AN APPROACH TO THE SECURITY PROBLEMS IN THE TCP/IP PROTOCOL SUITE FOR A NETWORK SECURITY MONITOR DESIGN

By

Pars MUTAF

A Dissertation Submitted to the
Graduate School in Partial Fulfillment of the
Requirements for the Degree of

MASTER OF SCIENCE

Department: Computer Engineering
Major: Computer Software

İzmir Institute of Technology
İzmir, Turkey

June, 1999

We approve the thesis of **Pars MUTAF**

Date of Signature

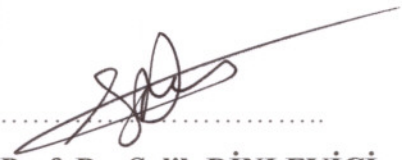
Asst. Prof. Dr. Ahmet KOLTUKSUZ

Supervisor

Department of Computer Engineering 29 / 06 / 1999

Prof. Dr. Sıtkı AYTAÇ

Department of Computer Engineering 29 / 06 / 1999



Asst. Prof. Dr. Salih DİNLEYİCİ

Department of Electrical and Electronics Engineering 29 / 06 / 1999

Prof. Dr. Sıtkı AYTAÇ

Head of Department 29 / 06 / 1999

ACKNOWLEDGEMENT

First, I would like to thank to my advisor *Asst. Prof. Ahmet KOLTUKSUZ* for encouraging me to write this thesis and for his endurance.

Thanks also to ALL my friends and my family.

Particulièrement, j'adresse mes plus vifs remerciements à *Madame Elisabeth SAYIN* pour sa considération et son support invaluable.

ABSTRACT

There are a number of security problems in the TCP/IP protocol suite. In this thesis these problems will be analyzed in detail. The problems in several existing prevention methods will be analyzed as well in order to show that security policies based merely on preventive measures are not completely secure and convenient. Therefore, "network security monitoring" will be proposed as an alternative and supplementary approach against Internet attacks.

ÖZ

TCP/IP protokol dizisinde önemli güvenli sorunları bulunmaktadır. Bu tezde bu sorunlar ayrıntılı olarak incelenecektir. Sadece önlemeci yaklaşımlara dayanan güvenlik politikalarının tamamen güvenli ve uygun olmadığını göstermek amacıyla, varolan bazı önleme yöntemleri de incelenektir. Bu yüzden, Internet saldırılarına karşı alternatif ve ek bir yaklaşım olarak "ağ güvenlik denetimi" önerilecektir.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

If we seek solace in the prisons of the distant past
Security in human systems we're told will always always last.

Emotions are the sail and blind faith is the mast
Without the breath of real freedom we're getting nowhere fast.

Sting

"History will teach us nothing"
...nothing like the sun

Chapter 1

INTRODUCTION

1.1 Purpose

The Internet has undergone a phenomenal growth in the recent past. Many organizations including businesses, agencies and universities are connected or in the process of connecting to the Internet. However, during this period the security problems in the TCP/IP protocol suite have been subjected to significant revelation as well, therefore security needs to be a major consideration when planning Internet connection.

An organization's posture toward security is referred to as a "security policy". This determines the limits of acceptable user behaviour and what the response to violations should be. Different organizations may have different security policies. An academic department of a university has generally different needs than a business, which, in turn, differs from a military site[1].

The first step in chosing a right security policy is "risk assesment". This involves the following[2]:

1. Identifying the assests
2. Identifying the threats

This thesis mainly focuses on identifying the important threats found in today's Internet, which will be categorized as: unauthorized access, denial of service and disclosure of information. Most of these threats comes from the vulnerabilities in the protocols. However, several important problems in the existing prevention techniques will be also analyzed.

This thesis also aims to provide background material for a future work on network security monitor design. Therefore several inportant points that must be considered in network security monitoring will be discussed as well.

1.2 Scope

This work is concerned with the security problems in the TCP/IP protocol suite. These are generic problems found in the definitions of the protocols rather than possible implementation specific flaws. Vendor-specific protocols are excluded as well.

Cryptography is beyond the scope of this work. Although suitable cryptosystems can solve most of the security problems which will be described throughout this thesis, currently they are not as convenient as they should be and it seems certain that they will not become de facto standards in the near future.

---

[1] W. R. Cheswick, S. M. Bellovin, *Firewalls and Internet Security: Repelling the Wily Hacker*, Professional Computing Series, Addison Wesley, p. 4, 1994.

[2] P. Holbrook, J. Reynolds (editors), "Site Security Handbook", RFC 1244, p. 11, 1991.

1.3 Organization

The rest of this work is organized as follows:

- Chapter 2 describes background material such as the TCP/IP internals and most common protocols.
- Chapters 3-6 presents the details of major protocols and systems from a security perspective as well as their flaws and the possible attacks they can be exposed to.
- Chapter 7 describes the common information disclosure points.
- Chapter 8 discusses the most common problems in the existing prevention techniques.
- Chapter 9 discusses the importance of network security monitoring and the important points which must be considered in designing a network security monitor.
- Chapter 10 presents a case study.
- Chapter 11 presents several conclusions.

Chapter 2

OVERVIEW OF TCP/IP INTERNALS

2.1 The Internet

The Internet is a worldwide network that uses the TCP/IP (Transmission Control Protocol/Internet Protocol) protocol suite for communications. The Internet was created initially to help communication among U.S. government sponsored researchers. Through the 1980's, the Internet grew steadily to include educational institutions, government agencies, commercial organisations, and international organisations. In the 1990's, the Internet has undergone a phenomenal growth, with connections increasing faster, many millions of users are now connected to the Internet[1].

2.1.1 Internet Hosts

The TCP/IP protocol suite allows computers of all sizes, from many different vendors, running different operating systems, to communicate with each other. A computer engaged in Internet communications is called a "host". A host generally executes application programs on behalf of user(s), employing network and/or Internet communication services in support of this function.

Many Internet hosts run a version of the Unix operating system. TCP/IP was first implemented in the early 1980's for the version of Unix written at the University of California at Berkeley known as Berkeley Software Distribution (BSD). Although Unix is the predominant Internet host operating system, many other types of operating systems and computers are connected to the Internet, including mainframe operating systems and personal computers running DOS, Microsoft Windows and Apple. Although personal computers often provide only client services, increasingly powerful personal computers are also beginning to provide, at low cost, the same services as larger hosts. Versions of Unix for the personal computer, including Linux, FreeBSD and BSDi, and other operating systems such as Microsoft NT, can provide the same services and applications that were, until recently, found on larger systems[2].

2.1.2 Internet is a Network of Networks

Part of the popularity of the TCP/IP protocol suite is due to its ability to interconnect different types of packet networks with different communication media and different protocols. Each host is directly connected to a particular network and these networks are interconnected by packet switching computers called "routers" or "gateways". The routers are capable of providing connections to many different types of physical networks such as Ethernet, token ring, point-to-point links, FDDI (Fiber Distributed Data Interface), etc.

---

[1] John P. Wack, Lisa J. Carnahan, "Keeping Your Site Comfortably Secure: An Introduction to Firewalls", NIST Special Publication 800-10, p. ix, 1994.

[2] ibid, p. 3.

2.2 TCP/IP Layering

The TCP/IP is the combination of many protocols. Although the commonly used name for entire protocol suite is TCP/IP, TCP and IP are only two of the protocols.

TCP/IP protocols operate at various layers. TCP/IP is normally considered to be a 4-layer system as shown in Figure 2.1.

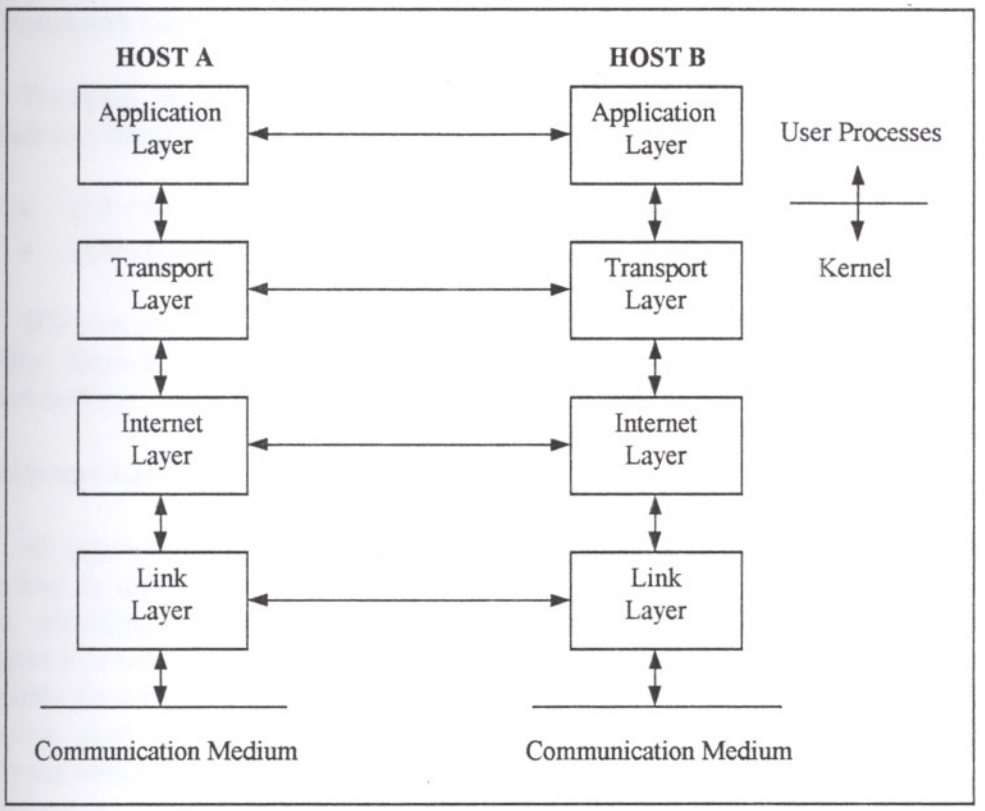


Figure 2.1 TCP/IP Layers.

Normally (in many implementations) the application layer is a user process while the lower three layers are usually implemented in the kernel (the operating system). Each layer has one or more protocols for communicating with its peer at the same layer.

2.2.1 Application Layer

The application layer handles the details of a particular application. There are two different categories of application layer protocols; user protocols that provide service directly to users, and support protocols that provide common system functions. The most common user protocols are:

- TELNET
- FTP (File Transfer Protocol)
- SMTP (Simple Network Management Protocol)
- HTTP (Hypertext Transfer protocol)
- NFS (Network File System)

There are many other standardised user protocols. Support protocols include:

- SNMP (Simple Network Management Protocol)
- BOOTP (Bootstrap Protocol)
- DNS (Domain Name system)

Similarly, there are many other standardised support protocols.

2.2.2 Transport Layer

Transport layer provides end-to-end communication services for applications. There are two transport protocols in TCP/IP:

- TCP (Transmission Control Protocol)
- UDP (User Datagram Protocol)

TCP is a reliable connection-oriented transport service that provides end-to-end reliability, sequencing and flow control, whereas UDP is a connectionless (datagram) transport service.

2.2.3 Internet Layer

All Internet transport protocols use the Internet Protocol (IP) to carry data from source host to destination host. IP is a connectionless or datagram type internetwork service, providing no end-to-end delivery guarantees. IP only tries to forward the datagrams to a next-hop router until the destination is reached. The layers above IP are responsible for reliable delivery service when it is required. IP includes provisions for addressing, routing, fragmentation and reassembly, type-of-service specification, and security information.

ICMP is a control protocol that is considered to be an integral part of IP that provides error reporting, congestion reporting and first-hop router redirection. IGMP (Internet Group Management Protocol) is used by hosts and routers that support multicasting. Like ICMP, IGMP is considered as an internal part of IP.

2.2.4 Link Layer

To communicate on its directly connected network, a host must implement the communication protocol used to interface to that network. This is implemented in the link layer. The link layer normally includes a device driver in the operating system and the corresponding network interface card (NIC) in the computer. Together they handle all the hardware details of physically interfacing with the communication medium.

ARP (Address Resolution Protocol) and RARP (Reverse Address Resolution Protocol) are used in order to convert between the addresses used by IP and the addresses used by some local network interfaces such as Ethernet and token ring.

2.3 Encapsulation

When an application sends data, the data is sent down the protocol stack, through each layer, until it is sent as a stream of bits across the network. Each layer adds information to the data by prepending headers to the data that it receives. Figure 2.2 shows this process.

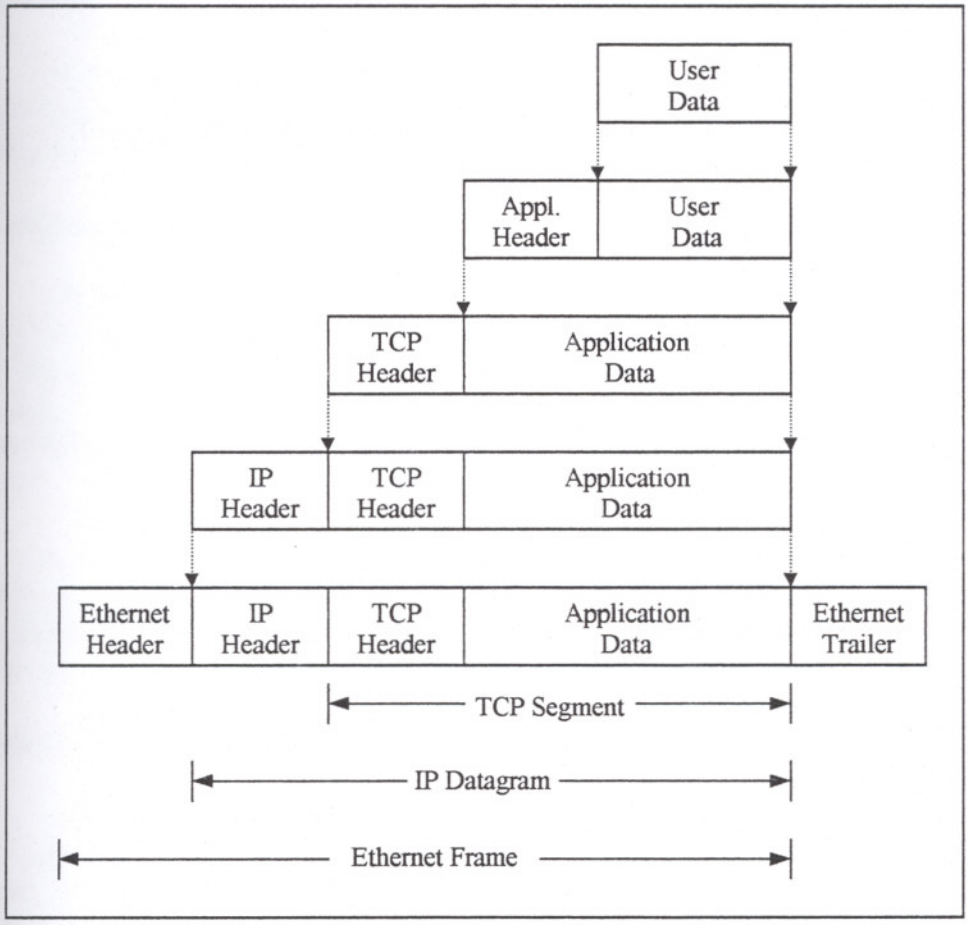


Figure 2.2 Encapsulation of Application Data.

2.3.1 TCP Segments and UDP Datagrams

A segment is the unit of end-to-end transmission in the TCP protocol, which consists of TCP header followed by application data. Similarly, an UDP datagram is the unit of end-to-end transmission in the UDP protocol, and consists of UDP header followed by application data.

The transport layer protocols must add some type of identifier to the transport layer header to indicate the application to which the data belongs. Both TCP and UDP use 16-bit source and destination port numbers to identify applications.

2.3.2 IP Datagrams

An IP datagram is the unit of end-to-end transmission in the IP protocol, which consists of an IP header followed by transport layer data (TCP segments or UDP datagrams).

IP adds an 8-bit protocol field to the IP header that it generates, in order to indicate the protocol to which the data belongs. For example, a value of 1 is for ICMP, 6 for TCP, and 17 for UDP[3].

2.3.3 Frames

A frame is the unit of transmission in a link layer protocol, and consists of a link layer header followed by an IP datagram or a fragment of an IP datagram.

There must be some identification in the link layer header as well, indicating which internet layer protocol generated the data. For example, Ethernet adds a 16-bit frame type field in the Ethernet header.

2.4 Demultiplexing

When a link layer frame is received at the destination host, it starts its way up the protocol stack and all the headers are removed by the appropriate protocol modules. Each protocol module looks at certain identifiers in its header in order to determine which module in the next upper layer protocol receives the data. This process is called "demultiplexing". Figure 2.3 shows the demultiplexing process of a received Ethernet frame.

It should be noted that although ICMP and IGMP messages are considered to be at the same layer as IP, they are located above IP in Figure 2.3, since they are encapsulated in IP datagrams. Similarly, ARP and RARP are located above the Ethernet device driver because they both have their own Ethernet frame types like IP datagrams[4].

2.5 The Client-Server Model

Most Internet applications are written assuming that one side is client and the other side is server. The server application provides some defined services for its clients. The server application waits for a client request to arrive, process the request and send the response back to the client.

Alternatively the server may start a new server to handle the client's request and in this case multiple clients can be serviced concurrently.

2.5.1 Example Services

There are a number of services associated with TCP/IP and Internet. However, TCP/IP does not clearly distinguish between a service, a protocol and an interface[5]. For example "TELNET" is a service, a protocol, and a shell command (user interface) at the same time.

---

[3] J. Postel, J. K. Reynolds, "Assigned Numbers", RFC 990, p. 1, 1986.
[4] W. Richard Stevens, *TCP/IP Illustrated Volume 1: The Protocols,* Professional Computing Series, Addison Wesley, p. 12, 1994.
[5] A.S.Tanenbaum, *Computer Networks,* Prentice Hall, p. 39, 1996.

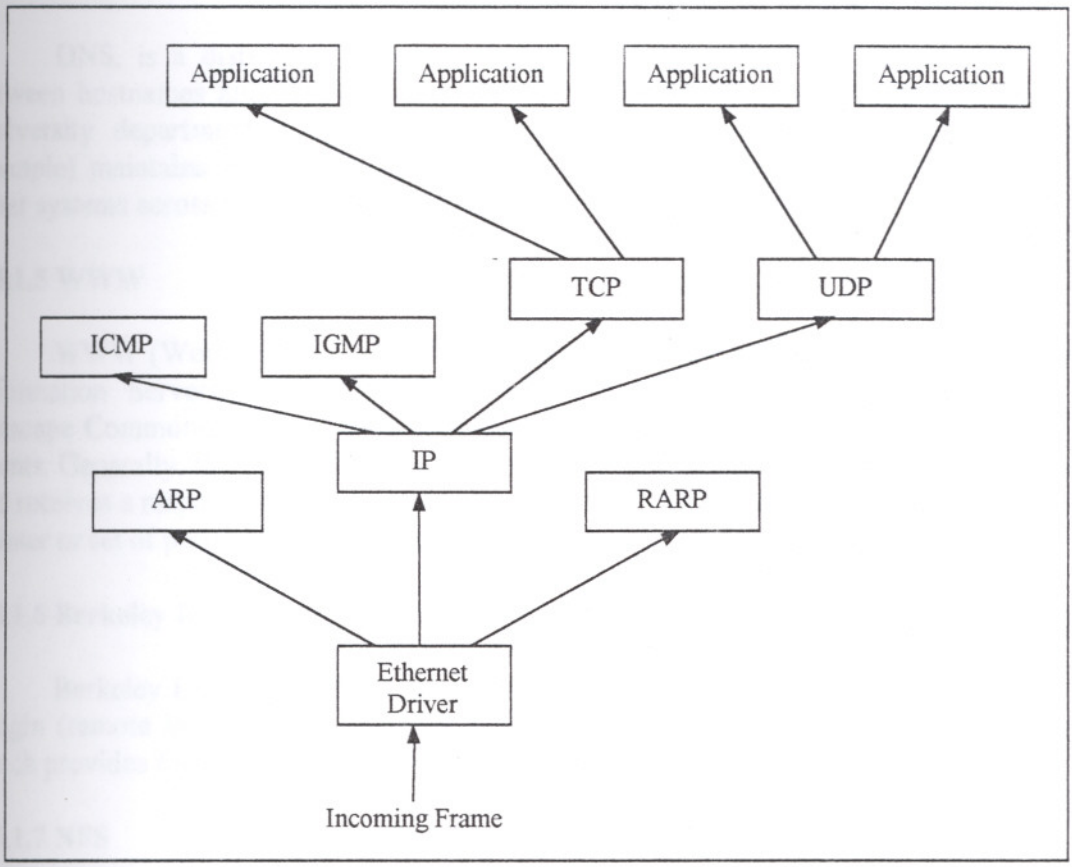


Figure 2.3 Demultiplexing Process of an Ethernet Frame.

2.5.1.1 SMTP

SMTP protocol is used for sending and receiving electronic mail. The user interface to the SMTP is a user agent, of which there are a multitude to choose from. Popular user agents for UNIX include Berkeley Mail, Elm and Pine. The exchange of mail using TCP is performed by a message transfer agent (MTA). The most common MTA for Unix systems is Sendmail. Users normally don't deal with MTA. It is the responsibility of the system administrator to set up the local MTA.

2.5.1.2 Telnet

Telnet is a standard protocol, application and interface that almost every TCP/IP implementation provides. Instead of having a hardwired terminal to each host, a user can login to one host and then remote login across the networks to any other host that runs a telnet server, provided that he/she has an account on that machine.

2.5.1.3 FTP

FTP is a commonly used protocol, application and interface, which is the Internet standard for file transfer. FTP copies a complete file from one system to another via connected networks. To use FTP, a user needs an account to login on an FTP server, or use it with a server that allows anonymous login.

2.5.1.4 DNS

DNS, is a distributed database that is used by TCP/IP applications to map between hostnames and IP addresses. The term distributed is used because each site (university department, campus, company, or department within a company for example) maintains its own database of information and runs a server program that other systems across the Internet can query.

2.5.1.5 WWW

WWW (World Wide Web) is the superset of FTP, gopher, WAIS (Wide Area Information Servers), and other information services using the HTTP protocol. Netscape Communicator and Microsoft Internet Explorer are the most popular WWW clients. Generally, the client contacts a server host, sends a query or information pointer and receives a response. The response may either be a file to be displayed or it may be a pointer or set of pointers to some other server.

2.5.1.6 Berkeley R-Utilities

Berkeley R-utilities employ a concept of mutually trusting hosts. Examples are Rlogin (remote login) which provides a remote login facility and Rsh (remote shell) which provides for executing commands on a remote host.

2.5.1.7 NFS

NFS is a popular service and protocol that provides transparent remote access to shared files across networks. The NFS protocol is designed to be portable across different machines, operating systems, network architectures, and transport protocols. This portability is achieved through the use of Remote Procedure Call (RPC) primitives built on top of an eXternal Data Representation (XDR).

2.5.1.8 X

The "X Window System", or X, is the dominant windowing system used on the Internet today. It uses the network for communications between applications and the I/O devices (the screen, the mouse, etc.), which allows the applications to reside on different machines. This is the source of much of the power of X.

2.5.2 Port Numbers

As mentioned above, every application is identified by a 16-bit port number. The servers are normally known by their "well-known" port numbers. For example, every TCP/IP implementation that provides an FTP server provides that service on port 21, and every Telnet server is on port 23. RPC servers do not use well-known ports, instead there is a registrar called "portmapper" that keeps track of which RPC programs are using which ports. The RPC portmapper listens to its clients on port 111[6].

[6] J. Postel, J. K. Reynolds, "Assigned Numbers", RFC 1340, p. 13, 1992.

The well-known port numbers range between 1 and 1023, and they are assigned by the IANA (Internet Assigned Numbers Authority), and their current policy is to assign both TCP and UDP protocols when assigning a port number.

A client application does not care what port number it uses on its end. Usually the underlying operating system allocates the port numbers on behalf of the client applications as long as the user running the client needs its service. These port numbers usually are between 1024 and 5000. The port numbers above 5000 are intended for other services that are not well known on the Internet.

Chapter 3

AUTHENTICATION PROBLEMS IN IPv4

IP (Internet Protocol) resides at the internet layer of the TCP/IP suite. Therefore, all information contained in higher layer protocols is encapsulated in IP datagrams while in transit in the Internet. Furthermore, IP handles the details of Internet addressing and moving of information between Internet hosts. It is worth focusing on this protocol, because the direction of the information flow and addresses are, together, related with one of the most important aspects in computer security, which is "authentication".

The current version of IP is 4 and in this chapter a fatal authentication weakness in so-called IPv4 will be introduced.

3.1 Background

The original specification for IP is RFC791, which is also the general reference for this section[1]. This section gives a brief description of the internet layer from a security point of view and most of the information provided here will be needed throughout the thesis for the reasons explained above.

3.1.1 IP Function Description

IP provides for transmitting blocks of data called datagrams from sources to destinations, where sources and destinations are hosts identified by fixed length addresses. IP also provides for fragmentation and reassembly of long datagrams.

IP is a connectionless protocol. It does not maintain any state information about successive datagrams. Each datagram is treated as an independent entity unrelated to any other internet datagram. There are no connections nor logical circuits, meaning that IP datagrams may get delivered out of order since they may follow different routes until they reach their same destination. IP does not provide either a reliable communication facility. There is no guarantee that an IP datagram successfully gets its destination. There is no error control of data, only a header checksum. When something goes wrong, IP simply throws away the datagram and tries to send an ICMP error message back to the source.

As a result, IP only implements two basic functions: addressing and fragmentation. This section focuses on the basic operation of the IP in these areas.

3.1.1.1 IP Addressing

IP provides every Internet host with fixed length Internet addresses of 32 bits. These 32-bit addresses are normally written as four decimal numbers, one for each byte of the address. This is called dotted-decimal notation. Instead of using a flat address space, there is a structure to Internet addresses. An address begins with a network number followed by a local address. As shown in Figure 3.1, there are five different

[1] J. Postel, "Internet Protocol", RFC 791, 1981.

classes of IP addresses: class A through class E. Class D addresses are used for IP multicasting, while class E addresses are reserved for experimental use[2].

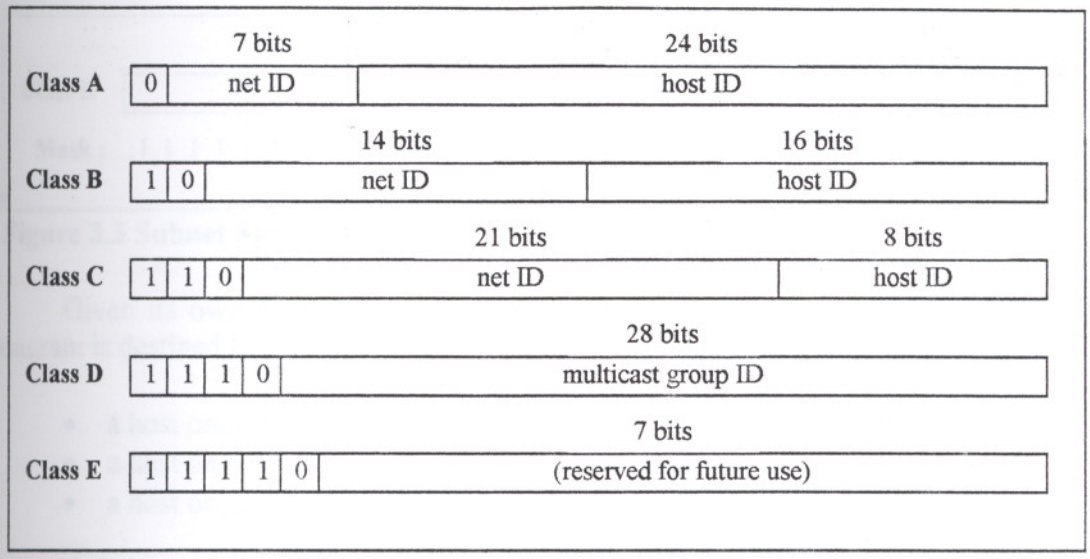


Figure 3.1 IP Address Classes.

3.1.1.1.1 Subnet Addressing

All hosts are required to support subnet addressing[3]. In this addressing method, the network part of the IP address is divided into a number of subnets. Figure 3.2 examplifies the subnetting procedure of a class B address.

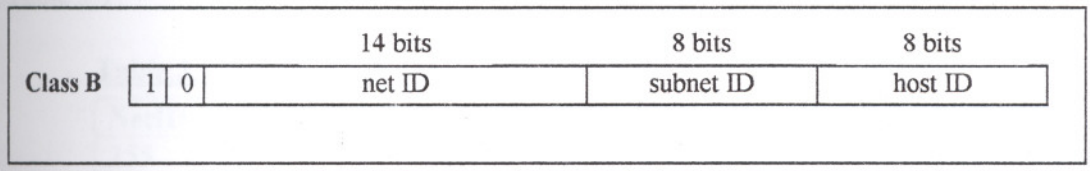


Figure 3.2 Subnetting Procedure of a Class B IP Address.

