Hardening Linux Operating System to Mask against Fingerprinting

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

Master of Engineering
in
Software Engineering

By:
Ratinder Kaur
(80731020)

Under the Supervision of
Dr. Maninder Singh
Associate Professor
Computer Science and Engineering Department

COMPUTER SCIENCE AND ENGINEERING DEPARTMENT
THAPAR UNIVERSITY
PATIALA - 147004

July 2009

Abstract

The Internet has become one of the most powerful and widely available communication medium on earth. Governments, corporations, banks, and schools have become more dependent on Internet systems for their proper functioning. The information and computer systems that are the critical assets, supporting the mission of an organization must be treasured and properly protected. Computers themselves do not cause any threat to individual security but when connected with networks, computers have become the tools that invite thieves and hackers. These cyber criminals have grown increasingly more sophisticated in executing their many methods of deception because of boom in the technology. Also, due to digital convergence, networked computers have become omnipresent and thus can be easily subjected to attack, misuse, and abuse. It is not uncommon to see articles in the media referring to Internet intruder activities.

The Internet is easy and cheap to access, but the systems attached to it lack a corresponding ease of administration. As a result, many Internet systems are not securely configured. Even the general public devices (like switches and routers) are put in use without tailoring their default configuration, which in turn can be very dangerous. Additionally the underlying network protocols that support Internet communication are insecure, and few applications make use of the limited security protections that are currently available. Thus, the combination of the data available on the network and the difficulties involved in protecting the data securely make Internet systems vulnerable attack targets.

The very basic step that led to vulnerability is called Operating System Fingerprinting. It is the process of identifying a remote operating system based on the characteristics of its TCP/IP network stack. Various automated tools are available to reveal remote operating system with great accuracy in minimum amount of time. The running operating system on any target machine is the key information for attackers. As long as this information is not revealed, the attacker is limited in the variety of attacks, probes and exploits. Therefore, hiding the operating system from the attacker is extremely important.

Masking the operating system can protect the system from easy detection. And one such masking technique is to harden the system. In this thesis, hardening

of the system is achieved by designing and developing a shell script module and kernel hooks, that minimizes the system's exposure to various computer threats by disabling non essential utilities and administration options that might unlock the door for the hackers to make a way into the system. The idea behind this approach is to increase system security by proactively configuring the Operating System and decreasing its susceptibility to be easily compromised.

Certificate

I hereby certify that the work which is being presented in the thesis entitled, **"Hardening LINUX Operating System to Mask against Fingerprinting"**, in partial fulfillment of the requirements for the award of degree of Master of Engineering in Software Engineering and submitted to Computer Science and Engineering Department of Thapar University, Patiala, is an authentic record of my research work carried out under the supervision of *Dr. Maninder Singh* and refers other researcher's works which are duly listed in the reference section.

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

(Ratinder Kaur)

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

(Maninder Singh)
Computer Science and Engineering Department
Thapar University
Patiala

Countersigned by

(RAJESH BHATIA)
Assistant Professor & Head
Computer Science and Engineering
Department
Thapar University
Patiala

(R.K.SHARMA)
Dean, (Academic Affairs)
Thapar University
Patiala

Acknowledgment

I would like to express my deep sense of gratitude to my supervisor, **Dr. Maninder Singh**, Associate Professor, Computer Science and Engineering Department, Thapar University, Patiala, for his invaluable help and guidance during the course of thesis. I am highly indebted to him for constantly encouraging me by giving his critics on my work. I am grateful to him for giving me the support and the confidence that helped me a lot in carrying out the research work in the present form. And for me, its an honor to work under him.

I also take the opportunity to thank **Dr. Rajesh Bhatia**, Assistant Professor & Head, Computer Science and Engineering Department, Thapar University, Patiala, for providing us with the adequate infrastructure in carrying the research work.

I would also like to thank my parents and friends for their inspiration and ever encouraging moral support, which went a long way in successful completion of my Thesis.

Above all, I would like to thank the Almighty God for His blessings and for driving me with faith, hope and courage in the thinnest of the times.

Ratinder Kaur
(80731020)

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Network Security has become a major concern for every network administrator. The Internet has accelerated at an amazing rate, the pace at which information is disseminated. With the growth of Internet, the things are becoming more and more insecure. Earlier attackers simply wanted to prove that they could break into systems but nowadays, the intrusions are prompted by financial, political, and military objectives. Today, anyone with little technical know-how can easily hack a system by using readily available intrusion tools and exploit scripts that capitalize on widely known vulnerabilities.

This chapter explains what exactly network security is and its importance in today's insecure environment.

1.1 Network Security

The current state of network is vulnerable as they are prone to increasing number of attacks. These attacks are very hard to detect because they are not seen previously before any subsequent damage is done. Thus securing a network from unwanted malicious traffic is of prime concern [11]. There is a myth surrounding network security, that, securing a network means defending against worms or viruses, or preventing access to the network by unauthorized users or protecting the privacy of network information and resources. But in actual, network security is all these things, and much more. The network administrator must define the characteristics and conditions of a secure network to address the threats both from outside and inside the organization. Thus, network security is a process, not a product or the latest patch [14].

Network Security involves all the activities that organizations, enterprizes, and

institutions must undertake to protect their private network from the hackers ,that may gain access to a data of vital significance, perhaps resulting in data loss, or even complete destruction of the system. Network security comprises the measures that protect usability, reliability, integrity and safety of the network and data. An effective network security is achieved by targeting various threats and stopping them from entering the network by choosing the most effective set of tools to combat them. No single tool is appropriate to provide complete security. A layered approach is beneficial. If one layer of the security fails, others are still up to provide protection. The more layers the user has to go through to access the desired network, the more secure the network is [3] [33].

1.1.1 A Taxonomy of Security

Security is associated with four core areas, which can be conveniently summarized by the acronym "CIAA". Any hacking event will affect any one or more of the essential areas.

- *Confidentiality:*Ensures that information is not accessed by unauthorized persons i.e. the data is only revealed to parties who have a legitimate need.
- *Integrity:*Ensuring trustworthiness of data or resources in terms of preventing improper and unauthorized changes. It is possible that as a file, electronic mail, or data is transmitted from one location to another, its integrity may be compromised.
- *Authenticity:* Ensuring that a sender and a receiver of information are the persons they claim to be. This means having the capability to determine who sent the message and from where and which machine.
- *Availability:* Ensuring that the data or computing resources needed by appropriate personnel are both reliable and available in a timely manner [20].

1.2 Importance of Network Security

It is remarkably easy to gain unauthorized access to information in an insecure networked environment. Even if users have nothing stored on their computer that they consider important, that computer can be a "weak link", allowing unauthorized access to the organization's systems and information [18]. Thus, Network Security is very essential for any organization.

Without security measures, a user can easily access another user's machine by

using well-known exploits, trust relationships and default settings. Most of the attacks require little or no skill, putting the integrity of a network at stake. On the other hand, with no security at all, internal users can be a major threat to many corporate internal networks because the user within the company already has access to many internal resources and does not need to bypass firewalls or other security mechanisms. Such internal users, equipped with hacking skills, can successfully penetrate and achieve remote administrative network rights while ensuring that their abuse is hard to identify or even detect.

If an external hacker breaks into a computer network, he/she can then access the rest of the internal network more easily. This would enable a sophisticated attacker to read and possibly leak confidential emails and documents; trash computers, leading to loss of information; and more. Or may use the compromised network and network resources to attack other sites, that when discovered will lead back to the hacked organization and not the hacker.

1.2.1 Increasing Threat of Attackers

Earlier an attacker had to use sophisticated computer, programming, and networking knowledge to make use of rudimentary tools for basic attacks. But as time went on, network attack tools and methods have evolved a lot. The attackers now no longer require the same level of sophisticated knowledge. Technology has made hacking a trivial job due to:-

- *Ease of use:* Today the technology is evolving focusing on the ease of use, so tools used to penetrate networks and systems have become simpler, requiring a relatively small amount of technical knowledge to use. Now it is easier to create more sophisticated exploit code with the help of toolkits and libraries that have been created to allow attackers to develop malicious code quickly and user-friendly. And the auto-update functionality in certain attacks, allows the attackers to download the attack code at the last minute.

- *Increasing Complexity:* Due to increasing complexity of computer infrastructure it is difficult to administer and manage i.e. it is hard to keep a check on each and everything.

- *Skill Level:* The skill level to exploit a system is decreasing. Even less knowledgeable attackers can construct and launch attacks.

- *Online Businesses:* Today almost every business is conducted on the internet, so due to increased network environment and network based applica-

tions things are more exposed to outer world, and thus are more vulnerable to attack.

1.3 Approaches to Network Security

There are two approaches to implement security measures, namely detection based, and prevention-based.

1.3.1 Detection Based

The first step in securing a network system is to detect the attack. The Intrusion Detection (ID) can be considered to be the first line of defense for any security system. Even if the system cannot prevent the intruder from getting into the system, noticing the intrusion will provide the security officer with valuable information [15]. Detection based systems are passive in response to network based attacks; because they are generally limited to logging, notification and disconnection at local host. They neither provide response to the needed real time attacks nor do they scale with network [7].

1.3.2 Prevention Based

Prevention based systems do packet filtering as compared with the filtering rules that are set up at security policies. Allowable applications use policy, protocol validity check and an attack and a module protective vulnerability Signature check. Prevention based system like, Stateful Inspection Firewall cuts off malicious network traffic, and along with Deep Packet Inspection technology carries out verification of an IP Fragment, TCP Segment and Acceptable Application Use through a pattern matching function [21].

Whatever may be the method being used to implement the network security measures, it is necessary to think network security as a complete methodological process rather than just any technique or product.

Chapter 2

Literature Survey

The critical infrastructures upon which our communities, states, nation rely are increasingly dependent on computer systems and networks and are thus also increasingly vulnerable to cyber attacks upon them. Because, most organizations (as well as individual users) regard security as an afterthought, a process that is overlooked in favor of increased power, productivity, and budgetary concerns. Proper security implementation is often enacted "postmortem" after an unauthorized intrusion has already occurred [37] [12]. Several computer security survey reports shows that the network security incidents are widespread and occur frequently. Even companies whose networks are not connected to the Internet are at risk due to unauthorized access by disgruntled employees.

2.1 Network Security Incidents

A network security incident is any network related activity that violates an explicit or implicit security policy. The consequences of an incident cover a broad range of possibilities. It can be a minor attack involving a minor loss of time in recovering from a problem, a decrease in productivity, a significant loss of money or staff-hours or a major attack involving devastating loss of credibility or market opportunity, legal liability and the loss of life [18].

The security incidents can be active or passive. Attacks that results in information release are passive, and those that involve message modification or denial of resources are active. These incidents are broadly classified in several kinds:-

The Probe (passive):A probe is an attempt to gain access or to discover information about the system through a known or probable weak point in that system.

Scan (passive):A large number of probes done using automated tool is scanning. It is done for identifying active hosts on a network for the purpose of attacking.

Authorization Violation (active):Authorization Violation is when an entity uses a service or resources it is not intended to use, and this might expose the victim to serious data loss or data theft.

Traffic Observation (passive):Often called eavesdropping, traffic observation concerns the interception of information between communicating parties, thereby resulting in the disclosure of information such as traffic type, content, frequency, presence/absence, amount [29].

Denial of Service (active):DoS attacks are probably the nastiest, and most difficult to address because they are very easy to launch and difficult to track [5]. Large number of bogus packets are sent with the intent of slowing or interrupting offered services. The goal of DoS is not to gain unauthorized access to machine or data, but to prevent legitimate users of a service from using it. DoS attacks availability.

Masquerading (active):Often called spoofing, masquerading concerns the issuance of information, the receipt of information, or the acquiral of access rights, by using an identity other than its own [29].

Malicious code (active):Malicious code is a general term used for the programs that, on execution, would perform undesirable functions on the system. The presence of malicious code is not detected until they do some sort of damage. Malicious code includes Trojans, viruses, and worms. Trojans are usually hidden in other legitimate programs and appears to perform a desirable and necessary function but performs functions unknown, whereas, worm is a self-replicating, self-propagating program and do not need to be associated with any particular program. A virus is also self-replicating, but needs to be associated with a particular program. These sorts of programs can lead to serious data loss, downtime, denial of service, and other type of security incidents.

Internet infrastructure attacks (active):These attacks are rare but serious as they target key components of the Internet infrastructure rather than specific system on the Internet. Key components like routers, network name servers, network access providers and large archive sites [18]. Example of such infrastructure

attacks are:-

- *Router Attacks:*Intruders understand router administration and modifies router configurations.
- *Route Name Server Attacks:*Intruders do not exploit any new vulnerability but target the infrastructure.
- *Internet Service Attacks:*In FTP archives, Trojan horse programs are attached and hidden [9].

These network security incidents occur when the security of the system is compromised by exploiting the vulnerability of that system. Vulnerability can be an existence of any weakness in the system design or an implementation error that leads to an unexpected, undesirable event compromising the security of the system. Vulnerability can be exploited to accomplish something that is not authorized as legitimate use of a network or system. Figure 2.1 depicts attackers perspective.

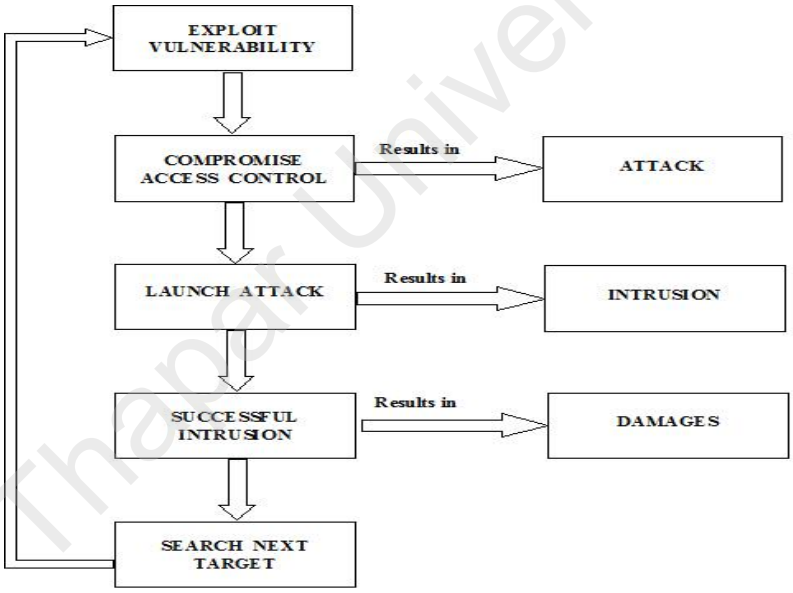

Figure 2.1: Attackers Perspective

To exploit a service an attacker may utilize a service in two ways. First, the service, as documented and intended to operate, may contain security holes and weaknesses. These may be compounded by poor operating practices of users and system administrators, e.g., poor choice of passwords. Second, due to bugs and trapdoors, the implementation of the service may allow attackers to use the service in ways not intended by the designers, e.g., in some Operating Systems, hitting an interrupt character before the login authentication allows a login without password or in some cases may even crash the system [13].

2.2 Software Vulnerabilities

Software vulnerability is a "bug" i.e. a flaw/weakness/defect in the code of any program (e.g. application, operating system) that can be exploited by malicious agents. Studies at Carnegie Mellon University (CMU) find that there are between five and fifteen 'bugs' per one thousand lines of code in released software. Vulnerabilities may arise from **boundary condition error**, commonly known as "buffer overflow" that may result due to improper bound sizes of buffers, arrays, and strings or due to incapability of program to handle **exceptional conditions** or because of **input validation error** that occurs when parameters and arguments don't fit the program or function specifications and are not checked for compliance or may be due to **configuration error** for the desired level of security and also when the **program loses its integrity** and behaves maliciously like email viruses and worms [17].

2.2.1 Life Cycle of Vulnerabilities

- Before the vulnerability is discovered
- After the vulnerability is discovered but before it is announced
- After the vulnerability is announced
- An automatic attack tool to exploit the vulnerability is published and
- The vendor issues a patch that corrects the vulnerability [24].

Vulnerabilities can be discovered, through testing by producers (in-house testing) before the product is released, after the product is in production and available to consumers, from full-disclosure environments, such as mailing lists, newsgroups or any kind of forum where individuals or organizations post information about vulnerabilities, and when it is exploited. Hackers try to use the limited time to launch attacks between vulnerability disclosures and patch availability. And, it has been known that the time between the disclosure of a new vulnerability and the emergence of an exploit is shrinking with time. Today, 80 percent of exploits become available within 60 days of a vulnerability being released, with an increasing number of exploits created in fewer than six days. This trend raises the concern about emerging "zero-day attacks," where networks will have to defend themselves against new exploits that come with no advance warning [35].

2.3 TCP/IP Vulnerabilities

First start with essential TCP/IP background knowledge, and then move onto various vulnerabilities in TCP/IP.

