.1.2 Logging

Honeyd framework has capability to provide different ways of logging network traffic.
Native Honeyd logging provides packet level logging with (-l) command line option, and service level

logging with (-s) which we used in the above-mentioned command line for running honeyd. In this case, honeyd logs to separate files: packet-level and service-level. Packet-level log file contains the timestamp when a packet was received, which protocols and ports are being used, source IP address and port, destination IP address and port. If the connection is established, then honeyd will log the information about the start and end time of the connection, and transmitted bytes. Below you can see example output from packet-level log file:

```
2010-05-11-15:35:00.1748 tcp(6)   S 207.46.204.178 56826 193.10.67.150 80    [Windows XP SP1]
2010-05-11-15:36:01.0798 tcp(6)   E 207.46.204.178 56826 193.10.67.150 80:   207 528
2010-03-14-17:14:18.9020 tcp(6)   - 190.167.32.173 59630 193.10.66.128 22:   60 S [Linux 2.6 .1-7]
2010-05-11-15:45:59.9397 icmp(1) - 64.72.33.158          193.10.66.128:     8(0): 61
```

The first field displays the time when the packet was received. The second field contains the protocol information- tcp and icmp. Next field displays connection type which either may be *S* (start of new connection), *E* (the end of the connection) or - (connection neither S nor E).Next fields represent source IP address, source port of the packets, and destination IP address, destination port of the virtual honeypots. The last fields represent how much transmitted bytes sent and received by Honeyd, and type of operating system which is identified by passive fingerprinting [2].
Service level logs are based on the output of the emulated scripts like web, telnet, IIS and etc. Compared to packet-level logging, service logs give us detailed information about the incoming traffic. According to collected honeyd service logs, we can give an example of a service-level attack.
In our honeyd configuration file, we bound script smtp.sh to the tcp port 25 which is emulating a mail server. From the generated smtp-.log, we can see that an attacker interacts with the emulated mail server in order to probe for open mail relays. And we can conclude that it is a spammer!

```
/var/log/honeyd/honeyd.log
Apr 21 03:46:38 armakedon honeyd[32706]: Connection request: tcp (114.45.56.176:4049 -
193.10.66.128:25)
Apr 21 03:46:39 armakedon honeyd[32706]: Connection established: tcp (114.45.56.176:4049 -
193.10.66.128:25) <-> sh /usr/share/honeyd/scripts/unix/general/smtp.sh
Apr 21 03:46:39 armakedon honeyd[32706]: Killing attempted connection: tcp (114.45.56.176:4196 -
193.10.66.223:25)
/tmp/honeyd/smtp-.log
Wed Apr 21 03:46:39 CEST 2010: SMTP started from Port
HELO 193.10.66.128
MAIL FROM: <z2007tw@yahoo.com.tw> …
```

Honeyd also supports using sniffer on the hosting machine to capture the incoming and outgoing traffic on virtual honeypots using the IDS alerts of the sniffers. Therefore, we used Snort IDS in parallel with honeyd logging and this combination of Honeyd and Snort logs gave us very useful information about the malicious activities. We used Honeydstats [3] tool to get all statistics of the threats which has been collected by honeyd, and showed them in a readable form. Honeyd logs will be analyzed in more detail in the next chapter.

5.2 Virtual honeynet

In this project we have built and tested our own virtual honeynet for attracting the attackers to interact with real (not simulated) operating systems, services and programs. Collecting information about the

"blackhats" and their methods that is used to compromise our honeypots would be useful building better security mechanism for the corporate network.
Virtual honeynets are not a new technology. It is actually based on the concept of traditional honeynet technologies. The idea was to run Honeynet on a single machine, and this idea was realized using virtualization software. Based on the proposed architecture in Figure 4.3 we are going to deploy our virtual honeynet. The main reason of choosing virtual honeynet was lack of hardware resources, thus we decided to use Virtualbox [13] as a virtualization software, and Honeywall roo [14] to deploy the virtual honeynet. For our implementation, we benefited from the honeynet chapters within Honeynet project [7] that also used virtualization softwares like VMware, Qemu and etc. in a honeynet deployment on a single physical machine. As an example, in the Pakistan Honeynet project [26], Vmware has been used for deploying a virtual honeynet. As we already mentioned, our virtual honeynet was implemented using Virtualbox on a single Pentium 4 CPU 3.00GHz, 2 GB RAM physical machine running Ubuntu 9.04. The operating system of this physical machine is called as the *host operating system* in Figure 4.3.
Setting up the Virtual Honeynet proved to be a bit more time consuming and complex than expected although we used some instructions given at honeynet.org. Main complexities were related to the configuration of the Honeywall OS with three network interfaces to handle the traffic from the production and honeynet network, and also compiling, installing the data capturing tool Sebek [3] on the guest machines. Below we are going to present the details of the implementation and configuration of our honeynet.

5.2.1 Virtualbox honeypoting

Virtualization software helps reducing the total cost of ownership of the IT infrastructure of organizations. Using virtualization can help them replace their servers with virtual machines on a single physical machine. Some organizations have been developing their own virtualization solutions which many of them are free and open source.
We first started our experiment with VMware Workstation trial version. After two weeks, we decided to shift to the free and open source virtualization software since VirtualBox has the following advantages over VMware Workstation [13]:

- Free to Use and Open Source
- Less Resource Usage in Host
- Better performance of virtual machines
- VirtualBox Supports VMDK
- Available patches and updates
- Less Saving and Resuming time of Virtual machine

It was easy to switch to the new virtualization software, thus we simply imported the existing VMware-based VMDK files into the Virtualbox, and selected them as hard disks for our new virtual machines. Consequently, there was no need to install Operating systems and softwares on the new virtual machines. VirtualBox also provides less memory usage and better performance for our host and virtual machines; it allows simultaneously running three virtual machines with using less hardware resources.
In our implementation, we installed Virtualbox version 3.0.12 and created three virtual machines upon it: Honeywall and two Honeypots (Windows, Ubuntu). You can see these virtual machines in Figure 5.3. Honeywall boots from an iso-image Honeywall Roo 1.4 and its operating system is based on CentOS 5. It is configured with 512 MB RAM, 40 GB storage and three network interfaces: one host-only interface and two bridged interfaces. Linux-based Ubuntu Honeypot was configured with 360 MB

RAM, 20 GB storage and one host-only network interface, but Windows Honeypot with 256 MB RAM, 20 GB storage and one host-only network interface.

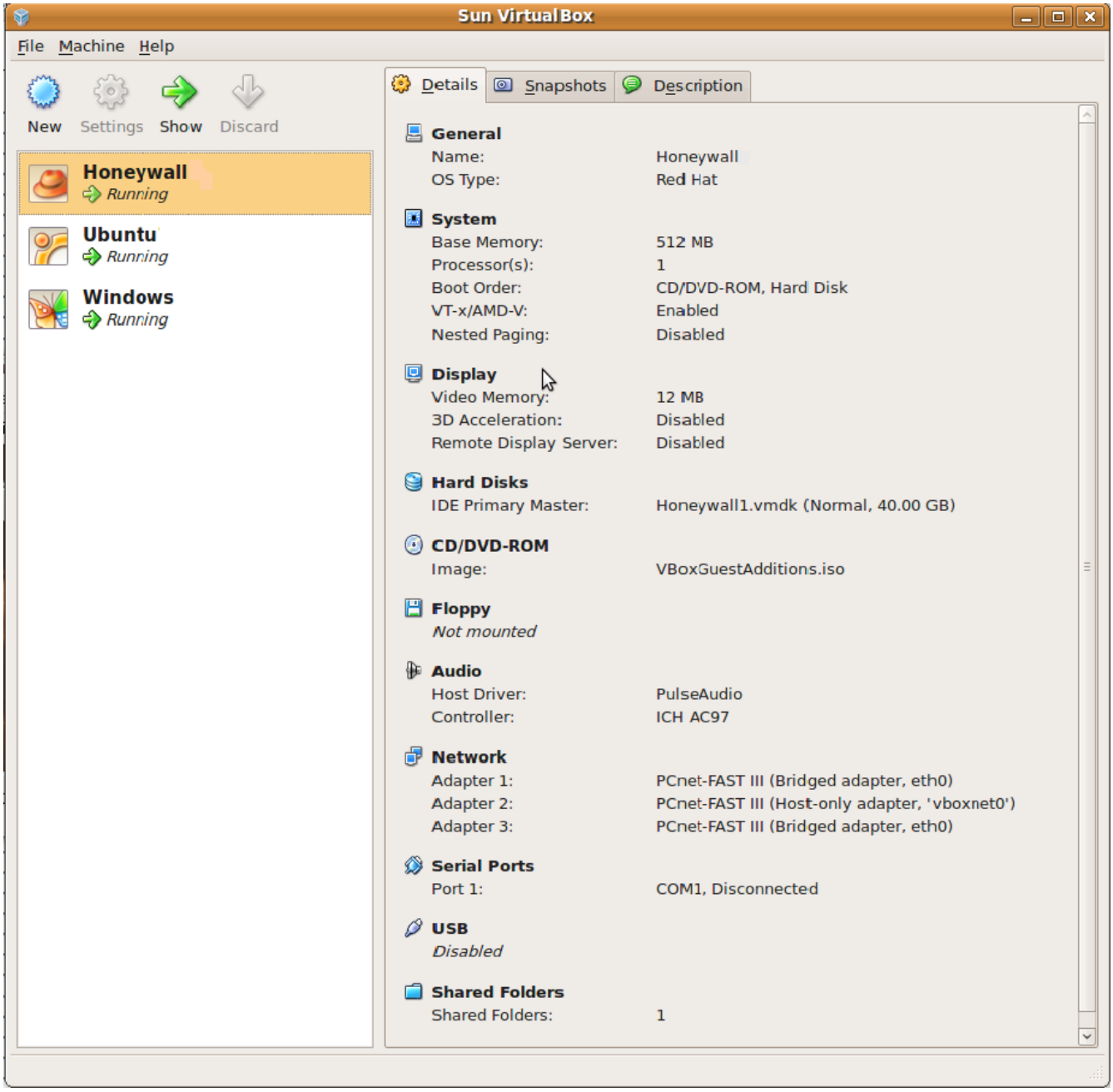


Figure 5.3 Installed virtual machines on Virtualbox

VirtualBox can provide different networking modes for the virtual PCI Ethernet cards which assigned to the virtual machines. Up to four virtual network cards can be configured for each virtual machine. In our experiment, we only used the following three of the virtual networking modes and assigned these modes to the selected PCNet FAST III virtual network card for all of our virtual machines, because this network card is supported by nearly all operating systems.

- **Network Address Translation (NAT)**

In the NAT mode, virtual machines can share the IP address of the host system which it allows to enable internet access to the virtual machines. In our virtual honeynet, we used NAT mode only for testing and configuring our virtual honeypots.

- **Bridged networking**

In this mode, host system plays a role of transparent bridge for the virtual machines in order to send their packets to the internal network or honeynet network. In other words, bridged networking connects virtual network interfaces of the virtual machines to the physical NIC of host machine.

- **Host-only networking**

Host-only networking creates a virtual network which is completely resided within the host machine. In this mode, host and virtual machines belong to one network and has no internet access [13] and it also allows us to control the traffic on the honeypots which is routed through Host machine.Indeed, virtual machines with the host-only network can communicate to each other without the need for the host's physical network interface. In our implementation, we created one host-only network interface *vboxnet0* on host machine and connected the NICs of the virtual honeypots to this interface.

5.2.2 Honeywall Roo

As we mentioned before, Honeywall is a main component of the GEN III honeynet, since it provides the main tasks of the honeynet such as Data Capture, Data Control and Data Analysis.“Honeynet Project” provided a bootable CD-ROM which is called Honeywall Roo.It allows network specialists and administrators to easily deploy and manage honeypots.This CDROM is based on CentOS distribution. The previous version of the CD-ROM -Eyore had limited capabilities and features, especially had poor Data Analysis capability. But the new has an improved administration capabilities, and data analysis functionality. Thus this system gives to administrators more flexibility and easy control. Honeywall Roo includes the following security tools:

- **Tcpdump:** Packet analyzer
- **Sebek:** Data capture tool
- **Snort**: Intrusion Detection System (IDS).
- **Snort_inline**: Intrusion Prevention System(IPS)
- **Hflow2**: A data correlation tool for Honeynet data analysis.
- **P0f:** Passive OS fingerprinting tool
- **Walleye Web Interface:** a web based interface for Honeywall configuration, administration and data analysis

Honeywall installation is fully automated. We downloaded the latest release of Honeywall- Roo 1.4 ISO image from the https://projects.honeynet.org/honeywall/ and burned it to the CD-ROM that is bootable. Then we booted our virtual machine off this CD-ROM .Once the installation was complete, the new Linux-based OS booted and configuration process started.
Honeywall comes with two user accounts (*roo* and *root*) which they are used to login to the system and Walleye GUI. *Root* login is not accessible by default so one has to login as *roo* and then use *“su -”* command to have root access.
There are two methods for configuring the Honeywall:

- Dialog menu interface. It is more common configuration method which is mostly used in the initial

setup of Honeywall. This interface opens automatically during the first login as root after the installation. It's also possible to run it by typing the command *menu* from console.

