NEAR EAST UNIVERSITY

FACULTY OF ENGINEERING

DEPARTMENT OF COMPUTER ENGINEERING

FIREWALLS AND NETWORK SECURITY

GRADUATION PROJECT
COM-400

Student: Mehmet Bahaddin Yetkin

Supervisor: Prof. Dr. Fakhreddin Mamedov

Nicosia - 2004

ACKNOWLEDGEMENT.

First of all I would like to greatly thank my dear parents for their constant encouragement and support through which this project came to its end.

Secondly I would like to thank my close friends from their great help, advise and devotion through which I gained experience and learned the skill of creating infrastructure of this documentation.

Finally I would like to thank my project supervisor Prof. Dr. Fakraddin Mamedov whose notes on Cryptography and Network Security were a great help for the prepairation of this project..

ABSTRACT

This project contains the general information about the security systems and the role of firewalls in any kind of computer networks including Internet. It contains all the necessary details for the use of security system, its working, its reqirements and its deployment. The thesis provides a comprehensive knowledge on network system security, from the lowest level of data transmission i.e. data encryption, authentication, digital signature, the protocols used and the deployment of the firewalls system.

Authentication, Digital signature, Basic protocols used in difference network layers are also analysed.

Table of Contents.

LIST OF ABBREVIATIONS

AES: Advanced Encryption Standard.

AFS: Andrew File System.

API: Application Program Interface.

ARP: Address Resolution Protocol.

ASCII: American Standard Code for Information Interchange.

ATM: Asynchronus Transfer Protocol.

Authantication: A method of confidentiality in which only the authorized person/computer can read the encrypted message.

BGP: Border Gateway Protocol.

Bombs: Kinds of attacks or defects.

Bug: Defect or imperfaction.

CCB: Cipher Block Changing Mode.

Channel: A communication path capable of transmitting data.

CIDR: Classless Inter-Domain Routing.

CIFS: Common Internet File System.

Credentials: Verifying documents.

Daemon: A machine or program that performs a task on behalf of the operator.

Decrytion: The process to change ciphered text into plain text(original form) with the help of a key.

DES: Data Encryption Standard.

Digital Signature: A method of condidentiality in which senders Id is concerned.

DNS: Domain Name System.

DNSsec: Domain Name System security.

ECB: Electronic Code Book.

Encryption: The process to change plain text into ciphered text with the help of a key.

ESP: Encapsulating Security Payload.

Firewalls: In a local area network or on the internet, hardware and software through which all incomming data must through for the purpose of verification and authontication.

FTP: File Transfer Protocol.

Gateway: A device that operates at the transport layer of the OSI model to connect two or more disimilar networks.

GPS: Global Positioning System.

Guest: A trusted user with minimum authority.

H.323: The ITU's Internet telephony protocol.

Host: The administrator or the owner of some entity.

Hacker: An attacker or an adversory.

ICMP: Internet Control Message Protocol.

IDC: International Data Coorporation.

IEEE: Institute of Electric and Electronic Engineering.

IKE: Internet Key Exchange.

IMAP: Remote access to mail box protocol.

IP v6: Internet Protocol version 6(latest).

IP: Internet Protocol.

IPsec: Internet security protocol.

IPv4: Internet Protocol version 4(old).

ISP: Internet Service Provider.

IV: Initialization Vector.

Kerberos Authantication: A security authantication system to validate a principal's identity.

KeyNote: KeyNote provides a simple notation for specifying both local security policies and credentials that can be sent over an untrusted network

Keystream: An algorithm modulated with plain text to change it into cipherd text or from cipherd text to plain text.

LAN: Local Area Network.

Link Layer: OSI model layers.

MAC: Message Authantication Codes.

MDC: Modification Data Codes.

MIB: Management Information Base.

MIME: Multipurpose Internet Mail Extension.

NAT: Network Address Translator.

NFS: Network File System.

NIS: Network Information Service.

NTP: Network Time Protocol.

OpenBSD: Open Berkleley Software Distibution.(UNIX)

P2P: Peer to Peer.

Packet: A chunk of data.

PKI: Public Key Infrastructure.

POP3: Post Office Protocol.

RIP/OSPF: Routing Information Protocol / Open Shortest Path First.

rlogin: Related login passwords.

Router: A machine which is used to transfer data packets from a particular station on a **LAN** to a remote station that is attached to another LAN.

RPC: Remote Procedure Call.

RSA: Rivest Shamir Adleman (public key cryptosystem).

RTP:Real-Time Transport Protocol.

SCTP: Stream Control Transmission Protocol.

SIP: Session Initiation Protocol.

SMB: Server Message Block.

SMTP: Simple Mail Transfer Protocol.

Smurf: An attack that primarily consumes the bandwidth on the access line from an ISP to the target site.

SNMP: Simple Network Management Protocol.

SNMP: Simple Network Management Protocol.

SPD: Security Policy Database

TCP/IP: Transmission Control Protocol/ Internet Protocol.

TFTP: Trivial File Transfer Protocol.

TKIP: Temporal Key Integrity Protocol

UDP: User Datagram Protocol.

URL: Uniform Source Locator.

VPN: Virtual Private Network.

1. INTRODUCTION

With the evolution of the Internet since 1993 use of computers is on the rise in almost every country of the world. Several hundreds of thousands of people connect to the Internet every day using personal computers to use services provided by the Internet. Leading technology companies are providing new means of connecting to the Internet using latest tools and gadgets. People of all sizes, gender and age can use Internet to their benefit for getting information, giving out information or sharing information. Thus it can be said that Internet has become a global medium for sharing information at personal level not only for humans but for other creatures as well.

There can be an enormous use of the internet for humans of different cultures and domains. People can cooperate, communicate, educate and entertain themselves and others. Students learn, teachers teach, researchers research, doctors diagnose, engineers build, marketers sell, buyers purchase, gamers play; and there are loads of people using internet for personal, organizational or national use. As internet can be put to good use by certain people, it is also being abused by others. Certain individuals or groups of people steal or corrupt information at great cost to serious users. These people cause havoc on the internet preventing serious users from accessing valuable information and create security concerns.

Serious measures have been taken to prevent abuse of the internet. Many new methods and techniques and deployed on the internet to deny hacking attempts. Some are great under certain conditions others fail, but fortunately these techniques are applicable to most parts of the internet and have shown promising signs.

This report covers a brief introduction to the information security and the need to keep the flow information secure. Some common security threats and their counter measures are discussed. Firewalls as a protection measure is discussed in detail followed by a case-study of use of firewalls. At the end a conclusion is presented with a proposed Internet security architecture.

1.1 Growth of Internet use

Many surveys and studies have been conducted during past few years about the growth of the use of internet. Since it has become very difficult to find the exact number of the people and computers logging into the internet many surveys suggest that the use of the internet is doubling every six months; however some of these surveys might not be absolutely correct. Nevertheless it is evident that the growth of computer and internet usage is exploding.

Only in the United States, which averages 38% of total Internet users in the world alone, the figures are rising sharply. Figure 1.1 shows the sharp increase in a matter of a few years. In 1994 only about 10% adults connected to the internet occasionally to check emails. About 60% adults are connecting to the internet in 2001. So in a matter of only
5-6 years time almost every other person connects to the internet in any regard.

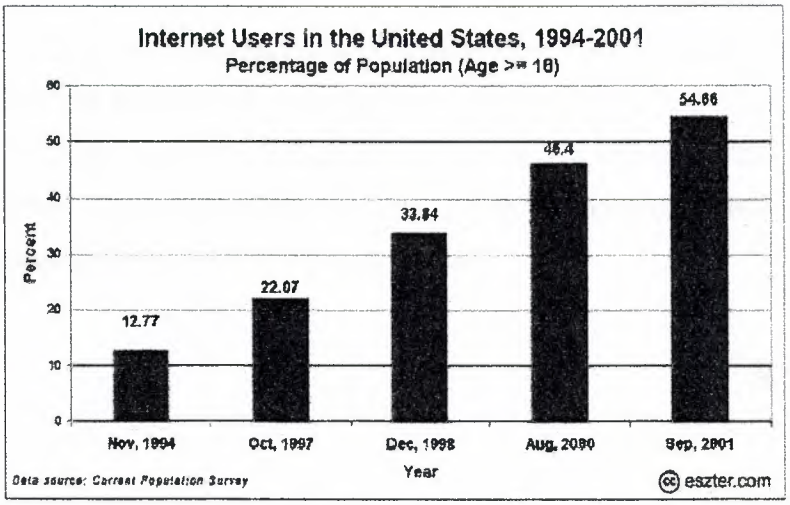


Fig 1.1: Internet Usage Statistics (Source: eszter.com)

Recently a study carried out in [2], suggest that in March 2003 the total internet population was about 650 million users. Out of 650 million users about 58% spoke English; which suggests that Internet has become very popular in other parts of the world as well. Internet Supports many languages allowing many native speakers to browse through the web in their own language. This native-ness of internet is the primary reason for foreign language speakers to connect and browse through the pages.

Internet Use in the world

World Total	605.60 million
Africa	6.31 million
Asia/Pacific	187.24 million
Europe	190.91 million
Middle East	5.12 million
Canada & USA	182.67 million
Latin America	33.35 million

Table 1.1: Internet Use in World continents (in millions) dated September 2002 [2]

Fig 1.2 shows the breakdown of internet users based on their native languages. Its is evident that English is the popular language with 35% of the share; but comparing this with previous data (1994) which shows that English was used 62% as the language of the internet. This suggests that with the growth of non-English Internet community Internet usage in foreign languages is on the rise.

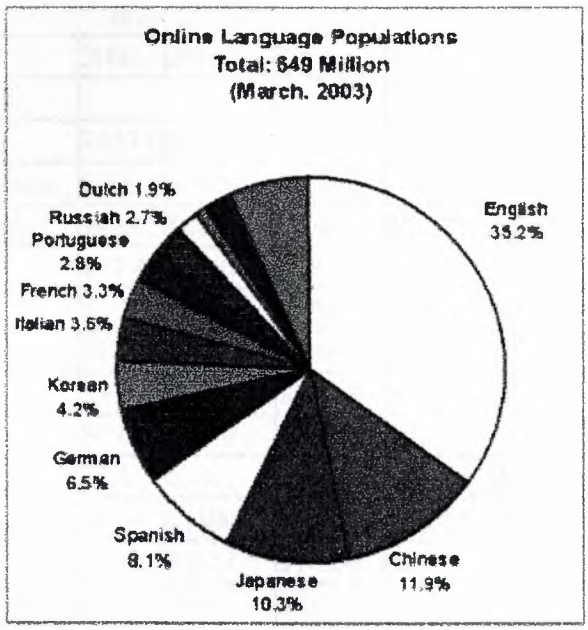


Fig 1.2: Internet Usage Statistics (Source: glreach.com)

One of the major reasons of this sharp internet growth is the E-commerce. Forrester Research group [2] predicts that by 2004, online commerce will reach $6.8 trillion. This huge amount comprises Forrester's projection for both business-to-business and business-to-consumer transactions online. The analyst firm projects that while the United States and North America currently preside over the majority of online transactions, that will shift in the coming years as Asia and European nations become more active.

Table 1.2 shows the estimated figures given by Forrester Research Inc [2]. The online business in 2000 was about 657 billions of dollars which almost doubled in 2001 to 1233 billions of dollars. Due to slump in online markets following the world trade center incident the boom in the online business cooled down yet e-businesses netted almost double of 2001 figures in 2002. It is predicted that in the following years the e-business would be growing and attracting new markets in the different areas of the world.

World Wide E-Commerce Growth

	2000	2001	2002	2003	2004	% of total sales in 2004
Total ($ B)	$657.0	$1,233.6	$2,231.2	$3,979.7	$6,789.8	8.6%
North America	$509.3	$908.6	$1,498.2	$2,339.0	$3,456.4	12.8%
United States	$488.7	$864.1	$1,411.3	$2,187.2	$3,189.0	13.3%
Canada	$17.4	$38.0	$68.0	$109.6	$160.3	9.2%
Mexico	$3.2	$6.6	$15.9	$42.3	$107.0	8.4%
Asia Pacific	$53.7	$117.2	$286.6	$724.2	$1,649.8	8.0%
Japan	$31.9	$64.4	$146.8	$363.6	$880.3	8.4%
Australia	$5.6	$14.0	$36.9	$96.7	$207.6	16.4%
Korea	$5.6	$14.1	$39.3	$100.5	$205.7	16.4%
Western Europe	$87.4	$194.8	$422.1	$853.3	$1,533.2	6.0%
Germany	$20.6	$46.4	$102.0	$211.1	$386.5	6.5%
United Kingdom	$17.2	$38.5	$83.2	$165.6	$288.8	7.1%
France	$9.9	$22.1	$49.1	$104.8	$206.4	5.0%
Italy	$7.2	$15.6	$33.8	$71.4	$142.4	4.3%
Netherlands	$6.5	$14.4	$30.7	$59.5	$98.3	9.2%

Table 1.2 Worldwide E-commerce Growth [2]

1.2 Uses of Internet

Users of the Internet tend to benefit from enormous resources that it offers. The Internet, especially the World Wide Web (WWW), has been the subject of an immense amount of media attention in the past year. Most reports present a few specific examples (see Hubble's latest astronomical pictures, view a movie clip from Universal's latest movie, or check on the latest sports scores) and go on to paint an enthusiastic picture of the present and future usefulness of the Web in glittering generalities for their general audience. Most of the time internet is used in one of the capacities which include academics, industry & commerce, health care and entertainments.

1.2.1 Academics

The Internet was originally developed to facilitate communication and dissemination of information among government and academic institutions in the U.S. It was primarily used for email, academic papers and research findings, and electronic transmission of binary files of various kinds. The advent of the Web and Web browsers

(a program that enables users to view Web content and navigate among Web content pages) has greatly broadened the kinds and mix of information available. Although the original media (text and binary files) are still important, new media (which are binary files, to be sure, but which are presented directly to the user by a Web browser) take up an increasing fraction of the total. These new media include sound, graphic images, and video.

Internet is being used for teaching since a long time as mentioned above. Recently new developments have made possible the availability of online courses. Leading universities like MIT, Harvard and UC Berkeley have been offering online courses for remote learning. This has made possible to learn remotely taking classes while not being on campus. People have luxury of taking classes any time of the day in remotest imaginable places. With the advert of technology Internet can be accessed remotely using mobile devices such as Laptop computers and Personal Digital Assistants. Besides Online courses, student can access lab equipment available in the labs of the university, remotely, allowing them to do experimentation while not physically being in the laboratory. Researchers, Scientists, college professors, doctors and engineers, all find Internet to be a useful resource.

1.2.2 Industry & Commerce

The International Data Corporations (IDC) report says that about 90 percent of all US companies have set up sites on the Web. The companies connected to their customers are finding cost savings of 50 percent to 90 percent in sales, customer support, distribution, and other areas. 80 percent of companies using Intranet applications have seen a positive return on investment, with an average annualized return of 38 percent. The business-to-business commerce on the Internet is growing three times faster than business-to-consumer commerce. Commerce-Net predicts that business-to-business transactions will represent 55 percent of all Internet commerce by 2005.

From an advertising perspective, the Internet is still just one of a number of avenues in which to promote your business and sell your products and services. As with research, each medium of advertising is unique with its own strengths and drawbacks. It is noticed that online businesses can take a better care of their customers. Customer

service is the essential part of business, studies have shown that the companies having a web presence had better customer relations than the one which didn't.

1.2.3 Health Care

A rapidly growing number of Internet sites are dedicated to helping consumers find the information they need to make decisions about their health and health care. Patients are creating online communities that provide peer support, information on the latest research, and personal stories about their experiences. Health care professionals are using the Internet for research, to get access to the latest information in their field, to consult with their colleagues, and to keep in touch with their patients. Almost every health care business from insurer to hospital to pharmaceutical company has a Web site.

