A Proactive Framework for Capturing FTP Brute Force and Application Level Flood Attacks

Thesis

submitted in partial fulfillment of the requirements

for the award of degree of

Master of Technology
In
Computer Science and Applications

Submitted By
Name: Aastha Bhandari
Roll No. : 601103001

Under the supervision of
Mrs. Sanmeet Kaur
Assistant professor

SCHOOL OF MATHEMATICS AND COMPUTER APPLICATIONS
THAPAR UNIVERSITY
PATIALA – 147004

JUNE 2012

CERTIFICATE

I hereby certify that the matter which is being presented in the thesis titled, "***A Proactive Frame work for Capturing FTP Brute Force and Application level DOS Attacks*** ", in partial fulfillment of the requirements for the award of degree of M. Tech. (Computer Science and Applications) submitted in School of Mathematics and Computer Applications, Thapar University, Patiala, is a survey carried out by me under the supervision of *Mrs. Sanmeet Kaur* and refers other researcher's works which are duly listed in the reference section.

Aastha Bhandari

Name: **Aastha Bhandari**
Roll No.:601103001

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
Thapar University
Patiala

Countersigned by

(Dr. Rajesh Kumar)

Head
SMCA
Thapar University
Patiala

(Dr. S.K. Mohapatra)

Dean (Academic Affairs)
Thapar University
Patiala

ACKNOWLEDGEMENT

First and foremost I thank Almighty for giving me this meaningful life to serve human beings with great intensity and quest to learn more and more.

Acknowledging individuals who have helped me directly, indirectly, tangibly and/or intangibly to this research effort reflects honestly my deep regards to their contributions. This note of acknowledgement is my sincere appreciation to those people who stand out most notably in my mind as contributing to the effort.

My revered supervisor that I must acknowledge due to her importance and inspiration to my work. She urged me on and on by way of her untiring support and seemingly unlimited belief in me. My profound regards to Mrs. Sanmeet Kaur.

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

My gratitude to my department, particularly my teachers and to all of my wonderful colleagues, for their stimulating discussion and invaluable support I have received during the period of this research.

There are many that from behind the scenes have encouraged and supported my work, and I wish to thank them. My deep regards to my family for their immense love and support, and my friends for all their enthusiastic encouragement throughout my life without which i could not complete this work.

At last but not the least, support provided by all the concerned people is highly appreciated and deeply acknowledged.

Aastha Bhandari

ABSTRACT

Security, over the last decade, has become a huge priority for the network administrators. They dedicate lots of time to make sure that each network has the best and latest security patches, firewalls, and intrusion detection systems. Unfortunately, the security patches such as firewalls and intrusion detection systems are not as effective as they used to be due to the generation of large log files. Both the above mentioned techniques have information overload problems which can be solved using Honeypot. Honeypot interact with the attacker and collect the data that is thereafter analyzed. Honeypot are resources which are targeted by attackers and upon attack it logs data about the attacks. FTP server can be emulated on the honeypot and various scans can be done on the emulated service. The IP addresses trying to perform some malicious activity on ftp service can be caught and their data can be logged in the log file. Thus honeypot is a very good tool to find the vulnerabilities in the security system which are used by intruders.

This thesis describes the various FTP attacks that are studied and prevented. Honeypot is used to emulate an ftp server which gives the attacker an illusion of existing ftp server. brute force attack is the most common attack which is implemented using metasploit. The brute force attack on ftp server can be prevented by limiting the number of attempts that the same IP address can try to logon to the server. On the other hand, if an attacker tries to flood the server with packets, the administrator can be informed about this activity. In brute force attack, Snort which is an IDS can be called when the user exceeds the threshold value of failed logon attempts. Snort in turn result in barring the further logon attempts from the same ip address. The stepwise approach to security is best suited integrated solution which is adopted in the thesis to learn attacks and thus coming up with remedies for the same at the earliest.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1 Network

A Computer network is a group of computers and associated devices connected to each other in order to provide communication, information sharing and resources sharing. The computer can be located in a small area, in a city or all over the world. They can be connected through metallic cables, optical cables or satellite links depending on the area of their location. Internet is a large network spread all over world connecting many smaller networks. It provides so many facilities like direct communication file transfer, e-mail etc. A message can be acknowledged, forwarded, stored, retrieved or add attachments to it. Similarly networks allow to share resource like applications, printers, modems, scanners, disk space etc. Though it provides so many services but now days the main attraction on it is electronic meetings, video conferencing etc that is made possible through special software called groupware. Groupware are special software that allow multiple users to communicate directly to each other with inclusion of audiovisual and other types of data. It helps business organizations to arrange important business meetings in a very short time while the people attending them are sitting at geographical distances. Thus networks are proved to be boons for public, private as well as government organizations. Not only this, it is helpful to individuals as it enhances learning and knowledge.

1.2 Network Security

The word 'secure' is made up of two words, se- means without and cure-means take care of; thus secure means something without care[1]. Network security is a process that keeps an unauthorized parties away from gaining access on the network. Network security is of two types, first is security of data information, i.e. protect the information from unauthorized access and loss. And the second is computer security, i.e. protect the data and to thwart hackers. on internet or any other network of an organization, lots of information is exchanged daily which can be misused by attackers. Computer security is

a conspicuous part of network ensuring confidentiality, integrity and availability. Network security ensures protection against various internal and external threats like viruses, worms and trojans, denial of service network security attacks, etc. Security is important to keep the hackers away from viewing confidential information. Security policies describe what must be secured to support the business or mission[2].

1.3 Need of Security

Security in the enterprise has lately become the primary concern of both IT managers and other executives. The challenges of securing enterprise networks in the face of intruders armed with the tools of compromise have become overwhelming and are still growing. The computer security Institute/FBI 2003 Computer Crime and Security survey indicate that the total annual losses reported in the 2003 were $201,797,340[3].

The information security is needed for the reasons that are discussed below.

- To protect the sensitive information from unauthorized access.
- To protect the information from unwanted editing, accidently or intentionally by unauthorized users.
- To protect the information from loss and make it to be delivered to its destination properly.

The need of computer security is needed for the following reasons.

- To protect from replicating and capturing viruses from infected files.
- Computer needs a proper protection from worms.
- There is a need of protection from trojan horses as it can damage the computer.

As technology is progressing the computers are getting more complex which leads to more complex ways to attack them, i.e. Attack vectors are becoming more complex. A graph is shown in Figure 1.1 on the next page which tells it is becoming easier for attackers to attack the systems.

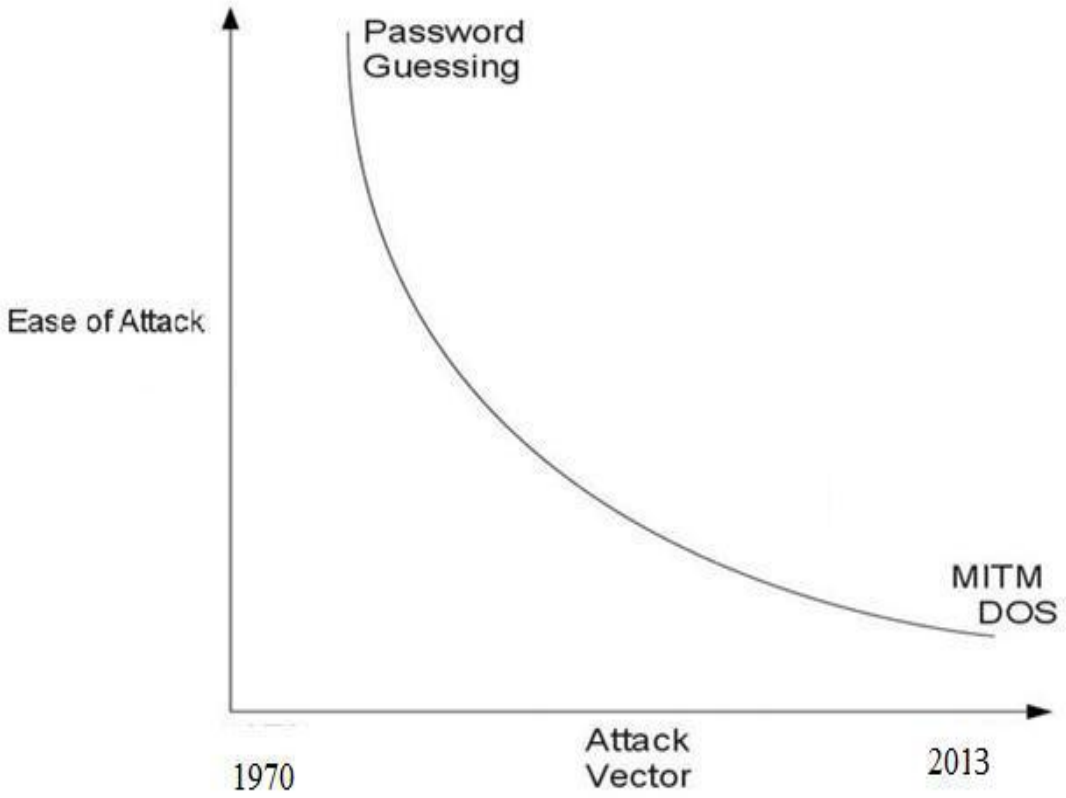

Fig 1.1 Ease of Attack vs. Attack Vector with Time[1]

The above graph shows that the computers are getting more complex with time, the attack vectors are becoming complex at the same pace. Earlier password guessing was mainly used to break the security but now a day effective and intelligent attacks like Man-In-The-Middle, Denial of Service attacks are used to break into the system. As the security is tightening so as the attackers are getting smarter to find out the ways to break the security and hack the system.

1.4 Hacking Terminology

Some terms that are used in IT world related to hacking are given below.

- **Hacker**

  The noun 'Hacker' refers to a person who enjoys learning the details of computer system and stretch their capabilities.

- **Hacking**

  The verb 'Hacking' describes the rapid development of new programs or the reverse engineering of already existing software to make the code better and efficient.
- **Cracker**

  The term 'Cracker' refers to a person who uses her hacking skills for offensive purposes.
- **Ethical Hacker**

  The term Ethical hacker refers to security professional who apply their skills for good.
- **Threat**

  Threat is an action or even that might prejudice security. A threat is a potential violation of security.
- **Vulnerability**

  Vulnerability is the existence of weakness in design or implementation error that can lead to an unexpected, undesirable event compromising the security of the system.
- **Attack**

  Attack is an assault of system security that derives from an intelligent threat. An attack is any action that violates security.
- **Exploit**

  Exploit is a defined way to breach the security of systems through vulnerability.

1.5 Types of Security

There are basically two types of security approaches used to deal with the vulnerabilities, namely proactive and reactive.

1.5.1 Proactive Security

Prevention is invariably a better approach than cure for both living beings and computer networks. Proactive security provides a method for maintaining the overall security of a system, even when individual components are repeatedly broken into and controlled by

an attacker[4]. Proactive security includes all measures that are taken with the goal of preventing attacks from successfully compromising systems. Security engineers have to establish various mechanisms to prevent the attacker from breaking in. Organizations use intrusion detection and response system to try to detect computer intrusions and then activate measures when an attack is detected.

1.5.2 Reactive Security

Reactive security is the approach in which includes the procedures that the organizations use once they discover that some of their systems have been compromised by an intruder or attack program. Reactive methods include disaster recovery plans, use reinstallation of operating systems and applications on compromised systems, or switching to alternate systems in other locations. For example, Intrusion prevention systems.

1.6 Attack Life Cycle

There are different phases through which an attacker passes to gain access to and attack. Various phases of attack life cycle are discussed below.

1.6.1 Reconnaissance

Reconnaissance is the first phase of attack life cycle in which an attacker tries to gather maximum information about the target. Two ways can be used to gather information: First is passive where an attacker can listen to the network traffic by using a sniffing software and the second is active in which an attacker can gather the information by probing the machine/network. The intention of the attacker is to know about the operating system running on the target and the various ports opened so as to tailor made an attack.

1.6.2 Scanning

In the scanning phase, an attacker starts the process in which he/she scans the various perimeter and internal network devices looking for weaknesses. The various perimeters which can be scanned to get the intended information are

- Open ports

- Open services
- Vulnerable applications, including operating systems
- Weak protection of data in transit
- Make and model of each piece of LAN/WAN equipment

1.6.3 Gaining Access

In this phase, an attacker has gained the access to the necessary resources required to launch an attack. Gaining access to resources is the whole point of a modern-day attack. The usual goal is to either extract information of value to the attacker or use the network as a launch site for attacks against other targets. In either situation, the attacker must gain some level of access to one or more network devices.

1.6.4 Maintaining Access

After gaining access to the resources in the previous phase, now it is important for an attacker to maintain the access for long enough to accomplish his/her objectives. Although an attacker reaching this phase has successfully circumvented all security controls, this phase can increase the attacker's vulnerability to detection.

1.6.5 Covering Tracks

After achieving the objectives, the attacker typically takes step to hide the intrusion and possible controls left behind for future visits. Covering tracks is important so that the intrusion does not point back to the attacker and he/she can get caught.

1.7 Security Measures

In the previous sections, various ways of breaching into a network by an attacker have been discussed. Now it is time to discuss various security measures that can be used to circumvent the break-ins into the network. various security measures that can be used for network security are discussed below.

1.7.1 Firewall

Firewalls are devices or programs that control the flow of network traffic between networks or hosts that employ differing security postures[5]. By employing firewalls to control connectivity, an organization can prevent unauthorized access to its systems and resources. Firewalls are often placed at the perimeter of a network. Such a firewall can be said to have an external and internal interface, with the external interface being the one on the outside of the network.

Various features of firewall are discussed below.

- **Packet Filtering**

  The most basic feature of a firewall is the packet filter, i.e. it controls the incoming and outgoing network traffic by analyzing the data packets and determining whether they should be allowed through or not, based on a ruleset.
- **Stateful Inspection**

  Stateful inspection improves on the functions of packet filters by tracking the state of connections and blocking packets that deviate from the expected state. As with packet filtering, stateful inspection intercepts packets at the network layer and inspects them to see if they are permitted by an existing firewall rule, but unlike packet filtering, stateful inspection keeps track of each connection in a state table.
- **Application Firewall**

  An addition os a stateful protocol analysis capability in stateful inspection is a newer trend. Stateful protocol analysis improves upon standard stateful inspection by adding basic intrusion detection technology-an inspection engine that analyzes protocols at the application layer.

