Design and Development of Virtual Honeypot Framework based on Linux

Thesis submitted in partial fulfillment of the requirements for the award of degree of

Master of Engineering

In

Computer Science and Applications

Submitted By

Ashish Girdhar

601003003

Under the supervision of:

Ms. Sanmeet Kaur

Assistant Professor

SCHOOL OF MATHEMATICS AND COMPUTER APPLICATIONS

THAPAR UNIVERSITY

PATIALA – 147004

June 2012

CERTIFICATE

I hereby certify that the work which is being presented in the thesis entitled, "*Design and Development of Virtual Honeypot Framework based on Linux* ", in partial fulfillment of the requirements for the award of degree of Master of Technology in *Computer Science and Application* submitted in School of Mathematics and Computer Application of Thapar University, Patiala, is an authentic record of my own work carried out under the supervision of *Ms. Sanmeet Kaur* and refers other researcher's work which are duly listed in the reference section.

The matter presented in the thesis has not been submitted for award of any other degree of this or any other University.

Ashish Girdhar

(Ashish Girdhar)

This is to certify that the above statement made by the candidate is correct and true to the best of my knowledge.

Sanmeet Kaur

(Ms.Sanmeet Kaur)

Assistant Professor

SMCA

Countersigned by

(Dr. S.S.Bhatia)

Head

School of Mathematics and Computer Applications

Thapar University

Patiala

(Dr. S. K. Mohapatra)

Dean (Academic Affairs)

Thapar University

Patiala

ACKNOWLEDGEMENT

First and foremost, I would like to express my sincere gratitude to my guide **Ms. Sanmeet Kaur,** Assistant Professor, School of Mathematics and Computer Applications for immense help, guidance, stimulating suggestions, and encouragement all the time with this thesis work, this work would have not been possible without her encouragement. She has provided me with all the necessary resources including motivation and research environment without which I could not complete this work. It was a great opportunity for me to do this work under her supervision.

I would like to thank Dr. S.S.Bhatia Professor and head, School of Mathematics and Computer Applications for his moral support and the research environment he had facilitated for this work.

I would like to thank all my teachers for their stimulating discussion and invaluable support I received during the period of this research. I am thankful to all the authors whose works I have consulted and quoted in this work.

Finally I wish to thank my family and friends for all their immense love, enthusiastic encouragement and support throughout my life without which I could not complete this work. Last but not least I would like to thank the God who has always been with me in my good and bad times.

Ashish Girdhar

(601003003)

ABSTRACT

A honeypot is a closely monitored network decoy serving several purposes: it can distract adversaries from more valuable machines on a network, provide early warning about new attack and exploitation trends, or allow in-depth examination of adversaries during and after exploitation of a honeypot.

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.

Honeypots are a relatively new technique for achieving network security. While other techniques for securing networks e.g. IDS, Firewall etc are made to keep the attackers out, for the first time in the history of network security there is a technique which intends to keep the attackers 'in' thus allowing the researchers to gain more insight into the workings of an attacker.

This thesis describes the design and implementation of Honeyd, a framework for virtual honeypots that simulates computer systems at the network level. Here we have tried to integrate the latest snort with the implemented solution to get the alerts of known attacks. Also an effort has been made to generate the automatic reports to get better view of the attacks occurring in the network.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter1 INTRODUCTION

Now a day's Internet has become the largest public data network, enabling and facilitating both personal and business communications worldwide. The volume of traffic moving over the Internet, as well as corporate networks, is expanding exponentially every day, so network and computer systems connected to it are still too vulnerable and attacks are becoming ever more frequent. Computer security is severely threatened by software vulnerabilities to identify and extract their fundamental characteristics; in depth analysis of vulnerabilities is required. Computer Networks are vulnerable to a variety of exploits that can compromise their intended operations.

1.1 Network Security

Network security is an ongoing process that helps to keep unauthorized parties away from gaining access on the network. The goal of network security is to support the network and computer business requirements, using methods that reduce risk. Network security is a prominent feature of the network ensuring accountability, confidentiality, integrity, availability, and above all protection against many external and internal threats such as email based network security problems, denial of service network security attacks, worms and Trojans, and wireless network security attacks. Network security is vital to keep hackers away from viewing sensitive information. Implementation of effective network security provides both physical and information security to paths, links, and databases. Security policies describe what must be secured to support the business or mission [1].

1.2 Need of Security

Today security is very important because as technology is progressing the computer systems are getting more and more user friendly and hence easier to use. Today computer users range from pre-teens to senior citizens as compared to earlier when computers where only in the domain of scientists in big corporations and universities. Also in order

to sustain ease of use today computers are becoming more and more complex to administer. Thus to sum up the need of security the following arguments can be given:

i. Evolution of technology focused on ease of use.
ii. Increasing complexity of computer infrastructure, administration and management.

As technology is progressing the computers are getting more and more complex and hence the ways to attack them i.e. Attack Vectors are also getting more and more complex. So it is becoming easy for attacker to attack the systems..This can be best summed up by the graph in Figure 1.1.

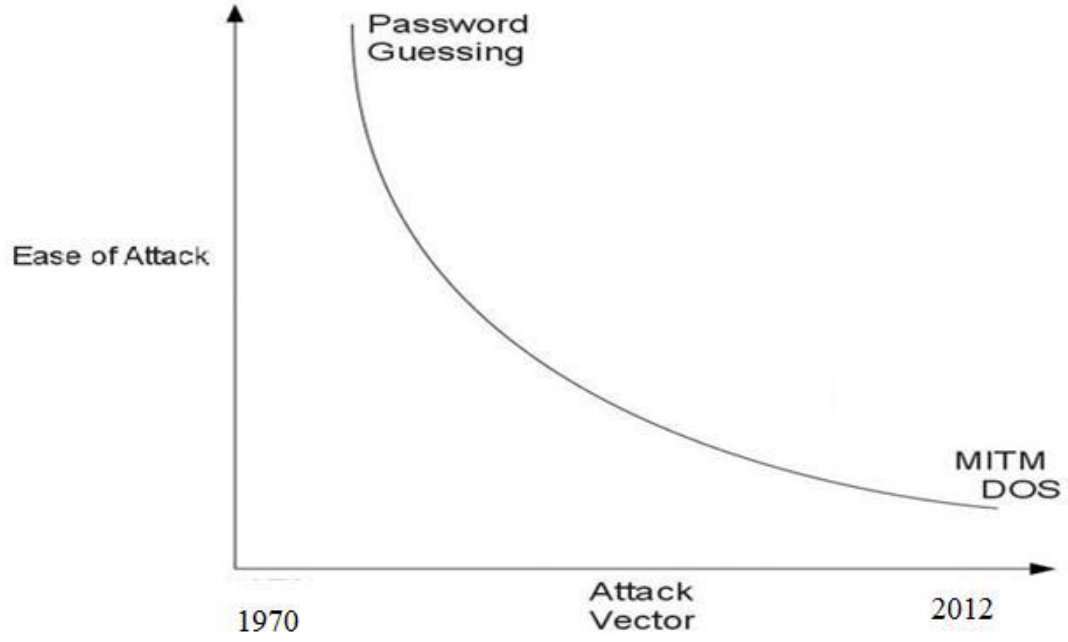

Figure1.1: Ease of Attack vs. Attack Vector with time [1]

The graph shows that with time as computers are getting complex the attack vectors are also getting very complex. Earlier password guessing was used to break into computers, nowadays attack vectors are shifted to more complex things like Man-In-The-Middle, Denial-Of-Service (MITM, DOS) attacks. With the increase in attack vector with time it is becoming easy for attacker to attack the systems.

1.3 Threats to Security

A threat is anything that can disrupt the operation, functioning, integrity, or availability of a network or system. There are various threats to network security such as:

- **Viruses**
- **Worms**
- **Trojans**
- **Attacks**

1.3.1 Viruses

A computer virus is a type of malware that propagates by inserting a copy of itself into and becoming part of another program. It spreads from one computer to another, leaving infections as it travels. Viruses can range in severity from causing mildly annoying effects to damaging data or software and causing denial-of-service (DoS) conditions. Almost all viruses are attached to an executable file, which means the virus may exist on a system but will not be active or able to spread until a user runs or opens the malicious host file or program. When the host code is executed, the viral code is executed as well. Normally, the host program keeps functioning after it is infected by the virus. However, some viruses overwrite other programs with copies of themselves, which destroys the host program altogether. Viruses spread when the software or document they are attached to is transferred from one computer to another using the network, a disk, file sharing, or infected e-mail attachments [2]**.**

1.3.2 Worms

Computer worms are similar to viruses in that they replicate functional copies of themselves and can cause the same type of damage. In contrast to viruses, which require the spreading of an infected host file, worms are standalone software and do not require a host program or human help to propagate. To spread, worms either exploit vulnerability on the target system or use some kind of social engineering to trick users into executing them. A worm enters a computer through vulnerability in the system and takes advantage of file-transport or information-transport features on the system, allowing it to travel unaided [2].

1.3.3 Trojans

A Trojan is another type of malware named after the wooden horse the Greeks used to infiltrate Troy. It is a harmful piece of software that looks legitimate. Users are typically tricked into loading and executing it on their systems. After it is activated, it can achieve any number of attacks on the host, from irritating the user (popping up windows or changing desktops) to damaging the host (deleting files, stealing data, or activating and spreading other malware, such as viruses). Trojans are also known to create back doors to give malicious users access to the system.

Unlike viruses and worms, Trojans do not reproduce by infecting other files nor do they self-replicate. Trojans must spread through user interaction such as opening an E-mail attachment or downloading and running a file from the Internet [2].

1.3.4 Attacks

There are various types of attacks which can occur in a network such as:

- **Man in the Middle Attack (MITM)**
- **Denial of Service Attack (DOS)**
- **SYN Flooding Attacks**
- **Man in the Middle Attack (MITM)**

  A man in the middle attack is one in which the attacker intercepts messages in a public key exchange and then retransmits them, substituting his own public key for the requested one, so that the two original parties still appear to be communicating with each other. The attack gets its name from the ball game where two people try to throw a ball directly to each other while one person in between them attempts to catch it. In a man in the middle attack, the intruder uses a program that appears to be the server to the client and appears to be the client to the server. The attack may be used simply to gain access to the message, or enable the attacker to modify the message before retransmitting it.

  Man in the middle attack is sometimes known as fire brigade attack. The term derives from the bucket brigade method of putting out a fire by handing buckets of water from one person to another between a water source and the fire.

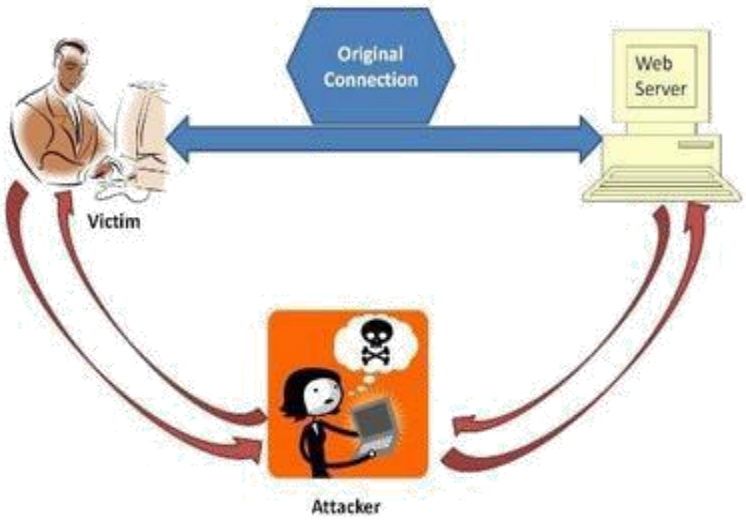

Figure1.2: MITM [3]

- **Denial of Service Attack (DoS)**

  In a denial-of-service (DoS) attack, an attacker attempts to prevent legitimate users from accessing information or services. By targeting the computer and its network connection, or the computers and network of the sites that the users are trying to use, an attacker may be able to prevent them from accessing email, websites, online account or other services that rely on the affected computer.

  The most common and obvious type of DoS attack occurs when an attacker "floods" a network with information. When the user types a URL for a particular website into the browser, they are sending a request to that site's computer server to view the page. The server can only process a certain number of requests at once, so if an attacker overloads the server with requests, it can't process your request. This is a "denial of service" because the user can't access that site.

  An attacker can use spam email messages to launch a similar attack on user's email account. Whether the users have an email account supplied by the employer or one available through a free service such as Yahoo or Hotmail, they are assigned a specific quota, which limits the amount of data they can have in their account at any given time. By sending many, or large, email messages to the account, an attacker can consume their quota, preventing them from receiving legitimate messages [4].

- **SYN Flooding**

  SYN flooding is a denial-of-service attack that exploits the three-way handshake that TCP/IP uses to establish a connection. Basically, SYN flooding disables a targeted system by creating many half-open connections. Figure 1.3 illustrates how a typical TCP/IP connection is established.

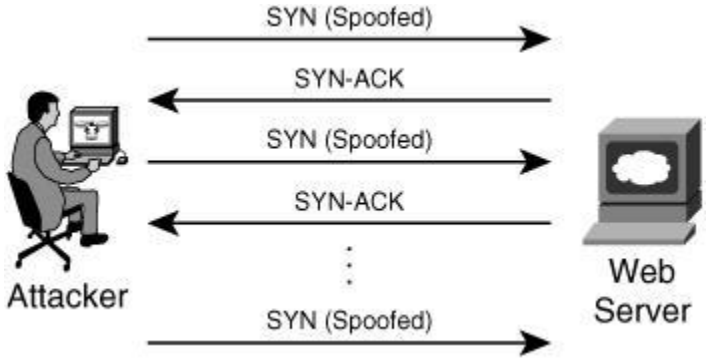

Figure 1.3: SYN flooding attack [5]

In Figure 1.3, the client transmits to the server the SYN bit set. This tells the server that the client wishes to establish a connection and what the starting sequence number wishes to establish a connection and what the starting sequence number will be for the client. The server sends back to the client an acknowledgment (SYN-ACK) and confirms its starting sequence number. The client acknowledges (ACK) receipt of the server's transmission and begins the transfer of data.