TCP/IP
TCP and IP were developed by the United States Department of Defense (DOD) Advanced Research Projects Agency (ARPA) in 1969 for a research project to connect a number different networks designed by different vendors into a network of networks (the "Internet"). It was initially successful because it delivered a few basic services that everyone needs (file transfer, electronic mail, remote logon) across a very large number of client and server systems. The IP component provides routing from the department to the enterprise network, then to regional networks, and finally to the global Internet. As with all other communications protocol, TCP/IP is composed of following layers:-

IP is responsible for moving packet of data from node to node. IP forwards each packet based on a four byte destination address (the IP number). The Internet authorities assign ranges of numbers to different organizations. The organizations assign groups of their numbers to departments. IP operates on gateway machines that move data from department to organization to region and then around the world.

TCP is responsible for verifying the correct delivery of data from client to server. Data can be lost in the intermediate network. TCP adds support to detect errors or lost data and to trigger retransmission until the data is correctly and completely received.

2.3.1 TCP/IP Protocol Suite

The TCP/IP protocol suite, also referred to as the Internet protocol suite, is the set of communications protocols that implements the protocol stack on which the Internet and most commercial networks run. It is named after the two most important protocols in the suite: the Transmission Control Protocol (TCP) and the Internet Protocol (IP).

The TCP/IP protocol suite-like the OSI reference model-is defined as a set of layers. Upper layers are logically closer to the user and deal with more abstract data, relying on lower layer protocols to translate data into forms that are transmitted physically over the network. TCP/IP applications use 4 layers:-

Application Layer

The application layer is the topmost level of TCP/IP suite. It receives data from user applications and issues requests to the transport layer. The details of moving data between the application and other computers are shielded by the underlying layers. As the TCP/IP application layer maps to the OSI application, presentation and session layers, it is also responsible for details such as character formats (For example, ASCII vs. EBCDIC) and basic encryption. Some well known examples of application level entities within the TCP/IP domain are:-

- FTP/Telnet/SSH
- HTTP/Secure HTTP(SHTTP)
- POP3/SMTP
- SNMP

Transport Layer

The transport layer of the TCP/IP model maps fairly closely to the transport layer of the OSI model. Two commonly used transport layer entities are TCP and User Datagram Protocol (UDP).TCP and UDP are responsible for delivery of a message from a process to another process.

Internet Layer

The Internet layer of the TCP/IP model maps to the network layer of the OSI model. Consequently, the Internet layer is sometimes referred to as the network layer. The primary component of the Internet layer is the Internet Protocol (IP), which in turn uses four supporting protocols: Address Resolution Protocol (ARP), Reverse Address Resolution Protocol (RARP), Internet Control Message Protocol (ICMP) and Internet Group Message Protocol (IGMP).

Network Access Layer

The lowest layer of the TCP/IP protocol stack is the network access layer. The network access layer contains two sublayers, the Media Access Control (MAC) sublayer and the physical sublayer. The MAC sublayer aligns closely with the data link layer of the OSI model, and is sometimes referred to by that name. The physical sublayer aligns with the physical layer of the OSI model. Examples of the network access layer include:-

- Ethernet
- Wireless Fidelity (Wi-Fi)/WiMAX)

- PPP, PPP over Ethernet (PPPoE)
- ATM/Frame Relay

2.3.2 TCP/IP Protocols

IP

The Internet Protocol is the delivery mechanism used by the TCP/IP protocols. IP is a best-effort delivery service i.e. it provides no error control or flow control (except for error detection on the header).Thus, its unreliable datagram protocol. IP is a connectionless protocol, which means that there is no continuing connection between the end points that are communicating. Each packet that travels through the Internet is treated as an independent unit of data without any relation to any other unit of data. For reliability, IP is paired with a reliable protocol such as TCP. Below is Figure 2.2 showing IP header.

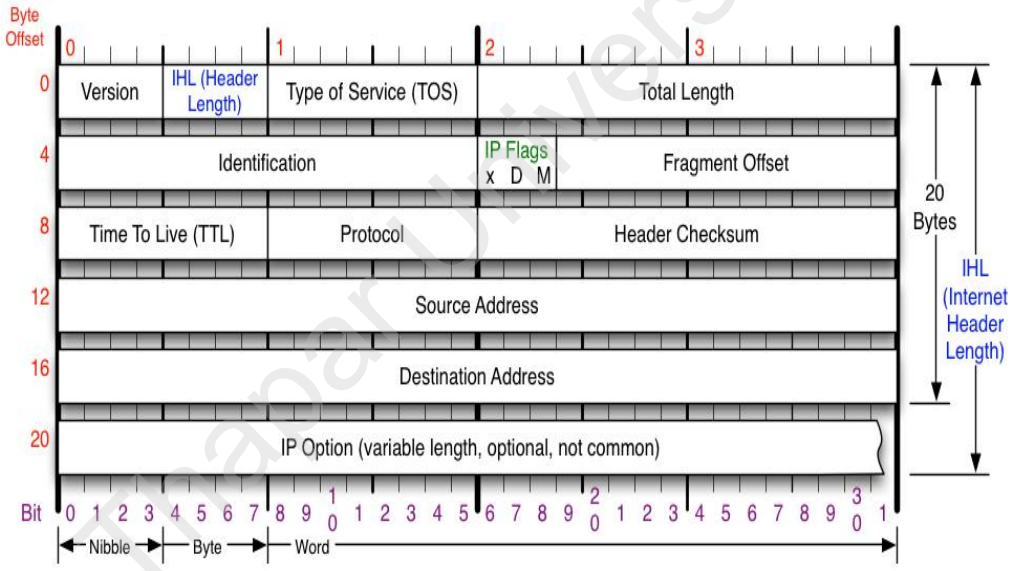

Figure 2.2: IP Header

ICMP

The Internet Control Message Protocol (ICMP) is a required protocol tightly integrated with IP. ICMP messages are sent in several situations: for example, when a datagram cannot reach its destination, when the gateway does not have the buffering capacity to forward a datagram, and when the gateway can direct the host to send traffic on a shorter route. The purpose of these control messages is to provide feedback about problems in the communication environment. Of course, since ICMP uses IP, ICMP packet delivery is unreliable, so hosts can't count on

receiving ICMP packets for any network problem. ICMP is usually not used directly by user network applications, with some notable exceptions being the ping tool and traceroute. Below is the Figure 2.3 showing ICMP header.

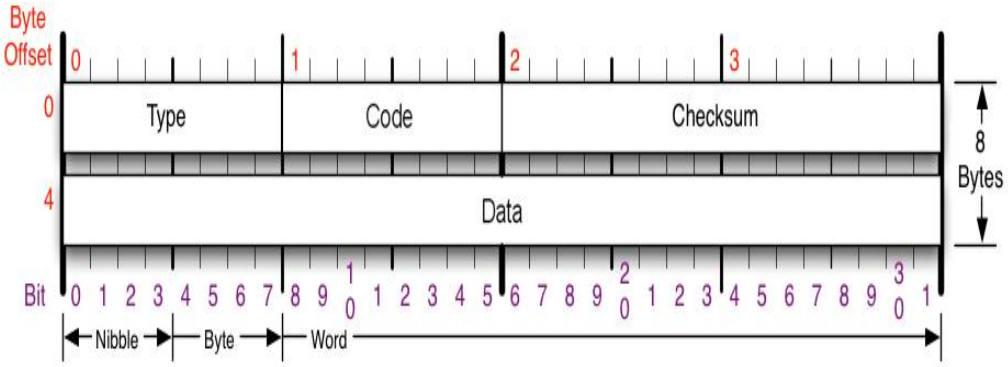

Figure 2.3: ICMP Header

TCP

The Transmission Control Protocol is a connection-oriented communication protocol i.e. the user data is not exchanged between TCP peers until a connection is established between the two end points. It defines how to set up and tear down a connection between the two peer layers through a standard exchange of SYN, ACK and FIN commands. TCP views data as a stream of bytes and data is transmitted in segments. TCP specifies a sliding window mechanism to implement flow control and sequencing of packets to guarantee no data loss during the transmission. It also provides various ways of congestion control. All these operations are managed by TCP header shown below. Figure 2.4 shows TCP header.

UDP

The User Datagram Protocol is very simple. The message unit used by UDP is called a datagram. Datagrams are considered unreliable, in that there is no guarantee datagrams will be received in the correct order, if at all. While UDP is unreliable, the lack of error checking and correction make UDP fast and efficient for many less data intensive or time-sensitive applications, such as the Domain Name Service (DNS), the Simple Network Management Protocol (SNMP), the Dynamic Host Configuration Protocol (DHCP) and the Routing Information Protocol (RIP). UDP is also well suited for streaming video [31] [8]. Figure 2.5 shows UDP header.

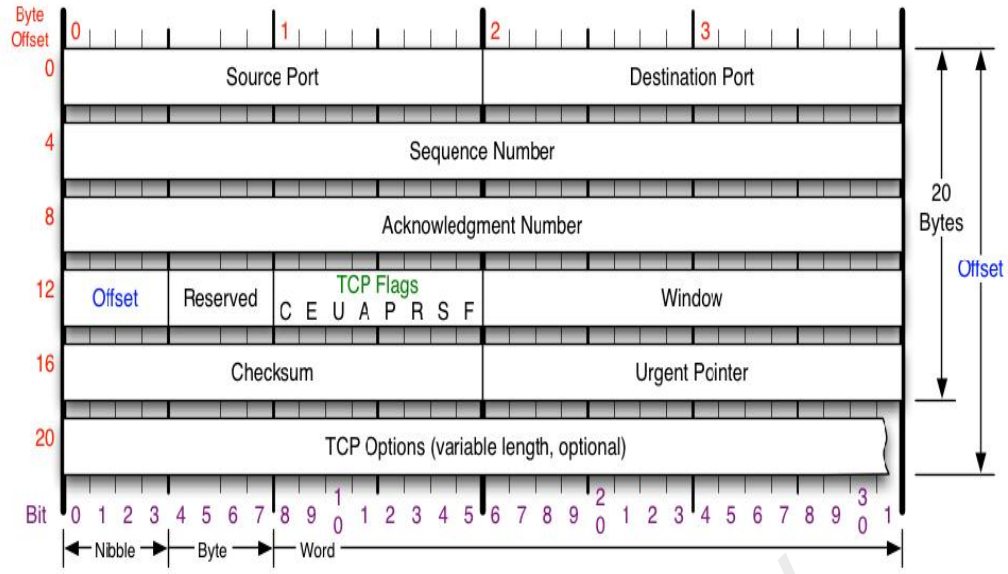

Figure 2.4: TCP Header

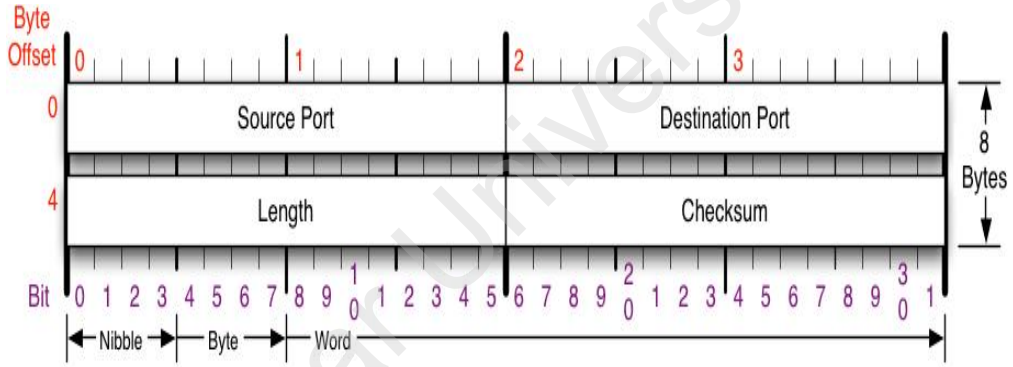

Figure 2.5: UDP Header

2.3.3 TCP/IP Weaknesses

In today's Internet, TCP/IP suit is used for communications. It enabled millions of computers to communicate globally. Since the first implementation of the TCP/IP suit, the users have suffered from the lack of secure data transfer via the Internet [1]. TCP/IP has a large number of serious security flaws inherent in the protocols, regardless of the correctness of any implementations. Some attacks based on TCP/IP flaws are discussed below.

Sniffing (Passive Attack) - Packet sniffing is act of intercepting and reading network traffic that is transmitted across a shared network communication channel. Protocols vulnerable to sniffing are:-

- *Telnet and Rlogin*: Sniffing can capture the keystrokes as the user types in,

including the username and password.

- *HTTP*: Many web sites use "Basic" authentication, which sends username and password across the wire in plain-text. Also data is sent in clear-text.
- *SNMP*: All SNMP traffic in SNMPv1 has no good security. SNMP passwords are sent across the wire in the clear-text.
- *NNTP*: Password and data is sent in clear.
- *POP*: Password and data is sent in clear.
- *FTP*: Password and data is sent in clear.
- *IMAP*: Password and data is sent in clear.

IP spoofing - To spoof a packet is a trivial job. In this, an attacker fakes or "forges" a packet to look like it that came from a "trusted" host. This is possible because IPv4 does not check the validity of the source address and source port in a packet's header. This is the main vulnerability in TCP/IP suite. IP spoofing can be used to cause DoS, or to gain access to a system as a "trusted" host.

SYN Flooding - To establish a connection TCP/IP does 3-way handshaking. A host sends an initial SYN packet to another host that they want to form a connection with. The server host then replies with the SYN/ACK packet. Till this time half connection is established between two, and the connection request is placed onto kernel's TCP/IP stack until it receives the final ACK from client host. Now if a lot of connection requests were made it is possible to saturate a host's resource via SYN flooding attack. This is major flaw due to bad design in TCP/IP.

Sequence Number Prediction Attacks - This is the ultimate attack in IP spoofing, to gain a connection with a host while pretending to be another "trusted" host. All that is required is to predict the sequence number of server host's SYN/ACK packet after sending a SYN packet. Old Operating Systems use simple set of rules to generate sequence numbers. This is known as 64k technique:

- Increment the Sequence Number counter EVERY second by 128000.
- Every time a new connection is formed, increment the Sequence Number counter by 64000.

Some Operating Systems use time incrementation technique, in this the sequence number is incremented by 'x unit_of_time' i.e. counter is increased by 'x' every

'unit_of_time'. These techniques are simple and easy to break. But modern Operating systems use random number generators to generate Sequence Numbers; this makes Sequence Numbers hard to guess [25] [6].

The software vulnerabilities are specific to a software's identification i.e. its version. So the first basic step to hack a system is to find the correct version of the target system. This process of identifying the identity of remote operating system is known as Operating System Fingerprinting. And every Operating System has a unique fingerprint according to its protocol stack implementation that's why, this process of finding remote host is also known as TCP/IP Stack Fingerprinting.

2.4 TCP/IP Stack Fingerprinting

TCP/IP stack fingerprinting is a method of detecting the remote host's operating system using the information leaked by that host's TCP stack. Due to differences in the way developers implement networking stacks, typically, unique identifiers within the packets transmitted by a host will allow for comparison based on known signatures [4]. Three approaches that are used to find running operating system of an unknown host are:-

2.4.1 Classical Fingerprinting

Even without resorting to stealth techniques of any kind, hosts will often announce their operating system to anyone making a connection to them through welcome banners or header information. For example, when connecting to a host via the standard Telnet protocol the operating system version is often sent to the client as part of a welcome message.

- File Transfer Protocol (FTP) will also provide this information as a welcome banner.
- If the Simple Network Management Protocol (SNMP) service is present on the machine, it is often left with the default 'public' community name, which will allow remote detailed querying of the service and hence host.
- Other services that send back 'free' useful information include Internet Message Access Protocol (IMAP), Post Office Protocol version 2 (POP2), Post Office Protocol version 3(POP3),Simple Mail Transfer Protocol(SMTP), Secure Shell (SSH) and Network News Transport Protocol (NNTP).

- A slightly more subtle approach is to use the anonymous ftp account (if supported), download a public binary required for ftp to function and examine it to determine what platform it was built for.

- A more primitive approach is to port scan the machine using any of the common port scanners freely available (eg Nessus) and examine the returned list of listening ports for patterns common to a particular operating system. Note that this approach is only effective against less secure hosts; particularly earlier Windows servers and those that do not implement accepted basic computer security concepts.

- Finally, it may be possible to determine the operating system of a system by a nontechnical solution, such as "social engineering". Learning about the target through phone calls, chatting to the System Administrator, or even a public site tour are all possibilities.

2.4.2 Active Fingerprinting

This is the predominant form of operating system fingerprinting, provoking the target into eliciting a response and analyzing it carefully.