1.7.2 Virtual Private Network

A Virtual Private Network utilizes public telecommunications networks to conduct private data communications. It uses additional protocols to firewall to encrypt traffic and provide user authentication and integrity checking[5]. VPN follows a client and server

approach. VPN clients authenticate users, encrypt data, and otherwise manage sessions with VPN servers utilizing a technique called tunneling.

1.7.3 Intrusion Detection System

Intrusion detection systems (IDS) are network security appliances that monitor network and/or system activities for malicious activity. The main functions of intrusion prevention systems are to identify malicious activity, log information about said activity, attempt to block/stop activity, and report activity[6]. IDS help information systems prepare for, and deal with attacks. They accomplish this by collecting information from a variety of systems and network sources, and then analyzing the information for possible security problems.

Intrusion detection provides the following

- Monitoring and analysis of user and system activity
- Auditing of system configurations and vulnerabilities
- Assessing the integrity of critical system and data files
- Statistical analysis of activity patterns base on the matching to known attacks
- Abnormal activity analysis
- Operating system audit

1.7.4 Intrusion Prevention system

Intrusion prevention systems are considered extensions of intrusion detection systems because they both monitor network traffic and/or system activities for malicious activity. The main differences are, unlike intrusion detection systems, intrusion prevention systems are placed in-line and are able to actively prevent/block intrusions that are detected[6][7].More specifically, IPS can take such actions as sending an alarm, dropping the malicious packets, resetting the connection and/or blocking the traffic from the offending IP address. An IPS can also correct Cyclic Redundancy Check (CRC) errors, not fragment packet streams, prevent TCP sequencing issues, and clean up unwanted transport and network layer options[6].

1.7.5 Honeypot

A honeypot is a trap set to detect, deflect, or in some manner counteract attempts at unauthorized use of information systems. Generally it consists of a computer, data, or a network site that appears to be part of a network, but is actually isolated and monitored, and which seems to contain information or a resource of value to attackers. A honeypot works by fooling attackers into believing it is a legitimate system; they attack the system without knowing that they are being observed covertly. When an attacker attempts to compromise a honeypot, attack-related information, such as the IP address of the attacker, will be collected. This activity done by the attacker provides valuable information and analysis on attacking techniques, allowing system administrators to "trace back" to the source of attack if required.

1.8 Chapter Summary

This chapter serves as an introduction to network security and lays a blueprint of attack lifecycle and techniques as used during network intrusion. It also serves to discuss various security measures.

In the next chapter, a literature survey has been presented.

CHAPTER 2

LITERATURE SURVEY

2.1 Introduction to Honeypot

A honeypot is a deception trap, designed to entice an attacker into attempting to compromise the information systems in an organization. The purpose of honeypot technology is to act as a network defense which will give security professionals insight into how the attacks on their networh is taking place[9].

2.1.1 Definition of Honeypot

The exact definition of a honeypot is contentious, however most definitions are some form of the following:

A honeypot is an information system resource whose value lies in unauthorized or illicit use of that resources.

A more practical, but more limiting, definition is given by pcmag.com:

"A server that is configured to detect an intruder by mirroring a real production system. It appears as an ordinary server doing work, but all the data and transactions are phony. Located either in or outside the firewall, the honeypot is used to learn about an intruder's techniques as well as determine vulnerabilities in the real system" [10].

In practice, honeypot is a computer which masquerade as unprotected. The honeypot records all actions and interactions with users. Since honeypot does not provide any legitimate services, all activity is unauthorized and possibly malicious. Talabis presents honeypots as being analogous to the use of wet cement for detecting human intruders [11].

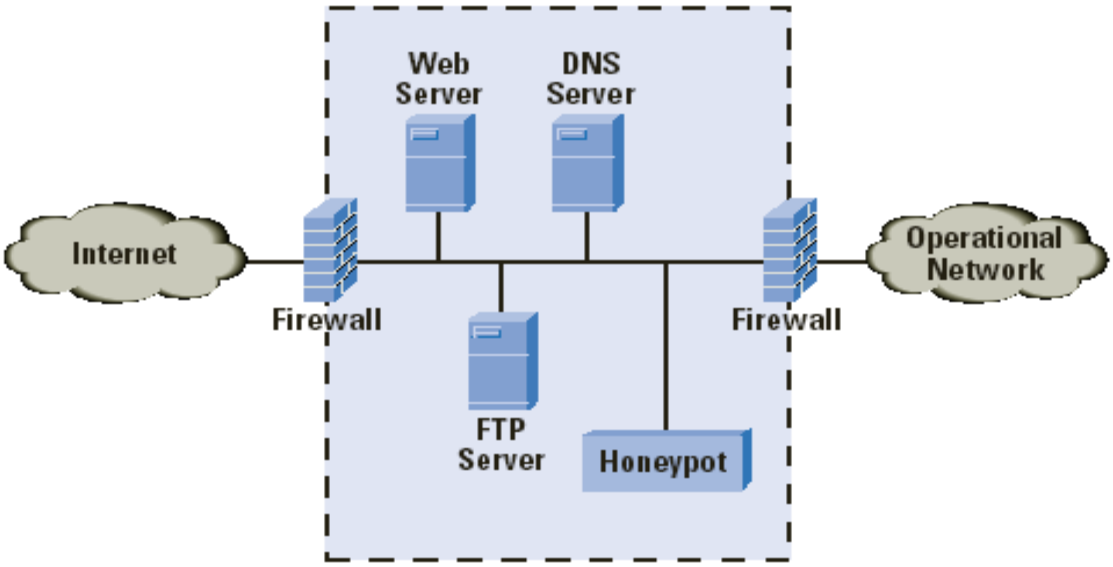

Fig2.1: Network Security using Honeypot[11]

2.1.2 History of Honeypot

The first publically available honeypot was Fred Cohen's Deception Tool Kit in 1998 which was "intended to make it appear to attackers as if the system running DTK had a large number of widely known vulnerabilities"[12]. Moreover honeypot became both publically and commercially available throughout the late nineties. As worms began to proliferate in the beginning of 2000, honeypot proved imperative in capturing and analyzing worms. In 2004, virtual honeypots were introduced which allow multiple honeypots to run on a single server [13]. A detailed history of honeypot can be found in [11] and [14].

2.1.3 Types of Honeypot

There are three categories of honeypots available today, high-interaction, low-interaction and medium interaction. These categories are defined based on the services, or interaction level, provided by the honeypot to potential hackers [14].

- **High Interaction Honeypot**

  High-interaction honeypots let the hacker interact with the system as they would any regular operating system, with the goal of capturing the maximum amount of information on the attacker's techniques. Any command or application an end-user would expect to be installed is available and generally, there is little to no

restriction placed on what the hacker can do once he/she comprises the system. For eg: Mantrap, Honeynet.

- **Low Interaction Honeypot**

  Low-interaction honeypots present the hacker emulated services with a limited subset of the functionality they would expect from a server, with the intent of detecting sources of unauthorized activity [13]. For example, the HTTP service on a low-interaction honeypot would only support the commands needed to indentify that a known exploit is being attempted.

- **Medium Interaction Honeypot**

  A medium-interaction honeypot might more fully implement the HTTP protocol to emulate a well-known vendor's implementation, such as Apache. However, there are no implementations of a medium interaction honeypot, the definition of low-interaction honeypot captures the functionality of medium-interaction honeypot in that they only provide partial implementation of services and do not allow typical, full interaction with the system as high-interaction honeypot.

2.1.4 Physical Existence of Honeypot

Honeypot can be built in many forms, either in the form of physical machine, virtual machine (VM) or emulated virtual host[8]. A physical honeypot is a real computing platform that has its own valid IP address. For example, a computer installed with Fedora Linux or Windows 7 with a running network services, like FTP, Telnet or SMTP. Honeypot VM can be built by using virtualization software like VMWare Workstation, Qemu-KVM, Virtualbox, Parallels Desktop or User Mode Linux (UML). By utilizing either of these tools, honeypot VM can be created with any type of operating system This software is installed on a computing platform and user can create one or multiple VM(s) running on the same platform[16].

2.1.5 Advantages of Honeypot

Tradition security solutions, such as intrusion detection systems, may not be enough in light of more complicated attacks. Honeypots provide a mechanism for detecting novel attack vectors, even in encrypted environments. Advances such as virtualization have made honeypots even more effective. Honeypots have drawbacks, though, so it is important to understand how honeypots operate in order to maximize their effectiveness. Honeypots provide several advantages over other security solutions, including network intrusion detection systems:

a) Fewer false positives since no legitimate traffic uses honeypot.

b) Collect smaller, higher-value, datasets since they only log illegitimate activity.

c) Do not require known attack signatures, unlike IDS [13]

2.1.6 Disadvantages of Honeypot

Honeypots are not perfect, though:

a) Can be used by attacker to attack other systems [15].

b) Only monitor interactions made directly with the honeypot-the honeypot cannot detect attacks against other systems.

c) Can potentially be detected by the attacker.

Everything has its own pros and cons. It is up to the administrator to employ the honeypot in a way that it provides use to the administrator and not the attacker.

2.2 Existing Low Interaction Honeypot

Low-interaction honeypots present the hacker a set emulated services with a limited subset of the functionality they would expect from a server, with the intent of detecting sources of unauthorized activity. Honeypots are generally implemented on the top of an operative system using its networking capabilities, TCP/IP stack[21].

Because of their simple design and basic functionality, Low interaction honeypots are the easiest to install, configure, deploy and maintain. Generally such technologies merely emulate a variety of services. The attacker can interact with the limited predefined services. For example, a low interaction honeypot can emulate a standard Linux or

Windows server with several running services like FTP, Telnet etc. An attacker can send FTP request to the honeypot, get banner that states the operating system and perhaps obtain a login prompt. The attacker can try to login to the FTP server by using brute force or by guessing the passwords. The honeypot would capture and collect those attempts, but there is no real operating system for the attacker to log on to.

Low interaction honeypots are simple, thus they have the lowest level of risk because less functionality offered leads to less mistakes. As there is no real operating system for the attacker to interact with thus attacker cannot use honeypot to attack and monitor other systems. there are many low level interaction honeypots available in market like BOF, Specter, Honeyd etc.

Disadvantages of low-interaction honeypots originate from the same properties as do the advantages. While we can emulate the responses an attacker would expect for known vulnerabilities and exploits, a low interaction honeypot may not respond accurately to exploits that have not been expressed for emulated services[19]. This limits the honeypot's ability to aid in discovering new vulnerabilities that are being exploited or new attack patterns that can utilize those vulnerabilities.

The brief description of these honeypots have been provided below

2.2.1 BackOfficer Friendly

BackOfficer Friendly was developed by Marcus Ranym in 1998. BOF is a low interaction honeypot which can run on Windows or Unix operating systems[18]. It can monitor up to seven emulated services. BOF is very simple, flexible and free. But it offers little interaction with the attacked such that false banners or bogus information like password files cannot be given to attacker to lure the attacker because it does not have such capabilities.

2.2.2 Specter

Specter is a smart honeypot-based intrusion detection system[20] Specter is a commercial honeypot created and supported by NetSec. Specter works only for some Windows systems. A software solution is installed on a system and emulates a variety of services

attackers can interact with which is limited to whatever functionality the specter software provides. Specter have the capability of emulating not only the interactions but vulnerabilities also. It can emulate 13 services. It can imitate like the real world applications by responding with Fake Replies which are more realistic than BOF . The problem with Specter is it operates at the application level which means IP stack is not emulated and it can only emulates the chosen OS based on the seven emulated services. For the tools like Nmap it is easy to detect Specter honeypot which can cause problem.

2.2.3 Honeyd

Honeyd is developed by Niels Provos. It is a low-interaction honeypot. Honeyd monitors the unused IP address space[17] within a local area network it is installed in. It can only emulate services and is used to detect attacks or unauthorized activity. It is an Open Source solution. It has two advantages, one is it can detect connections on TCP port and the other is the emulated services can be modified and the level of interaction as well. Honeyd has only one IP address: the IP address assigned to its single interface. This is the same interface that you use to administer the honeypot. It is this same interface that resides on the network and monitors for suspicious activity.

Various low interaction honeypots are compared in the tabular form as follows:

	BOF	Specter	Honeyd
Freely available	No	No	Yes
Open source	No	No	Yes
Log file support	No	Yes	Yes
OS Emulation	No	Yes	Yes
Services Emulation	7	13	Unrestricted

Table 2.1. Comparison of Various Low Interaction Honeypots

2.3 Honeyd in Detail

Honeyd is a low interaction honeypot and it can emulate various services like ftp, telnet etc. Honeyd is designed primarily as a production honeypot.

2.3.1 Introduction to Honeyd

Honeyd is developed and maintained by Niels Provos of the University of Michigan[14]. First released in April 2002, Honeyd is an OpenSource prepackaged honeypot designed for the Unix platform. OpenSource means that (1) the solution is free, and (2) you have access to the source, which you can customize. 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. However, it does have specific applications to research. As an OpenSource solution, the honeypot can be highly customized, such as designing our own emulated services. This means that honeypot can listen on any port we want, with as little or as much emulation as want. Honeyd detects activity on any TCP port; the emulated services are designed only to deceive attackers and capture their activity.

2.3.2 Architecture of Honeyd

Honeyd is designed to reply to network packets that have the destination IP address of one of the honeypots that it simulates. We need to configure the network appropriately for Honeyd to actually receive such packets. To do this, we either create a special route at the router for the virtual IP addresses or use Proxy ARP.

When the Honeyd daemon receives a packet for one of the virtual honeypots, it is processed by a central packet dispatcher. The dispatcher checks the length of the IP packet and verifies its checksum. The daemon knows only three protocols: ICMP, TCP and UDP. Packets for other protocols are discarded.

The dispatcher queries the configuration database for a honeypot configuration that corresponds to the destination IP address. If no such configuration ex-ists, the default

template is used. Then the dispatcher calls the protocol specific handler with the received packet and the corresponding honeypot configuration. For ICMP, the only packet that is currently supported is the ICMP ECHO request. The daemon answers with an ICMP ECHO reply packet.