This scheme makes sense because class A and class B addresses have too many bits in their host ID field. Generally, it is not necessary to attach that many hosts on a single network. Rather it is divided into a number of subnets with their own routers.

The most important advantage of subnetting is that it hides the details of internal network organization to external routers. For example, using a single class B address with 32 subnets instead of 32 class C addresses reduces the size of the Internet's routing tables.

To support subnets, it is necessary for an Internet host to store one more 32-bit quantity, called address mask or subnet mask[4]. This is a bit-mask with bits set in the

---

[2] W. R. Stevens, *TCP/IP Illustrated Volume 1: The Protocols,* Professional Computing Series, Addison Wesley, pp. 7-8, 1994.
[3] J. Mogul, J. Postel, "Internet Standard Subnetting Procedure", RFC 950, 1985.
[4] ibid, p. 4.

fields corresponding to the IP network number, and additional bits set corresponding to the subnet number field. This is shown in Figure 3.3.

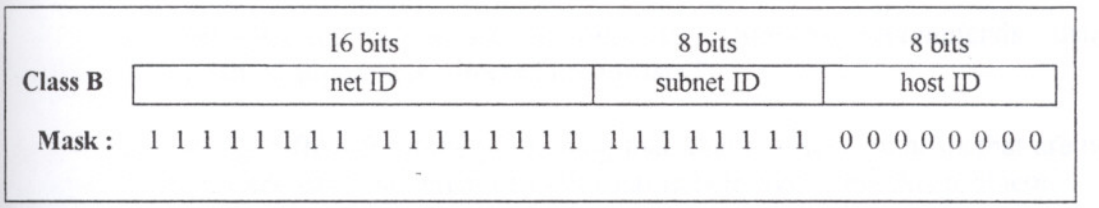


Figure 3.3 Subnet Mask.

Given its own IP address and its subnet mask, a host can determine if an IP datagram is destined for:

- a host on its own subnet
- a host on a different subnet on its own network
- a host on a different network

IP makes the necessary comparisons in order to make routing decisions.

3.1.1.1.2 Broadcast Addresses

Broadcast addresses are used when a host needs to send a datagram destined for all hosts on a given network. For example, during its configuration process, a host that does not know its subnet mask or IP address, may broadcast a datagram destined for all hosts on its local network. There are four types of broadcast addresses. These are shown in Table 3.1.

Table 3.1 Broadcast Addresses.

NetID	Subnet ID	host ID	Description
255		255	Limited broadcast
netID		255	Net-directed broadcast
netID	Subnet ID	255	Subnet-directed broadcast
netID	255	255	All-subnets directed broadcast

A datagram destined for limited broadcast addresses will be received by every host on the connected physical network. Limited broadcasts MUST NOT be forwarded by any router[5,6].

Net-directed broadcasts are destined for all hosts on a given network. A router MUST forward a net-directed broadcast by default, but it MAY also have a configuration option to disable this forwarding[7].

[5] In RFCs 1122 and 1812, the capitalized words MUST, SHOULD and MAY are used synonymously to mean "required", "recommended" and "optional", respectively.
[6] F. Baker, "Requirements for IP Version 4 Routers", RFC 1812, p. 93, 1995.
[7] ibid, pp. 93-94.

Subnet-directed broadcasts are destined for all host on the specified subnet. In a CIDR (Classless Inter Domain Routing) domain these are indistinguishable from net-directed broadcast. Therefore the two are treated together[8].

All-subnets-directed broadcast is obselete[9]. Stevens recommends using multicasting instead of all-subnets-directed broadcasts[10].

Broadcasting suffers from the processing load that is places on hosts that are not interested in the broadcasts. The intent of multicasting is to overcome this problem.

3.1.1.1.3 Special Addresses

The other special IP addresess are listed in Table 3.2.

Table 3.2 Special IP Addresses.

netID	subnet ID	host ID	Description
127		Any	Loopback
0		0	This host on this network
0		Host	Specified host on this network

The loopback address supports a "loopback interface" that allows a client an a server on the same host to communicate each other using TCP/IP. The Class A net ID 127 is reserved for the loopback interface. Addresses of this form MUST NOT appear outside a host[11].

An address with netID=hostID=0 means "this host on this network". This MUST NOT be sent by any router or host, except as a source address as part of an initialization procedure[12,13].

An address with netID=0, with a specific hostID means "specified host on this network". The same rules apply to this type of addresses[14,15].

3.1.1.1.4 Private Address Space

Three blocks of the IP address space were allocated for private internets. Table 3 lists these addresses.

---

[8] F. Baker, "Requirements for IP Version 4 Routers", RFC 1812, p. 94, 1995.
[9] ibid, p. 94.
[10] W. R. Stevens, *TCP/IP Illustrated Volume 1: The Protocols,* Professional Computing Series, Addison Wesley, p. 172.
[11] F. Baker, "Requirements for IP Version 4 Routers", RFC 1812, p. 48, 1995.
[12] ibid, p. 47.
[13] R. Braden, "Requirements for Internet Hosts – Communication Layers", RFC 1122, p. 30, 1989.
[14] ibid, p. 47.
[15] ibid, p. 30.

Table 3.3 Private Addresses.

Block	Prefix
10.0.0.0 –10.255.255.255	10/8
172.16.0.0.0 –172.31.255.255	172.16/12
192.168.0.0 –192.168.255.255	192.168/16

The first block is referred to as "24 bit block", the second as "20 bit block" and the third as "26 bit block". The first is a single class A network number, while the second is a set of 16 contiguous class B network numbers and the third is a set of 256 class C network numbers[16].

These addresses do not appear outside the private networks.

3.1.1.2 IP Fragmentation

Most types of networks have an upper limit on the number of bytes of data that can be transmitted. This limit is called the maximum transmission unit or MTU. An IP datagram is fragmented, whenever it must traverse a network with a smaller MTU.

To assemble the fragments of a datagram, the destination host IP combines the datagrams that all have the same value for the four fields in the IP header: identification, source, destination and protocol. The combination is done by concerning the relative position of a fragment indicated by the fragment offset field. The first fragment will have the fragment offset zero, and the last fragment will have the more-fragments flag reset to zero.

3.1.2 IP Header

IP header contains the following fields:

- Version (4 bits): version field, which indicates the format of the internet header. The current protocol version is 4, so IP is sometimes called IPv4.
- IHL (4 bits): Internet header length.
- Type of Service (8 bits): type of service.
- Total Length (16 bits) length of the datagram.
- Identification (16 bits): identifying value assigned by the sender to aid in assembling the fragments of a datagram.
- Flags:

  DF: 0 = May Fragment, 1 = Don't Fragment.
  MF: 0 = Last Fragment, 1 = More Fragments.

[16] B. Moskowitz, D. Karrenberg, G. de Groot, E. Lear, "Address Allocation for Private Internets", RFC 1918, p. 4, 1996.

- Fragment Offset (13 bits): fragment offset field indicating where in the datagram this fragment belongs.
- TTL (8 bits): time to live field.
- Protocol (8 bits): protocol field indicating the next level protocol used in the data portion of the datagram.
- Header Checksum (16 bits): IP header checksum.
- Source Address (32 bits): source IP address.
- Destination Address (32 bits): destination IP address.
- Options: A variable length list of optional information for the datagram.

3.1.3 TTL of an IP Datagram

In order to discard undeliverable datagrams, and to bound the maximum lifetime of a datagram, IP defines a TTL value for each IP datagram that is the maximum time the datagram is allowed to remain in the Internet. If this field contains the value zero, then the datagram must be destroyed. This field is modified in internet header processing. The time is measured in units of seconds, but since every module that processes a datagram must decrease the TTL by at least one even if it process the datagram in less than a second, the TTL must be thought of only as an upper bound on the time a datagram may exist.

3.1.4 IP Routing

Datagrams are transmitted to their destinations using packet-switching nodes called routers (or gateways). Routers on the path connecting two parties attempt to forward the datagrams until the destination is reached. The selection of the path is called "routing".

IP routing is simple. If the destination is on a directly connected network, then the IP datagram is sent directly to the destination. Otherwise, the host sends the datagram to a router on the directly connected network and lets the router deliver the datagram to its destination.

3.1.4.1 Local/Remote Decision

To decide if the destination is on a directly connected network, the following algorithm is used by the sending host[17]:

> If the destination IP address bits extracted by the address mask match the source IP address bits extracted by the same mask, then the destination is on the corresponding connected network.

[17] R. Braden, "Requirements for Internet Hosts – Communication Layers", RFC 1122, p. 47, 1989.

If not, then the destination is accessible only through a router.

3.1.4.2 Router Selection

IP employs a routing table that it searches whenever it receives an outbound IP datagram. Each entry in a routing table provides the necessary information about the appropriate router to a given destination.

IP performs three steps when it searches its routing table:

1. Search the routing table for an entry that matches the complete destination address (matching network ID and host ID).

2. Search the routing table for an entry that matches just the destination network ID. All the hosts on the destination network can be handled with this single routing table entry.

3. Search for an entry labelled "default".

If none of the steps works, the datagram is considered to be undeliverable. In this situation, an ICMP error message is sent back.

3.1.4.3 Static and Dynamic Routing

When the network is small and there is a single connection point to other networks, the routing table entries are simply created by a system administrator. In this situation the routes are called static and the routing policy is referred to as "static routing".

"Dynamic routing" occurs when the routers talk to adjacent routers, informing each other about which networks each router is currently connected to. The routing table entries are updated according to this information.

The routers communicate using a routing protocol such as RIP (Routing Information Protocol), EGP (Exterior Gateway Protocol), etc.

3.1.5 Interesting IP Options

As mentioned above, IP provides a number of optional fields, however they are not commonly used. This section focuses only on security and handling restrictions, and strict and loose source routing options.

3.1.5.1 Security and Handling Restrictions

This option provides for labeling each datagram with the sensitivity of the information it contains. The labels include both a hierarchical component (Secret, Top secret, etc.) and an optional category: nuclear weapons, cryptography, hammer procurement, etc.

The labels indicate the security level of the ultimate sending and receiving entities. An entitiy may not write to a medium with a lower security level, because that would allow disclosure of confidential information. For obvious reasons, it may not read from a medium containing information more highly classified. The combination of these two restrictions will usually dictate that the sending and receiving entities be at the exact same level[18].

3.1.5.2 Source Routing

The source routing option provides a means for the source of an IP datagram to supply routing information to be used by the routers in forwarding the datagram to the destination, and to record the route information. There are two different source routing options:

- Loose source routing: The sender specifies a list of IP addresses that the datagram must traverse, but the datagram can also pass through other routers between any two addresses in the list.

- Strict source routing: The sender specifies the exact path that the IP datagram must follow. If a router encounters a next hop in the source route that isn't on a directly connected network, an ICMP source route failed error is returned.

3.2 Source IP Address Forgery

An important point to note is that IP does not provide any authentication mechanism. The security and handling restriction provisions mentioned above fall into a different perspective of information security, which is "mandatory access control". These are primarily used by military sites[19].

In this section the potential risks caused by this authentication weakness and the action of forging source IP addresses will be introduced.

3.2.1 Description

The weakness in IP is that a source host itself fills the source address field in the header of an outbound IP datagram. In theory, any host can transmit an IP datagram with any source address[20]. Several implementations does allow privileged users to send such datagrams. However, an attacker may also bypass the operating system by directly injecting packets to the physical network[21].

The action of sending IP datagrams with forged source addresses is referred to as "IP spoofing". Although the term "IP Spoofing" is primarily used for describing fake TCP connections, it should be considered as an internet layer attack independent of any

---

[18] W. R. Cheswick, S. M. Bellovin, *Firewalls and Internet Security: Repelling the Wily Hacker*, Professional Computing Series, Addison Wesley, pp. 21-22, 1994.
[19] ibid, p. 22.
[20] R. T. Morris, "A Weaknes in the 4.2BSD Unix TCP/IP Software", AT&T Bell Laboratories, Computing Science Technical Report No. 117, p.1, 1985.
[21] ibid, p. 1.

other protocols encapsulated within IP datagrams. For example, the same technique may be used to forge UDP and ICMP traffics as well.

3.2.2 Impact

There is no guarantee that a datagram was actually sent from a given source address[22]. Herein lies most of the threat against the integrity, availability, and secrecy of today's Internet assets. The attackers may gain unauthorised accesses to the target sites or launch several denial-of-service attacks in order to interrupt the availability of Internet services.

It should be also noted that there exists neither any provision to discover the true origin of a datagram[23]. Therefore, in many cases, logging such activities will be meaningless.

Another important point to note is that when combined with network monitoring attacks this weakness may pose more serious problems. Network monitoring attacks will be described in Chapter 7.

3.2.3.1 Unauthorized Access

In the Internet world, there exist several situations where authorization decisions are made regarding the IP addresses (or names, which are obtained again from IP addresses) of clients. Usually, a list of trusted client addresses is maintained and controlled whenever a given client demands for access. At the application layer there are at least three major services which employ such schemes: Berkeley R-utilities, X and NFS. Several security systems such as TCP wrappers and Firewalls are also based on IP addresses.

Therefore, by impersonating the trusted clients, source address spoofing attacks may be used to gain unauthorized access to the Internet services or pass through mal configured Firewalls, which employ IP address based access control.

An important point to note here, is that in order to succeed in this kind of attacks an attacker may or may not have to be able to monitor the traffic generated by the target. This depends on several conditions, which will be studied in the following chapters. However, the ability of forging source IP addresses is always of considerable advantage to the attacker.

3.2.3.2 Denial of Service Attacks

As noted above, the ease of forging source IP addresses makes the discovery of the true origin of an attacker difficult. From an attacker's point of view, this situation reduces the risk of being located by any system administrator. Therefore, source IP address forgery techniques are useful when launching denial-of-service attacks.

---

[22] R. T. Morris, "A Weaknes in the 4.2BSD Unix TCP/IP Software", AT&T Bell Laboratories, Computing Science Technical Report No. 117, p.1, 1985.

[23] ibid, p.1.

Chapter 4

TCP UNRELIABILITIES

TCP (Transmission Control Protocol) is the connection oriented transport layer protocol of the TCP/IP suite, designed to provide a reliable logical circuit between pairs of applications in hosts attached to the Internet. TCP assumes that it can obtain an unreliable datagram service from lower level protocols. In the Internet, this service is provided by IP.

In this chapter, SYN-flooding, sequence number guessing and connection hijacking attacks which exploit the authentication weakness of IP will be described in detail.

4.1 Background

This section involves a brief description of TCP operation from a security point of view. The general reference for this section if RFC793, the original specification of TCP[1].

4.1.1 TCP Function Description

As noted above, the primary purpose of TCP is to provide a reliable connection service on top of a less reliable internet communication system. For this, TCP supports facilities in the following areas: reliability, flow control, multiplexing and connections. In this section, the basic operation of the TCP in each of these areas will be outlined.

4.1.1.1 Reliability

TCP must recover from data that is damaged, lost, duplicated or delivered out of order in the Internet. This is achieved by assigning a sequence number to each byte transmitted, and requiring a positive acknowledgment from the receiving TCP. If the acknowledgment is not received within a timeout interval, the data is retransmitted.

At the receiver, the sequence numbers are used to correctly order segments that may be received out of order and to eliminate duplicates. Damage is handled by adding a checksum to each segment transmitted, checking at the receiver, and discarding damaged segments.

4.1.1.2 Flow Control

TCP provides a means for the receiver to govern the amount of data sent by the sender. This is achieved by returning a "window" with every acknowledgment indicating a range of acceptable sequence numbers beyond the last segment successfully received. The window indicates an allowed number of bytes that the sender may transmit before receiving further permission.

[1] J. Postel, "Transmission Control Protocol", RFC 793, 1981.

4.1.1.3 Multiplexing

To allow many applications within a single host to use TCP services simultaneously, TCP provides a set of ports within each host. The combination of an IP address and a port number is called a "socket".

Binding of ports to applications is handled independently by each host. However, some frequently used services are generally attached to fixed port numbers called "well-known" ports. These services can be accessed through their known sockets.

4.1.1.4 Connections

In order to support the reliability and flow control, TCP initializes and maintains certain status information for each data stream. The combination of this information including sockets, sequence numbers, and window sizes, is called a connection.

A pair of socket (4 tuple consisting of the client IP address, client port number, server IP address and server port number) specifies the two end points that uniquely identifies each TCP connection in the Internet. A local socket may participate in many connections to different foreign sockets. A connection can be used to carry data in both directions, that is, it is full duplex.

4.1.2 TCP Header

The TCP header fields are:

- Source port (16 bits): source port number
- Destination port (16 bits): destination port number
- Sequence number (32 bits): the sequence number of the first data byte in the segment. The initial sequence number (ISN) if SYN control bit is set.
- Acknowledgment number (32 bits): If the ACK control bit is set this field contains the value of the next sequence number the sender of the segment is expecting to receive.
- Data offset (4 bits): offset of the data in the segment
- Control bits

  URG: Urgent pointer
  ACK: Acknowledgment
  PSH: Push function
  SYN: Synchronize sequence numbers
  RST: Reset the connection
  FIN: No more data from the sender

- Window (16 bits): window size of the sender

- Checksum (16 bits): checksum of the header and data
- Urgent Pointer (16 bits): urgent pointer
- Options: A variable length of TCP options

4.1.3 TCP States

A TCP connection progresses through a series of state during its lifetime. The states are: LISTEN, SYN-SENT, SYN-RECEIVED, ESTABLISHED, FIN-WAIT-1, FIN-WAIT-2, CLOSE-WAIT, LAST-ACK, TIME-WAIT and CLOSED. CLOSED state is fictional because it represents a state when there is no connection. Briefly the meanings of the states are:

- LISTEN: Waiting for a connection request from any remote TCP.
- SYN-SENT: Waiting for a matching connection request after having sent a connection request.
- SYN-RECEIVED: Waiting for a confirming connection request acknowledgment after having both received and sent a connection request.
- ESTABLISHED: Represents an open connection, data received can be delivered to the user. The normal state for the data transfer phase of the connection.
- FIN-WAIT-1: Waiting for a connection termination request from the remote TCP, or an acknowledgment of the connection termination request previously sent.
- FIN-WAIT-2: Waiting for a connection termination request from the remote TCP.
- CLOSE-WAIT: Waiting for a connection termination request from the local user.
- CLOSING: Waiting for a connection termination request acknowledgment from the remote TCP.
- LAST-ACK: Waiting for an acknowledgment of the connection termination request previously sent to the remote TCP (which includes an acknowledgment of its connection termination request).

A TCP connection progresses from one state to another on response to several events. The events are user calls: OPEN, SEND, RECEIVE, CLOSE, ABORT and STATUS; and the incoming segments containing: SYN, ACK, RST and FIN flags.

4.1.4 TCP Connection Establishment and Termination

When two applications wish to communicate, their TCPs must first establish a connection in order to initialize the status information on each side. When their communication is complete, the connection is terminated to free the resources for other uses.

4.1.4.1 Connection Establishment

For a connection to be established, the two TCPs must synchronize on each other's sequence numbers. This is done by exchanging connection establishing segments carrying a SYN control bit and initial sequence numbers. The synchronization requires each side to send it's own ISN and to receive a acknowledgment of it from the other side. Each side must also receive the other side's ISN and send an acknowledgment. This is illustrated in Figure 4.1.

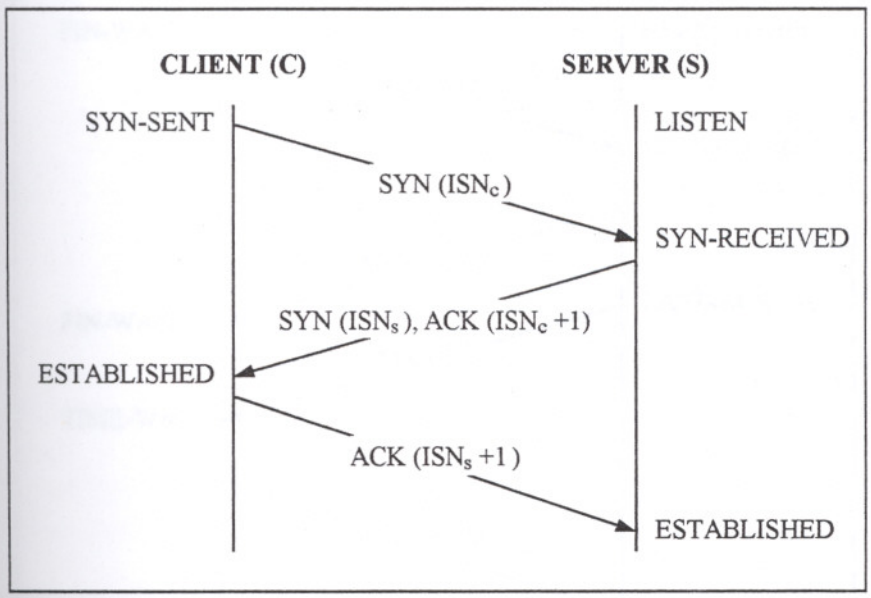


Figure 4.1 TCP 3-Way Handshake.

A three-way handshake is necessary because the client and server sequence numbers are not tied to a global clock in the network, and TCPs may have different mechanisms for picking the ISNs. The server which receives the first SYN has no way of knowing whether the segment is an old delayed or not, and thus it must ask the sender to verify this SYN.

4.1.4.2 Connection Termination

Closing a connection is an operation meaning "I have no more data to send." While it takes three segments to establish a connection, it takes four to terminate a connection. Since a TCP connection is full-duplex, each direction must be shut down independently.

When one end of a connection has no more data to send it sends a FIN segment. It is normally the client that determines when a connection should be terminated. The

receipt of a FIN only means there will be no more data flowing in that direction. A TCP can still send data after receiving a FIN. The termination procedure of a full duplex TCP connection is illustrated in Figure 4.2.

The TIME-WAIT state is necessary for the full-duplex reliable close handshake of TCP. This lets TCP resend the final acknowledgment in case it is lost. This state is also called 2MSL (Maximum Segment Lifetime).

Actually, the time to delay the final close step depends on the round-trip time of the path instead of MSL[2].

Section 4.1.6.2 gives additional information on the MSL and the TIME-WAIT state.

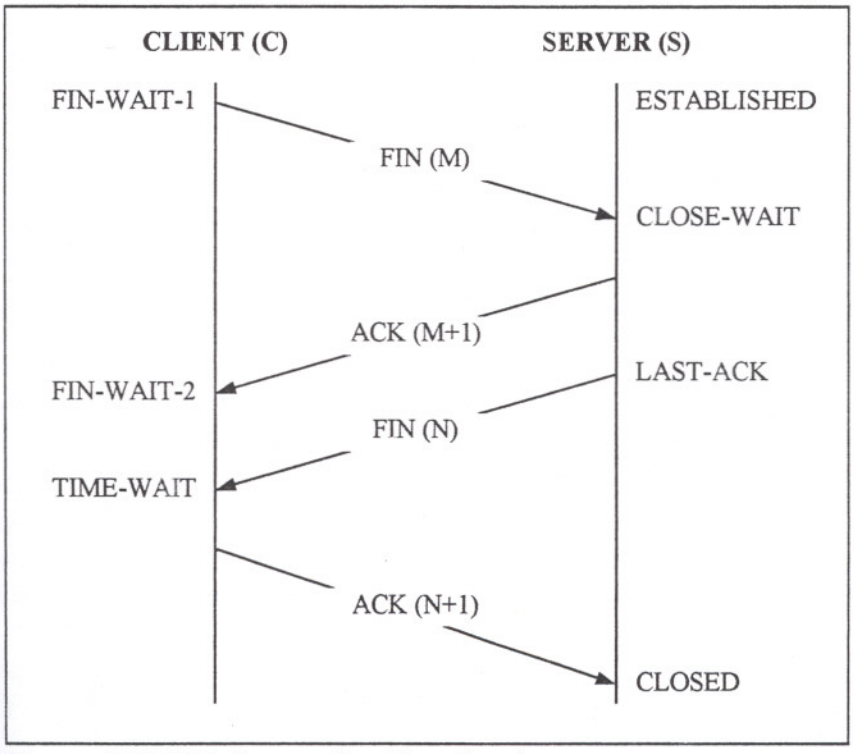


Figure 4.2 TCP Connection Termination.

4.1.4.3 Half-open Connections

An established connection is referred to as "half-open" if one of the TCPs has closed or aborted the connection at its end without the knowledge of the other, or if two ends become desynchronized.

TCP expects the half-open connections to be unusual and does not involve a complete recovery procedure.

[2] V. Jacobson, R. Braden, L. Zhang, "TCP Extension for High-Speed Paths", RFC 1185, p. 12, 1990.

4.1.4.4 Reset Generation and Processing

TCP sends a RST (reset) segment whenever a segment arrives that does not appear correct for the referenced connection.

In all states except SYN-SENT, all RST segments are validated by checking their sequence numbers. A reset is valid if its sequence number is in the window. In the SYN-SENT state (a RST received in response to an initial SYN), the RST is acceptable if the ACK field acknowledges the SYN.

The receiver of a RST first validates it, then changes state. If the receiver was in the LISTEN state, it ignores it. If the receiver was in the SYN-RECEIVED state and had previously been in the LISTEN state, then the receiver returns to the LISTEN state, otherwise the receiver aborts the connection and goes to the CLOSED state. If the receiver was in any other state, it aborts the connection and advises the user and goes to the CLOSED state.

4.1.5 Round-Trip Time Estimation

Dynamically estimating the round-trip time (RTT), the interval between the sending of a packet and the receipt of its acknowledgment, is a key function in TCP. As noted above if a packet remains unacknowledged for two long, it is assumed to have been lost and must be retransmitted. Estimated round-trip times are used to determine the RTO value (retransmission timeout).

4.1.5.1 RFC 793 RRT Estimation Algorithm

Fundamental to TCP RTT estimation algorithm is the measurement of RTTs experienced on a given connection. TCP attempts to track the changes in RTT by updating an estimate of the average RTT accordingly to these measurements. Jacobson details the description of the TCP RTT estimation algorithm as follows[3]:

Given a new measurement m of the RTT ,TCP updates an estimate of the average RTT a by,

$$a \leftarrow (1\text{-}g)\, a + gm$$

where g is a gain ($0<g<1$) that should be related to signal-to-noise ratio (or equivalently, variance) of m. Rearranged to collect the terms multiplied by g, the expression becomes,

$$a \leftarrow a + g\,(m\text{-}a)$$

where a is the prediction of the next measurement, m-a is the error in that prediction. The above expression states that a new prediction is based on the old prediction plus some fraction of the error in that prediction.

[3] V. Jacobson, "Congestion Avoidance and Control", Proceedings of SIGCOMM '88, p. 17, 1988.

The prediction error is the sum of two components:

1. Error due to noise in the measurement (random, unpredictable effects like fluctuations in competing network traffic).
2. Error due to a bad choice of a.

Calling the random error E_r and the estimation error E_e, the expression becomes,

$$a \leftarrow a + g E_r + g E_e$$

The term gE_e gives a value in the right direction while gE_r gives a value in a random direction. Over a number of samples, the random values cancel each other out so that the algorithm tends to converge to the correct average. It should be noted that a will oscillate randomly around the true average and the standard deviation of a will be g *sdev(m)* and a will converge to the average exponentially with the time constant $1/g$. Thus, a smaller g gives a more stable a at the expense of taking much longer time to get the true average. Typical gain choices are 0.1-0.2 in TCP implementations[4].

4.1.5.2 Extensions

Although TCP performance issues are not discussed here, it is worth mentioning several extensions to the RFC 793 rule for better RTT estimation.

Mills observed that RTTs were roughly Possion distributed, however with brief periods of high delay[5]. During these periods, he found that TCP algorithm often did not adapt quickly enough, causing unnecessary retransmissions which added to the network load. As a result he suggested a larger g when $a < m$, allowing a to adapt more quickly to sudden increases in network delay.

Jacobson detailed the solution by calculating the RTO based on both mean and mean deviation which provides better response to wide fluctuations in the RTTs[6].

4.1.6 Initial Sequence Number Selection

TCP employs a clock driven ISN selection mechanism. During the development of TCP, a great deal of effort was adopted to the problem of ISN selection. However, the solution depended on several factors concerning the reliability of the protocol. In this section the reasons behind TCP ISN selection scheme will be analyzed in detail.

4.1.6.1 Duplicate Segments

TCP places no restriction on a particular connection being used over and over again. Instances of a connection are referred to as "incarnations" of that connection. The problem that arises from this is "how does the TCP identify duplicate segments from previous incarnations of the connection?"