Why is the use of the Internet in health care growing so quickly? How sustainable is that growth? What kinds of health-related applications will develop over the next five years? How will the Internet affect health care delivery and health outcomes? All of these questions are answered or being answered by enormous amount of medicine related people and companies.

1.2.4 Entertainment

Internet has emerged as a great source for entertainment providers and entertainers. Tens of hundreds of websites have be put up the internet during the last few years. Entertainment magazines, reviewers and movie making companies such as Disney have all gone online to establish contact with the users and ordinary people. Several sites provide with latest entertainment news and reviews. Information about entertainers, actors, actresses, TV dramas, movies is available for public viewing at various locations. Databases storing information about movies such as imdb.com are popular among movie watchers and reviewers.

Apart from big or small screen entertainment, Sports is a choice of everyone. Many professional sports clubs have put their websites and are online providing various services to their users. Information about traveling, camping, site seeing, latest fashions, clothes, automobiles can all be found on the Internet.

Thus it can be said that Internet is a global medium of obtaining information for all kinds of people, men, women and children without age boundaries. All people regardless of their origin, religion, color or race can access all the information they need and use it as they please.

1.3 Abuses of Internet

Just like Internet can provide as a useful resource, certain domain of people can abuse it. Since it is an open medium without any government or individual influencing its use, it can be classified as a public resource. Because of its openness all sorts of people are allowed to access all parts of it, which makes computers on the Internet vulnerable to malicious attacks from hackers. Some common methods of abusing Internet are given below.

1.3.1 Hacking

Hacking is an act of penetrating computer systems to gain knowledge about the system and how it works. Technically, a hacker is someone who is enthusiastic about computer programming and all things relating to the technical workings of a computer.

However, most people understand a hacker to be what is more accurately known as a cracker. Crackers are people who try to gain unauthorized access to computers. This is normally done through the use of a 'backdoor' program installed on your computer. A lot of crackers also try to gain access to resources through the use of password cracking software, which tries billions of passwords to find the correct one for accessing a computer.

Hackers or Crackers can cause enormous amount of damage to computer systems. This depends upon what backdoor program(s) are hiding on ones PC. Different programs can do different amounts of damage. However, most allow a hacker to smuggle another program onto ones PC. This means that if a hacker can't do something using the backdoor program, he can easily put something else onto your computer that can. Hackers can see everything you are doing, and can access any file on your disk. Hackers can write new files, delete files, edit files, and do practically anything to a file that could be done to a file. A hacker could install several programs on to your system

without your knowledge. Such programs could also be used to steal personal information such as passwords and credit card information.

1.3.2 Virus

The most common question asked by not-so-informative net audience is the definition of a computer virus. Most people agree that Computer virus is a type of legitimate program, which is copied on ones computer and later causes damage. The one outstanding feature of a virus is that it sets out with the aim of reproducing itself. People usually associate viruses with other actions such as damaging a system by destroying data but this is not essential for a program to be classed as a virus. The name was given to this piece of malicious code due to its inherent ability to reproduce itself. So even if you have a piece of code that does nothing harmful to the system but keeps on making copies of itself then it can be branded as a computer virus.

Some viruses hog the computer resources by replication in the memory, while others delete on modify files stored locally. In general viruses can be harmless as observing the user or harmful as deleting or modifying important data. Examples of recent viruses are Melissa, the love bug and Win32.MTX which caused a great deal of losses to computers around the world. Viruses can be removed if Anti-viral programs are installed on computers. Many businesses spend lots of money to prevent their networks from viruses and giving the users a virus-free environment.

1.3.3 Spam

Spam is flooding the Internet with many copies of the same message, in an attempt to force the message on people who would not otherwise choose to receive it. Most spam is commercial advertising, often for dubious products, get-rich-quick schemes, or quasi-legal services. Spam costs the sender very little to send -- most of the costs are paid for by the recipient or the carriers rather than by the sender. Email spam targets individual users with direct mail messages. Email spam lists are often created by scanning Usenet postings, stealing Internet mailing lists, or searching the Web for addresses. Email spams typically cost users money out-of-pocket to receive. Many people - anyone with measured phone service - read or receive their mail while the

meter is running, so to speak. Spam costs them additional money. On top of that, it costs money for ISPs and online services to transmit spam, and these costs are transmitted directly to subscribers.

1.3.4 Fraud

The term Internet fraud refers generally to any type of fraud scheme that uses one or more components of the Internet - such as chat rooms, e-mail, message boards, or Web sites - to present fraudulent solicitations to prospective victims, to conduct fraudulent transactions, or to transmit the proceeds of fraud to financial institutions or to other connected with the scheme. Simply stated, Internet fraud covers criminal behavior that could be prosecuted under the Federal wire or mail fraud statutes or the Federal computer fraud statue. Some Internet fraud might also be prosecuted under state fraud statues.

A few commonly used abuse methods have been described above. Certain people can abuse Internet in other ways like cyber stalking, pornography, fraud, credit card stealing, classified information stealing and many others. Thus it is very necessary to prevent these people from doing such despicable things on the Internet.

1.4 Need for Information safety

"Computer break ins are still on the rise, often accompanied by significant financial losses. The Computer Emergency Response Team's manager says the number of reported violations was 130 in 1990, 800 in 1992, 1,300 in 1993, and 2,300 in 1994. A 1994 survey conducted by more than a 1,000 companies showed 20% reporting financial losses as a result of computer break-ins. A earlier study by USA research cited losses of $164 million in 1991 due to unauthorized intrusions"-- Technology Review, April '95 pg. 33.

When you connect your network to the Internet, it makes accessing any other Internet connected network as easy as accessing another department's Local Area Network across the hall. Whether it is downloading files from a server in Australia, opening a remote terminal connection to a supercomputer in San Diego, or browsing a

product literature from a web server in Germany your network could be broken in to very easily. What most people tend to forget is that it is just as easy for millions of Internet connected people to access your network. With a normal, unsecured Internet connection, outsiders have the ability to, say, access one of your server's file systems, get a console terminal connection to a multi-tasking machine using TELNET, or download files from your system using a variety of means like the FTP protocol.

Some computers such as NT or UNIX based machines boot with these and other lesser known servers like time, talk, finger, etc. that are turned on by default. Which makes it easy for hackers to break in. Now these services are typically password protected. However, someone who wants to break into your site can do so with relative ease.

We can summarize the need of information security as

Issues of Privacy

Users should feel safe in using the Internet. According to the survey done by Deloitte & Touche Consulting Group in 1997, the most common concerns about Internet security are the issues of privacy and security in e-mails, as well as the security of networks.

Confidentiality of Information

Confidential and sensitive information can be stolen or altered if corporations do not take effective measure es to protect their networks from intrusions.

Safety of Business Transactions Over the Internet

Potential customers should feel safe when using the sites for purchases, services or pay-per-use information and entertainment. Since the Internet is a growing distribution medium, the issue of security becomes critical in the context of business transactions.

2. NETWORK SECURITY

2.1 Overview of the Network Security

What is "computer security"? Broadly speaking, security is keeping anyone from doing things you do not want them to do to, with, on, or from your computers or any peripheral devices.

There are a number of aspects we should be aware of before deploying any security mechanism on our insecure environment. These should be

"What resources are we trying to protect?" Is it the CPU cycles?

The answers are not always obvious. At one time as well present, it made a great deal of sense that, a computer time is very expensive in cases like supercomputers or critical machines. An example in such a networked world is, a CPU—or rather, a CPU running certain software with certain configuration files—has a name, an identity, that lets it access other, more critical resources. These are often more sensitive than CPU time. A hacker who compromises or impersonates a host will usually have access to all of its resources: files, storage devices, phone lines, etc.

From a practical perspective, some hackers are most interested in abusing the identity of the host, not so much to reach its dedicated resources, but to launder further outgoing connections to other, possibly more interesting, targets. Others might actually be interested in the data on your machine, whether it is sensitive company material or government secrets.

The answer to this first question will, in general, dictate the host-specific measures that are needed. Machines with sensitive files may require extra levels of passwords or even file encryption.

Similarly, if the target of interest is the outgoing connectivity available, the administrator may choose to require certain privileges for access to the network. Possibly, all such access should be done through a daemon that will perform extra logging.

Often, of course, one wants to protect all such resources, in which case the obvious answer is to stop the attackers at the front door, ***"Nip the evil in the bud"***, i.e., not let them into the computer system in the first place. Such an approach is always a useful start, although it tacitly assumes that one's security problems originate from the outside.
This leads us to our second major question:
"Against whom must the computer systems be defended?"

> General survey within last few years analyze that, techniques that suffice against a teenager with a modem are quite useless against a major intelligence agency. For these people, enhanced password security might do the trick, whereas the latter can and will resort to wiretapping and cryptanalysis, monitoring spurious electronic emissions from your computers and wires, and even "black-bag jobs" aimed at your machine room.

Computer security is not a goal; it is a means toward a goal: information security. When necessary and appropriate, other means should be used as well. The strength of one's computer security defenses should be proportional to the threat from that arena.

Figure 2.1 shows two measures of the growth of the Internet. The top shows a count of hosts detected by automated sweeps of the Internet. The counts for recent years are certainly on the low side of the actual number: there is no reliable technology available to count all the computers connected to a large internet. The lower plot shows the number of networks registered on NSF net over the past few years. The vertical scale on both charts is logarithmic.

These growths are exponential. If there are two million hosts registered, how many people have access to those computers? How many would like to try their hand at hacking, perhaps even as a career?

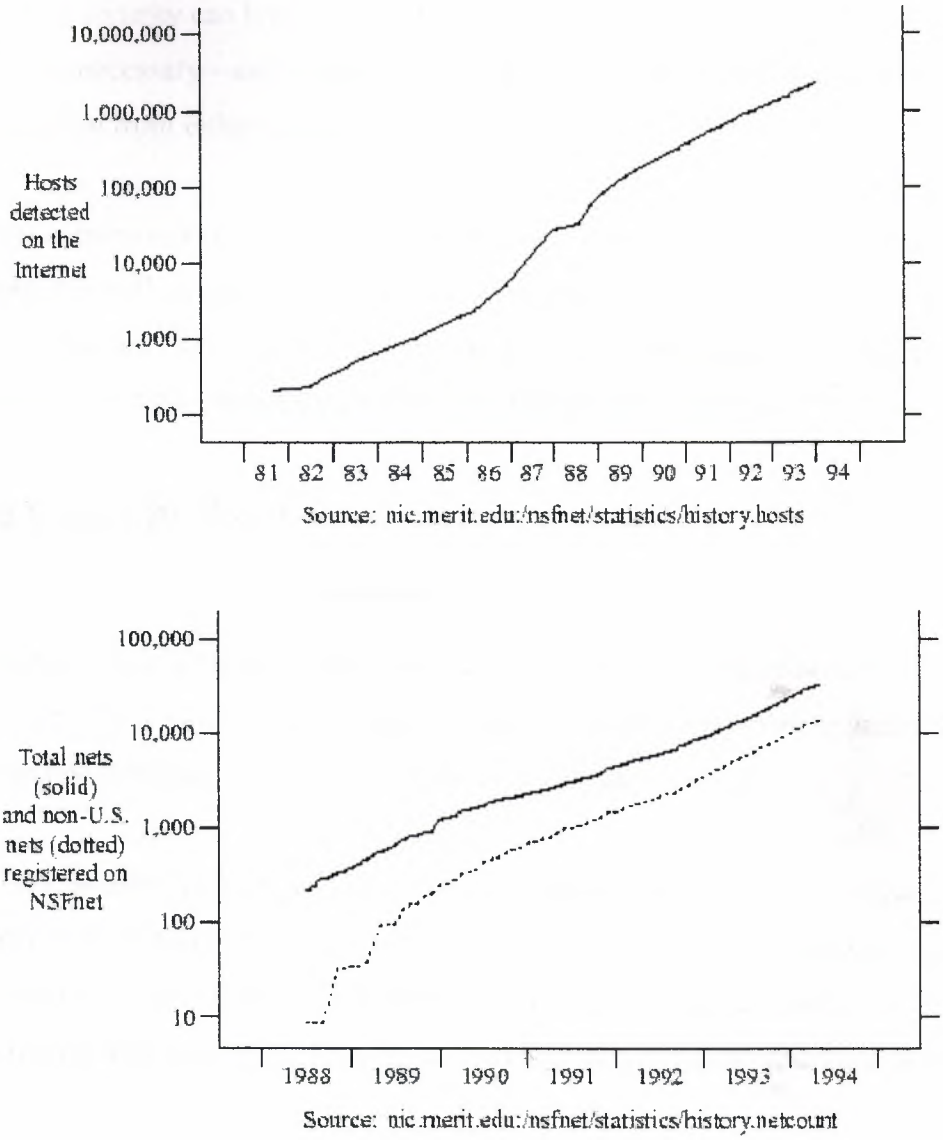


Figure 2.1: The internet Growth.

The third question one must answer before deploying a security mechanism represents the opposite side of the coin: how much security can you afford? Part of the cost of security is direct financial expenditures, such as the extra routers and computers to build a firewall gateway. Often the administrative costs of setting up and running the gateway are overlooked. But there is a more subtle cost, a cost in convenience and productivity, and even morale.

Too much security can hurt as surely as too little can. Finding the proper balance is tricky, but utterly necessary—and it can only be done if you have properly assessed the risk to your organization from either extreme.

One more point is worth mentioning. Even if we do not believe that we have valuable assets, it is still worth keeping hackers out of our machines. We may have a relaxed attitude, but that may not be evident to the attackers. There are far too many cases on record of systems being trashed by hackers who thought they had been detected.

2.2 Picking a Security Policy

"Even paranoids have enemies."

A *security policy* is the set of decisions that, collectively, determines an organization's posture toward security. More precisely, a security policy determines the limits of acceptable behavior, and what the response to violations should be.

Naturally, security policies will differ from organization to organization. An academic department in a university has different needs than a corporate product development organization, which, in turn, differs from a military site. But every organization should have one, at least if only to let it take action when unacceptable events occur.

Defining the limits of acceptable behavior is fundamental to the operation of a firewall.

2.2.1 Implementation of security policy

The first step for this is to decide what is and what is not permitted. To some extent, this process is driven by the business or structural needs of the organization. Some companies wish to restrict outgoing traffic, to guard against employees exporting valuable data. Other aspects may be driven by technological considerations: a specific protocol, though undeniably useful, may not be used, because it cannot be administered securely.

Still others are concerned about employees importing software without proper permission: the company doesn't want to be sued for infringing on someone else's rights.

Making such decisions is clearly an iterative process, and one's answers should never be carved in stone or etched into silicon. Some of the main problems that exists and the axiomatic answers to them are as follows

Axiom 1 (Murphy) *All programs are buggy.*
Theorem 1 (Law of Large Programs)
Large programs are even buggier than their size would indicate.
Proof: By inspection.
Corollary 1.1 *A security-relevant program has security bugs.*

Theorem 2
If you do not run a program, it does not matter whether or not it is buggy.
Proof: As in all logical systems, _ **(false=>true) = true**.

Corollary 2.1 *If you do not run a program, it does not matter if it has security holes.*
Theorem 3
Exposed machines should run as few programs as possible; the ones that are run should be as small as possible.
Proof: Follows directly from Corollaries 1.1 and 2.1.
Corollary 3.1 (Fundamental Theorem of Firewalls)
Most hosts cannot meet our requirements: they run too many programs that are too large. Therefore, the only solution is to isolate them behind a firewall if you wish to run any programs at all.

