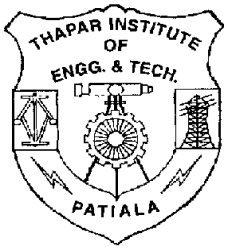
DESIGN AND DEVELOPMENT OF FINGERPRINTING SECURITY FRAMEWORK

A Thesis

Submitted in partial fulfillment of the
requirement for the award of degree of

Master of Engineering
In
Software Engineering



Under the Supervision of
Maninder Singh
Assistant Professor
Computer Science and Engineering Department
Batch 2003-2005

Submitted By
Surbhie Kalia
(8033122)

Computer Science & Engineering Department
Thapar Institute Of Engineering & Technology
(Deemed University), Patiala-147004 (India).
May 2005

ABSTRACT

As the scale of networking increased, the threats and security concerns also multiplied. Network is a concept that started to exist after the standardization of the protocols and the RFCs. Once the Network analysis method and techniques were introduced since early 90's, the security is under constant guard. Hence it's important to know the current techniques used by both the administrators and attackers to identify the resources and system vulnerabilities posted on network.

Remote OS Fingerprinting is a recent development on the Internet. The ability to remotely determine, with high accuracy, the Operating System of a remote host on the Internet is possible using tools and wide assortment of technologies. The implications of this technology are perhaps not yet fully understood; however, it is seen as enough of a threat that strategies are currently being developed to attain, analysis, deploy, prevent or spoof OS Fingerprinting. As the cause may call for the usage of fingerprinting is only understood once the data is collected and analyses.

".....it's not easy to defend one self from others attack until we know Who we are? What we Offer? And What are the vulnerabilities....so before other do that IDENTIFY ASSESRTS"

Technology that goes behind remote system operating system fingerprinting is under constant development and so are the countermeasures for it. The classical fingerprinting techniques were exploiting the vendor specific weakness existing in the operating system and the defensive techniques that were engaged consisted of fooling and hiding from the attackers, like banner hiding, change of default setting in common services and responses etc.

But with time the level of sophistication and depth of exploitation increased. The attackers started the use of weakness, default and specific behaviour of the network-transport protocols used by the remote system to attach itself to Internet and Intranet. This lead to accuracy of prediction and easy compromise of the system to fingerprinting tools. The attackers were now conveniently able to launch further attacks once they know the operating system running on the remote system.

Though the RFC’s and the standardization of protocols kept in mind the level of security from common threats, but did not have the relevant content when up against the Fingerprinting. So the basic security from fingerprinting can be achieved by employing a protective security framework. The steps that should be taken are: identifying security level, default behaviour of protocol are studied, basic technique of fingerprinting analysed and then masking of operating system by either hiding or by morphing.

DECLARATION

I hereby certify that the work which is being presented in the thesis entitled, ***"Design and Development of Fingerprinting Security Framework"*** *in* partial fulfillment of the requirements for the award degree of Master of Engineering in Software Engineering at Computer Science and Engineering Department of Thapar Institute of Engineering and Technology (Deemed University), Patiala, is an authentic record of my own work carried out under the supervision of Maninder Singh.

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

Surbhie Kalia+.

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

Mr. Maninder Singh
Assistant Professor
Computer Science and Engineering Department
Thapar Institute of Engineering and Technology
PATIALA- 147004

Countersigned by

Mr. R.S Salaria
Head
Computer Science and Engineering Department
Thapar Institute of Engineering and Technology
PATIALA- 147004

Dr. D. S. Bawa
Dean Of Academic Affairs
Thapar Institute of Engineering and Technology
PATIALA- 147004

ACKNOWLEDGMENT

It is proud privilege, nobility and an panorama for me to work under the subversion of my esteemed teachers and guide **Mr. Maninder Singh**, Assistant Professor, Computer Science & Engineering Department, Thapar Institute of Engineering and Technology, Patiala. This work would not have been possible without his able guidance, constant support, constructive criticism and painstaking effort inspired, provide and help me to conduct this research work in the present form.

I also take the opportunity to thank **Mr. R. S. Salaria,** Assistant Professor & Head**,** Computer Science and Engineering Department, TIET, Patiala, and the entire faculty and staff for their help, inspiration and moral support, which went a long way in successful completion of my Thesis.

I wish to express my indebtedness to my parents who have been a constant source of love and encouragement.

Finally I would like to thank God for not letting me down at the time of need and shower my way with his blessing so that no effort of mine does no go wasted. Amen.

Surbhie Kalia
(8033122)

TABLE OF CONTENTS

LIST OF FIGUERS:

LIST OF TABLES:

ORGANIZATION OF THESIS

The first chapter is an introduction to the networking, its enabling architecture, the network –transport protocols, the current scenario of security and threats, the concept of remote system analysis and classification leading to Os fingerprinting. Followed by second chapter conceptualising Operating System Fingerprinting in details. The third and fourth chapters are elucidation the two styles of OS Fingerprinting Passive fingerprinting and Active fingerprinting. Fifth chapter explain the general phases of Fingerprinting in order to understand the current technology being used by active fingerprinting tools .Following this two tools based on different Fingerprinting techniques are explained and exemplify. The design and implementation step to form security framework against these tools as a countermeasures follows. Concluding the work by summarizing what is explained and providing with further scope is done in the sixth chapter.

CHAPTER 1
INTRODUCTION

How secure is a software system? How secure does a system need to be? By how much can security be improved by putting safeguards into place?

The answer to these questions is no more subjective. “ISOLATION” in a “PHYSICALLY-GUARDED” environment is the complete solution to any known and unknown attack and potential threat to our software systems. The connection to network both trusted and non-trusted networks render us to security problems.

The foundations of computer networking is explained, then move on to an overview of some popular networks. Following that, more in-depth look at OSI model and TCP/IP stack, the network protocol suite that is used to run the Internet and many intranets.

Once covered this, some of the threats that managers and administrators of computer networks need to confront, the concept of *REMORT ANALYSIS OF COMPUTER SYSTEM* leading us to *OPERATING SYSTEM FINGERPRINTING*.

1.1. NETWORK AND INTERNET

A ``**network**'' has been defined as ``any set of interlinking lines resembling a net, *a network of roads* || an interconnected system, *a network of alliances*." This definition suits our purpose well: a computer network is simply a system of interconnected computers. *How* they're connected is irrelevant, and as we'll soon see, there are a number of ways to do this.

A “**LAN** (local area network)” is a group of computers and network devices connected together, usually within the same building. By definition, the connections

must be high-speed and relatively inexpensive (e.g., token ring or Ethernet). Most Indiana University Bloomington departments are on a LAN.

A "**MAN** (metropolitan area network)" is a larger network that usually spans several buildings in the same city or town. The IUB network is an example of a MAN.

A "**WAN** (wide area network)" is, in comparison, not restricted to a geographical location, although it might be confined within the bounds of a state or a country. A WAN connects several LANs, and may be limited to an enterprise (a corporation or an organization) or accessible to the public. The technology is high-speed, and is relatively expensive. The Internet is an example of a worldwide public WAN.

The "**Internet**" is a collection of thousands of networks linked by a common set of technical protocols which make it possible for users of any one of the networks to Communicate with, or use the services.

1.1.1 ISO - OSI Model

Understanding the function of each layer in OSI model is instrumental in understanding data communication within Local, Metropolitan or Wide networks. The ISO (International Standards Organization) has created a layered model, called the OSI (Open Systems Interconnect) model, to describe defined layers in a network operating system. The purpose of the layers is to provide clearly defined functions that can improve Internetwork connectivity between "computer" manufacturing companies. Each layer has a standard defined input and a standard defined output.

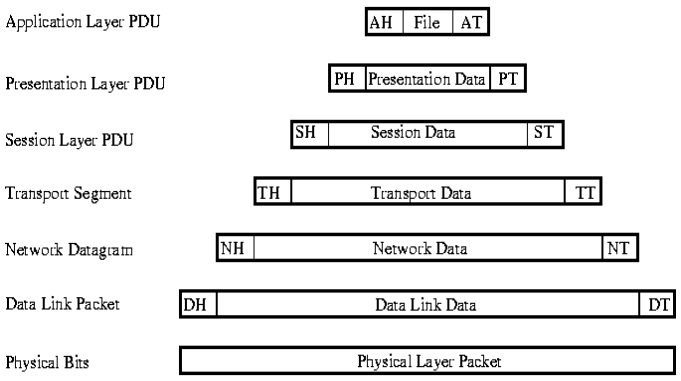


Figure 1.1: Protocol data unit(PDU) for each layer

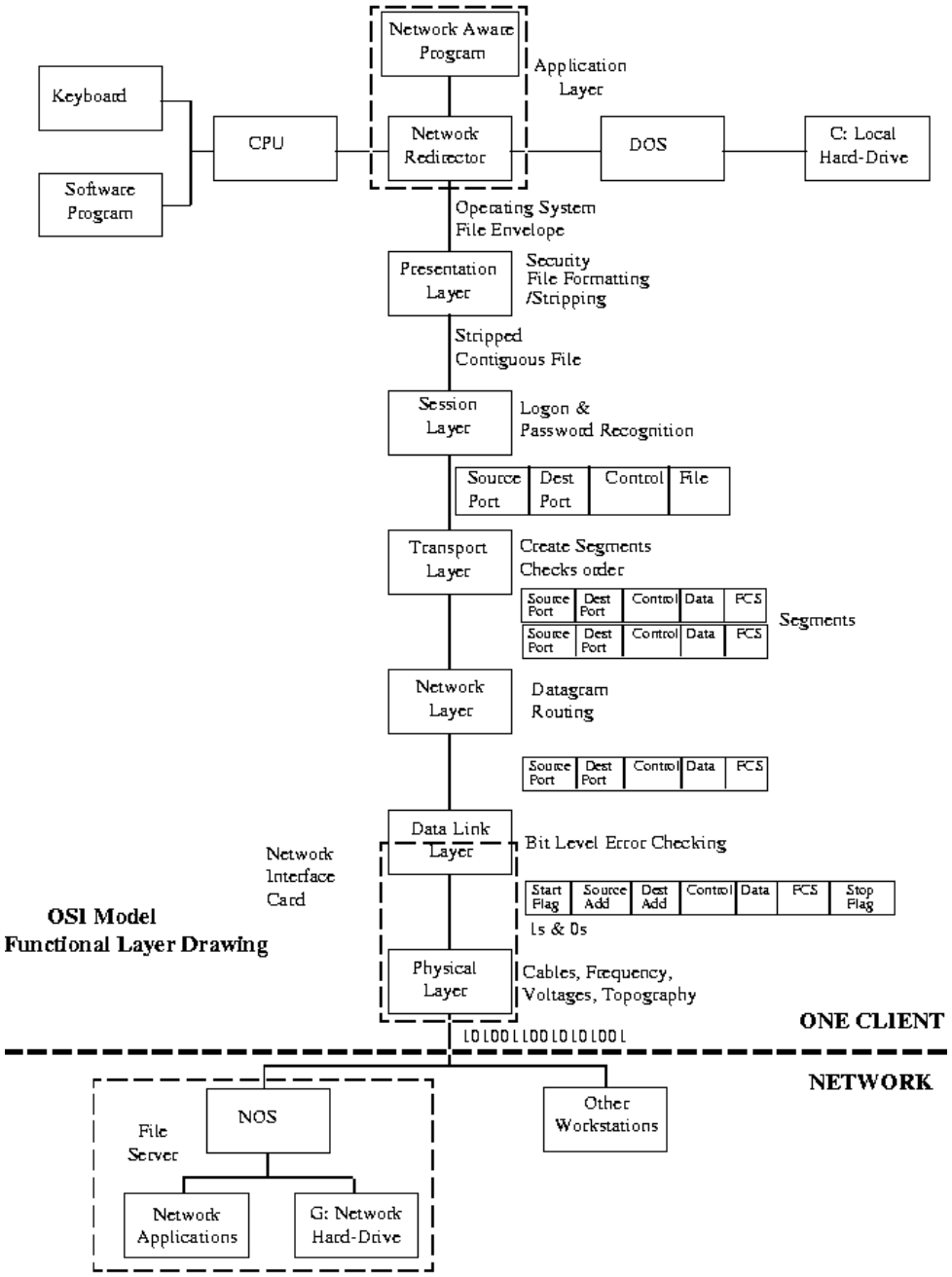


Figure 1.2 : OSI Model Functional Drawing

1.1.2 TCP/IP Stack vs. OSI model

The TCP/IP model uses four layers that logically span the equivalent of the top six layers of the OSI reference model; this is shown in Fig.1.3 (The physical layer is not

covered by the TCP/IP model because the data link layer is considered the point at which the interface occurs between the TCP/IP stack and the underlying networking hardware.) The following are the TCP/IP model layers, starting from the bottom.

The TCP/IP architectural model has four layers that approximately match six of the seven layers in the OSI Reference Model. The TCP/IP model does not address the physical layer, which is where hardware devices reside. The next three layers—*network interface*, *internet* and *(host-to-host) transport*—correspond to layers 2, 3 and 4 of the OSI model. The TCP/IP *application* layer conceptually "blurs" the top three OSI layers. It's also worth noting that some people consider certain aspects of the OSI session layer to be arguably part of the TCP/IP host-to-host transport layer.

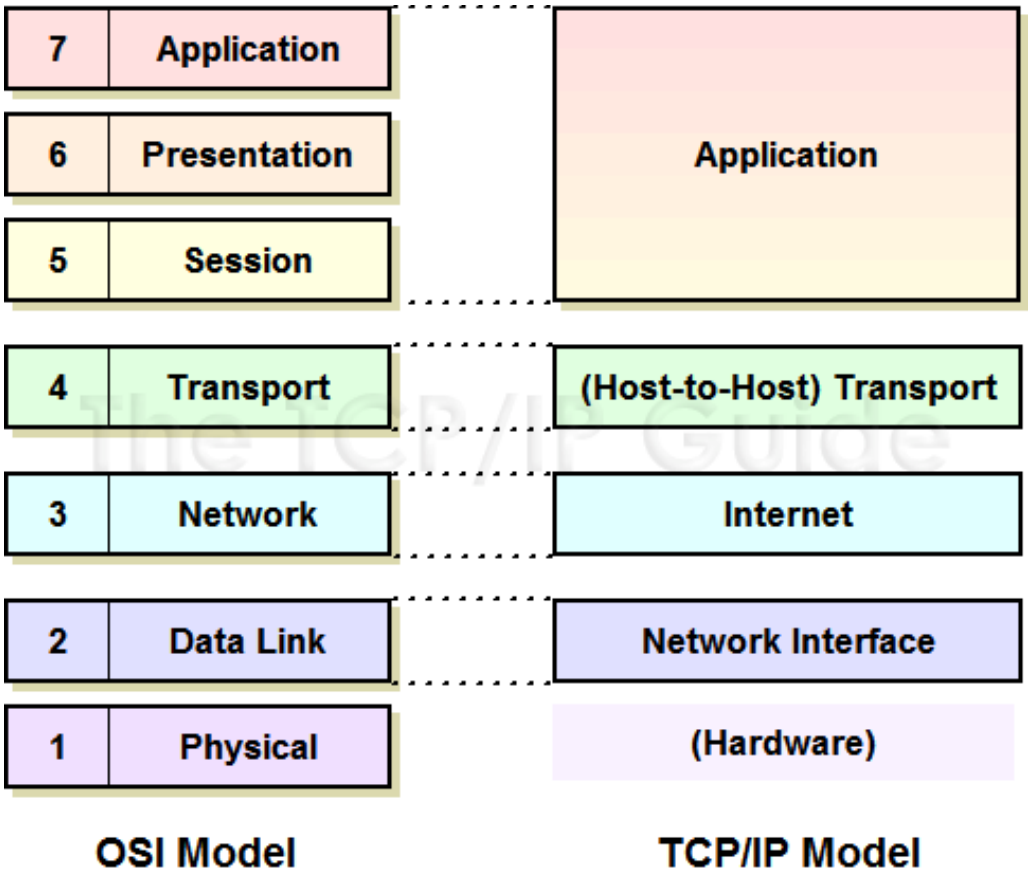


Figure 1.3: OSI Reference Model and TCP/IP Model Layers

1.1.3 TCP/IP Protocols

TCP is a standard protocol with STD number 7. The Transmission Control Protocol (TCP), documented in RFC 793, makes up for IP's deficiencies by providing reliable, stream-oriented connections that hide most of IP's shortcomings. The protocol suite gets its name because most TCP/IP protocols are based on TCP, which is in turn based on IP. TCP and IP are the twin pillars of TCP/IP.

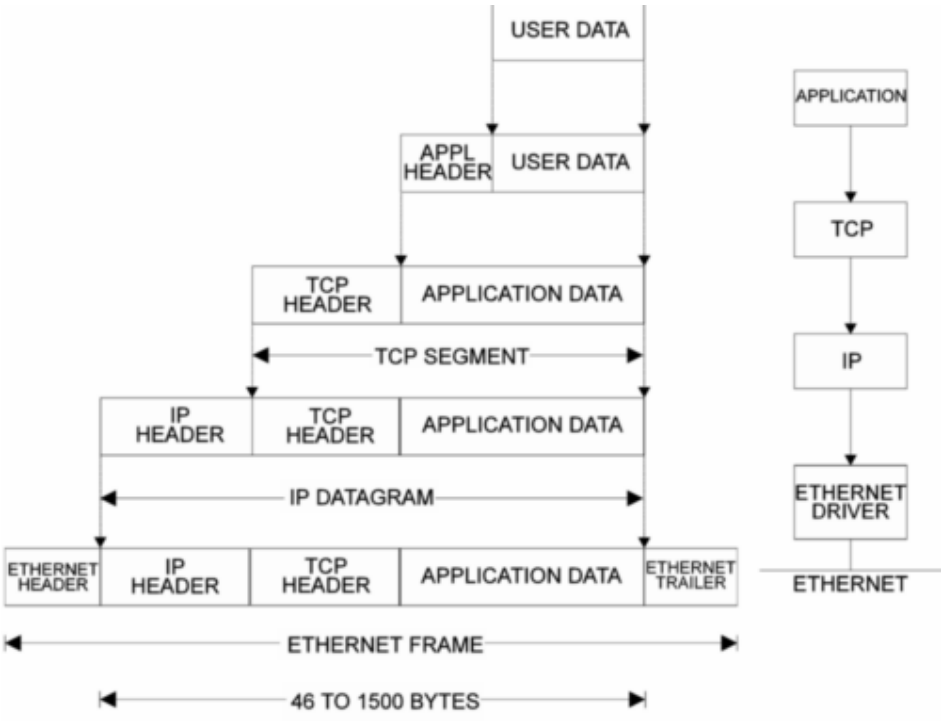


Figure 1.4: Encapsulation of data in TCP/IP stack

Before sending any data between two computers TCP first breaks that data into small data units called a packet. If any packet fails to reach its destination then TCP sends it again until it gets the data at destination.

TCP Header

Every unit of data transferred by TCP has a TCP header. TCP header contains all the details which help to keep TCP connections in order, to reassemble the packet at other end, and to detect any transmission errors. Here is TCP header format.

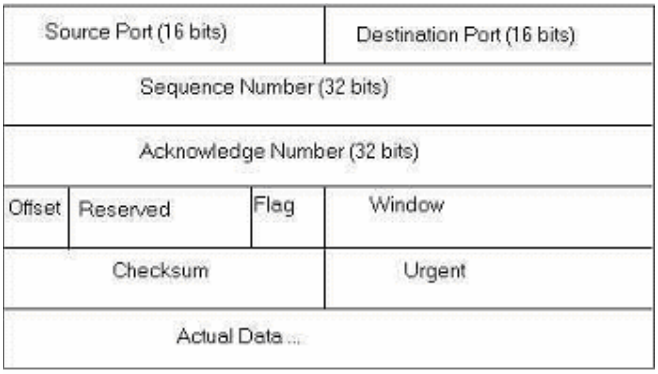


Figure 1.5: TCP Header

A TCP segment consists of a TCP header optionally followed by data. A TCP header includes six flag bits. One or more of them can be turned on at the same time:

- SYN Synchronize sequence numbers to initiate a connection.
- FIN The sender is finished sending data.
- RST Reset the connection.
- URG The urgent pointer is valid.
- ACK The acknowledgment number is valid
- PSH The receiver should pass this data to the application as soon as possible.

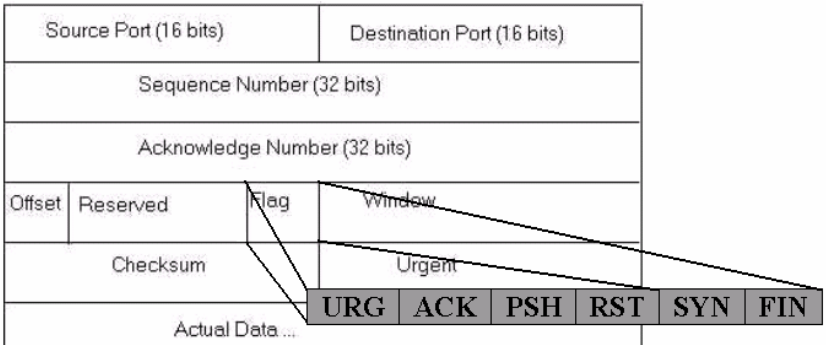


Figure 1.6: TCP Flags

IP Header

IP header is responsible for telling the TCP/IP that where in coming packets suppose to go. Header Checksum field makes sure that IP header does not get damaged and the information is not corrupted. Protocol field talks to IP on receiving computer and makes sure about receiving protocol is TCP or UDP. Time to live field is used to

control the life lifetime of a packet. Each time the packet passes through a system, the time-to-live field's value is decremented. When it reaches zero, the packet is destroyed. Two fields are for IP addresses of the source and the destination machines. As packet jumps from network to network, IP uses this information to decide where the packet should be sent next. Here is IP header:

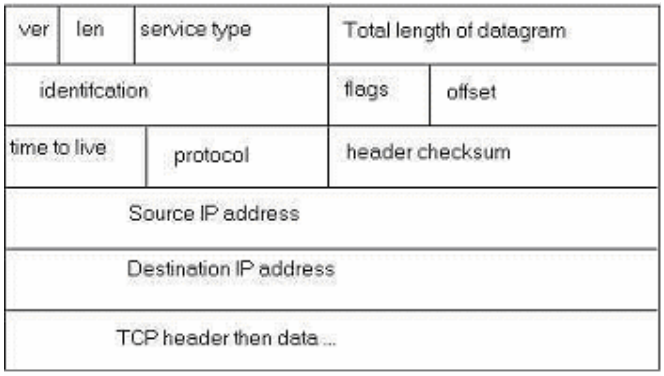


Figure 1.7: IP Header

TCP functions by opening connections to a remote host and is thus connection-oriented. TCP maintains status information regarding the connections it makes and is therefore a reliable protocol. A TCP connection is identified by the IP addresses and virtual port numbers used by both ends. During communication, additional numbers are used to keep track of the order or sequence number which indicates what order the segments of data should be reassembled. Finally, a maximum transmission size is constantly being negotiated via a fallback mechanism called windowing. The combination of port numbers, sequence numbers and window sizes constitutes a connection, or pipe.

To establish a connection, TCP uses the three-way handshake. This three way handshake will only be completed in one direction even if both sides initialize connections at the same time.

ESTABLISHING CONNECTIONS: The Three Way Handshake

Below is a description of the TCP 3-way handshake process. (Refer Table1.1)The variable "t" is used to refer to a series of points going forward in time.

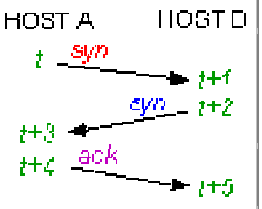
Time	Event	DIAGRAM
t	**Host A sends** a TCP **SYN**chronize packet to Host B	
t+1	Host B receives A's **SYN**	
t+2	**Host B sends** it's own **SYN**chronize	
t+3	Host A receives B's **SYN**	
t+4	**Host A sends ACK**nowledge	
t+5	Host B receives **ACK**. ***TCP connection is established.***	

Table 1.1: Three-way TCP/IP handshake

TCP knows the state of a connection by using the SYNchronize and ACKnowledge messages when establishing a connection. TCP can break up a message, transmit the pieces and reassemble them even if they are received out of order. Once connected, TCP hands off to the application for data transfer and communication. TCP always knows the state of the connection.