---

[4] V. Jacobson, "Congestion Avoidance and Control", Proceedings of SIGCOMM '88, p. 17, 1988.
[5] D. L. Mills, "Internet Delay Experiments", RFC 889, p. 10, 1983.
[6] V. Jacobson, "Congestion Avoidance and Control", Proceedings of SIGCOMM '88, 1988.

Duplications of sequence numbers might happen in either of two situations:

1. A terminated connection is immediately reopened.
2. A connection breaks with loss memory (due to a host crash).

Duplicate segments and may cause confusion at the receiver as to which data is new and which is old.

The handling of the duplicate segments depends on the upper bound of the lifetime of a segment. TCP assumes a "Maximum Segment Lifetime" or MSL for every segment. In TCP/IP the MSL bound is enforced by an IP mechanism, the "Time to Live" or TTL field.

4.1.6.2 Protection against Duplicate Segments

TCP combines three mechanisms in order to avoid duplicate segments. This section focuses on these mechanisms.

4.1.6.2.1 TIME-WAIT State

As noted above, one end of a closed connection is left in a "busy" state, known as the "TIME-WAIT" state, for a time of 2MSL. Another effect of this wait is that while the TCP connection is in the 2MSL wait, the socket pair defining that connection cannot be reused. That connection can be used when the 2MSL wait is over. Any delayed segments that arrive for a connection while it is in the 2MSL wait will be discarded.

4.1.6.2.2 Quite Time

The TIME-WAIT state provides protection against delayed segments from an earlier incarnation of a connection from being interpreted as part of a new connection. However, this works only if a host with connections in the TIME-WAIT state does not crash. To protect against this situation, TCP does not create any connections for MSL seconds after rebooting. This is called "Quite Time".

4.1.6.2.3 Clock Driven ISN Selection

Although the TIME-WAIT and "Quite Time" mechanisms together are logically sufficient to protect against duplicate segments, a clock driven ISN selection mechanism gives additional assurance for a host that has set its MSL too small (it should be noted that the MSL is loosely defined and even more loosely implemented in the Internet, the TCP specification only assumes a value of 120 seconds for MSL)[7].

Jacobson explains the clock driven scheme as follows[8]:

$ISN = (\text{integer}(R.t)) \bmod 2^{32}$

where t is the current time relative to an arbitrary origin, and R is a constant. R is chosen so that *ISN* will advance faster than sequence numbers. TCP proposes R=250KBps so

[7] V. Jacobson, R. Braden, L. Zhang, "TCP Extension for High-Speed Paths", RFC 1185, p. 19, 1990.
[8] ibid, p. 15.

that the *ISN* value will cycle every 2^{32}/250Kbps = 4.55 hours. In other words, the 32-bit *ISN* value should be incremented by one every 4 microseconds (1/*R*). Then, TCP will operate reliably without any MSL based mechanisms in the following restricted domain:

- Total data sent less than 2^{32} bytes, and
- Effective sustained rate less than 250Kbps, and
- Connection duration less than 4.55 hours.

Jacobson states that majority of TCP usage falls into this restricted domain and the third component is the most commonly violated (in 1990)[9].

4.2 SYN-Flooding Attack

This section describes a potential denial of service attack targeting the availability of public TCP services such as SMTP, HTTP and FTP, etc.

4.2.1 The Backlog Queue

TCP servers are concurrent. A TCP server starts a new process to handle each client therefore the listening server is always ready to handle the next coming connection request. However, there is still a chance that multiple connection requests arrive while the server starts a new process. In order to handle these incoming connection requests while the listening application is busy, TCP employs a fixed length queue for the connections that have not been accepted by the server application. This is referred to as the "backlog queue". However, there is an upper limit to the number of connection requests waiting in this queue. This limit is specified by the listening application[10].

When a new connection request arrives (i.e., a SYN segment), TCP acknowledges this if there is room for this new connection on the requested service's queue. However it should be noted that the server application will not see these new connection until the third segment of the 3-way handshake is received[11]. If there is no room on the requested service's queue, TCP ignores the received SYN packet and does not respond with a RST[12]. By ignoring the SYN packet the server forces the client to retransmit the SYN later, hoping that the queue will then have room[13].

4.2.2 The Attack

The TCP SYN-Flooding attack exploits this design. The attacker generates several TCP SYN segments with invalid source IP addresses to a target TCP server. Since the source hosts are non-existing or closed, the 3-way handshake for these

---

[9] V. Jacobson, R. Braden, L. Zhang, "TCP Extension for High-Speed Paths", RFC 1185, p. 20, 1990.
[10] W. Richard Stevens, *TCP/IP Illustrated Volume 1: The Protocols,* Professional Computing Series, Addison Wesley, p. 257, 1994.
[11] ibid, p. 258.
[12] ibid, p. 258.
[13] ibid, p. 260.

connections will never complete, resulting in several half-open connections filling up the server's backlog queue[14].

4.2.3 Impact

The target service will be unavailable (interrupted) until a connection establishment timer expires. Most implementations limit the SYN-RECEIVED state to 75 seconds[15]. However, the attacker may easily repeat the attack in 75 seconds periods.

The success of the attack depends on several factors such as the backlog queue limit, the timeout value for connection establishment and the interarrival times of SYN segments. Let

- T: SYN-RECEIVED state timeout in seconds.
- L: Per-port backlog queue limit.
- A: Number of received SYN packets per second.

Then, in order to succeed, the attacker will have a lower limit to A, which is L/T. For example, if T=75 seconds and L=15, the attacker must generate the SYN segments so that the target receives at least 1 SYN segment per five seconds.

4.3 IP Spoofing Attack

Morris described how an attacker can profit from a series of vulnerabilities found in TCP and IP in order to gain unauthorized access to a remote server[16]. Bellovin described the details of this attack[17].

Briefly, the attacker determines a trust pattern based on IP address authentication, disables the trusted host, predicts the target's sequence numbers and establishes a one way TCP connection to the target.

4.3.1 Berkeley R-utilities

The attacker must first determine a trust pattern based on IP address amongst the target server and a client host. The trust pattern assumed by Morris was being carried out between the target's Rsh (remote shell) server and a trusted client host. Therefore it is worth looking at Berkeley R-utilities before proceeding with the description of the attack:

The R-utilities rely on the BSD authentication mechanism. One can remote login to a host or execute remote shell commands without entering a password if the authentication criteria are met. These criteria are[18]:

---

[14] daemon9, route, infinity, "IP-Spoofing Demystified", Phreak Magazine, Volume 7, Issue 48, File 14, pp. 5-6, 1996.

[15] Schuba C. L. et al, "Analysis of a Denial of Service Attack on TCP", IEEE Symposium on Security and Privacy, 1997.

[16] R. T. Morris, "A Weaknes in the 4.2BSD Unix TCP/IP Software", AT&T Bell Laboratories, Computing Science Technical Report No. 117, 1985.

[17] S. M. Bellovin, "Security Problems in the TCP/IP Protocol Suite", Computer Communications Review, Vol. 19, No.2, pp. 32-38, pp. 1-4 (reprinted), 1989.

- The call must originate from a privileged TCP port.
- The calling user and machine must be listed in the destination host's list of trusted partners (typically `/etc/hosts.equiv` or in a user's `$HOME/.rhosts` file).
- The caller's name must correspond to its IP address.

The following example shows the content of a `.rhosts` file found in a user's home directory:

```
host1.iyte.edu.tr pars
host2.iyte.edu.tr root
```

where the user trusts `pars@host1.iyte.edu.tr` and `root@host2.iyte.edu.tr`.

The attacker is assumed to have a `trusted_user@trusted_host` pair determined. The attacker will impersonate the trusted during the attack by exploiting the authentication weakness in IP.

4.3.2 Trusted Host Disabling

Morris notes that the segments destined for the target will carry the trusted IP address and an imaginary source port number forming a fake socket where the target's responses will be sent. The important point to note is that the trusted host's TCP will certainly abort this unreferenced connection by generating a RST segment. This situation is illustrated in Figure 4.3.

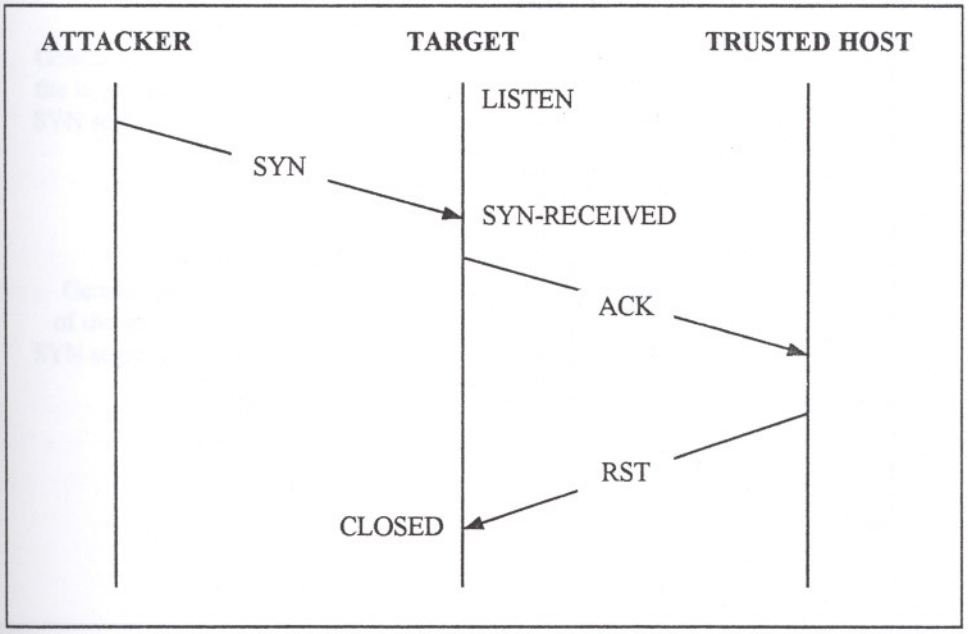


Figure 4.3 Trusted Host Resets the Unreferenced Connection.

[18] W. R. Cheswick, S. M. Bellovin, *Firewalls and Internet Security: Repelling the Wily Hacker*, Professional Computing Series, Addison Wesley, pp. 42-43.

In order to avoid this situation, the attacker must disable this imaginary socket prior to sending any packet to the target. Morris proposes the SYN-flooding attack described above, in order to avoid the trusted host factor.

4.3.3 Establishing a Forged TCP Connection

Morris also notes that, since the target's packets will be send to the disabled trusted host, where they will be ignored, the attacker will not see them. On the other hand, in order to establish a connection to the target, the attacker must successfully complete the TCP 3-way handshake, therefore predict the target's ISN.

It is important for the attacker to understand how the sequence numbers change with respect to time in order to predict them. Morris notes the problem in TCP which employs a predictable clock driven ISN selection, in order to predict the target's sequence numbers.

Bellovin details this process as follows:

In order to learn the current sequence number of the target, the attacker must send a legitimate SYN segment and wait for the target's response. Then, the attacker will send the spoof SYN segment as soon as possible, which will trigger the generation of the target's new ISN (which is sent to the trusted host). This ISN is uniquely determined by the time between the origination of the target's response to the legitimate SYN segment and the receipt at target of the spoof SYN segment. As illustrated in Figure 4.4, this number is precisely the round-trip time between the attacker's and target hosts.

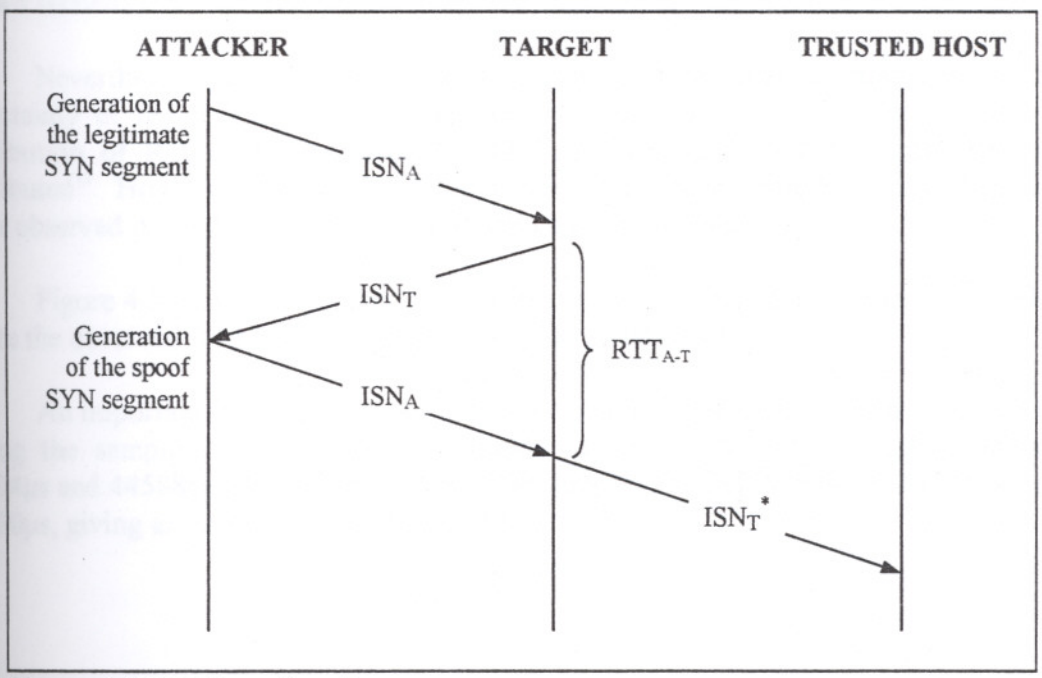


Figure 4.4 ISN Sampling Process.

As a result, if the attacker can accurately measure (and predict) this time it will be possible to predict the target's ISN as,

$$ISN_T^* = ISN_T + RTT_{A\text{-}T} / 4$$

where $RTT_{A\text{-}T}$ is the round-trip time between the attacker's and target hosts (in microseconds).

4.3.4 The Attack

Having a one-way TCP connection to the target established, the attacker will supply the target with the correct data required by the Rsh protocol and with correct sequence numbers.

4.3.5 Impact

The attacker will obtain unauthorized access to the target host and execute some malicious code.

Although the above scenario is based on the Rsh protocol, Morris notes that this attack will work in all situations where a system trusts another system's IP address.

The success of the attack depends on the success in the prediction of $RTT_{A\text{-}T}$. It should be noted that RTT prediction with an error $E < 4\mu s$ may be quite challenging, while not impossible. Bellovin states that if the stability is good (RTT < 10ms), there will be an uncertainty of 2500 (10ms/4μs) in the possible values for ISNs and the attacker will have a near-certainty likelihood of succeeding within a day if each trial takes 5 seconds (~17280 trials).

Nevertheless, several observations showed that an attacker may reduce the uncertainty of the ISNs (hence, the number of trials) by analyzing the probability distribution of RTTs. As noted above RTTs are reported to be roughly Poisson distributed[19]. However, the proposed method is based on empirical probabilities and other observed properties of RTTs, which will be described below.

Figure 4.5 illustrates an example observation consisting of 1000 samples. In this figure the observed RTTs are categorized into 500 μs intervals[20].

An important point about this example (which is not shown in Figure 4.5) is that during the sampling the network was highly unstable. The RTTs ranged between 17104μs and 445885μs. However, 510 of 1000 samples has fallen between 17000μs and 17500μs, giving an empirical probability of 510/1000=0.51.

---

[19] D. L. Mills, "Internet Delay Experiments", RFC 889, p. 10, 1983.

[20] The selection of a 500μs interval length was heuristic. However, it should be noted that the interval length must be chosen regarding two facts: larger intervals will be more vulnerable to noise whereas smaller intervals will miss useful samples.

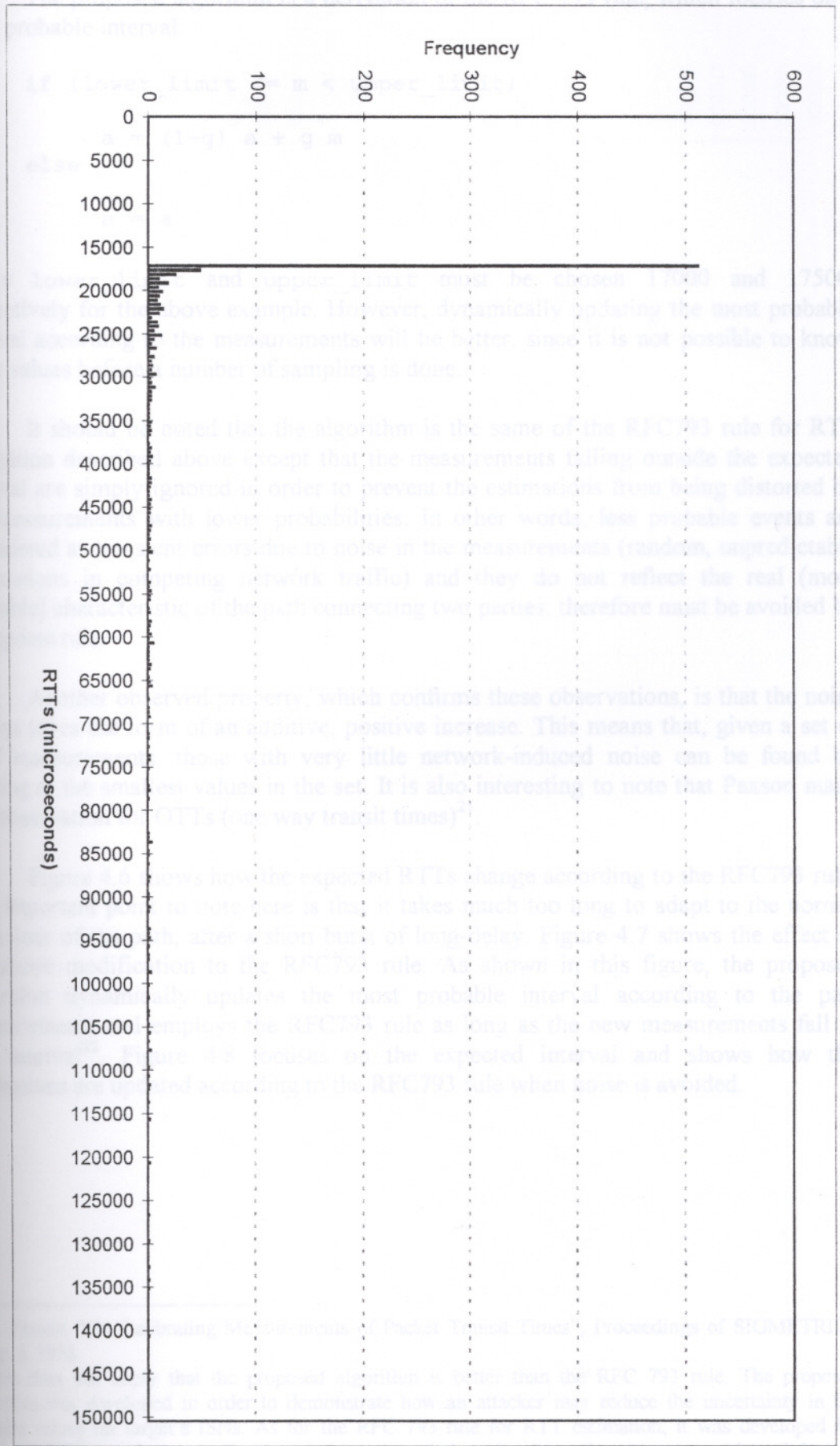


Figure 4.5 Distribution of RTTs over 500 Microseconds Intervals.

The proposed algorithm is a derivation of the RFC 793 rule, which focuses on a most probable interval:

```
if (lower_limit <= m < upper_limit)

      a = (1-g) a + g m
else

      a = a
```

where `lower_limit` and `upper_limit` must be chosen 17000 and 17500, respectively for the above example. However, dynamically updating the most probable interval according to the measurements will be better, since it is not possible to know these values before a number of sampling is done.

It should be noted that the algorithm is the same of the RFC793 rule for RTT estimation described above except that the measurements falling outside the expected interval are simply ignored in order to prevent the estimations from being distorted by the measurements with lower probabilities. In other words, less probable events are considered as transient errors due to noise in the measurements (random, unpredictable fluctuations in competing network traffic) and they do not reflect the real (most probable) characteristic of the path connecting two parties, therefore must be avoided by the update rule.

Another observed property, which confirms these observations, is that the noise always takes the form of an additive, positive increase. This means that, given a set of RTT measurements, those with very little network-induced noise can be found by looking at the smallest values in the set. It is also interesting to note that Paxson made this observation for OTTs (one way transit times)[21].

Figure 4.6 shows how the expected RTTs change according to the RFC793 rule. The important point to note here is that it takes much too long to adapt to the normal behaviour of the path, after a short burst of long-delay. Figure 4.7 shows the effect of the above modification to the RFC793 rule. As shown in this figure, the proposed algorithm dynamically updates the most probable interval according to the past measurements and employs the RFC793 rule as long as the new measurements fall in that interval[22]. Figure 4.8 focuses on the expected interval and shows how the estimations are updated according to the RFC793 rule when noise is avoided.

---

[21] V. Paxson, "On Calibrating Measurements of Packet Transit Times", Proceedings of SIGMETRICS '88, p. 5, 1988.

[22] This does not imply that the proposed algorithm is better than the RFC 793 rule. The proposed algorithm was developed in order to demonstrate how an attacker may reduce the uncertainty in the possible values for target's ISNs. As for the RFC 793 rule for RTT estimation, it was developed for improving TCP's performance. Employing the proposed algorithm for such purposes would result in very serious congestion.

The proposed algorithm is a derivation of the RFC 793 rule, which focuses on a most probable interval:

```
if (lower_limit <= m < upper_limit)

      a = (1-g) a + g m
else

      a = a
```

where `lower_limit` and `upper_limit` must be chosen 17000 and 17500, respectively for the above example. However, dynamically updating the most probable interval according to the measurements will be better, since it is not possible to know these values before a number of sampling is done.

It should be noted that the algorithm is the same of the RFC793 rule for RTT estimation described above except that the measurements falling outside the expected interval are simply ignored in order to prevent the estimations from being distorted by the measurements with lower probabilities. In other words, less probable events are considered as transient errors due to noise in the measurements (random, unpredictable fluctuations in competing network traffic) and they do not reflect the real (most probable) characteristic of the path connecting two parties, therefore must be avoided by the update rule.

Another observed property, which confirms these observations, is that the noise always takes the form of an additive, positive increase. This means that, given a set of RTT measurements, those with very little network-induced noise can be found by looking at the smallest values in the set. It is also interesting to note that Paxson made this observation for OTTs (one way transit times)[21].

Figure 4.6 shows how the expected RTTs change according to the RFC793 rule. The important point to note here is that it takes much too long to adapt to the normal behaviour of the path, after a short burst of long-delay. Figure 4.7 shows the effect of the above modification to the RFC793 rule. As shown in this figure, the proposed algorithm dynamically updates the most probable interval according to the past measurements and employs the RFC793 rule as long as the new measurements fall in that interval[22]. Figure 4.8 focuses on the expected interval and shows how the estimations are updated according to the RFC793 rule when noise is avoided.

---

[21] V. Paxson, "On Calibrating Measurements of Packet Transit Times", Proceedings of SIGMETRICS '88, p. 5, 1988.

[22] This does not imply that the proposed algorithm is better than the RFC 793 rule. The proposed algorithm was developed in order to demonstrate how an attacker may reduce the uncertainty in the possible values for target's ISNs. As for the RFC 793 rule for RTT estimation, it was developed for improving TCP's performance. Employing the proposed algorithm for such purposes would result in very serious congestion.

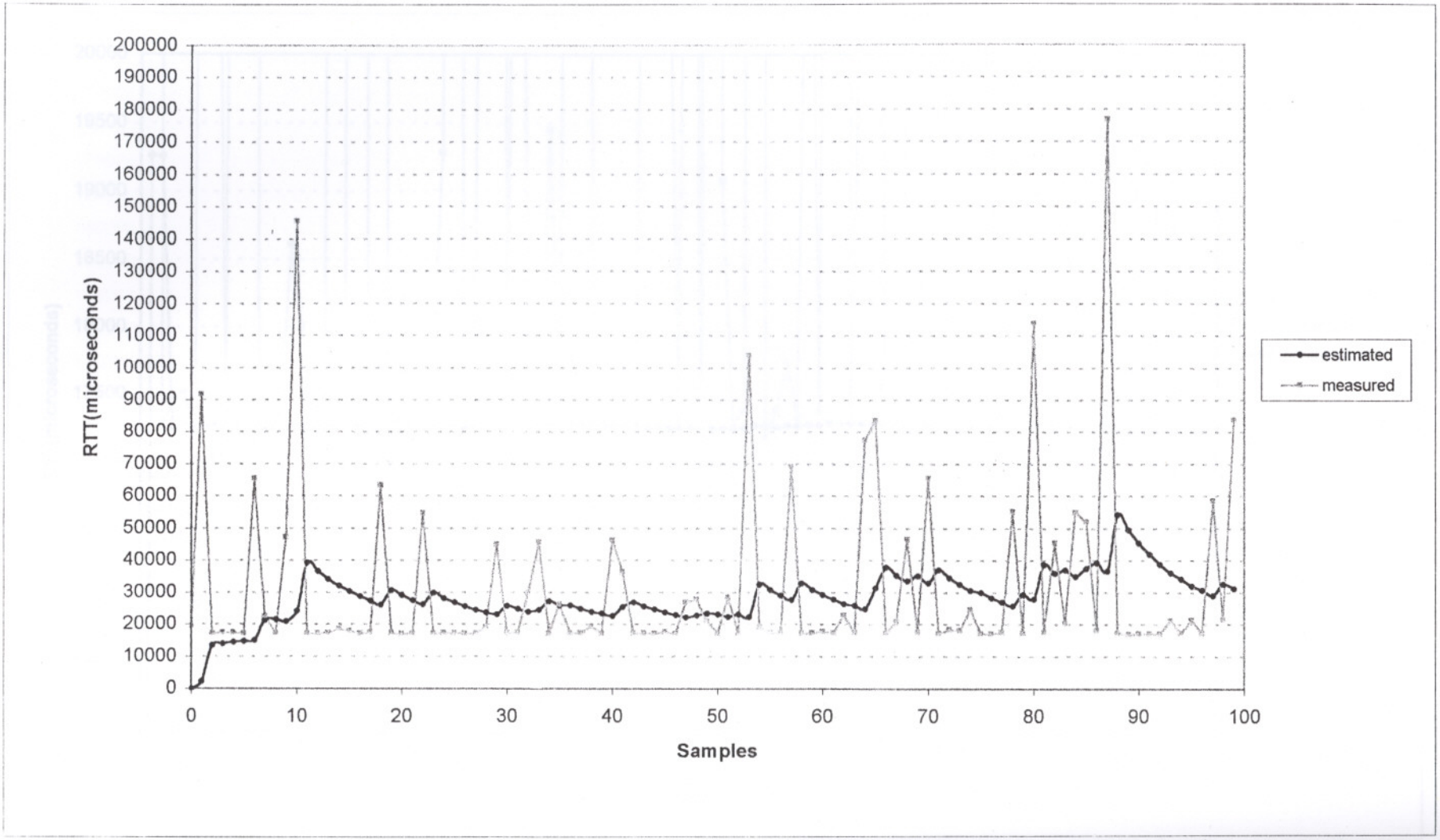


Figure 4.6 RTT Estimation with the RFC 793 Rule.