1.4 Security Measures:-

We can achieve network security by following methods

- **Firewalls**
- **Intrusion Detection Systems**
- **Honeypots**

1.4.1 Firewall

A firewall is a device that serves as a barrier between networks providing access control, traffic filtering, and other security features. Firewalls are commonly deployed between trusted and untrusted networks, for example between the Internet (untrusted) and an organization’s trusted private network. They can also be used internally to segment an

organization's network infrastructure, for example; deploying a firewall between the corporate financial information and the rest of the company network [6].

There are mainly three types of firewalls

i. **Packet Filter Firewalls**

The first basic type of firewall is commonly referred to as a packet-filter firewall. These firewalls work on the principal that every packet should adhere to a static rule base that can be defined by the user. The payload of the packet is not inspected for validity. In most cases, each packet will be treated as an isolated item. These types of firewalls tend to be fairly fast and are transparent to end-users.

ii. **Circuit-Level Firewalls**

These firewalls allow and deny packets based upon an administrator's defined rule base. This type of firewall can inspect multiple levels of the packet, and some contain rudimentary packet reconstruction. Many circuit-level firewalls inspect a few packets of the conversation against the rule base before documenting the

'circuit' or 'state' of the communication and allowing the data to pass without inspection. These types are also susceptible to upper OSI level attacks and IP spoofing.

iii. **Application Gateway Firewalls**

These types of firewalls are sometimes referred to as proxy firewalls. These firewalls can filter based on IP addresses and the specific function that an application is trying to execute. For example, application firewalls can prevent specified applications such as Microsoft NetMeeting or FTP from getting through. By monitoring application function, the firewall can even permit some application operations while blocking others. One example of this is a FTP site that permits the user to upload files to specified directories without permitting him to view or modify files in the directories. For example, McAfee and Zone Alarm all employ features of the application gateway model [7].

1.4.2 Intrusion Detection Systems

Intrusion detection systems serve three essential security functions: they monitor, detect, and respond to unauthorized activity. Intrusion detection systems use policies to define

certain events, if detected will issue an alert or respond automatically to the event. Such a response might include logging off a user, disabling a user account and launching of scripts.

There are two basic types of Intrusion Detection System:

i. **Host Based IDS**

Host-based IDSs examine data held on individual computers that serve as hosts; they are highly effective for detecting insider abuses. Examples of host-based IDS implementations include Windows NT/2000 Security Event Logs, and UNIX Syslog.

ii. **Network Based IDS(NIDS)**

Network based intrusion detection systems analyze data packets that travel over the actual network. These packets are examined and sometimes compared with empirical data to verify whether they are of malicious or benign nature. An example of NIDS is Snort, which is an open source network intrusion detection system that performs real-time traffic analysis. It can be used to detect a variety of attacks and probes, such as buffer overflows, stealth port scans, and OS fingerprinting attempts [8].

1.4.3 **Honeypots**

A honeypot is an information system resource whose value lies in unauthorized or illicit use of that resource. Honeypot is an exciting technology with great potential for the field of network security. It can be understood as a resource used to divert attackers and hackers away from critical resources i.e. it is an observed trap. It can also be used to study an attacker's methods and tools. The value of a Honeypot lies in unauthorized and illicit use. Neither any authorized activity runs on these resources nor do they have any production value i.e. no legitimate activity is carried out. It provides a large amount of valuable information for analysis and can detect variety of attacks, working even within encrypted environment. It acts as a cherished observation and early warning tool but on the contrary it should be used with caution as it has risks associated with it.

1.5 Comparison between Honeypots, Firewalls and IDS

A comparison is made between Honeypots, Firewalls and IDS in the following section.

- **Honeypots vs Firewalls**

  A honeypot is quite different from a firewall. A firewall is designed to keep the bad guys out of the network whereas honeypots are designed to entice the hackers to attack the system so that a security researcher can know how hackers operate once they have access of the system or a security analyst in a corporation can know which systems and ports the hackers are most interested in. Also firewalls log activities to all the systems of a corporation, so in case of an event it becomes very difficult to shift through all the logs in order to find a particular event log as logs also contains events related to production systems. However in case of honeypot, the logs are only due to non-productive systems, these are the systems that no one should be interacting with. So if a firewall log contains 1000 entries of all the systems of the network the honeypot's log only contain only 5-10 entries each of which is important for a researcher.

- **Honeypots vs IDS**

  The amount of useful information provided by NIDS is decreasing in the face of ever more sophisticated evasion techniques and an increasing number of protocols that employ encryption to protect network traffic from eavesdroppers. NIDS also suffer from high false positive rates that decrease their usefulness even further. Honeypots can help with some of these problems. A honeypot is a closely monitored computing resource that one intends to be probed, attacked, or compromised. The value of a honeypot is determined by the information that can be obtained from it. Monitoring the data that enters and leaves a honeypot lets us gather information that is not available to NIDS. For example, one can log the key strokes of an interactive session even if encryption is used to protect the network traffic. To detect malicious behavior, NIDS require signatures of known attacks and often fail to detect compromises that were unknown at the time it was deployed. On the other hand, honeypots can detect vulnerabilities that are not yet

understood. For example, one can detect compromise by observing network traffic leaving the honeypot even if the means of the exploit has never been seen before. Because a honeypot has no production value, any attempt to contact it is suspicious. Consequently, forensic analysis of data collected from honeypots is less likely to lead to false positives than data collected by NIDS [9].IDS and firewalls are the traditional approaches to network security. The goal of IDS is to

"identify, preferably in real time, unauthorized use, misuse, and abuse of computer systems by both system insiders and external penetrators". IDS is used as an alternative for building a shield around the network. The shielding approach is deficient in several ways, including failure to prevent attacks from insiders. IDS often depend upon signature matching or statistical models to identify attacks. This means that unknown or novel threats may not be detected. In contrast, honeypots are designed to capture all known and unknown attacks directed against them. Because any network activity related to the honeypot represents an anomaly, even the stealthiest activity will register on a honeypot. Because honeypots operate at the host level, encrypted or non-IPv4 communications that can often blind traditional network based sensors can still be captured.

Chapter 2 LITERATURE SURVEY

2.1 Introduction to Honeypot

A honeypot is a closely monitored network decoy serving several purposes: it can distract adversaries from more valuable machines on a network, provide early warning about new attack and exploitation trends and allow in-depth examination of adversaries during and after exploitation of a honeypot. 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" [10].Honeypots as an easy target for the attackers can simulate many vulnerable hosts in the network and provide us with valuable information of the attackers. Honeypots are not the solution to the network security but 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.

In general the term 'honeypot' is usually being used for representing "a container (or pot) of honey". But in the case of computer security, this term is being used to represent a computer security concept that is solely based on deception [11]. Honeypot is a resource to trap the attacker's tools and activities. Lance Spitzner, the founder of The Honeynet Project organization, defines a honeypots as:

"Honeypot is a security resource whose value lies in being probed, attacked or compromised" [12].

This definition tells the nature of honeypot. It means that if no one attacks honeypot, it is nothing. But honeypot is a valuable security tool if it is being attacked by the attacker. Other security tools such as firewall and IDS are completely passive for that their task is to prevent or detect attacks. Honeypot actively give a way to attacker to gain information about new intrusions. This nature makes honeypot outstanding to aid other security tools. Honeypot differ according to different use. It could be an emulated application, a fully

functional operating system with default configuration or an actual net including different OS and applications, even an emulated network on a single machine.

Honeypots are very different, and it is this difference that makes them such a powerful tool. Honeypots do not solve a specific problem. Instead, they are a highly flexible tool that has many applications to security. They can be used to slow down or stop automated attacks, capture new exploits to gather intelligence on emerging threats or to give early warning and prediction. They come in many different shapes and sizes. They can be either a Windows program that emulates common services, such as the Windows honeypot KFSensor3 or entire networks of real computers to be attacked, such as Honeynets.

2.2 Types of Honeypots

In general honeypots can be divided in to two categories:

- Production honeypots
- Research honeypots [13]

2.2.1 Production Honeypots:-Production honeypots are used to assist an organization in protecting its internal IT infrastructure. They are valuable to the organization especially commercial, as they help to reduce the risk that a specific organization faces. They secure the organization by policing its IT environment to identify attacks. These honeypots are useful in catching hackers with criminal intentions. The implementation and deployment of these honeypots are relatively easier than research honeypots .One of the reasons is that they have less purpose and require fewer functions. As a result, they also provide less evidence about hacker's attack patterns and motives**.**

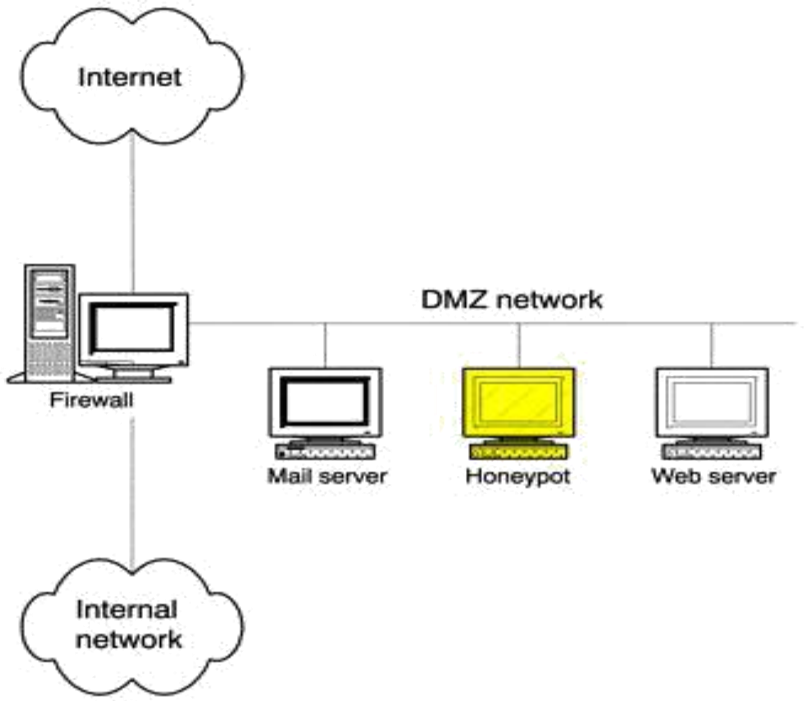

Figure 2.1: Network Diagram of a production honeypot deployed on a DMZ to detect attacks [13]

2.2.2 Research Honeypots: - Research honeypots are complex. They are designed to collect as much information as possible about the hackers and their activities. They are not specifically valuable to an organization. Their primary mission is to research the threats organization may face, such as who the attackers are, how they are organized, what kind of tools they use to attack other systems, and where they obtained those tools. While production honeypots are like the police, research honeypots act as their intelligence counterpart and their mission is to collect information about the attacker. The information gathered by research honeypots will help the organization to better understand the hackers attack patterns, motives and how they function. They are also an excellent tool to capture automated attacks such as worms.

2.3 Classification of Honeypots

According to the level of involvement between the attacker and the honeypots, the honeypots can be divided into three categories:

- Low-interaction honeypots
- Medium-interaction honeypots
- High-interaction honeypots.

2.3.1 Low-Interaction Honeypots*:* Low-interaction honeypots are the easiest to install, configure, deploy, and maintain because of their simple design and basic functionality. Normally these technologies merely emulate a variety of services. The attacker is limited to interacting with these pre designated services. For example, a low-interaction honeypot could emulate a standard UNIX server with several running services, such as Telnet and FTP. An attacker could Telnet to the honeypot, get a banner that states the operating system, and perhaps obtain a login prompt. The attacker can then attempt to login by brute force or by guessing the passwords. The honeypot would capture and collect these attempts, but there is no real operating system for the attacker to log on to. The attacker's interaction is limited to login attempts.

Since low-interaction honeypots are simple, they have the lowest level of risk. There is little functionality offered, there is less to go wrong. There is also no operating system for the attacker to interact with, so the honeypot cannot be used to attack or monitor other systems. Low-interaction honeypots are easy to deploy and maintain because they have limited interaction capabilities, which also reduces risk [14].

2.3.2 Medium-interaction Honeypots: In terms of interaction, medium-interaction honeypots are more advanced than low-interaction honeypots, but less advanced than high interaction honeypots. Medium-Interaction honeypots also do not have a real operating system, but the services provided are more sophisticated technically. Here the levels of honeypots get complicated so the risk also increases especially with regards to vulnerability.

2.3.3 **High-interaction Honeypots:** High-interaction honeypots are different; they are a complex solution and involve the deployment of real operating systems and applications. They capture the extensive amounts of information and allowing attackers to interact with real systems where the full extent of their behavior can be studied and recorded. Examples of high-interaction honeypots include Honeynets and Sebek. These kinds of

honeypots are really time consuming to design, manage and maintain. Among the three types of honeypots, these honeypots possess a huge risk. But, the information and evidence gathered for analysis is very large. With these types of honeypots we can learn what are the kind of tools hackers use, what kind of exploits they use, what kind of vulnerabilities they normally look for, their knowledge in hacking and surfing their way through operating systems and how or what the hackers interact about[14].

2.4 Tradeoffs between Honeypot Levels of Interaction

Table 2.1 summarizes the tradeoffs between different levels of interaction in four categories.

The first category is installation and configuration effort, which defines the time and effort in installing and configuring the honeypot. In general, if the level of interaction between the user and the honeypot is more then the effort required to install and configure the honeypot is also significant.

The second category is deployment and maintenance. This category defines the time and effort involved in deploying and maintaining the honeypot. Once again, the more functionality provided by the honeypot, the more is the effort required to deploy and maintain the honeypot.