For TCP and UDP, the daemon can establish connections to arbitrary services. Services are external programs that receive data on stdin and send their output to stdout. When a connection request is received, the daemon checks if the packet is part of an established connection. In that case, any new data is sent to the already started service program. If the packet contains a connection request, a new process is created to run the appropriate service.

Honeyd contains a simplified TCP state machine, i.e. the three-way handshake for connection establishment and connection teardown via FIN or RST are fully supported. However, receiver and congestion window management is not fully implemented.

An UDP packet to a closed port is correctly answered with an ICMP port unreachable message. This allows tools like traceroute to work correctly. Instead of establishing a connection with a service program, the daemon also supports dynamic redirection of the service. This allows us to forward a connection request for a web server running on a virtual honeypot to a real web server.

It is also possible to redirect connections to the adversary herself, e.g. a redirected SSH connection might cause an adversary to attempt to compromise her own SSH server. Before any packet is sent to the network, it is processed by the personality engine. It adjusts the packet's content so that it seems to originate from the network stack of the configured operating system.

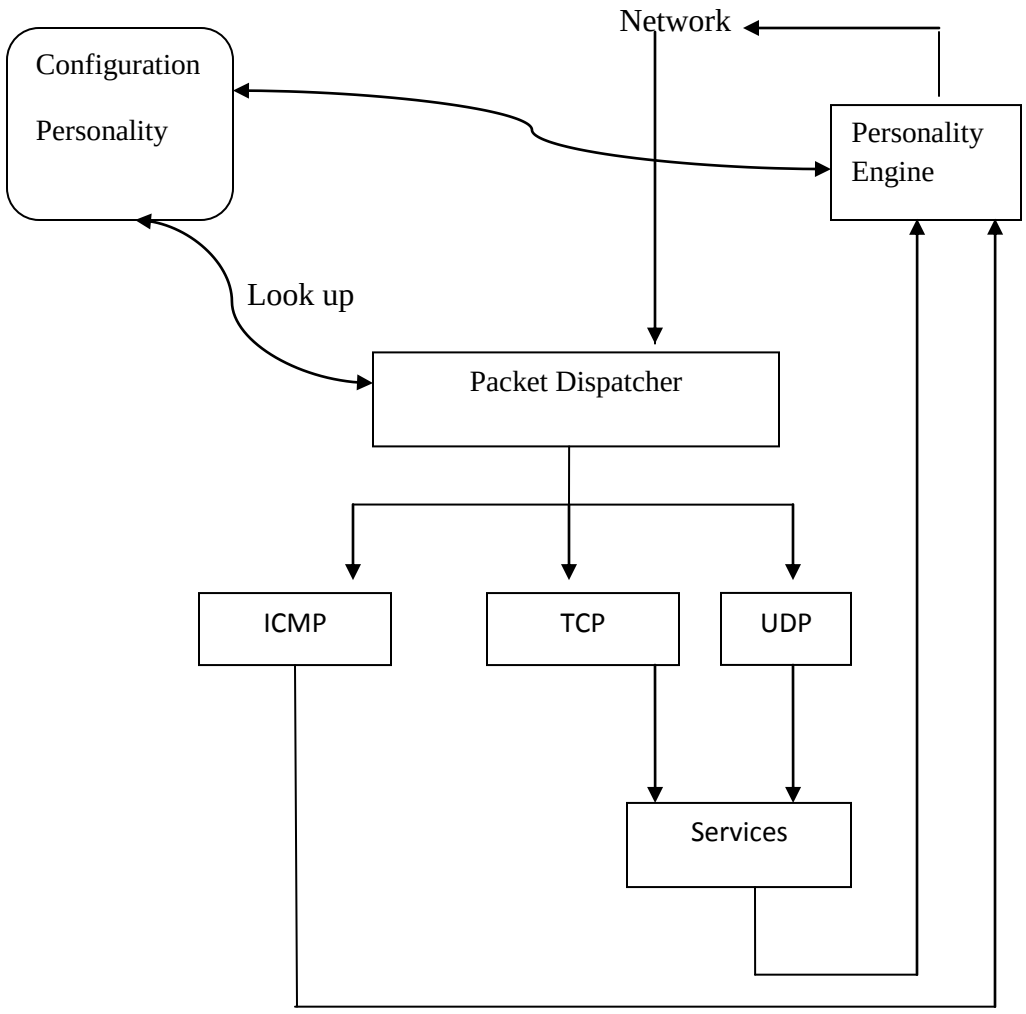

Fig 2.2 Honeyd Architecture[29]

This diagram gives an overview of Honeyd's architecture. Incoming packets are dispatched to the correct protocol handler. For TCP and UDP, the configured services receive new data and send responses if necessary. All outgoing packets are modified by the personality engine to mimic the behavior of the configured network stack.

2.3.3 How Honeyd Works

When an IP address of a nonexistent system is attacked, Honeyd assumes the identity of the victim and interacts with the attacker. It does not do this by assigning thousands of IP addresses to itself at once.

Instead, Honeyd has only one IP address: the IP address assigned to its single interface. This is the same interface that you use to administer the honeypot. It is this same interface that resides on the network and monitors for suspicious activity.

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. Currently, this ability to interact with an attacker is limited to TCP services and ICMP request and ICMP reply. Currently, all UDP ports are assumed to be closed, sending an ICMP port unreachable message.

In the next sub section, an Snort IDS is discussed as it has been used in this work.

2.4 Snort

Snort is an open source network intrusion prevention and detection system developed by Sourcefire[22]. Snort is a tool for small, lightly utilized networks. Snort is available under the GNU General Public License [GNU 89], and is free for use in any environment. A lightweight intrusion detection system can easily be deployed on almost any node of a network, with minimal disruption to operations. It should be cross-platform, have a small system footprint, and be easily configured by system administrator who need to implement a specific security solution in a short amount of time.

Snort is a libpcap-based packet sniffer and logger that can be used as a lightweight network intrusion detection system (NIDS). It features rules based logging to perform content pattern matching and detect a variety of attacks and probes, such as buffer overflows, stealth port scans, CGI attacks, SMB probes, and much more[23]. Snort has a real time alerting capability, with alerts being sent to syslog, Server Message Block (SMB) " WinPopup" messages, or a separate "alert" file.

Snort can run in 4 modes:

- **Sniffer mode**: snort will read the network traffic and print them to the screen.

- **Packet logger mode**: snort will record the network traffic on a file.

- **IDS mode**: network traffic matching security rules will be recorded.

- **IPS mode**: also known as snort-inline, which obtain packets from ip tables instead of from libpcap and then causes iptables to drop or pass packets based on Snort rules[24].

2.4.1 Architecture of Snort

Snort has several important components such as preprocessors and alert plug-ins, these components enable Snort to manipulate a packet to make the contents more manageable by the detection engine and the alert system. Once the packet has been passed through the preprocessors, passed through the detection engine, and then sent through the alert system.

Snort's architecture consists of four basic components:

- The sniffer
- The preprocessor
- The detection engine
- The output

In its most basic form, Snort is a packet sniffer. However, it is designed to take packets and process them through the preprocessor, and then check those packets against a series of rules through the detection engine.

The following figure offers a high-level view of the Snort architecture.

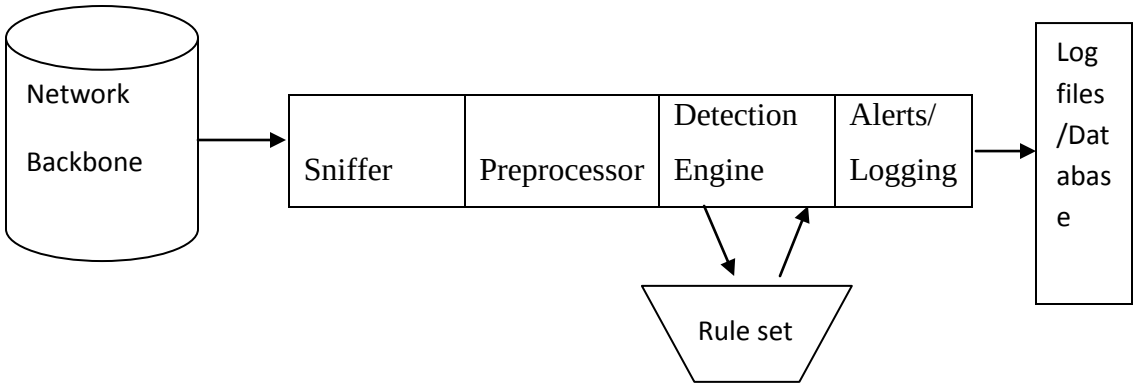

Fig 2.3 Snort Architecture[24]

The preprocessor, the detection engine, and the alert components of Snort are all plug-ins. Plug-ins are programs that are written to conform to Snort's plug-in API. These programs used to be part of the core Snort code, but they were separated to make modifications to the core source code more reliable and easier to accomplish.

1. Snort takes all packets from the network backbone.
2. Then it sends them through a preprocessor to determine if they are packets and how they should log.
3. Next, it sorts the packets according to the packet type. This is for storage of different packets on the IDS. This is the detection engine.

4. Finally, it is the administrator's task to decide what to do with the packets- usually log them and store them. This is logging and database storage.

2.4.2 Snort Rules

Snort uses a simple, lightweight rules description language that is flexible and powerful[24]. Snort rules have the following parts:

- **Rule Header**

  The rule header[23] contains the information that defines the who, where, and what part of a packet, as well as what to do in the event that a packet with all the attributes indicated in the rule should show up. Rule header in a Snort contains all those portions which are compulsory part.

- **Rule options**

  Rule option[23] form the heart of Snort's intrusion detection engine, combining ease of use with power and flexibility. All Snort rule options are separated from each other using the semicolon (;) character. Rule option keywords are separated from their arguments with a colon (:) character.

An example is given below.

```
alert tcp any any -> any 21 (content:"site exec"; msg:"site exec buffer overflow attempt";)
```

The text upto the first parantheses is the rule header:

```
" alert tcp any any -> any 21"
```

The text enclosed in the parantheses is the rule options:

```
"(content:"site exec"; msg:"site exec buffer overflow attempt";)"
```

The next sub section is the tool which is used to employ ftp attack on honeyd is discussed.

2.5 Metasploit

Metasploit came about primarily to provide a framework for penetration testers to develop exploits. Metasploit Framework (MSF) is an open source tool, which provides a framework for security researchers to develop exploits, payloads, payload encoders, and tools for reconnaissance and other security testing purposes[25]. Although, it initially started off as a collection of exploits and provided the ability for large chunks of code to be re-used across different exploits, in its current form it provides extensive capabilities for the design and development of reconnaissance, exploitation, and post-exploitation security tools[26]. The Metasploit project was originally started as a network security game by four core developers. It then developed gradually to a Perl-based framework for running, configuring, and developing exploits for well known vulnerabilities. The project core was dual-licensed under the GPLv2 and perl Artistic Licenses, allowing it to be used in both open-source and commercial projects[27].

2.5.1 Architecture of Metasploit

The Metasploit framework can be broken down into three main libraries. Firstly, there is the Rex library, which is short for the Ruby Extension Library [28]. It contains most of the framework's core features and tools, some of which are specific to the application domain, that were built to enhance the default Ruby library. The Rex module was designed to depend strictly on the default installation of Ruby (default libraries) and is a centerpiece of the framework.

MSF Core library works as an API and extension for Rex and whose purpose is to provide a low-level interface that will allow peripheral modules to interact with Rex. This core library is extended by the MSF Base library which is designed to provide a simpler interface to interact with the core framework and some utility classes. Core is responsible for all the required interfaces that allow for interactions with exploit modules, sessions (SessionManager), and plug-ins (PluginManager). This is extended by the framework base library, which is designed to provide simpler wrapping routines (Simple) for dealing with the framework core and providing utility classes for dealing with different aspects of the framework like serializing module state (Serializer) to different output formats.

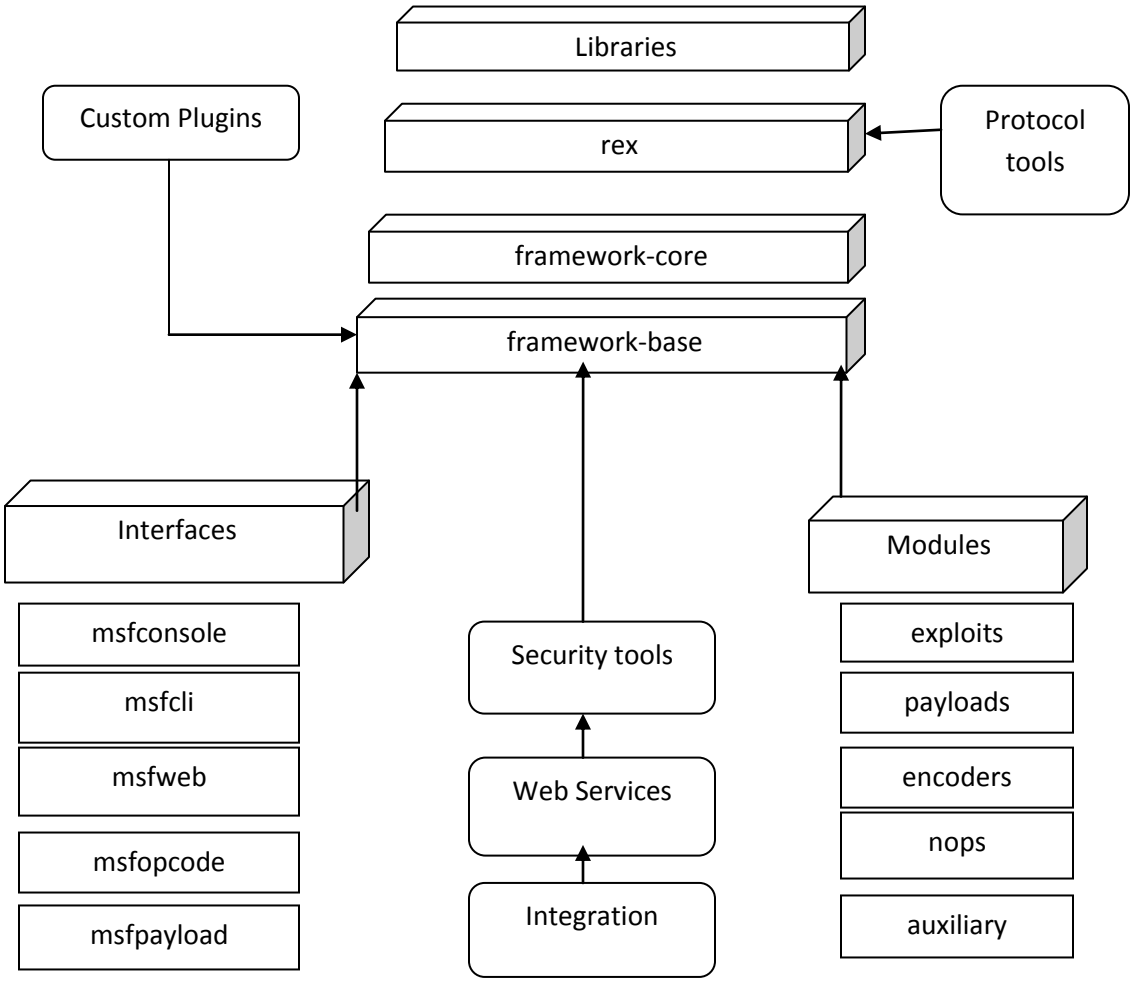

Fig 2.4 Metasploit Architecture[28]

The various interfaces present in meatsploit are discussed below

- **Msfconsole**

  Msfconsole is the entry point of the Metasploit console user interface. It starts by using Ruby's optparse[6] library as a helper to parse arguments given as input. Once arguments are validated, Msfconsole instantiates and runs as a thread in the Console::Driver class of the UI module.