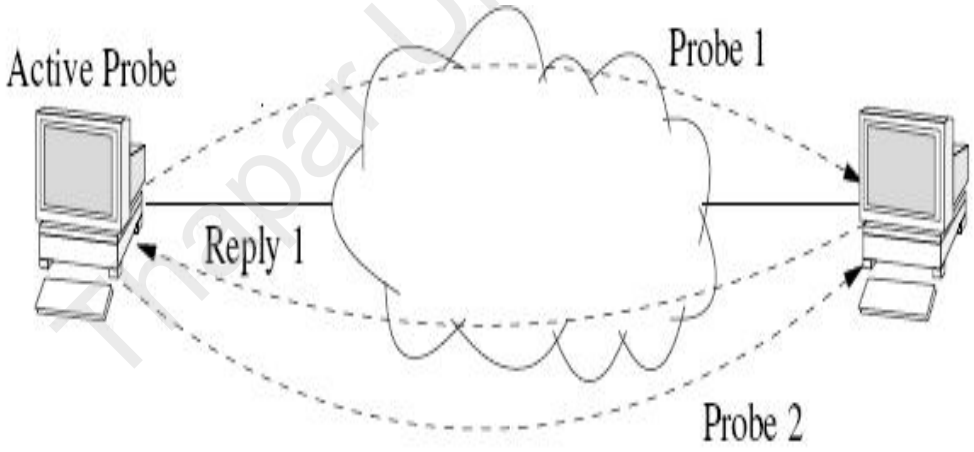

Figure 2.6: Active Fingerprinting

A huge amount of information can be gleaned about the response to a carefully crafted network packet. Figure 2.6 shows that probes are sent to the target host to grab the information about its Operating System on the basis of replies that it sends back in response to that probes. The three common IP packet types (ICMP, TCP, UDP) are all used in this technique and various valid (and invalid) packets

are sent to the host to refine the guess of the operating system. The most common techniques in use include the following:-

- *FIN Probing:* A single packet is sent to a known open port with the FIN flag set. This flag usually signals the end of a communication and as such is not expected without a connection being previously established. The standard behaviour is to simply ignore the packet; however, many stacks send a RST packet back. This difference is the means to begin creating a fingerprint.

- *TCP ISN Sampling:*TCP uses sequence numbers to keep track of the number of bytes successfully transferred across a connection. When an initial connection attempt is made to a host the operating system chooses an initial sequence number to begin the process. This choice can be anything from a constant value, through random increments of previous values, algorithms based on the host's internal clock, to true random systems. It is important to note that the predictability of this ISN also has security implications, leaving the host open to attacks similar to the Mitnick attack.

- *ICMP Error Quoting:*Various aspects of the structure of ICMP error packets are useful. ICMP error packets are required to return a small portion of the original message for identification purposes; however, some stack implementations return more than expected. This is especially useful as it allows some basic operating system identification of machines that no listening ports open at all.

- *ICMP Error Message Echo Integrity:* ICMP error message packets are required to include some of the original ICMP packet that caused the error. This makes it simple for implementations to use a copy of the original as a template for constructing the reply packet. Using the packet space as a 'scratch' area can leave telltale spurious data that identifies the operating system that created it.

- *ICMP Error Message Type of Service (TOS):* Nearly all implementations return a zero in the TOS field for ICMP port unreachable messages. Linux, however, currently returns a different value in this field making it easy to broadly identify.

- *TCP Options:* These are enhancements to the TCP protocol to improve performance in unreliable or high latency networks. TCP Options have been added as TCP RFCs over time as needs dictated and the patterns of compliance in responses can reveal the underlying operating system. Interestingly it is not just the number of options a stack supports that can identify it, but also the order in which the options are returned. Many other differences also exist and are used to a lesser extent. These include:-

  - IPID Sampling - Many operating systems utilise a system wide counter for IP Identification Number (IPID) generation. Other, more advanced implementations either randomise this number or set it to 0. Information can be gleaned from the choice of IPID as to the source operating system.

  - TCP Timestamp - This is a TCP option used to determine operating system type. It can also be used to determine host uptime if implemented and the update frequency are known.

  - ICMP Error Message Limiting - RFC 1812 suggests limiting the rate at which ICMP error messages are sent. Some IP stacks implement this suggestion (including Linux, Solaris) whereas Windows hosts do not. This technique is only viable over reliable connections to the remote host and extends scanning time; so, is generally not implemented.

  - Fragmentation Handling - Fragmented packets, particularly the handling of packets that contain overlapping fragments, are another source of anomalies. Some stacks retain the first version of the overlapped data, and some the second.

Nearly all active fingerprinting tools use some or all of the above tests to obtain data on a host and compare the results with a database of known operating systems [32].The problems with active scanning are mainly twofold: firstly, the packets can be readily firewalled that are used to fingerprint the system i.e. obfuscating the information; secondly, it can be detected quite easily. Because of this, it is less attractive for a truly stealthy adversary.

2.4.3 Passive Fingerprinting

Passive fingerprinting is a method of identifying the operating system, services, and applications of a remote host by using nothing more than sniffer traces. There is no risk of being detected while probing the remote host. Passive fingerprinting attempts to determine the host type by passively monitoring a network link, and not sending any traffic onto the wire. Existing passive operating system fingerprinting tools examine the values of fields in the IP and TCP headers of initial TCP SYN segments sent from clients. Common fields used are:-

- Initial Window Size (WS)
- Time To Live (TTL)
- Maximum Segment Size (MSS))
- Don't Fragment (DF) Bit
- Window Scale Value
- SackOK option presence
- Nop option presence [30].

These attribute values are then compared against a database of known responses until a match is found. Because various systems respond in different ways when they receive certain packet types, this information can be used to uniquely identify a given operating system. Figure 2.7 depicts how a remote host's fingerprint is observed passively.

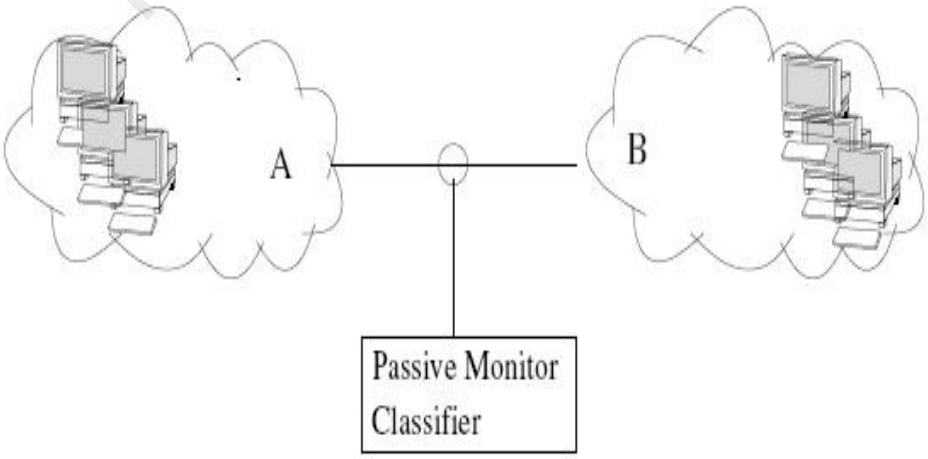

Figure 2.7: Passive Fingerprinting

Passive fingerprinting has some advantages over active fingerprinting. Passive fingerprinting tools act on all TCP/IP layers and can even detect the systems with low uptime. This method of fingerprinting provides a nearly undetectable method to map a network, finding vulnerable hosts. But passive fingerprinting is not perfect. Identifying the Operating System can be very time consuming or rather impossible, if the default values are changed i.e. the default information can be spoofed. It may also be the case where certain applications build their own packets and the signatures so produced may be different from the signatures produced by the Operating System itself.

2.4.4 Active vs. Passive Fingerprinting

Table 2.1 summarizes the difference between active and passive Operating System fingerprinting.

Passive Fingerprinting	**Active Fingerprinting**
Harder to detect	Easy to detect
Sometimes less accurate results	Usually more accurate
Personal firewalls not an issue	Personal firewalls block scans
No network effect	Consumes network bandwidth
Potentially time consuming	Quick to complete
Requires extensive access to target	Does not requires extensive access to target
Extremely stealthy technique	Not stealthy
Requires minimum number of open ports	Requires specific number and type of ports open to perform the test
Tools: Tcpdump, p0f, dsniff, ettercap, ethereal	Tools: Xprobe2, Nmap, ettercap, Nessus, MTR

Table 2.1: Active vs. Passive

2.5 Fingerprinting Tools

2.5.1 Nmap

Nmap is a network exploration tool and security scanner. It is designed to allow users to scan networks to determine which hosts are up and what services they offer. Nmap supports a number of scanning techniques that use the following protocols: TCP, ICMP, UDP, and IP. Nmap also includes features like remote

operating system detection, parallel scanning, port filtering detection, timing options, and flexible target and port specification.

Technique

Before Nmap runs its operating system detection method it runs a port scan against the target machine. It performs a port scan so it can find some open and closed ports on the target machine. Nmap works best when it finds at least one open TCP port, one closed TCP port, and one closed UDP port. Nmap works by conducting a set of tests against the target machine to try to determine what operating system it is running. And the all the nine Nmap tests are summarized in Table 2.2

- The first test is named T1 for test 1. In this test a TCP packet with the SYN, and ECN-Echo flags enabled is sent to an open TCP port.
- The second test is named T2 for test 2. It involves sending a TCP packet with no flags enabled to an open TCP port. This type of packet is known as a NULL packet.
- The third test is named T3 for test 3. It involves sending a TCP packet with the URG, PSH, SYN, and FIN flags enabled to an open TCP port.
- The fourth test is named T4 for test 4. It involves sending a TCP packet with the ACK flag enabled to an open TCP port.
- The fifth test is named T5 for test 5. It involves sending a TCP packet with the SYN flag enabled to a closed TCP port.
- The fifth test is named T5 for test 5. It involves sending a TCP packet with the SYN flag enabled to a closed TCP port.
- The sixth test is named T6 for test 6. It involves sending a TCP packet with the ACK flag enabled to a closed TCP port.
- The seventh test is named T7 for test 7. It involves sending a TCP packet with the URG, PSH, and FIN flags enabled to a closed TCP port.
- The eighth test is named PU for port unreachable test. It involves sending a UDP packet to a closed UDP port. The goal is to elicit an ICMP packet with an ICMP port unreachable message back from the target machine.
- The last test that Nmap performs is named TSeq for TCP sequenceability test. The test tries to determine the sequence generation patterns of the

Test	Send Packet	To Port	With flags enabled
T1	TCP	open TCP	SYN, ECN-Echo
T2	TCP	open TCP	no flags
T3	TCP	open TCP	URG, PSH, SYN, FIN
T4	TCP	open TCP	ACK
T5	TCP	closed TCP	SYN
T6	TCP	closed TCP	ACK
T7	TCP	closed TCP	URG, PSH, FIN
PU	UDP	closed UDP	
TSeq	TCP*6	open TCP	SYN

Table 2.2: Nmap Tests

TCP initial sequence numbers also known as TCP ISN sampling, the IP identification numbers also known as IPID sampling, and the TCP timestamp numbers. The test is performed by sending six TCP packets with the SYN flag enabled to an open TCP port.

After Nmap has received the results from all of the tests it builds an Operating System fingerprint signature, then tries to find a match in the fingerprint database. If the signature is found in the database, Nmap will list it as the remote operating system guess. If the signature isn't found in the database, Nmap will display a message saying "No exact matches for host".

Defences

There are some defensive measures to protect against the OS detection method used by Nmap. Among the defences are:-

- In a normal network environment systems should sit behind some type of firewall. Machines that are viewable from the Internet should only keep the needed ports open while the rest of the ports should be filtered by the firewall. This is a good defence since Nmap works best when it finds at least one open TCP port, one closed TCP port, and one closed UDP port. If all the needed ports aren't found then Nmap's accuracy drops off.
- Change characteristics of the machines TCP/IP stack. This can also be accomplished via kernel patches.

- Network Intrusion Detection Systems (IDS) can be used, if configured correctly, to detect the operating system detection method used by Nmap because of the utilization of malformed packets [27].

2.5.2 P0f

P0f is a tool which can fingerprint Operating System passively. It listens to the network and looks for first SYN packet in a TCP connection. When SYN packet is found, packet options are grabbed and matched with the database of Operating System signatures. P0f can identify the operating system on:-

- Machines that connect to in (SYN mode),
- Machines that connect to in (SYN+ACK mode),
- Machine that cannot connect to (RST+ mode),
- Machines whose communications can be observed.

P0f can also do many other tricks, and can detect or measure the following:-

- Firewall presence, NAT use (useful for policy enforcement),
- Existence of a load balancer setup,
- The distance to the remote system and its uptime,
- Hookup other network and ISP [38].

Technique
p0f uses a fingerprinting technique based on analyzing the structure of a TCP/IP packet to determine the operating system and other configuration properties of a remote host. The process is completely passive and does not generate any suspicious network traffic. The other host has to either:-

- *Connect to the network* either spontaneously or in an induced manner, for example when trying to establish a ftp data stream, returning a bounced mail, performing auth lookup, external html mail image reference and so on,
- *Contacted by some entity* on the network using some standard means (such as a web browsing); it can either accept or refuse the connection.

The method can see through packet firewalls . The main uses of passive operating system fingerprinting are attacker profiling (IDS and honeypots), visitor profiling

(content optimization), customer or user profiling (policy enforcement)and pen-testing [28].

p0f can run off-line and sift through large amounts of input data from various logs such as firewall logs, IDS logs, router logs etc. for long periods of time. All this information can be extracted and analyzed and give very interesting information of the systems connecting remotely to the network. The information in the packets being analyzed by p0f has often not been changed by the remote network's network devices such as proxies, network address translation. p0f also look for certain well-known signatures of the packet captured. The most common signatures to look for are the following fields in a packet:-

- *TTL (Time to Live):*is the maximum number of routers a packet can pass before it is being dropped. It is initialized by the sender and then decremented by every router handling the packet. When the value reaches 0, the packet is dropped and an ICMP message is returned to the sender. The TTL value set will differ from various operating systems. For instance Windows systems will have a value of 32 while Linux will have a TTL of 64.

- *Win (Window Size):* is the flow control option used by TCP. When a host initiates a connection it will advertise the size of its incoming packet buffer. The other host will then adjust the rate it sends packets to ensure that the receiving host is not flooded.

- *DF (Don't Fragment):* is the value set if the packet is not to be broken up into smaller fragments. This might be necessary if the packet is too large for the network to handle. If the DF flag is set and the packet is too large, it will be discarded an ICMP error message "fragmentation needed, but DF bit is set" will be sent to the source host.

- *TOS (Type of Service):*allows for 4 values to be set for each packet being sent. The value being set depends on the application being used and only one value can be set for each packet. The following values are available:-

  - Minimize delay
  - Maximize throughput
  - Maximize reliability
  - Minimize monetary costs [22].

2.6 Masking Techniques

Masking a system involves hiding/blocking the identifying details that the intruders could use to detect the Operating System version. This information may provide little or no utility to legitimate users, but it is often the starting place for crackers, "black hat" hackers because the security vulnerabilities tend to depend on software version. Blind probing reduces efficiency of any attack therefore; knowing the software details in advance increases the chances of successful cracking prior to detection.

To stop the attackers from exploiting the systems, protection against all types of fingerprinting attempts is required. There are many different ways through which the system can be shielded from different kind of fingerprinting attacks. Following are the figured out ways: -

2.6.1 Fingerprint Scrubber

Scrubbers are transparent, interposed mechanisms for explicitly removing network scans and attacks at various protocol layers. The fingerprint scrubber restricts a remote user's ability to determine t he operating system of another host on the network. The fingerprint scrubber works at both the network and transport layers to convert ambiguous traffic from a heterogeneous group of hosts into sanitized packets that do not reveal clues about the hosts' operating systems. Scrubber helps to block known stack fingerprinting techniques in a general, fast, scalable, and transparent manner, to block classes of scans, to handle many concurrent connections by introducing latency without losing any noticeable performance or behavioral differences in end hosts.

Scrubbing Techniques: Following are the most commonly used scrubbing techniques:-

IP Scrubbing: IP-level ambiguities facilitate Operating System fingerprinting. IP-level ambiguities arise mainly in IP header flags and fragment reassembly algorithms and make it easier to fingerprint an operating system. The IP scrubber modifies the IP header flags, removes uncommon and unused options, adjusts header checksum and reassembles the fragments so that the modified packet does not disclose hints about the hosts.

ICMP Scrubbing: ICMP messages contain ambiguities that can be used for fingerprinting. The ICMP scrubber modifies ICMP responses (error messages) and limits the rate of all outgoing ICMP messages in order to fool the fingerprinters.

TCP Scrubbing: A significant number of fingerprinting attacks take place at the TCP level; the majority of them are removed by the TCP protocol scrubber. Function of TCP scrubber is to enforce the standard TCP Three-Way Handshake,to reorder the options within the TCP header to a standard format and to modify the normal sequence number of new TCP connections [36].

2.6.2 Linux Kernel Patches

Following are the patches that are applied to kernel for dealing with fingerprinting.