The third category is information gathering which means how much information can the honeypot gain on attackers and their activities? High-interaction honeypots can gather vast amounts of information, whereas low-interaction honeypots are highly limited. Finally, level of interaction impacts the amount of risk introduced. The greater the level of interaction, the more functionality provided to the attacker and the greater the complexity. Combined, these elements can introduce a great deal of risk. On the other hand, low-interaction honeypots are very simple and offer a little interaction to attackers and thus a very little risk is associated with them.

Degree of involvement	Low	Medium	High
Installation and configuration effort	Easy	Medium	Difficult
Deployment and maintenance effort	Easy	Medium	Difficult
Information Gathering	Limited	Medium	Extensive
Level of Risk	Low	Medium	High

Table 2.1: Tradeoffs between Honeypot Levels of Interaction [10]

2.5 Different Honeypots Systems: Five honeypots are discussed in the following section.

- ManTrap
- Back Officer Friendly
- Specter
- Honeyd
- Honeynet

2.5.1 ManTrap: ManTrap is a high-interaction commercial honeypot created, maintained, and sold by Recourse Technologies. ManTrap creates a highly controlled operating environment that an attacker can interact with. It creates a fully functional operating system containing virtual cages rather than a limited operating system. The cages are logically controlled environments from which the attacker is unable to exit and attack the host system. However, instead of creating an empty cage and filling it with certain functionality ManTrap creates cages that are mirror copies of the master operating system. Each cage is a fully functional operating system that has the same capabilities as a production installation.

This approach creates a very powerful and flexible solution. Each cage is its own virtual world with few limitations. An administrator can customize each cage as he would a physically separate system. He can create users, install applications, run processes, and even compile his own binaries. When an intruder attacks and gains access to a cage, to the attacker it looks as if the cage is a truly separate physical system. He is not aware that he is in a caged environment where every action and keystroke is recorded [15].

2.5.2 Back Officer Friendly (BOF): Back Officer Friendly, or BOF as it is commonly called, is a simple, free honeypot solution developed by Marcus Ranum. It is extremely simple to install, easy to configure, and low maintenance. However, this simplicity comes at a cost. Its capabilities are severely limited. It has a small set of services that simply listen on ports, with notably limited emulation capabilities.

It works by creating port listeners, or open sockets, that bind to a port and detect any connections made to these ports. When a connection is made to the port, the port listeners establish a full TCP connection (if the service is TCP), log the attempt, generate an alert, and then close the connection, depending on how the service is configured. Everything BOF does happen in user space. It does not build or customize any packets when responding to connections. Because of this simple model, BOF can run on any Windows platform, including Windows 95 and Windows 98[10].

2.5.3 Specter: Specter is a commercially supported honeypot developed and sold by the folks at NetSec. Like BOF, Specter is a low-interaction honeypot. However, Specter has far greater functionality and capabilities than BOF. Not only can Specter emulate more services, it can emulate different operating systems and vulnerabilities. It also has extensive alerting and logging capabilities. Because Specter only emulates services with limited interaction, it is easy to deploy, simple to maintain, and is low risk. However, compared to medium- and high-interaction honeypots, it is limited in the amount of information it can gather. Specter is primarily a production honeypot. Specter shares the same limitations as BOF. Specifically, it cannot listen on or monitor a port that is already owned by another application. If any service listening on the FTP port (port 21), then Specter is unable to monitor on that port. Specter can only monitor ports that are not owned by any other applications. It also has the capability of emulating different operating systems. This is done by changing the behavior of the

services to mimic the selected operating system [15].

2.5.4 Honeyd: Honeyd is developed and maintained by Niels Provos of the University of Michigan and was first released in April 2002. It is designed as a low-interaction solution; there is no operating system intended for an attacker to gain access to, only emulated services. Honeyd is designed primarily as a production honeypot, used to detect attacks or unauthorized activity [10].

Honeyd works on the principle that when it receives a probe or a connection for a system that does not exist, it assumes that the connection attempt is hostile, most likely a probe, scan, or attack. When Honeyd receives such traffic, it assumes the IP address of the intended destination (making it the victim). It then starts an emulated service for the port that the connection is attempting. Once the emulated service is started, it interacts with the attacker and captures all of his activity. When the attacker is done, the emulated service exits and is no longer running. Honeyd then continues to wait for any more traffic or connection attempts to systems that do not exist. Honeyd assumes an IP address and runs an emulated service only when it receives a connection attempted for a system that does not exist, an extremely efficient method. As Honeyd receives more attacks, it repeats the process of assuming the IP address of the intended victim, starting the respective emulated service under attack, interacting with the attacker, and capturing the attack, and finally exiting. It can emulate multiple IP addresses and interact with different attackers all at the same time.

2.5.5 Honeynets: Honeynets represent the extreme of high-interaction honeypots. Not only does it provide the attacker with a complete operating system to attack and interact with, it may also provide multiple honeypots. Honeynets are nothing more than a variety of standard systems deployed within a highly controlled network. By their nature, these systems become honeypots, since their value is in being probed, attacked, or compromised. The controlled network captures all the activity that happens within the Honeynet and decreases the risk by containing the attacker's activity.

Honeynets are a simple mechanism that works on the same principle as a honeypot. You create a resource that has little or no production traffic. Anything sent to the Honeynet is suspect, potentially a probe, scan, or even an attack. Anything sent from a Honeynet implies

that it has been compromised— an attacker or tool is launching activity. However, Honeynets take the concept of honeypots one step further: Instead of a single system, a Honeynet is a physical network of multiple systems.

Honeynets are not a product you install or an appliance you drop on your network. Instead, Honeynets are an architecture that builds a highly controlled network, within which you can place any system or application you want [16].

2.6 Comparison of various Honeypots

In this section five honeypots are compared in the tabular form.

- The interaction level between the user and the Honeypot is high in case of Mantrap, specter and Honeynet and this level is low in case of BOF and Honeyd.
- Honeyd and Honeynet are freely available whereas Mantrap, specter and BOF are not freely available.
- Honeyd and Honeynet are open source whereas Mantrap, specter and BOF are not open source.
- BOF does not support Log file whereas rest of the honeypots support log file.
- BOF does not emulate the operating system whereas rest of the four honeypots can emulate operating system.
- Unlimited services are supported by the ManTrap, Honeyd and Honeynet whereas limited services are supported by the BOF and specter.

	ManTrap	BOF	Specter	Honeyd	Honeynet
Interaction Level	High	Low	High	Low	High
Freely Available	No	No	No	Yes	Yes
Open Source	No	No	No	Yes	Yes
Log file Support	Yes	No	Yes	Yes	Yes
OS Emulation	Yes	No	Yes	Yes	Yes
Supported Services	Unrestricted	7	13	Unrestricted	Unrestricted

Table 2.2: Comparison of various honeypots [10]

2.7 Advantages of Honeypots

1. **Small Data Sets:** Honeypots only collect data when someone or something is interacting with them. Organizations that may log thousands of alerts a day with traditional technologies will only log a hundred alerts with honeypots. This makes the data honeypots collect much higher value, easier to manage and simpler to analyze.

2. **Reduced False Positives:** One of the greatest challenges with most detection technologies is the generation of false positives or false alerts. It's similar to the story of the 'boy who cried wolf'. The larger the probability that a security technology produces a false positive the less likely the technology will be deployed. Honeypots dramatically reduce false positives. Any activity with honeypots is by definition unauthorized, making it extremely efficient at detecting attacks.

3. **Catching False Negatives:** Another challenge of traditional technologies is failing to detect unknown attacks. This is a critical difference between honeypots and traditional computer security technologies which rely upon known signatures or upon

statistical detection. Signature-based security technologies by definition imply that "someone is going to get hurt" before the new attack is discovered and a signature is distributed. Statistical detection also suffers from probabilistic failures – there is some non-zero probability that a new kind of attack is going to go undetected. Honeypots on the other hand can easily identify and capture new attacks against them. Any activity with the Honeypot is an anomaly, making new or unseen attacks easily stand out.

4. **Encryption:** It does not matter if an attack or malicious activity is encrypted, the Honeypot will capture the activity. As more and more organizations adopt encryption within their environments (such as SSH, IPSec, and SSL) this becomes a major issue. Honeypots can do this because the encrypted probes and attacks interact with the Honeypot as an end point, where the activity is decrypted by the Honeypot.

5. **IPv6:** Honeypots work in any IP environment, regardless of the IP protocol, including IPv6. IPv6 is the new IP standard that many organizations, such as the Department of Defense, and many countries, such as Japan, are actively adopting. Many current technologies, such as firewalls or IDS sensors, cannot handle IPv6.

6. **Highly Flexible:** Honeypots are extremely adaptable, with the ability to be used in a variety of environments, everything from a Social Security Number embedded into a database, to an entire network of computers designed to be broken into.

7. **Minimal Resources**: Honeypots require minimal resources, even on the largest of networks. A simple, aging Pentium computer can monitor literally millions of IP addresses [17].

2.8 Disadvantages of Honeypots

Apart from all the advantages, honeypots also have some disadvantages. Disadvantages of honeypots are listed below:

1. **Risk:** Honeypots are a security resource the bad guys to interact with, there is a risk that an attacker could use a honeypot to attack or harm other non-honeypot systems. This risk varies with the type of honeypot used. For example, simple honeypot such

as KFSensor has very little risk. Honeynets, a more complex solution, have a great deal of risk [18]. The risk levels are variable for different kinds of honeypot deployments. The usual rule is that the more complicated the deception, the greater the risk. Honeypots that are high-interaction such as Gen I Honeynets are inherently more risky because there is an actual computer involved.

2. **Limited Field of View:** Honeypots only see or capture that which interacts with them. They are not a passive device that captures activity to all other systems. Instead, they only have value when directly interacted with. In many ways honeypots are like a microscope. They have a limited field of view, but a field of view that gives them great detail of information.

3. **Discovery and Fingerprinting:** Though risk of discovery of a honeypot is small for script kiddies and worms, there is always a chance that advanced blackhats would be able to discover the honeypot [19]. A simple mistake in the deception is all a savvy attacker needs to "fingerprint" the honeypot. This could be a misspelled word in one service emulation or even a suspicious looking content in the honeypot. The hacker would be able to flag the honeypot as "dangerous" and in his next attacks; he would most certainly bypass the honeypot. In fact, armed with the knowledge, an advanced blackhat could even spoof attacks to the honeypot thus redirecting attention while he attacks other vulnerable systems in the network [20].

2.9 Architecture of Honeyd

Honeyd's architecture consists of several components: a configuration database, a central packet dispatcher, protocol handlers, a personality engine, and an optional routing component.

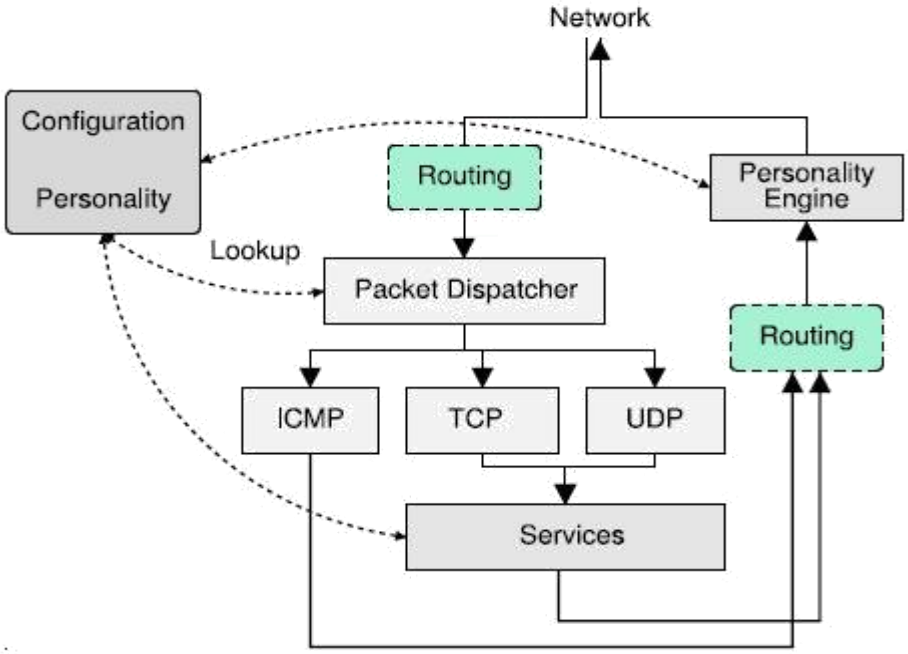

Figure 2.2 Architecture of Honeyd [9]

Incoming packets are processed by the central packet dispatcher. It first checks the length of an IP packet and verifies the packet's checksum. The framework is aware of the three major Internet protocols: ICMP, TCP and UDP. Packets for other protocols are logged and silently discarded. Before it can process a packet, the dispatcher must query the configuration database to find a honeypot configuration that corresponds to the destination IP address. If no specific configuration exists, a default template is used. Given a configuration, the packet and corresponding configuration is handed to the protocol specific handler. The ICMP protocol handler supports most ICMP requests. By default, all honeypot configurations respond to echo requests and process destination unreachable messages. The handling of other requests depends on the configured personalities. For TCP and UDP, the framework can establish connections to arbitrary services. Services are external applications that receive data on stdin and send their output to stdout. The behavior of a service depends entirely on the external application. When a connection request is received, the framework checks if the packet is part of an established connection. In that case, any new data is sent to the already

started service application. If the packet contains a connection request, a new process is created to run the appropriate service. Instead of creating a new process for each connection, the framework supports subsystems and internal services. A subsystem is an application that runs in the name space of the virtual honeypot. The subsystem specific application is started when the corresponding virtual honeypot is instantiated. A subsystem can bind to ports, accept connections, and initiate network traffic. While a subsystem runs as an external process, an internal service is a Python script that executes within Honeyd. Internal services require even less resources than subsystems but can only accept connections and not initiate them [9].