ICMP

Control Message Protocol, an extension to the Internet Protocol (IP) defined by RFC 792. ICMP supports packets containing error, control, and informational messages. The PING command, for example, uses ICMP to test an Internet connection. ICMP messages, delivered in IP packets, are used for out-of-band messages related to network operation or misoperation. Of course, since ICMP uses IP, ICMP packet delivery is unreliable, so hosts can't count on receiving ICMP packets for any network problem

Figure 1.8: ICMP Header

ICMP provides simple error reporting, diagnostics, and limited configuration for IP hosts. It is logically on the IP layer (part of layer 3), but ICMP packets are

encapsulated by IP as shown. Like IP, it is connectionless and unreliable The error reporting messages copy the offending packet's

– IP header and the first 8 bytes of its TCP or UDP header

– Or, the entire packet so that the ICMP message is < 576 bytes

(RFC1812)

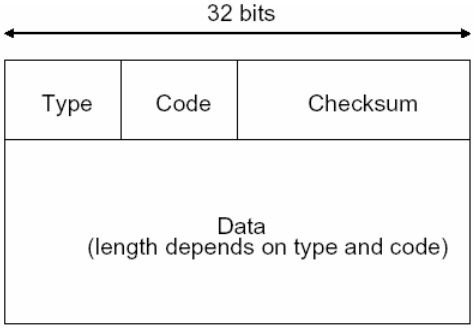


Figure 1.9: ICMP Format

Format and Fields

ICMP has a fixed "header"portion of 4 bytes:

– Type

_ Type of message

– Code

_ Subtype of message

– Checksum

1's compliment computed over entire ICMP message (except for the checksum field itself, which is set to 0).The rest of the ICMP data depends on the type and code, and is tailored to a specific purpose Can be simple or complex

1.2 BASIC APPROACH TOWARDS SECURITY

As long as the computer is in isolation only security seeks out by it is that from physical tampering, data extraction and basic virus from infected files. But the number of attacks, threats and their sources intensify when they are part of network, in accordance to the scale of connectivity.

This guide is written to provide basic guidance in developing a security plan for your site. One generally accepted approach is to include the following steps:

(1) Identify what is to be protected.
(2) Determine what to protect it from.
(3) Determine how likely the threats are.
(4) Implement measures which will protect the assets in a cost-effective manner.
(5) Review the process continuously and make improvements each time a weakness is found.

With the current worldwide emphasis on e-business, the development of sound network security policies for private networks connected to the Internet is becoming of increasing importance. With the increasing dependence of private organizations on information networks, the security of such networks is becoming essential, especially with the emergence of e-business over Intranets, Extranets, and the Internet. Security challenges to these networks have various undesirable business impacts on organizations, such as: business embarrassment, financial loss, degradation of competitiveness, and legal problems. Therefore security policies need to be emphasized, so that such challenges, together with their undesired consequence, can be avoided.

The rate at which the network traffic and number of host increasing, security -both awareness and need to security is also increasing.

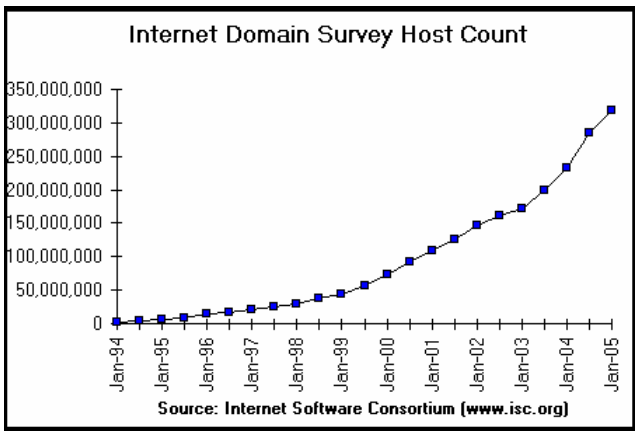


Figure 1.10: Growth of users on Internet

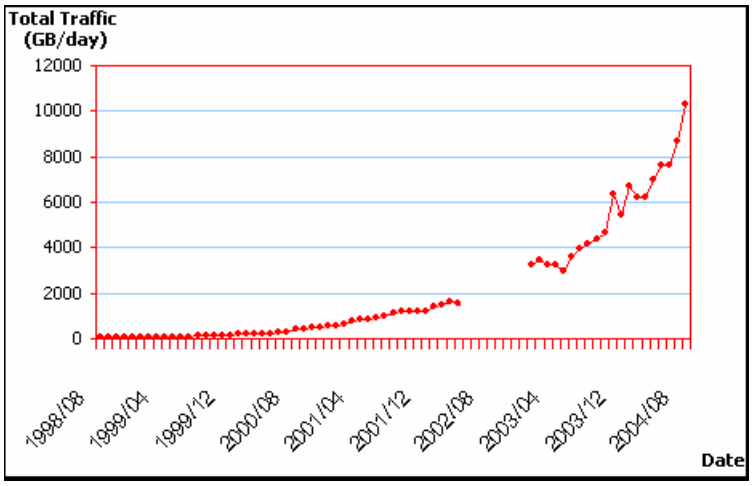


Figure 1.11: Total traffic in GB/Day

—Security in ISO Vocabulary—

In its information processing vocabulary, ISO defines the term computer system security as the technological and administrative safeguards established and applied to data processing to protect hardware, software, and data from accidental or malicious modifications, destruction or disclosures.4 This definition illustrates that IT security has four main dimensions.

• The first is concerned with the object that needs to be protected; that is, hardware, software and data.

• The second is associated with the undesired effect of security challenges on the object, that is, destruction and disclosure. Destruction applies to hardware, software and data, while disclosure applies to data and software. Of course, software can be viewed as data when it is not in action.

Together, the above ISO definitions identify network security according to a broad scope of integrated issues. However, additional security challenges need to be incorporated within the identified scope. These additional challenges are associated with the flow of data through the network, and are concerned with the following:

• Undesired external data streams flowing from the Internet to an Extranet, or from the Internet and Extranets to an Intranet, represent security challenges to the policies of the Extranets or the Intranets concerned. The same also applies to private internal data streams that should not flow beyond the limits of their Extranets or Intranets.

• Large volumes of data flow through the network may cause performance problems, or even denial of service. Such volumes also challenge network security. In addition,

unsecured channels where desired data streams may flow represent another security challenge.

In addition to the above challenges, accidental and malicious disasters such as fire, earthquakes, floods, and other similar events represent important security challenges that need to be taken into account. These challenges have been addressed by publications concerned with contingency or disaster recovery planning. By presenting the ISO definitions, and the additional network security challenges, this section has provided a general view of the concept and dimensions of network security. The question now is concerned with the need for the give taxonomy of security threats for networks.

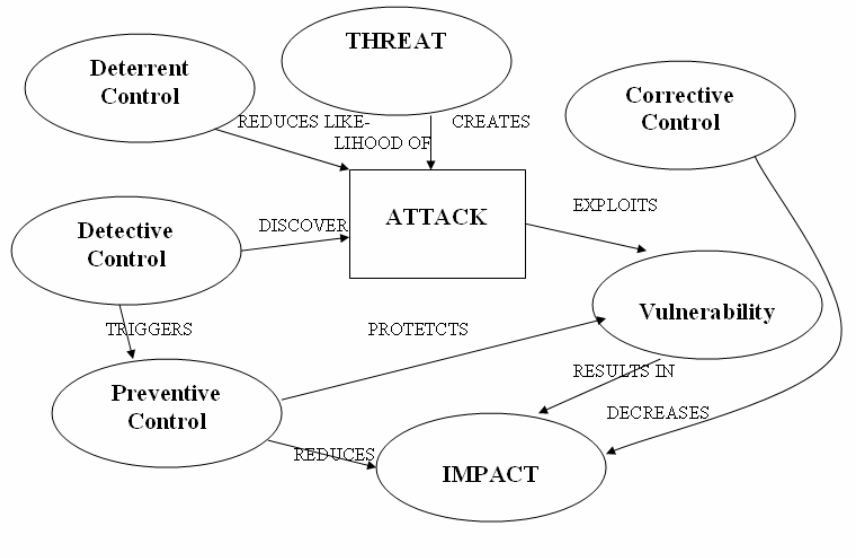


Figure 1.12: Relational model of Qualitative Attack Analysis

Question to be asked are about the following when access and/or being accessed by other nodes in network

- ✓ **Authentication:** Are you who you claim to be?
- ✓ **Authorization:** Are you authorized to do this?
- ✓ **Integrity:** Is the message what the user has sent?
- ✓ **Confidentiality:** Can one read this?
- ✓ **Availability** (Denial-of-service attacks)
- ✓ **Accountability (Non-repudiation):** How can we prove that the message was indeed sent by the sender?

Types And Sources Of Network Threats

Network security is different from system security since with the former, the attacker is mainly interested in sniffing, snooping, sweeping, or just plain looking around in order to gain information about a computing infrastructure. The attacker isn't interested in exploiting any software applications, but instead out to obtain unauthorized access to network devices. Unmonitored network devices are the main source of information leakage in organizations. Just as corporate networks (wires, cables) are the circulatory system of any organization, network devices (routers, hubs, switches) are the caretakers or traffic cops. Indeed, every email message, every web page request, every user logon, and every transmittable file is handled by a network device. Under some setups, telephone service and voice messaging are also handled by network devices. If the attacker is able to "own" one's network devices, then they "own" one's entire network. Network attacks cut across all categories of software and platform type.

There are at least six types of network attacks:

(1) **SNIFFING**: is the interception of data packets traversing a network. A sniffer program works at the Ethernet layer in combination with network interface cards (NIC) to capture all traffic travelling to and from an Internet host site. Further, if any of the Ethernet NIC cards are in promiscuous mode (easily done remotely via a sniffer program), the sniffer program will pick up any and all communication packets floating by anywhere near the Internet host site. A sniffer placed on any backbone device, inter-network link, or network aggregation point will therefore be able to monitor a whole lot of traffic. Also, since most LAN environments are Ethernet-based, messages are not sent securely to their designated parties, but bounce around in what is called the broadcast (network) channel for all addresses on Ethernet-based networks. Most packet sniffer are passive, listening (and possibly recording) all data-link-layer frames passing by the device's network interface. There are dozens of freely available packet sniffer programs on the Internet. The more sophisticated ones (like Back Orifice) allow more active intrusion.

(2) **HIJACKING**: is a technique that takes advantage of a weakness (trust relationships) in the TCP/IP protocol stack, and the way headers are constructed.

Each layer of the model adds what is called a header to the packets that are sent over the network. When creating an e-mail message at the application layer, it passes off to the transport layer, which passes it off to the network layer, and so on. E-mail and most common applications are handled by TCP software. Voice, music, and instant messaging (anything "fire-and-forget") are handled by UDP software. Therefore, all e-mails contain a TCP header implemented at the transport layer. Another header, the IP header is added at the network layer. By the time of click "Send" on an e-mail, the packet contains at least four headers (Ethernet, IP, TCP, application). Hijacking involves the use of tools that subvert the stack's header information. Someone might want to do this in order to spoof a fake message or send a payload inside the header field to the wrong port. There are 12 different IP header fields; one of these fields is ICMP (Internet Control Message Protocol), whose purpose is to negotiate traffic by setting the maximum size of IP packets received. Using commonly found tools like Juggernaut and Hunt, an attacker can adjust the maximum packet size allowed, and send what is called the Ping of Death attack. Such ICMP floods, like Smurf and DoS attacks will quickly consume all resources on the network. ICMP is the most common carriers for bandwidth consumption attacks. It's also used to amplify a DoS (Denial of Service) attack. For example, Tribe Flood Network (TFN) is a hacker program that exploits ICMP traffic, and communicates over ICMP once the compromised system has been turned into a zombie system for launching distributed denial of service attacks on other systems.

TCP is a full-duplex communication channel, which means that information flows between sender and receiver in both directions. UDP is connectionless, so it will gladly accept a packet from anyone despite never having sent an original packet. All TCP headers contain familiar port numbers (even the ones UDP listen on) so packets know which services to obtain on which ports. Savvy attackers know how to subvert the port services that a packet calls upon by modifying the TCP header. This means that an attacker can exploit telnet (port 23), for example, thru a web packet (port 80), or FTP (port 21) thru a telnet (port 23) connection; indeed, just about any port service or application. Attackers commonly scan for open ports (using programs like Nmap)

on DNS (port 53), Web (port 80), FTP (port 21), and mail (port 110) since these are rarely filtered by the firewall or router. TCP wrappers always do three-way handshaking to establish a connection, but half-scans and port scans over TCP are quite common and usually a prelude to a full-blown network attack.

Many attackers prefer the UDP header, which establishes multimedia communication, and among other things, is what Microsoft products use for logons. A little-known protocol called SNMP (Simple Network Management Protocol) is wrapped along with UDP, and is inherently insecure. SNMP is designed to allow viewing of device configurations by community names. All routers, for example, will have two community names: a read only community (with default password public); and a read/write community (with default password private, or secret). Other UDP exploits exist, most of which are aimed at making the attacker an administrator on a network device.

(3) **BACK DOORS**: are accounts left by manufacturers and vendors on devices that allow them to bypass a locked-out or clueless system administrator in case of emergency. Every network device comes shipped with more than one default username and password, and these built-in accounts offer administrative privileges to anyone who finds them. Some examples of generic usernames and passwords are: manager, security, admin, debug, monitor, and guest.

Some Cisco routers are known for using their own name, as in Cisco. In any event, most network devices store their passwords in a configuration file which uses weak encryption, and is easily cracked. The smarter administrator will use MD5 password encryption. Router configuration files are usually located on UDP port 69 and easily downloaded via Trivial File Transfer Protocol (TFTP). Use of TFTP to get configuration files and SNMP to get community names are common attacks on network devices.

(4) **TROJANS**: are programs that look like ordinary software, but actually perform unintended (and sometimes malicious) actions behind the scenes when launched. Most remote control spyware programs are of this type, as are various login programs that look just like a user's regular login screen. Other Trojans create back doors by running a TCP listener and shoveling back a UNIX shell to the

attacker. The numbers of Trojan techniques are only limited by the attacker's imagination.

A "trojanized" file will look, operate, and appear to be the same size as the compromised system file. The only protection is early use of a cryptographic checksum (or binary file digital signature) procedure. On Windows NT servers, a common Trojan is the driver file FPWNCLNT.DLL whose purpose is to grab usernames and passwords while masquerading as a valid system logon component. Scheduled batch job services like weekly virus-scanning or NT/2000's AT Scheduler can also be configured as Trojans.

(5) **SOCIAL ENGINEERING**: is the use of persuasion or deception to gain access to information systems. The medium is usually a telephone or e-mail message. The attacker usually pretends to be a director or manager in the company traveling on business with a deadline to get some important data left on their network drive. They pressure the help desk to give them the toll-free number of the RAS server to dial and sometimes get their password reset. At other times, the tactic is a malicious exploitation or manipulation of some poor clueless user. The human element has been referred to as the weakest link in network security. E.g Shoulders Surfing, Executive Privilege, Faked Email, Malicious Web Page, Chain Mail.

(6) **DENIAL OF SERVICE**: Often, attackers target particular systems, breaking into them so they can be used for specific purposes. Frequently, the host security of those systems will prevent attackers from gaining control over a host. But with *denial of service attacks*, attackers don't need to gain access to a system. The goal is simply to overload a system or network so that it cannot provide its service anymore. Denial of service attacks can have different goals, including *bandwidth consumption* and *resource starvation*.

1.2.1 Process Taxonomy -Network Security

An attack is a single unauthorized access attempt, or unauthorized use attempt, regardless of success..

.. *Security* is the protection of the integrity, availability and, if needed, confidentiality of automated information and the resources used to

enter, store, process, and communicate it. From an operational viewpoint, an attacker on computers or networks attempts to reach or "link" to ultimate objectives or motivations. This link is established through an operational sequence of "means, ways, and ends" that connects attackers to objectives. For this taxonomy, the terms will be "tools, access, and results." These link together attackers and objectives in the process of computer and network attacks as shown in Table 1.2:

Attackers	➡	tools	➡	access	➡	results	➡	Objectives

Table 1.2: Operational Sequence of Computer and Network Attack

- ATTACKERS are the obvious beginning point, the originators, for computer and network attacks. To some extent, they are best differentiated by motivation: The main motivation of a [hacker] is access to a system or data; the main motivation of a criminal is gain; the main motivation of a vandal is damage
- All attackers must either obtain unauthorized ACCESS, or use a system in an unauthorized way, in order to make the connection to their objective. An attacker must take advantage of a computer or network vulnerability, which is a flaw allowing the unauthorized access or use.
- RESULTS of attack, which includes the three traditional categories of corruption, disclosure and denial, but also includes a fourth category: theft of service.
- The final connection to be made in the operational sequence that leads attackers to their objectives is the TOOLS of attack. This is also the most difficult connection to make because of the wide variety of methods available to exploit vulnerabilities in computers and networks.
- People attack computer. They do so through a variety of methods and for a variety of OBJECTIVES.

DIRECTION OF SEQUENCES OF ACTION

Attackers	Tools	Access				Results	Objectives
Hackers	User Command	Implementation Vulnerability	Unauthorized Access		Files	Corruption of Information	Challenge, Status
Spies	Script or Program	Design Vulnerability	Unauthorized Use	Processes	Data in Transit	Disclosure of Information	Political Gain
Terrorists	Autonomous Agent	Configuration Vulnerability				Theft of Service	Financial Gain
Corporate Raiders	Toolkit					Denial-of-service	Damage
Professional Criminals	Distributed Tool						
Vandals	Data Tap						

Table1.3: Complete Computer and Network Attack Taxonomy

1.3 THE CONCEPT OF REMOTE ANALYSIS OF COMPUTER SYSTEM (RACS)

To clarify the purpose of RACS a definition, model and classification of the concept of RACS is needed. In addition to that have summarized what security is to make it possible to evaluate the techniques and software in the context of how to make a system more secure.

1.3.1 Definitions

To define RACS we have to go back to the definition of the words that form the concept.

1.3.1.1 'Remote computer system'

Acting, acted on, or controlled indirectly or from a distance (remote computer operation); also: relating to the acquisition of information

about a distant object (as by radar or photography) without coming into physical contact with it (remote sensing). 'remote'

By this definition of remote a *Remote computer system* can be defined as:

A computer system you can control or acquire information from indirectly (i.e. without direct physical access to).

This definition seems to fit quite well with what defines as a *Remote computer system*.

Any other computer in the network with which the local computer can communicate.

1.3.1.2 'System analysis'

The act, process, or profession of studying an activity (as a procedure, a business, or a physiological function) typically by mathematical means in order to define its goals or purposes and to discover operations and procedures for accomplishing them most efficiently. 'Systems analysis'

In this case the activity is the activity of a remote system.

1.3.1.3 'Remote Analysis of Computer Systems'

By combining *Remote Computer System* with *System Analysis* it is possible to define *Remote Analysis of Computer Systems* (RACS) as:

The act, process or profession of studying the activity of computer system(s), that you can control or acquire information from indirectly, in order to define its goals or purposes and to discover more efficient operations and procedures.

1.3.1.4 'Network Analysis'

Related to *System Analysis* is *Network Analysis* that can be defined with ideas as:

The act, process, or profession of studying a (computer) network typically by mathematical means in order to define its goals or

purposes to determine procedures and a structure for the network to perform most efficiently.]

1.3.2 Model for Remote Analysis of Computer System (RACS)

In addition an abstract model of the process of RACS is defined and used. The model is inspired by on how an artificial agent interacts with the environment.

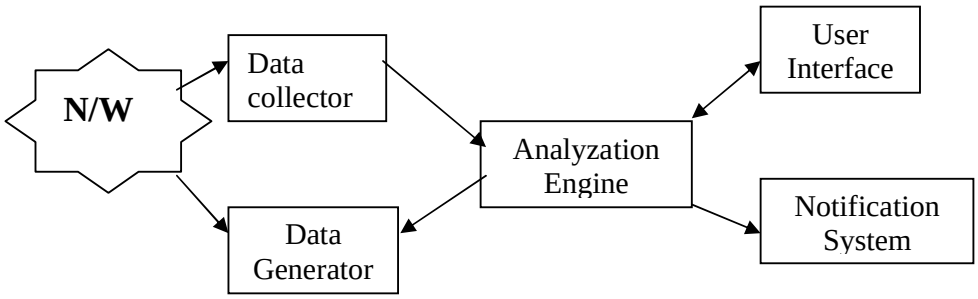


Figure 1.13 : Model of RACS

The two core parts of this model are: the *Data Collector* and the *Analyzation Engine*. The rest of the entities (boxes) are optional components that can make the process easier or more useful. The entities can be of any number and type. The arrows between the entities should be seen as a suggestion of the major information flow.

This model can be used to get an overview of the Analyzation process and should not be seen as a strict model on how things must work in order to be a RACS tool. In practice several parts of the model can be implemented as one single component, probably for performance reasons, but the parts can also be distributed to different software components on several network nodes.

1.3.3 Classification of RACS

The techniques that make RACS possible can be classified based on the model in Section 1.3.2. It is divided into *primary active* and *primary passive* techniques based on if there is an active *data generator* or not. These two major groups have then been divided into subgroups based on the existence of a notification system, the purpose of each technique and the degree of automation each technique has.

a) Active network analysis techniques:

- **Connectivity testing**

The most famous tool for connectivity testing is ping, which uses the Internet Control Message Protocol (ICMP). But there are other alternatives available that use other protocols. The basic principle is to send a message that triggers some mechanism on the remote side, which in turn sends a message back. By using techniques for connectivity testing it is possible to check that hosts are *available*. By locating the problem it is easier to fix it. Availability is one of the core components in the definition of security

- **Network path tracing (continuation of connectivity testing)**

Detection of the path that the network traffic takes from one point to another is in addition to simple connectivity testing an important part of the basic active Analyzation techniques. Traceroute is what the name implies a tool used to discover the links along a path from one IP to another. Path discovery is an essential step in diagnosing routing problems. By tracing the path of a network packet it is possible to know which networks that forwards the traffic. This kind of information is useful in order to know who can break the integrity and confidentiality of transmitted data

Advanced active techniques of Network Analysis

- **Network mapping**

Network mapping is the process of active detection of network devices and gaining enough information to see what services they provide. The area of network mapping include Service Location Protocol that is an IETF standard for locating services(to discover the existence, location and configuration of networked services in enterprise networks) and continues with port scanning which is an other technique to locate services. In some sense this includes operating system detection

- **Operating system detection (OS Fingerprinting)**

Operating system (OS) detection can bee seen as a special form of network mapping which means that it has the same basic properties in terms of the RACS model.

- **Vulnerability Assessment**

Tools to do *vulnerability assessment* are often referred to as security scanners or vulnerability scanners. That is to ASSESS.

ASSESS mean to judge something with respect to its worth or significance. ASSESS implies a critical appraisal for the purpose of understanding or interpreting, or as a guide in taking action <officials are trying to assess the damage.

This definition of the word *assesses* gives a better understanding of what vulnerability assessment is.

b) Passive network analysis techniques:

Packet analysis is a low-level data collection method (normally on the transport layer). This means that the Analyzation engine need knowledge about the higher network layers in order to produce a useful result. The data generator and notification system (see Section 1.2.2) is normally not used for packet analysis. Tools that listen on network packets destined for other hosts than the local host are commonly referred to as *sniffers* or *wire-tappers*. This technique is normally required for useful packet analysis and has the following properties:

- ✓ Passive flow analysis is a stealth method that does not interfere (in theory) or change the behavior of the packet flow. This means that no data generator is allowed to send data to the analyzed network.
- ✓ Requires good knowledge of the *above* layers to get anything useful about the result.

- ✓ Can be constructed in a way that they do not need to be configured at all if proper default values are used (like listening on default interface).
- ✓ Requires super user privileges on the machine that does the analysis because of the low-level nature of the method. The reason is that it has to set the network interface in promiscuous mode (listen on all traffic, not just the traffic for this host) and this requires super user privileges.

Access to traffic

To analyze network packets one have to have access to the network traffic. All packet analysis that require packets that are destined for other hosts (so called promiscuous mode) than the local one will require root-privileges on the host that collect the data. The reason is to protect the network from users that have an account on a computer connected to that network. If such a user is allowed to listen to any traffic and to send arbitrary network packets the *confidentiality* or *integrity* of the network traffic can be broken. To be able to perform this kind of analysis one will need physical access to or root-privileges on a machine that is connected to the network. This means that if a malicious user gain root privileges on one host that can be used to compromise other hosts. Switched networks protect from this in many cases, but a switch can be tricked so it is not foolproof.

Non-automated packet analysis tools can be useful for real-time monitoring or for in depth network debugging. The tcpdump utility can be used for basic manual analyzation or by other more automated tools via the libpcap library. EtherApe can be used to get an overview of which hosts that are using the network at the moment and how the traffic flows. Iptraf can be used in similar way but is more focused on current network statistics like throughput on a specific host or network device. Iptraf can also be used for detecting non-IP traffic.