2.3 Strategies for a Secure Network

2.3.1 Host Security problems

To some people, the very notion of a firewall is anathema. From all our above discussion we can say that in most situations, the network is not the resource at risk; rather, it is the endpoints of the network that are threatened. By analogy, con artists rarely steal phone service *per se*; instead, they use the phone system as a tool to reach their real victims. So it is, in a sense, with network security. Given that the target of the attackers is the hosts on the network, should they not be suitably configured and armored to resist attack?

The answer is that they should be, but probably cannot. Theorem 3 shows that such attempts are probably futile (worthless). There *will* be bugs, either in the network programs or in the administration of the system. It is this way with computer security: the attacker only has to win once. It does not matter how thick are your walls, nor how lofty our battlements; if an attacker finds one weakness say, a postern gate (backdoor), to extend our metaphor—our system *will* be penetrated.

Unfortunately, that is not the end of our woes. By definition, networked machines are not isolated. Typically, other machines will trust them in some fashion. It might be the almost-blind faith of *rlogin*, or it might be the sophisticated cryptographic verification used by the Kerberos authentication in which case a particular user will be trusted.

It doesn't matter—if the intruder can compromise the system, he or she will be able to attack other systems, by taking over either *root*, and hence the system's identity, or some user account. It might seem that we are unduly pessimistic (tendency to look at dark side) about the state of computer security. Nothing in the recent history of either network security or software engineering gives us any reason to believe otherwise. Nor are we alone in feeling this way.

That is, there must be more reason to believe that the system actually functions as designed. Despite those requirements, even the most trusted system, with an **A1** evaluation, is not trusted with the most sensitive information if unclear users have access to the system.

a) Human related problems

Few systems on the Internet meet even the **C2** requirements; their security is not adequate. Another challenge exists that is difficulty of creating secure systems: administering them. No matter how well written the code and how clean the design, later human error can negate (subtract) all of the protections. Consider the following sequence of events:

1. A gateway machine malfunctioned on a holiday weekend, when none of the usual system administrators was available.
2. The backup expert could not diagnose the problem over the phone and needed a guest account created.
3. The operator added the account *guest*, with no password.
4. The expert neglected to add a password.
5. The operator forgot to delete the account.
6. Some university students found the account within a day and told their friends.

Problems related to last case i.e. the penetration can only be discovered when some unwanted guest try to penetrate other gateway machine in the presence of error detection alarm system within the server machine. Some firewalls machines have the capabilities to eliminate such kind of problems or at least probes to the administrators.

Another human operator based technical mistakes exist is on off-the shelf machines that have lots of knobs, buttons, and switches with which to fiddle, and many of the settings are insecure.

b) Vendor based problems

Worse yet, many machines are shipped that way by the vendor; given that higher security generally makes a system less convenient to use and administer, some manufacturers choose to position their products for the "easy-to-use" market. They run old releases of the operating system, with bugs fixed if and only if they directly affect the user population.

2.3.2 Gateways and Firewalls

It should be no surprise that we recommend using firewalls to protect networks. We define a ***firewall*** *as a collection of components placed between two networks that collectively have the following properties*:

- All traffic from inside to outside, and vice-versa, must pass through the firewall.

- Only authorized traffic, as defined by the local security policy, will be allowed to pass.
- The firewall itself is immune to penetration.

We should note that these are design goals; a failure in one aspect does not mean that the collection is not a firewall; simply that it is not a very good one.

2.3.3 Basic Advantages of using Firewalls

Firewalls have several distinct advantages. The biggest single reason that a firewall is likely to be more secure is simply that it is not a general-purpose host. Thus, features that are of doubtful security but add greatly to user convenience—NIS, *rlogin*, etc.—are not necessary. For that matter, many features of unknown security can be omitted (failed) if they are irrelevant (non weight) to the firewall's functionality.

A second benefit comes from having professional administration of the firewall machines. We do not claim that firewall administrators are necessarily more competent than your average system administrator, but they may be more security conscious.

2.3.4 Use of firewalls and gateways in network security environment

Fewer normal users is a help as well. Poorly chosen passwords are a serious risk; if users and their attendant passwords do not exist, this isn't a problem. Similarly, one can make more or less arbitrary (opinions) changes to various program interfaces if that would help security, without annoying a population that is accustomed to a different way of doing things.

Many people resent (feel offended) them, or they may be too expensive to be furnished to an entire organization; a gateway machine, however, should have a restricted enough user community for these concerns are negligible.

More subtly, gateway machines need not, and should not, be trusted by *any* other machines. Thus, even if the gateway machine has been compromised, no others will fall automatically. As for example, other components of the firewall can shield vulnerable services on the gateway machine

On the other hand, the gateway machine can trust other machines, thereby eliminating the need for most passwords on the few accounts it should have. Again, something that is not there cannot be compromised.

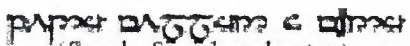
Gateway machines have other, non security advantages as well. They are a central point for mail and FTP administration, for example. Only one machine need be monitored for delayed mail, proper header syntax, return-address rewriting (i.e., to *Firstname.Lastname*@ORG.DOMAIN format), etc. Here outsiders will have a single point of contact for mail problems and a single location to search for files being exported.

2.3.5 Kinds of attacks

For the sake of host security, even if a firewall were impermeable, and even if the administrators and operators never made any mistakes, the Internet is not the only source of danger. Apart from the risk of insider attacks—and in some environments, that is a serious risk—an outsider can gain access by other means.

Strong host security policies are a necessity, not a luxury. For that matter, internal firewalls are a good idea, to protect very sensitive portions of organizational networks.

2.3.6 Protecting Passwords from threats


(Speak, friend, and enter.)

System bugs are the exciting way to crack a system, but they are not the most common attack. That honor is reserved for a rather mundane (worldly) feature: user passwords.

A high percentage of system penetrations occur because of the failure of the entire password system.

We write "password system" because there are several causes of failure. However, the most common problem is that people tend to pick very bad passwords. Repeated studies have shown that password-guessing is likely to succeed. We are not saying that *everyone*

will pick a poor password; however, enough people will, due to this, password-guessing remains a high-probability approach for an attacker.

Password-guessing attacks take two basic forms. The first involves attempts to log in using known or assumed user names and likely guesses at passwords. This succeeds amazingly often.

```
root:DZcoRWR.7DJuU:0:2:0000-Admin(0000):/:
daemon:*:1:1:0000-Admin(0000):/:
bin:*:2:2:0000-Admin(0000):/bin:
sys:*:3:3:0000-Admin(0000):/usr/v9/src:
adm:*:4:4:0000-Admin(0000):/usr/adm:
uucp:*:5:5:0000-uucp(0000):/usr/lib/uucp:
nuucp:*:10:10:0000-uucp(0000):/usr/spool/uucppublic:/usr/lib/uucp/uucico
ftp:anonymous:71:14:file transfer:/:no soap
research:nologin:150:10:ftp distribution account:/forget:/it/baby
ches:La9Cr9ld9qTQY:200:1:me:/u/ches:/bin/sh
dmr:laHheQ.H9iy6I:202:1:Dennis:/u/dmr:/bin/sh
rtm:5bHD/k5k2mTTs:203:1:Rob:/u/rtm:/bin/sh
adb:dcScD6gKF./Z6:205:1:Alan:/u/adb:/bin/sh
td:deJCw4bQcNT3Y:206:1:Tom:/u/td:/bin/sh
```

Figure 2.2: Showing the bogus /etc/passwd file in victim's anonymous FTP area.

Sites often have account-password pairs such as *field*-service, *guest*-guest, etc. These pairs often come out of system manuals. The first try may not succeed, nor even the tenth, but all too often, one will work—and once the attacker is in, our major line of defense is gone. A reminder in this case is that, over 70% attackers use brute force algorithms to find the password access, for their interest in the vice versa site

Regrettably, few operating systems can resist attacks from the inside. This approach should not be possible! Users should not be allowed an infinite number of login attempts with bad passwords, failures should be logged, users should be notified of failed login attempts on their accounts, etc. None of this is new technology, but these things are seldom done, and even more seldom done correctly. Many common mistakes still exists, but few developers have heeded (note) this problem. Worse yet, much of the existing logging on UNIX systems is in *login* and *su*; other programs that use passwords *ftpd*, *rexecd*, various screen-locking programs, etc do not log failures on most systems.

The second way hackers go after passwords is by matching guesses against stolen password files (/etc/passwd on UNIX systems). These may be stolen from a system that is already cracked, in which case the attackers will try the cracked passwords on other machines (psychologically users tend to reuse passwords), or they may be obtained from a system not yet penetrated. These are called *dictionary attacks*, and they are usually very successful. Reports show 25% cracking of the passwords is done due to the stolen passwords.

A third approach is to tap a legitimate terminal session and log the password used. With this approach, it doesn't matter how good a password we have chosen; our account, and probably our system, is compromised.

2.3.7 Research on the password theory

We can draw several conclusions from this. The first, of course, is that user education in how to choose good passwords is vital. Sadly, although almost 15 years have passed since Morris and Thompson's paper on the subject, user habits have not improved much. Nor have tightened system restrictions on allowable passwords helped that much, although there have been a number of attempts.

Others have tried How to enforce password security through retroactive checking. But perversity always tends toward a maximum, and the hackers only have to win once. The only vital solution to this bad passwords choosing dilemma is, that the password file itself be kept out of enemy hands. This means that one should

- Carefully configure the security features for services such as Sun's NIS,
- Restrict files available from *tftpd*, and
- Avoid putting a genuine /etc/passwd file in the anonymous FTP area.

Some UNIX systems provide you with the ability to conceal the hashed passwords from even legitimate users. These features are sometimes called a "shadow" or "adjunct" password file. Many other operating systems wisely hash and hide their password files.

Finally, the biggest risk of all may be our own memory. Do we remember what password we used a year ago?

2.3.8 Encryption

Encryption means, *changing plain text to cipher text using cryptography techniques*, where *cryptography is the study of mathematical techniques related to aspects of network security such as confidentiality, data integrity, entity authentication, and data origin authentication.*

Encryption is often touted as the ultimate weapon in the computer security wars. It is not. It is certainly a valuable tool toward an ultimate goal. Indeed, if encryption is used improperly, it can hurt the real goals of the organization. Some aspects of improper use are obvious. One must pick a strong enough cryptosystem for the situation, or an enemy might cryptanalyze it. As cryptography techniques requires key as a fundamental aspect, the key distribution center must be safeguarded, or all of our secrets will be exposed.

a) Problems using Encryption techniques

Other dangers exist as well. For one thing, *encryption is best used to safeguard file transmission,* rather than file storage, especially if the encryption key is generated from a typed password. Few people bequeath knowledge of their passwords in their wills; more have been known to walk in front of trucks. There are schemes to deal with such situations but these are rarely used in practice. Admittedly, you may not be concerned with the contents of your files after your untimely demise, but your organization, in some sense the real owner of the information you produce at work, might feel differently.

Even without such melodrama, if the machine you use to encrypt and decrypt the files is not physically secure, a determined enemy can simply replace the cryptographic commands with variants that squirrel away a copy of the key.
If a machine is physically and logically secure enough that you can trust the encryption process, encryption is most likely not needed. If the machine is not that secure, encryption may not help.

There is one exception to our general rule: backup tapes. Such tapes rarely receive sufficient protection, and there is never any help from the operating system. One can make a very good case for encrypting the entire tape during the dump process—*if* there is some key storage mechanism guaranteed to permit you to read the year-old backup tape when you realize that you are missing a critical file. It is the *information* that is valuable; if you have lost the contents of a file, it matters little if the cause was a hacker, a bad backup tape, a lost password, or an errant *rm* command.

2.4 Stance (the position of the feet)

The moral of this story is,

Anything you don't understand is dangerous until you do understand it.

LARRY NIVEN

A key decision in the policy is the *stance* of the firewall design. The stance is the attitude of the designers. It is determined by the cost of the failure of the firewall and the designers' estimate of that likelihood. It is also based on the designers' opinions of their own abilities. What people demand is "show me that it's both safe and necessary; otherwise, we won't run it." Those who are completely off the scale prefer to pull the plug on the network, rather than take any risks at all. Such a move is too extreme, but understandable. Why would a company risk losing its secrets for the benefits of network connection? One can best appreciate just how little confidence the U.S. military has in computer security techniques by realizing that connecting machines containing classified data to unsecured networks is forbidden.

We believe our firewall systems are still safe. Compare this approach to a simple packet filter. If the filtering tables are deleted or installed improperly, or if there are bugs in the router software, the gateway may be penetrated. This no fail-safe design is an inexpensive and acceptable solution if your stance allows a somewhat looser approach to gateway security.

We do not advocate disconnection for most sites. One cannot have complete safety; to pursue that chimera is to ignore the costs of the pursuit. Networks and internet works have advantages; to disconnect from a network is to deny oneself those advantages. *When all is said and done, disconnection may be the right choice, but it is a decision that can only be made by weighing the risks against the benefits.*

We advocate caution, not hysteria. For reasons that are spelled out below, we feel that firewalls are an important tool that can minimize the danger, while providing most—but not necessarily all—of the benefits of a network connection. But a paranoid stance is necessary for many sites when setting one up, and we can prove it.

2.4.1 Conclusions on the use of Firewalls

Firewalls must be configured as minimally as possible, to minimize risks. And if risks do not exist, why run a firewall? We forbear to label it an axiom, but it is nevertheless true that some paranoids have real enemies.

Another important point is, for one thing, we feel that any program, no matter how innocuous it seems, can harbor security holes. *We thus have a firm belief that everything is guilty until proven innocent.* Consequently, we configure our firewalls to reject everything, unless we have explicitly made the choice—and accepted the risk to permit it.

Taking the opposite tack, of blocking only known offenders, strikes us as extremely dangerous as the offenders can change their Ids and can attack. Furthermore, whether or not a security policy is formally spelled out, one always exists.

If you do not make explicit decisions, you have made the default decision to allow almost anything.

2.5 The Ethics of Computer Security

Sed quis custodiet ipsos custodes? (But who will guard the guards themselves?)

Satires, VI, line 347
JUVENAL, C. 100 C.E.

At first blush, it seems odd to ask if computer security is ethical. There are several different aspects to the question. From engineering point of view, of course, computer security is a proper goal..

- First, in a technological era, computer security is fundamental to individual privacy. A great deal of very personal information is stored on computers. If these computers are not safe from prying eyes, neither is the data they hold. Worse yet, some of the most sensitive data—credit histories, bank balances, and the like—lives on machines attached to very large networks. We hope that our work will in some measure contribute to the protection of these machines.
- Second, and more important, computer security is a matter of good manners. If people want to be left alone, they should be, whether or not you think their attitude makes sense. Our employer demonstrably wants its computer systems to be left in peace. That alone should suffice, absent an exceedingly compelling reason for feeling otherwise.
- Third, more and more of modern society depend on computers, and on the integrity of the programs and data they contain. These range from the obvious (the financial industry comes to mind) to the ubiquitous (the entire telephone system is controlled by a vast network of computers) to the life-critical (computerized medical devices and medical information systems).

The problems caused by bugs in such systems are legion; the mind boggles at the harm that could be caused—intentionally or not!—by unauthorized changes to any such systems.

Computer security is as important in the information age as once cities were walled a millennium ago. A hackers behave badly is no excuse for us doing the same. We can and must do better, as the rules and the ethics we have been taught are universal.

3. CRYPTOGRAPHY AND ROLE OF FIREWALLS

In previous chapters we have discussed the requirements of security in today's networked world where each and every computer has enormous number of other systems, called neighbors, connected to same one line or by telephone or by wireless communication.

In this chapter we are going to discuss the important apparatus i.e. firewalls and cryptography for network security with details including their kinds.