- Manually create honeywall.conf. Honeywall Roo comes with default configuration file

honeywall.conf which is ASCII text file. It contains the configuration parameters the OS and Honeywall will be using. You can modify that file and create your own honeywall configuration.
Honeywall doesn't use /etc/honeywall.conf as a runtime configuration file, it reads configuration variables that are maintained as files in the directory /hw/conf .These files are generated using the "hwctl" utility [7]:

#/usr/local/bin/hwctl -s -p /etc/honeywall.conf

We used *Dialog Menu Interface* in the configuration by setting the IP addresses, DNS servers of the Honeypot and Management Interface, accessible ports on honeywall, incoming and outgoing connection limitation, snort inline, sebek parameters, blocked IP addresses to honeypots and etc.

Virtualbox networking

According to our virtual honeynet architecture in Fig 4.3, Honeywall must have three virtual NICs:

- eth0 – connected to the external network, bridged to host's eth0 physical NIC
- eth1 – connected to the honeynet through vboxnet0 host-only virtual interface
- eth2 – interface for remote administration and management, bridged to Host

Eth0 and eth1 are together used to make br0 virtual bridge interface [7]. Eth2 is bridged interface to host and used for remote management of the Honeywall, especially for SSH access and managing the Walleye GUI. But there is restricted access to remote management interface, thus it is only accessible to the computers in the same subnet.
Host machine's eth0 interface, Honeywall management interface eth2 and the virtual honeypots 1, 2 has publically assigned IP addresses from the same subnet which we are running Virtualbox on.
Honeywall's eth0 is bridged to Host machine's eth0; eth1 is connected to virtual honeypots through vboxnet0 virtual network interface. We created vboxnet0 as host-only network interface on Virtualbox, and attached it to the virtual honeypots. Thus, virtual honeypots are accessible to internet through honeywall (roo).

Walleye Web Interface

Walleye Web Interface is a web-based Graphical User Interface that is used for Honeywall configuration, administration and data analysis. We used this web interface in order to analysis the inbound and outbound traffic through a web browser client by typing https://193.x.x.x (where 193.x.x.x is the Public IP address).For security reason, this interface is accessible over port 443(HTTS) . Walleye has two main functionalities: Data Analysis and System Administration. The Data Analysis is used for analyzing real-time flows, overview of incoming and outgoing flows, Sebek based data, alert flows by the Snort IDS, *whois* capability, and activity summary per day. Walleye also provides downloading the packet data in pcap format which we used for further analysis with Wireshark on our remote management machine. System Administration allows remote Honeywall administration, and also provides access to the Honeywall configuration, thus all the settings can be updated from this interface too. In Figure 5.4 you can see the examination of all connections for one day: aggregated connections

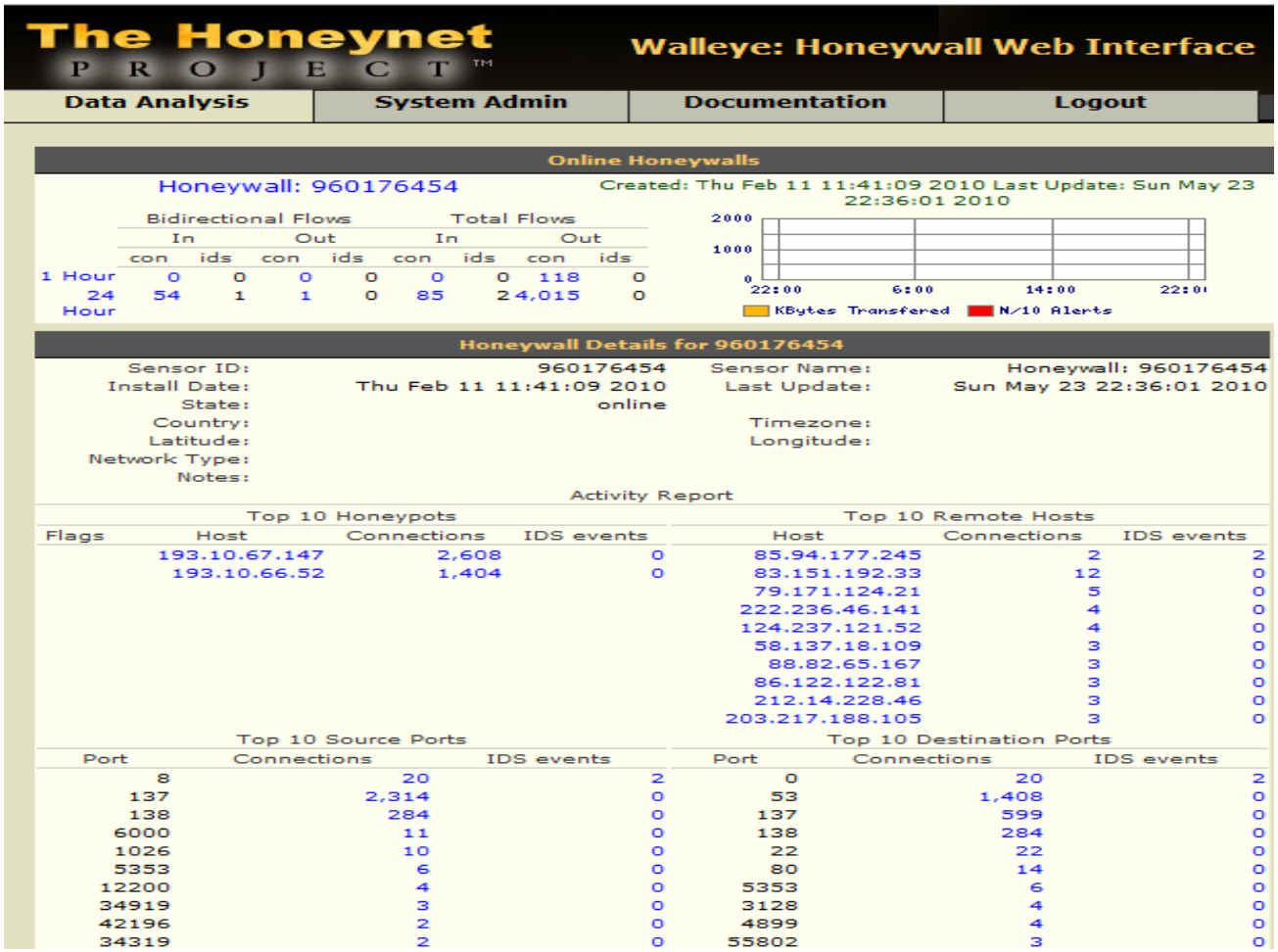


Figure 5.4 Detailed overview of Honeywall activity

to each honeypot per day, number of IDS events, and most connected destination and source IP addresses and ports. Moreover we can analyze one connection in detail for the particular IP address. In the following screen you can see that each line contains information about protocol type, number and bytes of the packets, and OS type of source IP address of the connection.

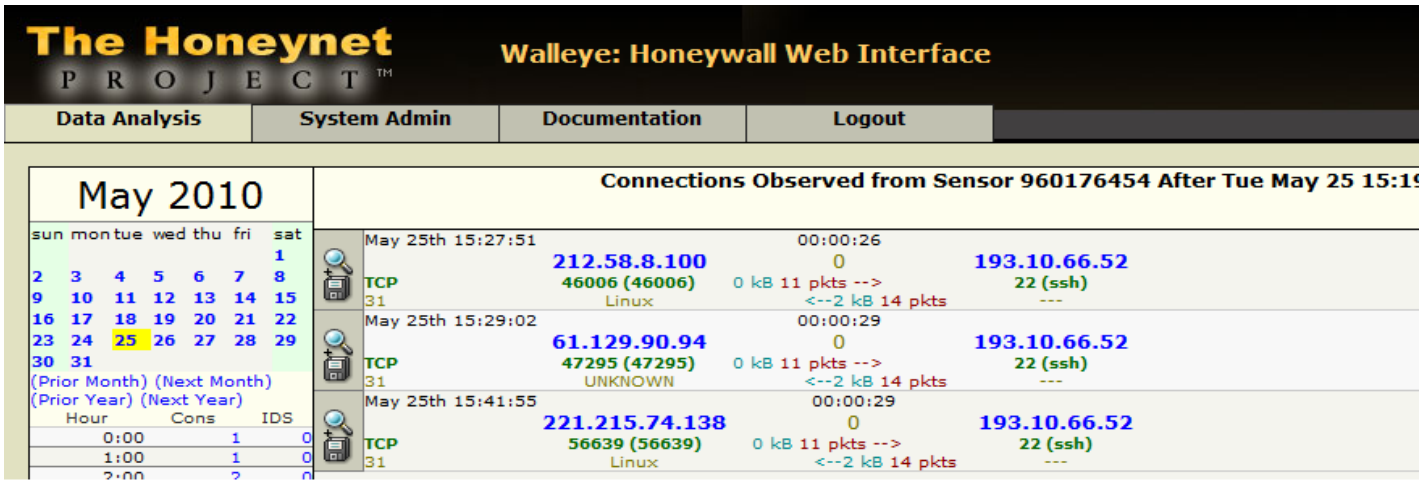


Figure 5.5 Detailed information for a flow

5.2.3 Snort IDS and Snort-inline IPS

Snort as an Intrusion Detection and Prevention System is integrated into Honeywall 1.4. It is an open-source IDS, rule- and signature-based engine that can be run in one of the following modes:

- Sniffer Mode

In this mode Snort is used as packet sniffer and displays IP headers on the screen.

- Logger Mode

All packets are logged into the file and can be used for further analysis.

- Network Intrusion Detection Mode

The core mode of Snort. All incoming packets will be analyzed based on the user-defined rules and signatures. Snort will log, detect and alert if there is any anomaly detection in the packets then Inline Mode. In this mode, Snort acts as an Intrusion Prevention System (IPS) which is called Snort-inline. It resides on the Honeywall where the packets are analyzed and monitored using iptables in order to control outgoing packets from the honeypots. If the honeypots are compromised by worm or attacks, Snort Inline will prevent the attackers from compromising other machines in the same network. By using inline mode, Snort is able to perform the *modify, reject, ignore* actions against the packets which content matches the known attack. *Modify* action is very important for a honeynet. Thus it allows modifying the content of the attack packets and rendering them harmless. In this case intruders can't be aware of the changes in the packets. Below is a example of inline rule:

```
alert tcp $HOME_NET any -> $EXTERNAL_NET 53 (msg:"DNS EXPLOIT
    named";flags: A+;content:"|CD80 E8D7 FFFFFF|/bin/sh";
    replace:"|0000 E8D7 FFFFFF|/ben/sh";)
```

In this example, snort-inline finds the packets which carry content in their payload in order to execute a DNS Exploit and get bash shell *"/bin/sh"*. According to this rule, the packet is modified and replaced with "/ben/sh" instead of "/bin/sh". In this case attacker cannot understand why the exploit wasn't successful. As a result, the attacker is prevented from compromising the DNS server. Snort IDS and IPS are also integrated in the Walleye management interface which provides a nice view of the IDS signature alerts and allows analyzing the attack data remotely. Below is a screenshot of the Walleye displays the logs taken from the IDS sensors of 21st March, 2010. The attempts from various geographical locations try to get access and compromise our honeypots.

5.2.4 Sebek as a Data Capture tool

Sebek is a data capture tool that is used to capture all attackers' activities (keystrokes, file transfer, encrypted traffic, and commands). It is based on client-server architecture, and it can be installed as a linux kernel module (LKM) on Linux and as an OS kernel driver on Windows [25]. Then, the data captured from the honeypots by the sebek clients will be sent to the Sebek Server which collects and processes the received logged activities in the form of Sebek packets (refer Fig.5.6). These packets are hidden from the attackers. Sebek itself can be hidden and configured in such a way that attackers cannot detect it. In our experiment, the Sebek Server was running on the Honeywall, that is, the honeynet gateway which all incoming and outgoing traffic to our honeynet passes through. But it can

also be installed and independently run on a remote machine. Sebek tracks all data activities related to "read" and "write" system calls and operates as a part of the kernel. Therefore it is called a kernel-based data capture tool and it uses some capabilities of the LKM based rootkits. Analyzing Sebek data from the packet files on Honeywall was possible using the Sebek Server scripts*: sbk_extract*, *sbk_ks_log.pl* and *sbk_upload.pl*. Thus the *sbk_extract* and *sbk_ks_log.pl* retrieves the attackers' keystrokes from the collected sebek data on the honeypots, and *sbk_upload.pl* loads this data into the database. Analysis of Sebek data is also supported by the Walleye Graphical User Interface.
We installed Sebek clients on each virtual honeypots (Linux, Windows), and configured them to send attackers' activities to the sebek server. However, it was not so easy to setup and configure sebek clients on the virtual machines in our experiment, especially on Linux machine. Because the installation of Sebek client on Linux required building and compiling the sebek library depending on the kernel version, and it made lots of problems related to the kernel.