If one has system administration rights on the switch one can make a *monitoring port* so as an authorized system administrator one should not need these techniques. one should be aware of these problems however because they can be used by other people. There can in some cases exist legitimate ways of using these techniques like network

debugging or product testing. Packet analysis can be automated, this means that the Analyzation engine in the RACS model have to be much more advanced.

CHAPTER: 2
LITTERATURE SURVEY

The basics of Operating system FINGERPRINT can't be understood unless and until the data exploration in the traffic on network is not understood and its techniques defined. Hence to begin with analysis of different field of the headers in packet on network is studied and O/S fingerprinting is targeted.

"… virtually every network attack is preceded by network reconnaissance. Hackers scan and probe networks before they attack in order to get information about the target" - Lenny Liebmann

2.1 EXPLORATION OF AVAILABLE DATA

Passive sniffing tools and active analysis techniques make available and use of all the header information contained in packets as they traverse. The most common protocols: ETHERNET TCP, IP, UDP, ICMP (with respect to OSI model discussed earlier) is believed to provide representative insights that could generally be applied to other protocols. The packet formats of these protocols are well documented. By carefully considering the relevancy of the direct data available and the applications to be analyzed it is possible to construct candidate visualizations.

As a simple example, if one is attempting to fingerprint a simple port scanning program it is useful to visualize the source IP, destination IP, source TCP port, source UDP port, destination TCP port and destination UDP port. In addition, the ability to analyze attack tools using direct data from a sniffing program is enhanced by the use of feature construction.

2.1.1 Link Layer (Ethernet)

Link layer headers are typically created by the node one link distant from the receiving node. For this information to be compromised a nearby node must also have been compromised. Link layer information is particularly useful for detecting anomalous behaviour initiated on a local network segment: for example, to detect 802.11b wireless network abuse, address resolution (ARP) spoofing and attempts to sniff across collision domains in a switched network (e.g. switch flooding, ARP redirects and MAC address spoofing). For purposes of O/S FINGERPRINTING to consider source MAC address, destination MAC address and the overall length (in bytes) of the Ethernet frame.

2.1.2 Network Layer (IP)

Network layer packets are used for host-to-host communication across the Internet and have been subject to much abuse by malicious entities. While chosen to focus visualizations on the source and destination IP address fields there are many areas for future work. Of particular interest are the time to live (TTL) field and the fragmentation offset which can be used for such activities as detecting Honeynet and insertion and evasion attacks to bypass intrusion detection systems.

2.1.3 Transport Layer (TCP and UDP)

Transport layer protocols provide process-to-process connectivity across the Internet. Both TCP and UDP use the notion of ports to support this connectivity. Due to the fact that ports are fundamental to Internet connectivity and that many attack tools probe these ports in an attempt to discover vulnerabilities we chose to include the source and destination ports for TCP and UDP for our visualizations. For future work we leave the visual examination of TCP sequence numbers and flags.

2.1.4 Application Layer

Application layer headers and data provide a great deal of information about the

nature of attacks, but due to the wide variety of application layer protocols we chose to limit our visualization research to raw hex and printable ASCII decodes of this data. There is a great deal of research potential in the visual examination of application layer data. As an example, many zero day network-based buffer overflow attacks will likely have distinct visual signatures.

2.2 REMOTE OS FINGERPRINTING

Remote OS Fingerprinting is a part of the reconnaissance phase of any targeted network attack. Reconnaissance is a practice used by skilled hackers to size up and gather information about their target. There are several ways to go about gathering any given piece of information regarding a target that would yield vulnerability. One of the most important pieces of information that a hacker could have is the flavor and version operating system of a remote host. With information in regards to the flavor and version of the operating system, a hacker could look for any number of possible vulnerabilities via information on the web that are specific to that operating system and version.

The term OS fingerprinting defines any method used to determine what operating system is running on a remote computer. OS fingerprinting is a key element in network reconnaissance as most exploitable vulnerabilities are operating system specific.

An attempt to exploit Microsoft IIS vulnerability on a Linux 2.4 machine is destined to fail. If an attacker is able to determine what remote operating system a target is running then he or she will likely be able to cross off a large number of exploits from their known exploit list and instead concentrate on exploits that may work. This will both decrease the likelihood of an attack being detected and greatly increase the chances of an attack being Successful

2.2.1 Usage of O/S FINGERPRINTING

Information gathering is an essential part of acute vulnerability assessment, especially when the whole process is automated. In this context, host operating system detection

must be precise, in particular when networks are well defended. In recent years, the need for automated Internet vulnerability assessment software has been identified and has resulted in the very fast growth of widely available solutions.
As an essential part of the assessment process, remote Operating Systems detection, a.k.a. Operating System Fingerprinting, must meet several requirements:

- **Accuracy**: no falsely detected Operating System.
- Firewall and IDS **neutrality**: not be disturbed by / do not disturb existing firewalls and IDS.
- **Politeness**: low network traffic and no dangerous segments.
- **Handiness**: easily extensible signature database and automation functions.
- **Speed**: depending on the usage, a fast fingerprinting tool might allow large network scans**.**

BLACKHAT (An attacker) who is able to successfully conduct recon on your network has a much higher likelihood of attempting to compromise your network than an attacker whose recon attempts are thwarted. Furthermore, the attacker with recon information has a higher probability of successfully compromising your network.

A WHITEHAT (network admin) who understands network scanning, and can implement both defence and detection from this knowledge, will be able to significantly increase the security level of his or her network while, at the same time, significantly reducing the response time if a network incursion occurs only if he understand the techniques and methods used for o/s fingerprinting.

Identifying a computer's operating system is often the first step of malicious users seeking to compromise a network. The type and version of an Operating System might be basic information, but it is hardly trivial to network security.

Reasons to hide Operating System to the entire world

Maybe the following reasons can convince:

- Revealing the Operating System makes things easier to find and successfully run an exploit against any of the devices.

- Having and unpatched or antique Operating System version is not very convenient for the company prestige. Imagine that the company is a bank and some users notice that it is running an unpatched box. They won't trust it any longer! In addition, these kind of 'bad' news are always sent to the public opinion.
- Knowing the Operating System can also become more dangerous, because people can guess which applications are running in that OPERATING SYSTEMS (data inference). For example if the system is a MS Windows, and is running a database, it's highly likely that its running MS-SQL.
- It could be convenient for other software companies, to offer a new OPERATING SYSTEMS environment (because they know which are the systems running).
- And finally, privacy; nobody needs to know the systems one's got running.

Operating System fingerprinting (Operating System identification) is more of an art than a science. Administrators and security professionals must understand both sides of the process – how information can be obtained and the countermeasures for such exploits –to properly secure their networks.

2.2.2 Operating System Identification Styles: Active vs. Passive

There are two styles of Operating System identification –active and passive. **Active Operating System fingerprinting** has been around the longest and is far more widely used. Active fingerprinting makes itself known to the target, sending unconcealed packets that help determine the target Operating System. The packets can take many forms – everything from simple banner grabbing to malformed packets that identify differences in TCP/IP stack implementation.

Passive Operating System fingerprinting is relatively new, and only a few tools employ this style. It is stealthy; it does not make its presence known to the target. By sniffing packets that a target broadcasts and analyzing how a target responds to packets sent to it during normal operation, passive fingerprinting is difficult to detect while it is taking place.

There important differences between these styles. Unlike passive fingerprinting, active fingerprinting does not require extensive access to a target. However, passive fingerprinting can analyze larger amounts of data over a longer time. Collecting more data about a target helps make fingerprinting more accurate. Passive fingerprinting is transparent to network address translation, firewalls, and packet filtering. The yields a high degree of accuracy for both Operating System fingerprinting and network mapping.

The following charge summarizes the pros and cons of active and passive OS identification.

Active vs. Passive OPERATING SYSTEMS (OS) Identification			
Active OS Identification		**Passive OS Identification**	
Pros	**Cons**	**Pros**	**Cons**
Quick to complete	Not stealthy	Extremely stealthy	Potentially time-consuming
Does not require extensive access to target	Requires minimum number of open ports	Highly accurate	Requires extensive access to target
	Susceptibility to "smoke screen" responses generated by target	Immunity to "smoke screen" responses generated by target	Requires one or more SYN packets
	Inaccurate when IP Personality modifications are made to target		Inaccurate when IP Personality modifications are made to target

Table2.1: Comparison of Active and Passive Fingerprinting

2.3 OS FINGERPRINTING TECHNIQUES & METHOD

OS FINGERPRINTING is not an exact science. Even the most sophisticated detection tools ultimately make educated guesses about a target OS, and no single

method is sufficient for foolproof fingerprinting. Administrators who gather information from all possible sources are in the best position to draw an accurate conclusion.

Security Assessors already have a choice of Fingerprinting techniques and tools, each of which may be suitable for some application context. Examples are:

- **Banner grabbing** allows OS deduction from services banner and is appreciated by most human assessors. This can be completed by binary file collect and analysis
- **TCP segments** (standard or not) response analysis relies on different Operating System responses to specifically prepared segments, particularly when response behaviour is not clearly specified in RFCs. Furthermore, vendors have introduced fine tuning and proprietary extensions into their TCP/IP stack, which will clearly identify those systems in case of such solicitations
- **ICMP response analysis** is recent. By sending UDP or ICMP solicitation and analyzing various ICMP replies.
- **Initial Sequence Number (ISN) analysis** exploits differences in TCP stacks random generators, identified through a sufficient number of tests
- **Operating System Specific** deny of service are recalled here for the sake of exhaustively, but are only used by hackers. Except for very precise situation, the overall accuracy of this method is rather bad.

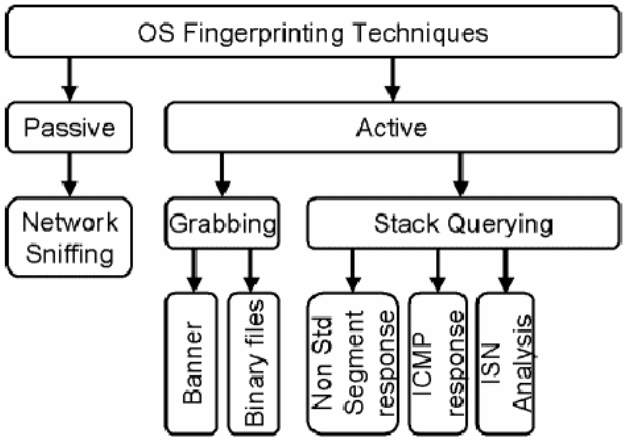


Figure 2.1: Synoptic of OS fingerprinting Technologies

Discussed below are the classic and modern techniques of OS FINGERPRINTING

2.3.0 “Classic” Techniques

2.3.0.1 PORT SCANNING

Port scanning is used to determine what ports are open on a system, and thus what services are listening (and available to be attacked. Systems with port 445 open are most likely Windows 2000 systems, while systems with ports 6000-6063 open are probably running X-Windows, and are most likely some flavour of UNIX.

2.3.0.2 BANNER GRABBING

Banner grabbing is the simplest and easiest technique. Any telnet client can be used to connect to an open port and see what logon information is advertised. Many telnet, Web, SMTP, and FTP servers proudly display their product name, version, and OS information. The following is an example “banner grabbing” session.

TELNET TO PORT 25 TO SEE WHAT SMTP SERVER THE TARGET IS USING:
>telnet mail.tiet.ac.in25
220-server.mail.tiet.ac.inESMTP Exim 3.33 #1 Sun, 09 Dec 2001 20:28:18 -0800
220-NO UCE. MajorISP does not authorize the use of its computers or network
220 equipment to deliver, accept, transmit, or distribute unsolicited e-mail.
>quit
221 server.mail.tiet.ac.inclosing connection

A quick web search reveals that “Exim is a message transfer agent (**MTA**) developed at the University of Cambridge for use on UNIX systems connected to the Internet”. So few things are already known, the MajorISP SMTP server is UNIX, the message transfer agent is Exim, and the version is 3.33. A quick perusal of the Exim website reveals complete documentation, including apparently much of the author’s design

philosophy, etc. A dedicated attacker would certainly find a lot of interesting information here. The hostname of the particular server in use can also be noted.

NEXT, EXAMINE PORT 110, POP3:

>telnet mail.tiet.ac.in 110

+OK POP server vMI_3_36 at mail.tiet.ac.inready <22321.1007964598@hostname>

Not a lot of information is revealed in this case, as a web search on the POP server identified does not reveal anything useful. The implementation appears possibly to be proprietary. However, again The hostname of the particular server in use can also be noted. (server's name is different from the SMTP server in this case).

TELNET TO PORT 80 DOES REVEAL SOME INTERESTING INFORMATION:

>telnet mail.tiet.ac.in 80><cr>

HTTP/1.1 400 Bad Request

Server: Netscape-Enterprise/3.6

Now MajorISP employs Netscape-Enterprise/3.6 on their web servers, and can begin searching for potential vulnerabilities. By examining the "Supported Platforms and System Requirements" of the Netscape Enterprise Server , leads to narrow downing of the OS in use, but still don't know which flavor of UNIX they are running.

Next, try FTP:

>telnet ftp.mail.tiet.ac.in 21

220-Welcome to the anonymous FTP server at MajorISP, Inc.

220-

220-If your FTP client crashes or hangs shortly after login, try using a

220-dash (-) as the first character of your password. This will turn off

220-the informational messages which may be confusing your ftp client.

220-

220-If you have any questions, please send mail to ftp@mail.tiet.ac.in

220-

220 bob FTP server (Version MISP-FTPD(2) Thu Feb 6 17:05:52 PST 1997) ready.

>SYST

215 UNIX Type: L8

Unfortunately, “UNIX Type: L8” is useless in determining the OS version . “bob FTP server (Version MISP-FTPD(2) …)” doesn’t reveal much about the FTP daemon in use, except that it appears to be quite old (1997), and it appears to be something home grown (or at least modified by the target organization). A web search reveals nothing useful.

So, in this example, target has done a decent job of hiding the OS versions of their servers, at least against simple banner grabbing. However, it was discovered that the target organization runs some variant of UNIX, they use Exim 3.33 for SMTP, and they serve web pages with Netscape’s Enterprise server 3.6. In many cases, these techniques will be completely successful, and are the first and easiest method of remote OS identification.

Several other ports might give up useful information, such as 22 (SSH), 23 (telnet), 143 (IMAP), 113 (Identd) etc., depending on what ports were found to be open in earlier port scans.

2.3.0.3 EMAIL HEADERS

Email headers can reveal significant details about a target system. If simply connecting to port 25 had not revealed any useful information, it can be gathered from the email headers of a message delivered through the target mail system. For example, email sent to an invalid user reveals several server names, as well as the SMTP server software and version (again, Exim 3.3):

Status: U
Return-Path: <>
Received: from server1.mail.tiet.ac.in([w.x.y.z])
by server2 (MajorISP SMTP Server) with ESMTP id u18fup.b73.37tiu4s
for <user@mail.tiet.ac.in>; Sun, 9 Dec 2001 20:53:45 -0800 (PST)
Received: from exim by server1.mail.tiet.ac.inwith local (Exim 3.33 #1)
id 16DIRk-00038m-00
for user@mail.tiet.ac.in; Sun, 09 Dec 2001 20:53:44 -0800

X-Failed-Recipients: bogustestemail@mail.tiet.ac.in

From: Mail Delivery System <Mailer-Daemon@server1.majorisp.net>

To: user@mail.tiet.ac.in

Subject: Mail delivery failed: returning message to sender

Message-Id: <E16DIRk-00038m-00@server1.majorisp.net>

Date: Sun, 09 Dec 2001 20:53:44 -0800

Email sent to an auto-reply address at the target (info@mail.tiet.ac.in) reveals additional useful information:

Status: U

Return-Path: <sales@mail.tiet.ac.in>

Received: from server3.earthlink.net ([w.x.y.c])by server4 (MajorISP SMTP Server) with ESMTP id u18os0.ikc.37tiu8v

for <user@mail.tiet.ac.in>; Sun, 9 Dec 2001 23:25:52 -0800 (PST)

Received: (from autoreply@localhost)

by server3.mail.tiet.ac.in(8.9.3/8.9.3) id XAA12332

for From: "User Name" <user@mail.tiet.ac.in>; Sun, 9 Dec 2001 23:25:52 -0800 (PST)

Date: Sun, 9 Dec 2001 23:25:52 -0800 (PST)

Message-Id: <200112100725.XAA12332@server3.majorisp.net>

To: user@mail.tiet.ac.in

Subject: Re: test request for information

References: <001101c1814c$32810b60$fb01a8c0@mail.tiet.ac.in>

In-Reply-To: <001101c1814c$32810b60$fb01a8c0@mail.tiet.ac.in>

Precedence: auto_reply

X-Loop: autoreply@mail.tiet.ac.in

From: Sales at MajorISP Inc sales@mail.tiet.ac.in

Notice that server3 reveals the text “8.9.3/8.9.3”. What would you bet this server is running sendmail 8.9.3? A quick telnet to port 25 confirms this:

>telnet server3.mail.tiet.ac.in25

220 server3.mail.tiet.ac.inESMTP Sendmail 8.9.3/8.9.3; Sun, 9 Dec 2001 23:39:48 -0800 (PST)

Again, this does not reveal the underlying OS, but it does present another SMTP application to check for vulnerabilities. As Sendmail 8.9.3 is not the current version, it seems likely to be an unpatched and possibly vulnerable system (assuming the sendmail. conf file has not been altered to present false version information). A quick search of the NIST vulnerabilities database reveals several potential weaknesses.

It would be interesting at this point to probe this target's email anti-virus defences by sending the EICAR test string to both valid and invalid email addresses. The EICAR test string is an innocuous string of characters that anti-virus software detects as a virus, and is used by administrators to confirm that anti-virus implementations are functioning properly. An attacker could include the string in the body of an email, or as an "innocuous" attachment (.TXT for example), inside of an attached archive file, inside of an attached password-protected archive file, or as an email attachment with various file extensions (.EXE, .COM, .SCR, etc.). Responses could reveal the anti-virus configuration, and possibly the software and version.

2.3.0.4 DNS QUERY

NSLOOKUP and Dig are two tools used to query DNS servers for various information, from full zone transfers to HINFO and TXT records, to the version of the DNS server itself. TXT records contain "descriptive text", exactly the sort of thing attackers might find useful. For example, an administrator might note the function of a particular server, or some other text that would provide clues to the attacker. HINFO records typically contain the Host's CPU type and OS [6]. Luckily, most DNS administrators are wise to this by now, and so don't use HINFO or TXT records in publicly available zone files. From the attacker's perspective it would be worthwhile checking for these, just in case.

The following is an example of an attempted zone transfer, and an attempted BIND version query:

>nslookup
Default Server: ns.majorisp.net
Address: w.x.y.z

>ls –d majorisp.net

[ns.majorisp.net]

*** Can’t list domain majorisp.net: Query refused

> set class=chaos

> set type=txt

> version.bind

Server: ns.majorisp.net

Address: w.x.y.z.

*** ns.mail.tiet.ac.incan't find version.bind: Query refused

The example target has implemented defenses against this information gathering technique, most likely editing the named.conf to allow zone transfers and BIND version queries only from particular hosts, as described in . As an alternative defense against the version query, they could have altered the version information returned as described in .An interesting area of future research would be to create a “DNS version interrogation cookbook”, as envisioned by “Mr. DNS” [9], that could “fingerprint” a DNS server, similar to the way that NMAP (discussed later) fingerprints the TCP/IP stack.

2.3.0.5 SNMP

An initial port scan would have turned up SNMP services running on a target host at port UDP 161. It’s undoubtedly wishful thinking that nobody in the world could possibly be running SNMPv1 on an Internet connected host, and certainly an attacker would find it worth investigating. SNMPv1 relies on “Community Strings” for authentication, and passes all data (including the community strings) in clear text. Default community strings are well known (e.g. “public” and “private”). If Read-only access can be achieved, it is trivial to access OS version information. Tools such as SNMPUTIL from the Windows NT Resource Kit (http://www.microsoft.com), and SNMPWALK (http://www.mkssoftware.com) can be used to gather the information. If read-write access can be achieved, then it’s no more secure.

2.3.0.6 SOCIAL ENGINEERING

It should be possible to call the target, pose as a vendor, and ask questions about the number of servers, the OS in use, and so on. Nobody at the target is likely to recognize you as an attacker, as this is a standard line of interrogation used by any pushy sales person. The target may be left feeling as though they need to go wash their hands after talking with you, but they are unlikely to realize that you are gathering information that will be used against them in a network attack.

2.3.1"Modern" Techniques

2.3.1.1 Telnet Session Negotiation

Telnet session negotiation (TSN) is the simplest way to determine a remote OS. All it requires is that you telnet to the server. It is surprising how many systems have telnet running for no reason. Worse, many networks respond with a banner that gives the exact OS version! Although this method is not elegant, it is nevertheless effective. TSN should be the first thing you check in fingerprinting.

It is worth noting that this weakness is rampant among software makers and is not limited to operating systems. For example, NTMail, a popular POP3 mail server from Gordano, returns the exact version of the software to anyone passing by on the Internet. Simply telnet to the default POP3 port (port 110) on a server running NTMail, and you learn the exact version. This access was provided so that Gordano could troubleshoot and also track piracy of their software. However, with the information it provides, a cracker can do a quick search for exploits for that version (such as the denial-of-service vulnerability affecting early versions of NTMail) and attack with ease. TSN is a classic method, but it is becoming less effective as administrators are learning to turn off their banners.

2.3.1.2 Fingerprinting Through Service (Daemon) Querying

Services that appear on a portscan list can be further queried to produce identifying information. These techniques are more recent in concept and the tools still in the

early stages of development. One example involves querying the LPD daemon. The technique employed by lpdfp15, a “proof of concept” tool, involves sending malformed Control File commands to the daemon and gauging the response. Various printer daemons respond to error conditions differently allowing a fingerprint to be built up for comparison with a database of expected results. Ftpmap16 is another tool still in the early stages of development, which attempts to fingerprint via an ftp daemon. Ftpmap then inspired SMTPscan17, a similar tool attempting to exploit differences in mail daemons. In this case 15 tests are employed to provoke error responses, and the author claims to have correctly identified 77 daemons to date.

More tools exist which build on this technology aiming to generalize the approach to service identification. One notable tool is Amap, which proudly proclaims to send a trigger packet to an unknown port and again compares the response to a database of fingerprint packets. It makes little assumptions as to convention and, as such, claims to be able to detect daemons listening on non-standard ports. This tool is being actively worked on to develop the database and even supplies an additional program called ‘amapcrap’ which the Readme states “sends random crap to a udp, tcp or ssl'ed port, to illicit a response”, which you can then send in to add to the database.

It is worth noting that this approach relies heavily on a direct match of the default application daemon to the underlying OS as you are inferring the nature of the OS rather than specifically testing it. Further, as can be seen by the SMTPmap example this technology is not as refined or perhaps reliable, with some 15 tests being required to resolve 77 daemons, compared to Nmap’s 7 tests, for 500+ OS. This may be due to the fact that daemons operate at the TCP/IP model application layer and hence part or all of the code can easily be ported between OS variants, particularly on Unix where source code is commonly available.

2.3.1.3 Fingerprinting Through TCP Retransmission Timeouts

The latest technique for OS fingerprinting uses the novel approach of TCP Retransmission timing. TCP relies on IP for packet transfer over the Internet. TCP is designed to be connection oriented and reliable. To enforce these design rules a connection must be requested and acknowledged before transfer can take place, and

every packet must be acknowledged as having been successfully received. If a packet is not received in a predetermined period of time, it must be resent. The original RFC 793 defining the TCP protocol does not explicitly impose an algorithm for the timing of resending 'lost' packets, instead only suggesting one. RFC 2988 notes the sensitivity of TCP to retransmission problems and specifies an algorithm for use when retransmission should occur. Unfortunately not all IP stacks implement the latter RFC or interpret the first in different ways.

This then allows for a novel approach to be taken in which a new connection is requested with a SYN packet, but ignores all SYN-ACK response packets, instead measuring the timing between successive attempts to acknowledge the connection and comparing these times to a fingerprint database. Requesting the TCP timestamp option (when possible) also serves to mitigate the fact that the Internet itself is prone to timing problems. Interestingly, the timestamp option is dependent on IP stack implementation, which further serves to differentiate OS.

An obvious advantage of this method is that only one valid TCP SYN packet is used to initiate the process. Thus the procedure is 'firewall friendly' allowing fingerprinting of hosts behind a firewall with at least one exposed TCP port

.

The initial proof of concept tool, RING (Remote Identification Next Generation) demonstrates this methodology, as implemented in C. RING is also implemented as an add-on to Nmap, making a very powerful combination. The technique was ported to Perl shortly thereafter and named Snacktime, independently confirming the concept and increasing portability.