3.1 Overview of Cryptography

As defined before Cryptography is the study of mathematical techniques related to aspects of network security such as confidentiality, data integrity, entity authentication, and data origin authentication.

The following are the goals of the Cryptography

- Confidentiality is a service used to keep the content of information from all but those authorized to have it. There are numerous approaches to providing confidentiality, ranging from physical protection to mathematical algorithms
- Data integrity is a service which addresses the unauthorized alteration of data. To assure data integrity, one must have the ability to detect data manipulation by unauthorized parties.
- Authentication is a service related to identification. This function applies to both entities and information itself. Aspect of cryptography is usually subdivided into two major classes: entity authentication and data origin authentication.
- Non-repudiation is a service which prevents an entity from denying previous commitments or actions.

A fundamental goal of cryptography is to adequately address these four areas in both theory and practice. Cryptography is about the prevention and detection of cheating and other malicious activities. A number of basic cryptographic tools

(primitives) used to provide network security. Examples of primitives include encryption schemes hash functions, and digital signature schemes

There are two main types of encryption

- Asymmetric encryption, also known as, public -key encryption
- Symmetric encryption

An asymmetric encryption is an encryption system in which the sender and receiver of a message share a single, common key that is used to encrypt and decrypt the message. Symmetric-key systems are simpler and faster, but their main drawback is that the two parties must somehow exchange the key in a secure way. One example of an asymmetric encryption system is the Data Encryption Standard.

Public-key encryption is a cryptographic system that uses two keys opposed to an asymmetric encryption that only uses one common key. The two keys used are -- the public key which is known to everyone and the private or secret key known only to the recipient of the message. An important element to the public key system is that the public and private keys are related in such a way that only the public key can be used to encrypt messages and only the corresponding private key can be used to decrypt them. Moreover, it is virtually impossible to deduce the private key if you know the public key. One example of a public-key encryption system would be Pretty Good Privacy.

3.2 Different types of Cryptosystems/Encryptions:

There are four ways of encryption that we mostly use in network system

- Rivest, Shamir, Adleman, RSA
- Pretty Good Privacy, PGP
- Keyless Encryption, KE
- One-Time Pads, OTP

3.3 Basic Terminology and Concepts in Cryptosystems

The scientific study of any discipline must be built upon exact definitions arising from fundamental concepts. Where appropriate, strictness has been sacrificed for the sake of clarity.

3.3.1. Encryption Domains and Co-domains

- $\mathcal{A}$ denotes a finite set called the alphabet of definition.
- $\mathcal{M}$ denotes a set called the message space. $\mathcal{M}$ consists of strings of symbols from an alphabet. An element of $\mathcal{M}$ is called a plaintext message or simply a plaintext.
- $\mathcal{C}$ denotes a set called the cypertext space. $\mathcal{C}$ consists of strings of symbols from an alphabet; differ from the alphabet of $\mathcal{M}$. An element of $\mathcal{C}$ is called a cypertext.

3.3.2 Encryption and Decryption Transformations

- $\mathcal{K}$ denotes a set called the key space. An element of $\mathcal{K}$ is called a key.
- Each element $e \in \mathcal{K}$ uniquely determines a bijection from $\mathcal{M}$ to $\mathcal{C}$, denoted by $\mathcal{E}e$.
- $\mathcal{D}_d$ denotes a bijection from $\mathcal{C}$ to $\mathcal{M}$ and $\mathcal{D}_d$ is called a decryption function.
- The process of applying the transformation $\mathcal{E}e$ to a message $m \in \mathcal{M}$ is usually referred to as encrypting m or the encryption of m.
- The process of applying the transformation $\mathcal{D}_d$ to a cypertext c is usually referred to as decrypting c or the decryption of c.
- The keys e and d are referred to as a key pair and denoted by (e; d).

3.3.3 Achieving Confidentiality

An encryption scheme may be used as follows for the purpose of achieving confidentiality. Two parties Alice and Bob first secretly choose or secretly exchange a key pair $(e; d)$. At a subsequent point in time, if Alice wishes to send a message $m \in \mathcal{M}$ to Bob, she computes $c = \mathcal{E}e\ (m)$ and transmits this to Bob. Upon receiving c, Bob computes $\mathcal{D}_d(c) = m$ and hence recovers the original message m.

The question arises as to why keys are necessary. If some particular encryption/decryption transformation is exposed then one does not have to redesign the entire scheme but simply change the key. Figure 1.3 provides a simple model of a two-party communication using encryption.

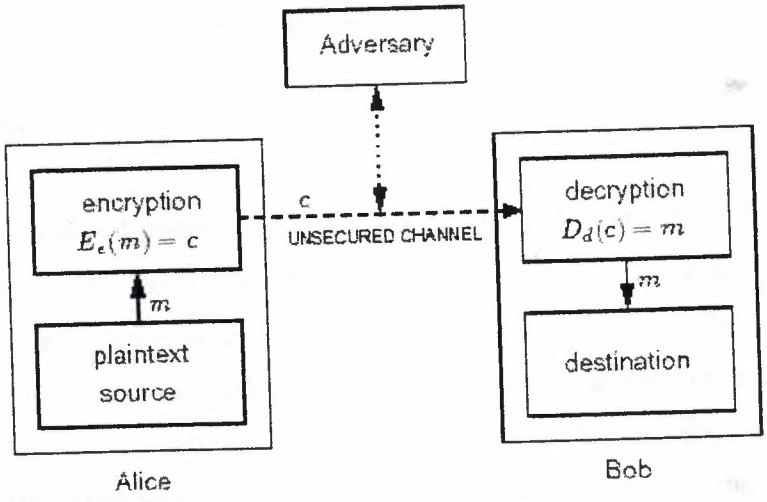


Figure 3.1: Schematic of a two-party communication.

3.3.4 Communication Participants

Referring to Figure 3.1, the following terminology is defined.

- An entity or party is someone or something which sends, receives, or manipulates information. An entity may be a person, a computer terminal, etc.
- A sender is an entity in a two-party communication which is the legitimate transmitter of information.
- A receiver is an entity in a two-party communication which is the intended

recipient of information.

- An adversary is an entity in a two-party communication which is neither the sender nor receiver, and which tries to defeat the information security service being provided between the sender and receiver.

3.3.5. Channels

A channel is a means of conveying information from one entity to another. A physically secure channel is one which is not physically accessible to the adversary. An unsecured channel is one from which parties other than those for which the information is intended can reorder, delete, insert, or read. A secured channel is one from which an adversary does not have the ability to reorder, delete, insert, or read. A secured channel may be secured by physical or cryptographic techniques.

3.3.6 Security

A fundamental principle in cryptography is that the sets $\mathcal{M}$; C; $\mathcal{K}$; $\{E_e: e \in \mathcal{K}\}$, $\{D_d: d \in \mathcal{K}\}$ are public knowledge. When two parties wish to communicate securely using an encryption scheme, the only thing that they keep secret is the particular key pair $(e; d)$, which they must select. One can gain additional security by keeping the class of encryption and decryption transformations secret but one should not base the security of the entire scheme on this approach. An encryption scheme is said to be breakable if a third party, without prior knowledge of the key pair $(e; d)$ can systematically recover plaintext from corresponding cypertext within some appropriate time frame. An encryption scheme can be broken by trying all possible keys to see which one the communicating parties are using. This is called an exhaustive search of the key space.

Frequently cited in the literature are Kerckhoffs' desiderata, a set of requirements for cipher systems. They are given here essentially as Kerckhoffs originally stated them:

1. The system should be, if not theoretically unbreakable, unbreakable in

practice.

2. Compromise of the system details should not inconvenience the correspondents.
3. The key should be remember able without notes and easily changed.
4. The cryptogram should be transmissible by telegraph.
5. The encryption apparatus should be portable and operable by a single person.
6. The system should be easy, requiring neither the knowledge of a long list of rules nor mental strain.

3.3.7 Network Security in General

So far the terminology has been restricted to encryption and decryption with the goal of privacy in mind. Network security is much broader, encompassing such things as authentication and data integrity.

- A network security service is a method to provide specific aspect of security.
- Breaking a network security service implies defeating the objective of the intended service.
- A passive adversary is an adversary who is capable only of reading information from an unsecured channel.
- An active adversary is an adversary who may also transmit, alter, or delete information on an unsecured channel.

3.4 Types of Ciphers

There are two basic types of Ciphers i.e. Block Cipher and Stream Cipher. In this section we are going discuss them and their instances in brief.

3.4.1 Block Ciphers

The most important symmetric algorithms are block ciphers. The general operation of all block ciphers is the same - a given number of bits of plaintext (a block) are encrypted into a block of ciphertext of the same size. Thus, all block

ciphers have a natural block size - the number of bits they encrypt in a single operation. This stands in contrast to stream ciphers, which encrypt one bit at a time. Any block cipher can be operated in one of several modes.

a) Iterated Block Cipher

An iterated block cipher is one that encrypts a plaintext block by a process that has several rounds. In each round, the same transformation or round function is applied to the data using a subkey. The set of subkeys are usually derived from the user-provided secret key by a key schedule. The number of rounds in an iterated cipher depends on the desired security level and the consequent trade-off with performance. In most cases, an increased number of rounds will improve the security offered by a block cipher, but for some ciphers the number of rounds required to achieve adequate security will be too large for the cipher to be practical or desirable.

b) Electronic Codebook (ECB) Mode

ECB is the simplest mode of operation for a block cipher. The input data is padded out to a multiple of the block size, broken into an integer number of blocks, each of which is encrypted independently using the key. In addition to simplicity, ECB has the advantage of allowing any block to be decrypted independently of the others. Thus, lost data blocks do not affect the decryption of other blocks. The disadvantage of ECB is that it aids known-plaintext attacks. If the same block of plaintext is encrypted twice with ECB, the two resulting blocks of ciphertext will be the same.

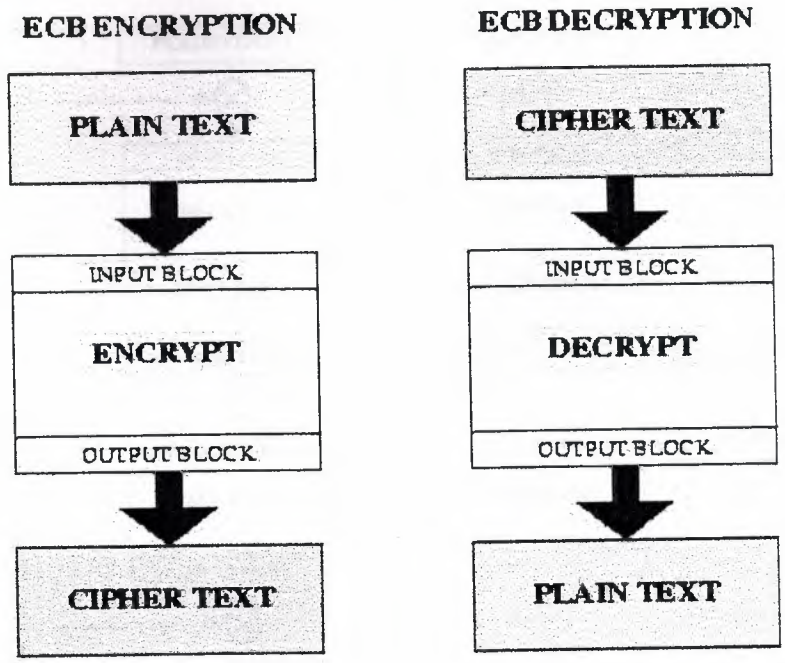


Figure 3.2: Shows a ECB Encryption/Decryption Model

c) Cipher Block Chaining (CBC) Mode

CBC is the most commonly used mode of operation for a block cipher. Prior to encryption, each block of plaintext is XOR-ed with the prior block of ciphertext. After decryption, the output of the cipher must then be XOR-ed with the previous ciphertext to recover the original plaintext. The first block of plaintext is XOR-ed with an initialization vector (IV), which is usually a block of random bits transmitted in the clear. CBC is more secure than ECB because it effectively scrambles the plaintext prior to each encryption step. Since the ciphertext is constantly changing, two identical blocks of plaintext will encrypt to two different blocks of ciphertext. The disadvantage of CBC is that the encryption of a data block becomes dependent on all the blocks prior to it. A lost block of data will also prevent decoding of the next block of data. CBC can be used to convert a block cipher into a hash algorithm. To do this, CBC is run repeatedly on the input data, and all the ciphertext is discarded except for the last block, which will depend on all the data blocks in the message. This last block becomes the output of the hash function, discussed latter.

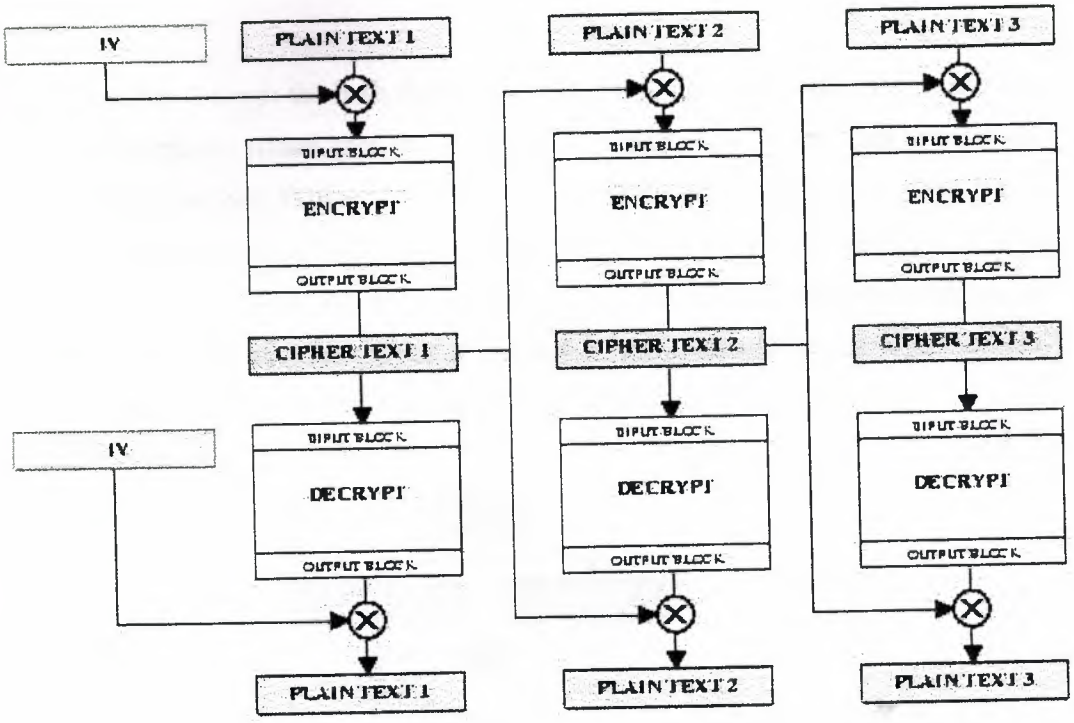


Figure 3.3: Shows a CBC Encryption/Decryption Model

3.4.2 Feistel Ciphers

The figure below shows the general design of a Feistel cipher, a scheme used by almost all modern block ciphers. The input is broken into two equal size blocks, generally called left (L) and right (R), which are then repeatedly cycled through the algorithm. At each cycle, a hash function (f) is applied to the right block and the key, and the result of the hash is XOR-ed into the left block. The blocks are then swapped. The XOR-ed result becomes the new right block and the unaltered right block becomes the left block. The process is then repeated a number of times.