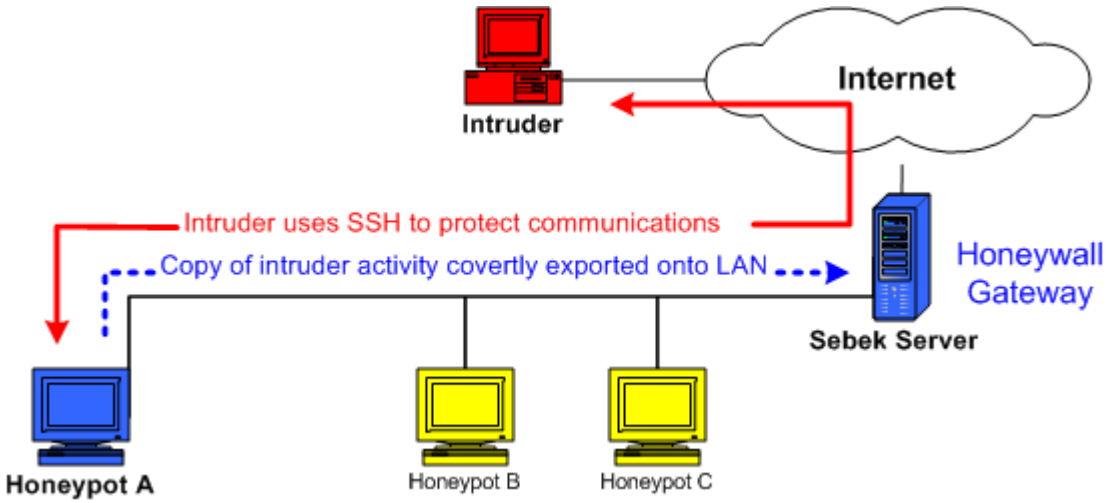


Figure 5.6 Sebek Client-Server Architecture [25]

5.2.5 Configuring virtual high-interaction honeypots

In this section we will talk about setting and configuration of virtual high-interaction honeypots in our virtual honeynet implementation. We set up the virtual honeypots in two steps. In the first step, we installed the operating systems on our virtual machines, which is very similar to installing an operating system on a physical machine. In the second step, we installed additional software and vulnerable applications in order to attract more attackers to the honeypots and collect information about them. Since we deployed our high-interaction honeypots using virtualization software it was not so easy to make concrete decision on choosing the operating system for the honeypots. To choose the "right" operating system depends on the need and the available software/hardware version and parameters. Basically, the honeypots within Honeynet project [7] were implemented based on Microsoft Windows and Linux operating systems. Machintosh and others exist too, but they are not so common on the Internet. There are also some other honeypots that has been deployed on mobile phone, network printer OS. We decided to use Windows and Linux operating systems in order to study attacks against different operating systems and the tools, methods used by attackers. Linux systems have more interesting opportunities for data gathering mechanisms which make the data logging easier compared to the Windows systems. Also, the source code of the windows systems is not freely available which means that it isn't possible to have changes to the operating system. And, the logging files which are

generated by user space programs can be visible to all users of the windows system, and attackers as well. Therefore we used a Sebek Client as a hidden data capture tool to log attackers activities on the Windows honeypot and send them to the Sebek Server which is located on Honeywall (roo) machine. Sebek can be hard detectable by attackers. In our experiment Ubuntu 7.04 (Gutsy) was used as a linux based honeypot, Windows 2003 Service Pack 1 as a windows honeypot. Virtualbox can run both of these operating systems.

5.2.5.1 Honeypot 1 -Ubuntu

Ubuntu is one of the most widely used Linux Distribution which provides a good platform to study zero-day attacker exploits. We successfully setup the older version of Ubuntu which is no longer supported, and installed the vulnerable web applications, configured commonly used services such as FTP, MYSQL, SSH, Apache and PHP on it to make an interesting target for attackers. For example, all incoming connections to SSH service (port 22) on honeypot were not restricted, MYSQL, PHP default ports were open to incoming connections, but outgoing connections were restricted by Honeywall in order to prevent attacking to another machines in the network using our honeypot.

First we started to deploy the honeypot by making the SSH service accessible to the attackers without creating any specific user accounts. After two weeks experiment, we observed only failed connections in the OpenSSH logs. These logs contain all the information about the SSH connection except login passwords. OpenSSH doesn't record passwords for login attempts by default since it is insecure to store password in plaintext, and possibly unethical. We found out that it is possible to log ssh passwords and other related information to a single file by patching the *auth-passwd.c* file from the Openssh library. Thus, we edited the *auth-passwd.c* and added the following code:

```
auth_password(Authctxt *authctxt, const char *password)
{
      struct passwd * pw = authctxt->pw;
      int result, ok = authctxt->valid;
  +    if(!sys_auth_passwd(authctxt, password))
  +   {
  +         struct tm *today;
  +         time_t  localTime;
  +         char  timeString[100];
  +         localTime = time(NULL);
  +         today = localtime(&localTime);
  +         strftime(timeString, 100, "%D %r", today);
  +         FILE *Logged;
  +         Logged= fopen("/usr/src/linux-headers/sshd_logs","a+");
  +         fprintf (Logged,"RemoteIP: %s Date: %s | user: s, password: %s\n",
  +         get_remote_ipaddr(),timeString, authctxt->user,password);
  +         chmod("("/usr/src/linux-headers/sshd_logs", 0600);
  +         fclose (Logged);
  }
```

Then we re-compiled the Openssh library, and started to log passwords, as well as usernames, log time and origin of attempted attack into the log file *sshd_logs*. Taking into consideration that password recording can make security problems in organization network, therefore we decided to protect and hide that log file deep within the system. We hide it under the folder: */usr/src/linux-headers/*

The analysis of this file gave us statistics of most attempted over the most attempted login combinations (username/password). By using this information we selected most common eight user accounts in the log file, created them with weak passwords and gave them the shell access in the guest system. The idea was to allow attackers to easily login into our honeypot using the non-privileged user accounts which we created. Some of these user accounts were easily guessed and successfully broken

by the attackers which used brute-force dictionary attacks. In the next chapter, we are going to analyze successful SSH connections in more detail.
As a Data Capture tool, we used Sebek 3.2.0b client in order to monitor the attackers' activities. Except Sebek Client we also installed the psacct package which contains utilities such as "*ac"*, "*lastcomm"*, "*accton"* and "*sa"* for monitoring and logging the users' activities [28]. This program works in the background of the system and records the detailed logs of each command which was run by the attackers, as well as the resources including CPU, time and memory that have been used by users. The *ac* command shows the statistics about how long users were logged on. The *lastcomm* - displays the previous executed commands by the attackers. The *sa* command prints out the summary of the commands that are executed. In parallel with *psacct*, we also used system logging and command history which provided us with beneficial information about attackers. Thus we parsed the history file (/.bash_history) and configured bash configuration file (/.bashrc) under the home directory of each user. The history file stores only the history of the executed commands.It does not provide us timestamp for the commands. Therefore we decided to edit the *.bash_history* file and add new variables, and change the default configuration in order to provide detailed logging to the history file. The following lines have been added into the bash configuration file (/.bashrc) for each user:

```
#limiting the number of these lines to be retained in memory
export HISTSIZE=100000
# timestamp for executed commands
export HISTTIMEFORMAT="%h/%d - %H:%M:%S"
# bash will never remove the lines in history file
unset HISTFILESIZE
```

Also we removed "*export HISTCONTROL=ignoredups"* line to allow the duplicate commands to be recorded into the history file. This allows us to analyze the most common commands ran by the attackers.
Some attackers are smart, first clear or stop the logging process when they login to the victim machines. Although we used psacct utility *lastcomm* as an alternative keystroke logging tool, it provides short information about the previously executed commands of the attacker compared to the bash history file. Thus, we also secured the bash history logging in order to prevent the intruder from removing the bash history file under the home directory of each user. The *psacct* and *bash history* logging on our ubuntu honeypot gave us comprehensive insight into what attacker ran, which command and at what time.

5.2.5.2 Honeypot 2 – Windows

As a second virtual high-interaction honeypot we choose a guest machine that was running Windows 2003 Service Pack1.To offer some "honey" for the attackers, we set up an insecure web server and deployed common applications and services with some open ports on the Windows honeypot. We installed XAMPP 1.6.6 which is cross-platform web server package containing tools Apache 2.2.8, MySQL 5.0.51, PHP 5.2.5 + PHP 4.4.8, OpenSSL 0.9.8g, phpMyAdmin 2.11.4 and FileZilla FTP Server 0.9.25.We used old versions of all applications and thus should be insecure. Also due to insecure configuration, these tools can be vulnerable to adversaries. The following vulnerabilities are known for each application. XAMPP installs the above-mentioned tools with insecure default passwords which make it easier for remote attackers to get access via using (1) a blank password for the "root" account in MySQL installation (2) phpadmin is accessible by network (3) XAMPP demo page is accessible by network (4) Default user/password of FileZilla and Mercury are known; All of

these points can make huge secure risk for the real machine and the applications which is installed within XAMPP environment. After installing and configuring XAMPP, we set up SMF 1.0.5 forum which is already outdated version [29] and copied the SMF files to the folder C*: /xampp/htdocs/forum* in the honeypot machine. By using phpmyadmin then we created a new MySQL user and database for SMF to use.

Figure 5.7 SICS forum

There are some common applications and services that are running on Windows and open network ports. We simply deployed some of these applications and services in order to make our honeypot more attractable and realistic for the attackers and also monitor exploits against those applications, such as: KaZaA (1214), Doom Game (666), Yahoo Messenger (5010), Internet Relay Chat (7000), HTTPS (443), Telnet Server, RPS, Microsoft-DS.

Compared to Linux there are many data capture and monitoring tools that you can use to monitor your Windows honeypot. But one thing is remarkable that most of the console keystroke loggers initially have been developed for Unix/Linux, later on for the Windows platforms. Sebek is also one of these data capture tools. Sebek 3.0.5 client is used as a data capture and monitoring tool in the Windows honeypot.

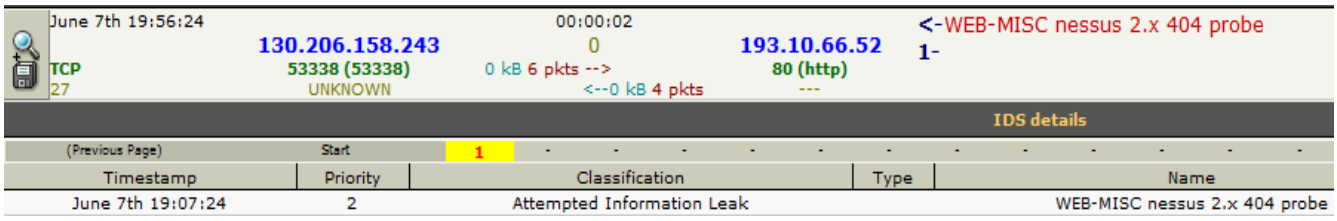
Chapter 6

Experimental results

This section describes the analysis of the attack patterns found on the honeypots and presents an overall statistical analysis of the results gathered from the combination of low- and high-interaction honeypots. Our honeypots were online for a period of approximately 120 days from March to July of 2010. During this period we received over 150,000 identified attack connections. These results provide the better insight to the readers about what was observed in our honeypot experiment.

6.1 Forensic analysis

Before starting to analyze attackers, the best way is first to study how or where they begin to launch attacks. Most of the experiment results show that they normally start with information gathering about their targets, and then determine what vulnerabilities exist before starting to exploit. As in our experiment, most attacks initially involved collecting the information. They usually use different port and vulnerability scanners, such as nmap, nessus, nikto and etc. to find open ports and vulnerabilities of their victims. If you look at the below Snort alert, you can see that the attacker uses Nessus to probe the *http* before beginning to launch an attack. After the scanning by Nessus, the attacker reviews the results, identifies the existing vulnerabilities and open ports, and then starts to exploit the existing vulnerabilities. He first probes the port 80, and then tries to get access into the honeypot machine. The same attacker also attempts to compromise other honeypots in our experiment. But how was the attack launched, and how does it work?