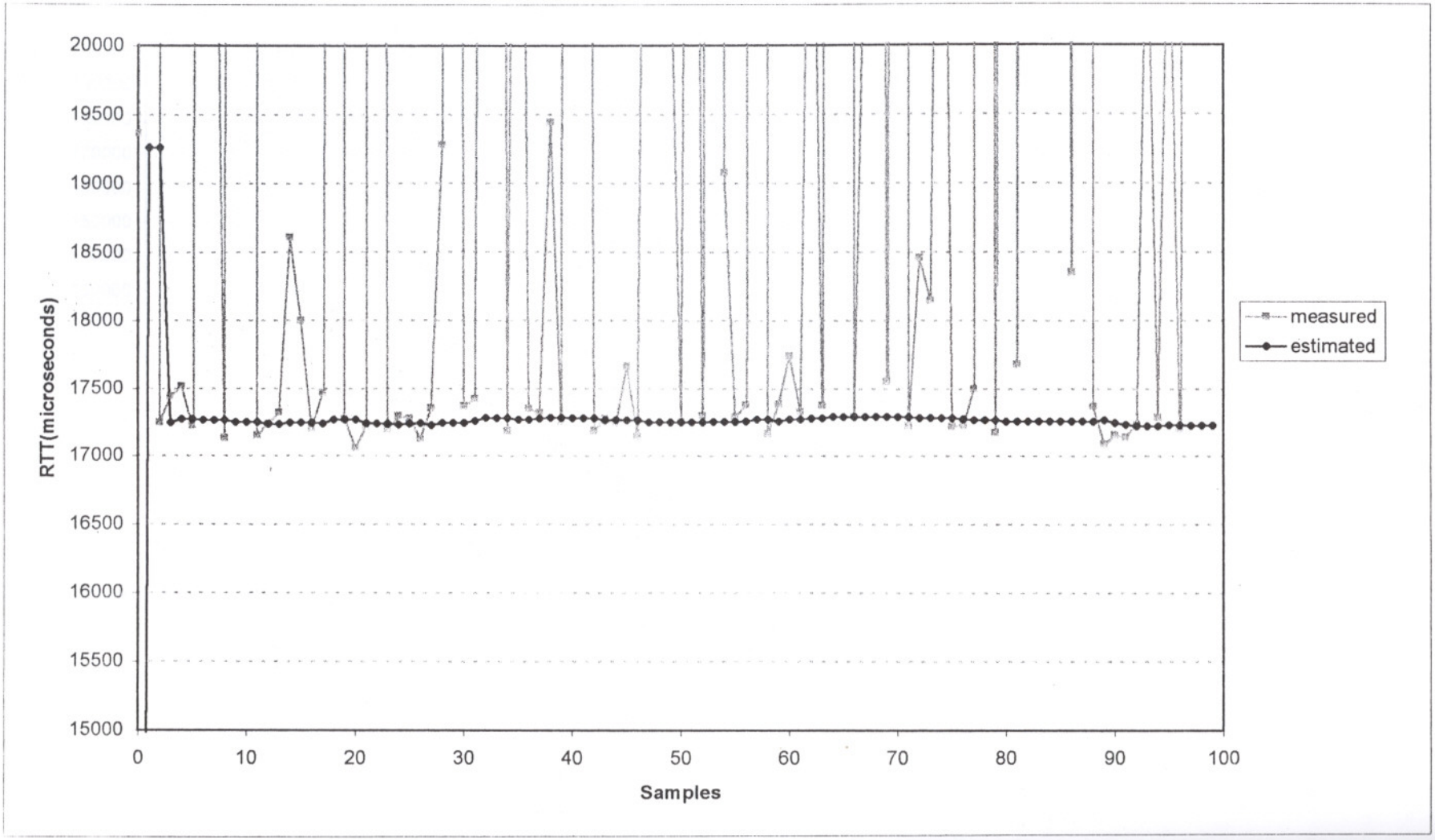


Figure 4.8 RTT Estimation with the Proposed Algorithm (focused on the most probable interval).

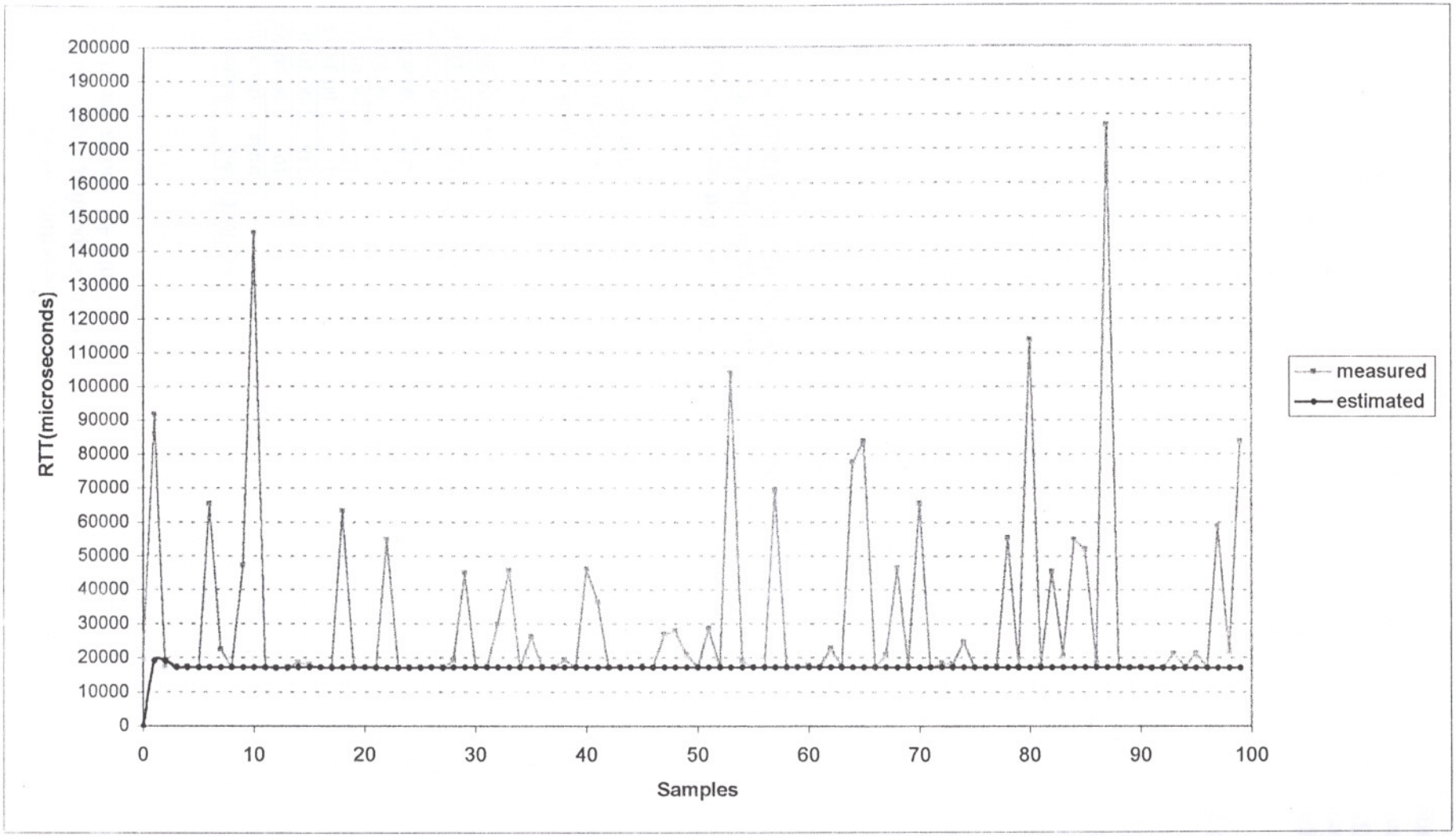


Figure 4.7 RTT Estimation with the Proposed Algorithm.

Observations showed such a RTT estimation algorithm would considerably reduce an attacker's number of trials for succeeding in an IP spoofing attack. Given that each trial takes 5 seconds (as assumed by Bellovin) Table 4.1 shows the results of several RTT estimation attempts.

Table 4.1 RTT Estimation Results.

Destination Host	Distance (hops)	RTT_{MIN} (μs)	RTT_{MAX} (μs)	Matching RTT (μs)	#of trials	Matched in (hh:mm:ss)
sardes.iyte.edu.tr	1	753	854	779	10	00:00:50
bornova.ege.edu.tr	4	13,471	440,136	13,677	248	00:20:40
bornova.ege.edu.tr	4	13,503	194,887	13,751	459	00:38:15
likya.iyte.edu.tr	5	17,052	25,056	17,251	23	00:01:55
likya.iyte.edu.tr	5	16,902	238,830	17,250	215	00:17:55
likya.iyte.edu.tr	5	17,013	530,747	17,196	488	00:40:40
narwhal.cc.metu.edu.tr	5	23,743	289,660	23,846	252	00:21:00
narwhal.cc.metu.edu.tr	5	23,718	208,045	23,877	529	00:44:05
narwhal.cc.metu.edu.tr	5	24,038	669,591	24,252	2,960	04:06:40

The RTTs were calculated by sending and receiving ICMP echo packets and the algorithm was running at the application layer of the probing host. Another point to note is that the routes were static. Dynamic routes would probably increase the number of trials.

Another important point is that it is thus far assumed that no processing takes place for the calculation of the ISN_T^*. In fact, depending on the CPU speed of the attacker's machine and the instruction path followed during the calculation, this will effect the success of the ISN prediction.

Bellovin also notes that whenever a connection request arrives, some processing takes place at the target machine as well. This will have a considerable effect on the actual value of the next ISN, depending again on the CPU speed and the instruction path followed upon a connection request arrival.

As a result, the randomizing effects of the variable instruction paths and CPU speeds are of considerable advantage to the target. However, faster machines are more vulnerable since the variability of the instruction path will take less time and it is not comforting to know that the security of a machine relies on its inconsistency or a network's low quality of service. Clearly, simply following the TCP specification for ISN selection is not secure enough[23].

4.3.6 Real World

Although the faulty implementations of TCP/IP are not discussed in this thesis, the gap between the ISN selection mechanism proposed by TCP and its real-world implementations is interesting. Several observations showed that the TCP specification for ISN selection is loosely implemented in today's Internet. Instead, different operating systems employ different methods some of which, being more or less vulnerable than the original TCP specification.

---

[23] S. M. Bellovin, "Security Problems in the TCP/IP Protocol Suite", Computer Communications Review, Vol. 19, No.2, p. 32-38, p. 3 (reprinted), 1989.

For example, regarding these observations, Microsoft NT Version 4.0 increments its ISN by 1 every milliseconds, which will tolerate the attacker's errors of up to 1 millisecond during the RTT prediction process.

To overcome the security problems caused by predictable ISNs, RFC 1948 proposes a more sophisticated ISN selection scheme consisting of randomizing the ISNs by using a cryptographic hash function[24]. Today, newer implementations such as new Linux kernels, employ the derivations of this method.

However, it should be noted that even pure random ISNs will not defend against sequence number attacks in situations where the attacker sees the target host ISNs. An attacker may achieve this by using one of the following methods:

1. Simple network monitoring: If the attacker is able to monitor the traffic (hence, the sequence numbers) generated by the target, the attacker will not have to predict the ISNs.

2. Using the IP "source routing" option[25]: An attacker may profit from the source routing option in IP, which is mentioned in the previous chapter. In this method the attacker will send to the target a source routed SYN segment carrying the source IP address of a trusted host as described above, however, with fake routing information. Then, if the target host uses the reverse path of the received source route (which is reasonable from the target's point of view, because replies may not reach the trusted host if a different path is followed), then the SYN segment carrying the target's ISN will be sent to the attacker's host in stead of the trusted host. Therefore, the attacker will not have to predict this number.

It should be noted that these methods may be used in all situations where other information contained in the target's packets are of interest.

4.4 TCP Connection Hijacking Attack

Joncheray described how an attacker can obtain the control of an already established connection between a legitimate client and a server[26]. Briefly, the attack consists of desynchronizing the targeted connection, becoming a transparent intermediary and copying the sender's packets to its destination by spoofing the IP address of the sender.

An important point to note is that the attacker is assumed to monitor the target connection.

---

[24] S. M. Bellovin, "Sequence Number Attacks", RFC 1948, pp. 3-4, 1996.
[25] S. M. Bellovin, "Security Problems in the TCP/IP Protocol Suite", Computer Communication Review, vol. 19, no. 2, pp. 32-48, p. 4 (reprinted), 1989.
[26] L. Joncheray, "A Simple Attack Against TCP", 1995.

4.4.1 A Desynchronized State

Joncheray describes a desynchronized state as follows:

Let,

- *SVR_SEQ*: Server's sequence number
- *CLT_SEQ*: Client's sequence number
- *SVR_ACK*: Server's acknowledge number
- *CLT_ACK*: Client's acknowledge number
- *SEG_SEQ*: Sequence number of a given segment
- *SEG_ACK*: Acknowledge number of a given segment
- *SVR_WIND*: Server's window size

Then, the term "desynchronized state" refers to a connection where both sides are in the ESTABLISHED state, no data is being sent and,

$$SVR\_SEQ \neq CLT\_ACK$$
$$CLT\_SEQ \neq SVR\_ACK$$

This state is stable as long as no data is sent. If some data is sent two cases can occur:

1. If $SVR\_ACK < CLT\_SEQ < SVR\_ACK + SVR\_WIND$, the packet is acceptable, the data may be stored for later use (depending on the implementation) but not sent to the user since beginning of the stream (sequence number SVR_ACK) is missing.

2. If $SVR\_ACK > CLT\_SEQ$ or CLT_SEQ $> SVR\_ACK + SVR\_WIND$, the packet is not acceptable and will be dropped. The data will be lost.

In both cases data exchange is not possible even if the state exists.

4.4.2 Creation of a Desynchronized State

Joncheray presents two methods for desynchronizing a TCP connection, which will be described in this section.

4.4.2.1 Early Desynchronization

This method consists of breaking the connection is its early establishment stage on the server and creating a new one with different a sequence number. This process (illustrated in Figure 4.9) is described as follows:

- The attacker waits for the second packet of the TCP 3-way handshake (the packet containing SYN and ACK) of the target connection. This is sent from the server to the client as the acknowledgment of the client's connection request.

- On detection of that packet the attacker sends the server a RST packet followed by a SYN packet with exactly the same parameters (forming the same socket pair) but with a different sequence number[27]. This new sequence number will be referred to as *ATK_SEQ* (sequence number chosen by the attacker).
- The server closes the first connection when it receives the RST packet and sends the acknowledgment of the new connection (containing SYN and ACK) with a new sequence number which will be referred to as *SVR_SEQ'*.

- On the detection of that packet the attacker sends the server an ACK packet and the server switches to the ESTABLISHED state.

- The client has already switched to the ESTABLISHED state when it received the first packet sent by the server (the packet containing SYN and ACK).

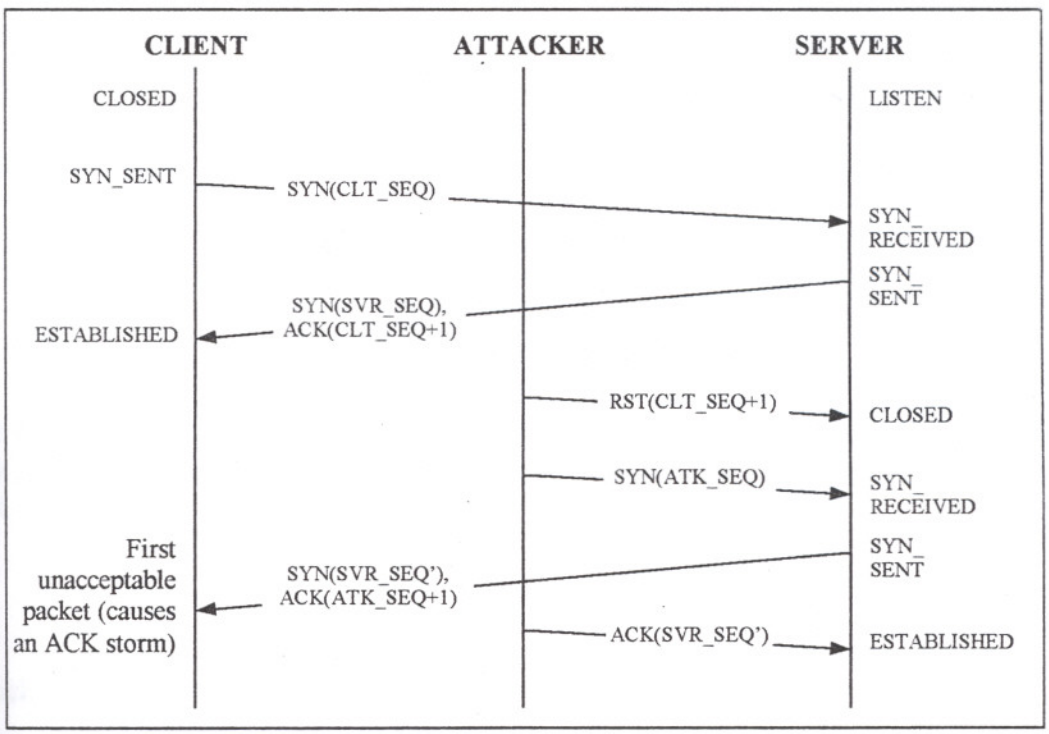


Figure 4.9 Early Desynchronization.

4.4.2.2 Null Data Desynchronization

This method consists of sending a large amount of "null data" to the server and client. Null data refers to data that will not affect anything on the target hosts (server and client) besides changing the TCP acknowledgment number. In this case the application layer protocol used by the server and client must support this kind of data. Joncheray assumes that this protocol is TELNET which provides a NOP (no operation) command.

[27] It should be noted here, a simple form of denial-of-service attack where the attacker breaks the target connection.

4.4.3 The Attack

The attack consists of creating a desynchronized state on both ends of the target TCP connection so that the client and server can not exchange data any longer. The attacker will then create acceptable packets for both ends, claiming every time to be sent from the opposite end. This is explained as follows:

If the target TCP session is in a desynchronized state and the client sends a packet with

$$SEG\_SEQ = CLT\_SEQ$$
$$SEG\_ACK = CLT\_ACK$$

this will not be accepted by the server since $CLT\_SEQ \neq SVR\_ACK$ and will be dropped. The attacker will then send the same packet but change SEG_SEQ and SEG_ACK such that

$$SEG\_SEQ = SVR\_ACK$$
$$SEG\_ACK = SVR\_SEQ$$

which will be accepted by the server.

Similarly, if the server sends a packet with

$$SEG\_SEQ = SVR\_SEQ$$
$$SEG\_ACK = SVR\_ACK$$

this will not be accepted by the client since $SVR\_SEQ \neq CLT\_ACK$ and will be dropped. The attacker will then send the same packet but change SEG_SEQ and SEG_ACK such that

$$SEG\_SEQ = CLT\_ACK$$
$$SEG\_ACK = CLT\_SEQ$$

which will be accepted by the client.

This explains how the attacker becomes a transparent full-duplex intermediary to the client and server.

4.4.4 Impact

The attacker will obtain the control of the connection, then an unauthorized access to the server.

The most important threat posed by this attack is that an attacker may pass sophisticated password authentication schemes such as one-time passwords. In fact, even a perfect password encryption scheme will not prevent the attacker from obtaining an unauthorized access, unless the data stream is encrypted as well. Because, the attacker will not need to know the authentication criteria. He/she will wait until the client supplies the necessary information.

During the tests of the attack, Joncheray used the second type of packets described in Section 4.4.1. However, Joncheray notes a side effect of this kind of packets: "ACK storms". When receiving an unacceptable packet the host acknowledges it by sending the expected sequence number and using its own sequence number. This packet is itself unacceptable to the other end and will generate another acknowledgment packet which in turn will generate an acknowledgment packet etc., creating a supposedly endless loop for each data packet sent. However, Joncheray observed that the ACK storms are not endless. This is related to the fact that these packets do not carry data and they are not retransmitted when lost. Furthermore, ACK storms are reported to be self-regulating: the more loops are created the more congestion and packet loss is experienced.

In order to minimize the duration of the ACK storm loops, the attacker must send the correct acknowledgment of the data packet as soon as it is monitored. However, Joncheray observed that the attacker may miss the data packet because of the network load.

The first type of packets described in Section 4.4.1 have the advantage of not generating ACK storms. On the other hand they may be dangerous (from the attacker's point of view) if the data actually processed. They are also difficult to use with small window connections.

4.5 A Note about UDP

Although this chapter mainly focuses on TCP, a note about UDP (User Datagram Protocol) is worth mentioning here.

UDP provides a procedure for applications to send messages with a minimum protocol mechanism, where delivery and duplicate protection are not guaranteed[28]. The important point to note here is that UDP does not support sequence numbers. Therefore, it is much easier for an attacker to forge UDP traffic since no sequence number guessing is needed[29].

---

[28] J. Postel, "User Datagram Protocol", RFC 768, p. 1, 1980.

[29] W. R. Cheswick, S. M. Bellovin, *Firewalls and Internet Security: Repelling the Wily Hacker*, Professional Computing Series, Addison Wesley, p. 25, 1994.

Chapter 5

ICMP FOR DENIAL-OF-SERVICE ATTACKS

ICMP (Internet Control Message Protocol) is an internet layer protocol used by applications and users for proper Internet operation. However, ICMP poses several threats against the availability of the Internet assets.

This chapter describes the ICMP denial-of-service attacks.

5.1 Background

The original specification for ICMP is RFC792[1]. However, several additional features were defined in the following RFCs such as RFC950, RFC1122 and RFC1812 [2,3,4].

This section provides a brief function description of ICMP and a detailed analysis of several interesting ICMP message types and codes from a security point of view.

5.1.1 ICMP Function Description

The IP is not designed to be absolutely reliable. Therefore a means for reporting errors in datagram processing is needed for administrative and diagnostic purposes. In the Internet the ICMP is generally used for such purposes. ICMP uses the basic support of IP as if it were a higher level protocol (ICMP packets are encapsulated in IP datagrams). However, ICMP is actually an integral part of IP, and is implemented by every IP module.

ICMP messages are sent in several situations. For example, when a datagram cannot reach its destination, when a router does not have the buffering capacity to forward a datagram, and when a router can direct a host to send traffic on a shorter route.

An important point to note is that the purpose of ICMP messages is to provide feedback about problems in the Internet and not to make the IP reliable. There are still no guarantees that a datagram will be delivered or an ICMP message will be returned. Some datagrams may still be undelivered without any report of their loss. The higher level protocols that use IP must implement their own reliability procedures if reliable communication is required.

---

[1] J. Postel, "Internet Control Message Protocol", RFC 792, 1981.
[2] J. Mogul, J. Postel, "Internet Standard Subnetting Procedure", RFC 950, 1985.
[3] R. Braden,"Requirements for Internet Hosts – Communication Layers", RFC 1122, 1989.
[4] F. Baker, "Requirements for IP Version 4 Routers", RFC1812, 1995.

5.1.2 Format of an ICMP Message

An ICMP message consists of the following fields:

- Type (8 bits)
- Code (8 bits)
- Checksum (16 bits)
- A variable length field that differs from one message to another.

There are 15 different values for the type field, which identify the particular ICMP message[5]. Some types of ICMP messages then use different values of the code field to further specify the condition. The checksum field covers the entire ICMP message.

The content of the last field depends on the ICMP message type and code. It contains specific information needed for the operation of different ICMP messages, which will be described below.

5.1.3 Interesting ICMP Messages

As mentioned above, there are 15 different types of ICMP messages. However, in this section, types 0 (echo reply), 3 (destination unreachable), 8 (echo request), 17 (address mask request) and 18 (address mask reply), will be described in detail.

5.1.3.1 ICMP Destination Unreachable Message

This section focuses on the ICMP destination unreachable message codes 0, 1, 2, 3, 4 and 5.

An important point to note about this message codes is that the codes 0, 1, 4 and 5 may be received from a router and codes 2 and 3 may be received from a host[6]. The situations in which the ICMP messages with these codes are sent, are described below.

For all destination unreachable codes, the ICMP message also includes the original IP header and 64 bits of the original datagram's data. This information is used by the host receiving the message to match the message to the appropriate process[7].

5.1.3.1.1 Codes 0, 1, 4 and 5

The ICMP destination unreachable message is sent by a router in response to a datagram which it cannot forward because the destination is unreachable or down.

---

[5] W. R. Stevens, *TCP/IP Illustrated Volume 1: The Protocols,* Professional Computing Series, Addison Wesley, p. 71, 1994.

[6] J. Postel, "Internet Control Message Protocol", RFC 792, p. 5, 1981.

[7] ibid, p. 4.

A router MUST generate ICMP destination unreachable messages with code[8]:

- 0 (network unreachable): whenever intermediate router is unable to forward a datagram, as its routing data-base gives no next hop for the datagram, or all paths were down.

- 1 (host unreachable): whenever the destination network is reachable, but a router on that network is unable to reach the destination host.

- 4 (fragmentation needed and DF set): whenever a datagram must be fragmented by a router yet the Don't Fragment flag is on.

- 5 (source route failed): whenever a next hop in the source route specified by the sender is not on a directly connected network.

5.1.3.1.2 Codes 2 and 3

A host SHOULD generate ICMP destination unreachable messages with code[9]:

- 2 (protocol unreachable), when the designated protocol is not supported

- 3 (port unreachable), when the designated transport protocol (e.g., UDP) is unable to demultiplex the datagram but has no protocol mechanism to inform the sender

5.1.3.1.3 Processing of ICMP Destination Unreachable Messages

A destination unreachable message that is received MUST be reported to the transport layer. The transport layer SHOULD use the information appropriately. UDP MUST pass all ICMP error messages that it receives from the IP layer to the application layer. TCP that has its own mechanism for notifying the sender that a port is unreachable (with a RST segment) MUST nevertheless accept a "port unreachable" message for the same purpose[10].

A destination unreachable message that is received with code 0, 1, or 5 may result from a routing transient and MUST therefore be interpreted as only a hint, not proof, that the specified destination is unreachable[11].

On the other hand destination unreachable codes 2 and 4, are hard error conditions, so a TCP SHOULD abort the connection when this messages are received[12].

---

[8] F. Baker, "Requirements for IP Version 4 Routers", RFC1812, p. 81, 1995.
[9] R. Braden,"Requirements for Internet Hosts – Communication Layers", RFC 1122, p. 40, 1989.
[10] ibid, p. 40.
[11] ibid, p. 40.
[12] ibid, p. 104.

5.1.3.2 ICMP Address Mask Request and Reply Messages

The ICMP address mask request is intended for a diskless system to obtain its subnet mask at bootstrap time. The requesting system broadcasts its ICMP request. The ICMP address mask request and reply messages contains the following fields as well:

- Identifier (16 bits)
- Sequence number (16 bits)
- Address mask (32 bits)

The identifier and sequence number fields are used to aid in matching requests and replies, and the address mask field carries the requested 32-bit mask. However, RFC950 states that these fields can be ignored, since there is only one possible address mask for a subnet, and there is no need to match requests with replies[13].

Regarding RFC1122, a host MAY support sending ICMP address mask request(s) and receiving ICMP address mask reply(s). When the use of address mask messages is enabled, then:

1. When it initializes, the host MUST broadcast an address mask request message on the connected network corresponding to the IP address. It MUST retransmit this message a small number of times if it does not receive an immediate address mask reply.

2. Until it has received an address mask reply, the host SHOULD assume a mask appropriate for the address class of the IP address, i.e., assume that the connected network is not subnetted.

3. The first address mask reply message received MUST be used to set the address mask corresponding to the particular local IP address. This is true even if the first address mask reply message is "unsolicited", in which case it will have been broadcast and may arrive after the host has ceased to retransmit address mask requests. Once the mask has been set by an address mask reply, later address mask reply messages MUST be (silently) ignored.

Conversely, if address mask messages are disabled, then no ICMP address mask requests will be sent, and any ICMP address mask replies received for that local IP address MUST be (silently) ignored[14].

A system MUST NOT send an address mask reply unless it is an "authoritative agent" for address masks. An authoritative agent may be a host or a router, but it MUST be explicitly configured as an address mask agent. Receiving an address mask via an address mask reply does not give the receiver authority and MUST NOT be used as the basis for issuing address mask replies[15].

Getting no reply to its address mask request messages, a host will assume there is no agent and use an unsubnetted mask, but the agent may be only temporarily

[13] J. Mogul, J. Postel, "Internet Standard Subnetting Procedure", RFC 950, p. 11, 1985.
[14] R. Braden,"Requirements for Internet Hosts – Communication Layers", RFC 1122, p. 45, 1989.
[15] ibid, p. 46.

unreachable. An agent will broadcast an unsolicited address mask reply whenever it initializes, in order to update the masks of all hosts that have initialized in the meantime[16].

5.1.3.3 ICMP Echo Request and Reply Messages

ICMP echo request and reply messages are used for diagnostic purposes, for example in order to control whether a host or router is alive or not. These messages also include the following fields:

- identifier (16 bits)
- sequence number (16 bits)
- a variable length field of data

The identifier and sequence number fields are used to aid in matching echos and replies[17]. Data received in an ICMP echo request MUST be entirely included in the resulting echo reply[18,19].

Every host and router MUST implement an ICMP echo server function that receives echo requests and sends corresponding echo replies[20,21].

A router SHOULD have a configuration option that, if enabled, causes the router to silently ignore all ICMP echo requests; if provided, this option MUST default to allowing responses[22].

An ICMP echo request destined for an IP broadcast or IP multicast address MAY be silently discarded by hosts. This neutral provision results from a passionate debate between those who feel that ICMP echo to a broadcast address provides a valuable diagnostic capability and those who feel that misuse of this feature can too easily create packet storms[23].

A host SHOULD also implement an application level interface for sending an echo request and receiving echo reply, for diagnostic purposes[24]. An application called "ping" is generally used in the Internet as an interface to ICMP echo and reply messages[25].

5.2 ICMP Echo Request Flooding Attack

This section describes a simple form of denial of service attack using the ICMP echo request packets.