2.3.1.4 Fingerprinting Through TCP/IP stacks

TCP stack fingerprinting involves hurling a variety of packet probes at a target and predicting the remote OS by comparing changes in responses against a database.

a) Active TCP/IP Fingerprinting

NMAP is considered to be the best tool available for port-scanning and active TCP/IP fingerprinting, and performs a variety of active probes to detect the remote OS. These

are: FIN flag probe, bogus flag probe, TCP initial sequence number sampling, don't fragment bit, TCP initial window size, ACK value, ICMP error message quenching, ICMP message quoting, ICMP error message echoing integrity, type of service, fragmentation handling, and TCP options.

Other methods of TCP/IP stack fingerprinting exist, such as examination of the TTL value, Maximum Segment Size, and source port.

b) Passive TCP/IP Fingerprinting

One problem with active TCP/IP fingerprinting is that it requires the fingerprinter to send a number of strangely formed packets to the target system. This pattern (and even some of these individual packets) is easily detectable and may alert the administrators of the target system (if they are examining their firewall and/or NIDS logs). Another problem is that an active scan can in some rare cases cause the target system to crash. The goal of passive TCP/IP fingerprinting is to determine the OS without sending any packets at all to the target, or at least nothing out of the ordinary. It can also be used by defenders to help ID attackers. This can be achieved by examining many of the same parameters that are examined in an active fingerprint session: Initial Sequence numbers, TTL, window sizes, don't fragment bit, type of service, etc. can be used to identify the OS.

The following is a syn-ack packet from our example web server, captured with Microsoft Network Monitor:

```
00000:  ww ww ww ww ww ww xx xx xx xx xx xx 08 00 45 00
00010:  00 30 F4 1D 40 00 2F 06 53 D2 yy yy yy yy zz zz
00020:  zz zz 00 50 09 5F A9 6F 12 08 D6 A5 B1 66 70 12
00030:  60 F4 D1 0F 00 00 01 01 04 02 02 04 05 B4
```

From this we see that the TTL is 47 (original TTL is most likely 64), the Don't Fragment bit is set, the Type of Service is 0x0, and the Window size is 24820. Consulting the database of signatures provided by [15], we determine (finally!) that the target system from my previous examples is most likely Solaris, version 8

2.3.1.5 The Final Straw – Chronological Exploits

If all else fails, less tactful attempts at OS identification can be made by launching known exploits for a given OS type against a target host, in chronological order. The theory is that exploits are patched as they are discovered so by starting with the oldest known exploit against a given host and working forward should yield a point at which an attack succeeds, which should thereby identify the revision of OS in use. As an example, Microsoft Windows 95, 98 and NT4 are difficult to distinguish supposedly because the IP stack code was only marginally revised between OS versions. Starting with a basic WinNuke attack and moving forward to more complex attacks such as Teardrop can eventually yield a vulnerability that points to the type and/or hotfix revision that is missing from the OS, thus indicating the current patch level.

OS Fingerprinting Techniques Comparison	Banner Grabbing	Non standard segments	ICMP replies	ISN Sampling	Temporal analysis
History					
Classical implementation	Plenty	NMAP	X-PROBE	?	RING
Created by	Hackers	Fyodor	Ofir Arkin F. Yarochkin	M. Zalewsky Guardent	Intranode
First released	?	Jun, 1998	Aug, 2001	Apr, 2001	Mar, 2002
IP Protocol & Service					
Used protocols	Service related	IP, TCP, UDP & ICMP	UDP & ICMP	TCP	TCP
Open TCP port required	Service related	The more the better	No	Yes	Yes
Closed TCP port required	No		No	No	No
Closed UDP port required	No		Yes	No	No
Firewall concerns					
Bypass filtering routers	Always	Generally	Rarely	Generally	Generally
Bypass SYN relays	Always	Rarely	Always	Never	Never
Bypass application proxies	Possible	Never	Never	Never	Never
Outgoing firewall neutral	Always	Generally	Generally	Always	Always
IDS concerns					
Detection	Hard	Easy	Possible	Possible	Hard
Blocking	Hard	Possible	Possible	Possible	Hard
Misc.					
Learning functions	No	Yes	No	?	Yes
KB size on Marsh 2002	?	> 600	~ 20	?	~ 30
Target hosts disturbance	None	Rare	None	Possible	None
Best match feature	No	Yes	?	?	Yes
Defensive measures	Banner rewriting	Firewall or host stack tuning	ICMP blocking at firewall	SYN Relaying	SYN Relaying or host stack tuning

Table. 2.2: Possible synergies mixing most efficient techniques and tools

CHAPTER: 3
PASSIVE FINGERPRINTING

3. PASSIVE FINGERPRINTING

Passive fingerprinting is a method for learning more about the attacker with out risking detection. You can potentially determine the operating system, services, and applications of a remote host by using nothing more then sniffer traces. Traditionally, using active tools. These tools operate on the principle that every operating system's IP stack and applications have unique properties and idiosyncrasies. One can send a sequence of probe packets to target systems and examine the responses very carefully. Many attributes, such as default TCP window size, supported TCP options, and ICMP error message characteristics 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.

Passive fingerprinting follows the same concept but is implemented differently. Passive fingerprinting is based on sniffer traces of traffic generated by the remote system. Instead of actively querying the remote system, it simply capture packets sent from the remote system. Remember, the Honeynet captures every packet sent by the remote system. Because this is being done passively, without black hat's knowledge, passive fingerprinting does not increase the risk of the black-hat's discovery of being connected to a honeypot. Its goal will be to learn the most information without the attacker's being aware of our data collection. Hence, attempt to identify the operating system, services, and, sometimes, the application used by the enemy. The more information obtained the better.

Passive OS mapping has become a new area of research in both white hat and black hat arenas. **For the white hat**, it becomes a new method to map their network and monitor traffic for security. For example, a new and possibly subversive host can be identified quickly, often with great accuracy. **For the black hat**, this method provides a nearly undetectable method to map a network, finding vulnerable hosts. To be sure,

passive mapping can be a time consuming process. Even with automated tools a sufficient quantity packets to arrive to build up a statistically significant reading of the subjects' operating systems.

Passive fingerprinting has some advantages over active fingerprinting.

- ✓ We are able to act on all TCP/IP layers.
- ✓ We can detect systems with low uptime.
- ✓ We can detect patterns of behavior.
- ✓ The action happens passively, with the remote user unaware of what we are learning.

  But passive fingerprinting is not perfect.
- ✓ It is not 100 percent accurate.
- ✓ Some applications build their own packets and will not produce the same signatures as the operating system itself would.
- ✓ Some of the default values we rely on can be changed; information can be spoofed.

3.1 TECHNIQUES AND APPROACHES

3.1.1The TCP techniques

Examine four TCP packet headers to determine the operating system; however other signatures can be used. The following fields in the header:

- **IP time – to- live**:- The number of routing hops allowed for a packet to reach its destination, or time to live. This is the field also used by tracerourte programs.
- **Window size**:- An internal TCP data flow control measure that varies by operating system (OS).
- **DF**:- The IP “Don’t’s Fragment” bit which some operating systems always set.
- **TOS**:- The IP “Type of Service” field, whose setting reveals information about the underlying OS.

By analyzing packet fields, you may be able to determine the remote operating system. This system is not 100 percent accurate and works better for some operating systems than for others. No single signature can reliably determine the remote operating system. However by looking at several signatures and combining the information, you increase the accuracy of identifying the remote host. Dozens of other packet attributes could be used. The easiest way to explain this is through an example. Following is the sniffer trace of a system sending a packet. This system launched a mounted exploit against the Honeynet, so we want to learn more about it. This technique studies the information passively. Using short captured this signature.

Based on four criteria, identify the following:-
TTL: 45
Window size : 0x7D78, or 32120 in decimal
DF: Don't Fragment bit set
TOS: 0x0

Compare this information to a database of signatures. First, look at the IP TTL used by the remote host. The sniffer trace shows that the TTL is set at 45. This most likely means that the original TTL was set to 64 and went through 19 hops to get to us. Based on this TTL, it appears that this packet went through 19 hops to get to us. Based on this TTL, it appears that this packet was sent by a Linux or a Free BSD box; however, more system signatures need to be added to the database. This TTL can be confirmed by doing a traceroute to the remote host. If you are concerned that the remote host will detect your traceoute, you can set its time-to-live (default 30 hops) to be one or two hops less than the remote host: $_{m}$ option for a UNIX system, $_{h}$ for Microsoft systems. In this case, doing a traceroute to the remote host, starting with a TTL of 18 hops (traceroute-$_{m}$ 18). This gives us the path information, including its upstream provider, without actually touching the remote host. Be careful with this method. Routing paths to and from your facilities may vary, making this method unpredictable.

The next step is to compare the TCP window size. The window size to be another effective tool: what window size is used and how often the size changes. In the preceding signature, it seems set at 0x7D78., a default window size commonly used by Linux. Also, Linux, Free BSD, and Solaris tend to maintain the same window size throughout a session, as this one did. However, Cisco routers (at least 2514) and

Microsoft Windows/ NT window size are continually changing. However, this may also be at least partially a characteristic of network latency and processing times rather than an inherent OS characteristic. We have found that window size is more accurate if measured after the initial three-way handshake, owing to TCP slow start.
Most systems set the DF bit, so this is of limited value. However this does make it easier to identify the few systems, such as SCO or Open BSD, that not use the DF flag.

After further testing, we feel that TOS is also of limited value. This seems to be more session based than operating system dependent. In other words, it's not so much the operating system that determines the TOS but the protocol used. TCP and ICMP for example handle the TOS field differently. TOS definitely requires some more testing. So based on the preceding information about TTL and window size, you can compare the results to the database of signatures and with a degree of confidence determine the OS- in our case, Linux based on Kernel 2.2.x.

This technique is not limited to the four TCP field values discussed so far. Other areas too can be tracked, such as initial sequence numbers, IP identification numbers, and TCP or IP options. For example, Cisco routers tend to start IP identification numbers at 0 instead of randomly assigning them. For TCP options, the option selective acknowledgment sackok is commonly used by Windows and Linux but not by Free BSD or Solaris. For maximum segment size (MSS), most operating systems use 1,460; however, Novel commonly uses 1,368 and some Free BSD variants may use 512. However this can depend on the interface type used and also the network infrastructure between the machines if path MTU discovery is used.

3.1.2 ICMP TECHNIQUES

ICMP ECHO request is unique in that almost every operating system has this capability. This makes ICMP- based applications one of the most commonly used by blackhats. Normally the ping utility is used to generate ICMP Echo requests. It can be a clear distinction between the ping implementation with UNIX and UNIX- like operation systems and the ping implementation with Microsoft- based operating

systems. This example will compare two ICMP Echo requests, one from a Microsoft-based operating system and one from a Linux machine.

- ✓ **ICMP Echo request datagram size**: With Microsoft – based operating systems, the ICMP Echo Request generated with ping will be 60 bytes long. With UNIX and UNIX- like operating systems, the ICMP Echo Request generated with the ping utility will be 84 bytes long.
- ✓ **ICMP Echo request data payload content**:- The data in an ICMP Echo Request sent with the ping utility on a Microsoft- based operating system will be composed of the alphabet, whereas UNIX and UNIX like operating systems' ping will use numbers and symbols.
- ✓ **ICMP Echo request timestamp**: With the ping output, we have a time calculation of the round- trip time (RTT) , or how long it took the datagram to travel from the initiating host to the target host and to come back. With ping on UNIX and UNIX- like operating systems, the first 8 bytes of the data payload are a timestamp helping us to calculate the RTT. If you look closely at the Microsoft- based ping data payload, you may discover that there is no such timestamp. The contents starts with the alphabet. So where is the timestamp being saved with MS- based machines? In memory, probably.
- ✓ **ICMP identification number used**:- Microsoft- based operating systems use constant values for this field. The value will not change. The values are 256, 512, and 768. With UNIX and UNIX- like operating systems, the ICMP ID will be the process ID assigned to ping when executed. This means that the value for UNIX will continually change.
- ✓ **ICMP sequence numbers**: Both UNIX and Microsoft- based systems incrementally increase Sequence (Seq) numbers with 256. However, UNIX systems always start the Seq number at 0, whereas Microsoft systems start the Seq number at the last Seq number used in the previous iteration of ping plus 256. For example, in the preceding example, the Microsoft version of ping set the initial Seq number at 5,120, meaning that the previous time ping was used, the last Seq was number 4,864. This will be reset to 0 only when the system reboots.

Some blackhats use different types of ICMP tools to generate ICMP query messages or malformed ICMP queries. This technique can be further using this information to also identify some of those tools. For example, this is how it would detect a ICMP Echo packet generated not by an operating system but by the application Hping2. Hping2 is a network tool able to send custom IP packets and to display target replies like ping does with ICMJP replies. Hoping2 handles fragmentation, arbitrary packet body, and size and can be used to transfer files under supported protocols.

3.1.3. An Alternative Approach

Methods, which rely solely on the IP/ICMP options present in normal traffic, are limited in the accuracy about the targets. One limitation of these methods, though, is that they only provide a measure of the operating system. Vulnerabilities may or may not exist, and further investigations must be undertaken to evaluate if this is the case. While suitable for the white hat for most purposes (like accounting), this is not suitable to a would-be attacker. Simply put, more information is needed.

Further inspection is also needed to determine avenues of vulnerability, as well. A method exist to rapidly identify target operating systems and version, as well as vectors of attack, based on data sent by client applications. While simplistic, it is robust. The accuracy of this method is also quite high in most cases. Four methods of fingerprinting a system are presented, with sample data provided.

3.1.3.1 Fingerprinting using Network Client Applications

An alternative method to merely fingerprinting the operating system is to perform an identification by using client applications. Quite a number of network clients send revealing information about their host systems, either directly or indirectly. We use application level information to map back to the operating system, either directly or indirectly. One very large advantage to the method described here is that in some situations, much more accurate information can be gained about the client. Because of stack similarities, most Windows systems, including 95, 98 and NT 4.0, look too similar to differentiate. The client application, however, is willing to reveal this information. This provides not only a measure of the target's likely operating system, but also a likely vector for entrance. Most of these client applications have numerous

security holes, to which one can point malicious data. In some cases, this can provide the key information needed to begin infiltrating a network, and one can proceed more rapidly. In most cases it provides a starting point for the analysis of vulnerabilities of a network. One major limitation of this method, however, comes when a system is emulating another to provide access to client software. This includes Solaris and SCO's support for Linux binaries. As such, under these circumstances, the data should be taken with some caution and evaluated in the presence of other information.

This limitation, however, is similar to the limitation that IP stack tweaking can place on passive fingerprinting at the IP level or the effect on active scanning from these adjustments or firewalling. Four different type of network clients are discussed here which provide suitable fingerprinting information. Email clients, which leave telltale information in most cases on their messages; Usenet clients, which, like mail applications, litter their posts with client system information; web browsers, which send client information with each request; and even the ubiquitous telnet client, which sends such information more quietly, but can just as effectively fingerprint an operating system.

Knowing this, one now only needs to harvest the network for this information and map it to source addresses. Various tools, including sniffers, both generic and specialized, and even web searches will yield this information. A rapid analysis of systems can be quickly performed. This works quite well for the white hat and the black hat hacker, as well.

3.1.3.1.1 Mail Clients

One of the largest types of traffic the network sees is electronic mail. Nearly everyone who uses the Internet on a regular basis uses email in those transaction sessions. They not only receive mail, but also send a good amount of mail, too. Because it is ubiquitous, it makes an especially attractive avenue for system fingerprinting and ultimately penetration.

Within the headers of nearly every mail message is some form of system identification. Either through the use of crafted message identification tags, as used by

Eudora and Pine, or by explicit header information, such as headers generated by OutLook clients or CDE mail clients.

The scope of this method, both in terms of information gained and the potential impact, should not be underestimated. If anything, viruses that spread by email, including ones that are used to steal passwords from systems, should illustrate the effectiveness of this method.

3.1.3.1.2 Usenet Clients

In a manner similar to electronic mail, Usenet clients leave significant information in the headers of their posts which reveal information about their host operating systems. One great advantage to Usenet, as opposed to email or even web tra_c, is that posts are distributed. As such, we can be remote and collect data on hosts without their knowledge or ever having to gain entry into their network. Among the various newsreaders commonly used, copious host info is included in the headers. The popular UNIX newsreader 'tin' is among the worst offenders of revealing host information. Operating system versions, processors and applications are all listed in the 'User-Agent' field, and when coupled to the NNTP-Posting-Host information, a remote host fingerprint has been performed. The standard web browsers also leave copious information about themselves and their host systems, as they do with HTTP requests and mail.

3.1.3.1.3 Using Web Traffic

A remarkably simple and highly effective means of fingerprinting a target is to follow the web browsing that gets done from it. Most every system in use is a workstation, and nearly everyone uses their web browsers to spend part of their day. And just about every browser sends too much information in it's 'User-Agent' field. RFC 194513 notes that the 'User-Agent' field is not required in an HTTP 1.0 request, but can be used. The authors state, "user agents should include this field with requests." They cite statistics as well as on the y tailoring of data to meet features or limitations of browsers. The draft standard for HTTP version 1.1 requests, RFC 2616, also notes similar usage of the 'User-Agent' field.

This information can be gathered in two ways. First, we could run a website and turn on logging of the User-Agent field from the client (if it's not already on). Simply generate a lot of hits and watch the data comes in. Get on Slashdot, advertise some pornographic material, or mirror some popular software (like warez) and you're ready to go.

Secondly, we can sniff web traffic on our visible segment. While almost any sniffer will work, oneof the easiest for this type of work is urlsnarf from the dsnipackage from Dug Song.

Examples of browsers that send not only their application information, such as the browser and the version, but also the operating system which the host runs include:
Netscape (UNIX, MacOS, and Windows) Internet Explorer
One shining example of a browser that doesn't send extraneous information is Lynx. On both 2.7 and 2.8 versions, only the browser information is sent, no information about the host.

3.1.3.1.4 Web Server Fingerprinting

In much the same way as one can use the strings sent during requests by the clients to determine what system type is in use, one can follow the replies sent back by the server to determine what type it is. Again we will use ngrep, this time matching the expression 'server:' to gather the web server type. While specifics about the operating system information are lost, this works to passively gather vulnerability information about the target server. This can be coupled to other information to decide how best to proceed with an attack.

3.1.3.1.5 Telnet Clients

While telnet is no longer in widespread use due to the fact that all of its data is sent in plain text, including authentication data, it is still used widely enough to be of use in fingerprinting target systems. What is interesting is that it not only gives us a mechanism to gather operating system data, it gives us the particular application in use, which can be of value in determining a mechanism of entry. This method of

system fingerprinting is not unique to this paper. A setup demonstrated by a security analyst from Bell Labs had a honey-pot system set upthat one would telnet to. An application would fingerprint the client and hence the operating system.

The specification for the telnet protocol describes a negotiation between the client and the host for information such as line speed, terminal type and echoing. What is interesting to note is that each client behaves in a unique way, even different client applications on the same host type. Similarly, the telnet server, running a telnet daemon, can be fingerprinted by following the negotiations with the client. This information can be viewed from the telnet command line application on a UNIX host by issuing the 'toggle options' command at the telneti prompt.

This information can be gathered directly, using a wedge application, or a honey-pot as demonstrated, or it can be sniffed off the network in a truly passive fashion. We discuss below gathering data about both the client system and the server being connected to. The same principles apply to both host identification methods.

CHAPTER: 4
ACTIVE FINGERPRINTING

4. ACTIVE FINGERPRINTING

There are many tools today that are used for remote active operating system fingerprinting. They all have their own fingerprinting techniques. An in-depth analysis of two such tools: Nmap, and Xprobe2 with the purpose to show how these tools work, and to understand the advantages and disadvantages they each offer.

Remote active operating system fingerprinting is the process of determining the identity of a remote host's operating system. This is done by actively sending packets to the remote host and analyzing the responses. Learning which operating system is running on a remote host can be very valuable for both white-hats and black-hats. It's valuable because when vulnerabilities are found they are normally dependent on the OS version.

4.1 ACTIVE IP PACKET FINGERPRINTING

This is the predominant form of OS fingerprinting, provoking the target into eliciting a response and analysing it carefully. A huge amount of information can be gleaned about the response to a carefully crafted network packet. 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 OS.The most common techniques in use include the following:

1) FIN Probing

Here 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 behavior (as defined in RFC 793) is to simply ignore the packet; however, many stacks send a RST packet back.

This difference is the means to begin creating a fingerprint. The correct RFC 793 behaviour is to NOT respond, but many broken implementations such as MS Windows, BSDI, CISCO, HP/UX, MVS, and IRIX send a RESET back. Most current tools utilize this Technique.

2) 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 OS 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. The idea here is to find patterns in the initial sequence numbers chosen by TCP implementations when responding to a connection request. These can be categorized in to many groups such as the traditional 64K (many old UNIX boxes), Random increments newer versions of Solaris, IRIX, FreeBSD, Digital UNIX, Cray, and many others), True random" (Linux 2.0. *, OpenVMS, newer AIX, etc). Windows boxes (and a few others) use a "time dependent" model where the ISN is incremented by a small fixed amount each time period. Needless to say, this is almost as easily defeated as the old 64K behaviour. The machines ALWAYS use the exact same ISN .You can also subclass groups such as random incremental by computing variances, greatest common divisors, and other functions on the set of sequence numbers and the differences between the numbers.It should be noted that ISN generation has important security implications. For more information on this, contact "security expert" Tsutomu "Shimmy" Shimomura at SDSC and ask him how he was owned. Nmap is the first program I have seen to use this for OS identification.

3) The BOGUS Flag Probe

The idea is to set an undefined TCP "flag" (64 or 128) in the TCP header of a SYN packet. Linux boxes prior to 2.0.35 keep the flag set in their response. I have not

found any other OS to have this bug. However, some operating systems seem to reset the connection when they get a SYN+BOGUS packet. This behaviour could be useful in identifying them.

4) Don't Fragment bit

Many operating systems are starting to set the IP "Don't Fragment" bit on some of the packets they send. This gives various performance benefits (though it can also be annoying this is why nmap fragmentation scans do not work from Solaris boxes). In any case, not all OS's do this and some do it in different cases, so by paying attention to this bit we can glean even more information about the target OS.

5) 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 OS identification of machinesthat no listening ports open at all.

The RFCs specify that ICMP error messages quote some small amount of an ICMP message that causes various errors. For a port unreachable message, almost all implementations send only the required IP header + 8 bytes back. However, Solaris sends back a bit more and Linux sends back even more than that. The beauty with this is it allows nmap to recognize Linux and Solaris hosts even if they don't have any ports listening.

6) ICMP Error Message Quenching

Some (smart) operating systems followthe RFC 1812 suggestion to limit the rate at which various error messages are sent. For example, the Linux kernel (in net/ipv4/icmp.h) limits destination unreachable message generation to 80 per 4 seconds, with a 1/4 second penalty if that is exceeded. One way to test this is to send a bunch of packets to some random high UDP port and count the number of

unreachables received. I have not seen this used before, and in fact I have not added this to nmap (except for use in UDP port scanning). This test would make the OS detection take a bit longer since you need to send a bunch of packets and wait for them to return. Also dealing with the possibility of packets dropped on the network would be a pain.

7) . 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 OS that created it. I got this idea from something Theo De Raadt (lead OpenBSD developer) posted to comp.security.unix. As mentioned before, machines have to send back part of your original message along with a port unreachable error. Yet some machines tend to use your headers as 'scratch space' during initial processing and so they are a bit warped by the time you get them back. For example, AIX and BSDI send back an IP 'total length' field that is 20 bytes too high. Some BSDI, FreeBSD, OpenBSD, ULTRIX, and VAXen fuck up the IP ID that you sent them. While the checksum is going to change due to the changed TTL anyway, there are some machines (AIX, FreeBSD, etc.) which send back an inconsistent or 0 checksum. Same thing goes with the UDP checksum. All in all, nmap does nine different tests on the ICMP
errors to sniff out subtle differences like these.

8) ICMP Error Message Type of Service (TOS)

Nearly allimplementations 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. For the ICMP port unreachable messages I look at the type of service (TOS) value of the packet sent back. Almost all implementations use 0 for this ICMP error although Linux uses 0xC0. This does not indicate one of the standard TOS values, but instead is part of the unused (AFAIK) precedence field. I do not know why this is set, but if they change to 0 we will be able to keep identifying the old versions and we will be able to identify between old and new.

9) Fragmentation Handling

This is a favorite technique of Thomas H. Ptacek of Secure Networks, Inc (now owned by a bunch of Windows users at NAI). This takes advantage of the fact that different implementations often handle overlapping IP fragments differently. Some will overwrite the old portions with the new, and in other cases the old stuff has precedence. There are many different probes you can use to determine how the packet was reassembled.