In order to find out how the attacks were launched, to determine the motivation and intention of the attackers, to use the knowledge to prevent future occurrences, we decided first to use forensic analysis. We performed this analysis by our remote management machine with minimum impact on the honeypot environment, thus attacker was not able to notice that he has been watched. According to Weise and Powell [30] computer forensics is, "The capturing, processing, preservation, and analysis of information obtained from a system, network, application, or other computing resource, to determine the source of an attack on those resources." Forensics is a procedure in any environment that contains the chain of processes used for methodically gathering, analysis and preservation of evidence in the situations where you know that an attack has occurred. It also creates facts that can be presented in a legal proceeding. The term "evidence" in forensics analysis means any kind of data, log files, electronic documentation, disk images, generated reports and so on that are collected during forensic investigation. The goal of collecting evidences not only can help to determine the source of attacks, also can help to solve the problems arising from the attack. Forensic analysis can also lead pursuing prosecution which it takes considerable time and effort, because computer images and data as evidences can be admitted in a court of law. Our analysis is entirely based on the collected log files from the honeypots.

6.2 Observed attack cases

In this section we will talk about the attacks against our honeypots and give some examples of them. An important fact is that we could observe malicious connections immediately after launching the honeypots in the network. Most of these connections occurred on the low-interaction honeypots. We examined the IP addresses of attack sources and found out that they are continually changing. Sometimes attackers can send the packets from the forged source IP addresses which makes it harder to identify their original source address. Most common activities which we observed were TCP, UDP and ICMP port scans by intruders in order to check the vulnerabilities on the operating systems. Some connections to the honeypots were successfully established and carried out different attack cases, such as BruteForce, Cross Site Scripting, Remote File Include, Spamming and others. Some of these attack cases were observed once or repeated many times during our observation.

Port Scanning

Port Scanning is often used by system administrators to verify the security level of their networks and by attackers to find the vulnerabilities. In fact it is legal to use port scanning, because you are accessing something that is public. It is like ringing the doorbell to confirm if someone is in the house. Port scanning doesn't mean scanning only TCP ports. Nevertheless, UDP port scanning is used more often as well. The basic attacks we observed from the snort log files were TCP, UDP and ICMP port scans. These scans have been performed by different scanner tools such as NMAP, Portsweep in order to scan multiple hosts for a specific open port. A portscan is used to scan open ports on a single target host.UDP scanning is slower than TCP scanning, therefore many system administrators usually do not secure these ports. Such scans are more useful for the attackers, thus they send empty UDP datagrams to the target port. If the port is closed, then the attacker will receive "ICMP Port Unreachable" message. If open or filtered, then an error message will be sent back or incoming datagram simply will be ignored. Thus, the attacker can determine which ports are open or closed. It is also useful to scan the undocumented UDP ports in high range and find out hidden services on these ports. We observed this case when intruders were scanning UDP ports in a higher range.
During the log analysis we observed that our honeypots were scanned from different IP addresses by some variation of Dfind and also many ICMP echo requests were sent by the CyberKit 2.2 software [31]. Dfind is a known port scanner usually used by attackers to scan old version IIS Servers and web banners. The IIS server on the honeypot has also been scanned continuously by DFIND and had requests for an unknown resource called w00tw00t.at.ISC.SANS.DFind:). Below is a part of the request made by DFIND port scanner:

```
GET /w00tw00t.at.ISC.SANS.DFind:) HTTP/1.1
GET / HTTP/1.1
Accept: application/x-ms-application, image/jpeg, application/xaml+xml, image/gif, image/pjpeg,
application/x-ms-xbap, */*
Referer: http://www.dunkels.com/adam/tfe/
Accept-Language: fr-FR
User-Agent: Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.1; WOW64; Trident/4.0; SLCC2;
.NET CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR 3.0.30729; Media Center PC 6.0)
Accept-Encoding: gzip, deflate
Host: tfe.c64.org
Connection: Keep-Alive
```

Cyberkit 2.2 is the software installed on Windows machines and used by the ping command to send ICMP echo requests to determine if a host is active or not. By using the Cyberkit an attacker can determine the live hosts in a network prior to launching an attack. This scanning is also possible from hosts infected with Welchi-Nachi worms [32].These worms use the ping utility to search for other hosts to attack. We got numerous "ICMP PING CyberKit 2.2 Windows"(false positive) Snort alerts and a few oversized ICMP ping packets which originated from different IP addresses.

SSH Brute Force attack

In our experiment approximately 235375 SSH login attempts have been observed against the honeypots. Only five of these attempts were successfully logged in the honeypot. The data of SSH connections were collected from both low-and high- interaction honeypots. Below is an example of Brute Force attack session that tries to gain access to the honeypot, but the attackers did not succeed downloading a remote file. At the end, he cleared all the command history from the bash history.

Brute Force Attack :
Honeypot received first SSH brute force attack from 95.16.192.243 on Jun 7 00:03:35.

Description
After several failed login attempts an attacker successfully gains user shell access on the system, checks the other existing users and their access rights on the system,cpuinfo such as the running processes and determines the users who have administration privileges, unsets the bash history configuration variables and removes the user .bash_history file.

Logging
When the attacker tries to brute-force the ssh service, he generates a lot of failed connections (failed username or password).The authentication log (/var/log/auth.log) on the honeypot creates the events for each failed and successful connections.

…
Jun 6 23:36:19 leeo1 sshd[834]: Failed password for invalid user anonimous from 95.16.192.243 port 44798 ssh2
Jun 6 23:36:21 leeo1 sshd[834]: Failed password for invalid user anonimous from 95.16.192.243 port 44798 ssh2
Jun 6 23:39:01 leeo1 CRON[837]: pam_unix(cron:session): session opened for user root by (uid=0)
Jun 6 23:39:01 leeo1 CRON[837]: pam_unix(cron:session): session closed for user root
Jun 7 00:03:35 leeo1 sshd[849]: Accepted password for test from 95.16.192.243 port 37082 ssh2

Below is the list of executed commands by attacker who successfully logged in to the system as user account "test "

```
linux@leeo1:/usr/src/linux-headers$ sudo lastcomm test
sshd                          Jun  7 00:03
uname-a                       Jun  7 00:03
id                            Jun  7 00:03
cat /proc/cpuinfo             Jun  7 00:03
ls –al                        Jun  7 00:03
ifconfig                      Jun  7 00:04
ps                            Jun  7 00:04
su                            Jun  7 00:05
ps -aux                       Jun  7 00:05
wget www.moused.as.ro/udp.pl  Jun  7 00:05
history -c                    Jun  7 00:06
```

Frontpage Extensions

The FrontPage Extensions are software technology deployed in the web server to provide Microsoft FrontPage clients to update and delete the content or upload files to a website. By using FrontPage extensions, attackers check the possibilities to access the web server, change permission and write new files to the directory of the web site. It relies on HTTP protocol and supports CGI/POST requests for server side processing. Although the FrontPage Extensions were not installed on our Windows honeypot, we received the following requests many times. The user agent show the signs of the bot called core-project. Core-project/1.0 crawler is used for finding the vulnerabilities in FrontPage powered websites. It sends a POST to /_vti_bin/_vti_aut/author.dll which is an authentication library. If this request is permitted then the server will respond with a bunch of configuration settings that is useful for the attacker to compromise the web server.

```
POST /_vti_bin/_vti_aut/author.dll HTTP/1.1
MIME-Version: 1.0
User-Agent: core-project/1.0
Host: 193.10.67.147
Accept:auth/sicily
Content-Length: 194
Content-Type: application/x-vermeer-urlencoded
X-Vermeer-Content-Type: application/x-vermeer-urlencoded
Connection: close
Cache-Control: no-cache
method=put+document%3a4%2e0%2e2%2e4715&service%5fname=&document=%5bdocument%5fname%3dc
ore%2ehtml%3bmeta%5finfo%3d%5b%5d%5d&put%5foption=overwrite&comment=&keep%5fchecked%5fou
t=false
```

Successful Administrator Privilege Gain

The attacker can get access to the FTP server on XAMPP using a default login and password weakness on the Windows honeypot. After only a few steps, he gained shell access to the honeypot and uploaded his own tools in order to attack other systems. But he could not be successful to compromise other machines, because his connections to the network were blocked by Honeywall. This attack was made from two different source addresses at the same time.

Attack steps

- The host with IP address 87.106.221.158 probes the open ftp port 21 on the honeypot, and successfully logins to the ftp server after several failed attempts.
- Enters into the web server directories in the FTP passive mode, searches for wget.exe file, but this file don't exist. He then uploads this file with the command "STOR wget.exe" and stores under the system folder.
- Similar to before, he first checks whether the file goonshell.php exists, and since it is not there, he uploads it with the STOR command as well.
- Another host with different IP address 213.112.109.122 (probably the same attacker uses another IP at the same time) uses goonshell.php backdoor to send HTTP POST requests in order to execute –cmd=<command> on the compromised machine.

- Then attacker uploads the nc.exe (Netcat for windows) via STOR command.
- The uploaded nc.exe runs by the PHP shell backdoor goonshell.php. The attacker tells netcat program to run the command: cmd=nc.exe+-l+-p+1450+-d+-e+cmd.exe via HTTP POST request. This utility runs with the arguments (parameter -L) on TCP port 1450 (-p 1450) and executes the command cmd.exe when a connection established on the network port.
- The attacker gains access to Windows command shell -cmd.exe and checks the system and web server directories.
- He downloads kill.exe, y.php, winhelp.exe files into the forum directory (C:\xampp\htdocs\forum) via ftp STOR. Then he moves the winhelp.exe to the system directory (C:\WINDOWS\system32). The attacker executes the winhelp.exe. It is self-extracted and unpacks itself to the system directory. Then he runs install.bat which copies necessary Windows binaries to different locations and overwrites the original binaries with the extracted files.
- Below you can see the list of extracted files into the C:\WINDOWS\system32 directory:
  Alc.exe, dipex.dll, install.bat, regg.exe, netshh.exe, libeay322.dll, ssleay322.dll, scc.exe, usb.sys, xml.mof, Xml.mfl, shellsd.exe2, PerfCounters.reg, pskill.exe, usb.sys
- Attacker uses the extracted files to run different system services on the honeypot.
- After finishing the monitoring process and examining the system and web forum directories, the attacker deletes all the files, clears the logs and history files in order not leaving traces, and also stops the services which he started. He only leaves dhcpcl, and shellsd.exe services running.

9th June, 2010
87.106.221.158 and 213.112.109.122

Description

An attacker using different IP addresses gains access to the system through FTP server installed on it. He uploads different rootkit tools and backdoors, and then gets shell access on the machine. He examines system directories, starts services, and tries to connect other machines in the same subnet. At the end he clears his tracks and removes the history, all uploaded files, stops services except shellsd.exe, dhcpcl).

Sebek logs:

USER newuser
PASS wampp
PWD
TYPE A
PASV
MLSD
CWD dump
PWD
PASV
MLSD
CWD bilder
PWD
CWD /forum
SIZE wget.exe
PASV
STOR wget.exe
TYPE I
SIZE goonshell.php
PASV
STOR goonshell.php
STOR nc.exe
SIZE kill.exe
PASV
STOR kill.exe
SIZE y.php
PASV
STOR y.php
PASV
MLSD
TYPE I
SIZE winhelp.exe
PASV
STOR winhelp.exe

Summary of involved tools

- *wget.exe* - first uploaded file by intruder. It is a utility used to download files from the remote host via command prompt.
- *goonshell.php* - is a backdoor uploaded into the web server to get remote control of the honeypot via web. It is more common used method by experienced attackers. In our case, he first uploads different executable files, but he can't succeed to run any commands. Therefore he uploads the file goonshell.php in order to get shell access.
- *winhelp.exe* - a self-extracted achieve contains files which can be used by attackers. It extracts all the included files into the system directory. Some of these files might be Troyan Horses, rootkits and so on.
- *install.bat* - batch file which is extracted from the winhelp.exe archive; controls the installation and distributing the extracted files to the different system directories.
- *shellsd.exe* - is able to use ports to connect LAN, hide itself and secretly monitor applications.
- *Scc.exe* - also acts as a backdoor, and is used to start dhcpcl service.Dhcpcl is not a core Windows file, it has the same functionally as shellsd.
- *nc.exe* – It is a widely used netcat utility that reads and writes data across network connections, and it is easy for remote attackers to control the rootkits and other tools on the compromised machine by using this utility.
- *y.php* – allows attacker to control processes via web and execute different tools in order to clear all his trace and logs on the honeypot.
- *Kill.exe* – is used to terminate the processes on the honeypot.