Honeyd contains a simplified TCP state machine. The three-way handshake for connection establishment and connection tear down via FIN or RST is fully supported, but receiver and congestion window management is not fully implemented. UDP datagrams are passed directly to the application. When the framework receives a UDP packet for a closed port, it sends an ICMP port unreachable message unless this is forbidden by the configured personality. In sending ICMP port unreachable messages, the framework allows network mapping tools like trace route to discover the simulated network topology [21]. In addition to establishing a connection to a local service, the framework also supports redirection of connections. The redirection may be static or it can depend on the connection quadruple (source address, source port, and destination address and destination port). Redirection lets us forward a connection request for a service on a virtual honeypot to a service running on a real server. For example, we can redirect DNS requests to a proper name server. Or we can reflect connections back to an adversary, e.g. just for fun we might redirect an SSH connection back to the originating host and cause the adversary to attack her own SSH server. Evil laugh. Before a packet is sent to the network, it is processed by the personality engine. The personality engine adjusts the packet's content so that it appears to originate from the network stack of the configured operating system.

2.9.1 Honeyd's Features: Various features of Honeyd are:-

- Honeyd has the ability to monitor millions of IP address and actively work as an IP address and all the functions work simultaneously.

- Honeyd has a corresponding fingerprint matching mechanism and fingerprint tool which can cheat attacker.
- Honeyd can interact with hackers by loading corresponding scripts with the form of plug-in and these scripts can imitate corresponding services to deepen interaction with attackers.
- Honeyd runs as a background process, and the honeypot generated by it is simulated by the background process. Therefore, the host can run Honeyd to control the system security effectively.
- Honeyd can produce many virtual honeypots in the network, and these honeypots can be both a loose collective between them and a tight network system, thus to form a virtual honeypot network [22].

2.10 Snort: Snort is an IDS which works on pre-defined rule sets known as signatures, if any attacks happens in to system and matches corresponding signatures, it will generate signatures as per their database. Snort is an open source network intrusion detection system (NIDS) created by Martin Roesch. Combining the benefits of signature, protocol, and anomaly-based inspection, Snort is the most widely deployed IDS/IPS technology worldwide. Snort is based on libpcap (for library packet capture), a tool that is widely used in TCP/IP traffic sniffers and analyzers. Through protocol analysis and content searching and matching, Snort detects attack methods, including denial of service, buffer overflow, CGIattacks, stealth port scans, and SMB probes. When suspicious behavior is detected, Snort sends a real-time alert to syslog or a separate alerts file [23].

Snort can be configured to run in four modes:

i. **Sniffer mode**, which simply reads the packets off of the network and displays them in a continuous stream on the console (screen).
ii. **Packet Logger mode**, which logs the packets to disk.
iii. **Network Intrusion Detection System (NIDS) mode**, the most complex and configurable configuration, which allows Snort to analyze network traffic for matches against a user-defined rule set and performs several actions based upon what it sees [24].

iv. **Inline Mode**, which obtains packets from ip tables instead of from libpcap and then causes iptables to drop or pass packets based on Snort rules that use inline-specific rule types .

Chapter 3 PROBLEM STATEMENT

3.1 Problem Definition

Honeypots are a new field in network security and it is very useful to signify the working of traditional network security devices like intrusion detection system, firewalls etc. As most of the traditional network security devices are working on pre-defined rules sets known as signatures, then what happens in case of attacks into network when there are no signatures in their database? We have tried here to design and develop a virtual honeypot framework based on Linux. We have also integrated the latest snort with the implemented solution to get the alerts of known attacks. Also an effort has been made to generate the automatic reports to get better view of the attacks occurs in network. The question arise why virtual environment has been chosen as that is also a one area of research, the answer of this question is obvious fewer resources are required, meaning that for this work only one PC is required on which multiple Honeypots can be set up which are not interlinked to each other.

3.2 Objectives

1. To design and develop a virtual Honeypot framework based on Linux.
2. To implement “Honeyd” as it is the most widely used honeypot to simulate the network.
3. To represent the attack’s information graphically.

Chapter 4 IMPLEMENTATION DETAILS AND RESULTS

The solution of the problem discussed in the previous section is to design and develop a honeypot. It is designed to lure the hackers and intruders in order to monitor their malicious activities and observe how they break into a computer. The production system is integrated with this designed honeypot with fake operating system and fake services running on it, to engage the intruder with virtual operating system and services. Data gathered by this honeypot is valuable and can also help the administrator.

4.1 Experimental Setup

The experimental setup of our work is shown in the figure 4.1.Initially we assign a static IP addresses to the window OS, virtual machine. We assign two unused IP address to two virtual honeypots respectively. We assign 192.168.1.243 IP address to virtual honeypot1 and 192.168.1.244 IP address to virtual honeypot2.These virtual honeypots corresponds to two different personalities as described in configuration file of Honeyd named honeyd.conf. The honeypot designed is proposed as a Low Interaction Honeypot which is integrated with Snort and MySQL.

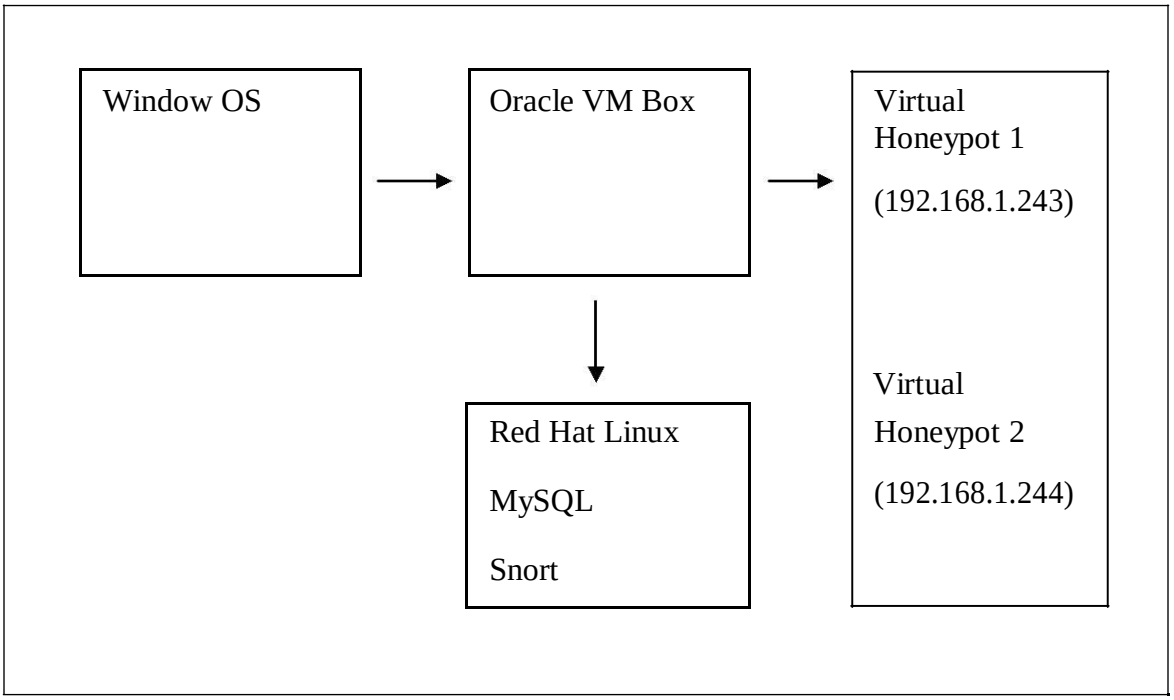

Fig 4.1 Experimental Setup

4.2 Oracle VM Virtual Box

Virtual Box is a cross-platform virtualization application. It can be installed on existing Intel or AMD-based computers, whether they are running Windows, Mac, Linux or Solaris operating systems. It also extends the capabilities of an existing computer so that it can run multiple operating systems (inside multiple virtual machines) at the same time. For example, you can run Windows and Linux on your Mac, run Windows Server 2008 on your Linux server, run Linux on your Windows PC, and so on, all alongside your existing applications. Virtual Box is deceptively simple yet also very powerful. It can run everywhere from small embedded systems or desktop class machines all the way up to data center deployments and even Cloud environments.

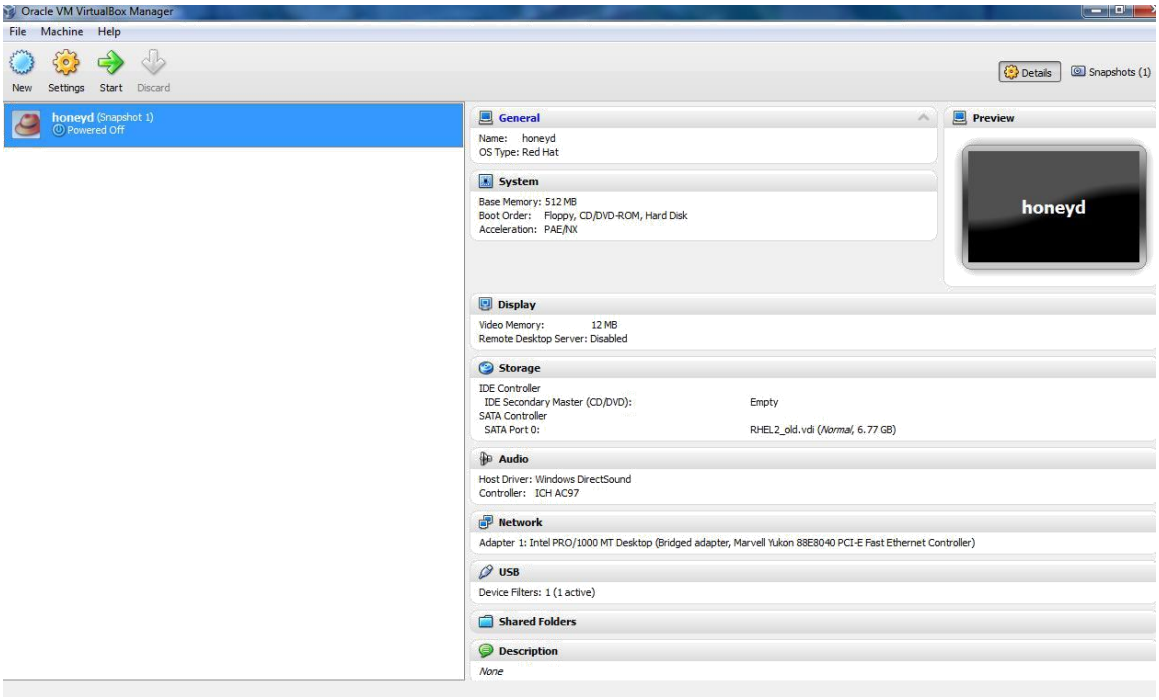

Fig 4.2 Virtualization Software

4.3 Steps to Implement Honeyd

- We start our work by writing personalities of virtual honeypots. We can write as many personalities in honeyd.conf file. We have written two different personalities in our configuration file.
- Then we write the command:

  arpd –d –i eth0 "IP addresses in configuration file of honeyd"

  Here we use the -d so that it does not run as a daemon or as a background process. Arpd daemon uses the free address space and allocate the Mac address of the machine with each IP address.
- After this honeyd is started by writing following command:

  honeyd -l /log/test –f honeyd.conf (IP addresses in configuration file of honeyd).

  -f is used so that honeyd runs in foreground. By this command honeyd is started as background process.
- Then the MySQL database is started. In new tab snort is started by the command:
  snort -i eth0 –vc /root/Desktop/honeyd/Latest_snort etc/snort.conf –l/var/log

  Honeydsum.pl is used to generate a report from the packets logged by the

honeyd. This is done by following command:

./honeydsum.pl –c honeydsum.conf /root/Desktop/honeyd/log/test>report

-c is used here to resolve domains.

- Snortalog.pl is used to generate a report (graphically) from the alerts generated by snort. This is done by the following command:

  perl snortalog.pl –file /var/log/alert –r N 30 – report

  Here N 30 specifies the number of lines in the result.

  For generating a graphical report in html format the following command is used

  perl Snortalog.pl –file /var/log/alert –r –i –g gif –o /root/ashish.html –report

  -r is used here to resolve IP address. -

  g is used here for graphical format. -

  file specifies input alert log file.

  -o specifies output directory.

4.4 Implementation Details

First install the required libraries.

a. Libevent

b. Libdnet

c. Libpcap: used libpcap 0.7 and its dev file.

d. Libpcre

A "Honeyd.conf" file is configured which contains virtual operating systems with emulated services. Each system is designed with the following commands.

- Create command create a new system.
- Add command used to add services, therefore binding scripts to a certain port.
- Set command assigns personality to a created system.

I. Running Arpd

The fake IP addresses are created with the help of Arpd daemon, which further bind these IP addresses with different templates specified in "honeyd.conf" configuration file. In this file The templates are created which are nothing but completely fake

script modules for the different operating systems. Some fake instances of different operating systems are created with fake IP addresses and then bind the templates accordingly. This assigning and binding is done as follows and shown in figure 4.3.