- **Msfcli**

  Msfcli provides a powerful command-line interface to the network. It is useful when a large number of systems need to be tested for the same vulnerability. When using msfcli, variables are assigned using '=' and that all options are case-sensitive.

- **Msfweb**

  The msfweb interface is the only GUI currently available to the MSF. It offers no security. This interface can be launched with a number of options, which are available with the -h switch.

- **Msfopcode**

  The msfopcode connects the metasploit web server to retrieve the information. msfopcode does not support the use of proxies yet.

- **Msfpayload**

  The msfpayload utility enables the user to modify existing payloads depending on supplied parameters on the command line, and obtain the output in C, Perl or Raw.

A penetration module can be one of an exploit, payload, encoder, NOP, or Auxiliary, which are described below.

- **Auxiliary**

  This module contains all the tools a penetration tester would use in the initial phases of planning out an attack. These are Tools such as packet sniffers, port scanners, etc.

- **Exploits**

  All standalone exploits belong in this module. It contains both passive and active exploits. An example of an active exploit is one that exploits a buffer overflow whereas a passive exploit is something along the lines of a fake DNS server which re-routes an unsuspecting user to a malicious site.

- **Encoders**

  This module contains various encoders which are used to encode the payload before it gets sent to a remote computer. This is done to prevent the payload from being detected by an anti-virus program.

- **Payloads**

  This module is composed of the various payloads a penetration tester may wish to deposit into a target system. Payloads usually consist of some code to run as well as some parameters defining how a connection to the compromised system might be made.

- **Nops**

  This module is composed of a few different generators whose purpose is to generate no operation instructions that are used as padding around some of the payloads in order to keep their size consistent.

2.6 FTP Attacks

File Transfer Protocol is the protocol used for exchanging files over the Internet. There are various ftp attacks that are known. Some of them are discussed below:

2.6.1 Brute Force Attack

A brute force attack is a way of cracking passwords by guessing. But these "guesses", delivered one after another, are done very rapidly. They are spewed forth by hacking tools that reference a really long list of possible passwords, often called a wordlist. A brute-force attack, or exhaustive key search, is a cryptanalytic attack that can, in theory, be used against any encrypted data except for data encrypted in an information-theoretically secure manner. Such an attack in the context of FTP is utilized when it is not possible to take advantage of other weaknesses. It consists of systematically checking all possible logins from a list of presupposed or known usernames and passwords.

2.6.2 Application Level Floods

There are various Denial of Service (DOS) causing exploits which can confuse the FTP server software and make it consume all available memory or disk space or CPU time such as buffer overflow. Various types of DOS rely primarily on brute force in which the target is to be flooded with an overwhelming number of login attempts in a short duration, over saturating its connection bandwidth or depleting the target system resources.

2.6.3 Bounce Attack

To conform with the FTP protocol, the PORT command has the originating machine specify an arbitrary destination machine and port for the data connection. However, this behavior also means that an attacker can open a connection to a port of the attacker's choosing on a machine that may not be the originating client. Making this connection to an arbitrary machine for unauthorized purposes is the FTP bounce attack.

2.6.4 Port Stealing

Many operating systems assign dynamic port numbers in increasing order. By making a legitimate transfer, an attacker can observe the current port number allocated by the server and "guess" the next one that will be used. The attacker can make a connection to this port, thus denying another legitimate client the ability to make a transfer. Alternatively, the attacker can steal a file meant for a legitimate user. In addition, an attacker can insert a forged file into a data stream thought to come from an authenticated client. This problem can be mitigated by making FTP clients and servers use random local port numbers for data connections, either by requesting ports from the operating system or using system dependent mechanisms.

2.6.5 Protecting Usernames

Standard FTP specifies a 530 response to the USER command when the username is rejected. If the username is valid and a password is required, FTP returns a 331 response instead. In order to prevent a malicious client from determining valid usernames on a

server, it is suggested that a server always return 331 to the USER command and then reject the combination of username and password for an invalid username.

The above given are some of the ftp attacks studied. A secure version of ftp is in market called FTPS is an extension to the FTP standard that allows clients to request that the FTP session be encrypted. The other option is SFTP, or secure FTP, is a program that uses Secure Shell (SSH) to transfer files. Unlike standard FTP, it encrypts both commands and data, preventing passwords and sensitive information from being transmitted openly over the network.

2.7 Chapter Summary

This chapter provides a thorough run through of various methodologies that surround the application which is presented in this thesis. Honeypots are essential because they are used to create dummy systems. Utilities such as Snort and metasploit are used to dump packets and conduct various attacks respectively.

In the following chapter, the problem statement and objectives of the thesis have been described.

CHAPTER 3

PROBLEM STATEMENT

3.1 Problem Definition

File Transfer Protocol (FTP) is used for exchanging files over the internet. FTP uses the TCP/IP protocols to enable the data transfer. FTP is most commonly used to download a file from a server using the Internet or to upload a file to a server. Everything is not perfect, there are many problems related to FTP protocol also. FTP protocol like other protocols is prone to brute force attacks, application level floods and many more.

Various security measures can be employed which can provide security like firewalls, IDS etc but these security measures create large logs which makes the job of administrator tough. These security measures have databases which consists of signature of attacks according to which the attack is detected. Administrator has to spend lots of time to go through large piles of data to find the valuable information. But these measures fail to detect the attacks which are new or signatures are not in the database. Here where comes a new technology -- Honeypot.

Honeypot is a tool through which various services are emulated according to the requirement. Honeyd forms an integral part of the proposed application. It emulates the FTP service so that various attacks and serves as a host on which FTP attacks are targeted. The attacks originating from various IP addresses are caught by the honeypot and their packets are logged. Honeyd uses a script to emulate the FTP service, this script has been modified according to the requirement. Snort gets invoked after a certain IP address crosses a threshold for the number of brute force attacks hence commencing a packet dump. A stack has been designed and developed for capturing FTP brute force and application level flood attacks using honeyd, Snort, JVM in a virtual machine. A java application has been written which serves as an intermediate processor for future steps which could include invoking snort, sending a message to administrator in case an application level flood attack is detected.

3.2 Objectives

1. To design and a develop a virtual honeypot framework to emulate FTP server.
2. Modify the ftp.sh script to add new users and invoke snort.
3. Generate attacks using metasploit which target the FTP honeypot.
4. Testing and validation of the proposed framework.

In the following chapter, the work done, experimentation and results have been showed.

CHAPTER 4

IMPLEMENTATION DETAILS AND RESULTS

4.1 Experimental Setup

For this experiment, two virtual machines were created using VMWare hypervisor. The first virtual machine has Ubuntu operating system running in it. This virtual machine also runs the honeypot and the remaining utilities used for packet capturing. The second virtual machine runs on the Backtrack operating system which comes with a pre-installed distribution of metasploit. This VM is used to test the honeypot stack running in the VM that runs Ubuntu.

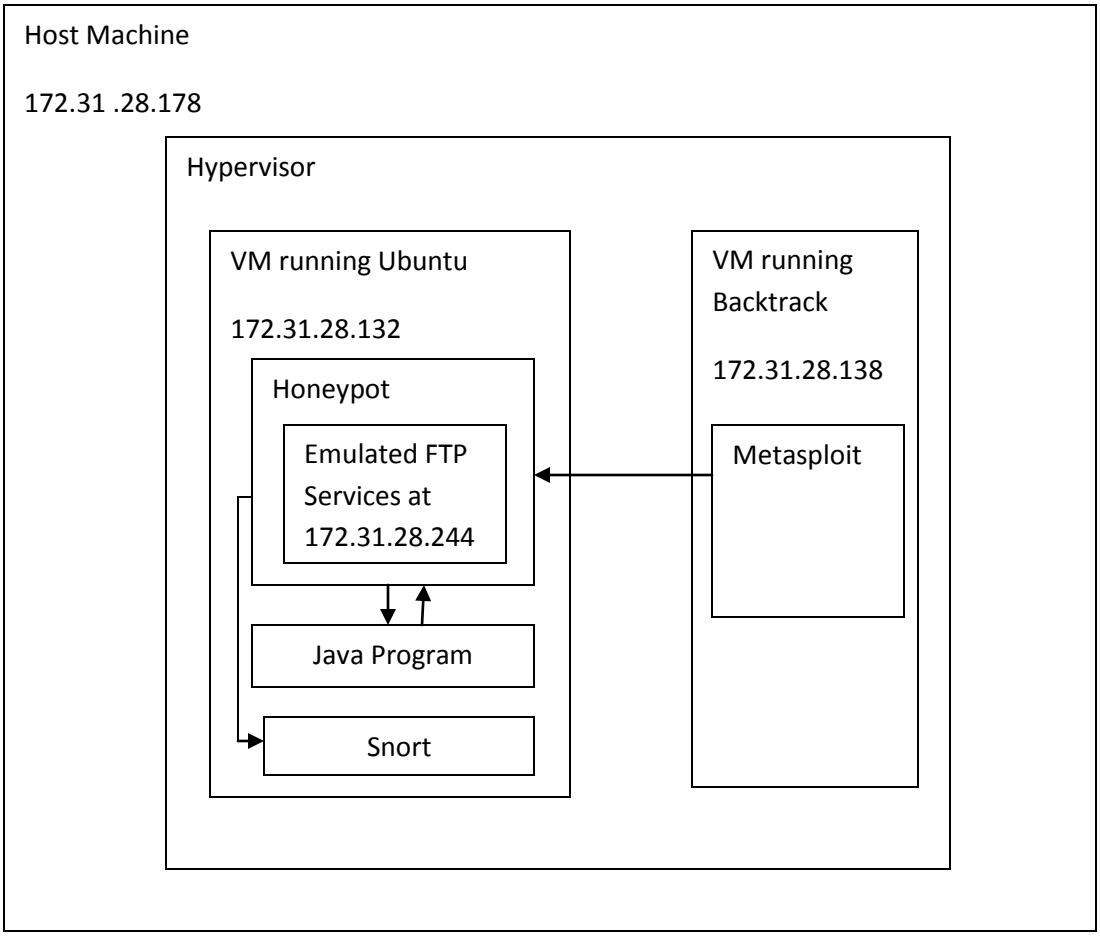

Fig 4.1 Experimental Setup

4.1.1 Software Requirements

1. VMware Workstation version 6.X

2. Any linux distribution capable of running honeyd and snort.

3. Honeyd version 1.5c.

4. Java Virtual Machine version 1.7

5. Snort version 2.9.2

4.1.2 Hardware Requirements

1. CPU: Core2duo compliant CPU or higher.

2. RAM: Minimum 4GB, 8GB recommended.

3. Storage: At least 20GB HDD space to build virtual machine.

4. Network: Standard 10/100mbps ethernet port.

4.1.3 Assumptions

The following information are used in the deploying environment but settings can be changed according to the experiment:

IP address of Windows Host computer	172.31.28.178
IP address of Ubuntu machine in VMWare	172.31.28.132
IP address of Honeyd emulated FTP server	172.31.28.244
IP address of Backtrack machine in VMWare	172.31.28.138

Table 4.1: Assumptions about IP addresses

4.2 Installation and Configuration

As an OpenSource solution designed for Unix, Honeyd does not provide any support or a nice GUI for installation or configuring. The following steps are pertinent to Ubuntu Linux.

4.2.1 Steps to Install Honeyd

1. Download the binary distribution for honeypot using the following command

    #apt-get install honeyd-common

2. Download the ftp emulation script from

    http://www.honeyd.org/contrib/mael/ftp.sh

3. Use the root terminal to place this file in "/usr/share/honeyd/scripts/unix/linux" directory.

Once started, Honeyd will interact with any traffic sent its way. The command to start Honeyd is

```
honeyd -p /etc/honeyd/nmap.prints -f /etc/honeyd/honeyd.conf
```

The first command is executing the binary Honeyd. The -f option is the location of the Nmap fingerprints file. This is the database of signatures used by Honeyd to emulate different operating systems at the IP stack level. If an attacker uses Nmap to fingerprint operating system, Honeyd will use the Nmap fingerprint database to determine how to respond to the attack, based on the operating system you are emulating.

The second option is -f, which is the location of the Honeyd configuration file. This configuration file determines which services you emulate and the operating system type.

The last option is the network that we want to monitor. This means that whenever Honeyd receives a packet for a nonexistent system, and the IP address belongs to the monitored network, Honeyd will assume the identity of the destination IP address and interact with the attacker. If no network address is given to the monitor, Honeyd will interact with any IP address sent its way.