This attacker may be a little experienced. Because it seems like he has prepared his tools and tactics in advance in order to quickly compromise the target machine.

Bots

Basically the honeypots can be found via specific searches from search engines such as Google, Yahoo or BING and this leads to numerous connection attempts between blackhats and the honeypots. The statistics and study within the Honeynet Project [7] shows that majority of the attacks occurs on the web applications. Because, the web applications have become the easier attack target and most of new malware focus on the application layer of OSI model. In our experiment, we have observed lots of bots in the form of web spiders or other parser tools (a spider is a program which fetches the information from web servers, for example Google and Yahoo web crawler). Typically spiders can be identified by the 'user-agent' field of an HTTP request such as 'msnbot/2.0b'. Thus we have observed many bots which try to fetch information from the installed web servers on the honeypots. Googlebot, MSNbot and other similar spider software are nothing more than a computer program which collects information about the web sites in order to build a searchable index for the search engines.
Each computer is infected with a malicious program called a "bot", which can communicate with other bots in the botnet or with common Command and Control (C&C) to receive commands from the botnet owner. Botnet is a network of computers which is compromised and remotely controlled by an attacker [11]. By using a botnet, attackers can conduct Distributed Denial-of-Service (DDoS) attacks, spamming, malware spreading and etc. over this remote control network. Many bots also use search engines to spread in the internet. Although bots were firstly used in Internet Relay Chat (IRC) networks, they can also use covert communication channels via a HTTP tunnel instead of the IRC protocol. Thus they can encode the commands to the bots inside HTTP requests.

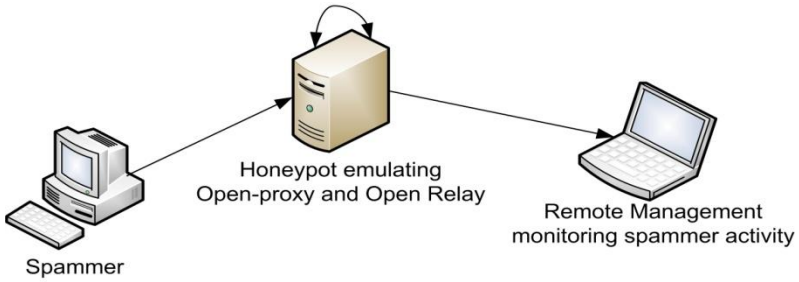
PhpMyamin

Web applications offer services such as mail, web shops, web forums or database administration tools like phpmyadmin which can attract large number of potential attackers.
The PhpMyadmin version installed in one of the honeypots allows code injection. Code injection is an attack type which can result in the execution of arbitrary code. The vulnerability in setup.php in PhpMyAdmin(CVE -2009-1151) [37] allows remote attacker to inject the arbitrary code by using configuration file to *config.inc.php*.

```
POST //phpmyadmin//scripts/setup.php HTTP/1.1
User-Agent: curl/7.15.5 (i686-redhat-linux-gnu) libcurl/7.15.5 OpenSSL/0.9.8b zlib/1.2.3 libidn/0.6.5
Host: 193.10.67.150
Accept: */*
```

Spam

As mentioned in the previous chapter, we emulated SMTP mail server on our low-interaction honeypot for understanding how spammers operate. We configured it in such a way that attacker sends spam emails via open proxy server or open mail relays. These spam emails have been collected and filtered with a spam trap-honeypot.



An open mail relay accepts email from the any sender address to any recipient address [1]. Spammers actually use open proxies to anonymize their identity from the spam recipients. Our honeypot has received hundreds of emails per month which most of them coming always from open relay scanners in a limited set of countries. The majority of open relay attacks were received from the cities in Taiwan which is one of the countries in the world that hosts the largest number of spam servers according to the statistics of ProjectHoneypot (a web based honeypot project used to collect the statistics about spam bots on the internet) [32]. The log below was generated when clients sent a message to the server via an open mail relay:

```
HELO 193.10.66.128
MAIL FROM: <hi7176s.pp4813@msa.hinet.net>
Sat Apr 17 22:47:01 CEST 2010: SMTP started from  Port
HELO 193.10.66.128
MAIL FROM: <z2007tw@yaoo.com.tw>
Thu Jun  3 01:20:58 CEST 2010: SMTP started from  Port
```

Directory (or path) traversal

Many attackers use directory traversal attacks to get the URL tree of a website before starting an attack. The main goal of this kind of attacks is to search for hidden configuration and setup files, also tools and software installed in the web server which are not well protected or have no authentication. This sort of search is more common for the old version of phymyadmin, webmail and web forums which run with default configuration. The below example from the honeypot log shows that an attacker tries to find the main.php file by requesting a list of all possible database administration tools and possible directories and phpmyadmin version releases by using multiple HTTP GET requests:

```
GET /db/main.php HTTP/1.0
GET /web/main.php HTTP/1.0
GET /PMA/main.php HTTP/1.0
GET /admin/main.php HTTP/1.0
GET /dbadmin/main.php HTTP/1.0
GET /PMA2006/main.php HTTP/1.0
GET /sqlmanager/main.php HTTP/1.0
GET /p/m/a/main.php HTTP/1.0
GET /phpmanager/main.php HTTP/1.0
GET /php-myadmin/main.php HTTP/1.0
GET /webadmin/main.php HTTP/1.0
GET /sqlweb/main.php HTTP/1.0
GET /websql/main.php HTTP/1.0
GET /mysqladmin/main.php HTTP/1.0
GET /phpmyadmin2/main.php HTTP/1.0
GET /phpMyAdmin2/main.php HTTP/1.0
GET /php-my-admin/main.php HTTP/1.0
GET /phpMyAdmin-2.6.1-pl1/main.php HTTP/1.0
GET /phpMyAdmin-2.6.2-rc1/main.php HTTP/1.0
GET /phpMyAdmin-2.6.2/main.php HTTP/1.0
GET /phpMyAdmin-2.6.2-pl1/main.php HTTP/1.0
GET /phpMyAdmin-2.6.3/main.php HTTP/1.0
GET /phpMyAdmin-2.6.3/main.php HTTP/1.0
GET /phpMyAdmin-2.6.3-pl1/main.php HTTP/1.0
GET /phpMyAdmin-2.7.0-rc1/main.php HTTP/1.0
GET /phpMyAdmin-2.7.0-pl1/main.php HTTP/1.0
GET /phpMyAdmin-2.8.0.2/main.php HTTP/1.0
GET /phpMyAdmin-2.8.0.3/main.php HTTP/1.0
GET /phpMyAdmin-2.8.1-rc1/main.php HTTP/1.0
GET /phpMyAdmin-2.8.1/main.php HTTP/1.0
GET /phpMyAdmin-2.8.2/main.php HTTP/1.0
                    .....
GET /admin/phpmyadmin/main.php HTTP/1.0
GET /admin/sqlmanager/main.php HTTP/1.0
GET /admin/phpMyAdmin/main.php HTTP/1.0
GET /admin/sysadmin/main.php HTTP/1.0
GET /admin/sqladmin/main.php HTTP/1.0
GET /admin/db/main.php HTTP/1.0
GET /admin/web/main.php HTTP/1.0
```

Authentication BruteForce

Most attackers use their dictionary in order to brute force log-in credentials. We observed multiple brute forcing attempts trying to get administrator access to the web forum on our honeypot. Brute force attacks against web applications are called attacks against log-in credentials [40]. Thus the attacker attempts to log-in to a system by guessing username and password. The logs below show the requests of the attacker which is trying to bruteforce with the log-in credentials "Administrator/admin"and "admin/jamboree". First his attempts fail because the credentials didn't match the forum administrator's username and password. After several attempts he could find the admin's password.

Mon, 07 Jun 2010 17:02:14 GMT
130.206.165.116 →193.10.67.147 POST /forum/index.php?action=login2 HTTP/1.1
Referer: http://leeo2.sics.se/forum/index.php?action=login2
Cookie: PHPSESSID=916e669e2509a9fbe979bc555fe8fcd7
user=*administrator*&*passwrd*=admin&cookielength=60
Mon, 07 Jun 2010 17:04:46 GMT
130.206.165.116 →193.10.67.147 POST /forum/index.php?action=login2 HTTP/1.1
Referer: http://leeo2.sics.se/forum/index.php?action=login2
Cookie: PHPSESSID=916e669e2509a9fbe979bc555fe8fcd7
user=*admin*&passwrd=*jamboree*&cookielength=60

6.3 Statistical Analysis

In this section we show an overall statistical analysis of the results collected from our honeypot experiment from March to July of 2010. During this period our honeypot environment suffered different kind of attacks. Table 3 shows the number of attack connections during the observation period:

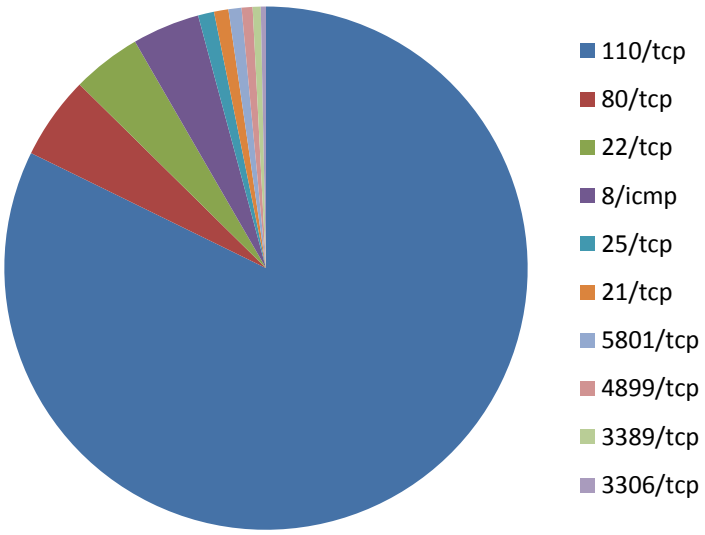
Protocol	Connections	Percentage
TCP	88365	95.42%
UDP	482	0.52%
ICMP	3759	4.03%
Total	92606	100%

Table 3 Total connections per protocol

By looking at Table 3, we can see that TCP is the most used protocol by attackers. This can be explained by the fact that multiple service and applications use TCP compare to other protocols. Therefore most of the probed ports were TCP ports [Fig.6.1]. Figure 6.2 shows the number of connection attempts on each honeypot during a month period. The number of established connections on two honeypots was more than on the other honeypots. It happened because of many open ports and real and emulated services run on those two honeypots. This indicates that those two machines have been continuously scanned by vulnerability scanners and the rate of the connection attempts on the network services HTTP (80), SMTP (25), POP (110), SSH (22) was higher.

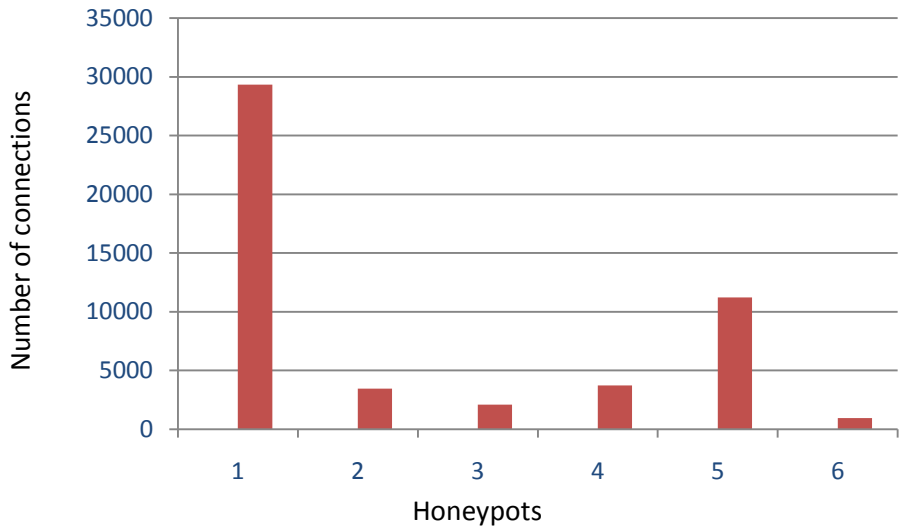
-Ports 100 and 25: POP and SMTP as emulated services was main target for spammers, and automatic programs in internet.

-Ports 80 and 22: HTTP and SSH services with vulnerabilities several times were successfully exploited by attackers.