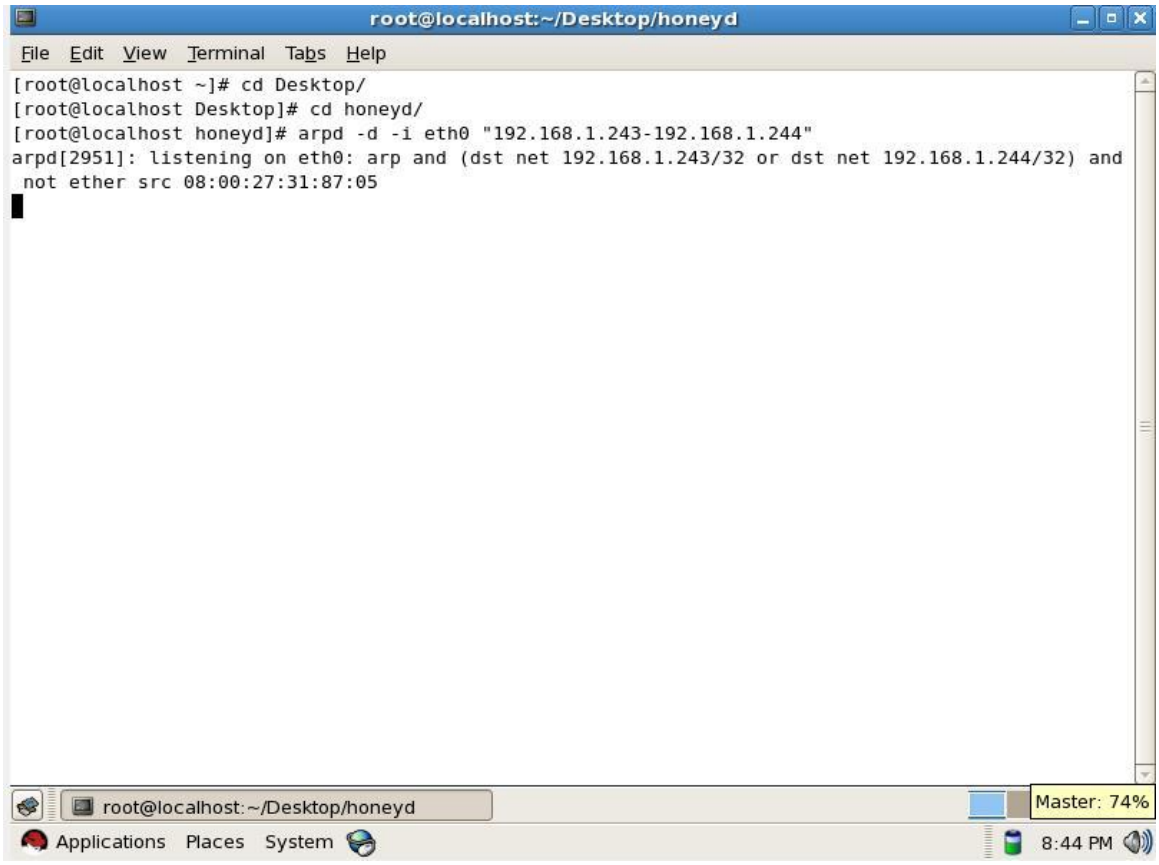

Fig 4.3 Binding IP addresses with Arpd

An Arpd ,also called farpd ,daemon can be installed which monitors the allocated IP address space and for any IP address that no host respond with MAC address , farpd will respond with the Honeyd physical machine MAC address.

II. Starting honeyd

Honeyd is a small daemon that runs both on UNIX-like and Windows platforms. It is used to create multiple virtual honeypots on a single machine. Entire networks can be simulated using honeyd. Honeyd can be configured to run a range of services like FTP, HTTP, or SMTP. Furthermore, a personality can be configured to simulate a certain operating system.

Honeyd Configuration File

```
create windows

set windows personality "Microsoft Windows XP Home Edition"

add windows tcp port 80 "perl scripts/iis-0.95/iisemul8.pl"

add windows tcp port 139 open

add windows tcp port 137 open

add windows udp port 137 open

add windows udp port 135 open

add windows tcp port 22 open

add windows tcp port 23 open

set windows default tcp action reset

set windows default udp action reset

bind 192.168.1.243 windows

create linux

set linux personality "Linux 2.2.12 - 2.2.19"

add linux tcp port 23 "sh scripts/telnet.sh"

add linux tcp port 22 open

set linux uptime 112211

set linux default tcp action reset

set linux default udp action

reset bind 192.168.1.244 linux
```

A honeyd configuration file named honeyd.conf is written above. We have created two personalities one is of window and the other is of Linux. It means two virtual honeypots are created here. IP address 192.168.1.243 is assigned to virtual honeypot1.Similarly IP address 192.168.1.244 is assigned to the virtual honeypot2.Then the packets are logged in a file named test. The command is

Honeyd –l/log/test –f honeyd.conf 192.168.1.243-192.168.1.244

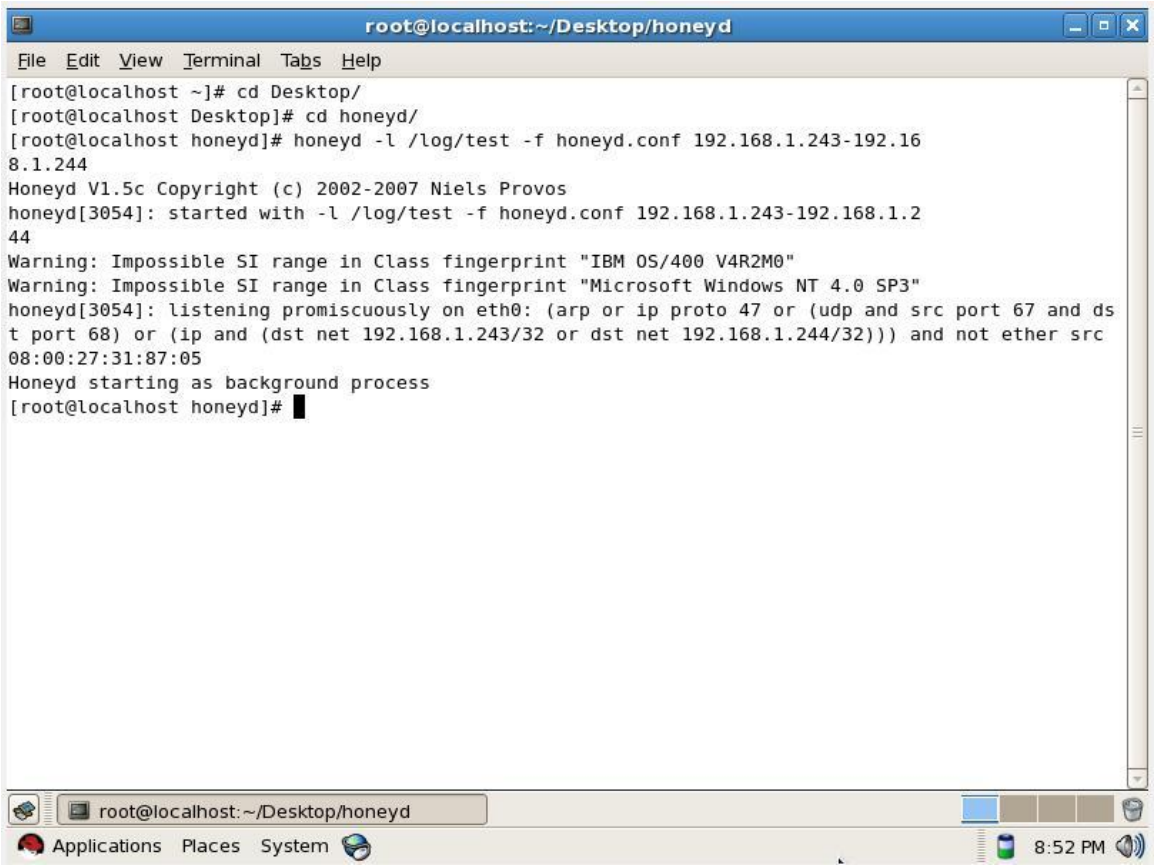

Figure 4.4 Starting Honeypot with honeyd

Then we see the content of test file located under the log folder. Honeyd log is started and the result is shown in figure 4.5

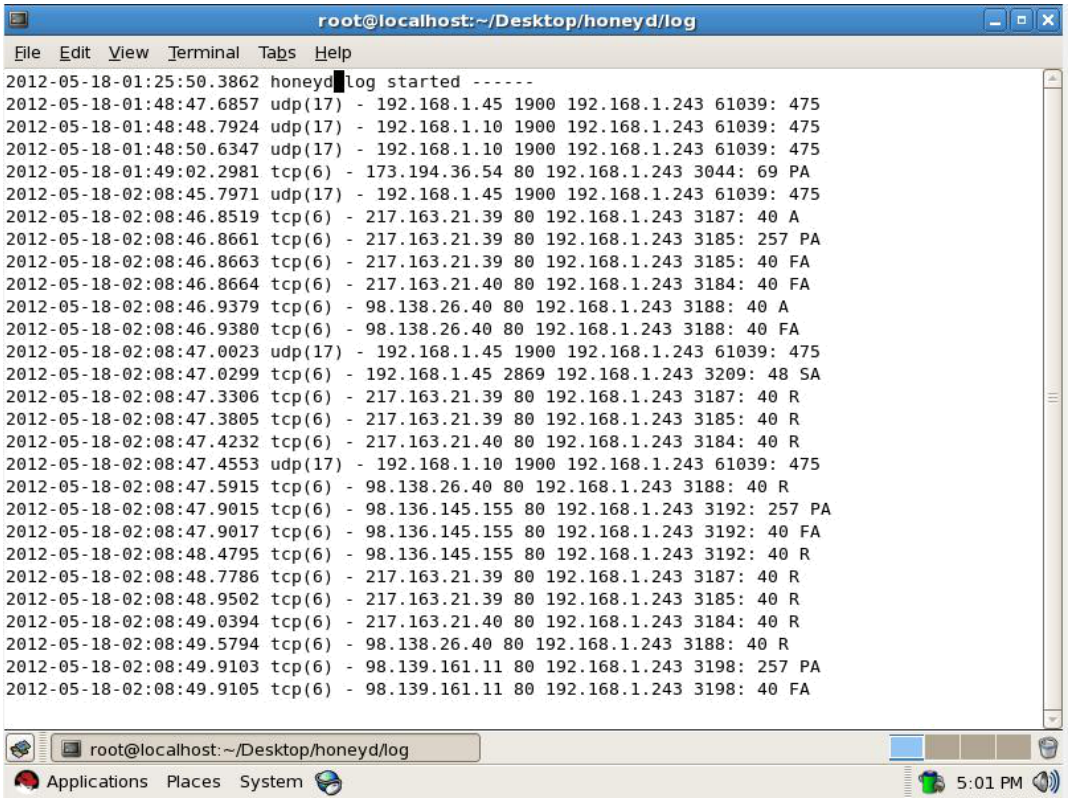

Figure 4.5 Packet Logged by honeyd

III. Starting MySQL services

MySQL is an Open Source Standard Query Language (SQL) database that is fast, reliable, easy to use, and suitable for applications of any size. MySQL can easily be integrated into Perl programs by using the Perl DBI (Database Independent interface) module. DBI is an Application Program Interface (API) that allows Perl to connect to and query a number of SQL databases.

MySQL is used here to store the alerts generated by Snort. The command used to start the MYSQL is

Service mysqld start;

MySQL is started and databases are shown in the figure 4.6.

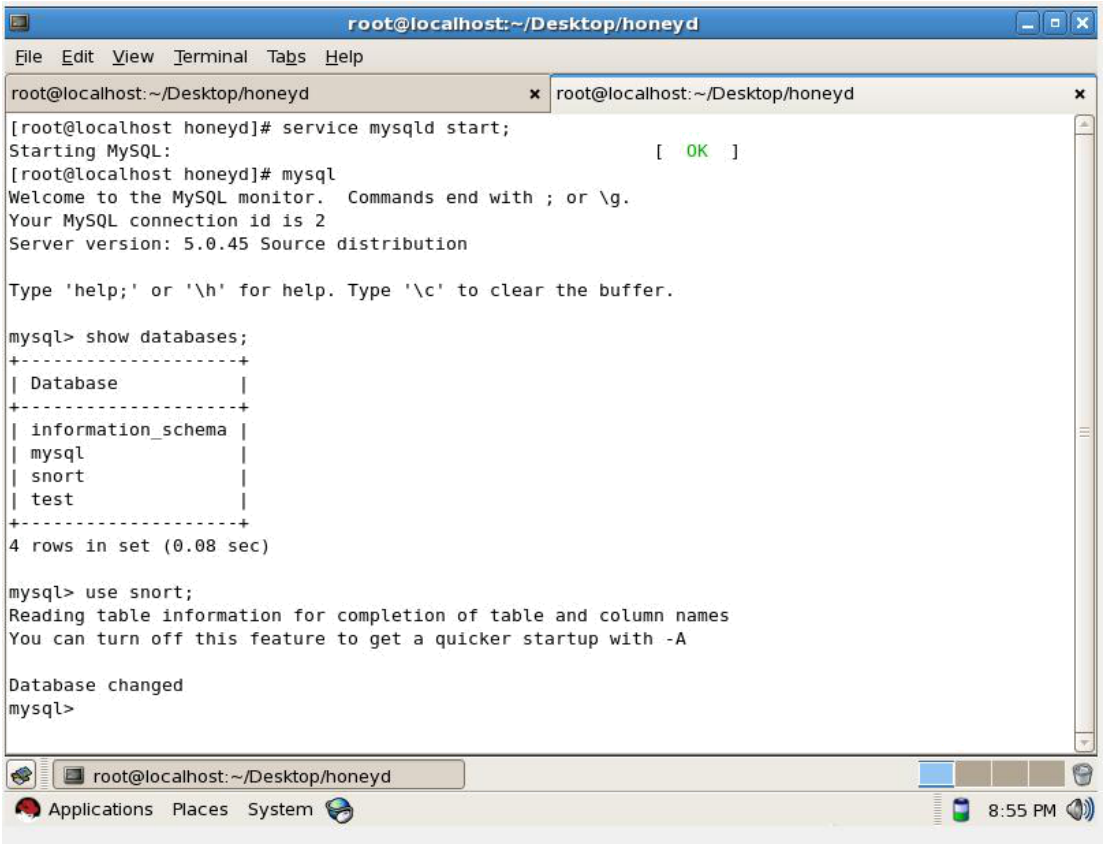

Figure 4.6 Starting MySQL services

IV. Starting Snort

Snort is a libpcap-based packet sniffer/logger which can be used as a lightweight network intrusion detection system. Snort has three primary uses: It can be used as a straight packet sniffer like tcp dump, a packet logger or as a full blown network Intrusion prevention system. Snort is used here to generate alerts and to capture more information about the attacks.