Honeyd works by creating templates that define the characteristics of a honeypot, including operating system type, the ports they listen on, and the behavior of the emulated services. Each template is given a name. we have two templates—default and windows—but we can use as many templates as you want. Then apply the IP addresses or systems to whichever templates we want. In the case of a default template, there is no IP address to which it applies. This means then when we ran our Honeyd honeypot to listen on the network, the default template applies to all connections it receives for the network.

Under each template are the configuration parameters—how the honeypot will behave. The first characteristic is the OS type, or personality. This determines how the system will behave at the OS IP stack level, the Nmap fingerprint that is to be used. This does not affect the behavior of the scripts.

After we have determined the personality of the honeypot (its operating system type), we need to know the behavior of each service. We have the following options whenever a connection is made to any TCP or UDP port.

• Reset: This means that Honeyd will send a TCP reset (RST) for TCP connections or an ICMP unreachable for UDP connections. This emulates a closed TCP or UDP port, there is no service bound to the port.

• Open: This means that Honeyd will act as if the service is open and will acknowledge (ACK) all TCP data that it receives. However, there is no service emulation.

• Block: The means that Honeyd will drop and ignore all connections to the ports. This emulates a firewall's port or service.

• Script: This calls on predefined scripts to interact with the attacker. It is these scripts that emulate services. This is limited to TCP only.

4.2.2 farpd

When we want a single honeypot to claim traffic for a few tens of IP addresses, it is impractical and resource consuming to do it with means of configuring its network interface for all the to ARP requests arriving to the network interface of the honeypot.

ARP requests are broadcast packets used to discover which machine (more specifically which MAC address) has a specific IP address. For a given IP address in a LAN, the LAN gateway directs the traffic to this IP to the host that replied to the corresponding ARP request. Under normal circumstances, it is the operating system that takes care of responding to ARP requests. By having farpd reply to these requests, the traffic can be effectively directed for any IPs in the LAN to the host want, without actually configuring the host network interface for all these addresses. The farpd command is used as follows:

 # farpd <ip address>

4.2.3 Steps to Install Snort

1. Make sure the apt is synced with the repositories, if not use:

#apt-get update

2. Use the following command to install snort:

#apt-get install snort

4.3 Emulation of Services on Honeyd

Honeyd is a generic framework application which is capable of running scripts. These scripts can be used to emulate any kind of service.

4.3.1 Emulation of FTP service on Honeyd

Honeyd uses personalities which is basically a configuration that ties a given service with an IP address. These personalities are stored in "honeyd.conf" file, after defining its personality as an FTP server the default behavior for both TCP and UDP has to be set as Reset. This means if an attacker connects to any port has not been defined, the default response is to send the attacker an RST packet for TCP connections or an ICMP port unreachable for UDP connections thus giving the impression of a closed port. The following line in honeyd.conf tells honeyd what action to take when a connection is made on port 21

add template tcp port 21"/usr/share/honeyd/scripts/unix/linux/ftp.sh $ipsrc"

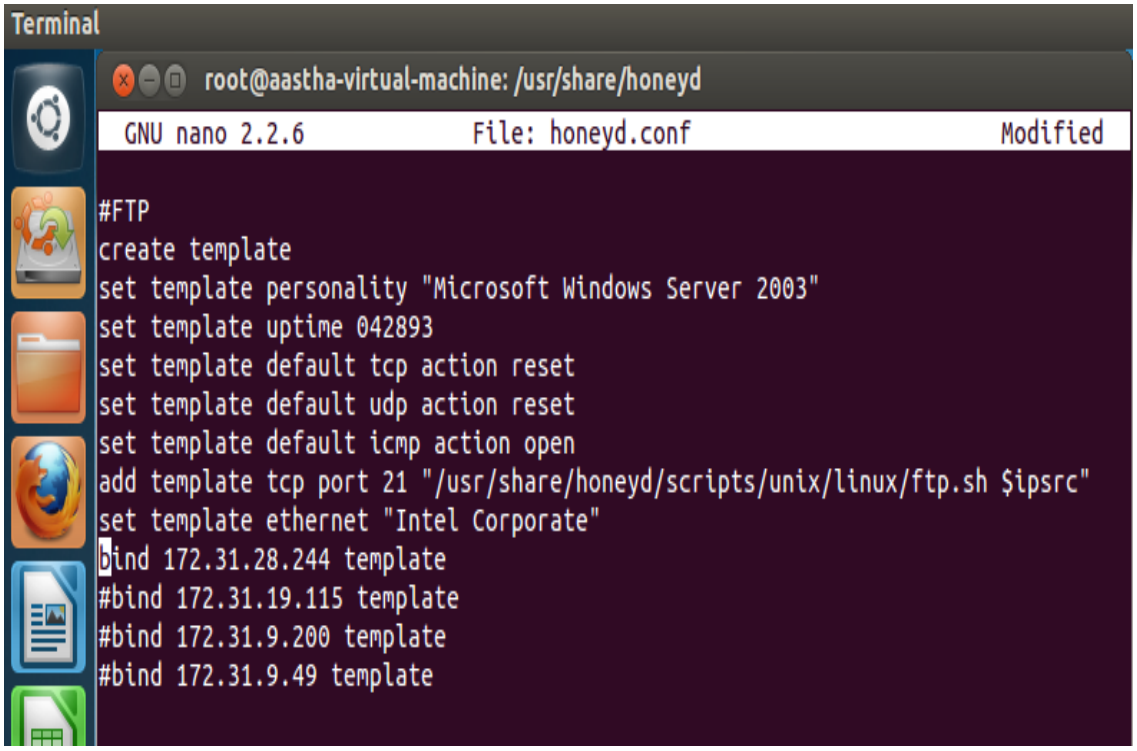

Fig4.2: Honeyd.conf file having ftp server emulation statements

Figure 4.2 shows how the default rules and port specific rules are added in honeyd.conf. The default actions have been set to reset for both TCP and UDP connections on unidentified ports.

4.3.2 ftp.sh

ftp.sh is the shell script which runs and give the illusion of real FTP server to the attacker. ftp.sh is modified according to the requirements. In original ftp.sh which is open for public use, only allows anonymous user to get connected to the ftp server and use the services provided by the emulated server. The original version of "ftp.sh" was modified to accommodate two more logins. Figure 4.3 shows these modifications

```
elif [ "$AUTH" == "USER" ] && [ "$PASS" == "user" ]
then
    echo -e "230-Hello User at $IPSRC,\r"
    echo -e "230-we have 911 users (max 1800) logged in your class at the moment.\r"
    echo -e "230-Local time is: $DATE\r"
    echo -e "230-All transfers are logged. If you don't like this, disconnect now.\r"
    echo -e "230 User login ok, no access restrictions apply.\r"
elif [ "$AUTH" == "POSTGRES" ] && [ "$PASS" == "postgres" ]
then
    echo -e "230-Hello User at $IPSRC,\r"
    echo -e "230-we have 911 users (max 1800) logged in your class at the moment.\r"
    echo -e "230-Local time is: $DATE\r"
    echo -e "230-All transfers are logged. If you don't like this, disconnect now.\r"
    echo -e "230 User login ok, no access restrictions apply.\r"
else
  echo -e "530 Login incorrect $1.\r"
```

Fig 4.3 Modified Shell script

The two users added to the script were 'User' and 'postgres' with same passwords as their usernames.

The second modification that was made to script was invocation of an intermediate java program that serves to co-ordinate the actions of invoking snort and alerting the system administrator.

```
else
  echo -e "530 Login incorrect $1.\r"
        java -cp ./scripts/unix/linux prog "$IPSRC" #invoke the java program and send the
        #source ip as an argument
        output=$? #store the value returned by the java program in a variable named output

        if [ $output == "10" ]
        then
                sudo /usr/share/honeyd/scripts/unix/linux/snort.sh
        elif [ $output == "13" ]
        then
                echo "The honeypot is under a DOS attack from $IPSRC !!"|wall
        fi
fi
;;
```

Fig 4.4 Modification made to invoke the java program, snort and admin alert.

4.4 Experiment 1

In this experiment the brute force attack detection capability of the stack was tested.

4.4.1 Methodology

This experiment targets testing the brute force threshold of the application, if the java program returns 10, it signals the ftp script about a brute force attack and the stack should then invoke snort to start dumping packets.

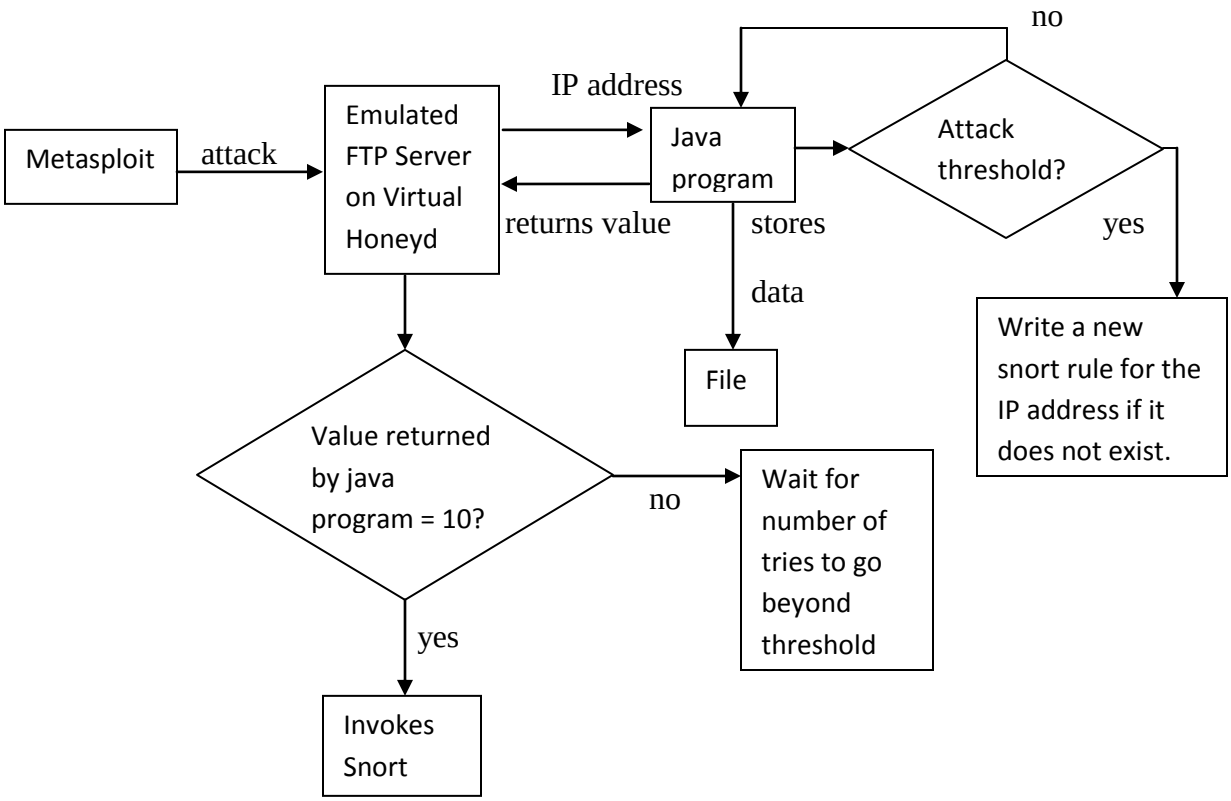

Fig 4.5 Flowchart for Prevention of Brute Force Attack

Figure 4.5 shows how a brute force attack is detected, after the number of attacks cross a certain threshold the shell script invokes snort. The java program is also responsible for writing the snort rule using which snort start saving the packets. The rule file's name is the same as the IP address of the remote computer. The file is not overwritten in case it is already there. The return code from the java program is stored in the $output shell variable, the script invoke snort when the value of this variable becomes 10. The value 10 is only returned by the java program finds there's no snort rule file for the IP, this saves from duplicate invocations of snort.

4.4.2 Brute Force Attack and Results

Following are the steps used to conduct the brute force attack.

- **Scan the Network**

  The first thing to do is to use a security scanner like nmap to discover what hosts are there on the network and the ports/services that are open. If nmap says that the port is open that means that port is hack able.

Fig 4.6 FTP Service Discovery

From the figure 4.6, it is obvious that the ftp port is open and can be attacked.

- **Generate a List of Valid Usernames and Passwords**

  After knowing that the port 21 is open, it is time to find a valid username and password by using dictionary attack. There are several tools that can be used to generate a wordlist. Metasploit Framework has a specific module for attacking FTP servers. Type msfconsole in the terminal to start metasploit.

Fig4.7 Starting Msfconsole

Figure 4.7 shows the starting of metasploit using the command 'msfconsole'. After writing the command, msf starts initializing and starts to give the cursor as msf>

Metasploit Framework has a specific module for attacking FTP servers. These modules cover a myriad of FTP attacks. The module which is pertinent to the application is brute force attack module. Name of this module is ftp_login, it also requires two variables to be set, these variables are file paths where a username database and a password database are found. These are used during the brute force attack to find a valid logon.