10) 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 not10. This technique is only viable over reliable connections to the remote host and extends scanning time; so, is generally not implemented.

11) TCP Options

These are enhancements to the TCP protocol to improve performance in unreliable or high latency networks7. TCP Options have been added as TCP RFCs over time as needs dictated and t he patterns of compliance in responses can reveal the underlying OS. 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 returned8. Many other differences also exist and are used to a lesser extent.These include:

- ✓ **IPID Sampling**

  Many OSs utilise a system wide counter for IPID generation. Other, more advanced implementations either randomize this number or set it to 0. Information can be leaned from the choice of IPID as to the source OS. Again, predictable IPID values can have important security implications outside of OS fingerprinting, including completely silent port scanning.

- ✓ **TCP Timestamp**

This is a TCP option (and hence is not supported by all IP implementations); however, can be used to determine OS type. It can also be used to determine host uptime if implemented and the update frequency is known.

Nearly all active fingerprinting tools use some or all of the above test to obtain data on a host and compare the results with a database of known OSs. As tools become established the OS fingerprint database becomes more extensive, enhancing the resolution of the tool. The age and popularity of a tool is often evidenced by the size of its database.

4.2 Disadvantages

The problems with active scanning are mainly twofold: first, we can readily firerewall the packets used to fingerprint our system, obfuscating the information; secondly, we can detect it quite easily. Because of this, it is less attractive for a truly stealthy adversary.

CHAPTER: 5
DESIGN AND IMPLEMENTAION

5. DESIGN

As the OS vendors have become caution about OS fingerprinting already and countermeasures against classical techniques have been designed and successfully implemented. To defend the attacks based upon “stack fingerprinting” using different and various combinations of network& transport protocols is necessary.

Stack *Fingerprinting:* Most security holes and vulnerabilities are Operating Systems dependent. So the possibility of remote OS detection should be one of the main concerns of system and/or security administrators. Remote OS detection is not a new issue. For years, several implementations of TCP/IP services have been offering Clues and information about their host's OS. FI'P, TELNET, HTTP, and DNS servers are well known examples of this behavior. However the fact that the information is usually quite incomplete or even intentionally misleading, encouraged the apparition of some basic remote OS detectors.

These first scanners worked by probing for well known differences that existed among some selected operating systems TCP/IP implementations. Because of the small number of probes used and the limited amount of differences *(fingerprints)* analyzed, they were able to identify at most a dozen of OS. Two recent scanners, NMAP and Xprobe2, introduced a new dimension in fingerprint scanning. Nmap additionally incorporated an important amount of OS detection techniques and defined a *template* structure to describe fingerprints. Xprobe2 made use of fuzzy approach to match the signatures of the OS with learning database

The security framework to play defensive against operating system fingerprinting can’t be designed unless we understand the way it is presently being done. To accomplish this measure the current tools the implementation and working principle

need to be discussed. Further followed by specific tools like Nmap and Xprobe2 detail techniques. Morph is slang for to change shape, from the word metamorphosis.

5.1 GENERAL METHOD

When someone with half a clue decides to attack your system, they will first try to identify the operating system. For the professional penetration tester or hacker, operating system (OS) identification is an essential step in probing.

So it is must to understand the procedure used by most of the OS fingerprinting attempts before safe guarding from them General approach for any OS fingerprinting using stack fingerprinting is consisting of following three phases:

PHASE #1 HOST DETECTION:

Host discovery is a term used to describe a certain phase of a penetration test, where one attempts to determine the accessible hosts on a network. Many times if a firewall rule set is written explicitly, it is difficult to accurately determine the number of hosts that are behind a firewall. Basic about this phase is to discover a reachable node .Finding out whether it's up or down.

Some times in different and difficult situations the command options are improved in discovery phase of a penetration test or vulnerability assessment.

PHASE # 2 PORT SCANNING:

It is one of the most popular techniques used in the wild to discover and map services that are listening on a specified port. Using this method an attacker can then create a list of potential weaknesses and vulnerabilities in the proposed open port leading to exploitation and compromise of a remote host.

One of the primary stages in penetrating/auditing a remote host is to firstly compose a list of open ports, using one or more of the techniques described below. Once this has been established, the results will help an attacker identify various services that

are running on that port using an RFC-compliant port list, allowing further compromise of the remote host after this initial discovery.

Port scanning techniques take form in three specific and differentiated ways.

- open scanning
- half-open scanning
- stealth scanning

Each of these techniques allow an attack to locate open/closed ports on a server, but knowing to use the correct scan in a given environment depends completely on the type of network topology, IDS, logging features a remote host has in place.

A OPEN SCAN METHODS

This type of scan method involves opening a full connection to a remote host using a typical three-way TCP/IP handshake. A standard transaction involves issuing the following flags to create an accepted connection:

client -> SYN
server -> SYN|ACK
client -> ACK

The above example shows a port answering our initial connection request with a SYN|ACK. This response means the port the packet was targeted to is in the LISTENING (open) state. Once this full handshake has taken effect, the connection will be terminated by the client allowing a new socket to be created/called allowing the next port to be checked, until the maximum port threshold has been reached. Reversely, taking a look at a response from a closed port would reveal the following:

client -> SYN
server -> RST|ACK
client -> RST

The RST|ACK flags returned by the server is telling the client to tear down the connection attempt since the port is not in LISTENING state thus is closed. This method is created through connect() system call, allowing almost instantaneous

identification of an open or closed port. If the connect() call returns true, the port is open, else the port is closed.

Since this technique issues a three-way handshake to connect to an arbitrary host, a spoofed connection is impossible, that is to say a client can not manipulate the true source IP, as a spoofed connection attempt involves sending a correct sequence number as well as setting the correct return flags to setup for data transaction. Obviously this technique is easily identifiable on any inbound traffic because it opens a full connection, thus all IDS and firewalls are able to detect and block against this scan. However, because the connect() method uses the three way handshake, results of this scan are about as accurate as you could get to determine open/closed ports.

B. HALF OPEN SCAN METHODS OR (SYN SCANNING)

The term 'half-open' applies to the way the client terminates the connection before the three-way handshake is completed. As such, this scan method will often go unlogged by connection based IDS', and will return fairly positive results (reliability of open/closed port recognition).

The implementation of this scan method is similar to a full TCP connect() three way handshake except instead of sending ACK responses we immediately tear down the connection. A demonstration of this technique is necessary to show a half open transaction:

client -> SYN
server -> SYN|ACK
client -> RST

This example has shown the target port was open, since the server responded with SYN|ACK flags. The RST bit is kernel oriented, that is, the client need not send another packet with this bit, since the kernel's TCP/IP stack code automates this. Inversely, a closed port will respond with RST|ACK.

client -> SYN
server -> RST|ACK

As is displayed, this combination of flags is indicative of a non- listening port. Although, this technique has become rather easy to detect by many IDS, owing to the fact that a paramount of Denial of Service (DoS) utilities base their Attacks by sending excess SYN packets. Fairly standard intrusion detection systems are no

doubt capable of logging these half-open scans: TCP wrappers, SNORT, Courtney, iplog, to a name a few, thus the effectiveness has dithered over recent years.

C. STEALTH SCANNING

The definition of a "stealth" scan has varied over recent years, originally the term was used to describe a technique that avoided IDS and logging, now know as "half-open" scanning. However, nowadays stealth is considered to be any scan that is concerned with a few of the following:

- setting individual flags (ACK, FIN, RST, ..)
- NULL flags set
- All flags set
- bypassing filters, firewalls, routers
- appearing as casual network traffic
- varied packet dispersal rates

Broadly this scan be called TCP Fragmenting, where the different flags and other offset of TCP packet are targeted to clearly scan the port from the response. Actually, TCP fragmenting is not a scan method so to speak, although it employs a method to obscure scanning implementations by splitting the TCP header into smaller fragments. IP reassembly on the server-side can often lead to unpredictable and abnormal results (IP headers carrying data can be fragmented). Many hosts are unable to parse and reassemble the tiny packets and thus may cause crashes, reboots, or even network device monitoring dumps. Alternatively, these tiny packets may be potentially blocked by IP fragmentation queues in the kernel or caught by a stately firewall ruleset.

Since much intrusion detection systems use signature-based mechanisms to signify scanning attempts based on IP and/or the TCP header, fragmentation is often able to defeat this type of packet filtering and detection, and naturally the scan will go undiscovered.

A minimally allowable fragmented TCP header must contain a destination and source port for the first packet (8 octect, 64 bit), typically the initialized flags in the next, allowing the remote host to reassemble the packet upon arrival. The actual reassembly is established through an IPM (internet protocol module) that identifies the fragmented packets by the field equivalent values of Source, Destination, Protocol, Identification.

PHASE #3 OS DETECTION:

It is the process of actively determining a targeted network node's underlying operating system by probing the targeted system with several packets and examining the response(s) received.

Once, the classical techniques of OS fingerprinting met with stronger countermeasure form both the vendors and users new modern techniques has surfaced. Eventually, more advanced techniques based on stack querying came about. Stack querying means to actively send packets to the network stack on the remote host and analyze the responses. This idea takes advantage of each OS vendor's network stack implementation. The first method to use stack querying was aimed at the TCP stack. It involves sending standard and non-standard TCP packets to the remote host and analyzing the responses. The next method was known as ISN (Initial Sequence Number) analysis. This identifies the differences in the random number generators found in the TCP stack. Up until that point all of the stack querying methods were found by looking at the TCP protocol. Later, researchers found a new method that used the ICMP protocol. The method is known as ICMP response analysis. It involves sending ICMP messages to the remote host and analyzing the responses.

TCP/IP stacks behavior of a targeted network element when probed with several legitimate and/or malformed packets. The received results would then be compared to a signature database in an effort to find an appropriate match.

Actually, TCP fingerprinting works by sending TCP packets to a port and noticing how the TCP stack responds. Many of the specifications for TCP/IP are left open to

interpretation, so each vendor implements the TCP/IP stack a little differently, creating a unique identifier or fingerprint.

Typically, seven packets are sent to a destination port using different flag variants -- including SYN, SYN/ACK, FIN, FIN/ACK, SYN/FIN, PSH, and SYN+Reserved. Based on the data returned, the results can be mapped to a particular operating system and version, decoding the OS for users of the tools.

5.2. TOOLS & TECHNIQUES

Discussion on Active fingerprinting tools and there actual behaviour becomes the relevant path in order to understand the countermeasures. Starting with TPC/IP Stack Fingerprinting technique of Nmap and followed by ICMP scanning technique of Xprobe2

5.2.1 NMAP

For the purpose of this paper focusing on the remote OS detection prevention method that Nmap version 3.81 since it was the current version at the time of this writing. NMAP interrogates the target machine's TCP/IP stack by sending it eight different packets and observing the response.

Each of the eight different packets are specially crafted to put the target machine in a position where there is a high probability that two things will happen. The target operating system's TCP/IP stack will respond unique in comparison to another operating system's TCP/IP stack. The target operating system TCP/IP stack will respond in a consistent manner. Knowing how a given operating system's TCP/IP stack would respond in advance to each of the eight tests allows NMAP to determine with a high degree of accuracy not only which operating system the target is running, but also what version it is running as well. The crafted test packets are sent one at a time by the source machine running nmap:

TEST	Description
T1	TCP packet with the SYN, and ECN-Echo flags enabled is sent to an open TCP port.
T2	Sending a TCP packet with no flags enabled to an open TCP port. This type of packet is known as a NULL packet.
T3	Sending a TCP packet with the URG, PSH, SYN, and FIN flags enabled to an open TCP port.
T4	Sending a TCP packet with the ACK flag enabled to an open TCP port.
T5	Snding a TCP packet with the SYN flag enabled to a closed TCP port.
T6	Sending a TCP packet with the ACK flag enabled to a closed TCP port.
T7	Sending a TCP packet with the URG, PSH, and FIN flags enabled to a closed TCP port. closed TCP port.
PU (port unreachable test)	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.
TSeq (TCP sequenceability test)	The test tries to determine the sequence generation patterns of the 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.

Table 5.1: Nmap crafted packets

Several different metrics are observed for each of the first seven tests to help determine the target operating system. They are:

1. Whether or not the target host responded.
2. Whether or not the target host responded with a packet that had the "Don't Fragment" bit set.
3. The Window Size set by the target host in the response packet.
4. The relationship of the acknowledgement number of the TCP packet sent in response to NMAP's prior TCP packet.
5. Flags set in the TCP packet sent in response.
6. TCP options that are in the responding packet.
7. The first test (Tseq) and last test (PU) uses different metrics, but the same principles apply.

5.2.1.1 Example OS Fingerprinting using Nmap

On the bases of the resultant packets send back from the target the OS signature are captured and matched . using the following matrices:

Metric	Valid Values	Description
Resp	**Y** = There was a response **N** = There was no response	Whether or not the host responded to the test packet by sending a reply.
DF	**Y**=DF was set **N** = DF was not set	Whether or not the host responding to the test packet sent the "Don't Fragment" bit in response.
W	Can be a two-byte integer expressed in hexadecimal.	Window advertisement size sent by the host responding to the test packet.
ACK	**0** = ack zero **S** = ack sequence number **S++** = ack sequence number + 1	The acknowledgement sequence number response type.
Flags	**S** = SYN **A** = ACK **R** = RST **F** = FIN **U** = URG **P** = PSH	Indicate what flags were set in the responding packet.
Ops	**M** = MSS **E** = Echoed MSS **W** = Window Scale **T** = Timestamp **N** = No Option	Options sent back by the host responding to the test packet. There can be any number of options set (including none) in any order.

Table 5.2: Nmap response matrices

The First line that state the operating system (and sometimes version) that owns the fingerprint. The next line that begins with **TSeq**, identifies the method for calculating TCP sequence numbers for a given TCP session. The lines that follow, starting with T1 through PU, are descriptive of how that operating system fingerprinted would respond to the given test.

```
nmap-3.81>nmap -O 172.31.5.113
Starting nmap 3.81 ( http://www.insecure.org/nmap ) at 2005-05-18 12:49 India
Standard Time
WARNING:  RST from port 111 -- is this port really open?
```

Insufficient responses for TCP sequencing (2), OS detection may be less accurate
Interesting ports on RESEARCH6 (172.31.5.113):
(The 1654 ports scanned but not shown below are in state: closed)
PORT STATE SERVICE
111/tcp open rpcbind
135/tcp open msrpc
139/tcp open netbios-ssn
445/tcp open microsoft-ds
707/tcp open unknown
1050/tcp open java-or-OTGfileshare
1058/tcp open nim
5800/tcp open vnc-http
5900/tcp open vnc
MAC Address: 00:08:A1:7E:4E:24 (CNet Technology)
Device type: general purpose
Running: Microsoft Windows 95/98/ME
OS details: Microsoft Windows Millennium Edition (Me)

Nmap finished: 1 IP address (1 host up) scanned in 6.532 seconds

Source Destination Summary For the T1-T7 and PU test

[172.31.5.116] [172.31.5.113] TCP: D=135 S=56414 SYN SEQ=3375903318 LEN=0 WIN=2048
[172.31.5.116] [172.31.5.113] TCP: D=135 S=56415 WIN=2048
[172.31.5.116] [172.31.5.113] TCP: D=135 S=56416 SYN FIN URG SEQ=3375903318 LEN=0 WIN=2048
[172.31.5.116] [172.31.5.113] TCP: D=135 S=56417 ACK=0 WIN=4096
[172.31.5.116] [172.31.5.113] TCP: D=1 S=56418 SYN SEQ=3375903318 LEN=0 WIN=2048
[172.31.5.116] [172.31.5.113] TCP: D=1 S=56419 ACK=0 WIN=4096
[172.31.5.116] [172.31.5.113] TCP: D=1 S=56420 FIN URG SEQ=3375903318 LEN=0 WIN=4096
[172.31.5.113] [172.31.5.116] TCP: D=56414 S=135 SYN ACK=3375903319 SEQ=1982576985 LEN=0 WIN=65535
[172.31.5.116] [172.31.5.113] TCP: D=135 S=56414 RST WIN<<10=0
[172.31.5.113] [172.31.5.116] TCP: D=56415 S=135 RST ACK=3375903318 WIN=0
[172.31.5.116] [172.31.5.113] UDP: D=1 S=56407 LEN=308
[172.31.5.113] [172.31.5.116] TCP: D=56416 S=135 SYN ACK=3375903319 SEQ=3856135382 LEN=0 WIN=65535
[172.31.5.116] [172.31.5.113] TCP: D=135 S=56416 RST WIN<<10=0
[172.31.5.113] [172.31.5.116] TCP: D=56417 S=135 RST WIN=0
[172.31.5.113] [172.31.5.116] TCP: D=56418 S=1 RST ACK=3375903319 WIN=0
[172.31.5.113] [172.31.5.116] TCP: D=56419 S=1 RST WIN=0
[172.31.5.113] [172.31.5.116] TCP: D=56420 S=1 RST ACK=3375903319 WIN=0
[172.31.5.113] [172.31.5.116] ICMP: Destination unreachable (Port unreachable)

Source Destination Summary For TSeq test

[172.31.5.116] [172.31.5.113] TCP: D=135 S=56408 SYN SEQ=3375903319 LEN=0 WIN=3072
[172.31.5.113] [172.31.5.116] TCP: D=56408 S=135 SYN ACK=3375903320 SEQ=1634073962 LEN=0 WIN=65535
[172.31.5.116] [172.31.5.113] TCP: D=135 S=56408 RST WIN<<10=0
[172.31.5.116] [172.31.5.113] TCP: D=135 S=56409 SYN SEQ=3375903320 LEN=0 WIN=1024
[172.31.5.113] [172.31.5.116] TCP: D=56409 S=135 SYN ACK=3375903321 SEQ=1378387651 LEN=0 WIN=65535
[172.31.5.116] [172.31.5.113] TCP: D=135 S=56409 RST WIN<<10=0
[172.31.5.116] [172.31.5.113] TCP: D=135 S=56410 SYN SEQ=3375903321 LEN=0 WIN=2048
[172.31.5.113] [172.31.5.116] TCP: D=56410 S=135 SYN ACK=3375903322 SEQ=2771984018 LEN=0 WIN=65535
[172.31.5.116] [172.31.5.113] TCP: D=135 S=56410 RST WIN<<10=0
[172.31.5.116] [172.31.5.113] TCP: D=135 S=56411 SYN SEQ=3375903322 LEN=0 WIN=2048
[172.31.5.113] [172.31.5.116] TCP: D=56411 S=135 SYN ACK=3375903323 SEQ=1187915667 LEN=0 WIN=65535
[172.31.5.116] [172.31.5.113] TCP: D=135 S=56411 RST WIN<<10=0

```
[172.31.5.116] [172.31.5.113] TCP: D=135 S=56412 SYN SEQ=3375903323 LEN=0 WIN=3072
[172.31.5.113] [172.31.5.116] TCP: D=56412 S=135 SYN ACK=3375903324 SEQ=2260369583 LEN=0
WIN=65535
[172.31.5.116] [172.31.5.113] TCP: D=135 S=56412 RST WIN<<10=0
[172.31.5.116] [172.31.5.113] TCP: D=135 S=56413 SYN SEQ=3375903324 LEN=0 WIN=3072
[172.31.5.113] [172.31.5.116] TCP: D=56413 S=135 SYN ACK=3375903325 SEQ=212338365 LEN=0
WIN=65535
[172.31.5.116] [172.31.5.113] TCP: D=135 S=56413 RST WIN<<10=0
```

Fingerprint Windows 2000

TSeq(Class=RI%gcd=<5%SI=>BBB&<FFFF)

T1(DF=Y%W=402E%ACK=S++%Flags=AS%Ops=MNWNNT)

T2(Resp=Y%DF=N%W=0%ACK=S%Flags=AR%Ops=)

T3(Resp=Y%DF=Y%W=402E%ACK=S++%Flags=AS%Ops=MNWNNT)

T4(DF=N%W=0%ACK=O%Flags=R%Ops=)

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

T6(DF=N%W=0%ACK=O%Flags=R%Ops=)

T7(DF=N%W=0%ACK=S++%Flags=AR%Ops=)

PU(DF=N%TOS=0%IPLEN=38%RIPTL=148%RID=E%RIPCK=E%UCK=E%ULEN=134%DAT=E
)

TSeq(Class=RI%gcd=<5%SI=>BBB&<FFFF)

This line represents the results of the TCP sequenceability test, ries 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=<5means that the greatest common divisor is less than five, which means that the sequence numbers don’t have any large factors in common. means that the SI=>BBB&<FFFF sequence increments are more than 0xBBB in hexadecimal, and less than 0xFFFF in hexadecimal. 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.

T1 (DF=Y%W=402E%ACK=S++%Flags=AS%Ops=MNWNNT)

This test states that the response packet of the target host to NMAP sending a SYN packet with options to it had the following characteristics: Resp= Resp is not defined;

which means the metric is satisfied whether or not the target replies. DF=Y The "Don’t Fragment" bit was set. W=402E The window size was 402E. ACK=S++ Acknowledgement number was one plus the initial sequence number. Flags=AS The packet had the SYN/ACK flags set. Ops=MNWNNT The packet had the following option flags set in this order: MNWNNT

T2(Resp=Y%DF=N%W=0%ACK=S%Flags=AR%Ops=)

This line represents the results of test 2. DF=N means that the Don’t fragment flag in the IP header was not 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.

T3(Resp=Y%DF=Y%W=402E%ACK=S++%Flags=AS%Ops=MNWNNT)

DF=Y means that the Don’t fragment flag in the IP header was enabled. W=402E The window size was 402E. 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= MNWNNT The packet had the following option flags set in this order: MNWNNT

T4(DF=N%W=0%ACK=O%Flags=R%Ops=)

Resp is not defined; which means the metric is satisfied whether or not the target replies. DF=N means that the Don’t fragment flag in the IP header was not enabled. W=0 means that the window size in the TCP header is 0.. ACK=O acknowledgment number in the TCP header is 0.. Flags=R means that the RST flags in the TCP header were enabled. Ops= means that there weren’t any options sent back.

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

Resp is not defined; which means the metric is satisfied whether or not the target replies. DF=N means that the Don’t fragment flag in the IP header was not 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=N%W=0%ACK=O%Flags=R%Ops=)

Resp is not defined; which means the metric is satisfied whether or not the target replies. DF=N means that the Don't fragment flag in the IP header was not enabled. W=0 means that the window size in the TCP header is 0.. ACK=O acknowledgment number in the TCP header is 0.. Flags= means that the no flags in the TCP header were enabled. Ops= means that there weren't any options sent back.

T7(DF=N%W=0%ACK=S++%Flags=AR%Ops=)

Resp is not defined; which means the metric is satisfied whether or not the target replies. DF=N means that the Don't fragment flag in the IP header was not 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.

PU(DF=N%TOS=0%IPLEN=38%RIPTL=148%RID=E%RIPCK=E%UCK=E|F%ULEN=134%DAT=E)

This line represents the results of the port unreachable test. DF=N means that the Don't fragment flag in the IP header was not enabled. TOS=0 means that the type of service (TOS) in the IP header was0. IPLEN=38 means that the total length in the IP header. 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.

5.2.1.2 DATA CAPTURED

By the sniffer when Nmap attacks to detect the OS of remote machine.