Top 10 Probed Ports		
Rank	**Port**	**Connections**
1	110/tcp	73510
2	80/tcp	4586
3	22/tcp	3822
4	8/icmp	3723
5	25/tcp	885
6	21/tcp	796
7	5801/tcp	721
8	4899/tcp	605
9	3389/tcp	444
10	3306/tcp	285

Figure 6.1 Top attacked ports



Honeypots	
1.	**193.10.67.150**
2.	**193.10.66.128**
3.	**193.10.66.96**
4.	**193.10.66.242**
5.	**193.10.66.52**
6.	**193.10.67.147**

Figure 6.2 Total numbers of connections per honeypot

The next graphic illustrates the main sources of the attackers. We observed 2343 unique IPs originating from 75 countries across the globe. Among these 75 countries the highest numbers of attacks were received from Europe, Mexico, US and Australia. Most attacked IPs and the countries which they belong were shown in Figure 6.3. Among these IP addresses there were some sources that scanned all of the honeypot machines. More probably these scans were made by automatic scanners.

Rank	Source IP	Connection	Country
1	201.144.203.78	28428	MX
2	95.51.84.35	27894	PL
3	96.254.8.11	15521	US
4	203.45.26.143	677	AU
5	91.122.68.138	592	RU
6	82.128.84.197	460	NG
7	207.114.153.18	442	US
8	85.214.45.60	369	DE
9	128.168.253.4	308	US
10	66.71.246.164	274	US

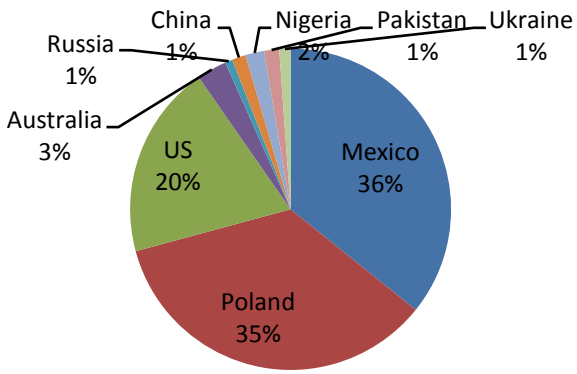


Figure 6.3 Top 10 Attack IPs and countries

According to the above-mentioned graph (Fig.6.2) the honeypots with web servers and other vulnerable web services were attacked more compared to other honeypots. The majority of these attacks were blind, thus they randomly scanned honeypot IPs and tried to compromise the web servers without checking the installed web applications, and used their web exploit methods to search for the vulnerabilities on the systems. We observed that the targeted attacks represent only 4% of total attacks. Statistics of the targeted and blind attacks to web server and applications (Fig.6.4) shows that PHP was the most attacked web language and PhpMyAdmin as the most attacked application on our honeypots, while other installed applications on the web server except web forum didn't attract many attacks. Basically we observed the scanning/exploitation attempts by blind attacks against the well-known web applications which do not exist on the honeypots. These web applications are commonly used over the Internet and have become main targets for the attackers who use random exploitations in order to attack multiple hosts.

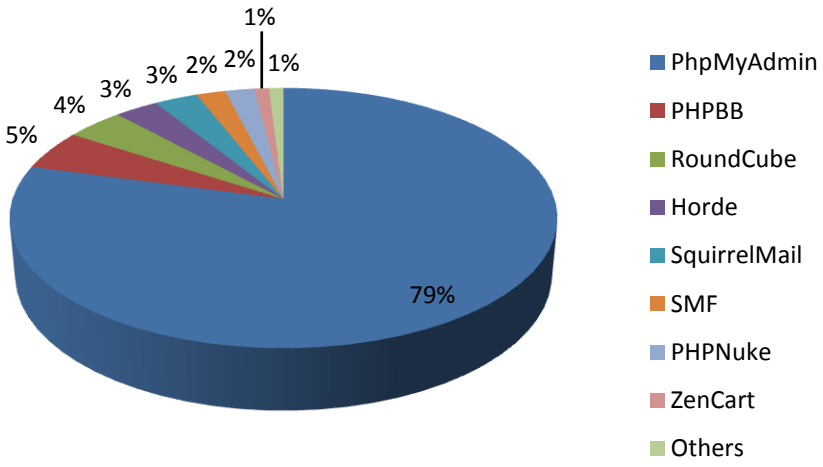


Figure 6.4 Attacks per web-applications

The frequency of the most observed attacks has been shown in Figure 6.5. As you can see the largest number of attacks was Brute Force, Directory (or Path) traversal, Remote File Inclusion, Command Injection and etc. Authentication bruteforce attacks were performed against the SSH server and web applications in order to obtain user's authentication credentials by guessing usernames and passwords.

Directory (or Path) traversal is used by attackers to find hidden applications and configuration directories residing on the web server. They basically search for default locations and version numbers of the applications by using known vulnerabilities. Some attacks that had occurred frequently were command injection attacks, thus they tried to inject commands specified by the attacker in the vulnerable web applications. Remote file inclusion was also a common exploitation technique which allowed attackers to execute remotely hosted malicious files usually through the existing scripts on the web server. Only 2% of the total observed attack attempts were SQL Injunction, Cross-Site Scripting (XSS), DDOS and etc.

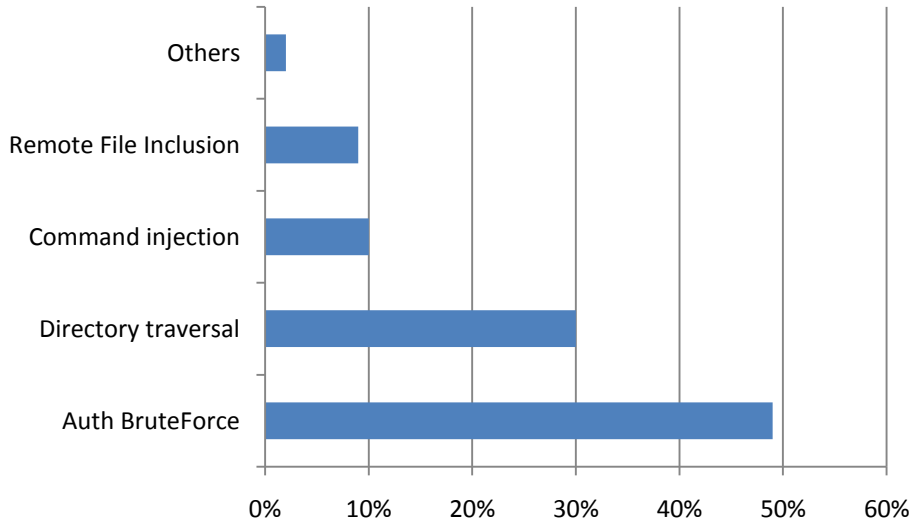


Figure 6.5 Percentage of attacks by type

As mentioned in the previous chapter, we have configured and implemented SSH Honeypot in order to study more about Brute Force attacks on the SSH Servers. After analysis of the login attempts we divided SSH bruteforce attacks into two types, one which target were root passwords, and another that used a dictionary of well-known username/password combinations which correspond to the brute-force dictionary attacks. During four months period from March till July of 2010 we observed 235375 SSH login attempts from 329 unique IP addresses. Out of total login attempts, only five of them logged in successfully where two of them changed the account passwords. Although it was impossible to get the root password, but some of the privileged SSH user accounts have been authenticated successfully. Table 4 shows the list of the usernames that are used more often by attackers. Next, we looked at the passwords used in the login attempts. Figure 6.6 shows the top 15 passwords which some of them have been tried with most of the usernames. The attackers tried 60553 different passwords in order to login to the SSH server. In many cases, the passwords were the same as usernames. There were also some complex passwords used with random letters and number sequences. Table 5 shows the list and frequency of most tested pairs.

Login Attempts	Username
72194	root
2419	test
1621	admin
1391	oracle
1281	nagios
1015	guest
899	user
889	123456
765	mysql
755	postgres
741	password
731	qwerty
726	a
713	1q2w3e
691	backup

Table 4 Top attempted usernames

Login attempts	Username	Password
331	root	123456
321	root	root
263	test	test
259	root	password
252	oracle	oracle
140	mysql	mysql
138	test	test123
137	postgres	postgres
118	admin	admin
111	backup	backup
110	root	1qaz2wsx
117	test	123456
103	michael	michael
94	alex	alex
90	apache	apache

Table 5 Frequency of most tested pairs

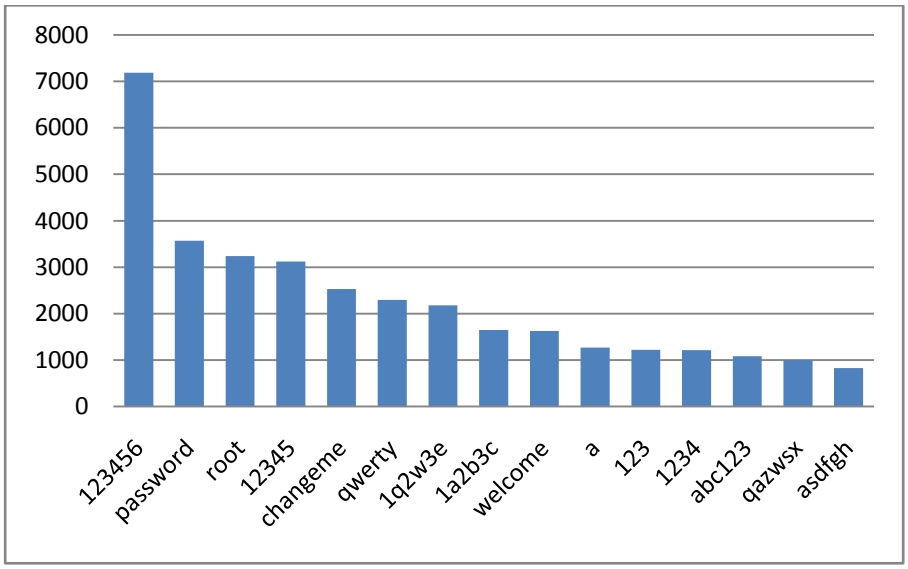


Figure 6.6 Top 15 attempted passwords

Chapter 7

Classification and skill level of attackers

In this chapter we describe the classification of attacks and exploited vulnerabilities by attackers on our honeypot environment. Then we evaluate the attacker's skill and knowledge level based on the lessons learned from the analysis and statistics of the collected data. By studying the characteristics (methodology, skills, and tactics) of the attackers organizations can anticipate their further malicious activities in the networks and create adequate security safeguards for their networks in the future.

7.1 Classification of attacks and exploited vulnerabilities

After data collection and analysis step, we focused on the classification of the attacks and exploited vulnerabilities based on the gathered data from the attackers. To achieve this, we first studied the existing classification and taxonomies to determine how they are applicable to our purpose.
Attack classification has always been an interesting area for security researchers. It gives an in-depth view of the attack, the target, attackers and exploited vulnerabilities. Many organizations and researchers have proposed different taxonomies and techniques for classifying attacks according to their characteristics and improved them with better techniques over the years. Each classification was developed for a certain goal. For example, understanding vulnerabilities and attack behavior, capturing the attack process in order to strengthen the defensive measures.
Howard [33] defined process-based taxonomy of the computer and network attacks. This taxonomy describes the process of the attacks, rather than an attack classification. His approach was to create a link between attackers and objectives in the attack process. Although we didn't use this taxonomy in the classification of the attacks, but it was useful for understanding the process of the attacks. The taxonomy consists of five stages: attackers, tools, access, results and objectives as it can be seen by the computer and network incident taxonomy in Fig7.1.