IP Personality: It'a netfilter module (only available for 2.4 Linux kernels) which changes the IP stack behavior and 'personality'. IP Personality changes the way in which certain packets were answered by having multiple network personalities that can be defined by specifying different parameters in an iptables rule. Following options can be changed:-

- TCP Initial Sequence Number (ISN)
- TCP initial window size
- TCP options (their types, values and order in the packet)
- IP ID numbers
- answers to some pathological TCP packets
- answers to some UDP packets

Stealth Patch: It is available as a kernel module for linux kernels 2.4.x. When patched, two new options appear in our config file. The first option is, IP: TCP Stack Options, provides an option to choose the stealth patch. And if this option is selected then it is enabled by default when the system is booted. Second option is, Log all dropped packets, it logs all packets with bad options. This patch simply discards the TCP/IP packets received with the following matches:-

- Packets with both SYN and FIN activated
- Bogus Packets: if the TCP header has the one of the reserved bits active or it does not have any of the following activated: ACK, SYN, RST, FIN (Nmap test2)
- Packets with FIN, PUSH and URG activated (Nmap test7)

Fingerprinter Fucker: Fingerprint Fucker is a kernel module available for Linux kernel 2.2.x which can hide OS's identity and make it behave like another. It accepts parameters from the command line to configure the answer. It uses a fing_parses.c file which parses Nmap signature files and loads the Fingerprint Fucker module with the right parameters (while executing fing_parses the OS type to emulate is specified). On receiving Nmap bogus packets, this patch answers as per its configuration. It is seen that it can only fool Nmap T1,T2 and T7 tests.

IPlog :IPlog is a TCP/IP logger that also detects some scans (like XMAS, FIN, SYN). It can easily fool Nmap queries, and can also help to behave as other Operating System. To run iplog following command is used:-

#iplog -o -L -z -i eth0

The options are: -o (don't fork and stay in foreground), -L (results to stdout), -z (fool Nmap) -i eth0 (listen to eth0). When nmap is run, iplog gets started and displays information about all connections made and even which type of scanning is being performed [2].

Table 2.3 summarizes functionality of above described kernel patches.

2.6.3 IPtable Ruleset

In Linux, Iptable rulesets can be written to block both incoming and outgoing network traffic. Iptables is used to set up, maintain, and inspect the tables of IP packet filter rules in the Linux kernel. A firewall rule specifies criteria for a packet, and a target. If the packet does not match, the next rule is examined; if it does match, then that rule is specified by the value of the target, which can be either user-defined or one of the special values ACCEPT, DROP, QUEUE or RETURN. ACCEPT means to let the packet through. DROP means to drop the packet. QUEUE means to pass the packet to userspace and RETURN means stop traversing and resume at the next rule. Following iptable ruleset blocks Nmap XMAS and NULL scans.

Stop nmap XMAS scans
-A INPUT -p tcp -m tcp -tcp-flags FIN, SYN, RST, PSH, ACK, URG -j DROP
Stop nmap NULL scans
-A INPUT -p tcp -m tcp -tcp-flags FIN, SYN, RST, PSH, ACK, URG NONE -j DROP

Linux Kernel Patch	Written by	Year	Function	Downside
IP Personality (only for 2.4 Linux kernel)	Gael Roualland and Jean-Marc Saffroy	2001	Change TCP/IP stack personality. (modifies ISN,WS,TCP options)	Not maintained anymore. Might break regular connectivity by changing n/w parameters.
Stealth Patch (for version2.2.x and version 2.4.x)	Sean Trifero and Derek Callaway	2002	Drop specific packets typical for OS detectors	Only effects Nmap T2 and T7
Fingerprint Fucker (for version2.2.x)	Cyrax	2000	Answers to some Nmap bogus packets as configured and emulates the behavior of other OS	Only effects nmap T1,T2 and T3.
Iplog (standalone application)	Ryan McCabe	2001	Detects TCP, SYN, Null, FIN, Xmas, UDP, ICMP scans	Not maintained anymore. Too old to still fool Nmap.

Table 2.3: Comparison of Kernel Patches

2.6.4 Modified Registery keys

The Registry contains information that Windows continually references during operation, such as profiles for each user, the applications installed on the computer and the types of documents that each can create, property sheet settings for folders and application icons, what hardware exists on the system, and the ports that are being used [19]. Registry keys in Windows (2000 and higher versions) can be modified to disable communication on certain ports which in turn prevents Nmap from detecting the Operating System type: Following registry script snippet disables communication on all ports besides : 27,38,30 ==80 ==http,35,32,30==520 ==rip.

[HKEY LOCAL_MACHINE_SYSTEM\CurrentControlSet \Services \ <CARD NAME> \Parameters\Tcpip]
"TCPAllowedPorts"=hex(7):27,38,30,00,00 ;http(80)
"UDPAllowedPorts"=hex(7):35,32,30,00,00; rip(520)

"RawIPAllowedProtocols"=hex(7):36,00,31,37,00,00;
tcp(6) and udp(17) [16].

2.6.5 TCP Wrapper

The TCP Wrapper program is computer program that provides firewall services for Unix servers. TCP Wrapper monitors incoming packets; if an external computer attempts to connect, the wrapper first checks if that external entity is authorized to connect or not. If it is authorized, then access is permitted; if not, access is denied. Wrappers can be configured according individual user or network needs. They can be installed without any change to the existing software or to existing configuration files. Wrappers are simple and easy to validate because the security logic is encapsulated into a single program. Because the wrapped program remains a separate entity, it can be upgraded without a need to recertify the program that is wrapping it [10]. To fool various fingerprinters a wrapper program can be written to filter out unwanted traffic and to block certain ports.

2.6.6 Honeypots

A honeypot is valuable as a surveillance and early-warning tool. Honey pots are effective detection tools to sense attacks such as port scanning activities in the network. To fool network fingerprinting tools, Honeypot simulates the networking stack of different operating systems. This is called the personality of a virtual honeypot. Different personalities can be assigned to different virtual honeypots.The personality engine of the honeypot makes network stack behave differently as specified by the changes into the protocol headers of every outgoing packet [23]. Honeypot allows for the emulation of particular server and device type via the use of operating system fingerprints derived from the NMAP fingerprint database [34].

Chapter 3

Problem Statement

3.1 Gap Analysis and Problem Statement

Machines reveal a great deal about themselves like Operating system(s), vendor, version, Ports, protocols and vulnerabilities. The advertised information about the system is very important for the black-hat because many security holes are dependent on Operating System version. For example, with a good fingerprinter a hacker can quickly find out that a particular machine is running 'Solaris 2.51'or 'Linux 2.0.35' and in case of buffer overflow attack , the attacker can adjust the payload (which is Operating System or architecture dependent) accordingly. Also the attackers will not waste their time trying IIS exploits against Linux hosts. Thus, an attacker must know the operating system running on the target system before launching any attack. Hence, it is necessary to hide the Operating System by masking at host level. But if somehow, the identity of target host is revealed, and then also the system must be tough enough to protect itself from the hacker by eliminating its basic vulnerabilities so that it can't be exploited easily.

As per literature survey carried out, in order to hide the real Operating System identity, certain masking techniques can be employed. But these techniques alone are not sufficient to completely secure the system. Using protocol scrubbing or IP Personality system can fool a scanner by changing certain aspects of the TCP/IP stack (like initial sequence numbers, window sizes, TCP options, etc.) but tweaking those might break regular connectivity by changing network parameters or could also make the system weaker if the emulated TCP/IP stack is not as strong as the initial one. Other Linux kernel patches are also available like Stealth patch and IPlog, but the downside is that they are capable of blocking only few Nmap probes. Also, Registry keys and Iptable rulesets can be used to control the incoming and outgoing network traffic and in turn can be helpful in fooling active

fingerprinters like Nmap (which guesses Operating System on the basis of response for a crafty packet), but it is difficult to fool any passive fingerprinter by using this method. Even TCP wrappers are not useful in preventing the general case of IP spoofing. However, they do provide some armor against specific variations on the spoofing problem, and they provide a space for extra checks to be inserted where they may not be expected. Honeypots can also provide some sort of security by allowing the user to create virtual/fake hosts. Honeypots take an nmap database file as part of their config file, and can thus completely emulate a given Operating System. However, honeypots are for making virtual hosts, and cannot help to disguise a real host. They are not a defense tool, but a passive monitoring tool. Moreover, it is expensive and time consuming to set up a honeypot and perform analysis.

So there is a compelling requirement to device some mechanism to mask the Operating System, based on two basic principles of security: minimization and least privileges. The objective of this work is to implement a security framework for Linux Operating System that will eliminate the basic vulnerabilities of the system by disabling all non-essential software programs and utilities from the system. Changing the default settings actually helps in misguiding the attacker by providing incorrect or confusing information to attackers so that future attacks are less effective or slower than they would have been.

Chapter 4

Design and Implementation

4.1 Experimentation with Tools

4.1.1 Testing Nmap

Operating System Signature

An example signature from the Nmap database is shown and explained. This signature is built based on the information in the packets that were sent back in response to the tests that Nmap performed to detect the remote Operating System.

Fingerprint Solaris 9

TSeq(Class=RI%gcd=<6%SI=<A927C&>116A%IPID=I%TS=100HZ)
T1(DF=Y%W=C0B7|807A%ACK=S++%Flags=AS%Ops=NNTMNW)
T2(Resp=N)
T3(Resp=N)
T4(DF=Y%W=0%ACK=O%Flags=R%Ops=)
T5(DF=Y%W=0%ACK=S++%Flags=AR%Ops=)
T6(DF=Y%W=0%ACK=O%Flags=R%Ops=)
T7(DF=Y%W=0%ACK=S%Flags=AR%Ops=)
PU(DF=Y%TOS=0%IPLEN=70%RIPTL=148%RID=E%RIPCK=E
%UCK=E|F%ULEN=134%DAT=E)

Explanation

TSeq(Class=RI%gcd=<6%SI=<A927C&>116A%IPID=I%TS=100HZ)
This line represents the results of the TCP sequenceability test. Remember that this test tries to determine the generation patterns of the TCP initial sequence numbers, the IP identification numbers, and the TCP timestamp numbers. Class, gcd, and SI all come from TCP ISN sampling. Class=RI means that it falls in the

"Random Increments" class which means that each new TCP sequence number increases in such a way that it is difficult to guess precisely from the last one. gcd=<6 means that the greatest common divisor is less than six, which means that the sequence numbers don't have any large factors in common. SI=<A927C&>116A means that the sequence increments are less than 0xA927C in hexadecimal or 692860, and greater than 0x116A in hexadecimal or 4458. The sequence increments value is a statistical measure of variance meaning that the higher the number the more random looking the sequence numbers tend to be. IPID comes from IPID sampling. IPID=I means that the IP identification numbers fall in the "Incremental" class meaning they increase by one for each packet sent. TS come from the TCP timestamp sampling. TS=100HZ means that the TCP timestamp option seems to increment by about 100 every second.

T1(DF=Y%W=C0B7|807A%ACK=S++%Flags=AS%Ops=NNTMNW)
This line represents the results of test 1. DF=Y means that the Don't fragment flag in the IP header was enabled. W=C0B7|807A means that the window size in the TCP header is either 0xC0B7 (49335) or 0x807A (32890) in hexadecimal. ACK=S++ means that the acknowledgment number in the TCP header is our initial sequence number plus 1. Flags=AS means that the ACK and SYN flags in the TCP header were enabled. Ops=NNTMNW means that the following options were in this order: <NOP><NOP><Timestamp><Maximum segment size (MSS)><NOP><Window scale>.

T2(Resp=N)
This line represents the results of test 2. Resp=N means that we did not get a response back.

T3(Resp=N)
This line represents the results of test 3. Resp=N means that we did not get a response back.

T4(DF=Y%W=0%ACK=O%Flags=R%Ops=)
This line represents the results of test 4. DF=Y means that the Don't fragment flag in the IP header was enabled. W=0 means that the window size in the TCP header is 0. ACK=O means that the acknowledgment number in the TCP header is 0. Flags=R means that the RST flag in the TCP header was enabled. Ops= means that there weren't any options sent back.

T5(DF=Y%W=0%ACK=S++%Flags=AR%Ops=)

This line represents the results of test 5. DF=Y means that the Don't fragment flag in the IP header was enabled. W=0 means that the window size in the TCP header is 0. ACK=S++ means that the acknowledgment number in the TCP header is our initial sequence number plus 1. Flags=AR means that the ACK and RST flags in the TCP header were enabled. Ops= means that there weren't any options sent back.

T6(DF=Y%W=0%ACK=O%Flags=R%Ops=)
This line represents the results of test 6. DF=Y means that the Don't fragment flag in the IP header was enabled. W=0 means that the window size in the TCP header is 0. ACK=O means that the acknowledgment number in the TCP header is 0. Flags=R means that the RST flag in the TCP header was enabled. Ops= means that there weren't any options sent back.

T7(DF=Y%W=0%ACK=S%Flags=AR%Ops=)
This line represents the results of test 7. DF=Y means that the Don't fragment flag in the IP header was enabled. W=0 means that the window size in the TCP header is 0. ACK=S means that the acknowledgment number in the TCP header is our initial sequence number. Flags=AR means that the ACK and RST flags in the TCP header were enabled. Ops= means that there weren't any options sent back.

PU(DF=Y%TOS=0%IPLEN=70%RIPTL=148%RID=E%RIPCK=E
%UCK=E|F%ULEN=134%DAT=E)
This line represents the results of the port unreachable test. DF=Y means that the Don't fragment flag in the IP header was enabled. TOS=0 means that the type of service (TOS) in the IP header was 0. IPLEN=70 means that the total length in the IP header is 0x0070 in hexadecimal or 112. RIPTL=148 means that the total length given in the IP header sent back to us is 0x0148 in hexadecimal or 328. RID=E means that the identification number in the IP header we got back in the copy of our original UDP packet was the same as what we sent. RIPCK=E means that the header checksum in the IP header was the same. UCK=E—F means that the UDP checksum in the UDP header sometimes is found to be the same and sometimes not. ULEN=134 means that the UDP length in the UDP header is 0x0134 in hexadecimal or 308. DAT=E means that the UDP data was echoed back correctly. Since most implementations don't send any UDP data back, they get DAT=E by default [27].

Usage Example The sample run was produced utilizing a Windows Vista Home

Basic machine running Nmap targeting a Linux 2.4 RELEASE machine on VMware. Figure 4.1 is a sample run that Nmap produced:-

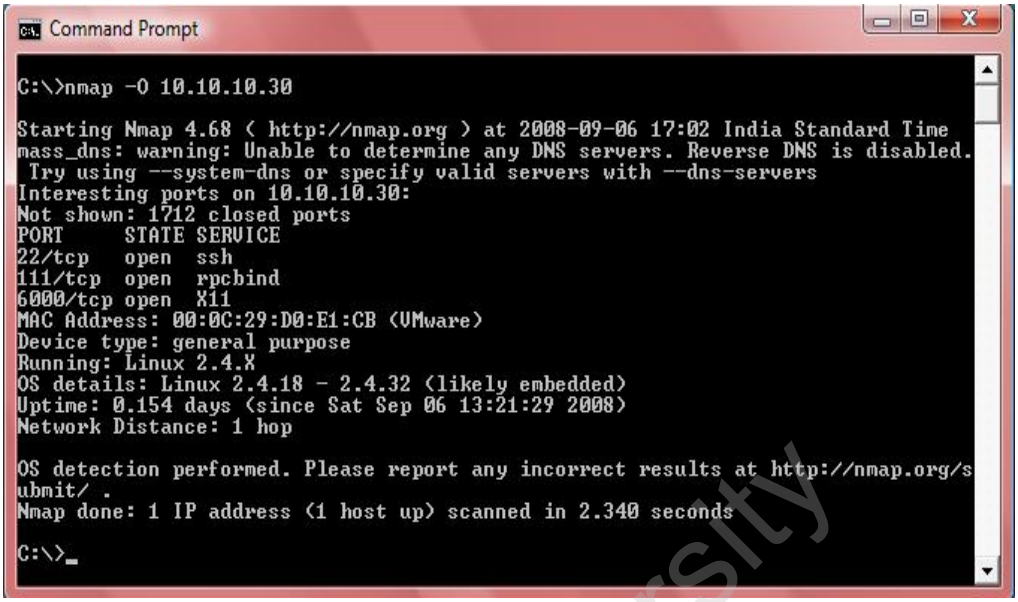

Figure 4.1: Nmap Output

4.1.2 Testing P0f

Command Line

P0f is easy to run with the following command:-

p0f [-f file] [-i device] [-s file] [-o file] [-Q socket [-0]] [-w file] [-u user] [-c size] [-T nn] [-e nn] [-FNODVUKAXMqxtpdlRL] ['filter rule']

The last part, 'filter rule', is a bpf-style filter expression for incoming packets. It is very useful for excluding or including certain networks, hosts, or specific packets, in the logfile.

Files

Default fingerprint database files are:-

- /etc/p0f/p0f.fp
- /etc/p0f/p0fa.fp
- /etc/p0f/p0fr.fp
- /etc/p0f/p0fo.fp [26].