For OPEN TCP PORT ATTACKS:

1) The attacks start with sending in TCP packets to discover that is the host up, down and/or behind firewall. To do so it send number of packets to different ports and view their status as closed, open or firewalled. Represented by **BLACK** line where there is linear constant increase in the tcp packet number per second destined to port number ++1.
2) The T1 a single TCP packet destined to open TCP port with syn and ech flag enabled is represented by **RED** line. PORT:N
3) The T2 packets destined to the remote host with no flag enabled is represented by **GREEN** impulse meaning that that just once a single packet was send with no flag set. PORT:N++
4) The T3 packets destined to remote host with fin, urg, push and syn is represented by **BLUE** Frequency bar meaning that only one such packet was generated by nmap and send. PORT:N++
5) The T4 packets ack flag enabled and destined is represented by **PINK** impulse and means that at the last tcp packet send to the destination with port ++1 was this one only. PORT:N++
6) The TSeq test can be verified when analysing the stable black line sending TCP packets to same port to get the Sequence number sampling to get the ISN

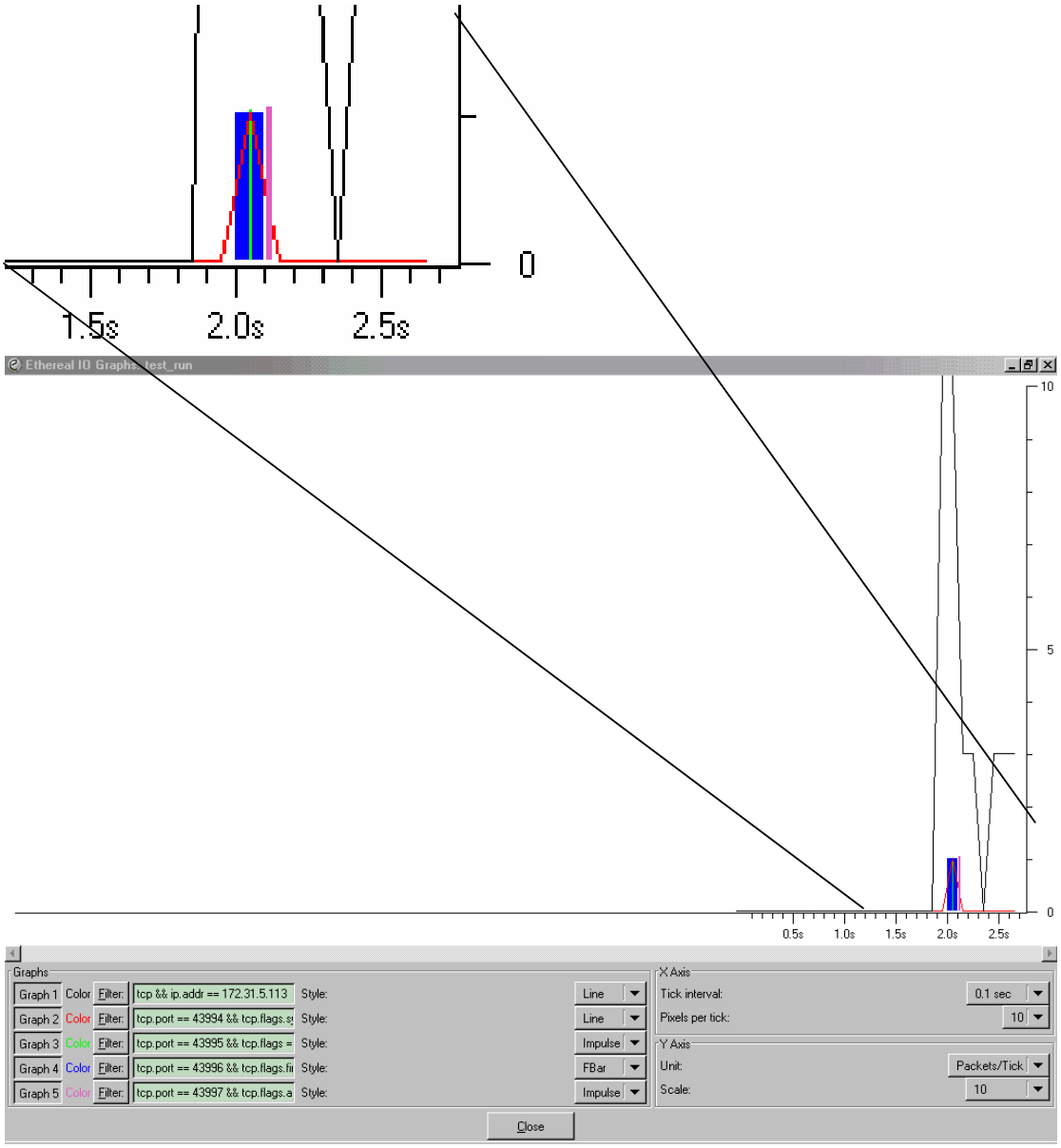


Figure5.1: OPEN TCP Attack by Nmap

For CLOSED TCP PORT ATTACKS:

1) To start the attacks to close port identification and selection of close TCP port is to to be made. Represented by **BLACK** line where there is linear constant increase in the TCP packet number per second destined to port number ++1 is send to identify and mark them
2) The T5 a single TCP packet destined to close TCP port with syn flag enabled is represented by **RED** line port:N

3) The packets destined to the remote host with ack flag enabled is represented by **GREEN** impulse meaning that that just once a single packet was send with no flag set. PORT:N++
4) The packets destined to remote host with fin, urg and push is represented by **BLUE** impulse meaning that only one such packet was generated By Nmap and send. PORT:N++

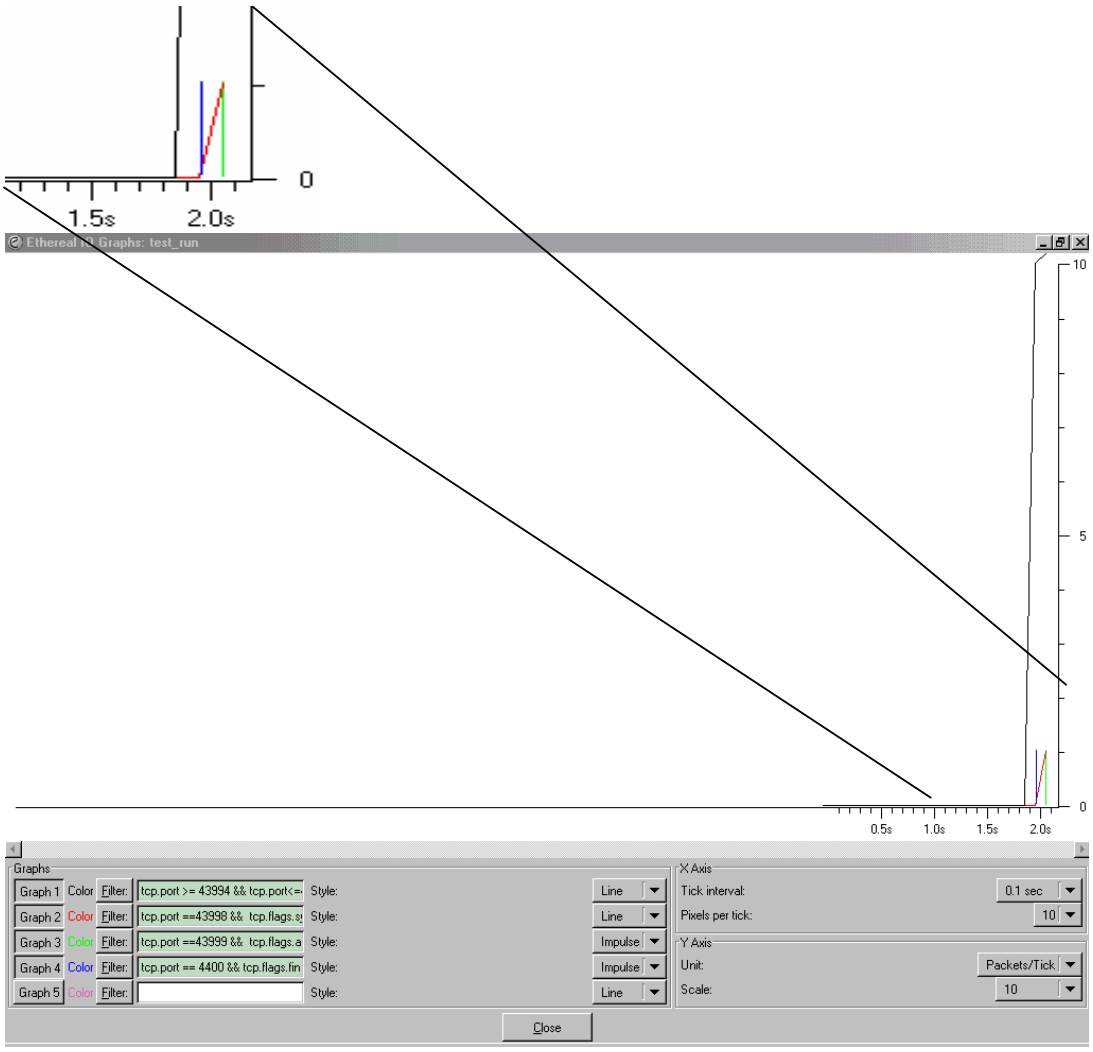


Figure5.2: Closed TCP Port attacks by Nmap

For CLOSED UDP PORT ATTACKS:

The **BLACK** line represents the number of UDP packets send to the closed UDP port. which ends abruptly once the right port number identified. After that block of UDP

packets are send to get the ICMP port unreachable message for PORT unreachable test represented by **RED** frequency bar.

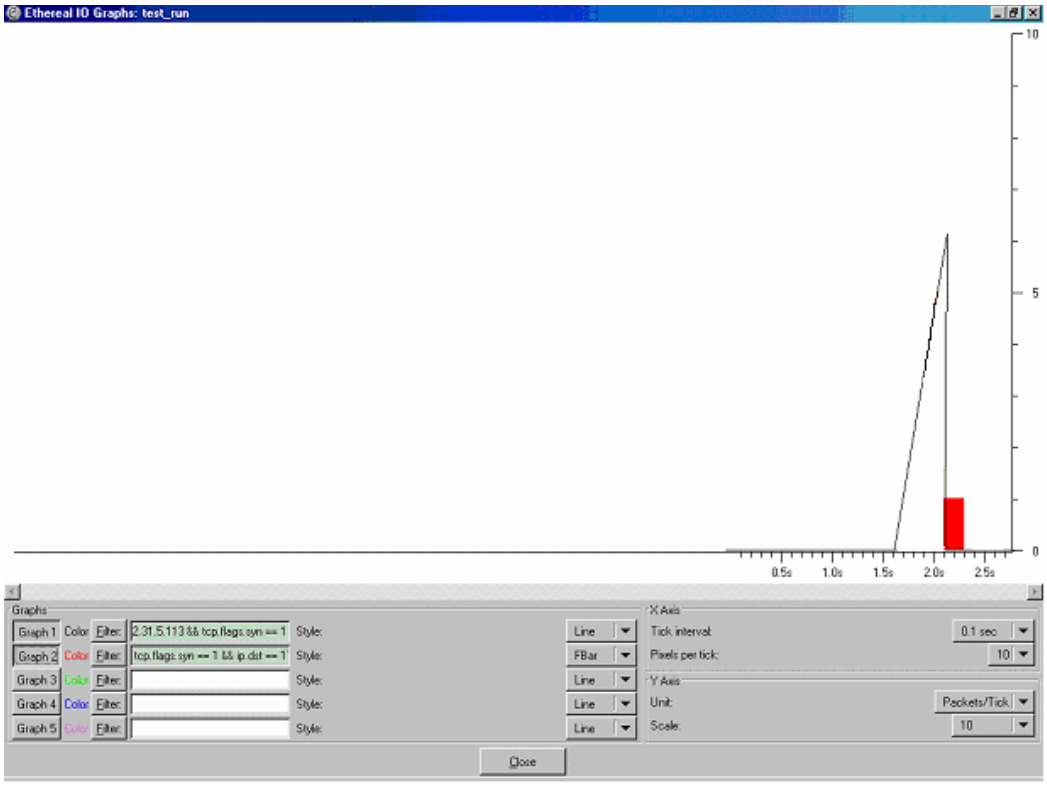


Figure5.3: Closed UDP attack by Nmap

5.2.1.3 NMAP Advantages

The Nmap tool offers many advantages over other remote active OS fingerprinting tools. Here are some reasons users might want to utilize this tool for its OS detection method:

- "Half-open" scanning2 is supported. The TCP SYN scan is one of these scan types. Theadvantage to this scan type is that it doesn't complete the TCP three-way handshake whichmeans that it will often not be logged by basic intrusion detection systems.
- The tests that Nmap performs include multiple flags and protocols so that it can still make an OS guess even if filtering devices block some of the tests. Some tests might not work because filtering devices might block protocols like ICMP from entering their network.

- There is a large OS signature database. For example, the latest Nmap (3.81) recognizes 867 fingerprints while the latest Xprobe2 (2.1) recognizes.
- There is a lot of support for network devices like routers, firewalls, switches, and printers.
- Strict signature matching is utilized by default, while fuzzy matching can be enabled byspecifying the undocumented --fuzzy option. It has built-in learning functions.

5.2.2 Xprobe2

IT is a remote active OS fingerprinting tool. It is designed with a different approach to OS fingerprinting. The Xprobe2 OS detection method identifies the type of the remote OS with a matrix based fingerprinting approach. This approach is also known as ‘’fuzzy’’ matching. In this case port scan against the target machine is not done. Xprobe2 needs at least one closed UDP port to work. Xprobe2 relies primarily on the use of the ICMP protocol.

Xprobe2 is modular in design so it has the capability to accept new modules, or other fingerprinting tests. Xprobe2 works by running different modules or tests against the target machine.AS follows

TEST	Description
REACHABILITY MODULES	
RM#1 ICMP echo (ping) test	ICMP packet with an ICMP echo request message is sent.
RM#2 TTL distance test	TCP packet with the SYN flag enabled is sent to a TCP port.
FINGERPRINTING MODULE	
FM#1	ICMP packet with an ICMP echo request message is sent
FM#2	ICMP packet with an ICMP timestamp request message is sent.
FM#3	ICMP address mask test. In this test an ICMP packet with an ICMP address mask request message is sent.
FM#4	ICMP packet with an ICMP information request message is sent.
FM#5	UDP packet acting as a DNS query result is sent. The goal is to elicit an ICMP packet with an ICMP port unreachable message back from the target machine.

Table 5.3: Xprobe2 crafted packets

Several different metrics are observed for each of tests to help determine the target operating system. They are:

1. The goal is to elicit an ICMP packet with an ICMP echo reply message back from the target machine.
2. The goal is to elicit, a TCP packet with the SYN and ACK flags enabled meaning the TCP port is opened or a TCP packet with the RST flag enabled meaning the TCP port is closed back from the target machine. If no response is received another TCP packet with the same options is sent to a different TCP port, with the same goal.
3. And the rest of the fingerprinting modules analysis the ICMP responses to gather data for matching against the fingerprinting database

5.2.2.1 Example OS Fingerprinting using Xprobe2

The usage of statistical analysis ('fuzzy logic') to match between probe response(s) to its signature database is not that easily done as Xporbe2 make use of well defined modules which capture specify data from different ICMP response packets send in by remote host.

Explaining in terms of the modules defined above:

Example entry

Module	Captured response
[ICMP ECHO Probe]	icmp_echo_reply = [y, n] icmp_echo_code = [0, !0] icmp_echo_ip_id = [0, !0, SENT] icmp_echo_tos_bits = [0, !0] icmp_echo_df_bit = [0, 1] icmp_echo_reply_ttl = [>< decimal num
[ICMP Timestamp Probe]	icmp_timestamp_reply = [y, n] icmp_timestamp_reply_ttl = [>< decimal num] icmp_timestamp_reply_ip_id = [0, !0, SENT]
[ICMP Address Mask Request Probe]	icmp_addrmask_reply = [y, n] icmp_addrmask_reply_ttl = [>< decimal num] icmp_addrmask_reply_ip_id = [0, !0, SENT]
[ICMP Information Request Probe]	icmp_info_reply = [y, n] icmp_info_reply_ttl = [>< decimal num] icmp_info_reply_ip_id = [0, !0, SENT]
[UDP -> ICMP Unreachable probe]	icmp_unreach_reply = [y, n] icmp_unreach_echoed_dtsize = [8, 64, >64] icmp_unreach_reply_ttl = [>< decimal num]

	icmp_unreach_precedence_bits = 0xc0, 0, (hex num) icmp_unreach_df_bit = [0 , 1] icmp_unreach_ip_id = [0, !0, SENT]
Original_data_echoed_ with_the_UDP_Port_ Unreachable_error_message	icmp_unreach_echoed_udp_cksum = [0, OK, BAD] icmp_unreach_echoed_ip_cksum = [0, OK, BAD] icmp_unreach_echoed_ip_id = [OK, FLIPPED] icmp_unreach_echoed_total_len = [>20, OK, <20] icmp_unreach_echoed_3bit_flags = [OK, FLIPPED]
[TCP SYN | ACK Module]	tcp_syn_ack_tos = [0, <value>] tcp_syn_ack_df = [0 , 1] tcp_syn_ack_ip_id = [0 , !0, SENT] tcp_syn_ack_ttl = [>< decimal num]
Information from the TCP header	tcp_syn_ack_ack = [<value>] tcp_syn_ack_window_size = [<value>] tcp_syn_ack_options_order = ["order"] tcp_syn_ack_wscale = [<value>, NONE] tcp_syn_ack_tsval = [0, !0, NONE] tcp_syn_ack_tsecr = [0, !0, NONE]
[TCP RST|ACK]	tcp_rst_reply = [y ,n] tcp_rst_df = [0, 1] tcp_rst_ip_id_1 = [0, !0] tcp_rst_ip_id_2 = [0, !0] tcp_rst_ip_id_strategy = [0, I, R] tcp_rst_ttl = [>< decimal num]

Table 5.4: Xprobe2 response matrices

EXAPMLE EXPLANATION

fingerprint {OS_ID = "Microsoft Windows 2000 Server Service Pack 4"

Module A icmp_echo_reply = y icmp_echo_code = 0 icmp_echo_ip_id = !0 icmp_echo_tos_bits = 0 icmp_echo_df_bit = 1 icmp_echo_reply_ttl = < 128	icmp_echo_reply = y an ICMP timestamp reply message from the target machine. icmp_echo_code = 0 code in the ICMP header does equal 0. icmp_echo_ip_id = !0 identification number in the IP header does not equal 0. icmp_echo_tos_bits =0 TOS or Differentiated Services Field in the IP header does equal 0. icmp_echo_df_bit = 1 Don't fragment flag in the IP header was enabled so is therefore equal to 1. icmp_echo_reply_ttl =<128 time to live in the IP header is less than or equal to 255.
Module B	icmp_timestamp_reply = y means that we

icmp_timestamp_reply = y icmp_timestamp_reply_ttl = <128 icmp_timestamp_reply_ip_id = !0	received an ICMP timestamp reply message from the target machine. icmp_timestamp_reply_ttl = <255 means that time to live in the IP header is less than or equal to255. icmp_timestamp_reply_ip_id = !0 means that the identification number in the IP header does not equal 0.
Module C icmp_addrmask_reply = n icmp_addrmask_reply_ttl = <128 icmp_addrmask_reply_ip_id = !0	icmp_addrmask_reply = n means that we didn't receive an ICMP address mask reply message from the target machine. The rest of the variables are sent back to their default values
Module D icmp_info_reply = n icmp_info_reply_ttl = <128 icmp_info_reply_ip_id = !0	icmp_info_reply = n means that we didn't receive an ICMP information request reply message from the target machine. The rest of the variables are sent back to their default values.
Module E icmp_unreach_reply = y icmp_unreach_echoed_dtsize = 8 icmp_unreach_reply_ttl = <128 icmp_unreach_precedence_bits = 0 icmp_unreach_df_bit = 0 icmp_unreach_ip_id = !0	icmp_unreach_echoed_dtsize = 8 means that the amount of data echoed back to us with the ICMP port unreachable error message from the original UDP packet sent to the target machine is equal to 8 bytes. icmp_unreach_reply_ttl= <128 means that the time to live in the IP header is less than or equal to 128. icmp_unreach_precedence_bits = 0 means that the precedence flag in the IP header isn't enabled so is therefore equal to 0. icmp_unreach_df_bit = 0 means that the Don't fragment flag in the IP header wasn't enabled and is therefore equal to 0. icmp_unreach_ip_id = !0 means that identificationnumber in the IP header does not equal 0.
icmp_unreach_echoed_udp_cksum = OK icmp_unreach_echoed_ip_cksum = OK icmp_unreach_echoed_ip_id = OK icmp_unreach_echoed_total_len = OK icmp_unreach_echoed_3bit_flags = OK	icmp_unreach_echoed_udp_cksum = OK means that the checksum in the UDP header echoed back to us with the ICMP port unreachable error message from the original UDP packet is correct. icmp_unreach_echoed_ip_cksum = OK means that the checksum in the IP header echoed back to us with the ICMP port unreachable error message from the original UDP packet is correct. icmp_unreach_echoed_ip_id = OK means that the identification number in the IP header echoed back to us with the ICMP port unreachable error message from the original UDP packet is correct. icmp_unreach_echoed_total_len = OK means that the total length in the IP header echoed

	back to us with the ICMP port unreachable error message from the original UDP packet same. icmp_unreach_echoed_3bit_flags = OK means that the 3-bit flags in the IP header echoed back to us with the ICMP port unreachable error message from the original UDP packet is correct.
Module F tcp_syn_ack_tos = 0 tcp_syn_ack_df = 1 tcp_syn_ack_ip_id = !0 tcp_syn_ack_ttl = <128	tcp_syn_ack_tos = 0The value of the received TOS field is 0. tcp_syn_ack_df = 1 The IP Header's Don't Fragment Bit is set. tcp_syn_ack_ip_id = !0 The IP header's IP ID value of the received SYN | ACK is any value other than zero. tcp_syn_ack_ttl = <128 The IP Header's Time-to-Live field value is 128
Information from the TCP header tcp_syn_ack_ack = 1 tcp_syn_ack_window_size= 65535,64240,17520 tcp_syn_ack_options_order = "MSS NOP WSCALE NOP NOP TIMESTAMP NOP NOP SACK" tcp_syn_ack_wscale = 0	tcp_syn_ack_ack = 1 The values of the received ACK minus the value of the sent SYN is 1.tcp_syn_ack_window_size = 65535,64240, 17520 The initial window size with the SYN | ACK. tcp_syn_ack_options_order = The OPTIONS order received with the SYN ACK. tcp_syn_ack_wscale 0 The numerical value of the WSCALE is present and equal to 0

Table 5.5: OS Fingerprint explanation

5.2.2.2 Xprobe2 Advantages

The Xprobe2 tool offers many advantages over other OS fingerprinting tools. Here are some reasonsusers might want to utilize this tool for its OS detection method:

- It is very quick because not many packets are sent to the remote machine.
- There is little target host disturbance.
- There is a small but growing OS signature database.
- There is support for network devices like routers and switches.
- Matrix based fingerprint matching approach is utilized. This approach is also known as‘’fuzzy’’ matching.
- Xprobe2 comes with an API that allows users to write their own modules.

5.3. COUNTERMEASURES

5.3.1 Secure Networking Layers

Secure networking has traditionally relied on securing the *application* traffic as it traverses the network. There are three categories of applications security.

• **Perimeter security** protects the network applications from outside attack. Key capabilities include firewall, intrusion detection, and network address translation.

• **Application security** provides data confidentiality, integrity, and non-repudiation, typically through the use of SSL or IPSec.

• **Transport security** ensures that the information reaches appropriate users exclusively. Within the enterprise, VLANs typically provide the required separation. When multiple enterprises share a service provider network, the Provider-Provisioned Virtual Private Network (PP-VPN) provides an additional layer of separation.

Secure networking extends this by protecting the underlying *infrastructure* from attack. Management and signaling traffic must be secured from hackers, often using the same techniques discussed above. In addition, other measures are required to ensure that the infrastructure is secure.

• **Platform security** ensures that each device is available to perform its intended function. Devices include system, clients (e.g., PDAs, workstations), servers (e.g., applications, DHCP), and networking equipment (e.g., routers, Ethernet switches). Examples include logging of events.

• **Operational security** ensures that each user has access to only those network elements and applications required to perform his/her job. Examples include use of strong passwords, RADIUS authentication, and password lockouts.

• **Physical security** protects from physical harm or modification, and underlies all security practices. The most obvious forms of physical security include locked doors and alarm systems.

All these are to be properly implemented strategies for taking care of the system vulnerabilities including unauthorized access and discovery of classified information including the detection of Operating system.

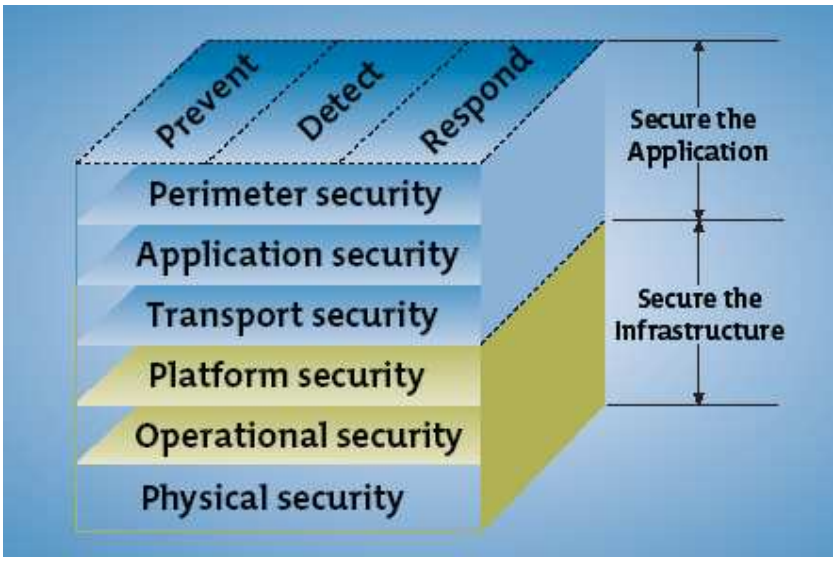


Figure 5.4: Secure Networking

5.3.2 Generalized Countermeasures

Just as there is a vast plethora of tools and techniques available for remote OS identification, there are also techniques and tools for OS obfuscation. Papers on the subject of defeating OS fingerprinting acknowledge that its use is limited5, not because it is technically infeasible or particularly performance limiting, but because the majority of exploit attacks are by 'script kiddies' who do no more than download the latest exploit script from a website and mass-scan the Internet looking for vulnerable hosts.