Command used to start snort is

Snort -i eth0 –vc etc/snort.conf –l/var/log.

After this command alerts generated by snort are stored in the file called log situated in var folder. This command is shown in the figure 4.7.

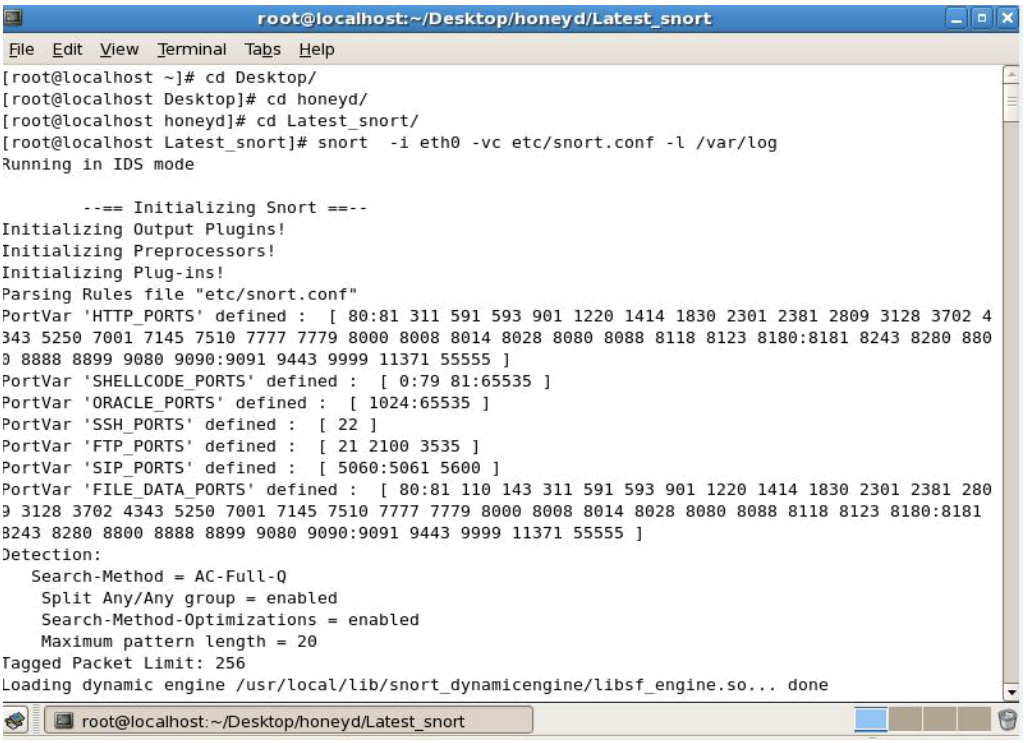

Figure 4.7 Initialization of Snort

After initializing the snort we use sql query to display the number of alerts generated by snort in descending order. The query is

Select s.sig_name,count(*)as count from event e,signature s where e.signature=s.sig_id group by e.signature order by count desc;

The result of this query is shown in the figure 4.8.

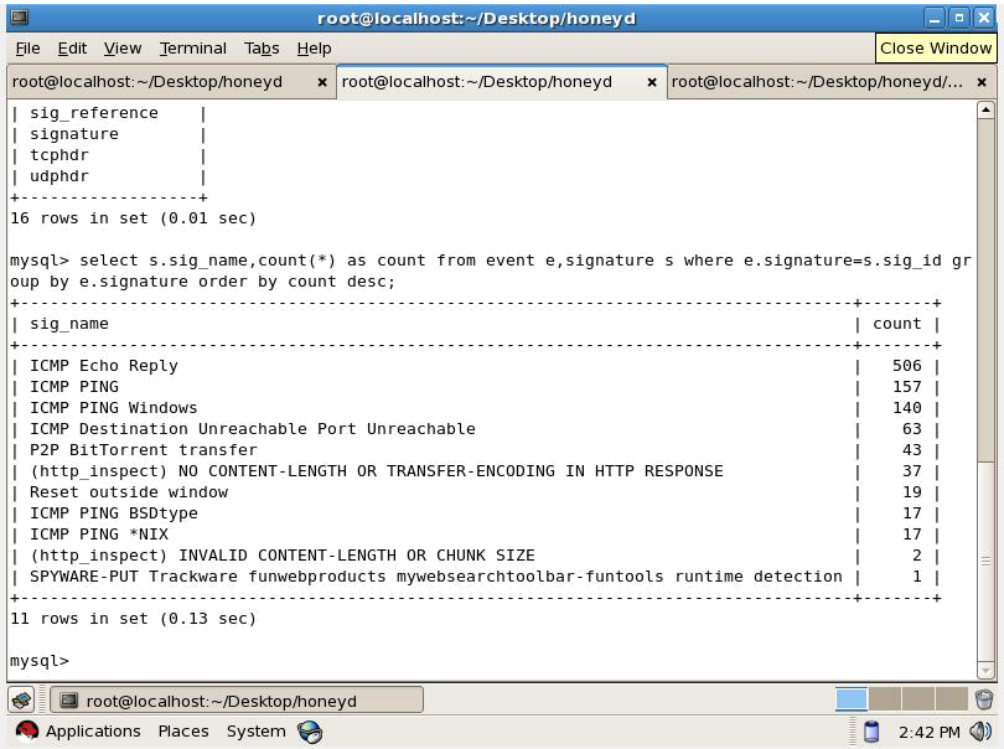

Figure 4.8 No. of alerts generated by Snort

V. Starting honeydsum

Honeydsum: It is a script written in Perl designed to generate a text summary from Honeyd logs. The summaries may be produced using different parameters as filters, such as ports, protocols, IP addresses or networks. It shows the top source and port access and the number of connections per hour, and supports input from multiple log files. The script can also correlate events from several honeypots.

The output of this script contains the information that provides several different summaries of all the traffic captured by the whole Honeyd environment. The command to generate a report from the honeydsum is:

./honeydsum.pl –c honeydsum.conf /root/Desktop/honeyd/log/test>report.

It takes the packet logged by honeyd as input and generates a report as shown in the figure 4.9.This report tells the total number of connections on the honeypot and the IP addresses of the sources which attacked on the virtual honeypots.

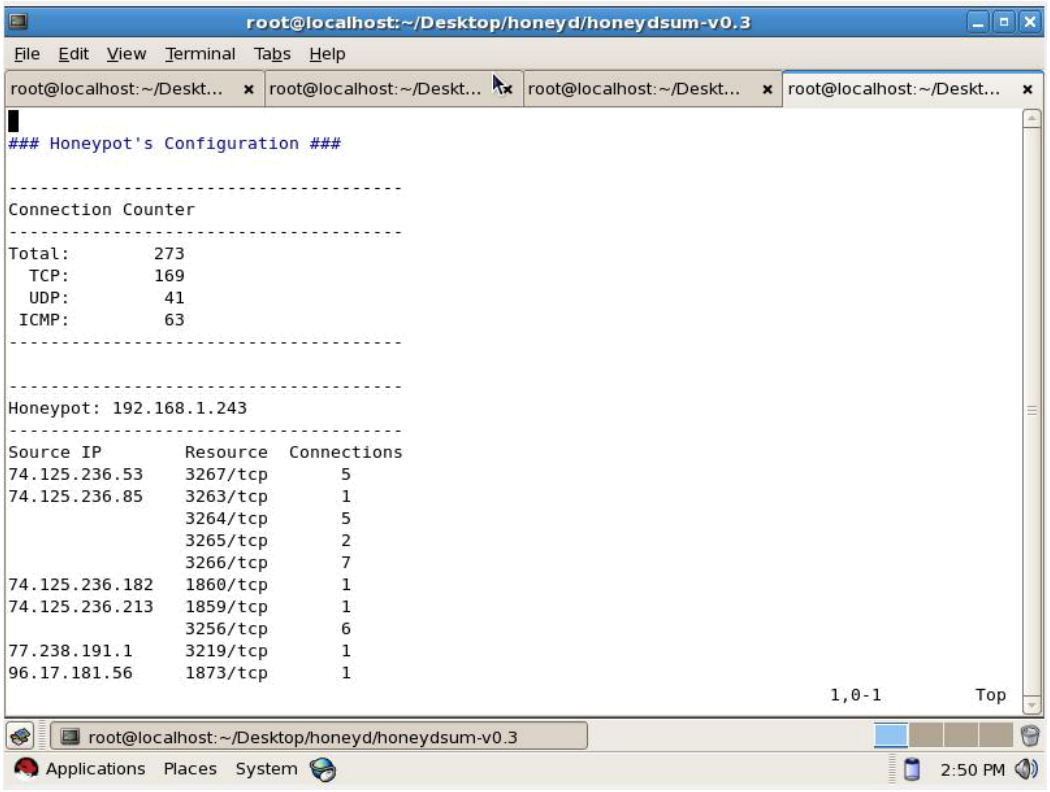

Figure 4.9 Total connections on Honeypots

Figure 4.10 shows the IP addresses of the sources which established the connections with the virtual honeypot1 (having IP address 192.168.1.243) and the total number of connections made by that source with the virtual honeypot1.

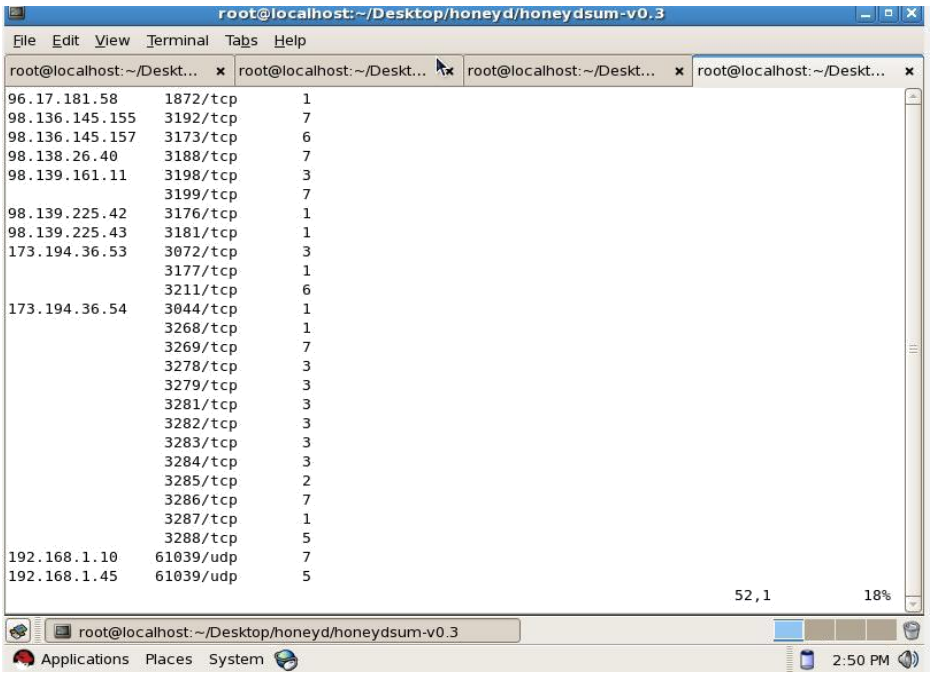

Figure 4.10 Packets established on virtual honeypot1

Figure 4.11 shows the IP addresses of the sources which established the connections with the virtual honeypot2 (having IP address 192.168.1.244) and the total number of connections made by that source with the virtual honeypot2

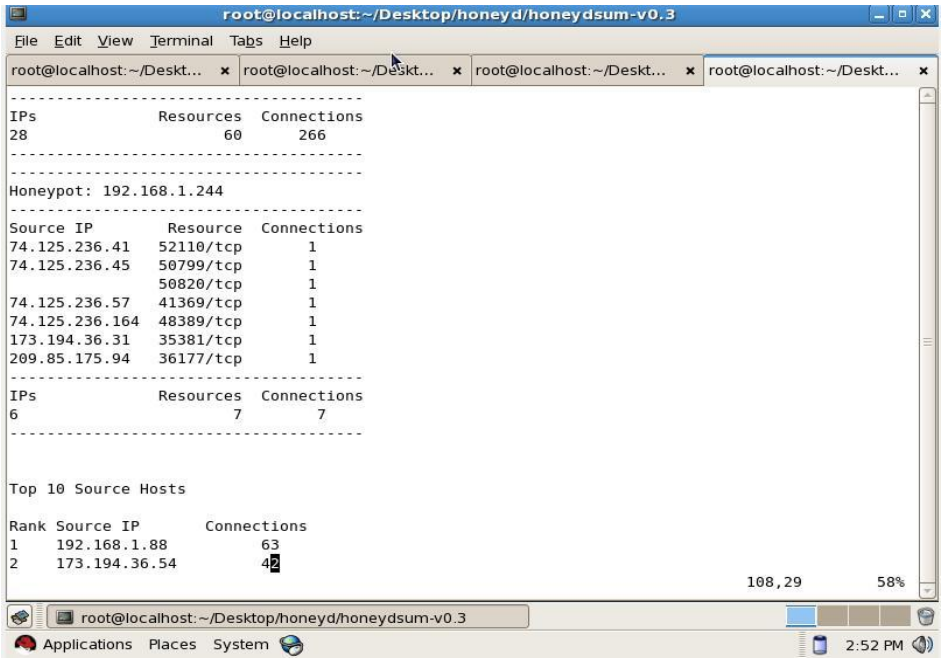

Figure 4.11 Packets established on virtual honeypot2

Figure 4.12 shows the Top 10 source hosts. The source IP which made the highest number of connections with the virtual honeypots is placed at the top.