Usage Example

An experimental network consists of two Linux boxes and one Windows 2000 box. P0f is installed on one of the Linux machine that also act as proxy. This experiment only captures the internal network traffic. The following command will start p0f:-

 # p0f -i eth1 -vt

The -i options allows for selecting the device which p0f should be extracting packets from. The -v option indicates that p0f is run in verbose mode while -t adds timestamps to the output. An example of the output from the above command is shown:-

```
# p0f -i eth1 -vt

    p0f: file: '/etc/p0f.fp', 139 fprints, iface: 'eth1',rule: 'all'

    192.168.1.10 [1 hops]: Windows 2000

    + 192.168.1.10:3169 -> 192.168.1.1:23

    192.168.1.10 [1 hops]: Windows 2000

    + 192.168.1.10:3171 -> 195.139.5.245:80

    192.168.1.10 [1 hops]: Windows 2000

    + 192.168.1.10:3172 -> 195.139.5.245:80
```

The fingerprint information is located in a file called /etc/p0f.fp and is the file used by p0f by default. However, p0f can be directed to use another fingerprint file using the -f option. The output can also be directed to a file using the -o option:-

```
# p0f -i eth1 -vto output.txt
```

The following output shows an nmap attack being picked up by p0f. p0f was analyzing live data.

192.168.1.14 [24 hops]: NMAP scan (distance inaccurate)

+ 192.168.1.14:52424 -> 192.168.1.1:932

192.168.1.14 [24 hops]: NMAP scan (distance inaccurate)

+192.168.1.14:52424 -> 192.168.1.1:1482 192.168.1.14 [24 hops]: NMAP scan (distance inaccurate)

+ 192.168.1.14:52424 -> 192.168.1.1:416 192.168.1.14 [24 hops]: NMAP scan (distance inaccurate)

+ 192.168.1.14:52424 -> 192.168.1.1:937 192.168.1.14 [24 hops]: NMAP scan (distance inaccurate)

+ 192.168.1.14:52424 -> 192.168.1.1:3141 192.168.1.14 [24 hops]: NMAP scan (distance inaccurate)

+ 192.168.1.14:52424 -> 192.168.1.1:546 192.168.1.14 [24 hops]: NMAP scan (distance inaccurate) [**?**].

4.2 Research Findings

Three University Servers were scanned using Nmap 4.76.

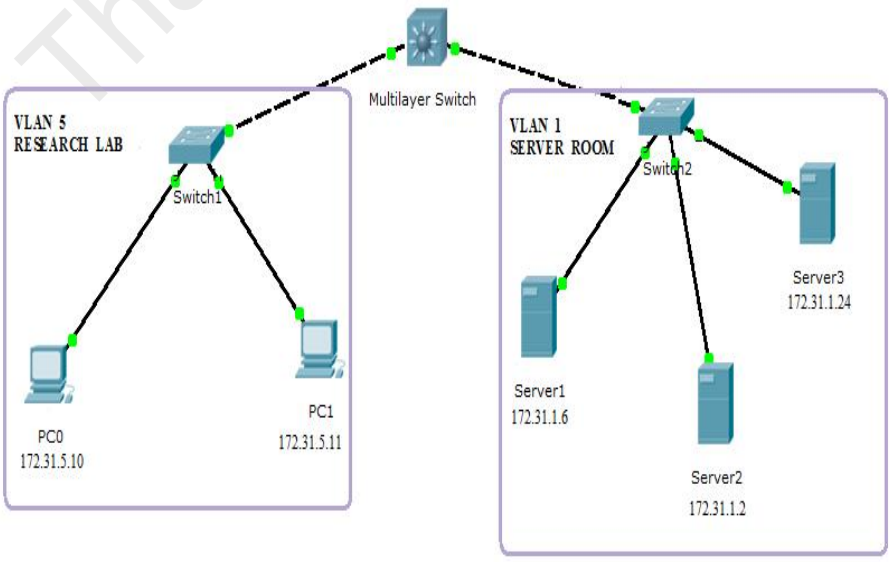

Figure 4.2: Testbed

Figure 4.2 shows the experimental scenario. The servers that were scanned are part of VLAN1 and are placed in the server room, whereas the host PC on which Nmap was installed is part of VLAN5 in the research lab. And the Table 4.1 depicts results of all Nmap commands executed in the experimental environment.

Table 4.1: Scan Results

Scan Type	**Server1**	**Server2**	**Server3**
-sT	No. of Filtered Ports: 995. Open TCP Ports: 23==telnet, 80==http, 443==https, 3128==squid-http. Close TCP Port: 1723==pptp. Time taken: 425.67 seconds.	No. of Filtered Ports: 998. Open TCP Ports: 22==ssh, 80==http. Time taken: 61.42 seconds.	No. of Closed Ports: 992. Open TCP Port: 23==telnet. Filtered TCP Ports: 135==msrpc, 139==netbios-ssn, 445==microsoft-ds, 4662==edonkey, 6346==gnutella, 6699==napster, 6881==bittorrent-tracker. Time taken: 261.95 seconds.
-sS	No. of Filtered Ports: 996. Open TCP Ports: 23==telnet, 80==http, 443==https. Close TCP Port: 1723==pptp. Time taken: 19.84 seconds.	No. of Filtered Ports: 998. Open TCP Ports: 22==ssh, 80==http. Time taken: 7.82 seconds.	No. of Closed Ports: 992. Open TCP Port: 23==telnet. Filtered TCP Ports: 135==msrpc, 139==netbios-ssn, 445==microsoft-ds, 4662==edonkey, 6346==gnutella, 6699==napster, 6881==bittorrent-tracker. Time taken: 2.55 seconds.

Continued on next page

Table 4.1 – continued from previous page

Scan Type	Server1	Server2	Server3
-sF	All 1000 scanned ports are open—filtered. Time taken: 22.36 seconds.	All 1000 scanned ports are open—filtered. Time taken: 9.53 seconds.	No. of Closed Ports: 993. Open—Filtered TCP Ports: 135==msrpc, 139==netbios-ssn, 445==microsoft-ds, 4662==edonkey, 6346==gnutella, 6699==napster, 6881==bittorrent-tracker. Time taken: 6.82 seconds.
-sN	All 1000 scanned ports are open—filtered. Time taken: 22.22 seconds.	All 1000 scanned ports are open—filtered. Time taken: 8.72 seconds.	No. of Closed Ports: 993. Open—Filtered TCP Ports: 135==msrpc, 139==netbios-ssn, 445==microsoft-ds, 4662==edonkey, 6346==gnutella, 6699==napster, 6881==bittorrent-tracker. Time taken: 2.51 seconds.
-sX	All 1000 scanned ports are open—filtered. Time taken: 21.97 seconds.	All 1000 scanned ports are open—filtered. Time taken: 8.84 seconds.	No. of Closed Ports: 993. Open—Filtered TCP Ports: 135==msrpc, 139==netbios-ssn, 445==microsoft-ds, 4662==edonkey, 6346==gnutella, 6699==napster, 6881==bittorrent-tracker. Time taken: 2. 44 seconds.
-sP	Host appears to be up. Time taken: 0.51 seconds.	Host appears to be up. Time taken: 0.51 seconds.	Host appears to be up. Time taken: 0.48 seconds.

Continued on next page

Table 4.1 – continued from previous page

Scan Type	Server1	Server2	Server3
-sU	All 1000 scanned ports are open—filtered. Time taken: 21.99 seconds.	All 1000 scanned ports are open—filtered. Time taken: 8.78 seconds.	All 1000 scanned ports are open—filtered. Time taken: 9.42 seconds.
-sO	No. of Open—Filtered Protocols: 255. Open Protocol Service: 1==icmp. Time taken: 3.28 seconds.	No. of Open—Filtered Protocols: 255. Open Protocol Service: 1==icmp. Time taken: 14.21 seconds.	No. of Open—Filtered Portocols: 254. Open Protocol Service: 1==icmp, 6==tcp. Time taken: 4.84 seconds.
-sI	Warning: No targets were specified, so 0 hosts scanned. Time Taken: 2.17 seconds.	Warning: No targets were specified, so 0 hosts scanned. Time Taken: 2.22 seconds.	Warning: No targets were specified, so 0 hosts scanned. Time Taken: 2.23 seconds.
-sV	No. of Filtered Ports: 996. Open TCP Ports: 23==telnet: ver==Linux telnetd, 80==http: ver==Apache htptd, 443==ssl/http: ver==Apache httpd. Closed Port: 1723==pptp. Service Info: OS: Linux. Time Taken: 27.35 seconds.	No. of Filtered Ports: 998. Open TCP Ports: 22==ssh: ver==OpenSSH 3.91p1(protocol 1.99), 80==http: ver==Apache htptd 2.0.52 (Red Hat). Service detection performed: report any incorrect results. Time Taken: 13.95 seconds.	No. of Closed Ports: 992. Open TCP Port: 23==telnet:ver==Cisco router. Filtered TCP Ports: 135==msrpc, 139==netbios-ssn, 445==microsoft-ds, 4662==edonkey, 6346==gnutella, 6699==napster, 6881==bittorrent-tracker. Service Info: OS: IOS:Device: router. Time taken: 2.69seconds.

Continued on next page

Table 4.1 – continued from previous page

Scan Type	Server1	Server2	Server3
-sA	All 1000 scanned ports are filtered. Time Taken: 23.53 seconds.	No. of Filtered Ports: 998. Unfiltered TCP Ports: 22==ssh, 80==http Time taken: 7.72 seconds.	No. of Unfiltered Ports: 993. Filtered TCP Ports: 135==msrpc, 139==netbios-ssn, 445==microsoft-ds, 4662==edon-key, 6346==gnutella, 6699==napster, 6881==bittorrent-tracker. Time taken: 2.77seconds.
-sW	All 1000 scanned ports are filtered. Time Taken: 23.38 seconds.	No. of Filtered Ports: 998. Closed TCP Ports: 22==ssh, 80==http. Time taken: 7.73 seconds.	No. of Closed Ports: 993. Filtered TCP Ports: 135==msrpc, 139==netbios-ssn, 445==microsoft-ds, 4662==edon-key, 6346==gnutella, 6699==napster, 6881==bittorrent-tracker. Time taken: 4.99seconds.
-sR	No. of Filtered Ports: 996. Open TCP Ports: 23==telnet, 80==http, 443==https, 1723==pptp. Time taken: 20.89 seconds.	No. of Filtered Ports: 998. Open TCP Ports: 22==ssh, 80==http. Time taken: 8.80 seconds.	No. of Closed Ports: 993. Open TCP Port: 23==telnet. Filtered TCP Ports: 135==msrpc, 139==netbios-ssn, 445==microsoft-ds, 4662==edon-key, 6346==gnutella, 6699==napster, 6881==bittorrent-tracker. Time taken: 2.55 seconds.
-sL	Host not scanned. Time taken: 0.19 seconds.	Host not scanned. Time taken: 0.20 seconds.	Host not scanned. Time taken: 0.23 seconds.

Continued on next page

Table 4.1 – continued from previous page

Scan Type	Server1	Server2	Server3
-T0	Host seems down. If it is really up, but blocking our ping probes, try -PN. Time taken: 1501.21 seconds.	Host seems down. If it is really up, but blocking our ping probes, try -PN. Time taken: 1501.24 seconds.	scan not finished.
-T1	Host seems down. If it is really up, but blocking our ping probes, try -PN. Time taken: 76.20 seconds.	Host seems down. If it is really up, but blocking our ping probes, try -PN. Time taken: 76.20 seconds.	No. of Closed Ports: 992. Open TCP Port: 23==telnet. Filtered TCP Ports: 135==msrpc, 139==netbios-ssn, 445==microsoft-ds, 4662==edonkey, 6346==gnutella, 6699==napster, 6881==bittorrent-tracker. Time taken: 16943.37 seconds.
-T2	Host seems down. If it is really up, but blocking our ping probes, try -PN. Time taken: 5.60 seconds.	Host seems down. If it is really up, but blocking our ping probes, try -PN. Time taken: 5.60 seconds.	No. of Closed Ports: 992. Open TCP Port: 23==telnet. Filtered TCP Ports: 135==msrpc, 139==netbios-ssn, 445==microsoft-ds, 4662==edonkey, 6346==gnutella, 6699==napster, 6881==bittorrent-tracker. Time taken: 416.22 seconds.

Continued on next page

Table 4.1 – continued from previous page

Scan Type	Server1	Server2	Server3
-T3	Host seems down. If it is really up, but blocking our ping probes, try -PN. Time taken: 3.50 seconds.	Host seems down. If it is really up, but blocking our ping probes, try -PN. Time taken: 3.50 seconds.	No. of Closed Ports: 992. Open TCP Port: 23==telnet. Filtered TCP Ports: 135==msrpc, 139==netbios-ssn, 445==microsoft-ds, 4662==edonkey, 6346==gnutella, 6699==napster, 6881==bittorrent-tracker. Time taken: 3.01 seconds.
-T4	Host seems down. If it is really up, but blocking our ping probes, try -PN. Time taken: 2.50 seconds.	Host seems down. If it is really up, but blocking our ping probes, try -PN. Time taken: 2.51 seconds.	No. of Closed Ports: 992. Open TCP Port: 23==telnet. Filtered TCP Ports: 135==msrpc, 139==netbios-ssn, 445==microsoft-ds, 4662==edonkey, 6346==gnutella, 6699==napster, 6881==bittorrent-tracker. Time taken: 3.32 seconds.
-T5	Host seems down. If it is really up, but blocking our ping probes, try -PN. Time taken: 2.01 seconds.	Host seems down. If it is really up, but blocking our ping probes, try -PN. Time taken: 2.01 seconds.	No. of Closed Ports: 992. Open TCP Port: 23==telnet. Filtered TCP Ports: 135==msrpc, 139==netbios-ssn, 445==microsoft-ds, 4662==edonkey, 6346==gnutella, 6699==napster, 6881==bittorrent-tracker. Time taken: 4.14 seconds.

Continued on next page

Table 4.1 – continued from previous page

Scan Type	Server1	Server2	Server3
-D	No targets were specified, so 0 hosts scanned. Time taken: 0.21 seconds.	No targets were specified, so 0 hosts scanned. Time taken: 0.17 seconds.	No targets were specified, so 0 hosts scanned. Time taken: 0.19 seconds.
-n	No. of Filtered Ports: 994. Open TCP Ports: 23==telnet, 80==http, 443==https, 3128==squid-http, 8090==unknown. Close TCP Port: 1723==pptp. Time taken: 26.58 seconds.	No. of Filtered Ports: 998. Open TCP Ports: 22==ssh, 80==http. Time taken: 8.14 seconds.	No. of Closed Ports: 992. Open TCP Port: 23==telnet. Filtered TCP Ports: 135==msrpc, 139==netbios-ssn, 445==microsoft-ds, 4662==edonkey, 6346==gnutella, 6699==napster, 6881==bittorrent-tracker. Time taken: 2.38 seconds.
-P0	No. of Filtered Ports: 994. Open TCP Ports: 23==telnet, 80==http, 443==https, 3128==squid-http, 8090==unknown. Close TCP Port: 1723==pptp. Time taken: 41.01 seconds.	No. of Filtered Ports: 998. Open TCP Ports: 22==ssh, 80==http. Time taken: 8.14 seconds.	No. of Closed Ports: 992. Open TCP Port: 23==telnet. Filtered TCP Ports: 135==msrpc, 139==netbios-ssn, 445==microsoft-ds, 4662==edonkey, 6346==gnutella, 6699==napster, 6881==bittorrent-tracker. Time taken: 4.67 seconds.

Continued on next page

Table 4.1 – continued from previous page

Scan Type	Server1	Server2	Server3
-PT	Host seems down. If it is really up, but blocking our ping probes, try -PN. Time taken: 2.50 seconds.	No. of Filtered Ports: 998. Open TCP Ports: 22==ssh,80==http. Time taken: 7.80 seconds.	No. of Closed Ports: 992. Open TCP Port: 23==telnet. Filtered TCP Ports: 135==msrpc, 139==netbios-ssn, 445==microsoft-ds, 4662==edonkey, 6346==gnutella, 6699==napster, 6881==bittorrent-tracker. Time taken: 3.55 seconds.
-PS	No. of Filtered Ports: 994. Open TCP Ports: 23==telnet, 80==http, 443==https, 3128==squid-http, 8090==unknown. Close TCP Port: 1723==pptp. Time taken: 43.02 seconds.	No. of Filtered Ports: 998. Open TCP Ports: 22==ssh, 80==http. Time taken: 12.32 seconds.	No. of Closed Ports: 992. Open TCP Port: 23==telnet. Filtered TCP Ports: 135==msrpc, 139==netbios-ssn, 445==microsoft-ds, 4662==edonkey, 6346==gnutella, 6699==napster, 6881==bittorrent-tracker. Time taken: 3.49 seconds.
-PU	Host seems down. If it is really up, but blocking our ping probes, try -PN. Time taken: 2.49 seconds.	Host seems down. If it is really up, but blocking our ping probes, try -PN. Time taken: 2.48 seconds.	Host seems down. If it is really up, but blocking our ping probes, try -PN. Time taken: 2.48 seconds.