OS obfuscation tends to be more useful in deterring the more advanced hacker who is explicitly looking for specific OS types to attack. It is worth digressing here to reinforce the usual "obscurity is not security" rule. That is, you should ensure the system you are trying to protect is fully patched and secured. As discussed above, OS obfuscation is not something to be relied upon as a security measure. Nevertheless, steps that could be taken include:

Altering all public banners on services to something non-committal. This can range from the simple editing of files (eg issue, issue.net, motd on Unix) to the more difficult editing of application source code and recompilation (eg Apache Server type response). This is, of course, much more difficult if using proprietary software or software without open source code (try changing your IISServer type response).

Searching content files for 'incriminating' strings that can give away the OS. Web pages may include comments automatically generated that identify the authoring tool itself, which in turn is known to be commonly used for authoring to a particular platform.

Microsoft Frontpage is particularly bad for this. A tool may produce HTML code that allows inference to the server OS. Microsoft Office products produce characteristic HTML which is commonly uploaded to IIS webservers, again identifying a 'Microsoft shop'. Other indicators might include scripted webpages with .aspx URL suffixes indicating a Microsoft Windows 2000 onwards host, or adopting thethree letter .htm suffix, again a Microsoft convention.

Take stock of the number and types of services you are running on the machine. Do they resemble the target one is trying to impersonate? Can these applications be queried further using a Service Query tool to reveal the nature of the service itself, and hence infer the underlying OS? Follow standard security practice and remove any unwanted services. Tools exist to fake ones that should appear to be there if you wish to impersonate a different OS.

Changing the way your IP stack responds to probe queries. This is much easier to accomplish on platforms that allow direct manipulation of incoming packets through kernel mode OS hooks (code intervention points), or altered network card drivers. Tools are available for this purpose on some Unix/Linux platforms. Of note is the netfilter/iptables framework inside the 2.4.x Linux kernel. This is a firewall package, implemented as a set of OS hooks. These allow a user routine to be registered as a module and be called whenever a packet matching some criteria is encountered. Thus packets can be inspected and 'mangled' as they are received by the host. Described

the use of this technology in conjunction with the “IP Personality” netfilter module to make a Linux 2.4.19 based machine appear as a Sega Dreamcast console .

IP PERSONALITY

can fool current versions of nmap and Xprobe2, and is very configurable, so that it can probably fool any similar tool. The patch allows one to emulate the behaviour of any system listed in nmap's list of OS fingerprints. Some of its features can even be applied to routed traffic, and thus disturb scans directed to machines that are behind it. Some features (eg TCP ISN rewriting) can also be used to improve overall network security.The characteristics that can be changed are:

- 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

This method is basically Deceiving OS fingerprinting. Example FPF is a Loadable Kernel Module (LKM) for Linux that changes the TCP/IP stack in order to emulate other OS's TCP fingerprint. The package contains the lkm and a parser for the nmap file that let you choose directly the OS you want.This enables Linux machines to masquerade as other operating systems, thereby making it harder for attackers to launch OS-specific attacks.

Other kernel level solutions simply look to detect odd IP packets and silently drop them with or without logging them for further analysis. Note that these approaches work well with strict rule-based fingerprinting, eg Nmap, effective against the newer ‘fuzzy matching’ techniques of Xprobe 2

Insulating the host from probe packets via a firewall or packet ‘Scrubber’. Many firewall products now provide this functionality out of the box. Checkpoint’s Firewall-1 software provides a language that allows responses to be crafted to particular packets, thus actively spoofing the target OS. BlackICE PC Protection and Sygate’s Personal Firewall also proclaim OS fingerprint protection; however, this

appears to be centred around the more usual practice of dropping strange packets rather than dealing with them.

Insulating the host from probe packets by using Network Address Translation (NAT) technology. In this scenario the host's network is typically given a 'private' network designation, eg 10.0.0.0, or 192.168.0.0. An intelligent gateway accepts outgoing packets from hosts and transparently alters them to make it appear the gateway node is the source. Upon return the gateway transparently reinstates the original address and forwards the packet to the original host.

This effectively makes all traffic to or from the network appear to be coming from the one node, making identification of hosts on the private network very difficult. One thing to note through all of this is that your host is presumably on Internet for a reason, probably to provide a service. OS Fingerprint 'hardening' should not impair the function host is providing. This may not be evident in machine itself failing to function, but perhaps in customers not being able to access services reliably because its confusing their client software with strange host ID strings, or dropping packets from their 'slightly broken' IP stack.

Block unassigned port TRAFFIC. Traffic to ports with unassigned services should be blocked. As the tools need both open and closed ports to successfully run their test. Blocking traffic to all the un assigned closed ports we further help to prevent their attack

Application-layer monitoring Locate spyware, adware, worms and other types of malicious software on your network by seeing the activity these applications generate. Using Service monitoring tool to get hold of communication that's taking place, i.e. keep check on emails, ftp, telnet and other SNMP sevices taking place

Monitor transport-layer connections (control of TCP,UDP, SYN, RST, ACK)
FOR TCP and UDP PACKETS

1. Enforce correct usage of TCP flags 2. Limit per-source sessions 3. Enforce minimum TCP header length 4. Block unknown protocols 5. Restrict FIN packets with no ACK 6. Enforce that TCP header length as indicated in header is not longer than packet size indicated by header 7. Block out state packets 8.Verify that first connection packet is SYN 9.Enforce 3-way handshake: Between SYN andSYN-ACK, client can send only RST 10. Enforce 3-way handshake enforcement: Between SYN and connection establishment, server can send only SYN-ACK or RST	11.Block SYN on established connection before FIN or RST packet is encountered 12. Restrict server-to-client packets belonging to old connections 13. Drop server-to-client packets belonging to old connections if packets contain SYN or RST 14. Enforce minimum TCP header length 15. Block TCP fragments 16. Block SYN fragments 17. Verify TCP packet sequence number for packets belonging to an existing session 18. Verify UDP length field 19. Match UDP request and response

Table 5.6: Safeguarding Measures

Need to keep a constant check on type of traffic that's coming. MRTG is a tool to monitor the traffic load on network-links. MRTG generates HTML pages containing GIF images which provide a LIVE visual representation of this traffic

Monitor source address: Match the source address with well known addresses eg. IPACL stands for `IP access list'. It has been designed to filter incoming and outgoing TCP/UDP packets in the SVR4/386 kernel with Lachamnn Streams TCP. Depending on source/destination addresses and port numbers packets can be passed through or dropped. It easy to run IPACL on Interactive SVR4 and on SINIX-L (on MX300 (Intel) and other SVR4 bases systems, e.g. the new Solaris.

Filter ICMP type 3 and 8 TYPE THREE : (**ICMP_UNREACH)**Port unreachable error. When the designated transport protocol (e.g., UDP) is unable to demultiplex the datagram but has no protocol mechanism to inform the sender.

TYPE EIGHT: (**ICMP_ECHO**)Source host isolated error. Obsolete.Ping is probably the most well known use of ICMP. When you ping someone, you send out a Type 8

(Echo Request) ICMP packet, and if they reply, you receive a Type 0 (Echo Reply) ICMP packet. All but the most paranoid networks want to allow their users to ping the outside world .DENY Type 8 ICMP INCOMIMG and Type 0 ICMP OUTGOING.This does not necessarily mean that the outside world will not be able to ping you but it means they will not get response.

5.3.3 Specific to Nmap and Xporbe2

There are some **defensive** measures to protect against the OS detection method used by NMAP and Xprobe2. Among the defenses mentioned above:

a) 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 defense 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 accuracy drops off. This also implies that there should only be a few ports left open. Filtering all of the UDP ports, and not leaving any open will stop Xprobe2. This is a good defense since Xprobe2 needs at least one closed UDP port to work

b) Users can change characteristics of the machines TCP/IP stack. This can also be accomplished via kernel patches.

c) Network intrusion detection systems (IDS) can be used, if configured correctly, to detect the OS detection method used by Nmap because of the utilization of malformed packets and OS detection method used by Xprobe2 because of the series of tests it performs.

d) Block ICMP traffic at the firewall. Blocking both incoming and outgoing traffic.

5.4. COUNTERACTIVE MEASURES

The more paranoid system administrators have long sought methods to defeat, confuse, or prevent the cracker underworld, or even the curious hacker, from

successfully fingerprinting their hosts. Those methods will now be described, as well as methods that might be successful in defeating probes by common OS Fingerprinting tools

5.4. 1 Firewalling

Stateful Firewalls and correct Firewall rules can be used to block all TCP traffic for unknown states. This blocks all nmap scans except ones beginning with a valid TCP SYN segment. Firewall rules can also be used to block most ICMP traffic. While this may have some impact on operations or network diagnostics, it can be considered an acceptable loss.

IP Personality is a Linux netfillter module that allows one to configure a host's firewall ruleset to mangle TCP and IP packet parameters. For example, here are some of the possible parameters it can mangle:

_ 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

5.4.2 Kernel level parameters

Most unix-like operating systems provide an interface to change TCP/IP parameters used by the network stack. On BSD systems, this interface is accessed via the sysctl utility, and some of the applicable parameters (on FreeBSD 5.1)

are under net.inet.tcp.*:

- rfc1323 - high performance TCP extensions
- mssdt - default receiver MSS
- always keepalive - send TCP keep-alive segments
- keepintvl - interval between keep-alive segments
- sendspace - default initial TCP window size
- delayed ack - use recommended delayed TCP acks.

- slowstart ightsize - initial congestion window (cwnd)
- size
- local slowstart ightsize - local initial cwnd size
- newreno - enable TCP NewReno algorithms
- net.inet.ip.ttl - default IP TTL used

The Linux kernel also has changable values for other parameters under net.ipv4.*:

- tcp dsack
- tcp fack
- tcp _n timeout
- tcp synack retries
- tcp syn retries
- tcp retries2
- tcp retries1
- tcp keepalive intvl
- tcp keepalive probes
- tcp keepalive time
- tcp retrans collapse
- tcp sack
- tcp window scaling
- tcp timestamps

5.4.3 Scrubbing

While the above parameters can be used to adjust some of the time-related tests, packet scrubbing in a firewall, such as OpenBSD's pf, can be used to prevent overlapping IP fragments and possibly TCP segments from entering a host or network.

5.4 .4 Active Mapping

In the field of intrusion detection, work has shown how both the IP and TCP protocols are vulnerable to evasion from network intrusion detection systems (NIDS).

To counter this vulnerability, one can use a tool, called an Active Mapper, to assess how the hosts on a network resolve those ambiguities, and then feed that information into a NIDS so it knows the resolution policies of all the hosts on the network. Listed below are the checks the tool makes.

- Hop count to the host
- Path-MTU to the host
- IP fragment reassembly policy
- TCP segment reassembly policy
- TCP RESET acceptance policy

The two interesting checks are IP fragment reassembly and TCP segment reassembly policies. The research reported _nding multiple IP fragment reassembly policies (BSD, BSDright, linux, _rst and last), and also reported on observed TCP segment reassembly policies in the wild.

5.4.5 PROCESS Level Parameters

Ever wondered what it would be like to have custom IP stack readily programmable? Ever wanted to be able to use stock clients connecting to stock servers, but still be able to tweak the underlying connection? Have ever wished that one could poke at individual packet bits within a real connection without having to patch your kernel? **Packet Purgatory** is a library that allows user land programs to do all of the above portably.

Any applicational implementation based on Packet Purgatory library will runs as a process and is not a kernel module. Making it highly portable.

Packet Purgatory acts as a wedge between the host's interface and the host's kernel. Packet Purgatory is built on libpcap and libdnet. Libpcap and libdnet provides interfaces to the kernel. Packet Purgatory is a library providing a portable API to developers that provides the fine-grained control of raw sockets, while still being able to use helpful constructs like the TCP stack, and preexisting IP software.

There are really only two core functions that need calling in packet purgatory Known as packet purgatory dependencies. Initialization and Start & handler function. These two dependencies are customized according to tool that's is to be defeated in case of Os fingerprinting The main concept to packet purgatory is the idea of packet handlers. The start function takes two of them, an inbound and an outbound handler.

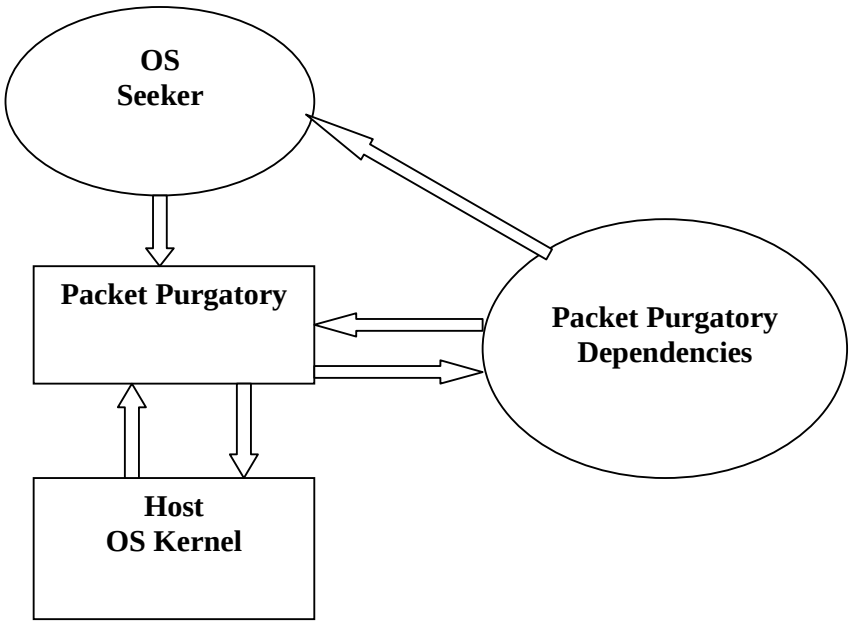


Figure 5.5: High-Level Packet Purgatory Architecture

The key variable in these functions is the variable "packet". This is a buffer that contains the contents of the packet. If the packet handler functions returns zero, whatever is in the buffer at that time will be reinjected to the network. So by modifying the buffer, it modify the packet that will be sent/received.

The inbound packet handler will be called for each packet heading out from the system running Packet Purgatory, before that packet reaches the network, and the inbound packet handler will be called for each packet inbound to the system, but before the system's kernel starts processing it. Either function can be defined as NULL, in which case packets will simply transit as they normally would, with no modifications being made. The State Table works when ever host sends packet by generating a "random" sequence number based on emulated OS. State table maintains

session sequence number offset information. Hence, Sequence number gets modified on the way to remote OS

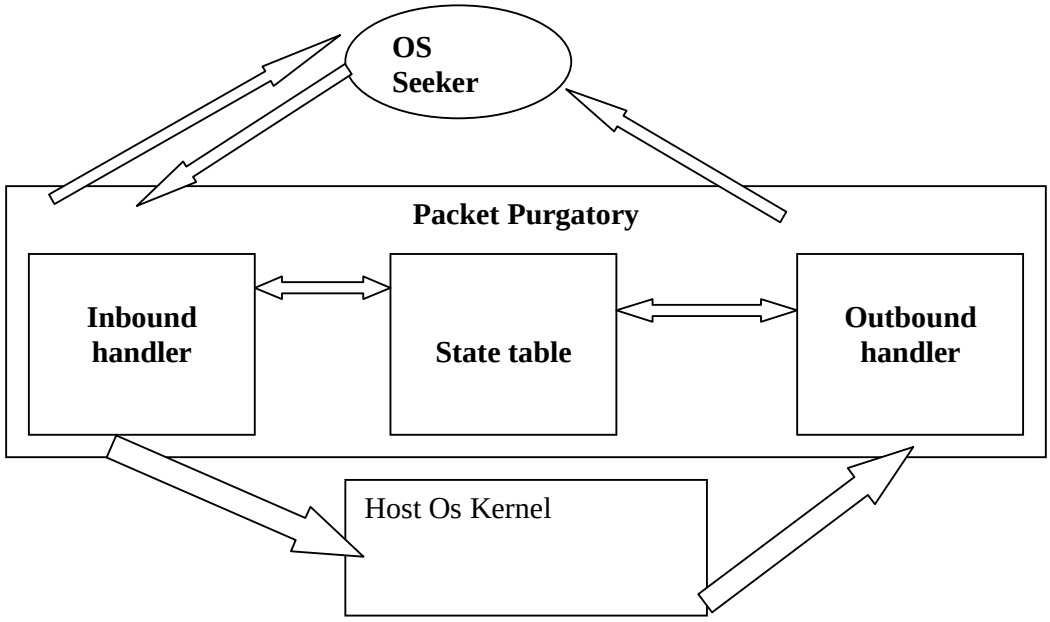


Figure5.6: Internal Architecture

By default, Packet Purgatory does some deep surgery in order to do what it does. It changes the route tables to point out the **loopback**, modifies firewall rules to block inbound traffic, and puts the interface into promiscuous mode. This can break in a number of situations:

- No built-in firewall exists on the system
- The built-in firewall is not supported by libdnet (IPTables)
- The loopback interface is not a full interface (Solaris)

If any of the above is true, then the sample program will not run successfully. All is not yet lost, however.

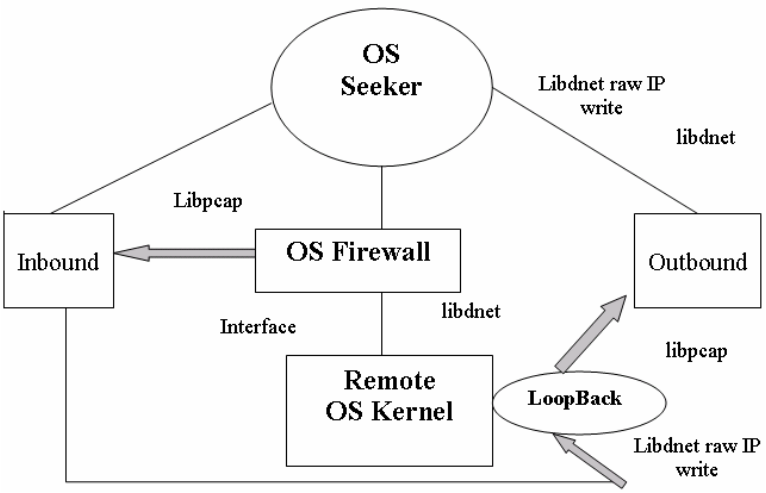


Figure 5.7: Loopback Mode

Packet Purgatory supports a second type of packet capture/redirection which is a little more complex in terms of programming, but should be portable across all systems. This type is called a **proxy IP,** or fake IP capture. Using proxy IP capture, an unused IP address on the local subnet is used to proxy communication. Packet Purgatory will handle ARPing for that IP address, so packets destined to it will be properly received by the network card. In order to enable proxy IP capture, a few changes must be made to the sample program.

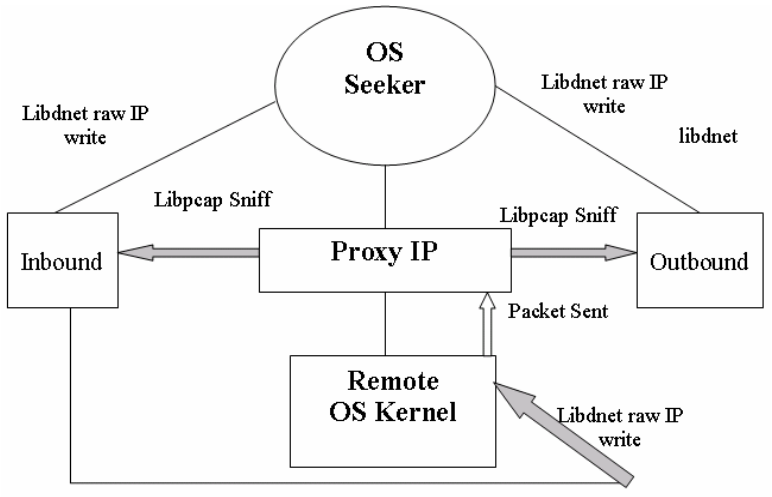


Figure 5.8: Proxy Mode

Hence using the techniques of “Packet Purgatory” a process that allows user to select desired OS emulate can be designed and implemented. It will handle inbound and outbound packets and change TCP, UDP, ICMP and IP headers to reflect selected OS.

Now, Usage prediction of Packet Purgatory dependencies for defeating Nmap:

		Handling Status for Nmap3.81		
Nmap Packet Types		**Inbound**	**State Table**	**Outbound**
Open Port	TCP Sequence Test	Pass packet to OS	Add SYN connection	Send response packet to
	SYN with Options	Pass packet to OS	Add SYN connection	reflect emulated OS
	NULL with Options	Respond on behalf of	Don't care	Send response packet to
	SYN-FIN-URG-PSH	emulated OS	Add connection	reflect emulated OS
	with Options	If OS accepts it, pass to OS.	If part of existing connection,	Don't care
Closed Port	ACK with Options	Otherwise, respond on	add ACK connection	If applicable, send response to
	SYN with Options	behalf of emulated OS	Don't care	reflect emulated OS
	ACK with Options	If connection exists, pass	Don't care	Send response packet to
	PSH-FIN-URG with	packet to OS. Otherwise,	Don't care	reflect emulated OS if part of

Table 5.7: Usage Prediction Nmap

Usage prediction of Packet Purgatory dependencies for defeating Xprobe2:

	Handling Status for Xprobe2		
Xprobe2 Packet Type	**Inbound**	**State Table**	**Outbound**
ICMP ECHO	Respond on behalf of emulated OS	Don't care	Don't care
ICMP Timestamp	Respond on behalf of emulated OS	Don't care	Don't care
ICMP Address Mask Request	Respond on behalf of emulated OS	Don't care	Don't care
ICMP Information Request	Respond on behalf of emulated OS	Don't care	Don't care
UDP -> ICMP Unreachable (Includes UDP Port Unreachable Error Message)	If port probed is open, pass to OS. Otherwise, respond on behalf of emulated OS	Don't care	Rewrite appropriate reply according to emulated OS
TCP SYN (Includes TCP Header Information)	If port is open pass packet to OS, else write RST as a response	Add SYN connection	Rewrite packet to reflect emulated OS

Table 5.8: Usage Prediction Xprobe2

5.5. IMPLEMENTATIONS EXAMPLES

Example for window NT and higher versions

Employing this TCP/IP Filters that will make automatic detection harder. The following registry keys should be modified (this can be done automatically by using the script below) to prevent Nmap from easily detecting the OS type:

Registry file:

Save the following into filter.reg, replacing 'CARD_NAME' with our network card name:

```
[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services
\Tcpip\Parameters]
"EnableSecurityFilters"=dword:00000001
[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\<CARD_NAM
E>\
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)
[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services
\Parameters\Tcpip]
"TCPAllowedPorts"=hex(7):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)
----filter.reg end
```

Explanation:

The registry script above disables communication on all ports beside:

27,38,30 ==80 ==http

35,32,30==520 ==rip

Also, it allows only the following protocols:

36 ==6 ==tcp

31,37 ==17 ==udp

This disables remote connections to any port except those specified above. This will prevent OS detection programs from gaining enough information to successfully detect the machine as a Windows NT and the higher versions.

Before Filters:

$ **nmap -v -sT -O 172.31.5.103**

Starting nmap 3.81 (http://www.insecure.org/nmap) at 2005-04-18 10:49 India Standard Time

Host (172.31.5.103) appears to be up ... good.

Initiating TCP connect() scan against (172.31.5.103)

Adding TCP port 139 (state Open).

Adding TCP port 135 (state Open).

The TCP connect scan took 44 seconds to scan 1517 ports.

For OSScan assuming that port 135 is open and port 422 is closed and neither are firewalled

For OSScan assuming that port 135 is open and port 422 is closed and neither are firewalled

For OSScan assuming that port 135 is open and port 422 is closed and neither are firewalled

WARNING: OS didn't match until the 3 try

Interesting ports on (172.31.5.103):

Port	State	Service
135/tcp	open	loc-srv
139/tcp	open	netbios-ssn

TCP Sequence Prediction: Class=random positive increments

Difficulty=10214 (Worthy challenge)

Sequence numbers: 82159285 821671B0 8216F2F9 8217B423 82187D56 82197619

Remote operating system guess: Microsoft NT 4.0 Server SP5 + 2047 Hotfixes

Nmap run completed -- 1 IP address (1 host up) scanned in 69 seconds

After applying the filters:

$ **nmap -v -sU -O 172.5.31.103**

Starting nmap 3.81 (http://www.insecure.org/nmap) at 2005-04-18 12:49 India Standard Time

Host (172.31.5.103) appears to be up ... good.