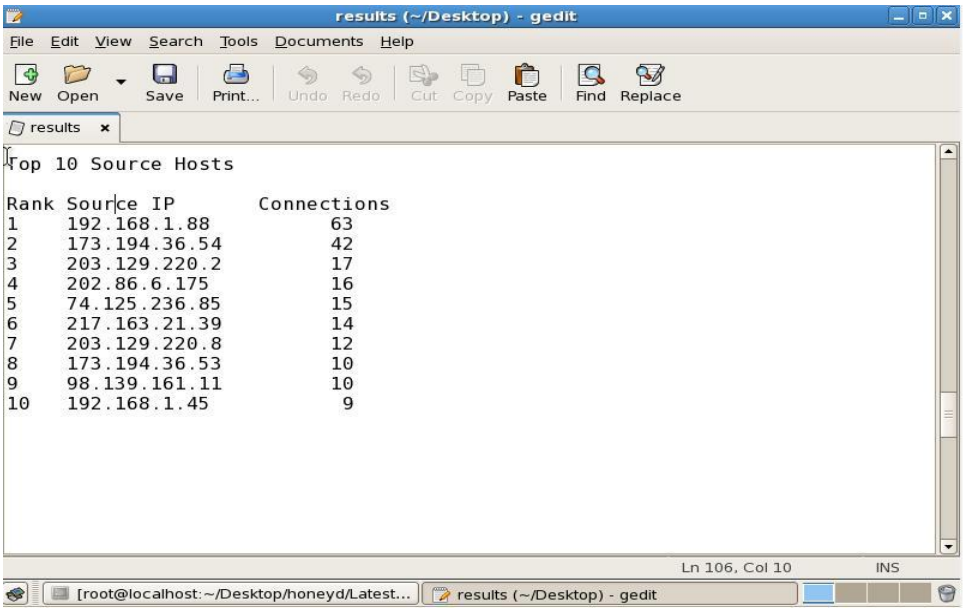

Figure 4.12 Top 10 source hosts

Figure 4.13 shows the connections per hour. This figure shows the connections established by the source on virtual honeypots in each hour during a day.

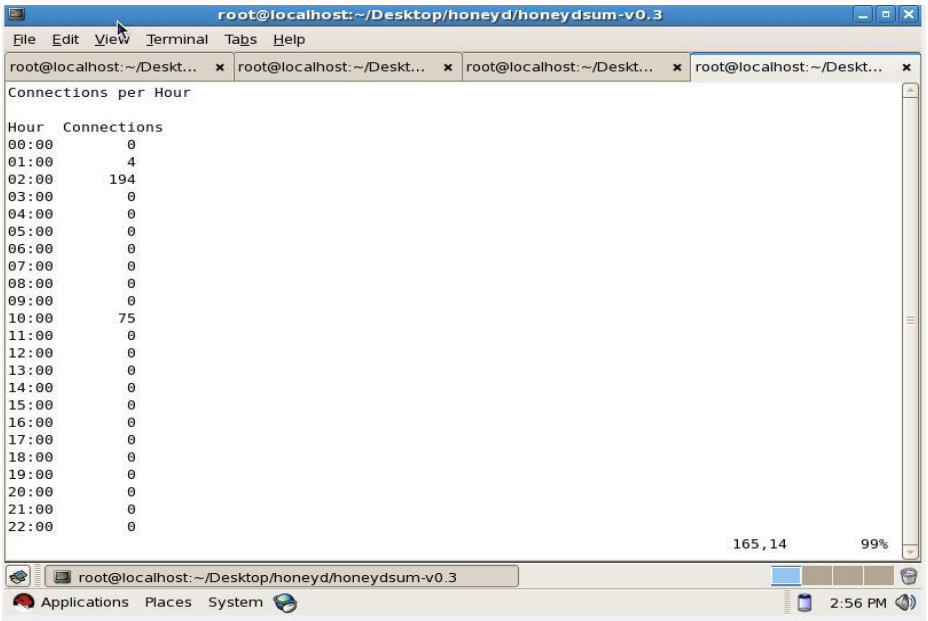

Figure 4.13 Connections per hour

VI. Starting Snortalog

Snortalog: It is a Perl script created by Jeremy Chartier that summarizes Snort logs, making it easy to view any network attacks detected by Snort. It can generate charts in HTML, PDF, and text output. It works with all versions of Snort, and can analyze logs in three formats: syslog, fast, and full snort alerts.

The command to start Snortalog is:

Snortalog.pl –file/var/log/alert –r N 30 – report.

This command takes the alert generated by snort as input and generate a report as shown in figure 4.14

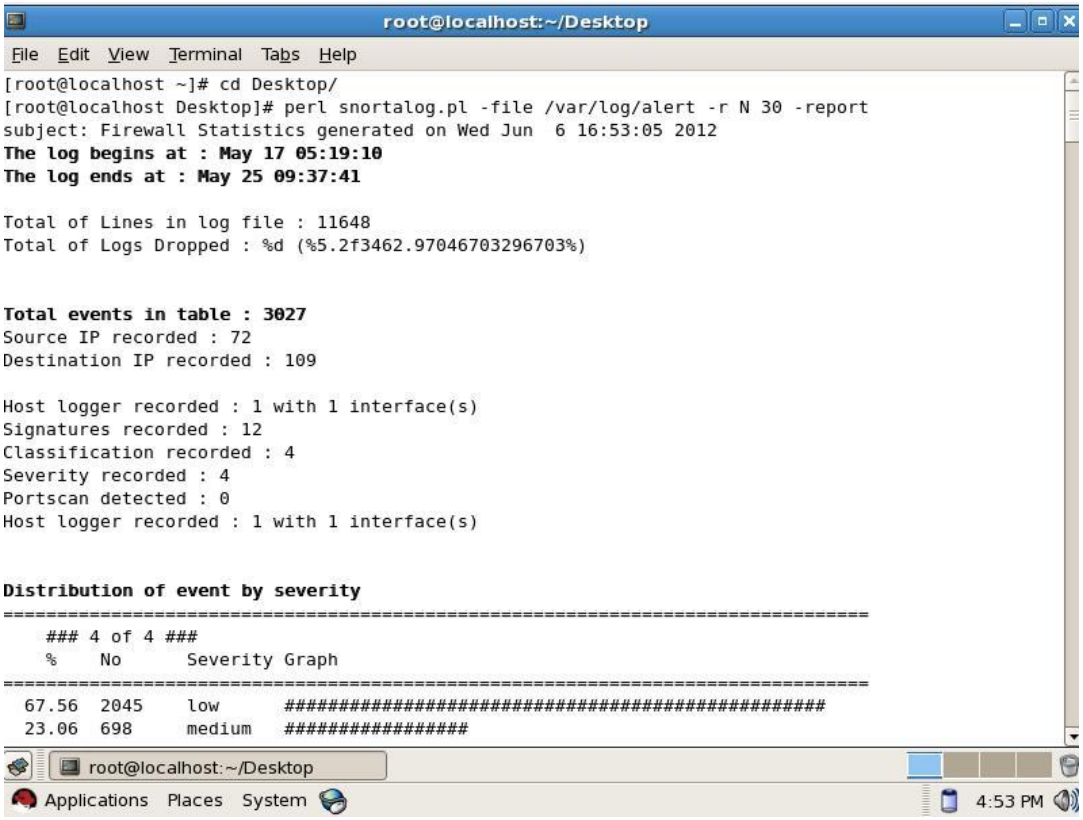

Figure 4.14 Starting snortalog.pl

This figure also shows the percentage of attacks according to their severity.

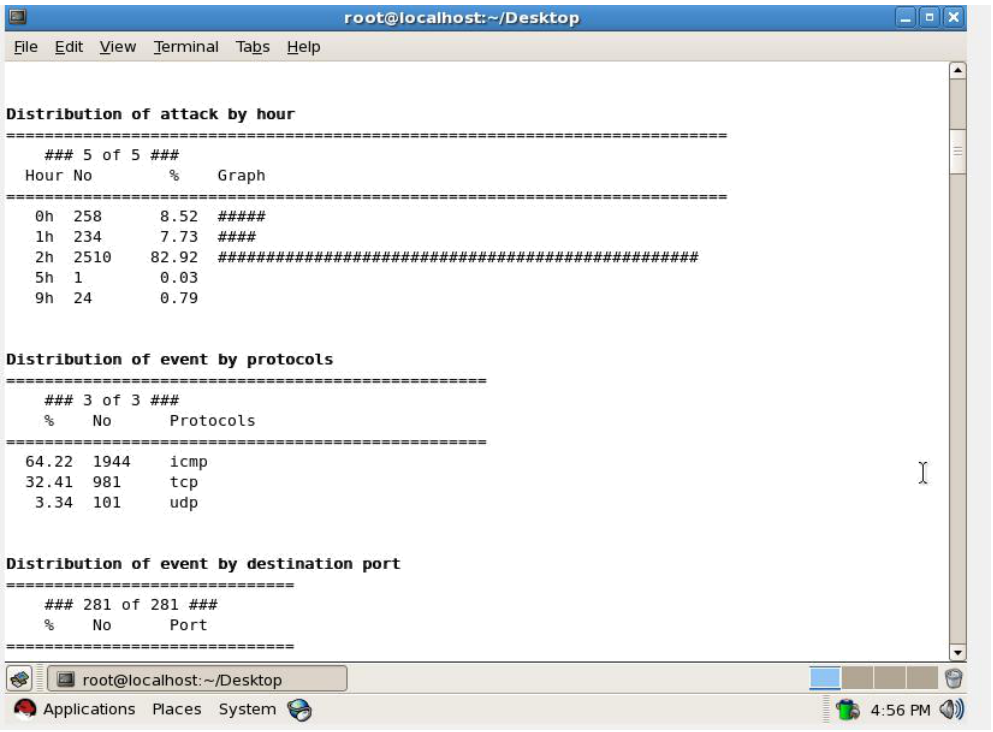

Figure 4.15 Distribution of events generated by Snort

Figure 4.15 shows the distribution of attacks by hour. It shows the number of attacks occurring in each hour during a day. It also shows the percentage and number of connections established by different protocols.

4.5Results

It is not easy and intuitive for everyone to query MySQL database in order to find some information about the honeypot's runs so to facilitate easy view and searching of data in the MySQL database.

After detecting all the attacks on virtual honeypots, our next task is to analyze those resultant attacks. This can be done by using "snort" tool. Snort is a tool that helps in easy understanding of attacks. Snort is a libpcap-based packet sniffer/logger which can be used as a lightweight network intrusion detection system. It can generate charts in HTML, PDF, and text output. It works with all versions of Snort, and can analyze logs in three formats: syslog, fast, and full snort alerts.

4.5.1 Traffic Report Generated by Snortalog

Traffic report summarizes all the traffic detail. It shows the count of TCP, UDP and other packets. Other packets include ICMP packets.

%	No	Severity
67.56	2045	Low
23.06	698	medium
9.35	283	High
0.03	1	unknown

Table 4.1 Distribution of event by severity

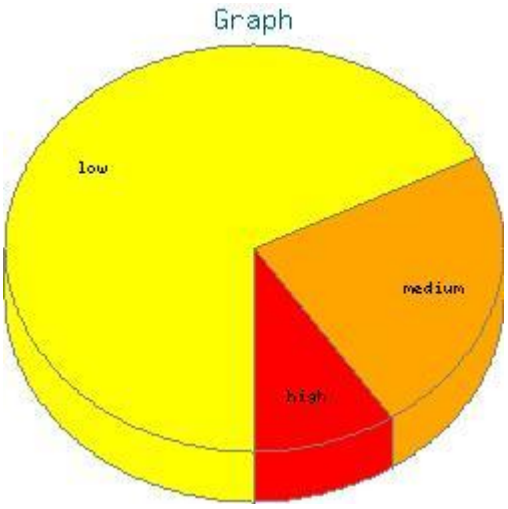

Figure 4.16 Distribution of event by severity

Hour	No	%	Graph
0h	258	8.52	
1h	234	7.73	
2h	2510	82.92	
5h	1	0.03	
9h	24	0.79	

Table 4.2 Distribution of attack by hour

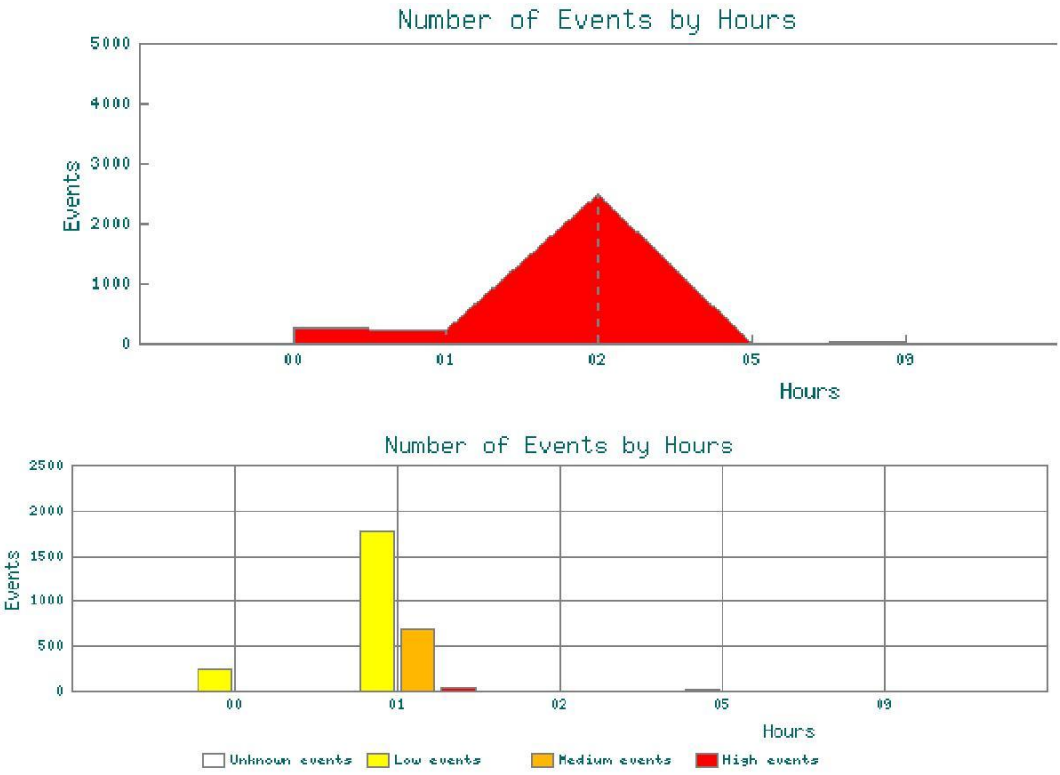

Figure 4.17 Distribution of attack by hour

	No	rotocols
64.22	1944	icmp
32.41	981	Tcp
3.34	101	Udp

Table 4.3 Distribution of event by protocols

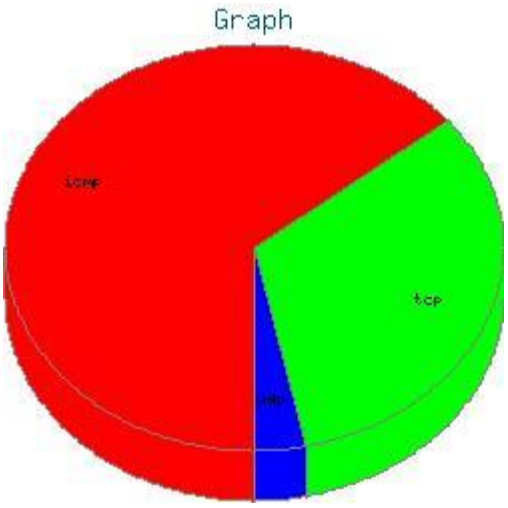

Figure 4.18 Distribution of event by protocols

%	No	Destination Port
0.03	1	28336
0.03	1	1674
0.03	1	1648
0.03	1	1346
0.03	1	1488
0.03	1	1360
0.03	1	1490
0.03	1	1273
0.03	1	1364
0.03	1	1312