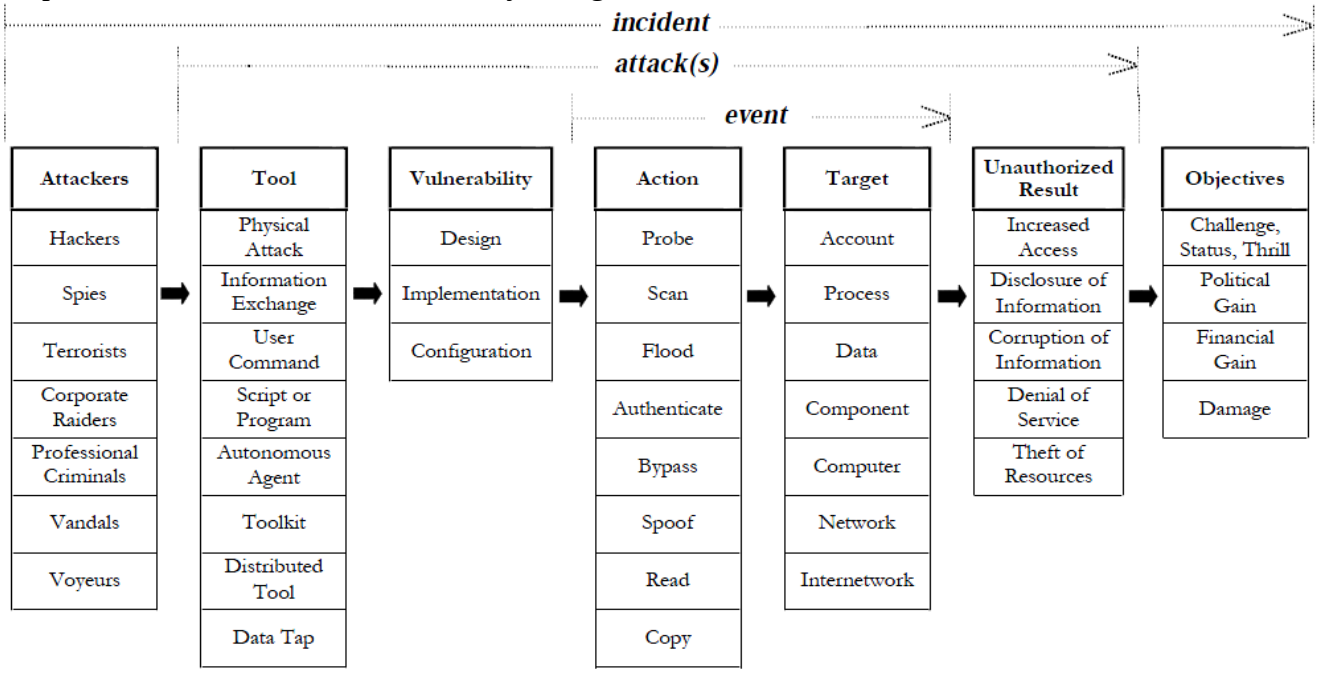


Figure 7.1 Computer and Network Incident Taxonomy [33]

Another taxonomy is proposed by Lough [34] which is based on the four characteristics of the attacks. This taxonomy was too general to gain detailed information about the attacks and according to the CERT (Computer Emergency Response Team) it may not be useful for identifying and classifying new attacks. Thus, it lacks the classification to the type of attack, such as worms, viruses, trojans and so on. The weakness of Howard and Lough taxonomies is that vulnerability classification is not clearly described in these taxonomies.

The above-mentioned taxonomies tend to be general in their approach to classify attacks. There are other proposed taxonomies that classify attacks based on a set of dimensions. Thus, the first of them was one dimensional that classified attacks based on their effects. Later, Lindqvist and Jonsson [35] presented such a taxonomy by using the two dimensions of an attack. Hansman and Hunt's taxonomy [36] which aim at classifying and categorizing the computer and network attackers based on their similarities rather than attack processes. This taxonomy proposes four unique dimensions for attack classification. The first dimension being attack vector is used to categorize the attack. The second dimension covers the classification of attack targets (e.g., OS, application, network protocol) which they also classified down to more specific targets. Attackers may target the memory, operating system, network, file system and application. The exploited vulnerabilities are covered in the third dimension. But sometimes an attack exploits multiple vulnerabilities which is not fully modeled in this taxonomy and provides the less information about vulnerabilities. In this taxonomy, the list of vulnerabilities is usually defined by the Common Vulnerability Exposures (CVE) [37]. In the case that a CVE entry does not exist for a certain exploited vulnerability, then Howard's vulnerability types (implementation, configuration and design) should be selected and described under the third dimension [36]. The fourth dimension shows the effects or payloads of attacks. The payload may also be an attack itself and damage the system files. For example, a worm may carry a Trojan in its payload. This dimension consists of the following categories:

- Corruption of information – it happens when attack effect destroys some information;
- Disclosure of information - enables an attacker to gain valuable information about a target;
- Theft of service –uses the system services without authorization;
- Subversion –gains access and controls the target;

Based on the taxonomical work conducted by Hansman and Hunt [36] we show the classification results of the observed attacks as [Table 6]:

Attack type	Target	Exploited vulnerabilities	Effect
Brute Force, Dictionary attack	OpenSSH,Web Forum (SMF 1.0.5)	Implementation	Disclosure of information, subversion
Brute Force	XAMPP 1.6.6, phpMyAdmin 2.11.4 and FileZilla FTP Server 0.9.25	CVE-2009-0919	Corruption and disclosure of information, to obtain authority, subversion
Cross-site scripting (XSS)	Microsoft Frontpage extensions,PHPBB, Apache	CVE-2000-0746 CVE-2000-0114 CVE-2007-5000 CVE-2000-0413	Information Disclosure CVE-2002-0902
Predictable Resource Location	Apache 2.2.6,IIS	WASC-34	Information Disclosure
DDOS attack: Trinoo,Shaft		CVE-2000-0138	UDP flooding

Directory Traversal	Apache 2.2.6,IIS, PhpMyadmin,SMF 1.0.5	Inappropriate Configuration, CVE-2008-6659	Disclosure of information
Remote File Include	Joomla 1.0.8	CVE-2008-5671	Remote access
Remote Code Execution	Horde Application Framework	CVE-2006-1491	Disclosure of information
DOS attack: Ping of Death	OS: Ubuntu 7.10,Windows 2003 Server	CVE-1999-0128	Loss of availability

Table 6 Classification results

In general this taxonomy works well and attacks can be easily classified. It is also improvable, and additional dimensions could be added to enhance the taxonomy. The only limitations of this taxonomy are difficulty of categorizing blended attacks and absence of the variations of the vulnerability exploited by various attacks. The blended attacks can contain many sub-attacks which make classification difficult. The results show that all of the observed attacks and exploited vulnerabilities were known. That is why our attack classification was based on the known attack and exploited vulnerabilities. All successful attacks were based on specific vulnerabilities that service or inappropriate operation discovered by attackers. After analysis of the observed attacks on our honeypot environment and being familiar with the attack attributes from the existing taxonomies [33][34][36], we categorized the attacks according to their attributes (Fig 7.2).

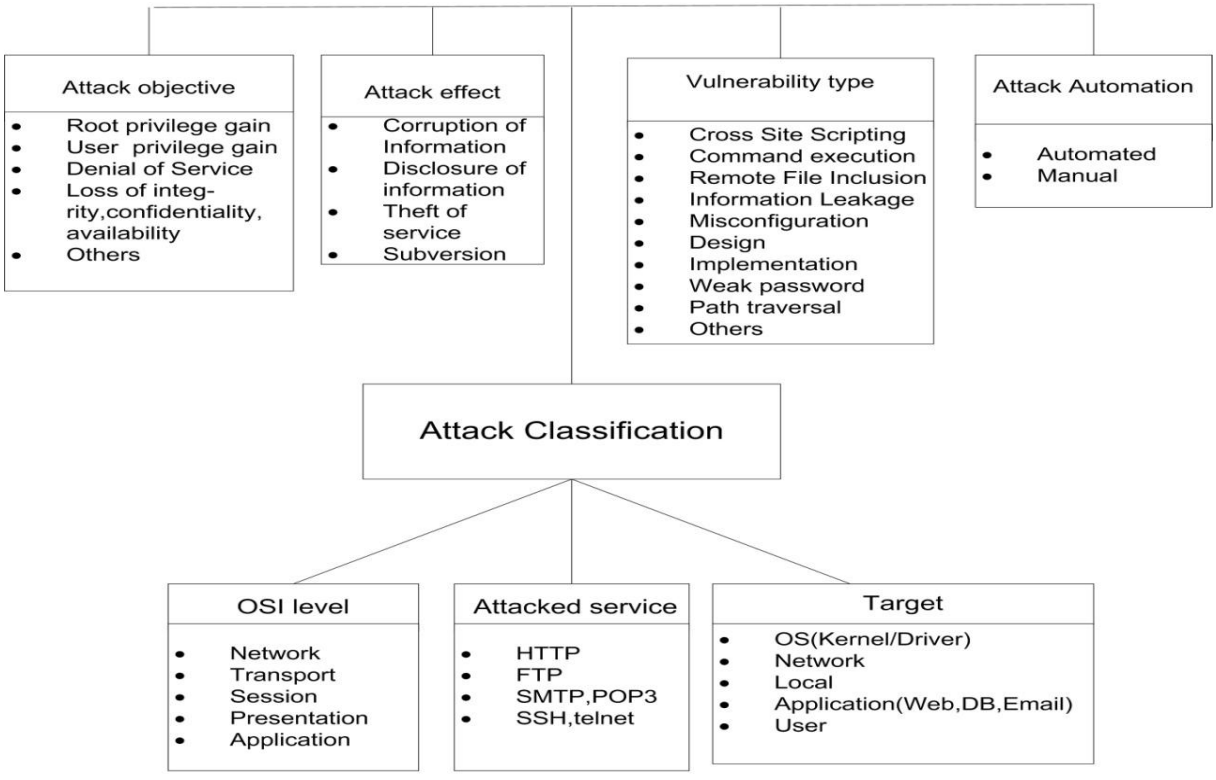


Figure 7.2 Attack classification

The effect type of the attacks [36] depends on their objectives. These objectives are classified down according to their impact level on the targets. After gaining the administration privileges an attacker can have the largest impact on the system. Having some specified user rights and privileges, attackers can cause the loss of the system confidentiality, integrity and denial of service. During our observation several brute force attacks aimed to acquire administrator rights after successfully gaining the user privileges, then some of these successful attacks attempted to compromise other machines in the same subnet.

We observed attacks almost in all layers of the OSI model. The application layer was more attacked because of its potentiality to perform attacks. Based on the degree of automation of the attack, we categorized attacks as manual and automatic. Thus the observed activities corresponding to automatic attacks were more than manual attacks'. The analysis in the previous chapter shows that about 80% of the targets on our honeypots are compromised completely automatically. Some targets are compromised both automatically and manually. For example, both automatic and manual attacks occurred during SSH brute force attack, thus two main steps of the attack process has been analyzed by means of automation level:

- Automatic: Brute-force dictionary attacks gained access to the system, but didn't carry out any activities after they succeeded. More probably it has been performed by automatic tools.
- Manual: Attackers carried out activities after they succeeded in breaking into the system. More probably this step has been performed by human beings.

Depending on the attack objectives and compromised targets, the exploited vulnerabilities may differ. The vulnerability types that categorized in Fig 7.2 are used by a majority of the observed attacks on OSI Application layer. The statistics shows that the web-applications on our honeypots were main targets by attackers and they exploited vulnerabilities of different risk level. Web applications with Brute Force Attack, Directory Traversal, Remote File Inclusion, Improper authentication vulnerabilities detected by automatic scanning. In general, the most exploited vulnerability categories were Cross-site Scripting, different types of Information Leakage, Directory Traversal, Remote File Inclusion, Command Execution, Insufficient Authentication and other vulnerabilities caused by improper configuration, design and implementation. All of the exploited vulnerabilities in our experiment were publicly known and have been evaluated in National Vulnerability Database (NVD) [38]. The NVD database provides CVSS score [39] for almost all known vulnerabilities. CVSS defines the severity level of the vulnerabilities and scores them based on several independent metrics, such as Base, Temporal and Environmental metrics. The Base metric is more commonly used in the different vulnerability databases. The overall severity score combines impact (the consequence level of exploitation) and exploitability (difficulty degree of exploitation) components. In the Table 7 we classified the exploited vulnerabilities according to the NVD severity rankings of "Low", "Medium", "High" and categorized them into vulnerability types which are identified in CWE (Common Weaknesses Enumeration). CWE is a community-developed formal list of software weaknesses types [40]. NVD assigns severity level of each vulnerability according to its CVSS base score [38]: Low: 0-3.9; Medium: 4-6.9; High: 7-10. Vulnerabilities with most severity have high risk to the organization network. The base metrics group measures the attributes of the vulnerability using two sub-scores: (i) exploitability sub-score and (ii) impact sub-score. These sub-scores are calculated based on the following metrics [39]:

- Exploitability: Access Vector, Authentication, Access Complexity
- Impact: Confidentiality, Integrity, Availability

The Exploitability of vulnerability can be captured using Access Vector, Access Complexity, and Authentication. Access Vector indicates the access required to exploit vulnerability, Authentication measures the number of times that an attacker should authenticate to a target in order to perform

exploitation. The complexity of an attack which is required to exploit vulnerability is quantified by the Access Complexity once an attacker accesses to the target system. It is measured as High, Medium, and Low complexity levels. This metric could also be useful in evaluating the attacker's skill required to exploit the vulnerability, thus low complexity shows that system are always exploitable.
The impact of vulnerability can be captured using the confidentiality Impact, Integrity Impact, and Availability Impact metrics. These metrics show the level of loss of confidentiality, integrity and availability. Every CVSS score accompanied by the corresponding vector. In the below table we showed the Base vector of each vulnerability, thus it includes the abbreviations of CVSS metrics and the values.