The hash function is just a bit scrambler. The correct operation of the algorithm is not based on any property of the hash function, other than it is completely deterministic; i.e. if it's run again with the exact same inputs, identical output will be produced. To decrypt, the ciphertext is broken into L and R blocks, and the key and the

R block are run through the hash function to get the same hash result used in the last cycle of encryption; notice that the R block was unchanged in the last encryption cycle. The hash is then XOR-ed into the L block to reverse the last encryption cycle, and the process is repeated until all the encryption cycles have been backed out. The security of a Feistel cipher depends primarily on the key size and the irreversibility of the hash function. Ideally, the output of the hash function should appear to be random bits from which nothing can be determined about the input(s).

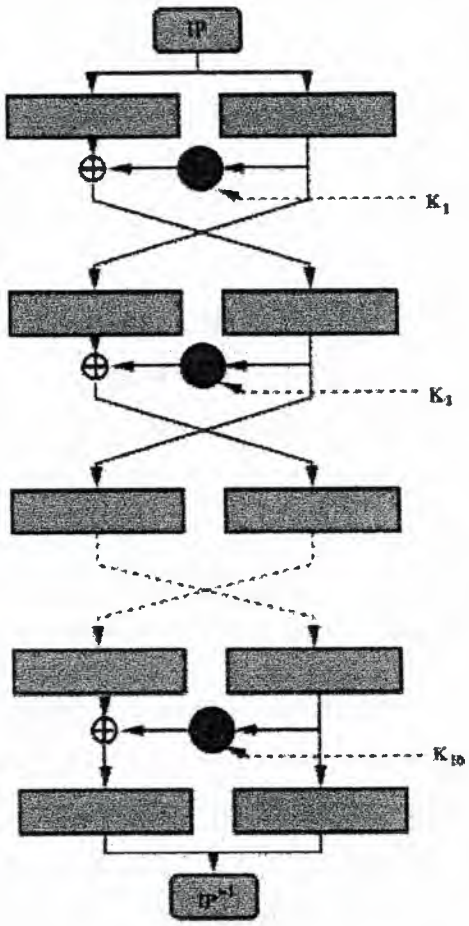


Figure 3.4: Shows a Feistel Model

3.4.3 Data Encryption Standard (DES)

DES is a Feistel-type Substitution-Permutation Network (SPN) cipher. DES

uses a 56-bit key which can be broken using brute-force methods, and is now considered obsolete. A 16 cycle Feistel system is used, with an overall 56-bit key permuted into 16 48-bit subkeys, one for each cycle. To decrypt, the identical algorithm is used, but the order of subkeys is reversed. The L and R blocks are 32 bits each, yielding an overall block size of 64 bits. The hash function "*f*", specified by the standard using the so-called "S-boxes", takes a 32-bit data block and one of the 48-bit subkeys as input and produces 32 bits of output. Sometimes DES is said to use a 64-bit key, but 8 of the 64 bits are used only for parity checking, so the effective key size is 56 bits.

a) Triple DES

Triple DES was developed to address the obvious flaws in DES without designing a whole new cryptosystem. Triple DES simply extends the key size of DES by applying the algorithm three times in succession with three different keys. The combined key size is thus 168 bits (3 times 56), beyond the reach of brute-force techniques such as those used by the EFF DES Cracker. Triple DES has always been regarded with some suspicion, since the original algorithm was never designed to be used in this way, but no serious flaws have been uncovered in its design, and it is today a viable cryptosystem used in a number of Internet protocols.

3.4.4 Stream Ciphers

A stream cipher is a symmetric encryption algorithm. Stream ciphers can be designed to be exceptionally fast, much faster in fact than any block cipher. While block ciphers operate on large blocks of data, stream ciphers typically operate on smaller units of plaintext, usually bits. The encryption of any particular plaintext with a block cipher will result in the same ciphertext when the same key is used. With a stream cipher, the transformation of these smaller plaintext units will vary, depending on when they are encountered during the encryption process.

A stream cipher generates what is called a keystream and encryption is provided by combining the keystream with the plaintext, usually with the bitwise XOR operation. The generation of the keystream can be independent of the plaintext

and ciphertext or it can depend on the data and its encryption.

Current stream ciphers are most commonly attributed to the appealing of theoretical properties of the one-time pad, but there have been no attempts to standardize on any particular stream cipher proposal as has been the case with block ciphers. Interestingly, certain modes of operation of a block cipher effectively transform it into a keystream generator and in this way; any block cipher can be used as a stream cipher. However, stream ciphers with a dedicated design are likely to be much faster.

a) Linear Feedback Shift Register

A Linear Feedback Shift Register (LFSR) is a mechanism for generating a sequence of binary bits. The register consists of a series of cells that are set by an initialization vector that is, most often, the secret key. The behavior of the register is regulated by a clock and at each clocking instant, the contents of the cells of the register are shifted right by one position, and the XOR of a subset of the cell contents is placed in the leftmost cell. One bit of output is usually derived during this update procedure.

LFSRs are fast and easy to implement in both hardware and software. With a sensible choice of feedback taps the sequences that are generated can have a good statistical appearance. However, the sequences generated by single LFSRs are not secure because a powerful mathematical framework has been developed over the years which allows for their straightforward analysis. However, LFSRs are useful as building blocks in more secure systems.

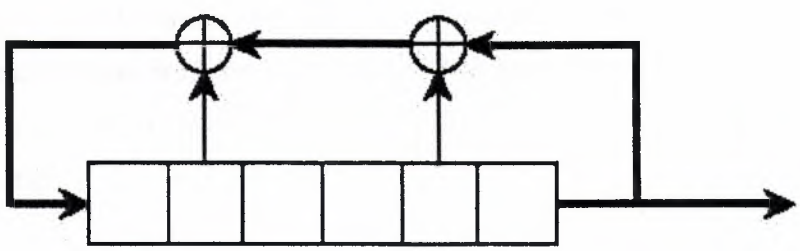


Figure 3.5: Shows a Linear Feed Back Register Model

i.) Shift Register Cascades

A shift register cascade is a set of LFSRs connected together in such a way that the behavior of one particular LFSR depends on the behavior of the previous LFSRs in the cascade. This dependent behavior is usually achieved by using one LFSR to control the clock of the following LFSR. For instance one register might be advanced by one step if the preceding register output is 1 and advanced by two steps otherwise. Many different configurations are possible and certain parameter choices appear to offer very good security.

ii) Shrinking and Self-Shrinking Generators

It is a stream cipher based on the simple interaction between the outputs from two LFSRs. The bits of one output are used to determine whether the corresponding bits of the second output will be used as part of the overall keystream. The shrinking generator is simple and scaleable, and has good security properties. One drawback of the shrinking generator is that the output rate of the keystream will not be constant unless precautions are taken. A variant of the shrinking generator is the self-shrinking generator, where instead of using one output from one LFSR to "shrink" the output of another, the output of a single LFSR is used to extract bits from the same output.

b) Other Stream Ciphers

There are a vast number of alternative stream ciphers that have been proposed in cryptographic literature as well as an equally vast number that appear in implementations and products world-wide. Many are based on the use of LFSRs since such ciphers tend to be more amenable to analysis and it is easier to assess the security that they offer.

There are essentially four distinct approaches to stream cipher design. The first

is termed the information-theoretic approach explained in one-time pad. The second approach is that of system-theoretic design. In essence, the cryptographer designs the cipher along established guidelines which ensure that the cipher is resistant to all known attacks. While there is, of course, no substantial guarantee that future cryptanalysis will be unsuccessful, it is this design approach that is perhaps the most common in cipher design. The third approach is to attempt to relate the difficulty of breaking the stream cipher to solving some difficult problem. This complexity-theoretic approach is very appealing, but in practice the ciphers that have been developed tend to be rather slow and impractical. The final approach is that of designing a randomized cipher. Here the aim is to ensure that the cipher is resistant to any practical amount of cryptanalytic work rather than being secure against an unlimited amount of work.

c) One-time Pad

A one-time pad, sometimes called the Vernam cipher, uses a string of bits that is generated completely at random. The keystream is the same length as the plaintext message and the random string is combined using bitwise XOR with the plaintext to produce the ciphertext. Since the entire keystream is random, an opponent with infinite computational resources can only guess the plaintext if he sees the ciphertext. Such a cipher is said to offer perfect secrecy and the analysis of the one-time pad is seen as one of the cornerstones of modern cryptography.

3.5 Attacks on Ciphers

Here are the different kinds of possible attacks what have been observed so for and can be expected are explained in brief.

3.5.1 Exhaustive Key Search

Exhaustive key search, or brute-force search, is the basic technique of trying every possible key in turn until the correct key is identified. To identify the correct key it may be necessary to possess a plaintext and its corresponding ciphertext, or if the

plaintext has some recognizable characteristic, ciphertext alone might suffice.

Exhaustive key search can be mounted on any cipher and sometimes a weakness in the key schedule of the cipher can help improve the efficiency of an exhaustive key search attack. Advances in technology and computing performance will always make exhaustive key search an increasingly practical attack against keys of a fixed length.

3.5.2 Differential Cryptanalysis

Differential cryptanalysis is a type of attack that can be mounted on iterative block ciphers. Differential cryptanalysis is basically a chosen plaintext attack and relies on an analysis of the evolution of the differences between two related plaintexts as they are encrypted under the same key

3.5.3 Linear Cryptanalysis

Linear cryptanalysis is a known plaintext attack and uses a linear approximation to describe the behavior of the block cipher. Given sufficient pairs of plaintext and corresponding ciphertext, bits of information about the key can be obtained and increased amounts of data will usually give a higher probability of success.

There have been a variety of enhancements and improvements to the basic attack. Differential-linear cryptanalysis is an attack which combines elements of differential cryptanalysis with those of linear cryptanalysis. A linear cryptanalytic attack using multiple approximations might allow for a reduction in the amount of data required for a successful attack.

3.5.4 Algebraic Attacks

Algebraic attacks are a class of techniques which rely for their success on some block cipher exhibiting a high degree of mathematical structure. For instance, it is conceivable that a block cipher might exhibit what is termed a group structure.

3.5.5 Data Compression Used With Encryption

Data compression removes redundant character strings in a file. This means that the compressed file has a more uniform distribution of characters. In addition to providing shorter plaintext and ciphertext, which reduces the amount of time needed to encrypt, decrypt and transmit a file, the reduced redundancy in the plaintext can potentially hinder certain cryptanalytic attacks.

By contrast, compressing a file after encryption is inefficient. The ciphertext produced by a good encryption algorithm should have an almost statistically uniform distribution of characters. As a consequence, a compression algorithm should be unable to find redundant patterns in such text and there will be little, if any, data compression. In fact, if a data compression algorithm is able to significantly compress encrypted text, then this indicates a high level of redundancy in the ciphertext which, in turn, is evidence of poor encryption.

3.6 Authentication

In computer networks the communicating parties share not only the media, but also the set of rules on how to communicate. These rules, or protocols, have become more and more important in communication networks and distributed computing. However, the increase of the knowledge of the communication protocols has also brought up the question of how to secure the communication against intruders. To solve this, a large number of cryptographic protocols have been produced.

Cryptographic protocols were developed to combat against various attacks of intruders in computer networks. Nowadays, the comprehension is that the security of data should rely on the underlying cryptographic technology, and that the protocols should be open and available. However, many protocols have been found to be vulnerable to attacks that do not require breaking the encryption, but instead manipulate the messages in the protocol to gain some advantage. The advantages range from the compromise of confidentiality to the ability to impersonate another user.

As there are different protocol designs decisions appropriate to different circumstances, there also exists a variety of authentication protocols. Protocols often differ in their final states, and sometimes they even depend on assumptions that one would not care to make. To understand what is really accomplished with such a protocol, a formal description method is needed. The goal of the logic of authentication is to formally describe the knowledge and the beliefs of the parties involved in authentication, the evolution of the knowledge and the beliefs while analyzing the protocol step by step. After the analysis, all the final states of the protocol are set out.

3.7 Digital Signatures

A cryptographic primitive who is fundamental in authentication, authorization, and non-repudiation is the digital signature. The purpose of a digital signature is to provide a means for an entity to bind its identity to a piece of information. The process of signing entails transforming the message and some secret information held by the entity into a tag called a signature.

3.7.1 Nomenclature and Set-up

The transformations $S_{\mathcal{A}}$ and $\mathcal{V}_{\mathcal{A}}$ provide a digital signature scheme for $\mathcal{A}$.

- $\mathcal{M}$ is the set of messages which can be signed.
- S is a set of elements called signatures, possibly binary strings of a fixed length.
- $S_{\mathcal{A}}$ is a transformation from the message set $\mathcal{M}$ to the signature set S, and is called a signing transformation for entity $\mathcal{A}$.
- $\mathcal{V}_{\mathcal{A}}$ is a transformation from the set $\mathcal{M} \times S$ to the set {true, false} $\mathcal{V}_{\mathcal{A}}$ is called a verification transformation for $\mathcal{A}$'s signatures, is publicly known, and is used by other entities to verify signatures created by $\mathcal{A}$.

3.8 Hash Functions

One of the fundamental primitives in modern cryptography is the cryptographic hash function, often informally called a one-way hash function. A simplified definition for the present discussion follows. A hash function is a

computationally efficient function mapping binary strings of arbitrary length to binary strings of some fixed length, called hash-values. For a hash function which outputs n-bit hash-values and has desirable properties, the probability that a randomly chosen string gets mapped to a particular n-bit hash-value (image) is 2^{-n}. The basic idea is that a hash-value serves as a compact representative of an input string. To be of cryptographic use, a hash function *h* is typically chosen such that it is computationally infeasible to find two distinct inputs which hash to a common value and that given a specific hash-value y, it is computationally infeasible to find an input x such that *h*(x) = y. The most common cryptographic uses of hash functions are with digital signatures and for data integrity Hash functions are typically publicly known and involve no secret keys. When used to detect whether the message input has been altered, they are called modification detection codes (MDCs). Related to these are hash functions which involve a secret key, and provide data origin authentication as well as data integrity; these are called message authentication codes (MACs).

3.9 Trusted computing

Trusted computing *(TC)* is an advanced technology that integrates data security. It is implemented in the core operations of the network rather than implementing it via add on application. It works simply as it cryptographically seal off the parts of the computer that deal with data and applications and give decryption keys only to programs and information that the technology judges to be trustworthy.

3.9.1 Cryptographic Trusted computing

Cryptographic TC technology protects data and programs on user's computers by sealing them in an encrypted virtual vault. If outside data or program wants access to the vault, they must pass muster with the TC system and obtain decryption keys. Only trusted processes would gain access to disk storage; the CPU memory space, including the stack and on chip cache; and main memory.

TC system doesn't actually decide whether code is safe. Instead, they identify users, their computing systems (based on unique identifying digital signatures), and

the application or data they want to run. Trusted agents would provide much of the information. The agents identify he users and their computers to TC systems, which would then consult directory services to determine whether the users are authorized to run the application and data on their systems, if the material is on the source deemed in advance to the trustworthy, and what level of access it should have to system resources.

Trusted worthy computing uses technologies such as digital certificates and public key infrastructure to authenticate participants and provide cryptographic keys.

For maximum protection, TC system would encrypt data not only as it moves from machine to machine but also as it moves between machine components such as the video card and the monitor.

This would thus address two thorny PC-security problems: users getting encrypted data from the Web but storing it unencrypted locally, leaving the information vulnerable; and hackers installing applications such as keystroke loggers or screen capture software on PCs to gain access to stored data.

3.9.2 Trusted Computing Initiatives

The major TC initiatives differ primarily in where the encryption/decryption functionality occurs. In NGSCB and La Grande, it is incorporate in the main CPU, thereby avoiding the problems of unencrypted data going over the data bus o dedicated processor. However, this would require a new CPUs that have the encryption/decryption functionality built in.

3.10 Classes of Attacks and Security Models

Over the years, many different types of attacks on cryptographic primitives and protocols have been identified. The attacks these adversaries can mount may be classified as follows:

1. A passive attack is one where the adversary only monitors the communication channel. A passive attacker only threatens confidentiality of data.