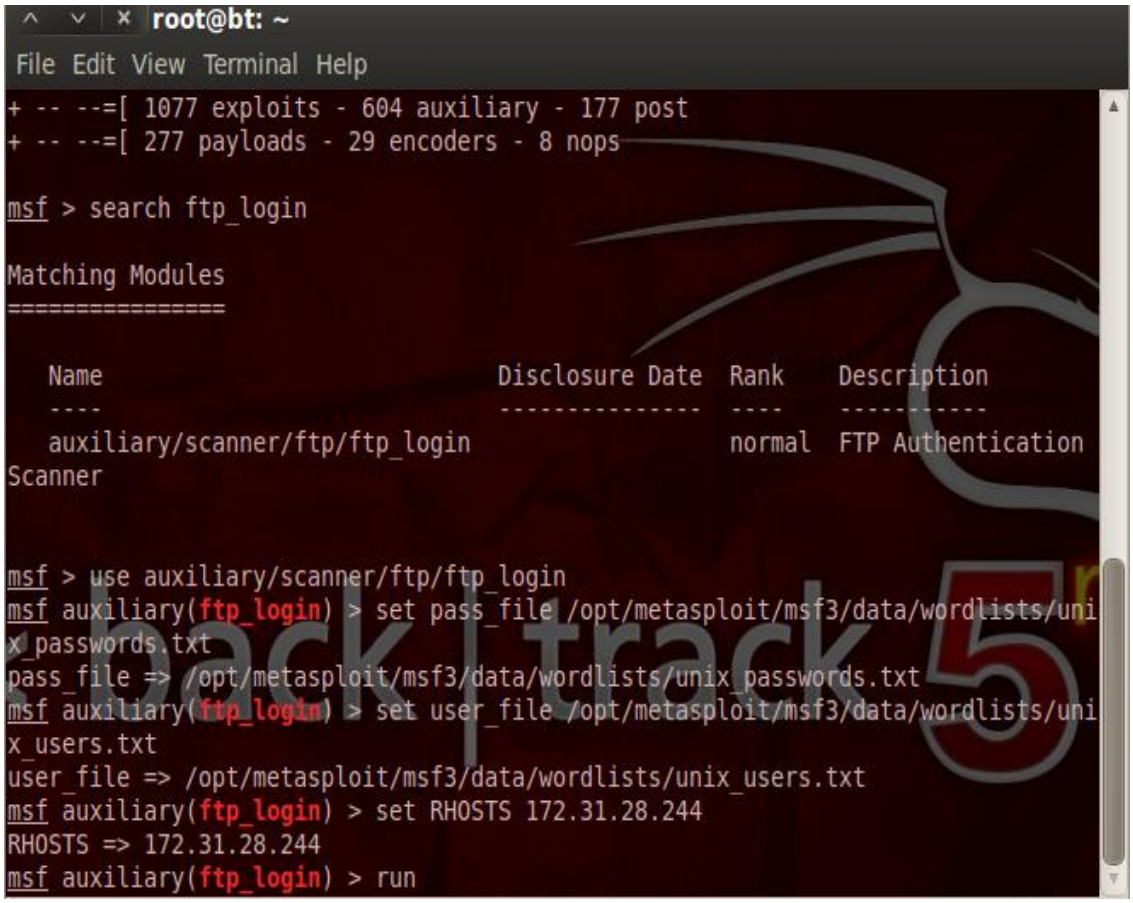

Fig 4.8 Ftp_Login Module

Now that the FTP scanner has been found it is time to configure it. Of course good wordlists for the usernames and the passwords is needed. Metasploit ships with a unix username and passwords files, these files have been used to conduct the attack. The scanner is set according to the above image and type run in order to start the scanner. The run command initiates the brute force attack, over its run length it comes across a logon which gives access to FTP service.

When the scanner finds any valid credential that satisfies the username and password for the ftp server it prefixes a '+' in green to that line whereas all other invalid credentials have '*' in blue.

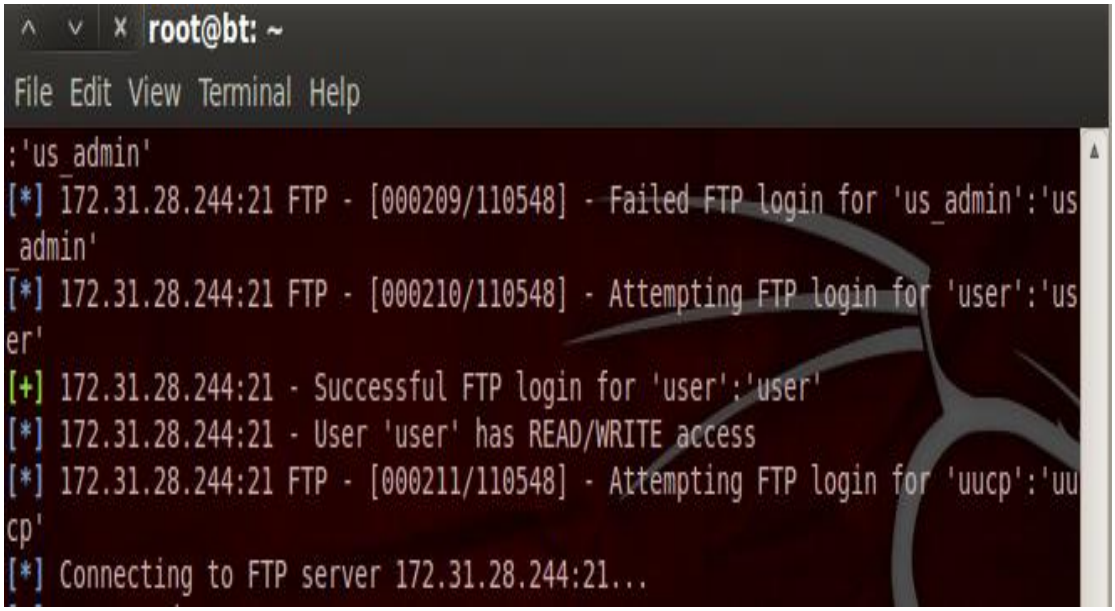

Fig4.9 Valid "user" credential

Figure 4.9 shows that the scanner has found a valid username and password for the ftp server. Hence the valid credential can be used to obtain the services of ftp server.

The other valid credential is 'postgres' user with password as 'postgres'.

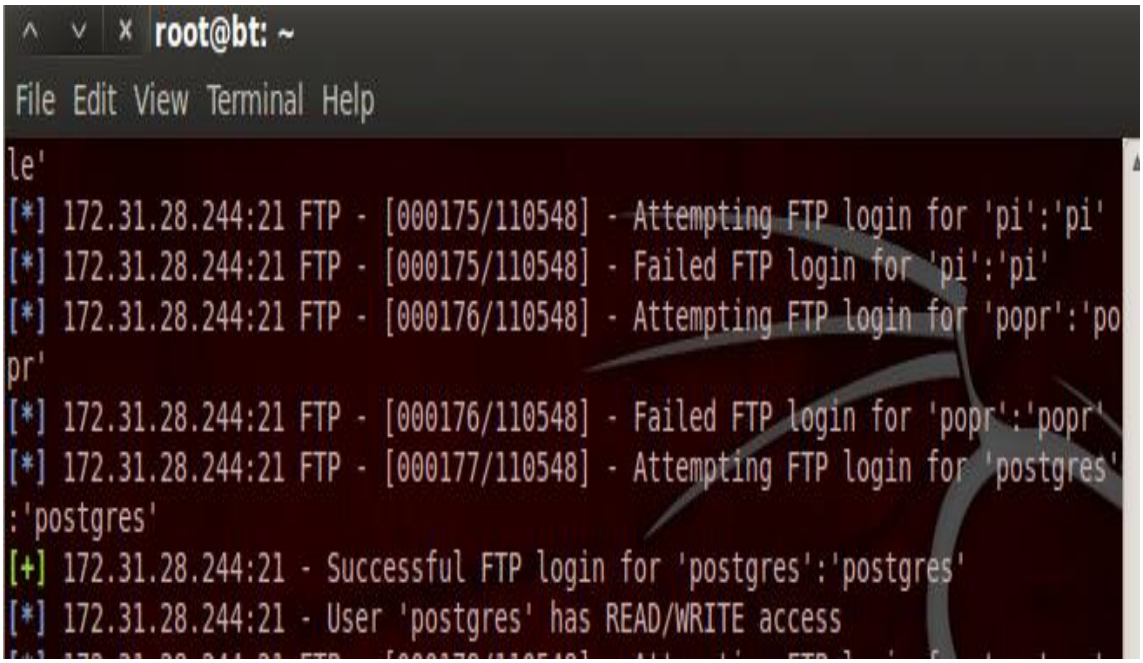

Fig.4.10 Valid Username 'postgres'

- **Confirming the logon**

  The logins brute forced by the scanner can be confirmed by using a simple ftp client .

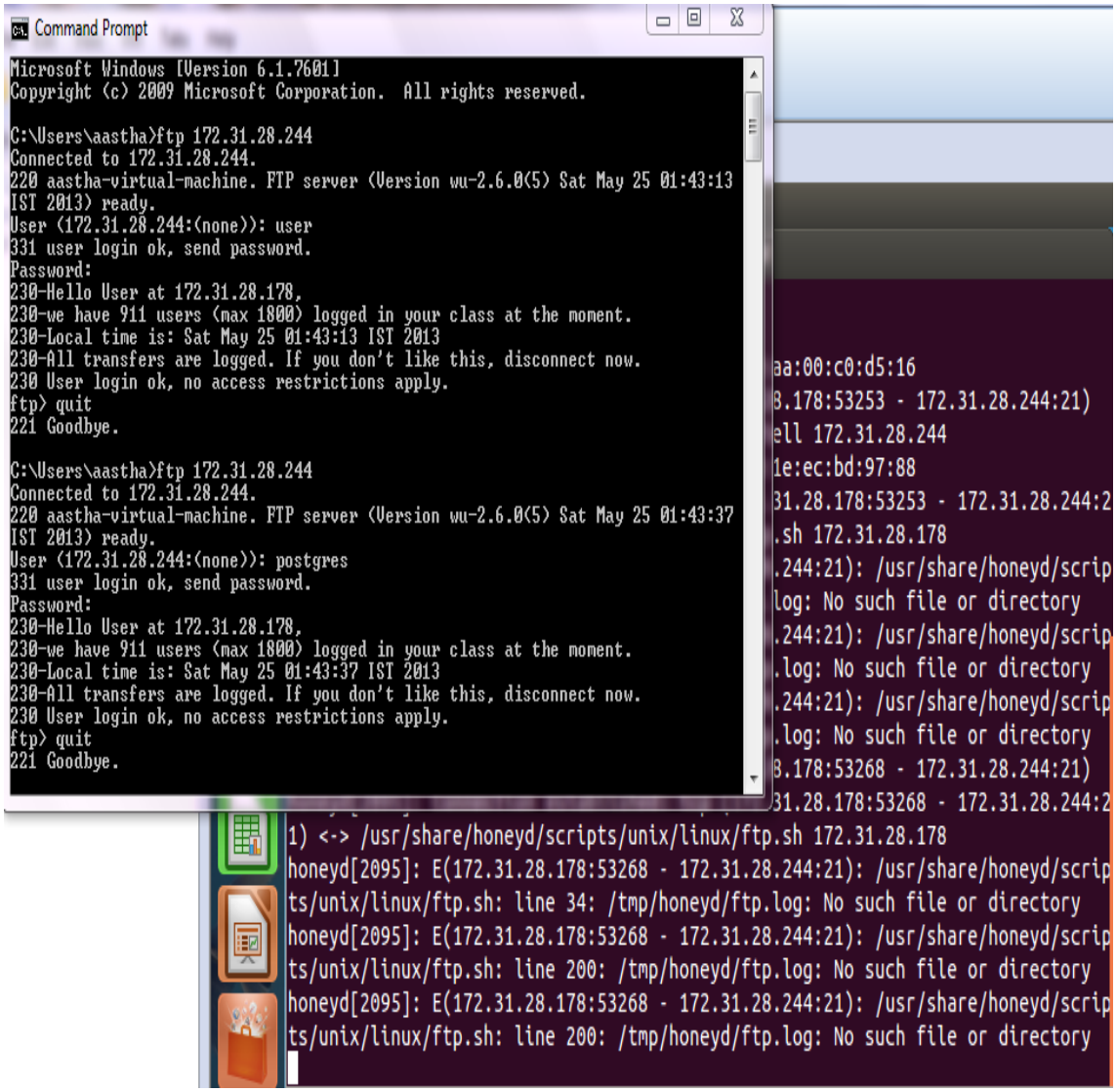

Fig4.11 Successful Logon to Ftp Server

Figure 4.11 shows the successful logon of both of the credentials given by the scanner.

- **Counteracting the Attack using Snort**

As the above steps shows how to get into ftp server, it is now time to prevent the attack. A hacker would always aim for your weak points. That's why FTP service, being practically devoid of any strong security feature, will make you a sitting duck. But there's still something that can be done about it. To turn the tables around, aim should be at the attack's weakest points. Incidentally, a brute force attack has a couple of characteristics that you can use to your advantage:

1. It makes a lot of noise which means it's easy to detect, and

2. It requires a lot of computational resources (e.g. storage and CPU) to crack strong passwords.

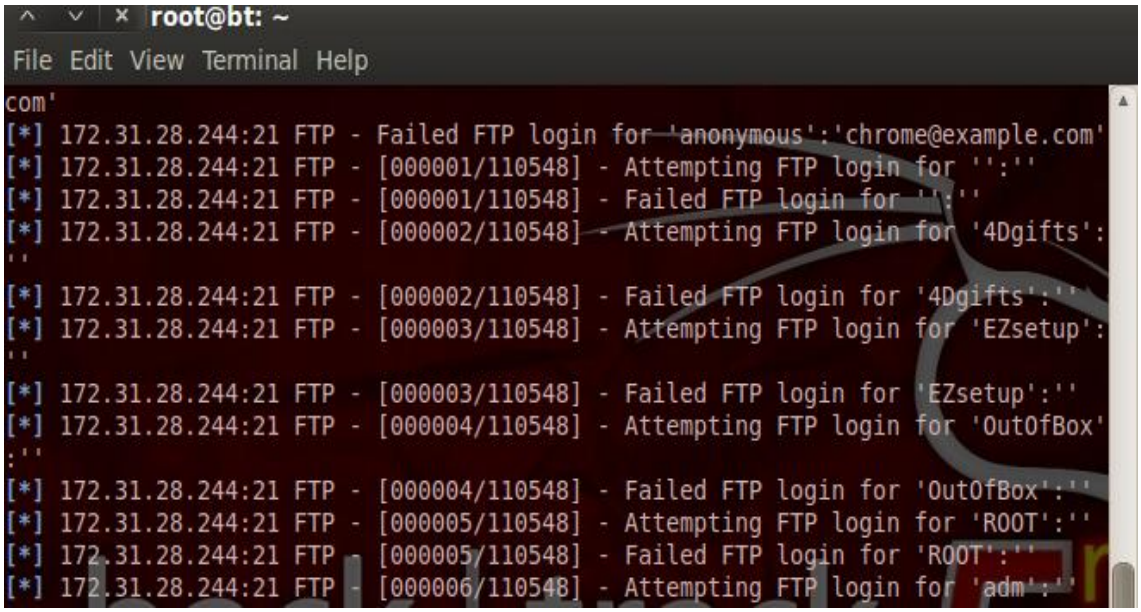

Fig4.12 Failed Login Attempts

Figure 4.12 shows the failed login attempts and a lot can be seen of those whenever ftp server's subjected to a brute force attack. There were a couple of things done to take advantage of this particular characteristic.