---

[16] R. Braden,"Requirements for Internet Hosts – Communication Layers", RFC 1122, p. 46, 1989.
[17] J. Postel, "Internet Control Message Protocol", RFC 792, p. 14, 1981.
[18] ibid, p. 43.
[19] F. Baker, "Requirements for IP Version 4 Routers", RFC1812, p. 59, 1995.
[20] R. Braden,"Requirements for Internet Hosts – Communication Layers", RFC 1122, p. 42, 1989.
[21] F. Baker, "Requirements for IP Version 4 Routers", RFC1812, p. 58, 1995.
[22] ibid, p. 59.
[23] R. Braden,"Requirements for Internet Hosts – Communication Layers", RFC 1122, pp. 42-43, 1989.
[24] ibid, p. 42.
[25] M. Muus, "Ping.c", 1983.

5.2.1 Description

The ICMP echo flooding is a simple attack that consists of generating echo request packets destined for a target machine, as fast as possible in order to cause CPU resources of the target machine to be consumed and reducing the network throughput.

The attacker will use forged source IP addresses for the echo request packets he/she is generating, in order not to be located.

5.2.2 Impact

During the attack, the targeted machine will attempt to process all incoming echo requests therefore will run out of the CPU cycles needed for its normal operation.

The success of the attack depends on the interarrival times of the echo request packets received by the target and its CPU power. Therefore, generally, first hope routers or directly connected hosts will be targeted. It should be noted that if an attacker launches an echo flooding attack destined for a host outside its network, the first hope router (attempting to forward all ICMP echo request packets) will first run out of CPU cycles instead of the targeted host. In this case all internal hosts will suffer from degraded network performance in their outside Internet connection (including the attacker's host).

5.3 The "Smurf" Attack

The "smurf" attack is named after its exploit program. This is a highly parallelled version of the ICMP echo request flooding attack described above, using echo replies for flooding the target, in stead of echo requests.

5.3.1 Description

In the smurf attacks, an attacker sends a large amount of ICMP echo request traffic at broadcast addresses, all of it having a spoofed source address of a target host or router. If the routers delivering traffic to those addresses performs the forwarding of broadcast ICMP echo requests, most hosts will take the ICMP echo request and will reply to it with an echo reply each, multiplying the traffic by the number of hosts responding. On a multi-access broadcast network, there can potentially be hundreds of machines to reply to each packet[26].

As described in Chapter 3, a router MUST forward net-directed broadcasts by default and as described above, a host MAY discard ICMP echo requests destined for an IP broadcast or IP multicast address (which implies that a host MAY allow those packets). The "smurf" attack expoits these requirements.

[26] Computer Emergency Response Team, " "smurf" IP Denial-of-Service Attacks", CERT Advisory: CA 98-01, September 1998.

5.3.2 Impact

Both the intermediary routers and the target host may suffer degraded network performance both on their internal networks or on their connection to the Internet. Performance may be degraded to the point that the network cannot be used[27].

5.4 Address Mask Forgery Attack

Bellovin first described the address mask forgery attack in April 1989, seven months before the host requirements RFC (RFC1122) was written[28]. Although the RFC1122 specifies further conditions for the implementation of address mask messages, the attack may be still workable.

5.4.1 Definition

Address mask forgery is a more global denial-of-service attack launched by sending a fake ICMP address mask reply message to a target host implementing ICMP address mask messages. The attacker does not have to know or predict the contents of the identifier and sequence number fields of the request, because the matching of requests and replies is not forced, therefore probably not implemented.

As noted above a system MUST NOT send an address mask reply unless it is an authoritative agent for address masks. Furthermore a host will broadcast its address mask request only when it initialize and only the first address mask reply message received will used to set the address mask. Although these specifications are of considerable advantage to the target, the attacker may use the source IP address of an authoritative agent (if needed) and send its fake reply at the appropriate time by monitoring whether the target host is alive or rebooting (for example, for some administrative purposes). The attacker will also have to send its reply before the authoritative agent. However, the authoritative agent can be temporarily down in the meantime.

5.4.2 Impact

All communications with the target host will be blocked[29].

5.5 Destination Unreachable Message Forgery Attack

This section describes another ICMP based denial-of service attack described by Bellovin[30].

---

[27] Computer Emergency Response Team, " "smurf" IP Denial-of-Service Attacks", CERT Advisory: CA 98-01, September 1998.
[28] S. M. Bellovin, "Security Problems in the TCP/IP Protocol Suite", Computer Communications Review, Vol. 19, No.2, pp. 32-38, p. 7 (reprinted), 1989.
[29] ibid, p. 7.
[30] ibid, pp. 6-7.

5.5.1 Description

This attack consists of sending a spoofed ICMP destination unreachable packet, claiming to be sent from a server host carrying out a TCP connection with a target client.

Generally, the attacker will choose destination unreachable codes 2 and 4, since they are considered as are hard error conditions, and a TCP receiving these messages SHOULD abort the referred connection, as mentioned above.

As noted above, destination unreachable messages include the original IP header and 64 bits of the original datagram's data to allow the host to match the message to the appropriate process. The referred 64 bits contain the source and destination port numbers and the sequence number for TCP. However, since the port numbers are enough to match the message to a given process, the sequence number field is generally ignored (and RFCs are not clear enough on that subject). Therefore, the attacker must only know the source and destination port numbers.

The attacker will generally need to be able to monitor the targeted connection in order to supply the correct numbers, however a brute-force version of the attack (e.g., trying all possible port combinations) is also workable[31].

5.5.2 Impact

The targeted connection will be broken.

[31] Cowzilla, Pixel Dreamer, "Puke.c", 1996.

Chapter 6

A WEAKNESS IN THE DNS ARCHITECTURE

DNS (Domain Name System) is a support service needed for convenient Internet access. DNS resides at the application layer, neverthless it merits much more attention then any other application layer protocol since it serves as an interface to IP.

However, DNS may cause unauthorized access. This chapter focuses on this weakness.

6.1 Background

The DNS is a distributed database system used to map host names to IP addresses and vice versa. The term "distributed" is used because no single site on the Internet maintains all the information. RFC 1034 specifies the concepts and facilities provided by DNS and RFC 1035 details the implementation and specification[1,2].

Stevens explains the DNS operation as follows[3]:

Applications and users access to DNS through a resolver. On Unix hosts the resolver is accessed primarily through two library functions: `gethostbyname()` and `gethostbyaddr()`. The first takes a host name and returns an IP address, and the second takes an IP address and looks up a hostname. Resolvers generally communicate with name servers via UDP.

The DNS name space is hierarchical. Figure 6.1 shows this hierarchical organization.

Every DNS node has a "label" of up to 63 characters. The root of the tree is a special node with a null label. The "domain name" of any node in the tree is the list of labels, starting at that node, working up to the root, using a period (dot) to separate the labels (for example "`arf.iyte.edu.tr`" shown in Figure 6.1). The top-level domain `arpa` is a special domain used for address-to-name mappings which will be described below.

One important feature of the DNS is the delegation of responsability within the DNS. No single entity manages manages every label in the tree. In stead, one entity which is NIC maintains a poriton of the tree (the top-level domains) and delegates responsibility to others for specific zones.

A "zone" is a subtree of the DNS tree that is administrated separately (for example `iyte.edu.tr`). Once the authority for a zone is delegated, it is up to person responsible for the zone to provide name server(s) for that zone. Whenever a new host

---

[1] P. V. Mockapetris, "Domain Names – Concepts and Facilities", RFC 1034, 1987.
[2] P. V. Mockapetris, "Domain Names – Implementations and Specifications", RFC 1035, 1987.
[3] W. Richard Stevens, *TCP/IP Illustrated Volume 1: The Protocols,* Professional Computing Series, Addison Wesley, p. 187-208, 1994.

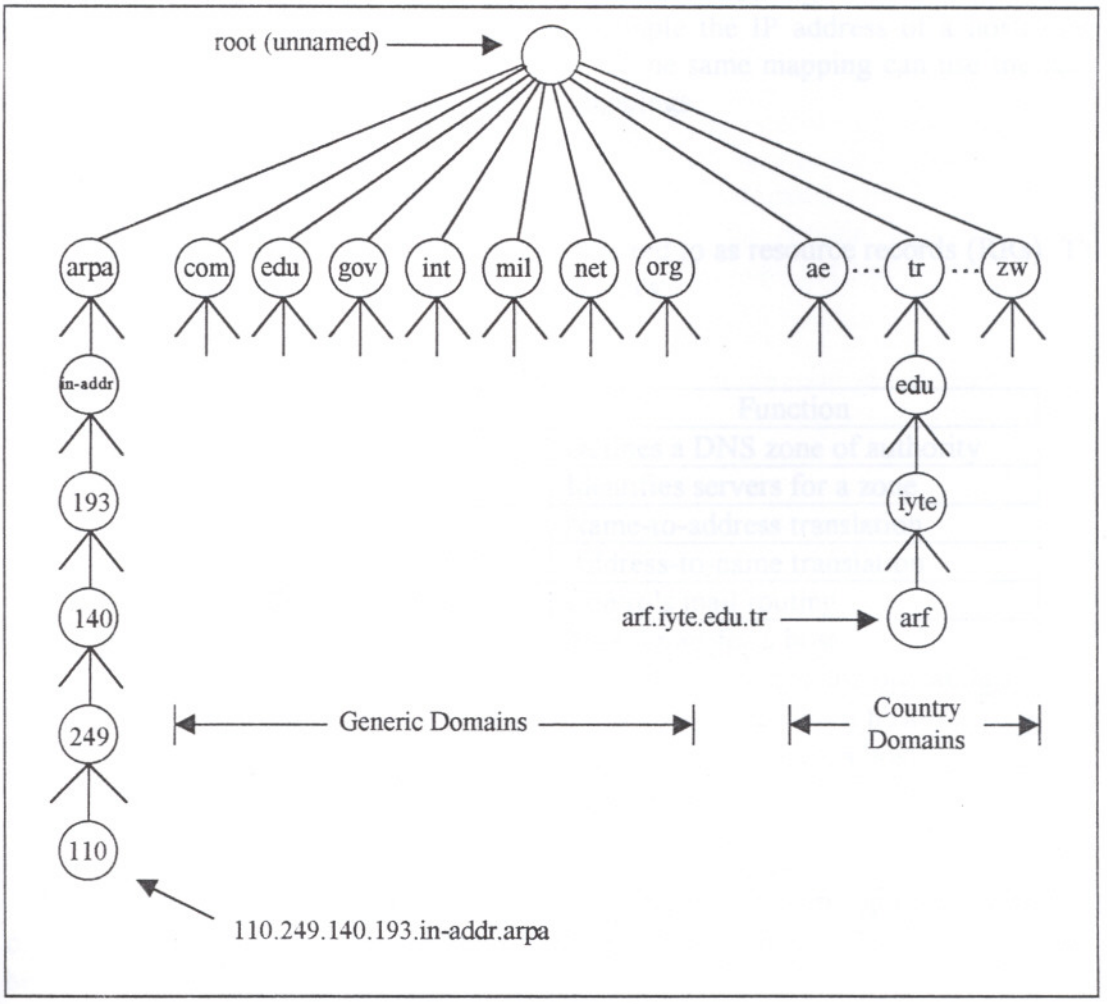


Figure 6.1 The Hierarchical Structure of the DNS.

is installed in a zone, the DNS administrator of the zone allocates a name and IP address for this host and enters these into the name server's database.

A name server is said to have authority for one or multiple zones. The person responsible for a zone must provide a "primary" name server for that zone and one or more "secondary" name servers. The primary and secondaries must be independent and redundant servers so that availability of the name service for the zone is not affected by a single point of failure. The main difference between a primary and secondary is that the primary loads all the information for the zone from disk files, while a secondary obtains the information from its primary. This operation is called a "zone transfer". Zone transfers are made via TCP.

When a name server does not contain the information requested, it must contact another name server, however, not every server know how to contact every other name servers. In stead, every name server must know how to contact the "root" name servers. The root servers know the name and address of each authoritative name server for all secondary-level domains. This implies an iterative process: the requesting server contacts a root server, the root server tells the requesting server to contact another server, etc.

Another fundamental property of the DNS is caching. When, a name server receives information about a mapping (for example the IP address of a hostname) it caches that information so that a later query for the same mapping can use the cached result. This increases the efficiency of DNS mappings.

6.1.1 Resource Records

The entries in the DNS database are referred to as resource records (RRs). Table 6.1 gives most common DNS resource record types.

Table 6.1 DNS Resource Records.

Type	Name	Function
SOA	Start of Authority	Defines a DNS zone of authority
NS	Name Server	Identifies servers for a zone.
A	Address	Name-to-address translation
PTR	Pointer	Address-to-name translation
MX	Mail Exchanger	Controls mail routing
CNAME	Canonical Name	Nicknames for a host
HINFO	Host Info	Identifies hardware and operating s.
RP	Responsible Person	Technical contact for a host
WKS	Well known services	Services provided by a host
TXT	Text	Comments

The CNAME, HINFO, RP, WKS and TXT records are optional, whereas the others are required. An example DNS database for the zone `iyte.edu.tr` is given below:

```
$ORIGIN iyte.edu.tr.

iyte.edu.tr IN    SOA   ns.iyte.edu.tr. admin.host1.iyte.edu.tr. (
            99041001 ; Serial
            3600     ; Refresh
            600      ; Retry
            3600000  ; Expire
            86400 )  ; Minimum time-to-live

            IN    NS    ns
            IN    NS    ns.iyte.edu.tr.

ns              IN      A     193.140.249.1
host1           IN      A     193.140.249.2
host2           IN      A     193.140.249.3
host3           IN      A     193.140.249.4
```

It should be noted that a `SOA` record also contains the serial number of the data (in the yymmdd## format), and several additional values (in seconds) such as the refresh, retry and expire values for secondary servers and a minimum time-to-live value used for cache expiry. The IN field indicates that the records belong to the Internet domain (the DNS is capable of storing information about many different types of networks).

6.1.2 The Inverse Mapping Tree

In the Internet the mapping of IP addresses to names (this is called "inverse mapping" as opposed to the one explained above which is called "forward mapping") is also generally required. However, normally, there is no way to achieve this other than starting at the root of the tree and trying every top-level domain, which is not feasable. This is due to the fact that the IP addresses do not contain information about organizational structure of the Internet. Therefore, clients can not know which server to query.

In stead, inverse mappings are implemented by a separate parallel tree: when an organization joins to the Internet, it obtains an authority for a portion of the DNS `in-addr.arpa` name space corresponding to their IP address on the Internet as well an authority for a portion of the name space. The level of the DNS tree beneath `in-addr.arpa` name space must be the first byte of the IP address, the next level is the second byte of the IP address, etc. For example, the DNS name for a an IP address of `193.140.249.110` is `110.249.140.193.in-addr.arpa`, which is shown in Figure 6.1.

An example inverse mapping database for the domain `249.140.193.in-addr.arpa` is given below (the `SOA` and `NS` records are ommitted):

```
$ORIGIN     249.140.193.in-addr.arpa
1      IN     PTR      ns
2      IN     PTR      host1
3      IN     PTR      host2
4      IN     PTR      host3
```

Generally, the inverse mapping database will reside on the same machine as the corresponding forward mapping database.

6.2 Hostname Spoofing Attack

Bellovin described a name authentication vulnerability due to the distinction of the two trees of DNS[4]. This section covers this vulnerability and how it can be exploited by an attacker.

6.2.1 Description

First, it should be noted that there exist situations (these will be described in the following sections) where host name based authentication is employed.

Table 6.2 examplifies a situation where a target host (`target.iyte.edu.tr`) is assumed to trust the name `trusted.iyte.edu.tr`. The attacker resides at `attacker.any.com.tr` [5].

---

[4] S. M. Bellovin, "Using the Domain System for System Break-ins", Proceedings of the 5th Usenix UNIX Security Symposium, 1995.

[5] The attack scenario is the same of the one imagined by Bellovin with names and addresses changed.

Table 6.2 DNS Attack Scenario.

Name	IP Address	Description
`target.iyte.edu.tr`	`193.140.249.25`	The target host
`trusted.iyte.edu.tr`	`193.140.249.110`	The trusted host
`attacker.any.com.tr`	`153.101.34.78`	The attacker's host

Bellovin assumes that the attacker has complete access (read/write) to the portion of the inverse mapping tree containing his/her `PTR` record (this implies that the attacker must have a corresponding name server, compromised). The attacker will change the corresponding inverse mapping record for 153.101.34.78 from the correct `attacker.any.com.tr` to `trusted.iyte.edu.tr`.

Then, Bellovin notes the following weakness in the DNS architecture: when the attacker attempts to connect to the target host, the target host will try to validate the name of the calling machine. It will do that by calling `gethostbyaddr()` and passing it the address `153.101.34.78`. In the DNS, that call translates to a name server query for the record associated with `78.34.101.153.in-addr.arpa`, which will retrieve the fake `PTR` record set by the attacker. As a result, the target server will believe that a trusted host is connecting.

6.2.2 Impact

The attacker will obtain unautorized access to the target host.

An important point to note is that this vulnerability is due to the distinction of the two trees of DNS. Therefore, Bellovin suggests cross-checking of the two trees in order to defeat this kind of attacks.

In an other version of this attack, the attacker contaminates the target's cache of DNS responses prior to initiating the call. In this situation, when the target does the cross-check, it appears to succeed, and the attacker gains access. A variation of this attack involves flooding of the target's DNS server with fake responses, thereby confusing it. However, Bellovin notes that attempts to contaminate the cache of a primary or secondary server for a domain will not work. The standard name servers will reject updates to zones for which they are authoritative. In this case cross-checking will defeat the attack. On the other hand, caching only servers are vulnerable. Furthermore, if a host trust another host not named in a local zone, its authoritative name server can not protect it.

Chapter 7

DISCLOSURE OF INFORMATION

This chapter concerns the secrecy of Internet related data. Information sharing is vital for convenient Internet operation, however this information may be valuable for the attackers as well.

This chapter focuses on this issue and describes several common information leakage points found in today's Internet.

7.1 Intrusively Network Monitoring[1]

Internet is an interconnection of packet-switched computer communications network. Packet switching provides a means for multiple access, time-sharing and prevents the monopolization of the network resources.

However, on a multiple access network, it is possible for an attacker to capture the bits belonging to a communication between two or more parties. This situation poses a serious security risk against the secrecy of the information in transit in the Internet. All protocol information contained in the four layers of TCP/IP suite as well as application data is available to an attacker monitoring the network activity. Important examples for protocol data would be: IP addresses, port numbers and sequence numbers. This information can be used when launching IP spoofing, session hi-jacking, and several denial-of-service attacks as described in the previous chapters. The application data may contain confidential information such as user names and passwords. This information can be used later for obtaining unauthorized access by impersonating legitimate users, which is referred to as a "replay" attack.

The only known defense against network monitoring attacks is cryptology, which is beyond the scope of this thesis.

7.2 Finger

Finger is a simple user information lookup service using the Finger User Information Protocol, which is specified in RFC1288[2].

Finger discloses information about users: login names, full names, the time the user last logged in, read mail and the host he/she connected from, the idle time, etc. Farmer and Venema state that Finger is the one of the most dangerous services, because it is useful for an attacker to investigate a potential target[3].

The output of an example finger command in the form `finger user@host` is given below:

---

[1] Network monitoring can also serve legitimate purposes such network management, fault detection, security etc. Therefore is used the term "intrusively".
[2] D. Zimmerman, "The Finger User Information Protocol", RFC 1288, 1991.
[3] D. Farmer, W. Venema, "Improving the Security of Your Site by Breaking into It", 1993.

```
Login: ugur                             Name: Ugur UNAL
Directory: /home/ugur                   Shell: /bin/bash
Last login Tue Mar 23 18:31 (GMT) on ttyp2 from muh-pc02.iyte.ed
Mail last read Tue Mar 23 18:32 1999 (GMT)
No Plan.
```

Furthermore, Finger provides a list the users currently logged on a system. An example output of a `finger @host` command is as follows:

```
Login       Name                TTY Idle    When      Office
nyildiz  Nazan Yildiz            p0      Wed 10:11
corez    Mehmet Ali Corez        p1   11 Wed 10:03
omer     Omer Zenciroglu        *p3      Wed 10:19
bakiyev  Asir Muhammet Bakiye   *p4    4 Wed 09:47
mirat    Mirat Satoglu           p5   43 Tue 14:07
aykan    Aykan Candemir         *p6   10 Wed 09:45
gyegin   Gultekin Yegin         *p7      Wed 10:23
ersoy    Banu Ersoy             *p8      Wed 10:22
ardac    Arda Kaan Cakir         p9    9 Wed 10:11
ecakmak  Elif Cakmak             pb   53 Wed 09:24
pabuscu  Harun Pabuscu           pa    7 Wed 10:11
sagnak   Hasan Sagnak            pc      Wed 10:17
alabacak Burak Alabacak          pd      Wed 10:12
turano   Orhan Turan            *pf   17 Wed 10:03
cetin    Sadik Cetin            *q0    4 Wed 10:17
obali    Murat Obali            *q1      Wed 10:19
mirat    Mirat Satoglu           q2   11 Wed 09:55
aydinf   Fatih Aydin            *q3      Wed 10:15
mirat    Mirat Satoglu           q4    7 Wed 09:56
sarac    Omer Saracoglu          q7      Wed 10:17
turhan   Cagdas Uysal Turhan    *q6      Wed 10:20
emret    Emre Taskaya            q8      Wed 10:20
dalkilic M. Emin Dalkilic       *q9      Wed 10:21
erciyes  Kayhan Erciyes          qa    2 Wed 10:22
muhsin   Muhsin Akoz            *qb      Wed 10:22
```

As shown in the above example, during the working hours, the Finger service may reveal information about a considerable number of users.

From an attacker's point of view, this information is useful for tracking conversations in progress and see where someone's attention is focused[4].

The information contained in the user lists is usually used by the attackers during the password guessing processes. In this case a user who is not aware of security precautions (for example, a user who has set his/her name for login password for the sake of simplicity) may compromise the security of the host.

The idle time (in minutes) may be used to determine if a user's attention is focused on his/her session. Regarding this information, an attacker can determine the correct time to launch an attack.

[4] D. Zimmerman, "The Finger User Information Protocol", RFC 1288, p. 8,1991.

The user's host name can be used to determine a trust relation. If the target host is known to employ name or address based authentication, this will tell an attacker that the returned user and the machine he/she is connected from is trusted. This information can also be used to break the user's connection by sending an forged ICMP destination unreachable message as described in Chapter 5. In this case, only the user's port number will be unknown to the attacker, which is subject to brute-force.

7.3 DNS

The use of the DNS is vital for convenient Internet access. On the other hand, DNS poses a risk against the secrecy of site related information.

DNS provides sensitive information about an Internet site: host names and addresses, organizational structure etc[5]. This situation may pose a number of possible impacts depending on the attackers' purposes. For example, entries of MX or NS types (which are described in Chapter 6) disclose information about the names and addresses of mail and name servers. These hosts will generally provide other services known to be vulnerable which can be further analyzed by "port scanning" as discussed in the next section.

Furthermore, the HINFO record provides information about hardware and operating systems running on the potential target hosts[6]. This information can be used to exploit several implementation specific vulnerabilities (for example, the ISN generation mechanism employed or errors found in a supported application).

7.4 Port Scanning

Port scanners are useful diagnostic tools used for determining if a given service is available. However, the information provided by port scanners may be of considerable advantage to an attacker as well.

The most common method for scanning TCP ports, is sending SYN segments destined for the ports of the target host. Then regarding whether the destination service is listening or non-existing, the target host's TCP will reply with a segment carrying SYN or RST flags respectively for each segment received. In some situations, TCP ports can be scanned by using the FIN flag as well. However, this depends on a fault in the implmentation of TCP, therefore not included here[7].

UDP ports can be scanned by sending UDP datagrams. An ICMP destination unreachable message will be received if the port is not listening[8].

The attacker may use this information to determine the availability of vulnerable services, or the degree of protection provided by the access control methods implemented by the target. For example, a well configured packet filtering router will

---

[5] W. R. Cheswick, S. M. Bellovin, *Firewalls and Internet Security: Repelling the Wily Hacker*, Professional Computing Series, pp. 28-29.

[6] Anonymous, *Maximum Security: A Hacker's Guide to Protecting Your Site and Network*, Macmillan Computer Publishing, URL: ftp://arf.iyte.edu.tr/pub/InternetSecurity/fm/fm.htm, 1998.

[7] U. Maimon, "Port Scanning without the SYN Flag", Phrack Magazin, Volume 7, Issue 49, File 15.

[8] ibid.

not pass the untrusted SYN segments through, as opposed to TCP wrappers. This will be explained in the next chapter.

7.5 ISN Sampling

As described in Chapter 4, prior to launching an IP spoofing attack, an attacker has to predict the ISN that will be chosen by the target. However, this requires sampling of the target's ISNs and RTTs between the attacker's and target hosts.

ISN sampling is done by sending legitimate SYN segments and receiving their replies. RTT samples may be collected during the ISN sampling process, however using ICMP echo request and replies is also possible.

7.6 RPC Portmapper

The RPC portmapper discloses information about RPC based protocols. Below is an example output of the `rpcinfo -p host` command:

```
program vers proto   port
100000    2   tcp    111  portmapper
100000    2   udp    111  portmapper
100004    2   udp    919  ypserv
100004    2   tcp    920  ypserv
100004    1   udp    919  ypserv
100004    1   tcp    920  ypserv
100007    2   tcp   1024  ypbind
100007    2   udp   1030  ypbind
100007    1   tcp   1024  ypbind
100007    1   udp   1030  ypbind
100009    1   udp   1023  yppasswdd
100069    1   udp    937
100069    1   tcp    939
100005    1   udp   1038  mountd
100005    3   udp   1038  mountd
100005    1   tcp    979  mountd
100005    3   tcp    979  mountd
100003    2   udp   2049  nfs
100003    3   udp   2049  nfs
100024    1   udp   1039  status
100024    1   tcp   1026  status
100021    1   tcp   1027  nlockmgr
100021    1   udp   1041  nlockmgr
100021    3   tcp   1028  nlockmgr
100021    3   udp   1042  nlockmgr
100021    4   tcp   1029  nlockmgr
100021    4   udp   1043  nlockmgr
100021    2   tcp   1030  nlockmgr
100021    2   udp   1044  nlockmgr
100020    3   tcp   1031  llockmgr
100020    3   udp   1045  llockmgr
100011    1   udp   1053  rquotad
```

Furthermore, RPC can used to determine patterns of trust based on NFS. The `showmount -e host` command can be used for such purposes. An example output would be as follows:

```
Export list for arf.iyte.edu.tr
/root      host1.iyte.edu.tr
/home1     host2.iyte.edu.tr
/home2     host3.iyte.edu.tr
```

where the host `arf` allows hosts `host1, host2 and host3` to access to directories `/root`, `/home1` and `/home2`, respectively.

Chapter 8

LIMITATIONS OF PREVENTION METHODS

Having the most common security problems in the TCP/IP protocol suite described, in this chapter several common prevention methods will be analyzed.

This chapter aims to be a background for the following chapters and describe why security policies based on merely preventive approaches will be insecure and/or inconvenient in several situations.

8.1 TCP Wrappers

TCP wrappers are Unix host security tools that provide access control, logging and booby traps[1]. In this section the first extension of TCP wrappers, which is access control, will be analyzed.

TCP wrappers rely on a simple mechanism. In stead of directly running the desired server program, the wrapper first compares the clients' names against their IP addresses (DNS cross-check) and the entries in two configuration files: `/etc/hosts.allow` and `/etc/hosts.deny`. When a client satisfies the access conditions, the wrapper executes the desired server.

Below are example TCP wrapper configuration files:

```
#hosts.allow

     in.smtpd: ALL

#hosts.deny

     ALL: host1
     in.ftpd: host2 host3
```

The first file describes which `service: host` combinations are allowed. In this example, the SMTP service is granted to all clients. The second file describes which of the `service: host` combinations are disallowed. In the above example, all services are denied for `host1` and FTP service is disallowed for `host2` and `host3`.

The `service: host` pairs that does not match any of these entries are allowed.

However, the safest configuration is having an `ALL: ALL` in the `hosts.deny` file and allowing services on a case by case basis in the `hosts.allow` file[2].

---

[1] W. Venema, "TCP WRAPPER: Network monitoring, access control and booby traps", Proceedings of the Third Usenix UNIX Security Symposium, pp. 85-92, 1992.
[2] T. Dawson, "Linux NET-3-HOWTO, Linux Networking", 1997.

This scheme may provide a very simple and powerful defense in several situations (if employed on every server). However, it should be noted that TCP wrappers are active only after the initial connection is established, furthermore source IP addresses are trusted. Therefore, TCP wrappers are vulnerable to IP spoofing and port scanning attacks.

8.2 IDENT

The Identification Protocol, or IDENT, provides a means to determine the identity of a user of a particular TCP connection. Given a TCP port number pair, it returns a character string which identifies the owner of that connection on the server's system. The IDENT is specified in RFC1413 [3].