2. An active attack is one where the adversary attempts to delete, add, or in some other way alter the transmission on the channel.

A passive attack can be further subdivided into more specialized attacks for deducing plaintext from ciphertext.

3.10.1 Attacks on Encryption Schemes

The objective of the following attacks is to systematically recover plaintext from ciphertext, or even more drastically, to deduce the decryption key.

1. A ciphertext-only attack is one where the adversary tries to deduce the decryption key or plaintext by only observing ciphertext.
2. A known-plaintext attack is one where the adversary has a quantity of plaintext and corresponding ciphertext.
3. A chosen-plaintext attack is one where the adversary chooses plaintext and is then given corresponding ciphertext.
4. An adaptive chosen-plaintext attack is a chosen-plaintext attack wherein the choice of plaintext may depend on the ciphertext received from previous requests.
5. A chosen-ciphertext attack is one where the adversary selects the ciphertext and is then given the corresponding plaintext. One way to mount such an attack is for the adversary to gain access to the equipment used for decryption
6. An adaptive chosen-ciphertext attack is a chosen-ciphertext attack where the choice of ciphertext may depend on the plaintext received from previous requests.

3.11 Firewall History

We are used to firewalls in other disciplines, and, in fact, the term did not originate with the Internet. Firewalls are barriers to fire, meant to slow down its spread until the fire department can put it out. In the case of computer firewalls we can define it as a single point between two or more networks where all traffic must pass

(choke point); traffic can be controlled by and may be authenticated through the device, and all traffic is logged.

In the past firewalls were effective, but limited. It was often very difficult to get the filtering rules right, for example. In some cases, it was difficult to identify all the parts of an application that needed to be restricted. In other cases, people would move around and the rules would have to be changed.

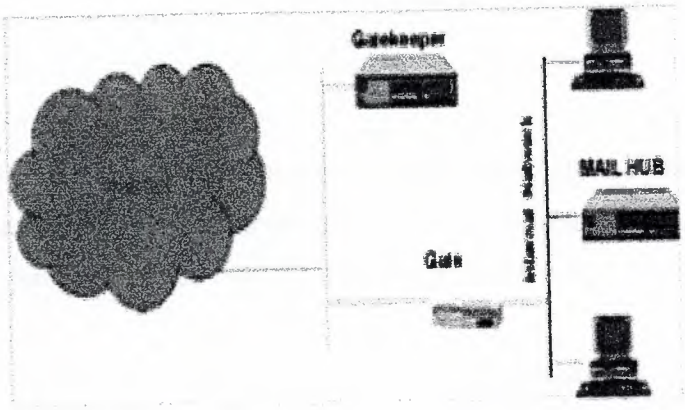


Figure 3.6: DEC SEAL--First Commercial Firewall

Early firewall requirements were easy to support because they were limited to the Internet services available at that time. The typical organization or business connecting to the Internet needed secure access to remote terminal services (Telnet), file transfer (File Transfer Protocol [FTP]), electronic mail (Simple Mail Transfer Protocol [SMTP]), and USENET News (the Network News Transfer Protocol--NNTP). To-day, we add to this list of "requirements" access to the World Wide Web, live news broadcasts, weather information, stock quotes, music on demand, audio and videoconferencing, telephony, database access, file sharing, and the list goes on.

3.12 Types of Firewalls

There are four types of Internet firewalls, or, to be more accurate, three types plus a hybrid. In this part of the chapter we are going to discuss all of these kinds briefly.

3.12.1 Packet Filtering

One kind of firewall is a packet filtering firewall. Filtering firewalls screen packets based on addresses and packet options. They operate at the IP packet level and make security decisions (really, "to forward, or not to forward this packet, that is the question") based on the headers of the packets.

The filtering firewall has three subtypes:

- Static Filtering, the kind of filtering most routers implement--filter rules that must be manually changed
- Dynamic Filtering, in which an outside process changes the filtering rules dynamically, based on router-observed events (for example, one might allow FTP packets in from the outside, if someone on the inside requested an FTP session)
- Stateful Inspection, a technology that is similar to dynamic filtering, with the addition of more granular examination of data contained in the IP packet

Dynamic and stateful filtering firewalls keep a dynamic state table to make changes to the filtering rules based on events.

3.12.2 Circuit Gateways

Circuit gateways operate at the network transport layer. Again, connections are authorized based on addresses. Like filtering gateways, they (usually) cannot look at data traffic flowing between one network and another, but they do prevent direct connections between one network and another.

3.12.3 Application Gateways

Application gateways or proxy-based firewalls operate at the application level and can examine information at the application data level. They can make their decisions based on application data, such as commands passed to FTP, or a URL passed to HTTP. It has been said that application gateways "break the client/server

model."

3.12.4 Hybrids

Hybrid firewalls, as the name implies, use elements of more than one type of firewall. Hybrid firewalls are not new. The first commercial firewall, DEC SEAL, was a hybrid, using proxies on a bastion host (a fortified machine, labeled "Gatekeeper" in Figure 3.6), and packet filtering on the gateway machine ("Gate"). Hybrid systems are often created to quickly add new services to an existing firewall. One might add a circuit gateway or packet filtering to an application gateway firewall, because it requires new proxy code to be written for each new service provided. Or one might add strong user authentication to a stateful packet filter by adding proxies for the service or services.

No matter what the base technology, a firewall still basically acts as a controlled gateway between two or more networks through which all traffic must pass. A firewall enforces a security policy and it keeps an audit trail.

3.13 What a Firewall Can Do

A firewall intercepts and controls traffic between networks with differing levels of trust. It is part of the network perimeter defense of an organization and should enforce a network security policy.

A firewall is a good place to support strong user authentication as well as private or confidential communications between firewalls because firewalls are an excellent place to focus security decisions and to enforce a network security policy. They are able to efficiently log inter-network activity, and limit the exposure of an organization.

The exposure to attack is called the "zone of risk." If an organization is connected to the Internet without a firewall (Figure 3.7), every host on the private network can directly access any resource on the Internet. Or to put it as a security officer might, every host on the Internet can attack every host on the private network.

Reducing the zone of risk is better. An inter-network firewall allows us to limit the zone of risk. As we see in Figure 3.3, the zone of risk becomes the firewall system itself. Now every host on the Internet can attack the firewall. With this situation, we should take Mark Twain's advice to "Put all your eggs in one basket--and watch that basket."

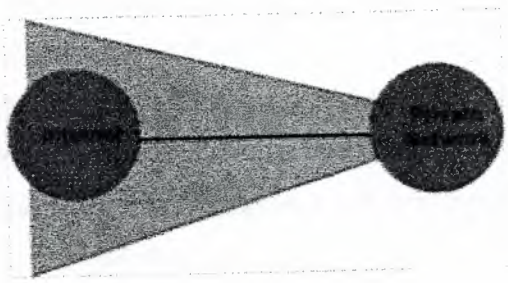


Figure 3.7: Zone of Risk for an Unprotected Private Network

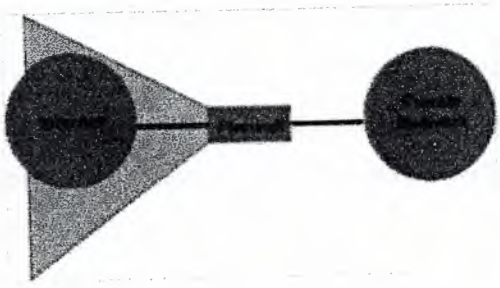


Figure 3.8: Zone of Risk with a Firewall

3.14 What a Firewall Cannot Do

Firewalls are terrible at reading people's minds or detecting packets of data with "bad intent." They often cannot protect against an insider attack. Firewalls also cannot protect connections that do not go through the firewall. In other words, if someone connects to the Internet through a desktop modem and telephone, all bets are off. Firewalls provide little protection from previously unknown attacks, and typically provide poor protection against computer viruses.

3.15 Firewalls Today

The first add-on to Internet firewalls was strong user authentication. If our security policy allows access to the private network from an outside network, such as the Internet, some kind of user authentication mechanism is required. User authentication simply means "to establish the validity of a claimed identity." A username and password provides user authentication, but not strong user authentication. On a non-private connection, such as an unencrypted connection over the Internet, a username and password can be copied and replayed. Strong user authentication uses cryptographic means, such as certificates, or uniquely keyed cryptographic calculators. These certificates prevent "replay attacks"--where, for example, a username and password are captured and "replayed" to gain access. Because of where it sits--on both the "trusted" and "untrusted" networks--and because of its function as a controlled gateway, a firewall is a logical place to put this service.

The next add-on to Internet firewalls was firewall-to-firewall encryption, first introduced on the ANS InterLock Firewall. Today, such an encrypted connection is known as a Virtual Private Network, or VPN. It is "private" through the use of cryptography. It is "virtually" private because the private communication flows over a public network--the Internet, for example. Although VPNs were available before firewalls via encrypting modems and routers, they came into common use running on firewalls. Today, most people expect a firewall vendor to offer a VPN option. Firewalls act as the endpoint for VPNs between the enterprise and mobile users or telecommuters, keeping communication confidential from notebook PC, home desktop, or remote office.

In the past few years, it has become popular for firewalls to also act as content screening devices. Some additions to firewalls in this area include virus scanning, URL screening, and key word scanners. If the security policy of your organization mandates screening for computer viruses--and it should--it makes sense to put such screening at a controlled entry point for computer files, such as the firewall. In fact, standards exist for plugging antivirus software into the data flow of the firewall, to intercept and analyze data files. Likewise, URL screening--firewall controlled access to the World Wide Web--and content screening of files and messages seem like logical additions to a firewall. After all, the data is flowing through the fingers of the

firewall system, so why not examine it and allow the firewall to enforce the security policies of the organization? The downside to this scenario is performance. Also virus scanning must ultimately be performed on each desktop because data may come in to the desktops from paths other than through the firewall—for instance, the floppy.

3.16 Firewalls and VPNs

Many firewalls have some kind of VPNs — encrypted firewall-to-firewall tunnels. All traffic between one firewall and another is encrypted, stuck inside of another IP packet, and sent over the Internet. At the remote site, the firewall pulls the encrypted payload out of the IP packet and decrypts it to get the original IP packet, which is forwarded to the final destination.

3.16.1 Firewall-to-Firewall With Controlled Access

As VPNs become more widespread on the Internet, and VPN establishment more automatic, most VPNs will be for privacy between sites, without a complete trust relationship between those sites.. Privacy for the communications may be desirable or needed, but an Internet firewall can be used to control or prohibit access to the internal, private network.

3.16.2 Firewall-to-Firewall With Open Access

A common configuration for VPNs between Internet Firewalls currently is in a trust relationship between offices of the same company.

With VPN connectivity using VPN-enabled Internet Firewalls between the sites, all traffic is encrypted, and so private. Additionally, when we add the trust derived from all sites being administered by the same organization, all having the same security policy implemented, and all being under the same management organization, we can, over this VPN, allow all network services. In this way we are virtually going around the firewall, though actually the communications flow is still under the protection of the firewalls. In this way we *extend* the network security perimeter to include those other offices. All sites are now *virtually* on the same LAN, with a *virtual network perimeter.*

3.16.3 Firewall-to-Remote System

The same sort of VPN technology can be used between a firewall and a single site. Often this is used to allow private access to a corporate from mobile users, working at a customer site, or connected in from home or a hotel. As with firewall-to-firewall VPNs, these can be set up with controlled or open access. Controlled access is useful for clients, customers, and partners needing access to particular systems on the inside of the security perimeter for particular services at particular hours of the day. Open access is useful for employees on the road who need to get access to shared files, printers, etc. on the inside of the network security perimeter. With a VPN they can do this securely.

3.17 Technologies

Various technologies are used to implement VPNs.

3.17.1 The Need for Standardization

Many commercial firewalls today support VPN functionality. Two VPN-enabled firewalls can be used to establish a VPN. When this technology was first developed, there were no standards. Consequently, vendors established their own ways of implementing IP encryption. Some vendors went with a mechanism called swIPe (for Software IP Encryption). This was freely available in source code form with no licensing restrictions, so it was attractive because it was already written (although it needed to be ported to platforms other than SunOS) and it was the only mechanism approaching a standard. Still, one vendor's VPN using swIPe didn't work with others and it was not possible to establish an inter-vendor VPN.

In order for VPNs to become widely and routinely used, sites using differing encryption products need to be able to communicate.

3.18 IPSEC

The IETF's IP Security (IPSEC) Working Group is developing standards for IP-layer security mechanisms for both IPv4 (the version use on the Internet at the time of this writing) and IPv6 (the next generation of TCP/IP). The IPSEC architecture includes authentication (how to know if the site communicating to your site really is

who it claims to be) and encryption. These mechanisms can be used together or independently.

For establishing a VPN system with firewalls using IPSEC protocol two factors are important.

1) Authentication Header

2) Encapsulating Security Payload (ESP)

4. A Security Review of Protocols

I. Role of the Lower Layers

This chapter covers the lower layers and some basic infrastructure protocols, such as DNS.

4.1 Basic Protocols

TCP/IP is the usual shorthand for a collection of communications protocols that were originally developed under the auspices of the U.S. Defense Advanced Research Projects Agency, and was deployed on the old ARPANET in 1983.

A schematic of the data flow is shown in Figure 4.1. Each row is a different *protocol layer*. The top layer contains the applications: mail transmission, login, video servers, and so on. These applications call the lower layers to fetch and deliver their data. In the middle of the spider web is the *Internet Protocol (IP)*. IP is a packet multiplexer. Messages from higher level protocols have an *IP header* prep ended to them. They are then sent to the appropriate *device driver* for transmission. In this chapter we are going to discuss the layers from bottom to top approach. We will examine the IP layer first.

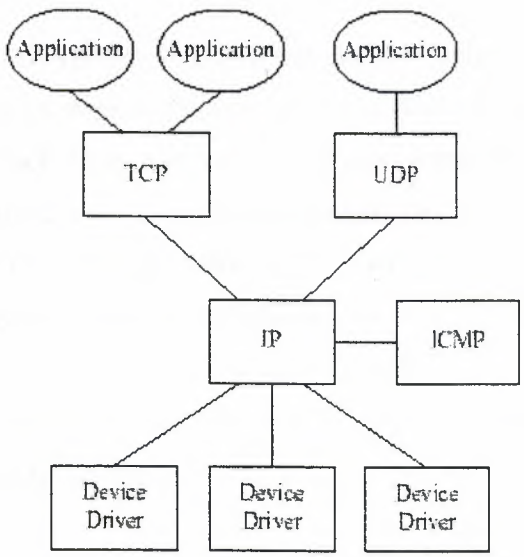


Figure 4.1: A schematic diagram of the different layers involving TCP/IP.

4.1.1 IP

IP *packets* are the bundles of data that form the foundation for the TCP/IP protocol suite. Every packet carries a source and destination address, some option bits, a header checksum, and a payload f data. A typical IP packet is a few hundred bytes long. These packets flow by the billions across the world over Ethernets, serial lines, SONET rings, packet radio connections, frame relay connections, *Asynchronous Transfer Mode (ATM)* links, and so on. But there is no notion of a *virtual circuit* or "phone call" at the IP level: every packet stands alone.

IP is an unreliable *datagram* service. No guarantees are made that packets will be delivered, delivered only once, or delivered in any particular order. Nor is there any check for packet correctness. The checksum in the IP header covers only that header.

If a packet is too large for the next hop, it is *fragmented.* That is, it is divided into two or more packets, each of which has its own IP header, but only a portion of the payload. The fragments make their own separate ways to the ultimate destination. During the trip, fragments may be further fragmented. When the pieces arrive at the target machine, they are reassembled. As a rule, no reassembly is done at intermediate hops.