0.03	1	1257
0.03	1	1370
0.03	1	1263
0.03	1	45020
0.03	1	1630
0.03	1	1550
0.03	1	1597
0.03	1	1421
0.03	1	1296
0.03	1	1638
0.03	1	1476
0.03	1	1207
0.03	1	1654
0.03	1	1233
0.03	1	1306
0.03	1	1572
0.03	1	1328
0.03	1	1546
0.03	1	1318
0.03	1	1506
0.03	1	1589
0.03	1	1609
0.03	1	1660
0.03	1	1322
0.03	1	1670
0.03	1	1603
0.03	1	1518
0.03	1	1259
0.03	1	1298
0.03	1	1459
0.03	1	1658
0.03	1	1570
0.03	1	1269
0.03	1	1308
0.03	1	1558
0.03	1	1429

0.03	1	1286
0.03	1	38270
0.03	1	1237
0.03	1	1467
0.03	1	1463
0.03	1	1502
0.03	1	1374
0.03	1	1203
0.03	1	1528
0.03	1	1425
0.03	1	1692
0.03	1	1678
0.03	1	1445
0.03	1	1354
0.03	1	1378
0.03	1	1427
0.03	1	1484
0.03	1	50108
0.03	1	1634
0.03	1	1389
0.03	1	1332
0.03	1	1241
0.03	1	1585
0.03	1	1441
0.03	1	1399
0.03	1	1433
0.03	1	1666
0.03	1	1498
0.03	1	1282
0.03	1	1694
0.03	1	1607
0.03	1	1267
0.03	1	49722
0.03	1	1386
0.03	1	1401
0.03	1	1680

0.03	1	1688
0.03	1	1536
0.03	1	1292
0.03	1	1350
0.03	1	1356
0.03	1	1419
0.03	1	1277
0.03	1	1618
0.03	1	1624
0.03	1	1215
0.03	1	1219
0.03	1	1336
0.03	1	1552
0.03	1	1650
0.03	1	13799
0.03	1	1560
0.03	1	1473
0.03	1	1628
0.03	1	1554
0.03	1	1451
0.03	1	1453
0.03	1	1530
0.03	1	1636
0.03	1	1534
0.03	1	1522
0.03	1	43410
0.03	1	1580
0.03	1	25677
0.03	1	1279
0.03	1	1504
0.03	1	1439
0.03	1	1395
0.03	1	1415
0.03	1	1243
0.03	1	1384
0.03	1	1471

0.03	1	1253
0.03	1	44301
0.03	1	1265
0.03	1	1407
0.03	1	1512
0.03	1	1644
0.03	1	1526
0.03	1	1494
0.03	1	1568
0.03	1	1247
0.03	1	1223
0.03	1	1403
0.03	1	1672
0.03	1	1316
0.03	1	1576
0.03	1	1304
0.03	1	1342
0.03	1	1622
0.03	1	1478
0.03	1	1368
0.03	1	1211
0.03	1	1449
0.03	1	1593
0.03	1	1542
0.03	1	1642
0.03	1	1591
0.03	1	1314
0.03	1	1376
0.03	1	1320
0.03	1	1457
0.03	1	1261
0.03	1	1352
0.03	1	1632
0.03	1	1423
0.03	1	1393
0.03	1	1235

0.03	1	1595
0.03	1	1443
0.03	1	1469
0.03	1	1646
0.03	1	1564
0.03	1	1310
0.03	1	1372
0.03	1	1239
0.03	1	1324
0.03	1	1380
0.03	1	1614
0.03	1	1556
0.03	1	1213
0.03	1	1500
0.03	1	1245
0.03	1	1201
0.03	1	1199
0.03	1	1300
0.03	1	1340
0.03	1	1676
0.03	1	1652
0.03	1	1516
0.03	1	1482
0.03	1	1231
0.03	1	1249
0.03	1	1601
0.03	1	1656
0.03	1	1664
0.03	1	1366
0.03	1	1255
0.03	1	1684
0.03	1	1690
0.03	1	1271
0.03	1	1582
0.03	1	1668
0.03	1	1348

0.03	1	1599
0.03	1	1435
0.03	1	1548
0.03	1	1227
0.03	1	37419
0.03	1	1417
0.03	1	1288
0.03	1	1662
0.03	1	1587
0.03	1	1540
0.03	1	1330
0.03	1	1284
0.03	1	1532
0.03	1	1221
0.03	1	1496
0.03	1	30639
0.03	1	1229
0.03	1	1682
0.03	1	1413
0.03	1	1334
0.03	1	1251
0.03	1	1431
0.03	1	1275
0.03	1	11/0
0.03	1	1338
0.03	1	1620
0.03	1	1362
0.03	1	1209
0.03	1	1447
0.03	1	1290
0.03	1	1510
0.03	1	1225
0.03	1	1605
0.03	1	41427
0.03	1	1538
0.03	1	1544

0.03	1	1616
0.03	1	1409
0.03	1	1626
0.03	1	1640
0.03	1	1574
0.03	1	1524
0.03	1	1508
0.03	1	1611
0.03	1	1326
0.03	1	1205
0.03	1	1578
0.03	1	1382
0.03	1	1566
0.03	1	31192
0.03	1	65085
0.03	1	1520
0.03	1	33546
0.03	1	1455
0.03	1	1405
0.03	1	1465
0.03	1	1302
0.03	1	1492
0.03	1	1397
0.03	1	1294
0.03	1	1437
0.03	1	1344
0.03	1	1562
0.03	1	1358
0.03	1	1514
0.03	1	1461
0.03	1	1486
0.03	1	1686
0.03	1	1411
0.03	1	1217
0.03	1	1480
0.03	1	29966

0.07	2	35139
0.13	4	43985
0.13	4	49523
0.23	7	53847
0.33	10	12277
0.43	13	63074
0.46	14	8080
0.83	25	63576
1.06	32	22311
1.45	44	25953
2.44	74	37159
2.68	81	29744
2.78	84	20036
3.24	98	24824
3.30	100	3/3
4.39	133	1696
8.39	254	0/0
20.38	617	8/0
38.39	1162	80

Table 4.4 Distribution of event by destination port

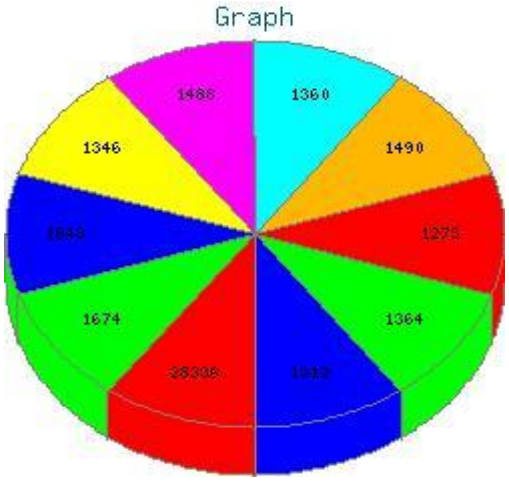

Figure 4.19 Distribution of event by destination port

Chapter 5 CONCLUSION AND FUTURE WORK

5.1 Conclusion

As per the research done so far and literature study, Honeypots are a current research field in the sector of network security. Currently there is a lot of ongoing research and discussions all around the world. Honeypots are resources which give full freedom to attackers and every activities of the attacker will get logged. Honeypots gives us the platform to gather the intelligent information about the attackers so that we can later study them and can take the remedial actions to tighten the network security. Honeypots are being classified into different categories such low interaction honeypots, high interaction honeypots etc. Low interaction honeypots provide the emulated environment to the attackers which is not a real environment therefore the volume of attacks may be less in case of low interaction honeypots whereas in case of high interaction honeypots which provide the real environment to the attackers to get logged. A honeypot is a valuable resource, especially to collect information about proceedings of attackers as well as their deployed tools. No other mechanism is comparable in the efficiency of a honeypot if gathering information is a primary goal, especially if the tools an attacker uses are of interest. But nevertheless, honeypots cannot be considered as a standard product with a fixed place in every security aware environment as firewalls or intrusion detection systems are today. The involved risk and need for tight supervision as well as time intensive analysis makes them difficult to use. Honeypots are in one's infancy and new ideas and technologies will surface in the next time. At the same time as honeypots are getting more advanced, hackers will also develop methods to detect such systems. A regular arms race could start between the good guys and the blackhat community.

5.2 Future Scope

Our developed Virtual Honeypot could certainly be improved by offering more and more varieties of honeypots than the ones which are currently available. In addition a GUI Based frontend is currently missing and is desperately needed. Also if the developed solution is being extended to distributed kind of environment which will definitely lead to get the good class of attack vectors.

Extending to a Distributed Honeynet Solution: The honeypot implemented in this thesis could be extended rather easily (now that the basic concepts are understood) to a full-fledged honeynet or Distributed Honeynet System which can be deployed geographically at various heterogeneous locations. This would enable us to detect exploited systems on the cyber space.

REFERENCES

[1] Gassman, B.,"Internet security, and firewalls protection on the internet", Somerset, NJ, USA, ELECTRO '96. Professional Program Proceedings, 30 April-2 May, 1996.

[2]"What Is the Difference: Viruses, Worms, and Trojans?", http://www.cisco.com/web/about/security/intelligence/virus-worm-diffs.html.

[3] "SSL Is Not Secure Anymore", http://swiki.fromdev.com/2009/11/ssl-is-not-secure-anymore-serious.html.

[4] "Understanding Denial-of-Service Attacks", http://www.us-cert.gov/cas/tips/ST04-015.html.

[5] "Network Based Attacks", http://secret-epedemiology-statistic.org.ua/1587052091/ch01lev1sec3.html

[6] "Firewalls", http://iac.dtic.mil/iatac/download/firewalls.pdf.

[7] "Application Firewalls", https://www.cs.columbia.edu/~smb/classes/f06/l16.pdf.

[8] Innella , P. and McMillan, O. "An Introduction to Intrusion Detection Systems" , http://www.securityfocus.com/infocus/1520.

[9] Provos, N., "A Virtual Honeypot Framework", SSYM'04 Proceedings of the 13th conference on USENIX Security Symposium, Volume 13, 2004.

[10] Lance Spitzner, Tracking Hackers. Addison Wesley, September 2002.

[11] Zanoramy, W. and Zakaria, A., "Deploying Virtual Honeypots on Virtual Machine Monitor" IEEE Proceedings, vol. 4, pp.1-5, 26 Aug 2008.

[12] Spitzner, L., "Honeypot: Definitions and Values", May 2002 http://www.spitzner.net.

[13] Levin, J. and Labella, R., "The Use of Honeynets to Detect Exploited Systems across Large Enterprise Networks", IEEE Proceedings, pp.92-99, 18 June 2003.

[14] Qassrawi, M. and Hongli, Z. "Deception methodology in virtual Honeypots", Second International Conference on Network Security, Wireless Communication and Trusted Computing, 2010.

[15] Bao, J. and Gao, M. "Research on network security of defense based on Honeypot", International Conference on Computer Applications and System Modeling, 2010.

[16] Levine, J. and Grizzard, J. "Using honeynets to protect large enterprise networks", Security & Privacy Magazine, IEEE, vol. 2, pp. 73-75, 2004.

[17] Ryan Talabis," Honeypots 101: What's in it for me?" http://www.philippinehoneynet.org/, Fetched 21/06/2011.

[18] Tang, X. "The Generation of Attack Signatures Based on Virtual Honeypots", International Conference on Parallel and Distributed Computing, Applications and Technologies, 2010, pp.435-439.

[19] "Snort", http://www.snort.org/assets/166/snort_manual.pdf.

[**20]** John E. Canavan," Fundamentals of Network Security", http://www.artechhouse.com.

[21] Provos, N. "A Virtual Honeypot Framework", SSYM'04 Proceedings of the 13th conference on USENIX Security Symposium, Vol. 13, 2004.

[22] Lance Spitzner, "Definitions and Value of Honeypots", http://www.trackinghackers.com/papers/honeypots.html.

[23] Peter, E. et al., "A Practical Guide to Honeypots", http://www.cse.wustl.edu/~jain/cse571-09/ftp/honey/index.html, Fetched 20/06/2011.

[24] Spitzner, L. "Know Your Enemy: Honeynets", http://www.honeynet.org/papers/honeynet.