Continued on next page

Table 4.1 – continued from previous page

Scan Type	Server1	Server2	Server3
-PE	No. of Filtered Ports: 996. Open TCP Ports: 23==telnet, 80==http, 443==https. Close TCP Port: 1723==pptp. Time taken: 17.29 seconds.	No. of Filtered Ports: 998. Open TCP Ports: 22==ssh, 80==http. Time taken: 7.76 seconds.	No. of Closed Ports: 992. Open TCP Port: 23==telnet. Filtered TCP Ports: 135==msrpc, 139==netbios-ssn, 445==microsoft-ds, 4662==edonkey, 6346==gnutella, 6699==napster, 6881==bittorrent-tracker. Time taken: 3.74 seconds.
-PP	Host seems down. If it is really up, but blocking our ping probes, try -PN. Time taken: 2.52 seconds.	Host seems down. If it is really up, but blocking our ping probes, try -PN. Time taken: 2.49 seconds.	Host seems down. If it is really up, but blocking our ping probes, try -PN. Time taken: 2.66 seconds.
-PM	Host seems down. If it is really up, but blocking our ping probes, try -PN. Time taken: 2.49 seconds.	Host seems down. If it is really up, but blocking our ping probes, try -PN. Time taken: 2.51 seconds.	Host seems down. If it is really up, but blocking our ping probes, try -PN. Time taken: 2.66 seconds.
-PB	No. of Filtered Ports: 995. Open TCP Ports: 23==telnet, 80==http, 443==https, 8090==unknown. Close TCP Port: 1723==pptp. Time taken: 18.92 seconds.	No. of Filtered Ports: 998. Open TCP Ports: 22==ssh, 80==http. Time taken: 7.60 seconds.	No. of Closed Ports: 992. Open TCP Port: 23==telnet. Filtered TCP Ports: 135==msrpc, 139==netbios-ssn, 445==microsoft-ds, 4662==edonkey, 6346==gnutella, 6699==napster, 6881==bittorrent-tracker. Time taken: 2.91 seconds.

Continued on next page

Table 4.1 – continued from previous page

Scan Type	Server1	Server2	Server3
-PN	No. of Filtered Ports: 995. Open TCP Ports: 23==telnet, 80==http, 443==https, 3128==squid-http. Close TCP Port: 1723==pptp. Time taken: 15.96 seconds.	No. of Filtered Ports: 998. Open TCP Ports: 22==ssh, 80==http. Time taken: 7.65 seconds.	No. of Closed Ports: 992. Open TCP Port: 23==telnet. Filtered TCP Ports: 135==msrpc, 139==netbios-ssn, 445==microsoft-ds, 4662==edonkey, 6346==gnutella, 6699==napster, 6881==bittorrent-tracker. Time taken: 4.23 seconds.
-sS -f	Warning: Packet fragmentation selected on a host other than Linux, OpenBSD, FreeBSD or NetBSD, may or may not work. All 1000 scanned ports are filtered. Time taken: 23.78 seconds.	Warning: Packet fragmentation selected on a host other than Linux, OpenBSD, FreeBSD or NetBSD, may or may not work. All 1000 scanned ports are filtered. Time taken: 23.08 seconds.	Warning: Packet fragmentation selected on a host other than Linux, OpenBSD, FreeBSD or NetBSD, may or may not work. All 1000 scanned ports are filtered. Time taken: 24.29 seconds
-sF -f	Warning: Packet fragmentation selected on a host other than Linux, OpenBSD, FreeBSD or NetBSD, may or may not work. All 1000 scanned ports are filtered. Time taken: 22.11 seconds.	Warning: Packet fragmentation selected on a host other than Linux, OpenBSD, FreeBSD or NetBSD, may or may not work. All 1000 scanned ports are filtered. Time taken: 23.54 seconds.	Warning: Packet fragmentation selected on a host other than Linux, OpenBSD, FreeBSD or NetBSD, may or may not work. All 1000 scanned ports are filtered. Time taken: 22.63 seconds

Continued on next page

Table 4.1 – continued from previous page

Scan Type	Server1	Server2	Server3
-sX -f	Warning: Packet fragmentation selected on a host other than Linux, OpenBSD, FreeBSD or NetBSD, may or may not work. All 1000 scanned ports are filtered. Time taken: 23.23 seconds.	Warning: Packet fragmentation selected on a host other than Linux, OpenBSD, FreeBSD or NetBSD, may or may not work. All 1000 scanned ports are filtered. Time taken: 23.73 seconds.	Warning: Packet fragmentation selected on a host other than Linux, OpenBSD, FreeBSD or NetBSD, may or may not work. All 1000 scanned ports are filtered. Time taken: 24.01 seconds
-sN -f	Warning: Packet fragmentation selected on a host other than Linux, OpenBSD, FreeBSD or NetBSD, may or may not work. All 1000 scanned ports are filtered. Time taken: 23.15 seconds.	Warning: Packet fragmentation selected on a host other than Linux, OpenBSD, FreeBSD or NetBSD, may or may not work. All 1000 scanned ports are filtered. Time taken: 22.18 seconds.	Warning: Packet fragmentation selected on a host other than Linux, OpenBSD, FreeBSD or NetBSD, may or may not work. All 1000 scanned ports are filtered. Time taken: 22.96 seconds
-O -v	Device Type: Firewall or WAP or telecom-misc. No exact OS matches for host. IPID Sequence Generation: All zeros. Time taken: 29.73 seconds.	Device Type: general purpose. Running: Linux 2.6.X. OS Details: Linux 2.6.9-2.6.15. Uptime guess: 32.843 days. IPID Sequence Generation: All zeros. Time taken: 9.48 seconds.	Device Type: WAP or specialized or switch or router or broadband router. No exact OS matches for host. IPID Sequence Generation: All zeros. Time taken: 5.87 seconds.

Continued on next page

Table 4.1 – continued from previous page

Scan Type	Server1	Server2	Server3
-oA	No. of Filtered Ports: 994. Open TCP Ports: 23==telnet, 80==http, 443==https, 3128==squid-http, 8090==unknown. Close TCP Port: 1723==pptp. Time taken: 31.62 seconds.	No. of Filtered Ports: 998. Open TCP Ports: 22==ssh, 80==http. Time taken: 7.60 seconds.	No. of Closed Ports: 992. Open TCP Port: 23==telnet. Filtered TCP Ports: 135==msrpc, 139==netbios-ssn, 445==microsoft-ds, 4662==edonkey, 6346==gnutella, 6699==napster, 6881==bittorrent-tracker. Time taken: 3.00 seconds.
-6	Error code 11001: No such host is known. Warning: No targets were specified, so 0 hosts scanned. Time taken: 0.19 seconds.	Error code 11001: No such host is known. Warning: No targets were specified, so 0 hosts scanned. Time taken: 0.17 seconds.	Error code 11001: No such host is known. Warning: No targets were specified, so 0 hosts scanned. Time taken: 0.75 seconds.
-F	No. of Filtered Ports: 95. Open TCP Ports: 23==telnet, 80==http, 443==https, 3128==squid-http. Close TCP Port: 1723==pptp. Time taken: 2.27 seconds.	No. of Filtered Ports: 98. Open TCP Ports: 22==ssh, 80==http. Time taken: 2.31 seconds.	No. of Closed Ports: 96. Open TCP Port: 23==telnet. Filtered TCP Ports: 135==msrpc, 139==netbios-ssn, 445==microsoft-ds. Time taken: 2.16 seconds.

Continued on next page

Table 4.1 – continued from previous page

Scan Type	Server1	Server2	Server3
-ttl3	No. of Filtered Ports: 996. Open TCP Ports: 23==telnet, 80==http, 443==https. Close TCP Port: 1723==pptp. Time taken: 571.59 seconds.	No. of Filtered Ports: 998. Open TCP Ports: 22==ssh, 80==http. Time taken: 7.85 seconds.	No. of Closed Ports: 992. Open TCP Port: 23==telnet. Filtered TCP Ports: 135==msrpc, 139==netbios-ssn, 445==microsoft-ds, 4662==edonkey, 6346==gnutella, 6699==napster, 6881==bittorrent-tracker. Time taken: 3.36 seconds.

As shown in the Figure 4.3, with sample set of Nmap commands;

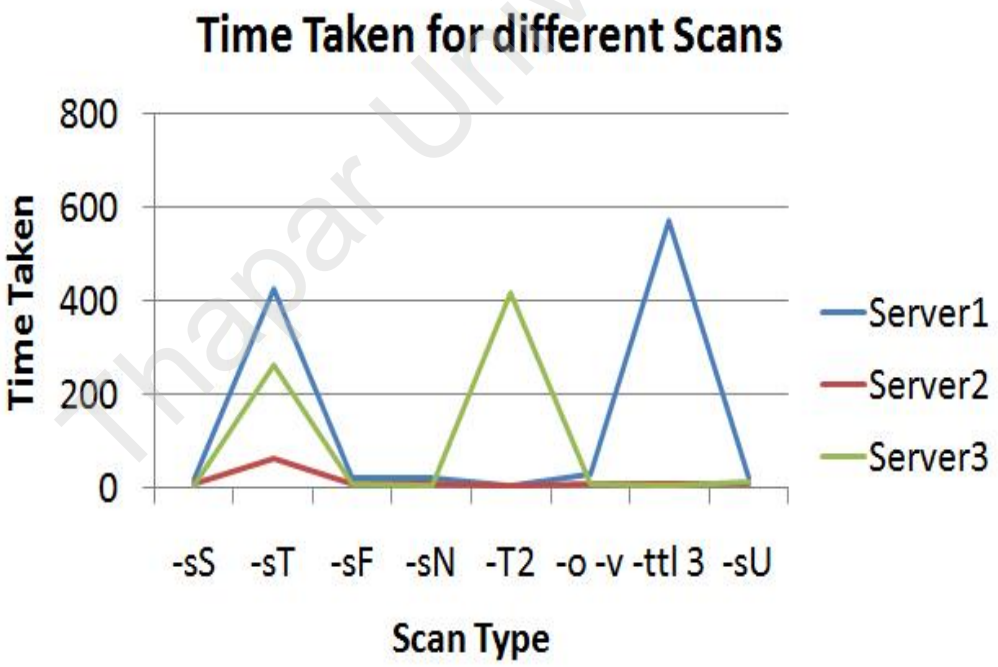

Figure 4.3: Scan Time

time taken to scan Server1 is more than other two servers because Server1 has Security Enhanced-Linux installed which is a hardened system. For Server3 scanning takes somewhat less time as this server is also hardened i.e. it has security enabled Internetworking Operating System installed. Server2 which is a general

purpose Linux machine takes very less time in responding to Nmap's crafted packets. The result says that, the more time it takes to scan a system means the more security enhanced Operating System it is. Or the other way around, hardened operating system's identity cannot be easily guessed whereas the identity of unhardened Operating System can be 100 percent revealed.

The above experiment demonstrates that in order to secure the system, Operating System hardening is must. Operating System hardening is a proactive technique for providing system security. It involves removing unnecessary services from the base operating system like restricting user access from some utilities that ordinary users shouldn't need, enforcing password restrictions, and controlling user and group rights.

To accomplish this task a shell script along with certain kernel hookups are implemented.

4.3 Implementation Hardening

4.3.1 Changing Kernel Parameters

Linux is a parameter driven system. Kernel parameters used for system configuration are found in /proc/sys/kernel, where there is an individual file for each configuration parameter. These parameters have a direct effect on system performance and viability. Modifying kernel parameters will both ignore inbound packets and refuse to send outbound packets. This prevents the host from being victimized and prevents it from being used to launch attacks.

Disabling ICMP Broadcast Echo Activity: ICMP broadcast echo activity is disabled because, otherwise the system could be used as part of a Smurf attack. This is a type of denial-of-service attack that floods a target system via spoofed broadcast ping messages. In such an attack, a perpetrator sends a large amount of ICMP echo request (ping) traffic to IP broadcast addresses, all of which have a spoofed source IP address of the intended victim. Most hosts on that IP network will take the ICMP echo request and reply to it with an echo reply, multiplying the traffic by the number of hosts responding and on a multi-access broadcast network, hundreds of machines might reply to each packet. So, to safeguard the system against this attack, the system can be augmented with boot time parameters such that it does not respond to these packets. This can be achieved by:-

 # sysctl -w net.ipv4.icmp_echo_ignore_broadcasts=1

Ignoring ICMP Requests: Most of the OS fingerprinters work in three phases to detect the OS of a target system. The first phase is called host detection where the attacker attempts to determine the accessible hosts on a network. This is done by sending a ping request to the target machine and if the machine will be alive, it will send back the ping reply. This host discovery can be failed if the system is configured to ignore Ping requests.

 # sysctl -w net.ipv4.icmp_echo_ignore_all=1

Disabling ICMP Routing Redirects : ICMP is heavily used by routers, as well as clients and servers to determine network errors and availability, as well as performance statistics through various types of ICMP Packets. There are certain cases where ICMP packets can be used to attack a network. This is the case with ICMP redirect, or ICMP Type 5 packet. ICMP redirects are used by routers to inform the host what the correct route should be if the host uses a non-optimal or defunct route to a particular destination. The security problem comes from the fact that ICMP packets, including ICMP redirect, are extremely easy to fake and it would be rather easy for an attacker to forge ICMP redirect packets. The attacker can alter the routing tables on the host and possibly subvert the security of the host by causing traffic to flow via a path that is not intended. Due to this fact and the security risks involved in such scenario, it is strongly recommended to disable ICMP redirect messages (ignore them) from all public interfaces. This can be accomplished by:

 # sysctl -w net.ipv4.conf.all.accept_redirects=0
 # sysctl -w net.ipv4.conf.all.send_redirects=0

Disable IP Source Routing: Source routing is an IP option which allows the originator of a packet to specify what path that packet will take, and on what path the packet will return back to the originator. With IP source routing enabled, packets containing the source route option are forwarded in accordance with the specified router addresses contained in the header. Source routing is useful when the default route fails, or for network diagnostic purposes. Unfortunately, source routing is often abused by malicious users, and used to make a machine A, think it is talking to a different machine B, when it is actually talking to a third machine C. This means that C has control over B's IP address. The proper way to fix this problem is to augment machine A with boot time parameters to ignore source-routed packets.

 # sysctl -w net.ipv4.conf.all.accept_source_route=0
 # sysctl -w net.ipv4.conf.all.forwarding=0
 # sysctl -w net.ipv4.conf.all.mc_forwarding=0

Enforcing Ingress Filtering: Ingress filtering is a packet filtering technique to prevent source address spoofing. The point is to drop a packet if the source and destination IP addresses in the IP header do not make sense when considered in light of the physical interface on which it arrived. By using ingress filtering all the spoofed packets can be sort out that may be directed towards the network with some purpose. For e.g.: Filtering may restrict traffic not bearing an IP address of a particular network from heading out of a network. This ensures that the computer cannot be used as an amplifier for SMURF attacks and this can be achieved by:

 # sysctl -w net.ipv4.conf.all.rp_filter=1

Logging and Dropping Martin Packets: A "Martian" packet is one for which the host does not have a route back to the source IP address. These days most hosts have a default route, meaning that there would be no such thing as a Martian packet, but to be safe and complete, these packets are dropped. This can be accomplished by passing following boot time parameter.

 # sysctl -w net.ipv4.conf.all.log_martians=1

Using TCP SYN Cookies: SYN cookies are used to guard against SYN flood attacks. With TCP SYN Cookies enabled, the kernel does not allocate the TCP buffers unless the server's ACK/SYN packet gets an ACK back, meaning that it was a legitimate request. In particular, the system that uses SYN cookies doesn't drop the connection when its SYN queue fills up, instead it will send back SYN/ACK and will wait for ACK for a specified time and will allocate buffer to that connection only if the system receives an ACK back within that time slot. Succeeding boot time parameter can be passed to enable SYN cookie.

 # sysctl -w net.ipv4.tcp_syncookies=1

Reducing number of allowed HALF_OPEN circuits: This is another way to handle the SYN attack. In this, the backlog queue can be modified to support more half-open connections without denying access to legitimate clients. With the increased size of backlog queue the server will not run out of memory and can serve additional incoming requests. Hence, performance of the server will not be compromised. The backlog queue can be modified by the following parameter:

 # sysctl -w net.ipv4.tcp_max_syn_backlog=1280

Controlling Number of TCP ACK Retransmissions: The main point is to decrease the total time of handling connection request. When a server receives a request, it immediately sends a response with SYN/ACK, puts the HALF_OPEN

connection into backlog queue and waits for ACK from the client. When no response from client is received, the server retransmits SYN/ACK packet. And it is known that in case of SYN flooding or spoofing the response will never come back. So it is recommended to speed up the time of retransmission and to change the total number of retransmissions by passing following boot time parameter.