IP Addresses

Addresses in IP version 4 (IPv4), the current version, are 32 bits long and are divided into two parts, a *network* portion and a *host* portion. The boundary is set administratively at each node, and in fact can vary within a site. (The older notion of fixed boundaries between the two address portions has been abandoned, and has been replaced by *Classless Inter-Domain Routing (CIDR).*

A CIDR network address is written as follows:

207.99.106.128/25

In this example, the first 25 bits are the network field (often called the *prefix*); the host field is the remaining seven bits.)

Host address portions of either all 0s or all 1s are reserved for broadcast

addresses. A packet sent with a foreign network's broadcast address is known as a *directed broadcast*; these can be very dangerous, as they're a way to disrupt many different hosts with minimal effort. Directed broadcasts have been used by attackers. For safety from such problems, most routers usually disable forwarding such packets.

From internet research it has been noted that people rarely use actual IP addresses: they prefer domain names. The name is usually translated by a special distributed database called the *Domain Name System*, discussed in coming sections.

4.1.2 ARP

IP packets are often sent over Ethernets. Ethernet devices do not understand the 32-bit IPv4 addresses: They transmit Ethernet packets with 48-bit Ethernet addresses. Therefore, an IP driver must translate an IP destination address into an Ethernet destination address. Although there are some static or algorithmic mappings between these two types of addresses, a table lookup is usually required. The *Address Resolution Protocol (ARP)* is used to determine these mappings. (ARP is used on some other link types as well; the prerequisite is some sort of link-level broadcast mechanism.)

Working of ARP

ARP works by sending out an Ethernet broadcast packet containing the desired IP address. That destination host, or another system acting on its behalf, replies with a packet containing the IP and Ethernet address pair. This is cached by the sender to reduce unnecessary ARP traffic.

The ARP mechanism is usually automatic. On special security networks, the ARP mappings may be statically hardwired, and the automatic protocol suppressed to prevent interference. If we absolutely never want two hosts to talk to each other, we can ensure that they don't have ARP translations (or have wrong ARP translations) for each other for an extra level of assurance. It can be hard to ensure that they never acquire the mappings, however.

4.1.3 TCP

TCP is a connection oriented service. The IP layer is free to drop, duplicate, or deliver packets out of order. It is up to the *Transmission Control Protocol (TCP)* layer to use this unreliable medium to provide reliable *virtual circuits* to users' processes by shuffling around, retransmitting, and reassembling packets to match the original data stream on the other end.

a) Basic working of TCP

The ordering is maintained by *sequence numbers* in every packet. Each byte sent, as well as the open and close requests, are numbered individually. A separate set of sequence numbers is used for each end of each connection to a host.

All packets, except for the very first TCP packet sent during a conversation, contain an *acknowledgment* number; it provides the sequence number of the next expected byte.

Every TCP message is marked as coming from a particular host and *port number*, and going to a destination host and port. The 4-tuple uniquely identifies a particular circuit.

(local host, local port, remote host, remote port)

If this 4-tuple shown above is not honored, every thing other than the network will behave normal.

Servers, processes that wish to provide some Internet service, *listen* on particular ports. By convention, server ports are low-numbered. This convention is not always honored, which can cause security problems. The port numbers for all of the standard services are assumed to be known to the caller. A listening port is in some sense half-open; only the local host and port number are known. (Strictly speaking, not even the local host address need be known. Computers can have more than one IP address, and connection requests can usually be addressed to any of the legal addresses for that machine.) When a connection request packet arrives, the other fields are filled in. If

appropriate, the local operating system will clone the listening connection so that further requests for the same port may be honored as well.

Clients use the offered services. They connect from a local port to the appropriate server port. The local port is almost always selected at random by the operating system, though clients are allowed to select their own.

Most versions of TCP and UDP for UNIX systems enforce the rule that only the superuser (*root*) can create a port numbered less than 1024. These are *privileged ports*. The intent is that remote systems can trust the authenticity of information written to such ports. The restriction is a convention only, and is *not* required by the protocol specification. In any event, it is meaningless on non-UNIX operating systems. The implications are clear: One can trust the sanctity of the port number only if one is certain that the originating system has such a rule, is capable of enforcing it, and is administered properly. It is not safe to rely on this convention.

b) TCP Open

TCP open, a three-step process, is shown in Figure 4.2. After the server receives the initial SYN packet, the connection is in a *half–opened* state. The server replies with its own sequence number, and awaits an acknowledgment, the third and final packet of a TCP open.

In addition, the first part of this three-step process can be used to detect active TCP services without alerting the application programs, which usually aren't informed of incoming connections until the three-packet handshake is complete.
The sequence numbers mentioned earlier have another function. Because the initial sequence number for new connections changes constantly, it is possible for TCP to detect stale packets from previous incarnations of the same circuit (i.e., from previous uses of the same 4-tuple). There is also a modest security benefit: A connection cannot be fully established until both sides have acknowledged the other's initial sequence number.

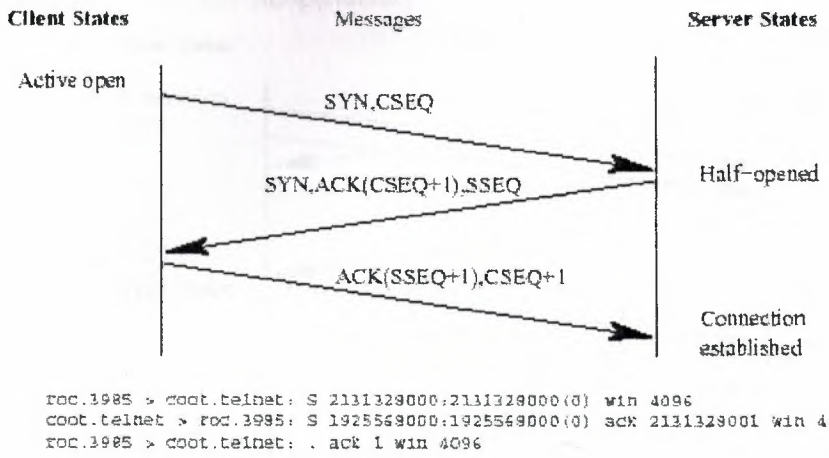


Figure 4.2: TCP Open: The client sends the server a packet with the SYN bit set, and an initial client sequence number CSEQ. The server's reply packet has both the SYN and ACK packets set, and contains both the client's (plus 1) and server's sequence number (SSEQ) for this session. The client increments its sequence number, and replies with the ACK bit set. At this point, either side may send data to the other.

In fact, TCP's three-way handshake at connection establishment time provides more protection than do some other protocols. The hacker community started using this attack in late 1995, and it is quite common now.

Many OS vendors have implemented various forms of randomization of the initial sequence number.

But the problem of hacking the sequence number is easily predictable. With everything from cell phones to doorbells running an IP stack these days, some updates to the security should be implemented to get stuff like these right.

c) TCP Sessions

Once the TCP session is open, it's full-duplex: data flows in both directions. It's a pure stream, with no record boundaries. The implementation is free to divide user data among as many or as few packets as it chooses, without regard to the way in which the data was originally written by the user process. This behavior has caused trouble for some firewalls that assumed a certain packet structure.

The TCP close sequence (see Figure 4.3) is asymmetric; each side must close its

end of the connection independently.

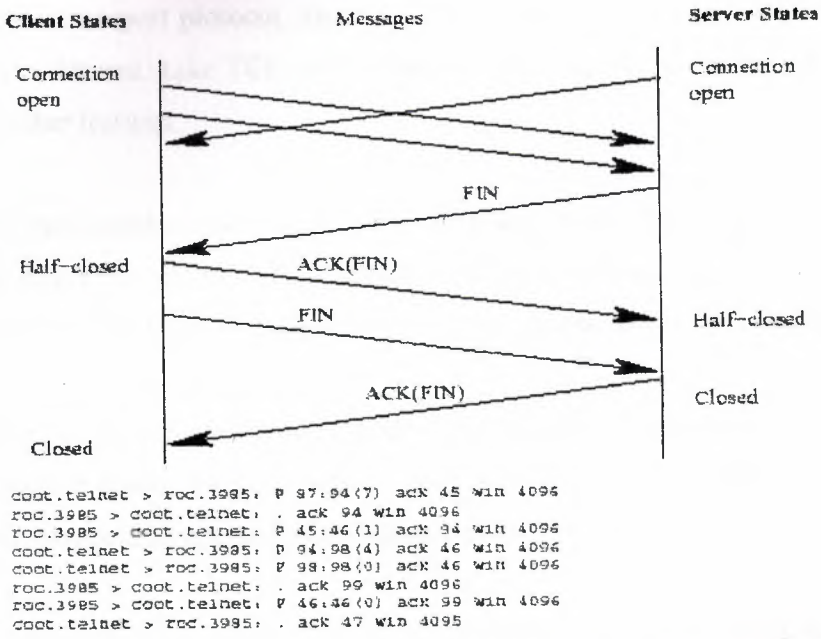


Figure 4.3: TCP I/O The TCP connection is full duplex. Each end sends a FIN packet when it is done transmitting, and the other end acknowledges. (All other packets here contain an ACK showing what has been received; those ACKs are omitted, except for the ACKs of the FINs.) A reset (RST) packet is sent when a protocol violation is detected and the connection needs to be torn down.

d) Threats on TCP

1. Attackers have gamed this half-open state. SYN attacks flood the server with the first packet only, hoping to swamp the host with half-open connections that will never be completed.

2. If the attacker can predict the target's choice of starting points, then it is possible for the attacker to trick the target into believing that it is talking to a trusted machine. In that case, protocols that depend on the IP source address for authentication can be exploited to penetrate the target system. This is known as a *sequence number attack.*

4.1.4 SCTP

A new transport protocol, *Stream Control Transmission Protocol (SCTP)*, has recently been defined. Like TCP, it provides reliable, sequenced delivery, but it has a number of other features.

The most notable new feature is the capability to multiplex several independent streams on a SCTP connection. Thus, a future FTP built on top of SCTP instead of TCP wouldn't need a PORT command to open a separate stream for the data channel.

Other improvements include a four-way handshake at connection establishment time, to frustrate denial-of-service attacks, record-marking within each stream, optional unordered message delivery, and multi-homing of each connection.

It's a promising protocol, though it isn't clear if it will catch on. Because it's new, not many firewalls support it yet. That is, not many firewalls provide the capability to filter SCTP traffic on a per-port basis, nor do they have any proxies for applications running on top of SCTP. Moreover, some of the new features, such as the capability to add new IP addresses to the connection dynamically, may pose some security issues. Keep a watchful eye on the evolution of SCTP; it was originally built for telephony signaling, and may become an important part of multimedia applications.

4.1.5 UDP

The *User Datagram Protocol (UDP)* extends to application programs the same level of service used by IP. Delivery is on a best-effort basis; there is no error correction, retransmission, or lost, duplicated, or re-ordered packet detection. Even error detection is optional with UDP. Fragmented UDP packets are reassembled, however.

To compensate for these disadvantages, there is much less overhead. In particular, there is no connection setup. This makes UDP well suited to query/response applications, where the number of messages exchanged is small compared to the connection setup and teardown costs incurred by TCP.

When UDP is used for large transmissions, it tends to behave badly on a network. The protocol itself lacks flow control features, so it can swamp hosts and

routers and cause extensive packet loss.

UDP uses the same port number and server conventions as does TCP, but in a separate address space. Similarly, servers usually (but not always) inhabit low-numbered ports. There is no notion of a circuit. All packets destined for a given port number are sent to the same process, regardless of the source address or port number.

Advantage of UDP protocols

It is much easier to spoof UDP packets than TCP packets, as there are no handshakes or sequence numbers. Extreme caution is therefore indicated when using the source address from any such packet. Applications that care *must* make their own arrangements for authentication.

4.1.6 ICMP

The *Internet Control Message Protocol (ICMP)* is the low-level mechanism used to influence the behavior of TCP and UDP connections. It can be used to inform hosts of a better route to a destination, to report trouble with a route, or to terminate a connection because of network problems. It is also a vital part of the two most important low-level monitoring tools for network administrators: *ping* and *traceroute.*

Disadvantages using ICMP

It is extremely inadvisable to block all ICMP messages at the firewall. Because of the MTU path. This is a mechanism by which hosts learn how large a packet can be sent without fragmentation i.e.it requires that certain Destination Unreachable messages be allowed through Specifically, it relies on ICMP Destination Unreachable procedural code 4.

4.2 Managing Addresses and Names

4.2.1 Routers and Routing Protocols

Routing refers to the process of discovering, selecting, and employing paths from one place to another (or to many others) in a network.

Routing protocols are mechanisms for the dynamic discovery of the proper paths through the Internet. They are fundamental to the operation of TCP/IP. Routing information establishes two paths: from the calling machine to the destination and back. The second path may or may not be the reverse of the first. When they aren't, it is called an *asymmetric route*. These are quite common on the Internet, and can cause trouble if you have more than one firewall

From a security perspective, it is the return path that is often more important. When a target machine is attacked, what path do the reverse-flowing packets take to the attacking host? If the enemy can somehow subvert the routing mechanisms, then the target can be fooled into believing that the enemy's machine is really a trusted machine. If that happens, authentication mechanisms that rely on source address verification will fail. The easiest way to defend against source routing problems is to reject packets.

Source routing is rarely used for legitimate (lawful) reasons, although those do exist. Besides, one abuse of source routing, by learning the sequence numbers of legitimate connections in order to launch a sequence-number guessing attack—works even if the packets are dropped by the application.

a) RIP and OSPF

Some routing protocols, such as RIP version 2 and *Open Shortest Path First (OSPF)*, provide for an authentication field. These are of limited utility for three reasons.

- First, some sites use simple passwords for authentication, even though OSPF has stronger variants. Anyone who has the ability to play games with routing protocols is also capable of collecting passwords wandering by on the local Ethernet cable.

- Second, if a legitimate speaker in the routing dialog has been subverted, then its messages—correctly and legitimately signed by the proper source—cannot be trusted.

  Finally, in most routing protocols, each machine speaks only to its neighbors, and they will repeat what they are told, often uncritically. Deception thus spreads.

Not all routing protocols suffer from these defects. Those that involve dialogs between pairs of hosts are harder to subvert, although sequence number attacks, may still succeed. A stronger defense is topological. Routers can and should be configured so that they know what routes can legally appear on a given wire. In general, this can be difficult to achieve, but firewall routers are ideally positioned to implement the scheme relatively simply.

This can be hard if the routing tables are too large. Still, the general case of routing protocol security is a research question.

b) IS-IS routing protocol

Some ISPs use OSI's IS-IS routing protocol internally, instead of OSPF. This has the advantage that customers can't inject false routing messages: IS-IS is not carried over IP, so there is no connectivity to customers. Note that this technique does not help protect against internal Bad Guys.

c) BGP

Border Gateway Protocol (BGP) distributes routing information over TCP connections between routers. It is normally run within or between ISPs, between an ISP and a multi-homed customer, and occasionally within a corporate intranet.
The details of BGP are quite arcane. The basics as well as important points are as follows.

BGP is used to populate the routing tables for the core routers of the Internet. The various *Autonomous Systems (AS)* trade network locations take information via announcements. These announcements arrive in a steady stream, one every couple of seconds on average. It can take 20 minutes or more for an announcement to propagate through the entire core of the Internet.

The path information distributed does not tell the whole story: There may be special arrangements for certain destinations or packet types, and other factors, such as route aggregation and forwarding delays, can muddle things.

Clearly, these announcements are vital and incorrect announcements, intentional or otherwise, can disrupt some or even most of the Internet. Corrupt announcements can be used to perform a variety of attacks.