An IDENT server listens for TCP connections on TCP port 113. Once a connection is established, the server reads a line of data which specifies the connection of interest. If it exists, the system dependent user identifier of the connection of interest is sent as the reply. The server may then either shut the connection down or it may continue to read/respond to multiple queries.

IDENT was not designed for access control or user authentication purposes. However, the "security considerations" section of RFC 1413 is worth noting in order to understand why access control based on such schemes would be insecure:

- The information returned by this protocol is at most as trustworthy as the host providing it OR the organization operating the host. For example, a PC in an open lab has few if any controls on it to prevent a user from having this protocol return any identifier the user wants. Likewise, if the host has been compromised the information returned may be completely erroneous and misleading.

- The Identification Protocol is not intended as an authorization or access control protocol. At best, it provides some additional auditing information with respect to TCP connections. At worst, it can provide misleading, incorrect, or maliciously incorrect information.

- The use of the information returned by this protocol for other than auditing is strongly discouraged. Specifically, using Identification Protocol information to make access control decisions - either as the primary method (i.e., no other checks) or as an adjunct to other methods may result in a weakening of normal host security.

- An Identification server may reveal information about users, entities, objects or processes which might normally be considered private.

Furthermore, Cheswick and Bellovin note that, IDENT may cause denial-of-service[4]. This can be explained as follows: IDENT is TCP service and as explained in

[3] M. St. Johns, "Identification Protocol", RFC 1413, 1993.
[4] W. R. Cheswick, S. M. Bellovin, *Firewalls and Internet Security: Repelling the Wily Hacker*, Professional Computing Series, Addison Wesley, p. 141, 1994.

Chapter 4, the availability of TCP services are vulnerable to SYN-flooding attacks. Therefore, if a server uses the IDENT for querying its clients' identities (for example for defeating forged TCP connections), situations may be created where legitimate clients' connections are blocked.

8.3 One-time Passwords

A one-time password is one that changes every time it is used. This scheme provides a strong defense against replay attacks, since even a user name and password pair is monitored by an attacker, the attacker can not reuse this information to obtain unauthorized access.

One-time passwords are generally based on secure hash functions. For example, a widely used one-time password system: S/KEY which is defined in RFC 1760 uses the MD4 Message Digest algorithm designed by Rivest[5,6].

Although cryptosystems are beyond the scope of this thesis, it should be noted that, regardless of the strength of the algorithm used, one-time passwords are vulnerable to connection hijacking attacks as described in Chapter 4.

8.4 Firewalls

This section gives a brief description of firewalls' components, and the general philosophy behind security policies based on firewalls. Firewalls can significantly improve the level of a site's security, furthermore in most situations firewalls seem to provide the most comprehensive solutions. However, there are problems with firewalls as well. These problems, which will be described in this section, fall into three perspectives: security, convenience and performance.

8.4.1 Packet Filtering Routers

Packet filtering is done usually using a router with packet dropping capability instead of a router that simply forwards all IP datagrams between networks. Packet filtering routers parse the headers of each packet and then apply predefined rules to determine whether to forward or drop the packets. Generally the header fields that merit attention are:

- Packet type (TCP, UDP, ICMP, etc.)
- Source IP address
- Destination IP address
- TCP/UDP source port
- TCP/UDP destination port
- TCP Flags

Packet filtering routers should also examine which of the router's network interfaces a packet arrived at, and then use this as an additional filtering criterion.

---

[5] N. Haller, "The S/KEY One-Time Password System", RFC 1760, 1995.
[6] R. Rivest, "The MD4 Message-Digest Algorithm", RFC 1320, 1992.

Generally, packet filtering rules are expressed, as a table of conditions and actions that are applied in a certain order until a decision to forward or drop the packet is reached. When a particular packet meets all the conditions specified in a row of the table, the action specified in the row is carried out[7].

8.4.1.1 Topological Solution to IP Spoofing Attacks

Packet filtering routers rely on address based authentication which is subject to forgery. However, they can be used to block spoofed IP datagrams by considering a simple topological knowledge[8]. With this method, the packets purporting to be from a local host but arriving on an outside interface of a router are simply dropped.

Table 8.1 shows the ruleset for inbound datagrams, which will implement this topological solution:

Table 8.1 Ruleset for Implementing the Topological Solution

Rule	Direction	Source IP Address	Destination IP Address	Action
A	In	External	Internal	permit
B	In	Internal	Any	deny

While it provides a strong defense in most situations, it should be noted that this solution will work only if no trust is granted to outside machines[9]. Otherwise, IP datagrams purporting to be from an outside trusted host will satisfy the condition A and be forwarded to the destination host. The important problem in this configuration is that the real world is often too complex to make such decisions. Companies often wish to let support personnel from vendors connect in order to diagnose problems, or they may be engaged in joint venture agreements with other companies and need access to shared resources[10].

8.4.1.2 Filtering Invalid Datagrams

As noted in Chapter 3, the attackers generally forge the source IP addresses of their packets when launching denial-of-service attacks. In several situations the success of this kind of attacks does not depend on the validity of these addresses, therefore attackers may use invalid source addresses for the sake of simplicity. Furthermore, a particular attack, SYN-flooding specifically requires the use of invalid source IP addresses for the reasons explained in Chapter 4.

Therefore, a solution which is generally proposed for defending against this kind of attacks is filtering the packets carrying invalid source IP addresses[11]. It should be

---

[7] D. B. Chapman, "Network (In)Security Through IP Packet Filtering", Proceedings of the Third USENIX UNIX Security Symposium, pp. 2-3, 1992.

[8] R. T. Morris, "A Weaknes in the 4.2BSD Unix TCP/IP Software", AT&T Bell Laboratories, Computing Science Technical Report No. 117, p. 3, 1985.

[9] ibid, p. 3.

[10] W. R. Cheswick, S. M. Bellovin, *Firewalls and Internet Security: Repelling the Wily Hacker*, Professional Computing Series, Addison Wesley, pp. 80-81, 1994.

[11] Computer Emergency Response Team, "TCP SYN Flooding and IP Spoofing Attacks", CA-96.21, CERT Advisory CA 95-01, January 1995.

noted that given the IP address ranges and several special case IP addresses (e.g., net IDs 127.0.0.0, 10.0.0.0 etc.) which must not appear as source addresses (which are explained in Chapter 3), this solution does not require too much effort.

However, this will not defend against situations where an attacker discovers a set of unassigned IP addresses or IP addresses belonging to PCs which are temporarily down (this can be achieved by sending ICMP echo request messages).

8.4.1.3 Network Ingress Filtering

The internal interface of a packet filtering router may be configured to block outbound packets that have source addresses from outside the internal network. This method is called "network ingress filtering" and limits the ability to launch source address spoofing attacks from a given network, because an attacker will only be able to generate packets with internal addresses[12].

Table 8.2 shows the ruleset required for outbound datagrams necessary for implementing the network ingress filtering.

Table 8.2 Ruleset for Implementing the Network Ingress Filtering Method

Rule	Direction	Source IP Address	Destination IP Address	Action
A	Out	Internal	External	Permit
B	Out	External	Any	Deny

Figure 8.1 illustrates an example portion of the Internet. In this example, the "router 2" provides Internet access to the network 9.0.0.0/8, where resides an attacker. It should be noted that, if this router employs an input traffic filter on its ingress (input) link, the attacker will still be able to forge IP datagrams, however the source IP addresses of these forged datagrams will be restricted within the 9.0.0.0/8 prefix.

An important point that should be noted is that network ingress filtering method can be effective, but if taken in large scale. As more Internet Service Providers (ISPs) configure their routers to implement network ingress filtering, the ground for launching source address spoofing attacks may be reduced[13]. However, large scale implementation of network ingress filtering does not seem realistic and one should not trust measures outside of one's administrative control.

In addition, Mobile IP which is defined in RFC2002, is specifically affected by this method[14]. In mobile IP, traffic to the mobile node is tunnelled, however traffic from the mobile node is not tunnelled. This results in packets from the mobile node(s) which have source IP addresses that do not match with the network where the station is attached. Therefore the "reverse tunneling" methods must be employed in order to solve these problems.

---

[12] P. Ferguson, D. Senie, "Network Ingress Filtering: Defeating Denial of Service Attacks Which Employ IP source Address Spoofing", RFC 2267, 1998.

[13] Schuba C. L. et al, "Analysis of a Denial of Service Attack on TCP", IEEE Symposium on Security and Privacy, 1997.

[14] C. Perkins, "IP Mobility Support", RFC 2002, 1996.

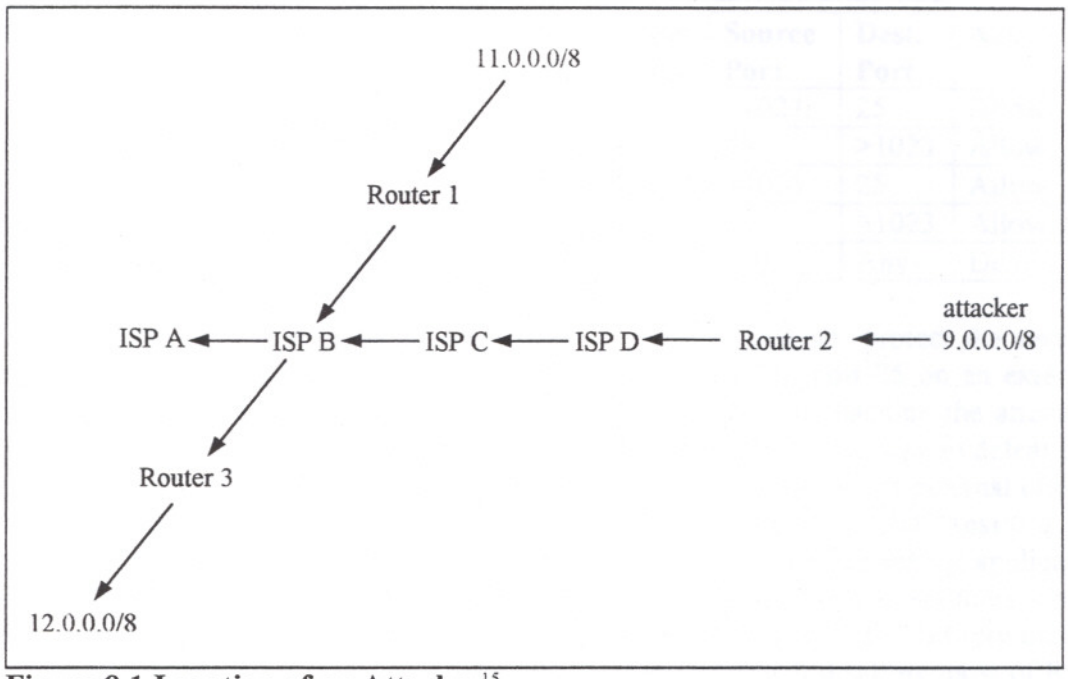


Figure 8.1 Location of an Attacker[15].

8.4.1.4 Filtering Based on Transport Layer Data

A packet filtering router can block TCP and UDP sessions to or from specific ports, then one can implement policies that allow certain types of connections to and/or from specific hosts, but not other hosts. However, this scheme may pose implementation problems in several situations. This section describes the problems in IP packet filtering based on transport layer data.

8.4.1.4.1 Port Numbers

Table 8.3 examplifies a ruleset for a network where only outbound and inbound SMTP traffic is allowed. The important point to note here is that a TCP connection consists of packets flowing in two directions. Thus, rules A and B, together, are needed to allow the inbound SMTP connections and, rule C and D to allow outbound SMTP connections. In addition, in order to make such filtering decisions, packet filtering rules should rely on the distinction between privileged (<1024) and non-privileged (>=1024) ports[16]. However, this distinction is not stated in any RFC, and is therefore best regarded as a widely used convention, but not as a standard (for example, X servers usually listen on port 6000)[17].

[15] P. Ferguson, D. Senie, "Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP source Address Spoofing", RFC 2267, p. 5, 1998.
[16] D. B. Chapman, "Network (In)Security Through IP Packet Filtering", Proceedings of the Third USENIX UNIX Security Symposium, p.8, 1992.
[17] ibid, p. 7.

Table 8.3 Ruleset for Allowing Outbound and Inbound SMTP Traffic[18].

Rule	Direction	Type	Source IP Address	Destination IP Address	Source Port	Dest. Port	Action
A	In	TCP	External	Internal	>1023	25	Allow
B	Out	TCP	Internal	External	25	>1023	Allow
C	Out	TCP	Internal	External	>1023	25	Allow
D	In	TCP	External	Internal	25	>1023	Allow
E	Either	Any	Any	Any	Any	Any	Deny

Another important point to note is that these rules will not protect the servers listening on or above port 1024, from an attack launched from port 25 on an external machine, which is certainly possible if the attacker controls the machine the attack is coming from. Rules C and D, together, will allow such packets[19]. One way to defeat this kind of attacks is to consider some control bits(e.g., SYN, ACK) so that external clients can not initiate connections from a port numbered 25. Nevertheless, UDP sessions are connectionless, therefore filtering rules relying on SYN or ACK flags are not applicable and in this situation the packet filters are forced to rely on source port numbers which are subject to forgery[20,21]. A solution for UDP is often to disallow UDP entirely except for a specific exception for DNS[22]. However, blocking UDP will disallow most of RPC services which use UDP, thus filtering UDP results in a dilemma[23].

Furthermore, RPC-based services do not reliably appear on a given UDP and TCP port number. The associated servers listen on ports that are assigned randomly at system startup. The only RPC-related service that is well-known is the "portmapper" service which listens to its clients on port 111. The portmapper maps initial calls to RPC services to the assigned numbers, but there is no such equivalent for a packet filtering router. This situation makes the RPC based services very difficult to filter effectively[24].

8.4.1.4.2 IP Fragments

The existence of IP fragmentation makes the filtering of TCP and UDP traffics very difficult. Except for the first one, fragments do not contain port numbers, thus there is little information on which to base a filtering decision[25,26].

---

[18] D. B. Chapman, "Network (In)Security Through IP Packet Filtering", Proceedings of the Third USENIX UNIX Security Symposium, p.8, 1992.
[19] ibid, p. 9.
[20] ibid, p. 9.
[21] W. R. Cheswick, S. M. Bellovin, *Firewalls and Internet Security: Repelling the Wily Hacker*, Professional Computing Series, Addison Wesley, pp. 69-70, 1994.
[22] D. B. Chapman, "Network (In)Security Through IP Packet Filtering", Proceedings of the Third USENIX UNIX Security Symposium, p.9, 1992.
[23] John P. Wack, Lisa J. Carnahan, "Keeping Your Site Comfortably Secure: An Introduction to Firewalls", NIST Special Publication 800-10, p. 28, 1994.
[24] D. B. Chapman, "Network (In)Security Through IP Packet Filtering", Proceedings of the Third USENIX UNIX Security Symposium, pp. 9-10, 1992.
[25] ibid, p.6.
[26] W. R. Cheswick, S. M. Bellovin, *Firewalls and Internet Security: Repelling the Wily Hacker*, Professional Computing Series, Addison Wesley, pp. 56-57, 1994.

8.4.1.4.3 FTP, X11 and DNS

At least three major services are not handled well by packet filtering routers: FTP, X11 and DNS.

The problem in FTP and X11 filtering is that the normal operation of these protocols requires incoming calls to the user's host. In FTP, files are transferred via a secondary connection, which is initiated by the server. Similarly, in X11, the user's host is a server. Therefore filtering based on the direction of the call is not possible with X11 and FTP. The problems with the DNS, concern the sensitivity of the information itself as explained in the previous chapter. DNS may cause the leakage of sensitive information. However, inside hosts need to use the DNS to reach outside sites[27].

8.4.2 Application Gateways

To counter some of the weaknesses associated with packet filtering routers, firewalls need to use software applications to forward and filter connections for services. Such an application is referred to as a "proxy" service, while the host running the proxy service is referred to as an "application gateway"[28]. It should be noted that application gateways represent the opposite extreme in firewall design: rather than using a general purpose mechanism to allow different kinds of traffic to flow, special purpose code is used for desired applications[29].

Application level gateways and packet filtering routers are generally combined to provide higher levels of security and flexibility than if either were used alone.

The following example concerns a site which blocks all incoming TELNET connections using a packet filtering router. The router allows TELNET packets to go to one host only, the TELNET application gateway. An external user who wishes to connect to a site system will have to connect first to the application gateway and then to the destination as follows[30]:

1. The user first TELNET to the application gateway and enters the name of an internal host.

2. The gateway checks the user's source IP address and accepts or rejects it according to an access criterion.

3. The user supplies a password.

4. The proxy service creates a TELNET connection between the gateway and the internal host.

---

[27] W. R. Cheswick, S. M. Bellovin, *Firewalls and Internet Security: Repelling the Wily Hacker*, Professional Computing Series, Addison Wesley, p. 57, 1994.
[28] John P. Wack, Lisa J. Carnahan, "Keeping Your Site Comfortably Secure: An Introduction to Firewalls", NIST Special Publication 800-10, p. 29, 1994.
[29] W. R. Cheswick, S. M. Bellovin, *Firewalls and Internet Security: Repelling the Wily Hacker*, Professional Computing Series, Addison Wesley, p. 75, 1994.
[30] John P. Wack, Lisa J. Carnahan, "Keeping Your Site Comfortably Secure: An Introduction to Firewalls", NIST Special Publication 800-10, p. 29, 1994.

5. The proxy service then passes bytes between the two connections.

6. The application gateway logs the connection.

The important advantage of this scheme is that the security of site systems rely on a single point of access and only desired services are allowed. Therefore, firewall administrators are not worried about interactions among different packet filtering rules, nor about the level of security provided by each internal host [31].

In order to take the advantage of this scheme application gateways generally employ more sophisticated access control rules such as one-time passwords.

Another benefit of using an application gateway is information hiding, in which the names of internal systems need not necessarily be made known via DNS to outside systems, since the application gateway may be the only host whose name must be known.

Application gateways can also be used to log and filter all incoming and outgoing traffic. For example, FTP connections can be controlled to deny the use of the FTP put command, which is useful to defend against the attacks, which involve depositing of fake information or malicious code to a target host, or flooding the secondary storage units (such as disks) of a server by uploading unlimited amount of data.

On the other hand application gateways suffer from at least one security problem: TCP connection hijacking attacks which provides a means for bypassing one-time passwords.

8.4.3 Circuit-Level Gateways

Another firewall type that is sometimes included under the category of application gateway, is circuit-level gateway. A circuit-level gateway relays TCP connections as an application gateway, but does no extra processing or filtering of the protocols[32].

8.4.4 Generic Problems

Firewalls suffer from several problems, which are independent of the design approaches. This section focuses on these problems.

8.4.4.1 Inside and Outside Attacks

From a firewall administrator's point of view, a firewall divides the Internet into two logical regions: the protected network which resides behind the firewall (inside) and the rest of the Internet (outside). Therefore, regarding whether the attacker's and

---

[31] W. R. Cheswick, S. M. Bellovin, *Firewalls and Internet Security: Repelling the Wily Hacker*, Professional Computing Series, Addison Wesley, p. 75, 1994.

[32] John P. Wack, Lisa J. Carnahan, "Keeping Your Site Comfortably Secure: An Introduction to Firewalls", NIST Special Publication 800-10, pp. 31-32, 1994.

target hosts reside inside or outside the firewall, the attacks in the Internet can be grouped in the four categories listed in Table 8.4.

Table 8.4 Possible Locations of an Attacker and Target.

#	Attacker	Target
1	Outside	Outside
2	Outside	Inside
3	Inside	Outside
4	Inside	Inside

By its nature a firewall provides defense against the attacks that fall into the second category shown in Table 8.4 where an outside attacker launches an attack targeting an inside machine[33].

Attacks that fall into the first category are the ones originated from external hosts and targeting other external hosts. Therefore they are totally invisible to the protected network. At first glance this type of attacks does not seem to pose any risks against the internal network, however there are important exceptions. For example, the target may be an external trusted host and, once it is compromised it can be used as an intermediary step to the inside network and firewalls are not able to know whether a given pattern of trust is breached or not.

Another important type of attack that is not handled by firewalls, is the fourth one where both the attacker and the target are inside the protected network. Examples to this kind of attacks would be: an inside user (legitimate or not) who launches an attack to another internal host, or an inside host which is compromised and used as an origin for further attacks targeting other internal hosts.

8.4.4.2 Public Services

It is generally necessary to support various public services. These include SMTP, WWW and anonymous FTP. It should be noted that it is almost impossible to apply pre-defined access control rules for these services. On the other hand, as described in Chapter 4, SYN-flooding attacks pose a serious security risk against the availability of TCP services, and since access control mechanisms relying on trust are not applicable to public TCP services, they are not protected at all.

Solutions for SYN-flooding attacks will be discussed in Chapter 10.

8.4.4.3 Ease-of-use against Security

In configuring a firewall, a decision must be made as to whether security is more important than ease-of-use, or vice versa. There are two basic approaches that summaries this conflict[34]:

---

[33] Except the network ingress filtering method which is proposed to defeat IP spoofing attacks falling in the third category.

[34] M. J. Ranum, "Thinking About Firewalls", Proceedings of Second International Conference on Systems and Network Security and Management, p. 2, April 1993.

- That which is not expressly permitted is prohibited.
- That which is not expressly prohibited is permitted.

Where, in the former case, the firewall must be designed to block everything, and services must be enabled on a case-by-case basis only after a careful assessment of need and risk. This tends to impact the users directly in terms of ease-of-use[35]. For example, application level gateways generally require modified client softwares or a modification in user behaviour (the user has to first connect to the Firewall as opposed to connecting directly to the host).

In the second case, the system administrators are placed in a reactive mode, having to predict user behaviour which might weaken the security of the site systems, and preparing defenses against them. In this situation a user can generally compromise the security of the hosts if they are not aware of reasonable security precautions[36].

On the other hand, this conflict may impact the system administrators themselves as well. For example, defending against several ICMP attacks described in Chapter 6, may necessite the filtering of ICMP traffic, which will make some useful ICMP based administrative tools unavailable. Another example would be the filtering of the source routed traffic in order to defend against risks explained in Chapter 4.

A proposed solution to this conflict, is assuming that security is always of a higher-priority than ease-of-use[37].

8.4.4.4 Security of the Firewalls Themselves

Another important point is that, the firewalls are accessible to the outsiders. Therefore, they must be as secure as possible to defend against the attacks targeting themselves, which is not convenient in all situations.

8.4.4.4.1 Packet Filtering Routers

As described in Chapter 6, there exist several denial-of-service attacks targeting the routers. Therefore, packet-filtering routers must be configured so as to be invulnerable to these attacks. For example, a router must not respond to, nor forward the ICMP echo traffic in order to defend against ICMP echo flooding attacks. More specifically, it must silently discard the ICMP echo and reply messages destined for broadcast addresses in order to defend itself against the "smurf" attacks. However, these solutions will result in inconvenience as mentioned above.

8.4.4.4.2 Application Gateways

If an application gateway could be penetrated and reconfigured, it could permit open access to any host within the private network. Therefore, a design goal in

---

[35] M. J. Ranum, "Thinking About Firewalls", Proceedings of Second International Conference on Systems and Network Security and Management, p. 2, April 1993.
[36] ibid, p. 2.
[37] M. J. Ranum, "A Network Firewall", Proccedings of World Conference on System Administration and Security, pp. 1-2, July 1992.

application gateways should be being relatively unobtrusive, services should be make to work as if the gateway is not there at all, if possible[38].

8.4.4.5 Throughput

Regardless of the design, a firewall may cause reduced throughput, since all incoming traffic must be checked against several security criteria before it is allowed or denied[39].

[38] M. J. Ranum, "A Network Firewall", Proccedings of World Conference on System Administration and Security, p. 1, July 1992.

[39] W. R. Cheswick, S. M. Bellovin, *Firewalls and Internet Security: Repelling the Wily Hacker*, Professional Computing Series, Addison Wesley, p. 74, 1994.

Chapter 9

DISCUSSION: NETWORK SECURITY MONITORING

9.1 Overview of Network Monitoring

Network monitoring tools are used to capture and display packets exchanged between different hosts in a network. They help developers and maintainers of software to detect and solve network problems. Network monitoring tools can also be used to monitor performance of communication software. Nevertheless, this chapter focuses on the benefits of network monitoring in the context of network security.

Most commercially available network monitors are stand-alone units dedicated to monitoring specific protocols. However, a network monitor integrated with a general-purpose operating system running on a workstation has several important advantages over a dedicated monitor [1]:

- All the tools of the workstation are available for manipulating and analyzing packet traces. Problem solving requires appropriate diagnostic tools and ideally these tools should be available where the problems are.

- A user can write new monitoring programs to display data in novel ways, or to monitor new or unusual protocols.

Therefore an integrated network monitor appears to be far more useful than a dedicated one. Currently, there are three major approaches for capturing packets in a workstation:

1. Kernel Level Demultiplexing

In many operating systems, network code resides in the kernel. In this case data is typically demultiplexed to user end-points by processing packets through the protocol layers in the kernel. Each layer of the protocol stack looks at its corresponding header and demultiplexes the packet to the next higher layer of the stack, until the user process receives the packet. Such kernel demultiplexing is efficient, requiring minimal context swiching and system calls per packet. However, kernel level network code is much harder to write and debug[2]:

- Each time a bug is found, the kernel must be recompiled and rebooted.
- Bugs in kernel code are likely to cause system crashes.
- Functionally independent kernel modules may have complex interactions over shared resources.
- Kernel-code debugging can not be done during normal time-sharing; single-user time must be scheduled, resulting in inconvenience for time-sharing users and odd work hours for system programmers.

---

[1] J. C. Mogul, R. F. Rashid, M. J. Accetta, "The Packet Filter: An Efficient Mechanism for User-level Network Code", Proceedings of 11th Symposium on Operating Systems Principles, ACM pp. 39-51, p. 2 (reprinted), November 1987.

[2] ibid, p. 14.

- Sophisticated debugging and monitoring facilities available for developing user-level programs may not be available for developing kernel code.
- Kernel source is not always available.

2. User Level Demultiplexing

In spite of the drawbacks, network code is still implemented in the kernel because the drawbacks of putting it outside the kernel seem worse. User level network code suffers from the overhead of context switches and system calls: with each received packet, the system must switch into the demultiplexing process and then switch again when the intermediate process transfers the packet to the final recipient process[3].

3. Packet Filters

To overcome the disadvantages of the above two approaches, the idea of packet filtering was developed[4,5,6]. A packet filter is essentially a kernel agent close to the network device that checks incoming packets and sends them to appropriate user endpoints. Each endpoint provides a specification of the packets it wants, giving just enough information to be able to identify desired packets. The packet filter demultiplexes all packets that satisfy a particular specification to the associated endpoints. The specification is protocol independent, thus allowing the filter to demultiplex packets belonging to any protocol. Packet processing is left to the endpoint. This scheme combines the advantages of kernel and user level demultiplexing.

9.2 The Need for a Network Security Monitor

1. Complete prevention does not seem realistic

In the previous chapter, several important problems with preventive measures were described. These problems are generally caused by the authentication weakness in IPv4 and leakage of sensitive data. Therefore such measures do not seem completely workable until cryptosystems become de facto standards. However, a network security monitor can employ intrusion detection methods where preventive measures are insecure or inconvenient.

2. Most of the threat resides at lower layers

As described in the previous chapters, there exits important vulnerabilities in the link, internet and transport layers of the TCP/IP suite. It should be noted that no matter how secure it is, application data is encapsulated within these unreliable protocols. Therefore, one should have instantaneous access to low level data in order to see what

---

[3] M. Jayaram, R. K. Cytron, "Efficient Demultiplexing of Network Packets by Automatic Parsing", Washington University Department of Computer Science Technical Report: WUCS-95-21, p.2, July 1995.

[4] J. C. Mogul, R. F. Rashid, M. J. Accetta, "The Packet Filter: An Efficient Mechanism for User-level Network Code", Proceedings of 11th Symposium on Operating Systems Principles, ACM pp. 39-51, p. 2 (reprinted), November 1987.

[5] SUN Microsystems Inc., "NIT (4P); SunOS 4.1.1 Reference Manual", Part Number: 800-5480-10, October 1990.

[6] S. McCanne, V. Jacobson, "The BSD Packet Filter: A New Architecture for User-level Packet Capture", 1993 Winter USENIX Conference, January 1993.

traffic is actually on its network. By its nature a network monitor can access all protocol fields of packets in transit.

3. Packet logs can reveal valuable information

Since a network monitor has access to all protocol data at four layers of the TCP/IP suite, it is possible to log critical information considering each layer. These logs can reveal important information about different network based attacks and new vulnerabilities in the protocols.

4. Invisibility is important

Another advantage of a network monitor is that it can be hidden from the attackers because it passively listens to network traffic. Although an attacker has full knowledge of the techniques (and possibly their source codes) used by the monitor he/she can not access it nor exploit design faults in order to obtain unauthorized access to the monitor itself.