 # sysctl -w net.ipv4.tcp_synack_retries=0

4.3.2 Renaming Root Account

Renaming root account is one way to hide root password from the system cracker. For e.g. the "root" can be renamed to "guest" and this will put the cracker in illusion mode, that the root account s/he is looking for is not present on the system. And the cracker will never think of cracking "guest" account because according to him the "guest" account is non-privileged account. Hence, root account will be protected.

4.3.3 Removing Read, Execute and Write Access

The default file permissions set by most vendors are fairly secure. To make them more secure the non-root user access to some administrator functions can be removed. This option will change the permissions on some common system administration utilities so that they are not readable or executable by users other than the root. These utilities are the ones that most users should never have a need to access like the following:-

- *setserial*: this is a program designed to set and/or report the configuration information associated with a serial port like what I/O port and IRQ (Interrupt Request) a particular port is using.
- *badblocks*: is used to search badblocks on disk partition.
- *ctrlaltdel*: for setting function of ctrl-alt-del combination i.e. either to set hard (immediately reboots) or soft (first send Interrupt signal to init process and then reboot) system reset.
- *chkconfig*: for directly managing symlinks in /etc/rc[0-6].d.
- *debugfs*: a debugger to examine and change state of etx2 file system.
- *depmod*: use to handle dependency descriptions for loadable kernel modules, so that correct module dependencies will be available immediately while rebooting.

- *dump*: use to examine files on ext2 file system and to determine which files need to be backed up.
- *dump2fs*: prints super block and blocks group information for the file system present on devices.
- *fdisk*: manipulates partition table.
- *fsck*: checks and repairs file system.
- *fsck.ext2*: checks and repairs second extended file system.
- *fsck.minix*: it is file system consistency checker.
- *halt*: halts, reboot or poweroff the system.
- *hdparam*: to get or set hard disk parameters.
- *hwclock*: to query and set hardware clock.
- *ifconfig*: to configure network interfaces.
- *ifdown*: takes network interface down.
- *ifport*: selects transceiver type (auto, 10baseT, 10base2, etc) for a network interface.
- *ifup*: brings network interface up.
- *init*: Init is the parent of all processes. It creates and controls processes required by the system.
- *insmod*: to install a loadable module in the running kernel.
- *kerneld*: automatically loads kernel modules, rather than doing manually by insmod or modprobe utilities.
- *killall5*: sends a kill signal to all processes except to kernel threads and the processes in its own session.
- *lilo*: to install Linux boot loader.
- *mingetty*: is a minimal getty for use on virtual consoles.
- *mkbootdisk*: creates a standalone boot floppy for running system.
- *mke2fs, mkfs.ext2*: to create an ext2/ext3 file system on a disk partition.
- *mkfs*: to build a Linux file system.

- *mkfs.minix*: makes a Linux MINIX file system.
- *mkfs.msdos*: creates MSDOS file system under Linux.
- *mkinitrd*: creates initial ramdisk images for preloading modules.
- *mkswap*: sets up a Linux swap area on a device or in a file.
- *modinfo*: shows information about a Linux kernel module.
- *modprobe*: program to add and remove modules from the kernel.
- *netreport*: requests network management scripts to notify any network interface changes.
- *quotaon*: it turns file system quotas on.
- *restore*: restore files or file systems from backups made with dump.
- *runlevel*: reads system "utmp" file to locate runlevel record and then prints current and previous system runlevel.
- *swapon*: enable devices and files for paging and swapping.
- *tune2fs*: allows adjusting various tunable file system parameters on Linux ext2/ext3 file systems.
- *kernelcfg*: tool for kernel configuration.
- *netcfg*: tool for network configuration.
- *atrun*: run jobs queued for later execution.
- *crond*: daemon to execute scheduled commands.
- *dhcpd*: dynamic host configuration daemon, for assigning dynamic IP addresses.
- *edquota*: is a quota editor for editing user quotas.
- *groupadd, groupdel, groupmod*: commands for adding, deleting and modifying user groups.
- *in.identd*: identification server that sends identity of the owner of TCP connection to client.
- *in.rexecd*: it is remote execution server use for logging in machines remotely.

- *in.rlogind*: is remote login server that provides remote login facility with authentication based on privileged port numbers from trusted hosts.
- *in.rshd*: is remote shell server for remote login.
- *in.telnetd*: telnet server for remote login. It connects to the server at port number 23.
- *in.tftpd*: trivial file transfer protocol server for IPv4. It is used to support remote booting of diskless devices.
- *in.inetd*: it returns user information to a remote host that a user is requesting a service from.
- *klogd*: is a system daemon that listens, logs, processes and prioritize Linux kernel messages.
- *named*: update and create new users in batch.
- *ntsysv*: simple interface for configuring runlevels.
- *pwck*: to verify integrity of password file.
- *quotastats*: prints a report of quota system statistics gathered from the kernel.
- *setup*: initializes devices and file systems which are configured into the kernel and then mounts the root file system.
- *tcpd*: use to access control facility for internet services.
- *tcpdchk*: examines tcp wrapper configuration and reports all potential and real problems.
- *tcpdump*: is a packet sniffer and dumps traffic on a network.
- *tmpwatch*: removes files which haven't been accessed for a period of time.
- *useradd, userdel, usermod*: to add, delete or modify user accounts.
- *usernetctl*: allows a user to manipulate a network interface if permitted to do so.
- *vipw*: to edit password or group file.

4.3.4 Disable SUID bit from Ping, Traceroute, At

SUID bit allows the non-root users to run the programs with the root access. Disabling SUID permission from the programs is required because if a security weakness or vulnerability is found in these programs, then that vulnerability can be exploited to gain root-level access through any user account. "Ping" is used for testing network connectivity and must be used by root user only for troubleshooting the network. The "traceroute" utility is used to test network connectivity and is also useful for debugging network problems, but it is generally not necessary for non-privileged users, so it is suggested to remove SUID bit from both "ping" and "traceroute". The "at" is used for scheduling an individual task to run at a single later time. It has been found that "at" has many security loopholes and all necessary functions of "at" are found in "cron" so there is no need to retain root access for it.

4.3.5 Removing SUID bit from the r-tools

The r-tools have traditionally been used to make remote connections to other machines. They rely on IP address for authentication and transmit data in clear text, including password. Because of these insecurities, ordinary users should not be allowed to use the r-tools. It is recommended to remove the permissions on the r-tools so that non-root users cannot run them and the administrators must re-enable them when needed. This option will disable the "client" side of these tools, so that non-root users cannot use them to connect to other machines.

4.3.6 Disable r-protocols using IP-based Authentication

The r-tools (rlogin, rcp, rsh/remesh, etc) are insecure because they use IP-based authentication methods which can be easily fooled. IP-based authentication means that anyone with root access on 192.168.1.1 can have root access on 192.168.1.2, this was found useful, as it allows administrators to connect from one host to another without having to retype a password. The ".rhosts" file contains the names of the accounts and machines that are considered to be trusted. But, the problem with IP-based authentication is that an intruder can craft a fake packet which claims to be from a trusted user on a trusted machine. Since the r-tools rely entirely on IP addresses and remote username for authentication, a spoofed packet will be accepted as real. Rhosts can be restricted by modifying PAM files, removing execute permission from r-tools, and commenting out the services in inted.conf file. All these options will disable both the "client" and "server" sides of r-tools.

4.3.7 Enforce Password Aging

The default behavior of the operating system will disable an account when the password has not been changed in 99,999 days. This interval is too long to be useful. The script sets the default to 60 days. At some point before the 60 days have passed, the user will be prompted to change the password. At the end of the 60 days, if the password has not been changed, the account will be temporarily disabled. And this warning period will be at least 5 days long. All this can be done by changing /etc/login.defs. This measure keeps passwords fresh and also prevents inactive accounts from being attacked by system crackers.

4.3.8 Setting New Default Umask

The umask sets the default permission for the files that are created by the user. The default value for umask is 002 (everyone can read the files and the group members can alter them) or 022 (everyone can read the file but no one can write to them). This option in the script sets the default umask to 077 where no one on the system can read or write to the files.

4.3.9 Disallow Root Login on tty's 1 to 6

Root login on tty can be restricted, so that an admin must login with an ordinary user account and then use "su" command to become root. This can stop an attacker who has only been able to steal the root password from logging in directly. The attacker has to steal a second account's password to make use of the root password via the ttys.

4.3.10 Password Protect GRUB

To password protect GRUB is strongly recommended for general use workstations and servers which are not locked away in their room. If an attacker has physical access to these machines, s/he could get super-user access through the Grand Unified Bootloader (GRUB) command line. And if the GRUB prompt is password-protected, any user can reboot the machine normally, but only users with the password can pass arguments to the GRUB prompt.

4.3.11 Disable ctrl+alt+del

Disabling ctrl+alt+del rebooting is designed to prevent an attacker who has the physical access to the machine. The attacker can reboot the machine in a way that may potentially damages the file system. Thus, to maintain the integrity of

the file system is wholly necessary precaution, since having to repair/ignore the damage and wait for the file system checks may slow the attacker down.

4.3.12 Password Protect Single-User Mode

Anyone who can physically interact with the system can bring the machine up in "single user mode" where s/he is given root privileges and everyone else is locked out of the system. This doesn't require a password on most UNIX systems. So it is suggested to password-protect the single-user mode and there is no need to remember another password as the single-user mode will require the root password.

4.3.13 Limit System Resources

In case of Denial of Service attacks, the goal is not to gain access but to disrupt the normal operation of the computer. Effectiveness of such attacks can be limited by setting the limits on the resources available to each user. These attacks can be crippled by modifying /etc/security/limits.conf file. Modification will set the following initial limits on resource usage:-

- The number of allowed core files will be set to zero. Core files are useful for diagnosing system problems. They are very large files and can be exploited by an attacker to fill up the file system. They can also be used to tune vulnerability exploitation tools. And finally, an attacker might use the core file from a crashed program to obtain privileged data that was dumped by the program.
- Individual users will be limited to 150 processes each. This limit is more than enough for normal system usage, and will not bring down the machine's performance.

4.3.14 Disable Remote File Systems

It is strongly recommended to disable both network file systems: NFS (Network File System, common to most Unix variants) and SMB (Samba, which comes with most Linux distributions). NFS allows its host machine to export file systems onto other designated machines on the network but, NFS has a history of major security vulnerabilities, as well as being a clear-text protocol and relying on the presented username for authentication any data transferred by NFS can be monitored and may be tampered with by any other network machine. Also the transferred data includes file handles, which can be used to modify files. On the other hand, unlike NFS, Samba requires to authenticate users before reading or writing to files. But,

Samba is also a clear-text protocol and a shared file system; therefore it also raises severe security concerns.

4.3.15 Binding Web Server

If the user does not want to use the system as a web server, then it is recommended to deactivate the Apache web server immediately because the programs that require an Apache server installed but do not bind to port 80 will still be able to start their own instances of the web server. Therefore, the web server should be disabled if not required as it will be risky if the server be set to allow connections from the entire internet.

The Apache web server must be bind to the local interface in case the administrator don't want the web server to be accessible worldwide. And the web server must be bind to a specific IP address if the web server is built for the internal LAN access only.

4.4 Snapshots

In Figure 4.4 the shell script modifies kernel parameters and renames the root account as a guest account.

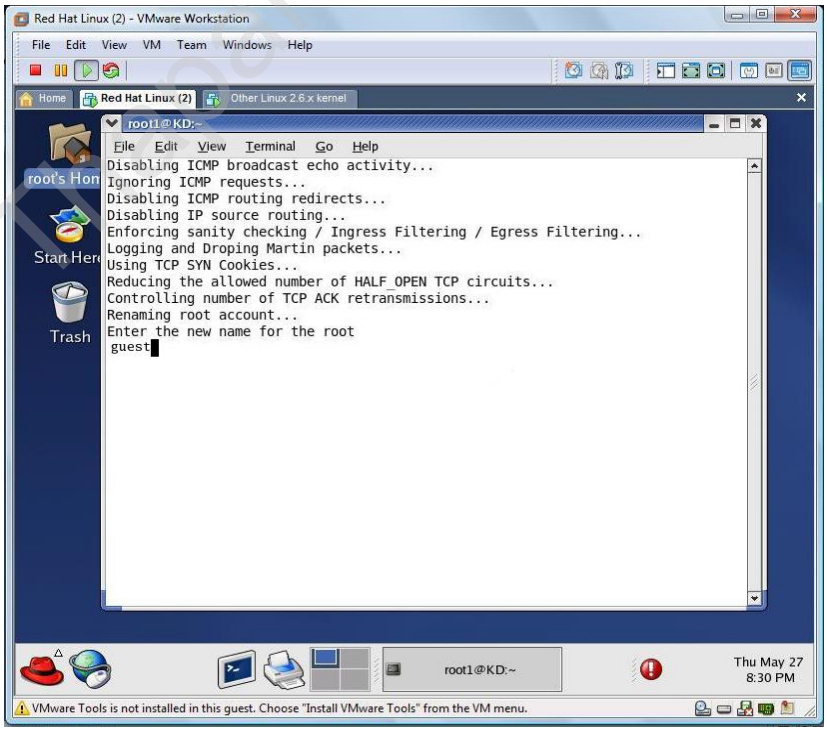

Figure 4.4: Shell Script Output1

In Figure 4.5 the shell script enforces strict permissions on critical utilities, removes SUID bit, disable all r-tools, imposes password aging and protects the GRUB.

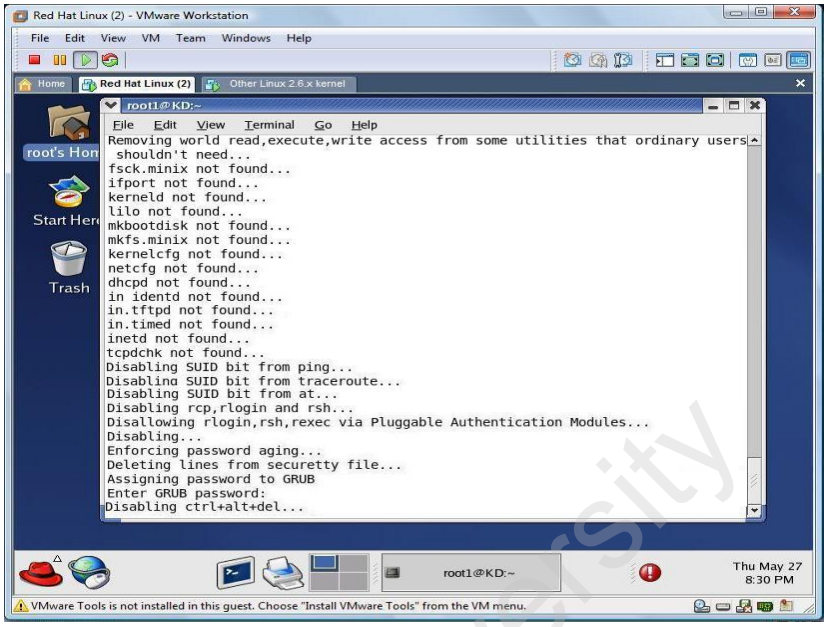

Figure 4.5: Shell Script Output2

In Figure 4.6 the script protects single-user mode, limits system resources, disables remote file systems and ristricts the Apache web server. In the end, it restarts system to save the changes done so far.

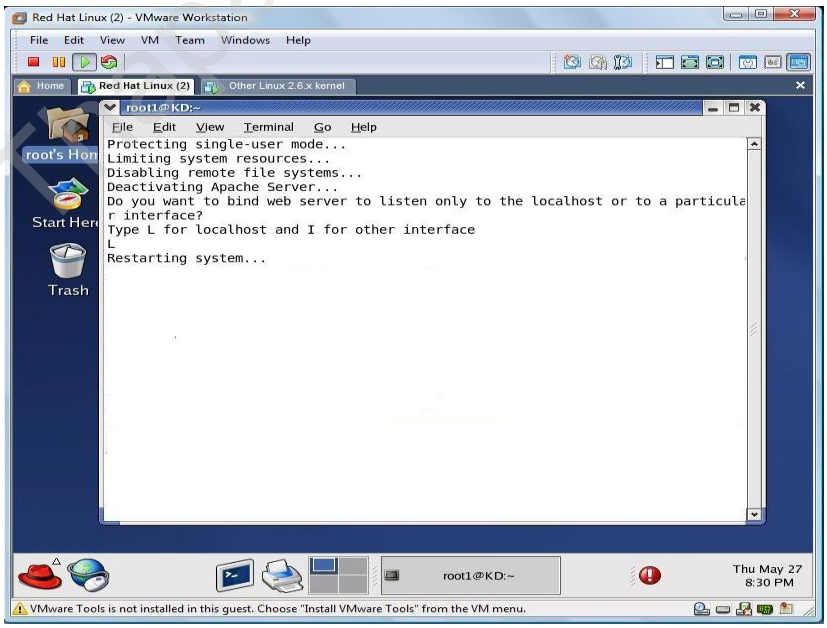

Figure 4.6: Shell Script Output3

4.5 Results with and without Hardening

A Linux system was scanned by Nmap 4.85 (an active fingerprinter) both before and after running the shell script. The difference in output of Nmap scans before and after hardening the Operating System is shown below.