Limit the number of times a user account or an IP address can login. The server was configured so that once the limit has reached, it automatically blocks further attempts from the same account or IP address. This was done by invoking Snort.

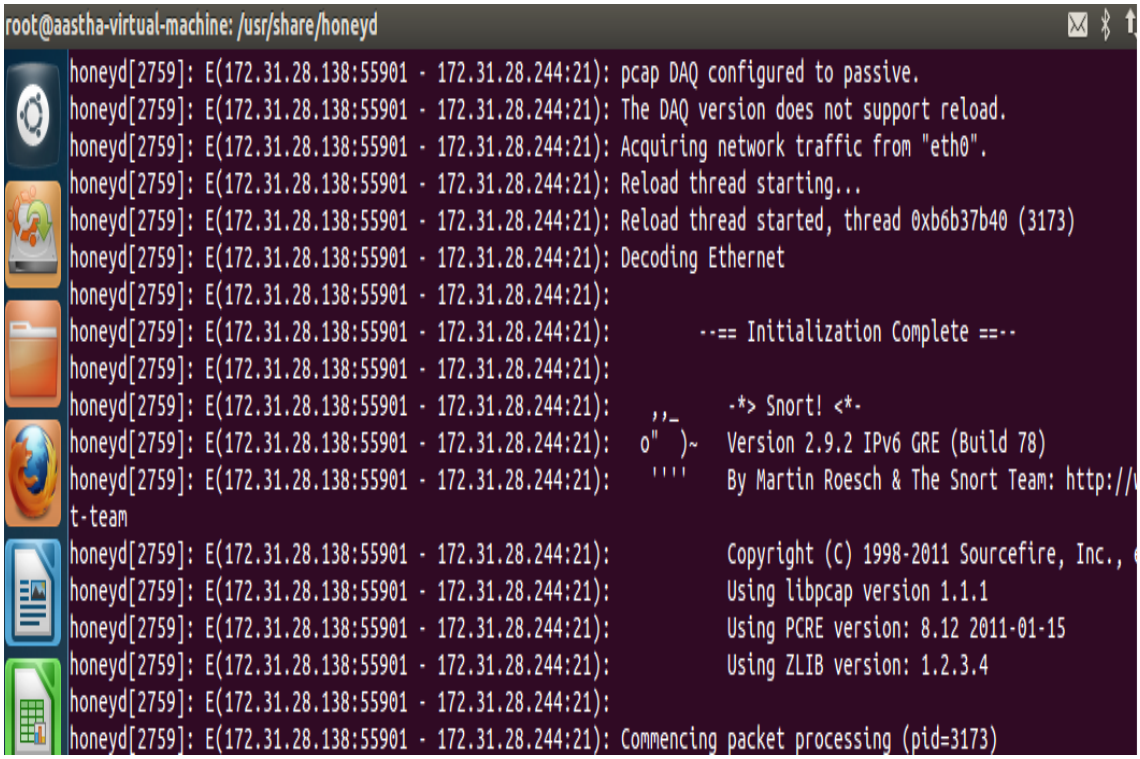

Fig4.13 Invoking Snort

Figure 4.13 shows how Snort was initialized when the threshold value for failed login attempts had exceeded. It was invoked automatically by the shell script. After Snort was invoked, the rest of the login attempts whether valid or invalid were barred. In other words if once snort was invoked for a particular IP address then that IP address cant logon to the ftp server again.

Figure 4.14 shows how the metasploit looks like once it had reached its maximum limit of failed logon attempts:

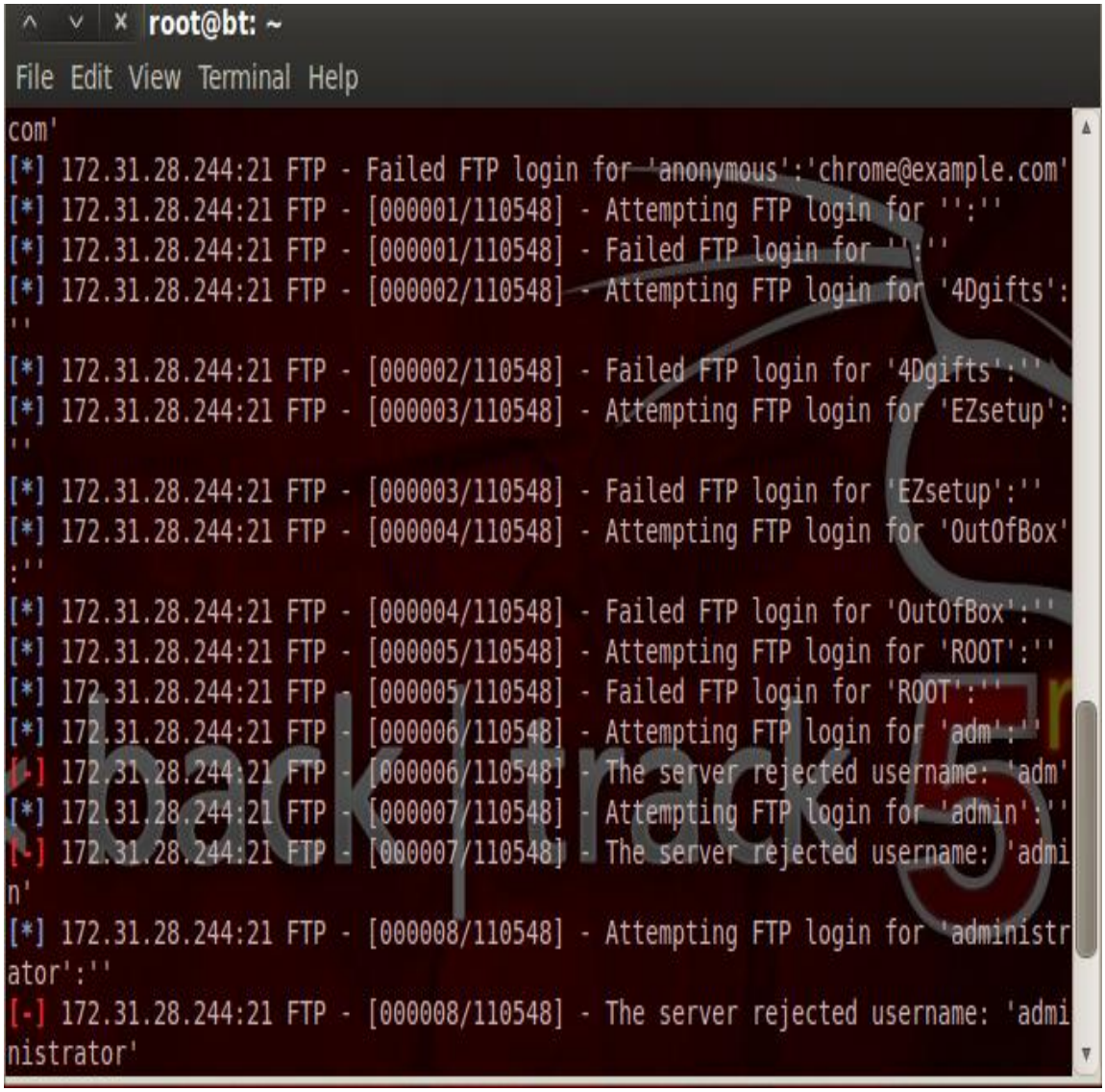

Fig4.14 Metasploit After Crossing Threshold Value

The '-' in red shows that the metasploit tried to enter the username to login but failed to do so as the server rejected the username.

4.5 Experiment 2

In this experiment the application level DOS detection capability of the stack was tested.

4.5.1 Methodology

This experiment targets testing the application level DOS threshold of the application, if the java program returns 13, it signals the ftp script about a application level DOS attack and the stack should then alert the administrator.

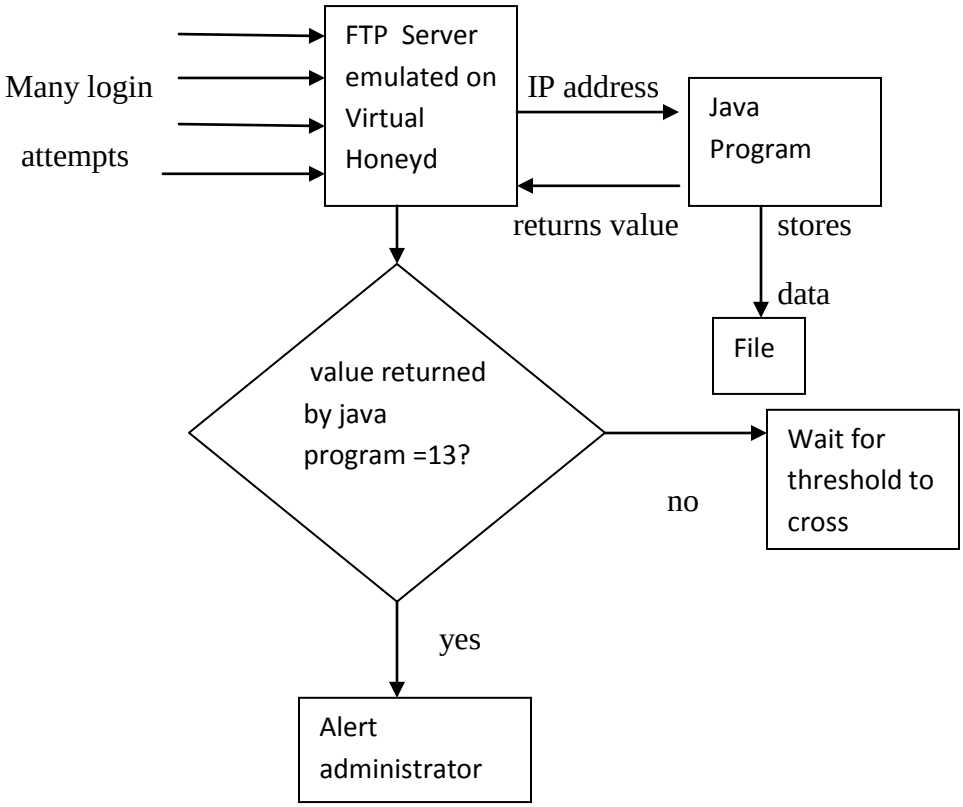

Fig 4.15 Flowchart for detection of Application level DOS attack

Figure 4.15 shows how an application level DOS attack is detected, after the number of attacks cross a certain threshold the shell script sends an alert message to the administrator about the impending DOS attack. The java program stores a time stamp with the attack and the number of such attempts, the threshold is set to a certain number of connections per minute, if the frequency exceeds this number then the attack is flagged and administrator is alerted.

4.5.2 Application level DOS attack and results

In this attack resources are depleted by the attacker so that intended users experience a huge delay in service. To consume the bandwidth of the server the attacker sends loads of requests to the server so that server deals with the fake requests and valid ones won't get service.

To prevent this from happening, a counter can be used which counts the number of requests for login coming from the same IP address in 1 minute or according to the requirement. Once the counter met the threshold value within the specified time, it should alert the administrator that an attack is trying to prevail the network. A java program is made which has a counter and an inbuilt function which is used to calculate the time.

Figure 4.16 shows the number of failed logon attempts tried within a minute. Once the threshold value for failed logon attempts within a specified amount of time is met, an alert is started to show on the screen which alerts the administrator about the attack and so that the administrator can take the necessary action to prevent the attack.

The java program again plays a pivotal role in this scenario, it detects the DOS attack with the help of timestamps and directs the shell scripts to take action.

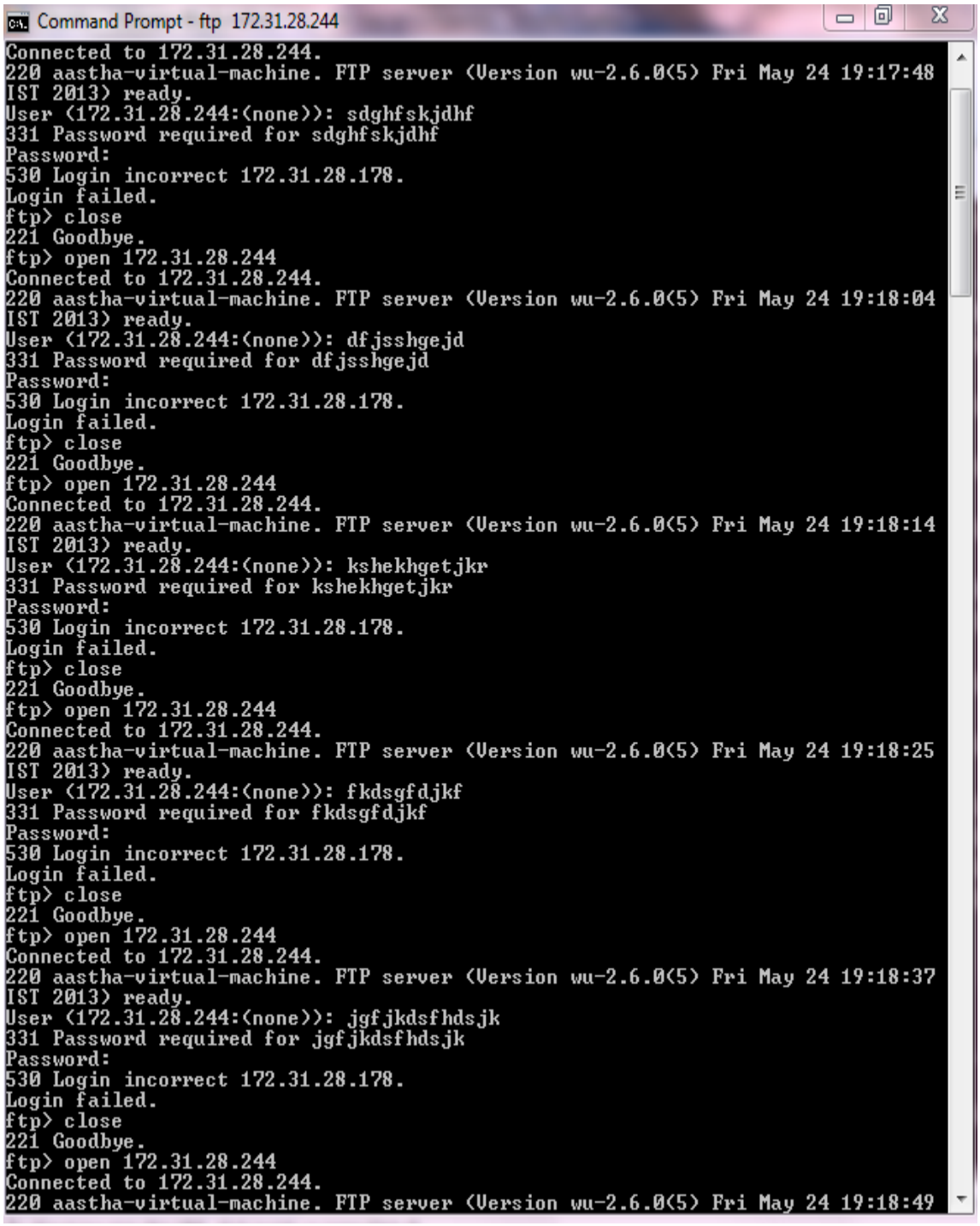

Fig4.16 Failed Login Attempts Within a Minute

After crossing the threshold value, a message should be displayed on screen to the administrator to alert her/him about the attack.