9.3 Design Issues

1. High-speed, large volume monitoring

A network security monitor should be able to process large volumes of network data at high speeds. For example, if the link is an FDDI ring, the monitoring will require a means to capture traffic at speeds of up to 100Mbps and the volume of traffic over such a link may be 20GB/day[7]. Therefore, the unnecessary network data should be filtered. A kernel packet filter will be generally the best choice for such purposes since it minimizes the context switching and system call overheads, which is of considerable advantage in high speed monitoring

2. Avoiding packet drops

If an application using a packet filter can not process packets as quickly as they arrive on the monitored link, then the filter buffers the packet for later consumption. However, eventually the filter will run out of buffer and at this point it will drop any further packets that arrive. From a security monitoring perspective, packet drops can completely defeat the monitoring, since missing packets may contain exactly the interesting traffic that identifies a particular attack[8].

3. Security of the monitor itself

Another point that must be considered in network security monitor design, is that an attacker will eventually have full knowledge about the monitor and attempt to launch attacks against it. Although a network monitor can be easily configured to defeat unauthorized access at itself, it can be still exposed to denial-of-service attacks. In this case all site systems, which rely on the security of the monitor, may be left defenseless. A possible form of denial-of-service attack against the monitor can be overloading the

[7] V. Paxson, "Bro: A System for Detecting Network Intruders in Real-Time", Proceedings of the 7th USENIX Security Symposium, pp. 1-2, January 1998.
[8] ibid, p. 2.

monitor in order to cause it to drop packets. A possible defense strategy against such attacks may be leaving some doubt about the exact power and typical load of the monitor[9].

9.4 Logging

In order to construct more information about attacks, one will need as much as data as possible from which to draw inferences and determine what actually happened [10]. As mentioned above, a network monitor is able to record network data including all critical information considering each layer.

Ideally, one would want to have a record of every single packet that crossed the network during a time period around an attack however this may not be always possible for the following reasons[11]:

- Secondary storage capabilities are limited.
- It is difficult to draw inferences from low-level data.

There are three major questions that must be answered in order to overcome these difficulties:

1. What shall be logged?

In order to answer this question one must have a security policy, hence define his/her assets and the risks associated with them. Therefore, a decision has to be made about the hosts and services wanted to be protected. Early dropping of packets that are not destined for these sockets will help in reducing the storage requirements and unwanted packet drops.

However, determining the risks requires knowledge about particular vulnerabilities found in different protocols. For example, logging information about source IP addresses will not help in determining IP spoofing attacks. More information such as Ethernet addresses will be generally needed in order to differentiate between packets received from outside (containing the source Ethernet address of a router) and those received from inside trusted hosts. ACK segments can help in determining the ACK storms caused by TCP connection hi-jacking attacks, and SYN segments will generally contain valuable information concerning SYN flooding, port scanning and ISN sampling attempts, therefore should be logged. ICMP message types and codes should be also logged since they can reveal information about ICMP based denial-of-service attacks.

As a result, knowledge about the security problems found in the protocols can be useful in managing logs. However, the success of such measures will generally depend on the level of expertise of the security administrators.

---

[9] V. Paxson, "Bro: A System for Detecting Network Intruders in Real-Time", Proceedings of the 7th USENIX Security Symposium, pp. 10-11, January 1998.

[10] M. J. Ranum, Network Flight Recorder Inc., "Network Forensics: Network Traffic Monitoring", White Paper, URL: http://www.nfr.net/forum/publications/monitor.html, 1997.

[11] ibid.

2. *How it will be presented?*

Another important decision that has to be made concerns the form of the data rather than its content. The human mind is a powerful tool for seeing information. Fed with the correct data, presented carefully, it will draw interesting conclusions very quickly[12]. However, choosing the correct data and presenting it in an appropriate way may not be easy in all situations. There are two basic facts that summarizes the conflict in log presentation:

- It is difficult to draw inferences from low-level data.
- High-level data can miss valuable information.

Besides its storage difficulties, low-level network data usually contains large amounts of information both hard to track and convey. For example, TCP sequence number failures observed in a given connection may be the sign of sequence number attack attempt. However, this requires the comparison of all sequence and acknowledgment numbers, which is not practical. Although intrusion detection systems can help in such situations, the security administrators will generally desire summaries rather than raw data, such as information about the initiator of a connection, the duration, the amount of data transferred, and possibly the states followed. However, these summaries may miss important low-level anomalies, which could point to specific signs of attacks. In addition, different activities, which seem legitimate when analyzed separately, can be in fact the particular steps of an attack. Therefore, crosschecking of network activities may be needed, as well. Another point that must be considered in data summarising is that it will require the processing of raw data at high speeds.

3. *When shall be discarded?*

It is easier to throw away data later than it is to get it back. Therefore, the network activity logs should be conserved for as long as possible[13]. From a security perspective, past logs will generally merit considerable attention since they can reveal interesting commonalities among different observations and lead the security administrators to note particular signs about different attacks, vulnerabilities and possibly an individual attacker.

9.5 Intrusion Detection

In the previous chapter several important failures in preventive approaches were described. There are important cases where prevention techniques are not acceptably secure. Furthermore, the well-known conflict between security and convenience is mostly caused by such measures. Another issue is authentication. Such measures are generally based on patterns of trust, however proper authentication does seem possible without the use of cryptographic methods.

In this section an alternative approach to computer and Internet security will be overviewed: detection. A network security monitor can employ intrusion detection

[12] M. J. Ranum, Network Flight Recorder Inc., "Network Forensics: Network Traffic Monitoring", White Paper, URL: http://www.nfr.net/forum/publications/monitor.html, 1997.
[13] ibid.

methods so that attacks exploiting the vulnerabilities in the protocols can be detected in real-time[14].

Current approaches to detecting intrusion can be classified into two major categories[15]:

1. Anomaly Detection

Anomaly detection techniques assume that all intrusive activities are necessarily anomalous. This means that if one could establish a "normal activity profile" for a system, it would be possible for him/her to flag all system states varying from the established profile by statistically significant as intrusion attempts. However, if the set of intrusive activities only intersects the set of anomalous activities instead of being exactly the same, two possible problems may arise:

- False positives: anomalous activities that are not intrusive, flagged as intrusive.
- False negatives: intrusive activities that are not flagged intrusive.

The main issue in anomaly detection thus becomes the selection of threshold levels so that neither of the above problems is unreasonably magnified.

The primary disadvantage of anomaly detection systems is that attackers who know that they are being monitored can train such systems over a lengthy of time to the point where intrusive behaviour is considered normal.

In the next chapter an anomaly detection method for detecting TCP SYN-flooding attacks will be discussed.

2. Misuse Detection

The concept behind misuse detection schemes is that there are ways to represent attacks in the form of a signature so that even variations of the same attack can be detected. The main issues in misuse detection systems are how to write a signature that encompasses all possible variations of an attack, and how to write signatures that do not also match non-intrusive activity.

The primary disadvantage of this approach is that it looks for only known vulnerabilities and is of little use in detecting unknown future intrusions.

---

[14] B. Mukherjee, L. Heberlein, K. Levitt, "Network Intrusion Detection", IEEE Network, 8(3), pp. 26-41, May/June 1994.

[15] A. Sundaram, "An Introduction to Intrusion Detection", URL:http://www.acm.org/crossroads/xrds2-4/xrds2-4.html, 1996.

Chapter 10

CASE STUDY: DEFENDING AGAINST SYN-FLOODING ATTACKS

10.1 Host Based Approaches

In order to immunize server machines against SYN-flooding attacks, two different measures are generally proposed [1]:

1. Decreasing the SYN-RECEIVED state timeout value.
2. Increasing per-port backlog queue limit.

As mentioned in Chapter 4, the timeout value associated with the SYN-RECEIVED state is generally 75 seconds. Decreasing this value will reduce the time a half-open connection will occupy the backlog queue, therefore the duration of denial-of-service after the attack. However, a timeout value set too short will increase the risk of aborting legitimate connections over low-speed paths.

As for the second choice, which is increasing the backlog queue limit, it depends on physical memory capacities. Each entry in the backlog queue will allocate an amount of memory which may be different for different implementations of TCP. In Sun Microsystems Security Bulletin on SYN-flooding attacks a queue length of 8,192 is proposed[2].

It should be noted the upper limit to the length of the backlog queue is loosely defined and it is not guaranteed that a given length will suffice in all situations.

10.2 Firewall Solutions

In this section several Firewall based solutions to the SYN-flooding attacks will be explained. These solutions are already implemented by several vendors[3,4].

10.2.1 Circuit Level Gateway

This approach requires a circuit level gateway. The gateway machine answers an incoming connection request on behalf of the destined server, controls if the 3-way handshake successfully completes and establishes a second connection to the server only after the client completes the 3-way handshake. Then the gateway acts as relay by copying bytes from the sender to the destination. The operation of circuit level gateway is illustrated Figure 10.1. An important advantage of this scheme is that a target host never receives forged SYNs. However, there are two major problems with this approach. First, the firewall itself must not be vulnerable to SYN-flooding attacks.

---

[1] CISCO Systems Inc., "Defining Strategies Against TCP SYN Denial of Service Attacks", White Paper, September 1996.
[2] SUN Microsystems Inc., "SUN's TCP SYN Flooding Solutions", SUN Microsystems Security Bulletin #00136, October 1996.
[3] L. S. Laboratories, "Livermore Software Lab. Announces Defense against SYN Flooding Attacks", October 1996.
[4] C.P.S.T. Ltd., "TCP SYN Flooding Attack and the Firewall-1 SYNDefender", October 1996.

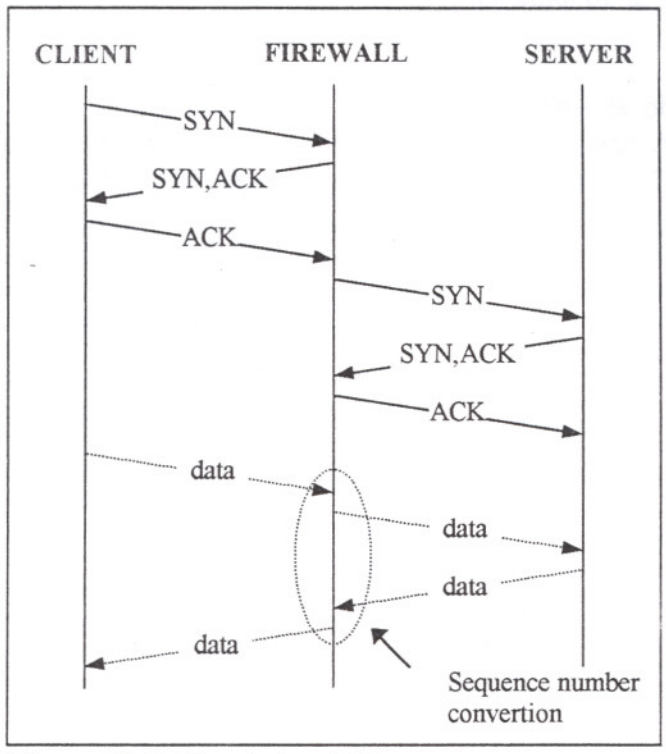


Figure 10.1 Circuit-level Gateway.

Second, by its nature the circuit level gateway introduces new delays for legitimate connections, both at connection establishment time and for each data packet.

When the 3-way handshake is not completed by a given client (for example, the final ACK segment does not arrive), the firewall terminates that connection, and the server never receives the packets from that client. This is illustrated in Figure 10.2.

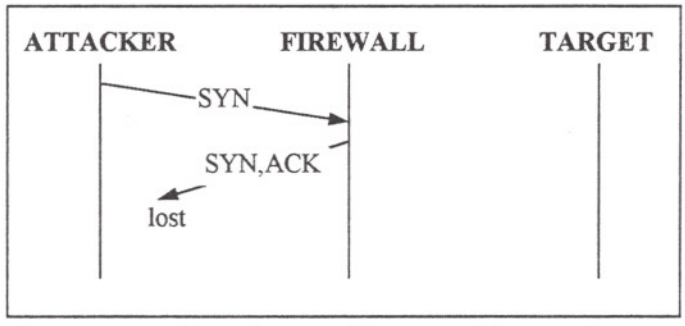


Figure 10.2 Circuit-level Gateway Filters the Attack.

10.2.2 Semi-transparent Gateway

With this approach, the firewall allows SYN and ACK segments to pass through, as opposed to the circuit level gateway approach. However, it monitors the

TCP traffic and reacts to it. Schuba et al call this kind of a firewall as a "semi-transparent gateway"[5].

When a SYN segment destined for an internal host is observed, the firewall waits for the server's answer (a segment containing a SYN and ACK), then reacts by generating the necessary ACK segment on behalf of the client. This completes the 3-way handshake. If the client is legitimate, it sends its own ACK segment, resulting in a duplicate ACK. However, TCP can handle duplicate segments. The client's ACK is silently discarded by the server, the connection is already established and data flows in both directions, without any further firewall intervention. This is illustrated in Figure 10.3.

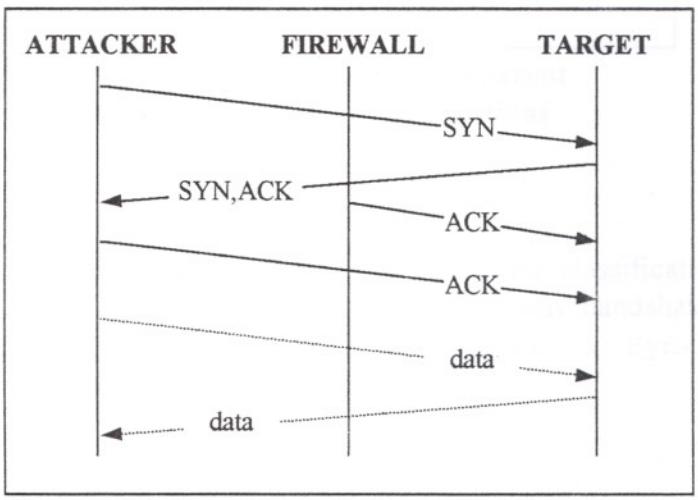


Figure 10.3 Operation of a Semi-transparent Gateway During Normal Operation.

In the case of an attack, the client's ACK are never seen, then (after a timeout period) the firewall sends a RST packet in order to close the connection. Figure 10.4 illustrates the operation of a semi-transparent gateway when a SYN segment appears to be illegitimate.

The obvious advantage of this scheme is that the firewall does not cause any delay to legitimate connections. However, the timeout period before breaking incorrect connections is loosely defined and a small timeout period denies access to legitimate clients. Furthermore, a flood of SYNs will result in a large number (undefined) of established connections at the target hosts, which will represent extra loads for these hosts. Schuba et al call this situation as "service degradation"[6].

[5] Schuba C. L. et al, "Analysis of a Denial of Service Attack on TCP", IEEE Symposium on Security and Privacy, 1997.
[6] ibid.

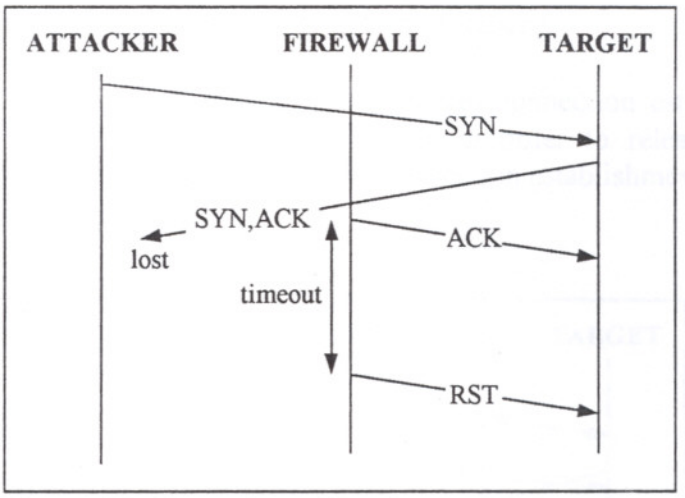


Figure 10.4 Reaction of Semi-transparent Gateway against Half-open Connections

10.3 Synkill

A method proposed by Schuba et al is based on the classification of source IP addresses of the clients and keeping track of the TCP 3-way handshakes as described above[7]. The tool employing this method is called Synkill. Synkill runs on a network monitor.

10.3.1 Algorithm

Synkill algorithm classifies the source IP addresses as follows:

- NULL: Addresses that have never seen.
- GOOD: Addresses belonging to correctly behaving hosts
- NEW: Potentially spoofed addresses
- BAD: Most certainly spoofed addresses
- PERFECT: Addresses that are administratively configured as GOOD
- EVIL: Addresses that are administratively configured as BAD

Synkill performs several processing steps on every TCP packet observed by the network monitor. These steps can be divided into:

- Address pre-filtering, where the observed addresses are classified as described above.
- A decision process based on a state machine to determine correct state membership and actions.

[7] Schuba C. L. et al, "Analysis of a Denial of Service Attack on TCP", IEEE Symposium on Security and Privacy, 1997.

`Synkill`, then takes one of the two possible actions:

- Send RST segments whenever it observes connection establishments from impossible, BAD or EVIL addresses, in order to release the resources allocated at the destination host for connection establishments. This is shown in Figure 10.5.

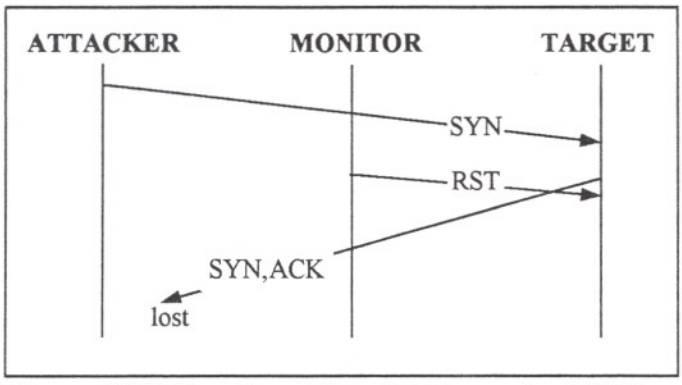


Figure 10.5 Reaction of `Synkill` against BAD or EVIL Addresses.

- Complete the TCP connections by generating the third message of the 3-way handshake, and sending it to the server as shown in Figure 10.6. The purpose of this action is to move a connection quickly from the SYN_RECEIVED to ESTABLISHED state. Then `Synkill` resets the connection if the client's ACK is not seen in a timeout period.

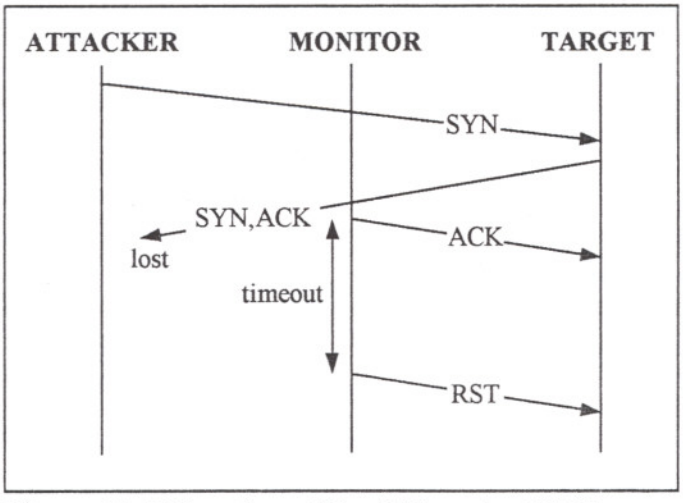


Figure 10.6 Reaction of Synkill against Half-open Connections.

10.3.2 Operation

In addition to address classification, `Synkill` performs the following steps:

- processing of administrative input
- handling of expiry events
- handling of staleness events
- sending RST for all impossible addresses (e.g., 0.0.0.0 or 127.0.0.0)
- sending ACK to complete observed SYN+ACK connections
- sending RST for all evil addresses (e.g., networks 10.0.0.0, 172.16.0.0., 192.168.0.0)

10.3.3 State Machine

After the preprocessing steps are taken, `Synkill` operates as a state machine. The source addresses of each TCP packet is examined to determine the set membership of the address (NULL, NEW, BAD, or GOOD). NULL addresses are not saved explicitly, because it is not practical to keep data structures for all possible IP addresses. If an address is not present in the database, it is considered to be in state NULL.

Figure 10.7 depicts the `Synkill` state machine. The symbol *u* denotes when the timestamp of a given address is updated. These timestamps are used to generate timer events which will be described below. *Record* denotes where datagram information (IP addresses, ports, and sequence numbers) is recorded, so that a RST segment can be generated later if necessary.

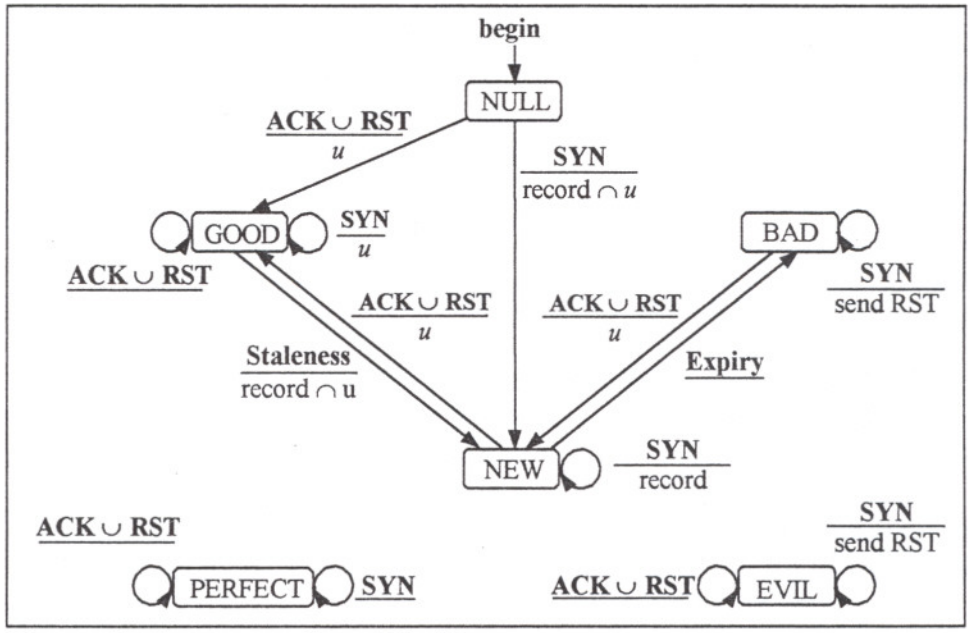


Figure 10.7 `Synkill` State Machine.

There are several distinct sets of events: observed TCP packets, timer events, and administrative commands.

1. Observed TCP Segments

- SYN: The state machine is designed to ignore SYN segments for addresses that are in the NEW, GOOD, or PERFECT states. For addresses that are in the BAD or EVIL states, a reset is generated and sent. The very first SYN segment received from an address is moved into the NEW state to indicate suspicion. As soon as further valid TCP traffic from that address is observed (ACK, RST) the address is moved into the GOOD state.

- ACK, RST: If `Synkill` receives a valid ACK or RST packets from an address, it means that the host generates valid packets and the address can be considered GOOD. The address is moved into the GOOD state.

2. Timer Events

- Expiry: An expiry event occurs if the timer associated with an address in the NEW state expires. This means that `Synkill` has not observed any valid TCP traffic from that addresses. The address is therefore moved into the BAD state and RST packets are generated and sent for all SYN packets from that address that were observed while the address was in the new state.

- Staleness: `Synkill` employs a mechanism to allow addresses in the GOOD state to leave the GOOD state after no TCP traffic was observed from that address for a period of time, i.e., staleness period. The purpose of the staleness period is to correctly classify spoofed addresses as BAD even if they were once GOOD.

10.3.4 Problems

The authors of `Synkill` note several problems about this approach.

First, the expiry timer: ideally it would be much smaller than the current 75 seconds timout. However, the smaller the chosen value the more likely it is for legitimate connections to be erroneously denied.

Second, it is possible for an attacker to "teach" `Synkill` GOOD addresses that are in fact forged. This can be achieved by sending fake RST or ACK segments. Then these addresses which are erroneously classified as GOOD, can be used for a SYN-flooding attack. Although `Synkill` artificially completes each connection in order to protect the target server from SYN-flooding attacks, this will result in service degradation as explained in Section 10.2.2. In order to defend against this kind of attacks the authors propose keeping state information about all observed TCP

connections, which requires sequence number checking. In this case an attacker will have to predict the target's sequence numbers.

10.4 Detecting SYN-flooding Attacks

In this section, our approach to the SYN-flooding attack will be explained. Briefly, we propose a statistical anomaly detection method.

10.4.1 Objectives

Our objective is the detection of a SYN-flooding attack in real-time and each time we detect an attack we want to be able to define the level of threat we are facing and act accordingly. We note here that none of the above methods were designed in this sense. In stead, they were designed to analyze the correctness of each TCP connection and act for them separately, regardless of the existence of a considerable threat. However, being able to differentiate an attack from the normal mode of operation may have considerable advantages.

A common flaw that we note in the methods mentioned above is early timeout expiries, which cause denial-of-service to legitimate clients suffering from low bandwidth. A long timeout period causes service degradation at the server as opposed to a short one that results in denial-of-service to particular legitimate clients. Currently, the common policy about this dilemma is: "clients already suffering from low quality of service should not victimize others", therefore short timeout periods are generally preferred. One vendor calls this an "aggressive timeout". Our statement is that the aggressive timeouts should not be employed when there is no obvious reasons to do so. In today's Internet not every user has the chance to connect via high-speed paths, nor there is a guarantee that the others will have the same level of quality of service twenty-four hours a day. We note here, the importance of detecting a SYN-flooding attack, which will provide us the ability of employing aggressive timeouts only when it is necessary: during an attack.

To simplify the discussion, we first consider the protection of only one host: `ephesus` [8] which is one of the Internet servers of Izmir Institute of Technology and in order to detect a SYN-flooding attack destined for this host we use a network monitoring machine attached to the same physical network. This machine analyzes the SYN segments destined for `ephesus`, classifies them regarding their destination ports and aims to detect it whenever a SYN-flooding attack is launched against a given service. The method that we propose depends also on the proper setting of the backlog queue length of `ephesus`. Therefore we also propose a method for more precisely determining this value.

Our study is based on a set of traces obtained by monitoring all SYN segments destined for `ephesus` during 68 days (between Monday 23/11/98 and Friday 29/01/99).

[8] The name of the host has been changed for the reasons explained in Section 10.4.3.1.

10.4.2 Detection Method

The detection method that we propose is based on the intensity measures of SYN segments. At time of writing there exists at least one reference which mentions such a possibility[9]. The parameters that can be associated with the SYN flooding attack are:

- T: SYN-RECEIVED state timeout in seconds (usually 75).
- L: Per-port backlog queue length.
- A: Number of received SYN segments per second by a given TCP port.

We call here A, as the intensity of SYN segments. In order to succeed in a SYN-flooding attack, the minimum number of SYN segments that an attacker must send in T seconds is L. Thus, the average intensity of SYN segments in L seconds must be L / T. However this is a minimum value and the higher the chosen rate, the more effective the attack will be. We note that, this situation is of considerable advantage in the detection of an attack. An additional parameter needed for our detection method is:

- A_c: the maximum acceptable (critical) A value.

Then, our network monitor computes A for each second and for each port of `ephesus`, and whenever it observes the condition $A > A_c$ satisfied, it considers the situation as an attack and acts accordingly.

10.4.3 Backlog Queue Length Requirements

In this section we describe the necessary size of the backlog queue for `ephesus`. Briefly, we aim to protect this host against undetectable attacks (hence, avoid false negatives) and decrease the probability of having false positives.

10.4.3.1 Undetectable Attacks

The possible response of an attacker against this method is illustrated in Figure 10.8.

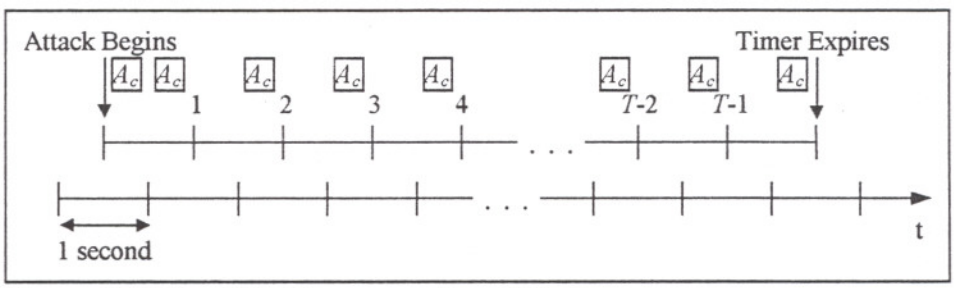


Figure 10.8 Possible Response of an Attacker.

[9] P. A. Porras, A. Valdes, "Live Traffic Analysis of TCP/IP Gateways", Proceedings of the ISOC Symposium on Network and Distributed Systems Security, p. 6., 1998.