Initiating FIN,NULL, UDP, or Xmas stealth scan against (172.31.5.103)

The UDP or stealth FIN/NULL/XMAS scan took 0 seconds to scan 1 ports.

Interesting ports on (172.31.5.103):

Port State Service

520/udp open route

Too many fingerprints match this host for me to give an accurate OS guess

TCP/IP fingerprint:

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

T5(Resp=N)

T6(Resp=N)

T7(Resp=N)

T7(Resp=Y%DF=N%W=0%ACK=S++%Flags=AR%Ops=)

PU(Resp=N)

Nmap run completed -- 1 IP address (1 host up) scanned in 18 seconds

FOR OPENBSD

Although port 0 is a valid TCP / UDP port number, it is highly recommend that one should block any traffic using this port at your firewall. No program should be listening on port 0 and no program should connect from port 0 thus, it should be blocked. Port 0 fingerprinting can be tested usingboth nmap and xprope and this is highly sensetuve port. As port 0 is reserved for special use as stated in RFC 1700. Coupled with the fact that this port number is reassigned by the OS, no traffic should flow over the internet use this port. As the specifics are not clear different OS's have, different ways of handling traffic using port 0 thus they can be fingerprinted.

Firewall Configurations

Untested IPTables Rules for port 0 fingerprint blocking	OpenBSD's Packet Filter Rules for port 0 fingerprint blocking
$IPTABLES -A DROP -p tcp --dport 0 $IPTABLES -A DROP -p udp --dport 0	block in log quick on $EXT inet proto tcp from any port 0 to any block in log quick on $EXT inet proto udp from any port 0 to any

$IPTABLES -A DROP -p tcp --sport 0 $IPTABLES -A DROP -p udp --sport 0	block in log quick on $EXT inet proto tcp from any to any port 0 block in log quick on $EXT inet proto udp from any to any port 0

Table 5.9: Port0 Fingerprinting Ruleset

Also, Falsify your BSD OS fingerprint *block all OS guessing by setting options TCP_DROP_SYNFIN in my kernel.*

EXAMPLE FOR SOLARIS

Solaris, what Solaris 2.x would respond, a determining factor seems to be the window field, which happens to be a modifiable parameter in Solaris's TCP/IP stack. This field help us to differentiate between- Firewalled Solaris 2.x and Solaris 2.x for Solaris detection two factor that are most widely used are

1)**TCP_MSS_DEF** This parameter determines the default MSS (maximum segment size) for non-local destinations. The MSS is the largest chunk of data that TCP will send to the receiving end. Whenever a connection is about to be established, the segment size used will be set to the minimum of the smallest MTU of an outgoing interface, and from the MSS announced by the peer. Each end announces it's MSS. Nmap and Xprobe2 expects to see a window of 8760 (2238 in hex) or 9112 (2398 in hex) and uses this determine the OS.

2) **ip_path_mtu_discovery** If path MTU discovery is active, all outgoing PDUs have the IP option DF (don't fragment) set. The route a PDU travels along a TCP virtual circuit may change with time. For this reason RFC 1191 recommends to rediscover the path MTU of an active connection after 10 minutes. Solaris aggressively tries to rediscover the path MTU every 30 seconds, and is apparently one of the only operating systems to do so. Another alternative to not setting path MTU off would be to increase the delay of the ip_ire_pathmtu_interval parameter to 600000.

Hence, to mask the detection in case of Solaris 2.x following is the way to cahe\nge the above parameters.**TCP_MSS_DEF** value can be set higher that 536, without any serious side effects. You can add this to the secureip script from a previous module.

```
ndd -set /dev/tcp tcp_mss_def 546
echo "setting default MSS to 546"
```

This parameter defines the interval Solaris rediscovers the path MTU.

```
ndd -set /dev/ip ip_ire_pathmtu_interval 600000
echo "setting ip_ire_pathmtu_interval to 600000."
```

This parameter switches path MTU discovery on or off. If you enter a 0 here, Solaris will never try to set the DF bit in the IP option - unless your application explicitly requests it.

```
ndd -set /dev/ip ip_path_mtu_discovery 0
echo "setting ip_path_mtu_discovery off."
```

Example for linux:

For linux, the response is different, but the solution the same. (Linux 2.0.35 to 2.0.9999 :)

One could easily get the ip filter package and setup the many types of rules block packetboth incoming and outgoing

IPtable Ruleset

```
filter
:INPUT DROP [0:0]
:FORWARD DROP [0:0]
:OUTPUT DROP [0:0]
-A INPUT -i eth0 -f -j DROP
-A INPUT -m state --state INVALID -j DROP
-A INPUT -s 127.0.0.0/255.0.0.0 -i ! lo -j DROP
-A INPUT -i lo -j ACCEPT
[test rulesets will go here]
-A INPUT -p tcp -m tcp --dport 22 --tcp-flags SYN,RST,ACK SYN -j ACCEPT
-A INPUT -p tcp -m tcp --dport 25 --tcp-flags SYN,RST,ACK SYN -j ACCEPT
-A INPUT -p tcp -m tcp --dport 80 --tcp-flags SYN,RST,ACK SYN -j ACCEPT
-A INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT
-A INPUT -m state --state NEW -j DROP
-A OUTPUT -o lo -j ACCEPT
-A OUTPUT -p tcp -m state --state NEW,ESTABLISHED -j ACCEPT
-A OUTPUT -p udp -m state --state NEW,ESTABLISHED -j ACCEPT
-A OUTPUT -p icmp -m state --state NEW,RELATED,ESTABLISHED -j ACCEPT
COMMIT
```

Xprobe2 Successful

```
% xprobe2 -c /usr/local/etc/xprobe2.conf 172.31.5.23
XProbe2 v.0.1 Copyright (c) 2002 fygrave@tigerteam.net, ofir@sys-security.com
[+] Target is 172.31.5.23
[+] Loading modules.
[+] Following modules are loaded:
[x]ICMP echo (ping)
[x]TTL distance
[x]ICMP echo
[x]ICMP Timestamp
[x]ICMP Address
[x]ICMP Info Request
[x]ICMP port unreach
[+] 7 modules registered
[+] Initializing scan engine
[+] Running scan engine
[+] Host: 130.126.112.183 is up (Guess probability: 50%)
[+] Target: 130.126.112.183 is alive
[+] Primary guess:
[+] Host 130.126.112.183 Running OS: "Linux Kernel 2.4.5 and above" (Guess probability: 45%)
[+] Other guesses:
[+] Host 130.126.112.183 Running OS: "Linux Kernel 2.2.x" (Guess probability: 45%)
[+] Host 130.126.112.183 Running OS: "Microsoft Windows NT 4 Service Pack 4 and Above" (Guess probability: 45%)
[+] Host 130.126.112.183 Running OS: "OS X 10.1.5" (Guess probability: 40%)
[+] Host 130.126.112.183 Running OS: "Microsoft Windows 98/98SE" (Guess probability: 25%)
[+] Cleaning up scan engine
[+] Modules deinitialized
[+] Execution completed.
```

Xprobe2 Countermeasures test rule set:

Under the test conditions mentioned previously Xprobe2 abort if no echo-reply is received from the target in response to an echo-request.If an echo-reply is received.Xprobe2 guesses Linux kernel 2.4.5 and up with 45% probability. ping-ability" with this ruleset during test:

-A INPUT -p icmp –icmp-type echo-request -j ACCEPT

Xprobe2 Unsuccessful

```
% xprobe2 -c /usr/local/etc/xprobe2.conf 172.31.5.23
XProbe2 v.0.1 Copyright (c) 2002 fygrave@tigerteam.net, ofir@sys-security.com
[+] Target is 172.31.5.23
[+] Loading modules.
[+] Following modules are loaded:
[x]ICMP echo (ping)
[x]TTL distance
[x]ICMP echo
[x]ICMP Timestamp
[x]ICMP Address
[x]ICMP Info Request
[x]ICMP port unreach
[+] 7 modules registered
[+] Initializing scan engine
[+] Running scan engine
[+] Host: 130.126.112.183 is down (Guess probability: 0%)
[+] Cleaning up scan engine
[+] Modules deinitialized
[+] Execution completed.
```

Nmap Successful

```
% nmap -O -P0 -p1-100 -sS 172.31.5.23
Starting nmap V. 3.00 ( www.insecure.org/nmap/ )
sendto in send_tcp_raw: sendto(3, packet, 60, 0, 130.126.112.183, 16)
=> Operation not permitted
sendto in send_tcp_raw: sendto(3, packet, 60, 0, 130.126.112.183, 16)
=> Operation not permitted
Interesting ports on 172.31.5.23 (130.126.112.183):
(The 97 ports scanned but not shown below are in state: filtered)
Port State Service
22/tcp open ssh
25/tcp open smtp
80/tcp closed http
Remote operating system guess: Linux Kernel 2.4.0 - 2.5.20
Uptime 5.326 days (since Tue Jan 7 08:15:25 2003)
Nmap run completed -- 1 IP address (1 host up) scanned in 18 seconds
```

Iptables Rulesets for Nmap

1 Stop nmap XMAS scans

-A INPUT -p tcp -m tcp --tcp-flags FIN,SYN,RST,PSH,ACK,URG -j DROP

2 Stop nmap NULL scans

-A INPUT -p tcp -m tcp --tcp-flags FIN,SYN,RST,PSH,ACK,URG NONE -j DROP

3 Stop nmap TCP connect() and SYN scans

-A INPUT -p tcp -m tcp --tcp-flags FIN,SYN FIN,SYN -j DROP

4 Stop nmap FIN scans

-A INPUT -p tcp -m tcp --tcp-flags FIN,ACK FIN -j DROP

5 And add these in for good measure

-A INPUT -p tcp -m tcp --tcp-flags PSH,ACK PSH -j DROP

-A INPUT -p tpc -m tcp --tcp-flags URG,ACK URG -j DROP

Nmap Unsuccessful

```
 nmap -O -P0 -p1-100 -sS 172.31.5.23
Starting nmap V. 3.00 ( www.insecure.org/nmap/ )
sendto in send_tcp_raw: sendto(3, packet, 60, 0, 130.126.112.183, 16)
=> Operation not permitted
[ NOTE: some output removed here ]
Interesting ports on prospero.cso.uiuc.edu (130.126.112.183):
(The 97 ports scanned but not shown below are in state: filtered)
Port State Service
22/tcp open ssh
25/tcp open smtp
80/tcp closed http
No exact OS matches for host (If you know what OS is running on it,
see http://www.insecure.org/cgi-bin/nmap-submit.cgi).
TCP/IP fingerprint:
SInfo(V=3.00%P=i686-pc-linux-gnu%D=1/12%Time=3E21E095%O=22%C=80)
TSeq(Class=RI%gcd=1%SI=33F6EB%IPID=Z%TS=100HZ)
TSeq(Class=RI%gcd=1%SI=33F70A%IPID=Z%TS=100HZ)
TSeq(Class=RI%gcd=1%SI=33F6E1%IPID=Z%TS=100HZ)
T1(Resp=Y%DF=Y%W=16A0%ACK=S++%Flags=AS%Ops=MNNTNW)
T2(Resp=N)
T3(Resp=N)
T4(Resp=N)
T5(Resp=Y%DF=Y%W=0%ACK=S++%Flags=AR%Ops=)
T6(Resp=N)
T7(Resp=N)
PU(Resp=N)

Nmap run completed -- 1 IP address (1 host up) scanned in 63 seconds
```

Hence we see that the basically targeted protocol for fingerprinting are TCP,UDP and ICMP. The parameters of offset of the packets being send and received are vendor (OS) specific and the similarity and difference between these parameters help the tools to identify our operating system easily

The basic need is to stop unknown packets from unknown source that are targeted to scan our system –that is to stop the SCAN. The next step is putting in and changing the **default parameters** of the OS that does to hamper its proper working like changing IP Personality, the MSS size, and applying TCP/IP filters where ever possible.

CHAPTER No: 6
CONCLUSION AND FUTURE SCOPE

CONCLUSION

The network offers us both the mode of identifying and hiding the fingerprints of remote terminals .Like human fingerprint that define the identity of human has we can extract them and match them with the database maintained .So are the remote computer attached to the network with identified by its fingerprint are unique to each computer consisting its recourses values like operating system, MAC address, revealed by different header extracted during *network analysis techniques.*

Active and passive OS fingerprinting differ significantly in approach, and there are variety of techniques and tools that employ one or both of these styles. These techniques trade off varying levels of accuracy, stealth, speed, and resistance to tactics used to fool or mislead OS fingerprinting.

Banner grabbing is an easy, effective technique and administrators should remove or modify service banners to reveal as little information as possible. TCP/IP, ICMP analysis presents a larger set of issues.

Remote OS Fingerprinting is a recent development on the Internet and one to watch. The ability to remotely determine, with high accuracy, the Operating System of a remote host on the Internet is a powerful one. The implications of this technology are perhaps not yet fully understood; however, it is seen as enough of a threat that strategies are currently being developed to prevent and spoof OS Fingerprints.

The most relevant concept to remember is the old adage “Obscurity isnot Security”. The ease with which exploit tools can be scripted and used *enmasse* to find vulnerable hosts largely trivialises the benefits of OS obscurity in today’s world. This may change over the coming years as the larger software companies put an emphasis on

network security and more specialized attacks are required to exploit systems. The general trend towards increasing penalties for getting caught as the world's cyber laws improve may also serve as a driver towards more refined attacks in the future. The first developments have already occurred in this area, with OS fingerprinting worm toolkits being developed to refine attacks.

Malformed packets sent during active fingerprinting can be filtered by a firewall or responded to with "smoke screens". Manually changing IP Personality settings is the strongest defense for both active and passive fingerprinting, but it carries substantial negative consequences. Using a patch or OS option is usually the better choice. Of course, the mind of a trained, skilled administrator is ultimately the best single tool for OS fingerprinting.

Though there are few implication both at kernel and process level to stop the detection of the OS fingerprints of remote system. But implementing one is not the rightful solution.

The need is answered by carefully examining the vulnerabilities of the system OS as per vendor specified. Taking steps in chronological order defending each possible method of attack measure should be taken to defeat TCP/IP stack fingerprinting and ICMP pattern sampling.

FURTHUR DEVELPOMENT

Fingerprinting tools continue to improve, as do both analysis in continue and necessary way or defences against them. A current focus of software development houses is one of computer security, with Microsoft launching its "Trustworthy Computing Initiative" and many OS vendors initiating an automated patch download/update service. Examples include Microsoft's Windows Automatic Update service included in Windows 2000 and onwards, and the Redhat Network service available via the up2date utility in Redhat Linux. These developments, coupled with the general improvement in the world's cyber laws and prosecution rates, are slowly

'raising the bar' on cyber attacks. In this climate, general 'script kiddy' mass-scans may prove too dangerous or fruitless to pursue.

One emerging technology is fingerprinting tools that are themselves automated as part of OS refined attack tools. The first products in this area are already in existence. It represents the current step of evolution in an OS fingerprinting vulnerability scanner that tests for over 200 exploits and is designed to be incorporated into the design of an Internet worm. The flexibility of Sscan is demonstrated in the Sscan-readme file, which states *"you can have sscan launch off programs, initiate tcp connections and have dialogs with hosts that fit certain criterion, negotiate telnet connections and have sscan log into a host and execute shell commands (useful in coding internet worms), etc, all via a simple built in scripting language defined below."*

The attacks of the future may be well directed and customized according to OS and services running on the target. This may be considered normal worm activity in the future.

As per the technique of OS Fingerprinting is concerned there is shift in the scenario. Instead of TCP/IP stack fingerprinting or use of ICMP packets the TEMPORAL RESPONSE ANLYSIS of TCP packets is under study to increase the predictability of the Operating system while OS fingerprinting attack.

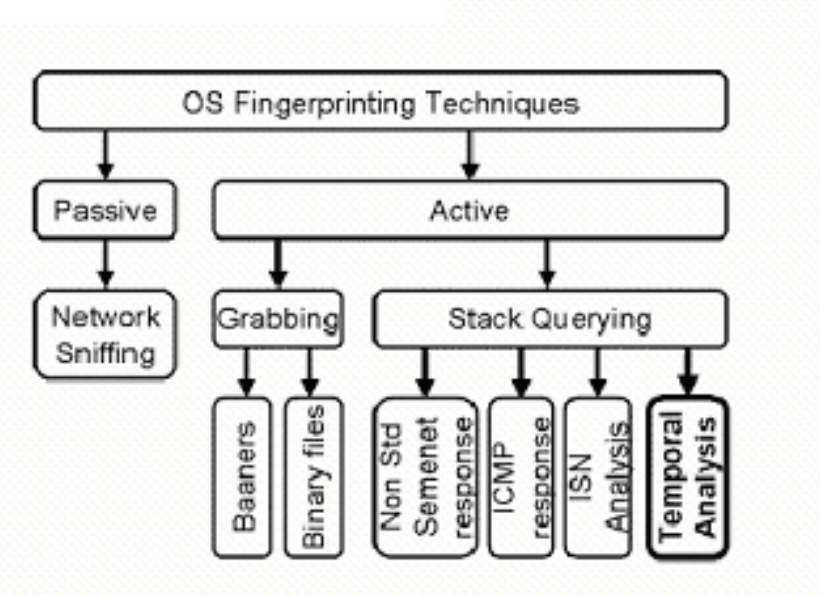


Figure 6.1: Synoptic of OS fingerprinting now including Temporal analysis

Packet retransmission offers a pattern that can be analyzed from a remote host. Then Temporal reponse based tools performs the following internal and network actions:

- Source port choice.
- Using libdnet, set up a local filtering function for blocking every incoming packet from thetarget machine.
- Using libpcap (pcap descriptor opening), start the packet listening using the filter defined above.
- Using libnet, send a TCP SYN packet to the target machine.
- Listen to the responses for a default or user adjusted delay.
- Compare the obtained responses to the known signatures.

REFERENCE

[1] Arkin, Ofir. “ICMP Usage in Scanning – The Complete Know How.” June 2001. URL: http://www.sys-security.com/archive/papers/may_addition/ ICMP_Scanning_v3.0.pdf , May 2, 2003.

[2] Arkin, and Yarochkin. “Xprobe v2.0: A “Fuzzy” Approach to Remote Active Operating SystemFingerprinting.” August 2, 2002. URL: http://www.sys-seurity.com/archive/papers/Xprobe2.pdf, May 7, 2003.

[3] Beck, Rob. “Passive-Aggressive Resistance: OS Fingerprinting Evasion” URL: http://www.linuxjournal.com/article.php?sid=4750, July 2003.

[4] CERT coordination center, A Taxonomy of Computer and Network Attacks URL: http://www.cert.org/research/JHThesis/Chapter6.html, Feb2003

[5] David Barroso Berrueta, “A practical approach for defeating Nmap OS-Fingerprinting”, URL: http://voodoo.somoslopeor.com/papers/nmap.html, November 2002.

[6] Dethy, “Examining port scan methods – Analysing Audible Techniques.” URL: http://www.synnergy.net/downloads/papers/portscan.txt,May, 2003.

[7] Ethereal: A Network Protocol Analyzer. URL: http://www.ethereal.com/, last accessed April 2004.

[8] Fyodor, “Remote OS Detection via TCP/IP Stack Fingerprinting”, June 11, 2002, http://www.insecure.org/nmap/nmap-fingerprinting-article.html .

[9] Fyodor Yarochkin and Ofir Arkin, “Xprobe2 - A'Fuzzy' Approach to Remote Active Operating System Fingerprinting” http://www.sys-security.com/archive/papers/Xprobe2.pdf .

[10] Fingerprinting: The Complete Documentation, http://www.l0t3k.org/security/docs/fingerprinting/

[11] Gael Roualland and Jean-Marc Saffroy, “IP Personality”, URL: http://ippersonality.sourceforge.net/

[12] Honeynet Project, “Know Your Enemy: Passive Fingerprinting, Identifying remote hosts, without them knowing”, URL: http://project.honeynet., March,2002

[13] Insecure.org. “Nmap Man Page.” URL: http://www.insecure.org/nmap /data/nmap_manpage.html (May 2, 2003).

[14] Introduction to Network Security, URL: http://www.interhack.net/pubs/network-security/

[15] James Hurnall, “Nmap tutorial”, URL:http://members.dodo.net.au/~ps2man /Nmap/nmap.html

[16] Kathy Wang, “Frustrating OS Fingerprinting with Morph”, Syn Ack Labs, DEFCON 12 www.synacklabs.net, Nov 2004.

[17] Know Your enemy, The Honeynet Project by Addison Wesley,2002

[18] Mark Wolfgang, “Host Discovery with nmap”, Nov 2002

[19] Matthew Smart G. Robert Malan Farnam Jahanian, "Defeating TCP/IP Stack Fingerprinting”, 9th USENIX Security Symposium Paper Pp. 229–240,JULY 2000

[20] McGraw-Hill. Let’s Talk: Computer Networks TechCONNECT Online: Technology Foundations,2002

[21] Mick Bauer,"Checking Your Work with Scanners, Part I (of II): nmap",Linux Journal archive Volume 2001 ,Issue 85es,article No. 13, www.acm.org/nmap_docs\MAINthibng.htm , May 2001

[22] Marco de Vivo,Eddy Carrasco, “A Review of Port Scanning Techniques”, ACM SIGCOMM Computer Communication Review ,http://portal.acm.org/citation.cfm?id=505733.505737,April 1999.

[23] Ste Jones, “Port 0 OS Fingerprinting” , URL: www.Networkpenetration.com 2003 .

[24] NMAP http://www.insecure.org/nmap/index.html

[25] Netfilter and IPTables. Available http://www.iptables.org/ (retrieved 25 Mar 2005).

[26] Ola Lundqvist, “Remote Analysis of Computer Systems A survey of software and techniques”Version 1.1, March 2000

[27] Ofir Arkin, Fyodor Yarochkin, Kydyraliev , “The Present and Future of Xprobe2-The Next Generation of Active Operating System Fingerprinting”, July,2003.

[28] Packet Purgatory Tutorial Outlining part 1& part 2, Syn Ack Labs, http://www.synacklabs.net/OOB/packetp_guide_1.html, Nov 2003

[29] Paul A. Watson, “SLIPPING IN THE WINDOW: TCP RESET ATTACKS” Technical Whitepaper,www.terrorist.net ,October , 2003.

[30] RFC describing TCP Extensions for High Performance with two of the three TCP options used in OS detection: http://www.faqs.org/rfcs/rfc1323.html

[31] RFC 793 “Transmission Control Protocol”, USC, University of Southern California, http://www.faqs.org/rfcs/rfc793.html, 1981.

[32] Richard Stevens, TCP/IP Illustrated, Volume 1, Addison-Wesley.

[33] Rich Jankowski , “Scanning and Defending Networks with Nmap” Source: Linuxsecurity.com, URL: http://www.linuxsecurity.com/content/view/117/49/, June 2002.

[34] Robert Beverly, “A Robust Classifier for Passive TCP/IP Fingerprinting”, March 2004

[35] Ryan Spangler, “Analysis of Remote Active Operating System Fingerprinting Tools”, Packetwatch Research, May 2003.

[36] See Everything With Ethereal URL: http://www.diversedev.com/misc /bcomm/manual.doc

[37] Seong Soo Kim, A. L. Narasimha Reddy and Marina Vannucci, "Detecting traffic anomalies through aggregate analysis of packet header data", in Proceedings of Networking 2004, Lecture Notes in Computer Science (LNCS) vol. 3042, pp. 1047-1059, Athens, Greece, May 2004.

[38] Thomas Glaser, Intrusion Detection FAQ, “TCP/IP Stack Fingerprinting Principles” URL:http://www.sans.org/resources/idfaq/index.php,Oct,2002.

[39] Veysset, Courtay, and Heen. “New Tool And Technique For Remote Operating System Fingerprinting.” URL: http://www.intranode.com/fr/doc/ring-full-paper.pdf , aPRIL,2002.

[40] Xprobe2 URL: http://sourceforge.net/projects/xprobe.

[41] Zalewski, Michal; Strange Attractors and TCP/IP Sequence Number Analysis - One Year Later, URL: http://lcamtuf.coredump.cx/newtcp/,2002.

LIST OF PAPER PUBLISHED

[1] Surbhie Kalia, Maninder Singh, "Masking approach to secure system from Operating System Fingerprinting.", in the IEEE Tencon05, Nov 2005(communicated)