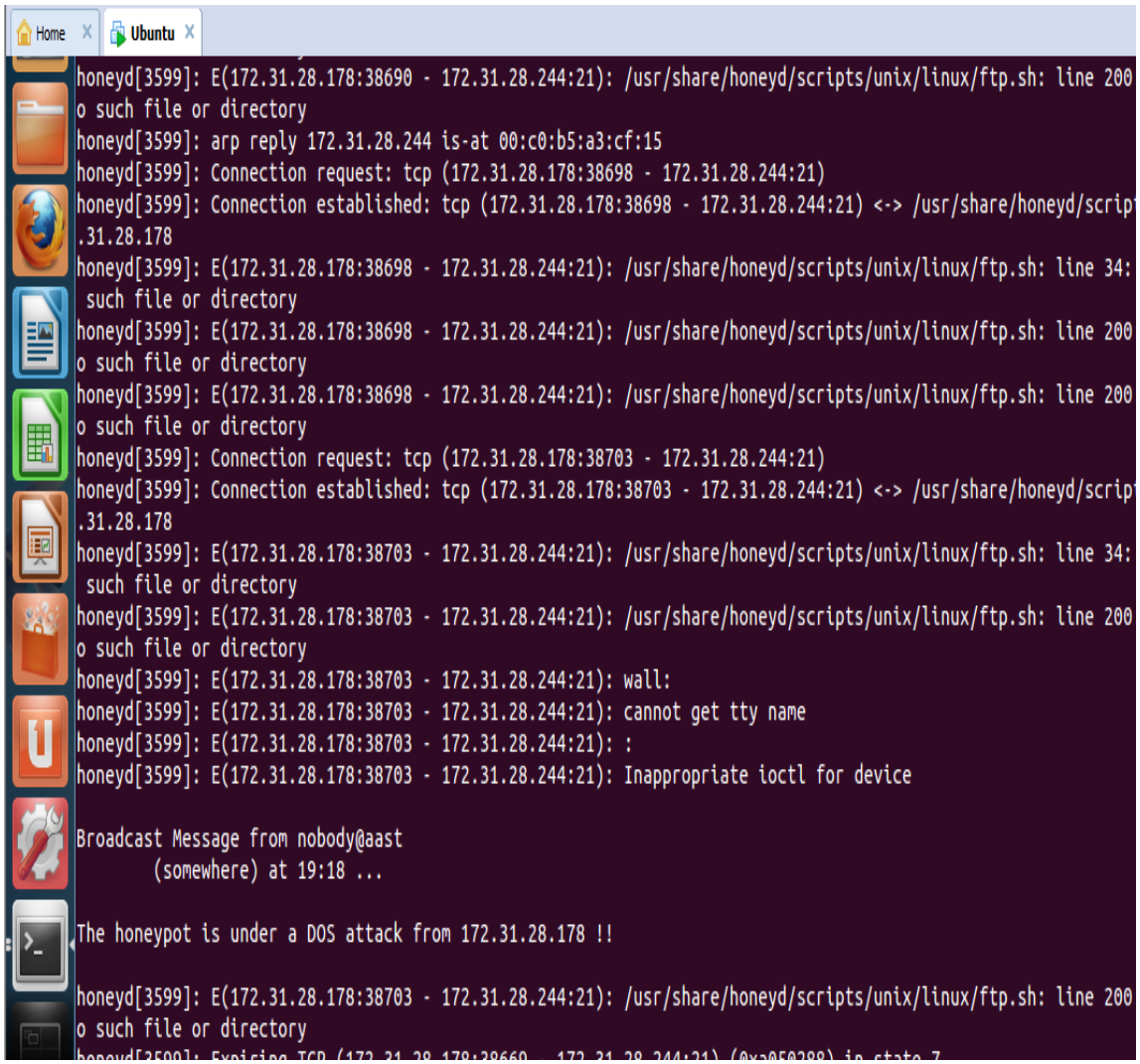

Fig 4.17 Alert to Administrator

Figure 4.17 shows the alert message being displayed on the screen to alert the administrator about the attack along with the IP address which was trying to attack the server.

4.6 prog.java

prog.java is a java program which works as a glue between honeyd and snort. All the ftp activity is recorded by this program. It stores the IP address, the last time this ip address tried to access the honeypot ftp and the number of login attempts made. If the number of login attempts is over a threshold value, it signals the shell to start Snort which in turn starts logging all the packets. If the number of login attempts is more than the threshold value, the java program returns a code which tells the shell script to broadcast a message to all users of the system.

Threshold value for brute-force attack =5

Threshold time for application level flood attack = 60 seconds

The following activities are done by prog.java

1. logs activity in a file called "ipstore".

2. writes snort rule

3. returns code to linux shell which help it perform the following function

- start snort
- send alert

4.6.1 Pseudocode

1. flag = 0

2. while(ipstore file has lines)

    1. Read ipstore file line by line.

    2. If IP address in current line == IPSRC then

        1. Read value of counter from the current line.

        2. Increment value of counter and set flag =1.

3. Store the value of time from the line in variable oldtime.

else

1. Append line to string buffer.

[endif]

[endwhile]

3. currtime = current system time.

4. if flag == 1 then

1. Append a new line to the string buffer with IPSRC, updated value of counter and currtime.

else

1. Append a new line with IPSRC, count =1 and currtime.

[endif]

5. Truncate the IPSTORE file and store the string buffer into it.

6. if counter > 5 then

1. If there is no snort file with same as IPSRC then

1. Store a new snort rule file under the rules directory.

2. return 10.

[endif]

2. If currtime - oldtime >= 60 then

1. return 13.

else

2. return 12.

[endif]

else

1. return 1.

[endif]

4.7 Chapter Summary

This chapter gives an in-depth look into how the application actually functions. It presents the various modules that make up the application and how they interact with each other. It also presents two experiments which validate the claimed functionality.

In the next chapter, the conclusion and future scope of the work done have been discussed.

CHAPTER 5

CONCLUSION AND FUTURE SCOPE

5.1 Conclusion

The research work carried out in this thesis is mainly focused to design and develop a network surveillance framework to solve FTP security problem which is known to be difficult to circumvent. The easiest means to detect an attack is to have a system which is dedicated to be attacked. Honeypots allow such systems to be built with ease, a honeypot in turn maybe compromised by the attacker to gain access to the system. But such a scenario will become useless if the honeypot were to run inside a sandbox. Virtual machines allow guest operating systems to be run on top of a host operating system just as any other application program.

Such a setup cannot be compromised easily as it adds two layers of abstraction, first at the level of the honeypot and the other at the hypervisor level. The proposed setup comprises of a virtual machine running the ubuntu operating system, inside it the honeyd honeypot was chosen to emulate an FTP server. Snort IDS was employed to save packets emanating from the FTP brute force attack and it was used to circumvent the attack. A program was written in java to bind the honeypot and snort IDS together. Without this program such a functionality is not easy to achieve.

Two experiments were conducted to cross reference the validity of the claimed functionality.

Though a significant amount of research has been done in the past to solve FTP attacks, but it still remains a challenging task due to increased complexity and various threat vectors peeping into the networks daily.

5.2 Future Scope

Though the design, development, deployment and testing of the network surveillance for solving ftp security problem has been successfully demonstrated but there have been some limitations in the research work carried out in the thesis.

One of the limitations, is only two attacks has been solved in the thesis. The thesis does not include the remedy for all the known Ftp attacks. Other limitation is this work has been carried out using low interaction honeypot which limits the amount of bait to lure the attackers.

In future this research effort can be put on to a level where all the left ftp attacks can be prevented and the honeypot implemented in this thesis could be extended rather easily to a full- fledged security measure against ftp server which can be deployed geographically at various heterogeneous locations. Hence, there is a lot of future scope of the research work to be carried out in this vital area of great significance to mankind.

REFERENCES

[1] Online Etymology Dictionary, http://www.etymonline.com/index.php?term=secure. Fetched 10/6/2013

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

[3] Kuwatly, Iyad, Sraj, Malek, Masri, Zaid Al, & Artail, Hassan, "A Dynamic Honeypot Design for Intrusion Detection" IEEE, 2004.

[4] Canetti, Ran, Gennaro, Rosario, Herzberg, Amir, & Naor, Dalit. "Proactive Security: Long-term protection against break-ins". http://u.cs.biu.ac.il/~herzbea/Papers/proactive/cryptobytes%20paper.pdf

[5] Scarfone, Karen, & Hoffman, Paul. "Guidelines on Firewalls and Firewall Policy". Recommendations of the National institute of Standards and Technology, September 2009. pp. 11-18.

[6] Newman, Robert C. "Intrusion Detection and Prevention" in Computer Security Protecting Digital Resources, Pennsylvania, 19 february 2009, ch. 10, pp.289

[7] Whitman, Michael E., & Mattord, Herbert J. "Intrusion Detection and Prevention Systems" in Principles of Information Security, Cengage learning EMEA, ch.7 pp 289.

[8] Zakaria, Wira, & Kia, Laiha. "A Review on Artificial Intelligence Techniques for Developing Intelligence Honeypot" in UMRG, pp. 696.

[9] Chavez, P., Veeraghattam, R., & Sung, A. "Network Based Detection of Virtual Environments and Low Interaction Honeypots" in Proceedings of the 2006 IEEE Workshop on Information Assurance, US Military Academy, West Point, NY, 2006.

[10] Honeypot Definition-PC Magazine. pcmag.com, 10 June 2013. Available: http://www.pcmag.com/encyclopedia_term/0,2542,t=honeypot&i=44335,00.asp

[11] Tabalis, Ryan. " Honeypots 101: A Honeypot by Any Other Name." 2007

[12] Cohen, Fred. "The Deception ToolKit." The Risks Digest 9 March 1998.

[13] Provos, Niels and Thorsten Holz. Virtual Honeypots: From Botnet Tracking to Intrusion Detection. Addison-Wesley Professional, 2007.

[14] Spitzner, Lance. Honeypots: Tracking Hackers. Addison-Wesley Professional, 2002. pp. 68–70.

[15] Honeynet Project, "Know Your Enemy: Honeynets", 24 March 2008. Available: http://old.honeynet.org/papers/honeynet/

[16] Antonatos, Spiros, Kostas Anagnostakis and Evangelos Markatos. "Honey@home: a new approach to large-scale threat monitoring." Proceedings of the 2007 ACM workshop on Recurring malcode. ACM, 2007, pp. 38 - 45.

[17] "Developments of the Honeyd Virtual honeypot", Available: http://www.honeyd.org

[18] Spitzner Lance, Honeypots: Tracking Hackers, Addison Wesley, 2002. pp. 46-50.

[19] Robert McGrew, Rayford B. Vaughn, Experiences with Honeypot Systems: Development, Deployment and Analysis, Proc. IEEEIntl. Conf. System Sciences,2006

[20] "SPECTER Intrusion Detection System", http://www.specter.com/default50.htm

[21] Vukasin Pejovi, Ivana Kovacevi, Slobodan Bojani, Corado Leita, Jelena Popovi, Octavio Nieto-Taladriz, Migrating a Honeypot to Hardware, Proc. IEEEIntl. Conf. Emerging Security Information, Systems and Technologies,2007, pp.35

[22] "Snort::Home Page", Available: http://www.snort.org

[23] Roesch Martin. "Snort- Lightweight Intrusion Detection for Networks". Proceedings of LISA: 13th Systems Administration Conference, Seattle, Washington, Nov. 7-12' 99.

[24] Rehman, Rafeeq Ur. "Intrusion detection with Snort". Prentice Hall, 2003. chapter 3, pp. 5.

[25] Gates, C., & Ceballos, M. "Oracle penetration testing using the metasploit framework". Proceedings of the BlackHat USA, 2009.

[26] Maynor, D., & Mookhey, K.K. "Metasploit toolkit for penetration testing, exploit dvelopment, and vulnerability assessment". Syngress publishing inc., 2007, pp. 2-4.

[27] Marques, Carlos Joshua. "An analysis of the IDS Penetration Tool: Metasploit" https://www.infosecwriters.com/text_resources/pdf/jmarquez_Metasploit.pdf

[28] Liben Joshua , Tomazeli Linker Araujo, & Georgescu Andrei. "Metasploit: A Study of the Software Architecture". Available: https://community.rapid7.com/thread/3126

[29] Provos, Neil. "Honeyd: A Virtual Honeypot Daemon". Available: www.citi.umich.edu/u/provos/papers/honeyd-eabstract.pdf