It should be noted that the maximum number of SYNs that an attacker will be able to send (without being detected) before the SYN-RECEIVED timer expires is:

$$N = T \times A_c + A_c$$

Therefore, in order to be immune to this worst case (the most intensive but undetectable attack), the backlog queue length of `ephesus` should be at least N +1, which gives:

$$L = T \times A_c + A_c + 1$$

Rearranged to collect the terms multiplied by A_c, this becomes:

$$L\,(A_c,\ T) = A_c \times (T+1) + 1$$

where T is a configurable parameter which is generally 75 seconds. As for A_c, it can be chosen so that the probability of having a false positive ($A > A_c$ but, in fact this is not an attack) is minimum. We want to be able to minimize this probability, because a false positive will cause the network monitor to act unnecessarily.

We note here a flaw: if L is too large, an undetectable attack may result in another form of denial-of-service: service degradation, since in this case too much of the system resources will be allocated by half-open connections. However, such an attack requires knowledge about A_c and the correct synchonization to the monitor's one-second time intervals. It is also possible to leave a doubt as to the exact time intervals of the monitor. In addition, there is no obvious way for an attacker to understand whether his/her attack is detected or not.

10.4.3.2 False Positives

Observations showed that the probability of receiving large numbers of SYNs in a given time interval is likely to be small during normal operation. Therefore we note that the larger the chosen A_c value the smaller the probability of false positives will be. This illustrated in Figure 10.9. In this figure, α is the area limited by A_c and represents the probability of false positives.

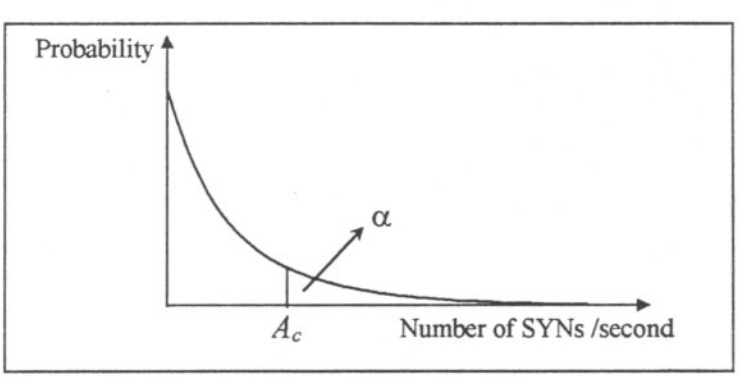


Figure 10.9 Probability of False Positives.

However in practice there will be an upper limit to A_c since it comes as a factor in the calculation of the backlog queue length L needed to be immune to undetectable attacks as formulized above (it should be noted such a setting in L forces the attackers to fall in the α region, which guarantees that we will not have any false negatives). Therefore, a proper statistical model is needed for precisely determining A_c. Nevertheless, there are important limitations in modeling SYN arrivals and setting L accordingly, which will be covered in the next section.

10.4.4 Limitations in Modeling SYN Arrivals

1. The Poisson model will fail

In the classical teletraffic theory, call arrivals are often modeled as Poisson processes. We would like to have such an analytical traffic model for TCP call arrivals (connection arrivals) because they are easier both to convey and analyze. However, a number of past studies have shown that analytic models and specifically the Poisson model (which is interesting for our purpose), does not suite well in today's Internet in all situations. Although we have not further examined the validity of these findings for our case (e.g., establishing statistical tests), we have reasonable pre-known evidences that using a Poisson model would be difficult.

First, our observations showed that we can not reasonably hope to model connection arrivals using homogeneous Poisson processes, which require constant rates. This is also observed by Paxson and Floyd[10]. Figure 10.10 shows the variation of hourly connection arrival rates at `ephesus` (obtained from the first 30 days portion of our traces). The important point that we note in this figure is that the connection arrival rate changes regarding both the hour of the day and the day of the week. For example, the arrival rates generally smaller around midnights and during weekends.

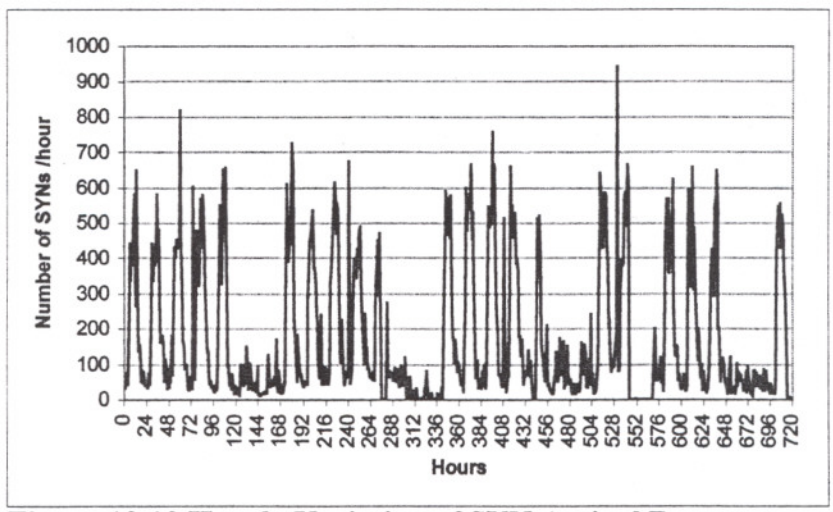


Figure 10.10 Hourly Variation of SYN Arrival Rates.

Paxson and Floyd note that during fixed length intervals (for example 1 hour or 10 minutes) the arrival rates can be assumed constant and the arrivals within each

[10] V. Paxson, S. Floyd, "Wide-Area Traffic: The Failure of Poisson Modeling", IEEE/ACM Transactions on Networking, 3 (3), pp. 226-244, p. 3 (reprinted), June 1994.

interval can be modeled by a homogeneous Poisson process. Such a model is referred to as a "nonstationary Poisson process". User session arrivals are likely to be nonstationary Poisson processes since a user session arrival corresponds to the time when a human decides to use the network for a specific task, hence they are generally uncorrelated. However, for our purposes we are interested in connection arrivals rather than user session arrivals. Paxson and Floyd note that individual TCP connections that comprise each session are not well modeled by a Poisson process[11]. TELNET and RLOGIN connection arrivals generally correspond to a new user therefore are likely to be close to uncorrelated, memoryless arrivals and can be described using nonstationary Poisson processes. However, X11, FTP data transfer and HTTP sessions generally consist of more than one connections which do not have this property therefore are not Poisson[12]. For example, an HTTP session consists of several connections corresponding to a user selecting links to other pages on the same server. Furthermore, in a HTTP session, not every connection is necessarily initiated by the user. The images to appear on a Web page for example, are transferred immediately after the transfer of the text of the page.

Finally, SMTP connection arrivals are not Poisson since they are machine-initiated and can be timer-driven. Furthermore, SMTP connections may be perturbed by mailing list explosions in which one connection immediately follows another and the timer effects due to using the DNS to locate MX records[13].

2. Backlog queue length is static

Another limitation that we note is that L is static. Currently no TCP/IP implementation provides a means for dynamically changing this parameter (furthermore, this would require the server and the monitor to communicate each other). However, as noted above, the clients' behaviour change in time and adapting L to these changes is not possible.

3. Different services have different characteristics

We note that, each service has different characteristics and possibly different clients. Therefore, it would be hard to find a single model that fits in all services. However, in most systems a single deamon, `inetd` issues the `listen()` system call (hence, determine the backlog limit) on behalf of the daemons for which it watches ports.

---

[10] V. Paxson, S. Floyd, "Why We Don't Know How to Simulate The Internet", Proceedings of the 1997 Winter Simulation Conference, Atlanta, GA, p. 5. 1997.
[12] V. Paxson, S. Floyd, "Wide-Area Traffic: The Failure of Poisson Modeling", IEEE/ACM Transactions on Networking, 3 (3), pp. 226-244, p. 4 (reprinted), June 1994.
[13] ibid, p. 4.

10.4.5 Estimating A_{MAX}

For the reasons explained above, we prefer setting L regarding the maximum SYN arrival rate that is likely to occur in the near future. Therefore, we propose:

$$A_c = A_{MAX}$$

Figure 10.11 shows the changes in the maximum SYN arrival rates at services provided by `ephesus` over one-day intervals.

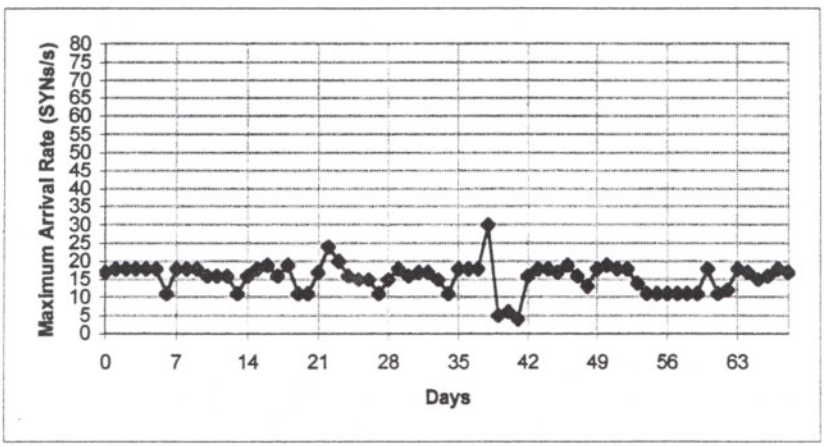


Figure 10.11 Daily Variation of Maximum SYN Arrival Rates.

In this figure, we note that maximum SYN arrival rates are generally bound within the interval 15-20 (48 of 68). Therefore, we can reasonably hope that over one-day intervals the maximum SYN arrival rates will be fairly consistent. Furthermore, some of the deviations from the consistent behaviour can be related to known social events. For example, the maximum arrival rates generally decrease during weekends and holidays (the days 39-41 correspond to new-year vacation and we observe low arrival rates during these days). Therefore, in order to capture the most likely maximum arrival rate we can look at working days. The low arrival rates during days 54-59 are known to be caused by the degradation in our ISP's quality of service and the spike in day 38 is related to the new-year traffic (just before the vacation). Nevertheless, there is no obvious reason for the spike that we observe in day 23. As a result, completely avoiding false positives seems not possible with this method, however we can expect a significant decrease.

Figure 10.12 shows the theoretical numbers of false positives that would occur during the 68 days from which our traces were obtained, for A_c values between 15-20. We note that even for the worst case A_c value which is 15 we can except a significant decrease in the number of false positives (the monitor would act for only 110 seconds during 68 days) and for A_c values of 19 and 20 the number of false positives are negligible (monitor reaction for 9 and 5 seconds during 68 days, respectively).

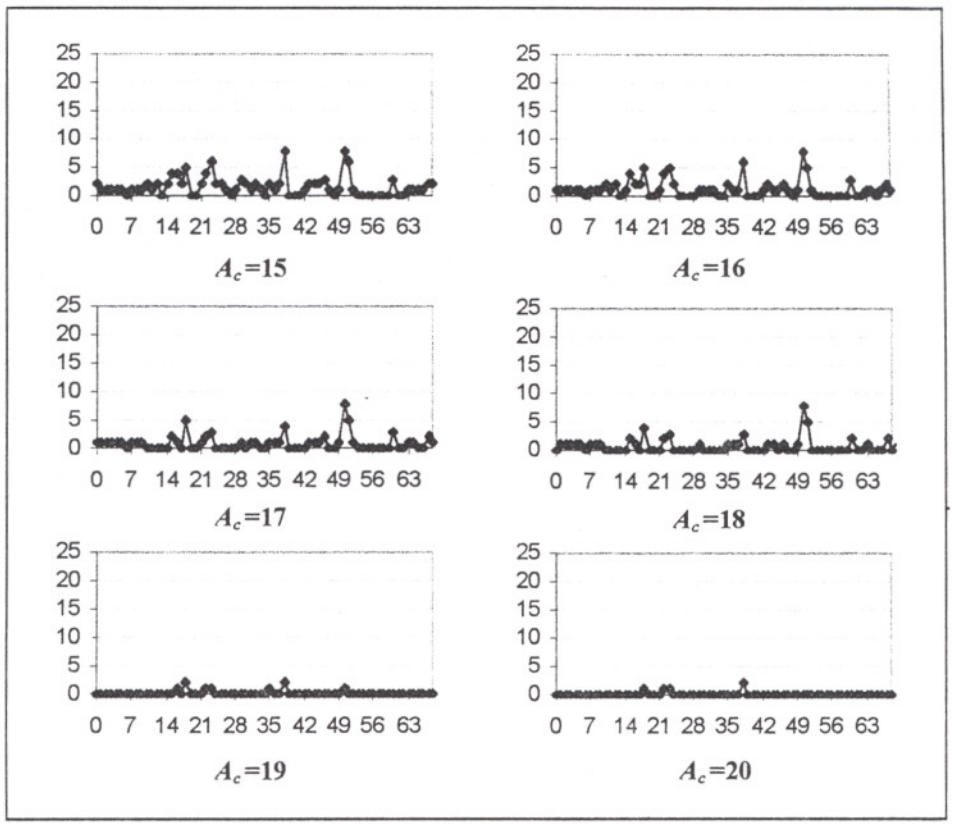


Figure 10.12 Number of False Positives per Day.

10.4.6 Model of Operation

Our network monitor can operate in two different modes: *training* and *defence*. The *training* mode consists of observing the normal client behaviour. Running in this mode, requires several administratively supplied parameters:

- The duration of the *training* period in days.
- SYN-RECEIVED state timeout in seconds (normally 75).
- The list of the hosts wanted to be protected.

Regarding this information, the network monitor observes the SYN arrival rates, computes $A_c = A_{MAX}$ and outputs a backlog queue length, separately for each host in the list. For the reasons noted above, we prefer working days (and when the quality of service is normal) for running our network monitor in this mode.

We use this information for configuring the backlog queue lengths of our servers, and start the network monitor in *defence* mode. In this mode, the network monitor uses the A_c values that it computed in the end of the training period. Then, whenever the $A > A_c$ condition is observed for one of the servers, it records the necessary information (IP addresses, port and sequence numbers) about the suspicious set of connection attempts observed in that second.

Given a set of suspicious connection attempts, the network monitor can take one of the following actions:

1. Directly Sending RSTs: we do not currently propose this kind of an action, since there may be legitimate SYNs in the set (which is certain if this a false positive). This will deny access to legitimate clients.

2. Sending RSTs after an aggressive timeout period: with this method, the network monitor will send RSTs for the connections, which are not established within an aggressive timeout period.

3. Sending ACKs: with this method, the network monitor will immediately send the necessary ACKs to complete the TCP 3-way handshakes and send RSTs if an aggressive timeout period expires.

4. Operating as `Synkill`: this implies that our network monitor can supply `Synkill` with a set of suspicious connection attempts and `Synkill` can compare the source IP addresses in this set against its database and state machine, then act accordingly. This requires `Synkill` to run on the same machine.

For each set, the percentage of connections that were reset is also logged in order to provide security administrators with a means for differentiating between true and false positives. It should be noted that we can expect a small percentage of reset connections for false positives. Therefore, by looking at these logs, system administrators can decide whether the number of false positives is acceptable or not. These decisions can lead to heuristic changes in A_c and in this situation the network monitor is input a larger A_c value and it outputs the necessary L. However, the backlog queue must be reconfigured in this situation.

10.4.7 Notes

We note that early timeout expiry and service degradation problems remain since the actions that we take are not new and still based on aggressive timeouts. However, with this method, these problems will arise only when a considerable threat (true or false) is detected. Therefore, the risk of breaking legitimate connections and causing service degradation is minimized during normal mode of operation.

The performance of the method depends on the correct estimation of A_c. A small A_c will result in a large number of false positives. Then, the network monitor will act for more legitimate connections, resulting in a risk of denying service to more legitimate clients. Therefore, heuristic changes in A_c may be useful. For example, multiplying A_c by a factor of two will almost guarantee a minimum number of false positives. However, this will double the necessary backlog queue length as well.

Another issue is "busy servers". We expect a larger A_c (which requires a larger L) for a busy server. Stevens analyzed the number of SYNs received per second by a

commercial ISP's busy Web server, which was providing service for 22 organizations[14]. He observed $A_c = A_{MAX} = 20$ during one day and for $A_c = 20 \times 2 = 40$, this would require $L = 3041$ (if $T = 75$). We note that this is less than the half of the backlog queue length proposed by one vendor mentioned in Section 10.1. Therefore, we conclude that such a detection method can considerably decrease the primary storage requirements necessary for coping with SYN-flooding attacks.

There is also an important problem with our approach: given the drastic development rate of the Internet and the computer sector, the highest arrival rates will also increase over long term. Such increases can be related to administrative policies as well. For example, installation of a large mailing list or a popular Web server will certainly cause new clients to arrive, therefore increase the arrival rates. Running the network monitor in the *training* mode may be necessary whenever such an increase in the SYN arrival rates is expected or observed.

Finally, we note in this method several differences from the general definition of anomaly detection systems. First, this method is deprived of the primary advantage of anomaly detection methods: capability of detecting unknown attacks, instead it focuses on only a known vulnerability. Second, anomaly detection systems generally suffer from the fact that they can be trained by the attackers so that eventually, intrusive activities are considered normal. However, in our case such attacks can be avoided by considering only established connections during the *training* period. The possible response of an attacker can be establishing connections very frequently, however given the difficulty of sequence number prediction attacks against up to date TCP implementations, this can be too easily detected: too many and frequent connections from a same host.

[14] W. R. Stevens, *TCP/IP Illustrated, Volume 3, TCP for Transactions, HTTP, NNTP, and the UNIX Domain Protocols*, Professional Computing Series, Addison Wesley, pp. 177-182, 1994.

Chapter 11

CONCLUSION

There are a number of important vulnerabilities in the TCP/IP protocol suite. These are generic problems in the protocols themselves and are independent of any implementations of TCP/IP.

TCP/IP does not provide host address authentication. Theoretically, any host can send packets with any source IP addresses. This situation poses two threats: unauthorized access and denial-of-service attacks. Obtaining unauthorized access generally requires TCP sequence number guessing. However, the initial sequence number generation method specified in RFC 793, the official specification for TCP, is vulnerable to such attacks. In this RFC, an incremental ISN generator is proposed in order to avoid duplicate segments. It should be noted that segment duplication problems arise from the fact that TCP/IP is not able to correctly define a MSL for a segment. The ISN generation method described in RFC 1948 seems far more strong (from a security perspective) therefore must be employed in newer implementations. However, by their natures UDP and ICMP protocols does not provide sequencing, therefore are subject to forgery.

The combination of this authentication weakness with network monitoring attacks is even worse. In this case, an attacker is able to monitor the sequence numbers generated by the target, hence can obtain unauthorized access. One-time passwords were proposed and are being used for defending against replay attacks, however regardless of the strength of the algorithm used, one-time paswords are vulnerable to TCP hijacking attacks. Therefore, user authentication does not seem possible unless IP addresses are authenticated and/or data stream is encrypted.

A flaw in the domain name system allows attackers to forge host names, then obtain unautorized access to the sites employing name based patterns of trust. Although there are several possible countermeasures, trust based on host names is not proposed.

There are several points of information disclosure at different layers of the TCP/IP suite. This information can be used by the attackers for revealing vulnerabilities found at potential targets and then launching denial-of-service attacks or obtaining unauthorized access.

Firewalls; although they can provide strong defense in a number of situations, they seem far from being completely secure, unless proper user and host address authentication is provided. Futhermore, they pose a serious contradiction with convenience.

Finally, these conclusions can lead to a future work on a network security monitor design which must be considered as an alternative and supplementary method for defending against, and understanding the risks. Where preventive approaches are not applicable, network monitors can be used for logging intrusive activity, and automated detection of network based attacks.

SUMMARY

There are several vulnerabilities in the TCP/IP protocol suite. These vulnerabilities can be summarized as follows: authentication weakness of IPv4, predictable TCP initial sequence numbers, vulnerability to denial of service attacks and several information leakage points.

Prevention methods and tools such as TCP wrappers, one-time passwords and firewalls suffer from two major problems: in many situations they are not as secure as they should be and/or they pose a trade-off with convenience.

Network security monitoring methods can be used for logging and detecting intrusive activity, however there are a number of points that must be considered. These can be summarized as follows: security of the monitor, monitoring efficiency, the form and the content of the logs, and intrusion detection methodologies to follow.

ÖZET

TCP/IP protokol dizisinde bazı güvenlik açıkları bulunmaktadır. Bu açıklar şöyle özetlenebilir: IPv4' te özgünlük denetimi eksikliği, tahmin edilebilir TCP başlangıç sıra numaraları, hizmet kesme saldırılarına karşı zayıflık ve bilgi sızıntıları.

TCP "wrapper" yazılımları, tek kullanımlık şifreler ve güvenlik duvarları gibi engelleme yöntemleri karşımıza genel olarak iki problem çıkarmaktadır: yeterli güvenlikten yoksunluk ve/veya uygunluk ile çelişki.

Ağ güvenlik denetimi, kütükleme ve saldırı algılama alanlarında fayda sağlamaktadır, fakat dikkate alınması gereken bazı noktalar bulunmaktadır. Bunlar söyle özetlenebilir: denetimin güvenliği ve etkinliği, kütüklerin sunuluş şekli, içeriği ve izlenecek algılama yöntemleri.

BIBLIOGRAPHY

Anonymous, *Maximum Security: A Hacker's Guide to Protecting Your Site and Network*, Macmillan Computer Publishing, URL: ftp://arf.iyte.edu.tr/pub /InternetSecurity /fm /fm.htm, 1998.

Baker F., "Requirements for IP Version 4 Routers", RFC 1812, 1995.

Bellovin Steven M., "Sequence Number Attacks", RFC 1948, 1996.

Bellovin Steven M., "Using the Domain System for System Break-ins", Proceedings of the 5th Usenix UNIX Security Symposium, 1995.

Bellovin Steven M., "Security Problems in the TCP/IP Protocol Suite", Computer Communications Review, Vol. 19, No.2, pp. 32-38, 1989.

Braden R., "Requirements for Internet Hosts – Communication Layers", RFC 1122, 1989.

Chapman D. B., "Network (In)Security Through IP Packet Filtering", Proceedings of the Third USENIX UNIX Security Symposium, 1992.

Cheswick William R., Bellovin Steven M., *Firewalls and Internet Security: Repelling the Wily Hacker*, Professional Computing Series, Addison Wesley, 1994.

CISCO Systems Inc., "Defining Strategies Against TCP SYN Denial of Service Attacks", White Paper, September 1996.

Computer Emergency Response Team, " "smurf" IP Denial-of-Service Attacks", CERT Advisory: CA 98-01, September 1998.

Computer Emergency Response Team, "TCP SYN Flooding and IP Spoofing Attacks", CA-96.21, CERT Advisory CA 95-01, January 1995.

Cowzilla, Pixel Dreamer, "Puke.c", 1996.

C.P.S.T. Ltd., "TCP SYN Flooding Attack and the Firewall-1 SYNDefender", October 1996.

daemon9, route, infinity, "IP-Spoofing Demystified", Phreak Magazine, Volume 7, Issue 48, File 14, 1996.

Dawson T., "Linux NET-3-HOWTO, Linux Networking", 1997.

Farmer Dan, Venema Wietse, "Improving the Security of Your Site by Breaking into It", 1993.

Ferguson P., Senie D., "Network Ingress Filtering: Defeating Denial of Service Attacks Which Employ IP source Address Spoofing", RFC 2267, 1998.

Haller N., "The S/KEY One-Time Password System", RFC 1760, 1995.

Holbrook P., Reynolds J., "Site Security Handbook", RFC 1244, 1991.

Jacobson V., Braden R., Zhang L., "TCP Extension for High-Speed Paths", RFC 1185, 1990.

Jacobson V., "Congestion Avoidance and Control", Proceedings of SIGCOMM '88, 1988.

Jayaram M., Cytron R. K., "Efficient Demultiplexing of Network Packets by Automatic Parsing", Washington University Department of Computer Science Technical Report: WUCS-95-21, July 1995.

Joncheray Laurent, "A Simple Attack Against TCP", 1995.

L. S. Laboratories, "Livermore Software Lab. Announces Defense against SYN Flooding Attacks", October 1996.

Maimon U., "Port Scanning without the SYN Flag", Phrack Magazin, Volume 7, Issue 49, File 15.

McCanne Steven, Jacobson Van, "The BSD Packet Filter: A New Architecture for User-level Packet Capture", 1993 Winter USENIX Conference, January 1993.

Mills D. L., "Internet Delay Experiments", RFC 889, p. 10, 1983.

Mockapetris P. V., "Domain Names – Concepts and Facilities", RFC 1034, 1987.

Mockapetris P. V., "Domain Names – Implementations and Specifications", RFC 1035, 1987.

Mogul J. C., Rashid R. F., Accetta M. J., "The Packet Filter: An Efficient Mechanism for User-level Network Code", Proceedings of 11th Symposium on Operating Systems Principles, ACM pp. 39-51, November 1987.

Mogul J., Postel J., "Internet Standard Subnetting Procedure", RFC 950, 1985.

Morris Robert T., "A Weaknes in the 4.2BSD Unix TCP/IP Software", AT&T Bell Laboratories, Computing Science Technical Report No. 117, 1985.

Moskowitz B., Karrenberg D., de Groot G., Lear E., "Address Allocation for Private Internets", RFC 1918, 1996.

Mukherjee B., Heberlein L., Levitt K., "Network Intrusion Detection", IEEE Network, 8(3), pp. 26-41, May/June 1994.

Muus M., "Ping.c", 1983.

Paxson Vern, "On Calibrating Measurements of Packet Transit Times", Proceedings of SIGMETRICS '88, 1988.

Paxson Vern, "Bro: A System for Detecting Network Intruders in Real-Time", Proceedings of the 7th USENIX Security Symposium, January 1998.

Paxson Vern, Floyd S., "Wide-Area Traffic: The Failure of Poisson Modeling", IEEE/ACM Transactions on Networking, 3 (3), pp. 226-244, June 1994.

Paxson Vern, Floyd Sally, "Why We Don't Know How to Simulate The Internet", Proceedings of the 1997 Winter Simulation Conference, Atlanta, GA, 1997.

Perkins C., "IP Mobility Support", RFC 2002, 1996.

Porras P. A., Valdes A., "Live Traffic Analysis of TCP/IP Gateways", Proceedings of the ISOC Symposium on Network and Distributed Systems Security, 1998.

Postel John, "Internet Protocol", RFC 791, 1981.

Postel John, "Transmission Control Protocol", RFC 793, 1981.

Postel J., "User Datagram Protocol", RFC 768, 1980.

Postel J., "Internet Control Message Protocol", RFC 792, 1981.

Postel J., Reynolds J. K., "Assigned Numbers", RFC 990, 1986.

Postel J., Reynolds J. K., "Assigned Numbers", RFC 1340, 1992.

Ranum Marcus J., "Thinking About Firewalls", Proceedings of Second International Conference on Systems and Network Security and Management, April 1993.

Ranum Marcus J., "A Network Firewall", Proccedings of World Conference on System Administration and Security, July 1992.

Ranum Marcus J., Network Flight Recorder Inc., "Network Forensics: Network Traffic Monitoring", White Paper, URL: http: //www.nfr.net /forum /publications /monitor.html, 1997.

Rivest Ron, "The MD4 Message-Digest Algorithm", RFC 1320, 1992.

SUN Microsystems Inc., "NIT (4P); SunOS 4.1.1 Reference Manual", Part Number: 800-5480-10, October 1990.

SUN Microsystems Inc., "SUN's TCP SYN Flooding Solutions", SUN Microsystems Security Bulletin #00136, October 1996.

Schuba C. L. et al, "Analysis of a Denial of Service Attack on TCP", IEEE Symposium on Security and Privacy, 1997.

St. Johns M., "Identification Protocol", RFC 1413, 1993.

Stevens W. Richard, *TCP/IP Illustrated Volume 1: The Protocols,* Professional Computing Series, Addison Wesley, 1994.

Stevens W. Richard, *TCP/IP Illustrated, Volume 3, TCP for Transactions, HTTP, NNTP, and the UNIX Domain Protocols*, Professional Computing Series, Addison Wesley, 1994.

Sundaram A., "An Introduction to Intrusion Detection", URL:http: //www.acm.org/ crossroads/xrds2-4 /xrds2-4.html, 1996.

Tanenbaum Andrew S., *Computer Networks,* Prentice Hall, 1996.

Venema Wietse, "TCP WRAPPER: Network monitoring, access control and booby traps", Proceedings of the Third Usenix UNIX Security Symposium, 1992.

Wack John P., Carnahan Lisa J., "Keeping Your Site Comfortably Secure: An Introduction to Firewalls", NIST Special Publication 800-10, 1994.

Zimmerman D., "The Finger User Information Protocol", RFC 1288, 1991.