#	CVE -ID	Title	Type	Severity	Base Vector
1	CVE-2008-5671	PHP remote file inclusion vulnerability in index.php in Joomla	Remote File Inclusion	High	AV:N/AC:L/Au:N/C:P/I:P/A:P
2	CVE-2008-6659	Directory traversal vulnerability in index.php in Simple Machines Forum (SMF)	Path traversal	Medium	AV:N/AC:L/Au:S/C:P/I:P/A:N
3	CVE-2009-0919	XAMPP installs multiple packages with insecure default passwords	Credentials Management	High	AV:N/AC:L/Au:N/C:P/I:P/A:P
4	CVE-2006-1491	Eval injection vulnerability in Horde Application Framework	Remote code execution	High	AV:N/AC:L/Au:N/C:P/I:P/A:P
5	CVE-2000-0138	A system has a distributed denial of service (DDOS) attack master, agent, or zombie installed, such as (1) Trinoo (2)Shaft	Denial Of Service	Medium	AV:N/AC:L/Au:N/C:N/I:N/A:P
6	CVE-2007-2243	OpenSSH 4.6 and earlier, when ChallengeResponse-Authentication is enabled, allows remote attackers to determine the existence of user accounts	Improper authentication	Medium	AV:N/AC:L/Au:N/C:P/I:N/A:N
7	CVE-2007-5000	Cross-site scripting (XSS) vulnerability in the (1) mod_imap module in the Apache HTTP Server	Cross Site Scripting	Medium	AV:N/AC:M/Au:N/C:N/I:P/A:N
8	CVE-1999-0128	Oversized ICMP ping packets can result in a denial of service, aka Ping o' Death.	Denial Of Service	Medium	AV:N/AC:L/Au:N/C:N/I:N/A:P

Table 7 Classification of exploited vulnerabilities based on the severity level

7.2 Skill level of the attackers

Once attacks, vulnerabilities have been identified, analyzed and classified, we also need to study the exploitation skill of the attackers. Notice that each attacker is a part of the attacker community, and thus, we do not study them individually in the terms of skill level, but as a group. Every attacker has a certain amount of skills and knowledge according to difficulty degree of the exploitation of the vulnerabilities which he has gained access to. The complexity score is based on the difficulty of the vulnerability exploitation, and thus, it also allows us to learn how the attackers are skilled when they successfully exploit the vulnerabilities of our honeypots [39]. National Vulnerability Database (NVD) uses the scale of low, medium, or high complexity, as well as defines and scores the complexity of the exploitation of the known vulnerabilities. The vulnerabilities classified as medium and high are harder to exploit than low complexity vulnerabilities, so less skill is required for attackers in order to exploit vulnerabilities of low complexity. But high and medium complexities require high levels of skill, specialized access conditions and social engineering methods for performing the exploitation [39]. Basically most of the potential attackers search for the vulnerabilities of complexity low enough in order to achieve his objective easily in a short time period. Such attackers are not so experienced or use automatic scripts once they have access to the vulnerabilities. Thus, attackers and their skills vary depending upon the difficulty of the exploitation or complexity of the vulnerabilities.

Learning how to perform an attack can take a huge amount of time. That is why, many attackers because of the time are trying to use already developed scripts by others or automated tools to gain access to their targets in a short time. The attacker's skill level is even an important factor in order to estimate the time required to exploit certain vulnerabilities and gain access to the target systems. According to the proposed model -"The Time to Compromise" by M.A. McQueen and others [41], the time required to compromise the systems depends on known vulnerabilities and the skill level of the attacker. That is why studying and evaluation of attacker skill level becomes so important and attracts many researchers in the IT security.

In order to gain more insight into the attackers' skills, we first need to know exactly who is attacking our honeypots. To begin with, we classify the attackers into the following skill-levels based on the vulnerabilities they exploit:

- **Script kiddies:** Members of this group have no advanced computer or network skills. However, they use scripts developed by others without necessarily understanding them. Some of the script kiddies usually follow simple instructions and have no idea what they are doing, and they usually try to exploit a small number of vulnerabilities. But some of them might be advanced users, thus can develop their own tools and use behind sophisticated backdoors. All of the script kiddies share a common methodology, and randomly search for the specific vulnerabilities on the wide networks. Their goal is to get root access the easiest way possible on their targets. Referring to the statistics, the honeynets which simulate several networks are attacked more often by script kiddies [7].

- **Hackers:** They are the group that usually act alone and skilled programmers. They have deep knowledge of the vulnerabilities of the applications, operating systems and also ability to study unknown exploits using their experience. This kind of attackers uses special precautions against the detection and usually clears their tracks and logs when they leave the systems. That is why sometimes it is difficult to get information about hacker attacks.

- **Botnet Owners:** Members of this group aim to install bots on their targets and control them remotely. Botnets are exploited for different purposes, such as DDOS attacks, creation of the SMTP mail relays for spam and etc. They have also similar strategies similar to script kiddies, but they

differ in terms of skill level, because botnet owners are more skilled than script kiddies. They can succeed with hiding their bots inside the system. Phishing attacks can also be distributed via botnet mechanisms.

The majority of the attackers we faced in our experiment were script kiddies. Most of them were not familiar with the access rights and privileges of the operating systems on honeypots (e.g., they tried to kill the processes for which they don't have the right privileges). And they were also searching for the vulnerable applications on the web server even without fingerprinting. This kind of attacks was mainly driven by script kiddies who searching the latest disclosed exploits or vulnerabilities throughout the Internet. For example, we observed that several script kiddies were searching for the same exploits on all the honeypots. These script kiddies more likely were looking for certain vulnerabilities along a predefined range of IP addresses in the company's network. Our honeypots were installed in a company IP address range and have no real data value. The installed applications on the honeypots tried to simulate confidential data value such as web forum, web servers and different database and website administration panels with known vulnerabilities. Any skilled attacker first collects information about the target and finds out that it is located in a research institute, thus it seems like he has special reason to attack this target. Although most of the script kiddies found the known vulnerabilities of the honeypot applications, we also observed a few hackers who were more skilled compared to script kiddies. They also successfully exploited the known vulnerabilities in order to access unauthorized information. There was a hacker which even could identify that he logged into the honeypot, i.e., he got information about the hardware by taking a look at the "cpuinfo" and other files under the /proc/ tree on the Linux honeypot. This file contains information about the CPUs and system architecture dependent items. That is why the hacker could easily figure out that Virtualbox was installed on the system (which allowed the hacker to detect the honeypot).
Another interesting fact is that almost all of the observed hackers cleaned their tracks, and deleted the history files (such as .bash_history) when they left the honeypots.

Chapter 8

Conclusion

In this thesis, we have implemented a honeypot architecture and used it for gathering attack data and tracking the activities carried out by the attackers. Then we have analyzed and classified the observed attacks and vulnerabilities. The aim was to study the attackers' skill and knowledge based on this analysis. It appears that most of the observed attacks were automated and carried out by script kiddies.
We hope that this work will help organizations to select proper protection mechanism for their networks by evaluating the impact of detected attacks, and taking into consideration the attacker's skill and knowledge level.
As a future work, we have proposed an improved hybrid honeypot architecture with a different approach to collecting attack data and learning about the attackers' skills.

References:

[1] M. Jakobsson and Z. Ramzan. Crimeware: Understanding New Attacks and Defenses. Addison-Wesley Professional, 2008.

[2] Honeyd, http://www.honeyd.org/

[3] Virtual Honeypots: From Botnet Tracking to Intrusion Detection 2007
by Niels Provos; Thorsten Holz

[4] Conceptual framework for a Honeypot solution
Christian Döring, M.Sc.University of Applied Sciences Darmstadt, Department of Informatics (FHD)

[5] A Guide to Different Kinds of Honeypots
http://www.securityfocus.com/infocus/1897

[6] Lance Spitzner.Honeypots, Definitions and Value of Honeypots .
http://www.spitzner.net May, 2002

[7] The Honeynet Project.”Know your enemy” (http://project.honeynet.org).

[8] Clifford Stoll.The Cuckoo’s egg. ISBN: 0743411463

[9] http://en.wikipedia.org/wiki/Honeypot_(computing)

[10] Niels Provos. A virtual honeypot framework. In Proceedings of 13th USENIX Security Symposium, pp. 1–14. USENIX, 2004.

[11] Lance Spitzner . Honeypots: Tracking hackers Addison Wesley Professional, September 2002

[12] Nepenthes. http: //nepenthes.mwcollect.org

[13] SUN Microsystems. VirtualBox. http://www.virtualbox.org/.

[14] “Know Your Enemy: Honeywall CDROM Roo”,
http://old.honeynet.org/papers/cdrom/roo/index.html

[15] Honeypotting with VMware - basics
http://seifried.org/security/ids/20020107-honeypot-vmware-basics.html

[16] The Value of Honeypots, Part One: Definitions and Values of Honeypots
by Lance Spitzner with extensive help from Marty Roesch last updated October 10, 2001
http://www.securityfocus.com/infocus/1492

[17] F.Pouget,M.Dacier,LEURRE'COM:The Eurocom Honeypot Project

[18] [Kaâniche et al. 2006] M.Kaâniche, E.Alata, V.Nicomette, Y.Deswarte, M.Dacier, Empirical Analysis and Statistical Modeling of Attack Processes based on Honeypots, 25-28 June 2006

[19] Alata, Eric;Nicomette, V;Kaâniche, M;Dacier, Marc;Herrb, M
Lessons learned from the deployment of a high-interaction honeypot

[20] A Hybrid Honeypot Architecture for Scalable Network Monitoring
Michael Bailey, Evan Cooke, David Watson, Farnam Jahanian Niels Provos
University of Michigan October 27, 2004

[21] Hybrid Honeypot System for Network Security
Kyi Lin Lin Kyaw, Department of Engineering Physics, Mandalay Technological University

[22] Advanced Honeypot Architecture for Network Threats Quantification ,Robin G. Berthier 2009

[23] Know your enemy: Web Application Threats
http://www.honeynet.org/papers/webapp/

[24] A hybrid honeypot framework for improving intrusion detection systems in protecting organizational networks. Hassan Artail

[25] Honeynet Project, Sebek, Honeynet Project website
http://project.honeynet.org/tools/sebek/

[26] Shuja, F. (October, 2006). Virtual Honeynet: Deploying Honeywall using VMware. Available: http://www.honeynet.pk/honeywall/index.htm. Last accessed June, 2008.

[27] Know Your Enemy: Defining Virtual Honeynets
http://old.honeynet.org/papers/virtual/

[28] Psacct utility
http://linux.maruhn.com/sec/psacct.html

[29] SMF
http://www.simplemachines.org/

[30] Joel Weise and Brad Powell. Using computer forensics when investigating system attacks. Sun BluePrints OnLine, Sun Client Solutions Security Expertise Center, April 2005.

[31] Phillipine Honeypot project
http://www.philippinehoneynet.org/

[32] ProjectHoneypot http://www.projecthoneypot.org

[33] J. Howard and T. Longstaff. A common language for computer security incidents. Sandia Intelligence Labs, 1998.

[34] Lough, Daniel. “A Taxonomy of Computer Attacks with Applications to Wireless Networks,” PhD thesis, Virginia Polytechnic Institute and State University, 2001.

[35] Lindqvist U, Jonsson E. How to systematically classify computer security intrusions. IEEE Security and Privacy 1997:154e63.

[36] Hansman, S., Hunt R., “A taxonomy of network and computer attacks”. Computer and Security (2005).

[37] Common Vulnerabilities and Exposures (CVE) http://cve.mitre.org/

[38] National Vulnerability Database http://nvd.nist.gov/

[39] Forum of Incident Response and Security Teams (FIRST). Common Vulnerabilities Scoring System (CVSS). http://www.first.org/cvss/.

[40] MITRE Common Weakness Enumeration http://cwe.mitre.org/

[41] M.A. McQueen et al., “Time-to-Compromise Model for Cyber Risk Reduction Estimation”, Quality of Protection: Security Measurements and Metrics, Springer, 2005.

[42] Paulauskas N, Garsva E. Attacker skill level distribution estimation in the system mean time-to-compromise

[43] G. Álvarez, S. Petrović, 'A new taxonomy of web attacks suitable for efficient encoding,' Computers and Security, 22(5), pp. 435-449, 2003.

[44] Automated Reaction based on Risk Analysis and Attackers Skills in Intrusion Detection Systems (2009) Wael Kanoun, Nora Cuppens-boulahia, Frédéric Cuppens

[45] Olsson, Tomas (2009) Assessing Security Risk to a Network Using a Statistical Model of Attacker Community Competence. In: Eleventh International Conference on Information and Communications Security (ICICS 2009), 14-17 Dec 2009, Beijing, China.