Before hardening: When the system was not hardened against computer attacks, the Nmap effortlessly scanned it and correctly depicted the Operating System version running on that machine. Figure 4.7 depicts the output of Nmap scan before hardening the Linux Operating System.

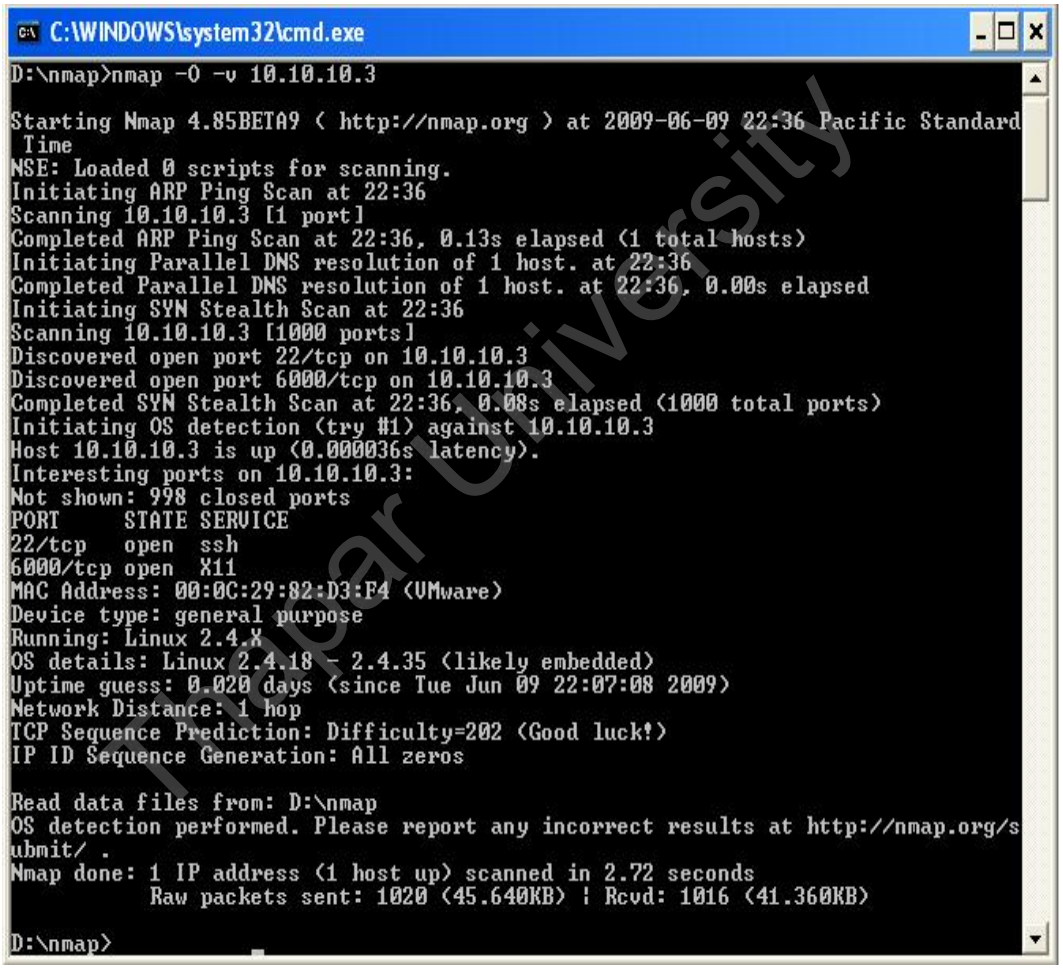

Figure 4.7: Before System Hardening

After hardening: After system hardening, the Nmap was not able to ping the system and therefore, could not scan it to identify the running Operating System version. The Nmap was unable to ping the target system because during the first phase of scanning, Nmap sends ICMP request messages to check whether the target system is alive or not and if the system is alive, then the crafty packets are sent

to reveal the Operating System Fingerprint. But, after running the shell script the system blocked the ICMP request messages that were coming from Nmap and thus, shielded the system from being recognized. Figure 4.8 shows the output of Nmap scan after the Operating System hardening.

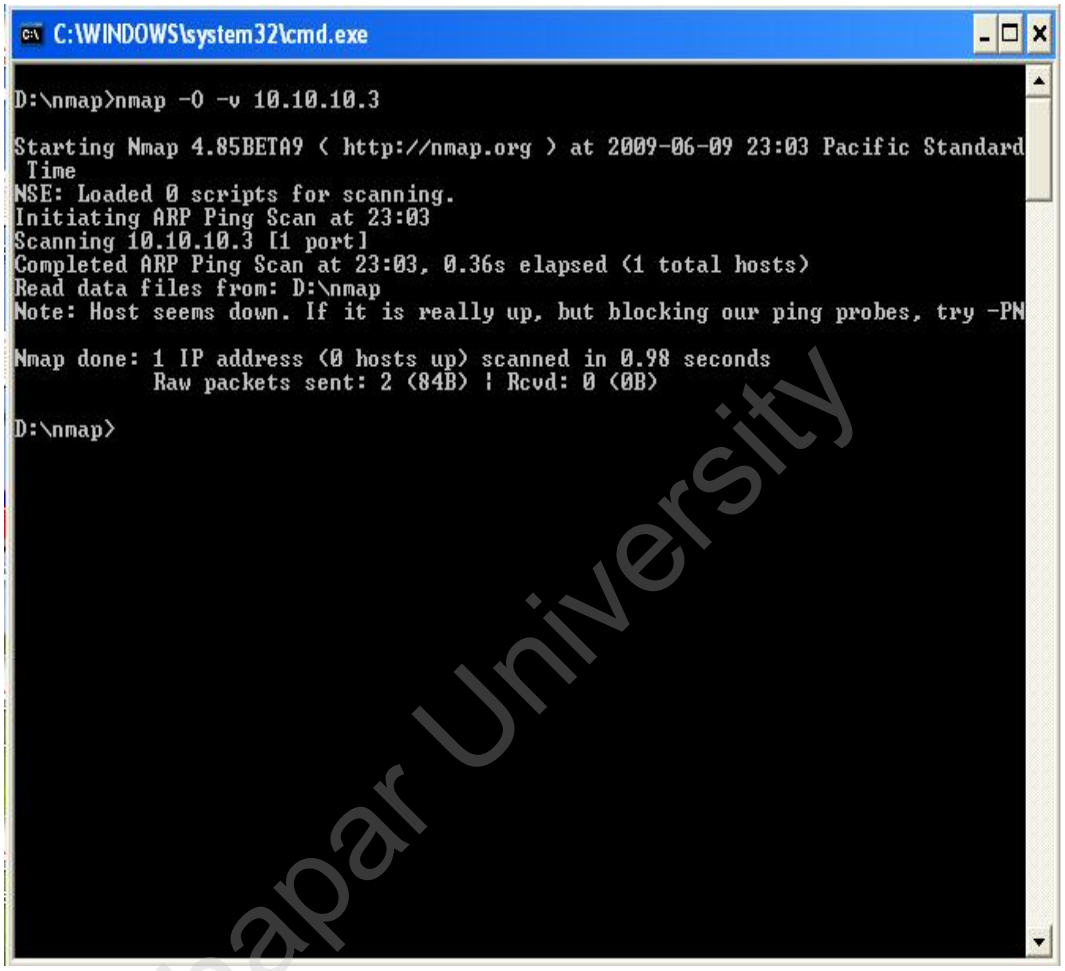

Figure 4.8: After System Hardening

Chapter 5

Conclusions and Future Scope

5.1 Conclusions

Operating System Fingerprint helps in mapping the remote networks and in determining vulnerabilities that might be present on the remote system to exploit. Learning remote Operating System version can be an extremely valuable network reconnaissance tool for the Security Professionals ("White-hats") because many security holes are dependent on the Operating System version. And it is equally important for the Crackers ("Black-hats") because they can easily attack the system with the exploits which the Operating System advertises.

The attacker often searches for "open windows" in the target machine like IP addresses in use, Operating Systems in use, Ports in use, and Services on that ports, so that the target can be probed further to find its weaknesses. By probing a wide variety of hosts ahead of time, the attackers can prepare a hit-list to use as soon as the Operating System exploit is released. This allows attackers not to waste their time trying Windows exploits against Linux machines.

Operating System fingerprinting can be accomplished passively by sniffing network packets traveling between hosts, actively by sending carefully crafted packets to the target machine and analyzing the response, or through non-technical means. The active and passive fingerprinting techniques have their own pros and cons based upon speed, accuracy, stealth and power to fool upcoming masking techniques.

The proposed solution modifies certain TCP/IP features of an Operating System to make it more secure. The customized behavior of TCP/IP stack reveals less information about their host's identity and thus, can easily fool the known

fingerprinters. The presented idea aims at confusing the fingerprinting tools like Nmap, and not just attempting to lie about their Operating System.

5.2 Future Work

The proposed security framework is designed to harden only Linux systems to protect against various computer threats and to prevent computer identification information leakage. However this solution can be further modified to support other platforms like Windows, Solaris, and various Linux clones.

In this thesis, design and development was concentrated only on the shell scripting and kernel hookups. New security framework devices like Unified Threat Management (UTM) systems can be augmented with the given methodology.

The working of the security solution presented in the thesis is more cryptic and requires network administrator's intervention in order to deploy. It could be made more flexible and user friendly so that even a home user can utilize the outcome of this research.

References

[1] Mohammad Al-Jarrah and Abdel-Karim R. Tamimi. *A Thin Security Layer Protocol over IP Protocol on TCP/IP Suite for Security Enhancement*. IEEE Innovations in Information Technology, Nov 2006.

[2] David Barroso Berrueta. *A practical approach for defeating Nmap OSFingerprinting, Free Software Foundation*. Free Software Foundation, Inc., 2003.

[3] CISCO. *What is Network Security*. CISCO Systems Inc. http://www.cisco.com/cisco/web/solutions/small_business/resource_center/articles/secure_my_business/what_is_network_security/index.html.

[4] Justin Clarke and Nitesh Dhanjani. *Network Security Tools*. O'Reilly, April 2005.

[5] Matt Curtin. *Introduction to Network Security*. Kent Information Services, Inc.

[6] Dave Dittrich. *Some TCP/IP Vulnerabilities: Weaknesses, attack tools, defenses*. Dec 1999. http://staff.washington.edu/dittrich/talks/agora/.

[7] Ahmed Eddaoui and Abdellatif Mezriou. *An Active Network Approach for Security Management*, volume 6,No.5B. IJCSNS International Journal of Computer Science and Network Security, May 2006.

[8] Behrouz A Forouzan. *Data Communications and Networking*. McGraw-Hill Science/Engineering/Math, 3 edition, Aug 2003.

[9] Barbara Fraser. *Security Incidents, Threats, and Vulnerabilities*. CERT Software Engineering Institute Carneige Mellon University, February 1998. http://www.apricot.net/apricot97/apII/Presentations/ SecurityIncidentsThreats / Hsld010.htm.

[10] Simson Garfinkel and Gene Spafford. , *Practical UNIX Internet Security*. Second edition, April 1996. http://docstore.mik.ua/orelly/networking/puis/ch22_01.htm.

[11] Kapil Kumar Gupta, Baikunth Nath (Sr. Member IEEE), and Kotagiri Ramamohanarao. *Network Security Framework*, volume 6 No.7B. IJCSNS International Journal of Computer Science and Network Security, 2006.

[12] Red Hat. *Red Hat Linux Security Guide*. Red Hat,Inc., 2002. http://www.redhat.com/docs/manuals/linux/RHL-9-Manual/pdf/rhl-sg-en-9.pdf.

[13] L. Todd Heberlein, Gihan V. Dias, Karl N. Levitt, Biswanath Mukherjee, Jefl Wood, and David Wolber. *A Network Security Monitor*. IEEE, 1990.

[14] HP. *ProCurve ProActive Defense: A Comprehensive Network Security Strategy*. Hewlett-Packard Development Company, February 2007. http://www.hp.com/rnd/pdfs/ProCurve_Security_paper_final.pdf.

[15] Peyman Kabiri and Ali A. Ghorbani. *Research on Intrusion Detection and Response: A Survey*, volume 1,No.2. IEEE International Journal of Network Security, September 2005.

[16] Surbhie Kalia and Maninder Singh. *Masking approach to secure system from Operating System Fingerprinting*. IEEE Tencon05, Nov 2005.

[17] Susan C. Lee and Lauren B. Davis. *Learning from Experience: Operating System Vulnerability Trends*. IEEE Computer Society, Jan—Feb 2003.

[18] Thomas A. Longstaff, James T. Ellis, Shawn V. Hernan, Howard F. Lipson, Robert D. MCmillan, Linda Hutz Pesante, and Derek Simmel. *Security of the Internet*, volume 15. CERT Software Engineering Institute Carneige Mellon University, February 1998. http://www.cert.org/encyc_article/.

[19] Microsoft. *The Registry Keys and Values for the System Restore Utility*. Microsoft, 2006. http://support.microsoft.com/kb/295659.

[20] George S. Oreku and Jianzhong Li. *Rethinking E-commerce Security*, volume 1. IEEE Computational Intelligence for Modelling, Control and Automation, 2005 and International Conference on Intelligent Agents, Web Technologies and Internet Commerce. (CIMCA-IAWTIC'05), November 2005.

[21] Dea-Woo Park. *A study about dynamic intelligent network security systems to decrease by malicious traffic*, volume 6,No.9B. IJCSNS International Journal of Computer Science and Network Security, September 2006.

[22] Bente Petersen. *Intrusion Detection FAQ: What is p0f and what does it do?* SANS Institute, 2009. http://www.sans.org/resources/idfaq/p0f.php.

[23] Niels Provos. *A Virtual Honeypot Framework*, volume 13. 13th conference on USENIX Security Symposium, 2004. http://www.citi.umich.edu/u/provos/papers/honeyd.pdf.

[24] Jaziar Radianti and Jose. J. Gonzalez. *Understanding Hidden Information Security Threats: The Vulnerability Black Market*. IEEE 40th Hawaii International Conference on System Sciences, 2007.

[25] Shaunige. *Tcp/ip Vulnerabilities And Weaknesses*. Aug 2003. http://www.governmentsecurity.org/forum/index.php?showtopic=1753.

[26] Matthew Smart, G. Robert, and Malan Farnam Jahanian. *Defeating TCP/IP Stack Fingerprinting*, volume 9. 9th USENIX conference on Security Symposium Paper, 2000.

[27] Ryan Spangler. *Analysis of Remote Active Operating System Fingerprinting Tools.* May 2003. http://www.packetwatch.net/documents/papers/osdetection.pdf.

[28] William Stearns. *Manual Reference Pages - P0F (1).* IronGeek, 2009. http://www.irongeek.com/i.php?page=backtrack-3-man/p0f.

[29] Ballardie T. and Crowcroft J. *Multicast-specific security threats and countermeasures.* IEEE Network and Distributed System Security, February 1995.

[30] Greg Taleck. *Ambiguity Resolution via Passive OS Fingerprinting.* Springer-Verlag Berlin Heidelberg, 2003. http://www.itsec.gov.cn/webportal/download/2003-Ambiguity Resolution via Passive OS Fingerprinting.pdf.

[31] Andrew S. Tanenbaum. *Computer Networks.* Prentice Hall PTR, 4, illustrated edition, Aug 2002.

[32] Chris Trowbridge. *An overview of Remote Operating System Fingerprinting.* SANS Institute, July 2003. http://www.sans.org/reading_room/whitepapers/testing/1231.php.

[33] Bogazici University. *What is Network Security.* http://www.cc.boun.edu.tr/network_security.html.

[34] Craig Valli. *Honeyd - A OS Fingerprinting Artifice.* 1st Australian Computer, Network Information Forensics Conference, Nov 2003.

[35] Rod Wallace. *Addressing the new security challenges of real-time, multimedia, and mobile networks.* The Nortel Technical Journal Issue 3, Feb 2006.

[36] David Watson, Matthew Smart, G. Robert Malan, and Farnam Jahanian. *Protocol Scrubbing Network Security through Transparent Flow Modification*, volume 12,No.2. IEEE/ACM Transactions on Networking, April 2004.

[37] Dr.George B. White. *The Community Cyber Security Maturity Model.* IEEE 40th Hawaii International Conference on System Sciences, 2007.

[38] Michal Zalewski. *The new p0f:2.0.8.* Sep 2006. http://lcamtuf.coredump.cx/p0f.shtml.

List of Papers Published

1. Ratinder Kaur, Maninder Singh, "Hardening Operating System Identity by Customized Masking Techniques", in ICCNT 2009 (International Conference on Computer and Network Technology), Chennai, India (June 27-29, 2009). (Status: Accepted)