Attacks using BGP

- Attackers play BGP games by diverting packet flows via GRE tunnels through convenient routers to eavesdrop on, hijack, or suppress Internet sessions.
- Others announce a route to their own network, attack a target, and then remove their route before forensic investigators can probe the source network.

ISPs (Internet Service Provider) have been dealing with routing problems since the beginning of time. Some BGP checks are easy: an ISP can filter announcements from its own customers. But the ISP cannot filter announcements from its peers—almost anything is legal. Fixing this infrastructure is still to be worked on.

Solving announcement procedures for ISPs

Theoretically, it is possible to hijack a BGP TCP session. MD5 BGP authentication can protect against this and is available, but it is not widely used. It should be used.

Some proposals have been made to solve the problem.

One proposal, S-BGP, provides for chains of digital signatures on the entire path received by a BGP speaker, all the way back to the origin. Several things, however, are standing in the way of deployment:

- Performance assumptions seem to be unreasonable for a busy router. A lot of public key cryptography is involved, which makes the protocol very compute-intensive. Some precomputation may help, but hardware assists may be necessary.
- A *Public Key Infrastructure (PKI)* based on authorized IP address assignments is needed, but doesn't exist.

- Some people have political concerns about the existence of a central routing registry. Some companies don't want to explicitly reveal peering arrangements and customer lists, which can be a target for salesmen from competing organizations.

For now, the best solution for end-users (and, for that matter, for ISPs) is to do regular *traceroutes* to destinations of interest, including the name servers for major zones. Although the individual hops will change frequently, the so-called AS path to nearby, major destinations is likely to remain relatively stable. The *traceroute-as* package can help with this.

4.2.2 The Domain Name System

The *Domain Name System (DNS)* is a distributed database system used to map host names to IP addresses, and vice versa. In its normal mode of operation, hosts send UDP queries to DNS servers. Servers reply with either the proper answer or information about smarter servers.

Queries can also be made via TCP, but TCP operation is usually reserved for *zone transfers*. Zone transfers are used by backup servers to obtain a full copy of their portion of the namespace. They are also used by hackers to obtain a list of targets quickly.

A number of different sorts of *resource records (RRs)* are stored by the DNS The DNS namespace is tree structured. For ease of operation, sub trees can be delegated to other servers. Two logically distinct trees are used.

- The first tree maps host names such as SMTP.ATT.COM to addresses like 192.20.225.4. Other per-host information may optionally be included, such as HINFO or MX records.
- The second tree is for *inverse queries*, and contains PTR records. In this case, it would map 4.225.20.192.IN-ADDR.ARPA to SMTP.ATT.COM.

There is no enforced relationship between the two trees, though some sites have attempted to mandate such a link for some services. The inverse tree is seldom as well-maintained and up-to-date as the commonly used forward mapping tree.

There are proposals for other trees, but they are not yet widely used.

Security Problems and solutions related with DNS

***Problem 1*:** The separation between forward naming and backward naming can lead to trouble. A hacker who controls a portion of the inverse mapping tree can make

it lie. That is, the inverse record could falsely contain the name of a machine your machine trusts. The attacker then attempts an *rlogin* to your machine, which, believing the phony record, will accept the call.

***Solution1*:** Most newer systems are now immune to this attack. After retrieving the putative host name via the DNS, they use that name to obtain their set of IP addresses. If the actual address used for the connection is not in this list, the call is bounced and a security violation logged.

***Problem 2*:**There is a more damaging variant of this attack. In this version, the attacker contaminates the target's cache of DNS responses prior to initiating the call. When the target does the cross-check, it appears to succeed, and the intruder gains access. A variation on this attack involves flooding the target's DNS server with phony responses, thereby confusing it..

Solution 2: Although the very latest implementations of the DNS software seem to be immune to this, it is imprudent to assume that there are no more holes. It is strongly recommend that exposed machines not rely on name-based authentication. Address-based authentication, though weak, is far better.

Problem 3: DNS contains a wealth of information about a site: Machine names and addresses, organizational structure, and so on.

Some have pointed out that people don't put their secrets in host names, and this is true.

Restricting zone transfers to the authorized secondary servers is a good start, but clever attackers can exhaustively search your network address space via DNS inverse queries, giving them a list of host names.

a) DNSsec

The obvious way to fix the problem of spoofed (+ve changes) DNS records is to digitally sign them. Note, though, that this doesn't eliminate the problem of the inverse tree—if the owner of a zone is corrupt, he or she can cheerfully sign a fraudulent record. This is prevented via a mechanism known as *DNSsec*. The basic idea is simple enough: All "RRsets" in a secure zone have a SIG record. Public keys (signed, of course) are in the DNS tree, too, taking the place of certificates. Moreover, a zone can be signed offline, thereby reducing the exposure of private zone-signing keys.

4.2.3 BOOTP and DHCP

The *Dynamic Host Configuration Protocol (DHCP)* is used to assign IP addresses and supply other information to booting computers (or ones that wake up on a new network). The booting client emits UDP broadcast packets and a server replies to the queries. Queries can be forwarded to other networks using a relay program. The server may supply a fixed IP address, usually based on the Ethernet address of the booting host, or it may assign an address out of a pool of available addresses.

DHCP is an extension of the older, simpler BOOTP protocol. Whereas BOOTP only delivers a single message at boot time, DHCP extensions provide for updates or changes to IP addresses and other information after booting. DHCP servers often interface with a DNS server to provide current IP/name mapping. An authentication scheme has been devised, but it is rarely used.

The protocol can supply quite a lot of information i.e. the domain name server and default route address and the default domain name as well as the client's IP address. Most implementations will use this information. It can also supply addresses for things such as the network time service, which is ignored by most implementations.

For installations of a network of any size, it is nearly essential to run DHCP. It centralizes the administration of IP addresses, simplifying administrative tasks. Dynamic IP assignments conserve scarce IP address space usage. It easily provides IP addresses for visiting laptop computers.

DHCP logs are important for forensics (public), especially when IP addresses are assigned dynamically. It is often important to know which hardware was associated with an IP address at a given time; the logged Ethernet address can be very useful. Law enforcement is often very interested in ISP DHCP logs (and RADIUS or other authentication logs) shortly after a crime is detected.

The protocol is used on local networks, which limits the security concerns somewhat. Booting clients broadcast queries to the local network. These can be forwarded elsewhere, but either the server or the relay agent needs access to the local network. Because the booting host doesn't know its own IP address yet, the response must be delivered to its layer 2 addresses, usually its Ethernet address. The server does this by either adding an entry to its own ARP table or emitting a raw layer 2 packet. In any case, this requires direct access to the local network, which a remote attacker doesn't have.

Because the DHCP queries are generally unauthenticated, the responses are subject to man in-the-middle and DOS attacks, but if an attacker already has access to the local network, then he or she can already perform ARP-spoofing attacks. That means there is little added risk in choosing to run the BOOTP/DHCP protocol. The interface with the DNS server requires a secure connection to the DNS server; this is generally done via the symmetric-key variant of SIG records.

Some other problems and attacks related to DHCP

Rogue DHCP servers can beat the official server to supplying an answer, allowing various attacks. Or, they can swamp the official server with requests from different simulated Ethernet addresses, consuming all the available IP addresses.

Finally, some DHCP clients implement lease processing dangerously. For example, *dhclient*, which runs on many UNIX systems, leaves a UDP socket open, with a privileged client program, running for the duration. This is an unnecessary door into

the client host: It need only be open for occasional protocol exchanges.

4.3 IP version 6

IP version 6 (IPv6) is much like the current version of IP, only more so. The basic philosophy—IP is an unreliable datagram protocol, with a minimal header—is the same, but there are approximately ___ details that matter. Virtually all of the supporting elements are more complex.

Renumbering doesn't occur instantaneously throughout a network. Rather, the new prefix—the low-order bits of host's addresses are not touched during renumbering—is phased in gradually.

At any time, any given interface may have several addresses, with some labeled "deprecated," i.e., their use is discouraged for new connections. Old connections, however, can continue to use them for quite some time, which means that firewalls and the like need to accept them for a while, too.

4.3.1 IPv6 Address Formats

IPv6 addresses aren't simple 128-bit numbers. Rather, they have structure and the structure has semantic implications. There are many different forms of address, and any interface can have many separate addresses of each type simultaneously.

The simplest address type is the *global unicast address*, which is similar to IPv4 addresses. In the absence of other configuration mechanisms, such as a DHCP server or static addresses, hosts can generate their own IPv6 address from the local prefix and their MAC address. Because MAC addresses tend to be constant for long periods of time, a mechanism is defined to create temporary addresses. This doesn't cause much trouble for firewalls, unless they're extending trust on the basis of source addresses (i.e., if they're misconfigured). But it does make it a lot harder to track down a miscreant's (villain) machine after the fact. If you need to do that, your routers will need to log what MAC addresses are associated with what IPv6 addresses—and routers are not, in

general, designed to log such things.

There is a special subset of unicast addresses known as *anycast addresses*. Many different nodes may share the same anycast address; the intent is that clients wishing to connect to a server at such an address will find the closest instance of it. "Close" is measured "as the packets fly," i.e., the instance that the routing system thinks is closest. Another address type is the *site-local address*. Site-local addresses are used within a "site"; border routers are supposed to ensure that packets containing such source or destination addresses do not cross the boundary. This might be a useful security property *if* you are sure that your border routers enforce this properly.

At press time, there was no consensus (agreement) on what constitutes a "site." It is reasonably likely that the definition will be restricted, especially compared to the (deliberate) early vagueness. In particular, a site is likely to have a localized view of the DNS, so that one player's internal addresses aren't visible to others. Direct routing between two independent sites is likely to be banned, too, so that routers don't have to deal with two or more different instances of the same address.

It isn't at all clear that a site boundary is an appropriate mechanism for setting security policy. If nothing else, it may be too large. Worse yet, such a mechanism offers no opportunity for finer grained access controls.

Link-local addresses are more straightforward. They can only be used on a single link, and are never forwarded by routers. Link-local addresses are primarily used to talk to the local router, or during address configuration.

Multicast is a one-to-many mechanism that can be thought of as a subset of broadcast. It is a way for a sender to transmit an IP packet to a group of hosts. IPv6 makes extensive use of multicast; things that were done with broadcast messages in IPv4, such as routing protocol exchanges, are done with multicast in IPv6. Thus, the address FF02:0:0:0:0:0:0:2 means "all IPv6 routers on this link." Multicast addresses are scoped; there are separate classes of addresses for nodes, links, sites, and organizations, as well as the entire Internet.

Border routers must be configured properly to avoid leaking confidential information, such as internal video casts.

4.3.2 Neighbor Discovery

In IPv6, ARP is replaced by the *Neighbor Discovery (ND)* protocol. ND is much more powerful, and is used to set many parameters on end systems. This, of course, means that abuse of ND is a serious matter; unfortunately, at the moment there are no well-defined mechanisms to secure it. The ND specification speaks vaguely of using *Authentication Header (AH)* (which is part of IPsec), but doesn't explain how the relevant security associations should be set up.)

Advantages using ND

There is one saving grace: ND packets *must* have their hop limit set to 255, which prevents off-link nodes from sending such packets to an unsuspecting destination. Perhaps the most important extra function provided by ND is prefix announcement. Routers on a link periodically multicast *Router Advertisement (RA)* messages; hosts receiving such messages update their prefix lists accordingly. RA messages also tell hosts about routers on their link; false RA messages are a lovely way to divert traffic. The messages are copiously larded with timers: what the lifetime of a prefix is, how long a default route is good for, the time interval between retransmissions of *Neighbor Solicitation* messages, and so on.

4.3.3 DHCPv6

Because one way of doing something isn't enough, IPv6 hosts can also acquire addresses via IPv6's version of DHCP. Notable differences from IPv4's DHCP include the capability to assign multiple addresses to an interface, strong bidirectional authentication, and an optional mechanism for revocation of addresses before their leases expire.

The latter mechanism requires clients to listen continually on their DHCP ports, which may present a security hazard; no other standards mandate that client-only machines listen on any ports. On the other hand, the ability to revoke leases can be very

useful if you've accidentally set the lease time too high, or if you want to bring down a DHCP server for emergency maintenance during lease lifetime.

4.3.4 Filtering IPv6

We do not have wide area IPv6 yet on most of the planet, so several protocols have been developed to carry IPv6 over IPv4. If you do not want IPv6, tunneled traffic should be blocked. If you want IPv6 traffic (and you're reading this book), you'll need an IPv6 firewall. If your primary firewall doesn't do this, you'll need to permit IPv6 tunnels, but only if they terminate on the outside of your IPv6 firewall. This needs to be engineered with caution.

A final scheme for tunneling IPv6 over today's Internet is based on circuit relays. With these, a router-based relay agent maps individual IPv6 TCP connections to IPv4 TCP connections; these are converted back at the receiving router.

4.4 Network Address Translators

We're running out of IP addresses. In fact, some would say that we have already run out. The result has been the proliferation (spread rapidly) of NAT boxes. Conceptually, NATs are simple: they listen on one interface (which probably uses so-called *private address space*, and rewrite the source address and port numbers on outbound packets to use the public source IP address assigned to the other interface. On reply packets, they perform the obvious inverse operation. But life in the real world isn't that easy.

Many applications simply won't work through NATs. The application data contains embedded IP addresses; if the NAT doesn't know how to also rewrite the data stream, things will break.

- Incoming calls to dynamic ports don't work very well either. Most NAT boxes will let us route traffic to specific static hosts and ports; they can't cope with arbitrary application protocols.

- To be sure, commercial NATs do know about common higher-level protocols.
- From a security perspective, a more serious issue is that NATs don't get along very well with encryption. Clearly, a NAT can't examine an encrypted application stream.
- Less obviously, some forms of IPsec are incompatible with NAT. IPsec can protect the transport layer header, which includes a checksum; this checksum includes the IP address that the NAT box needs to rewrite.

Some people think that NAT boxes are a form of firewall. In some sense, they are, but they're low-end ones. At best, they're a form of packet filter. They lack the application level filtering that most dedicated firewalls have; more importantly, they may lack the necessarily paranoid designers.

4.5 Wireless Security

A world of danger can lurk at the link layer. We've already discussed ARP-spoofing. But wireless networks add a new dimension. It's not that they extend the attackers' powers; rather, they expand the reach and number of potential attackers.
The most common form of wireless networking is IEEE 802.11b, known to marketeers as WiFi. 802.11 is available in most research labs, at universities, at conferences, in coffeehouses, at airports, and even in peoples' homes.

To prevent random, casual access to these networks, the protocol designers added a symmetric key encryption algorithm called *Wired Equivalent Privacy (WEP)*.

Problems in Wireless Security

The idea is that every machine on the wireless network is configured with a secret key, and thus nobody without the key can eavesdrop on traffic or use the network.

- Although the standard supports encryption, early versions supported either no encryption at all or a weak 40-bit algorithm.
- As a result, you can cruise through cities or high-tech residential neighborhoods and obtain free Internet access, complete with DHCP support. This is

devastating. In most places, the 802.11 key does not change after deployment, if it is used at all. Considering the huge deployed base of 802.11 cards and access points, it will be a monumental task to fix this problem.

- A number of mistakes were made in the design. Most seriously, it uses a stream cipher, which is poorly matched to the task. All users on a network share a common, static key. The alleged *initialization vector (IV)* used is 24 bits long, guaranteeing frequent collisions for busy access points. The integrity check used by WEP is a CRC-32 checksum, which is linear. In all cases, it would have been trivial (some value) to avoid trouble.
- They should have used a block cipher; failing that, they should have used much longer IVs and a cryptographic checksum.
- The attack (often referred to as the FMS attack) requires one byte of known plaintext and several million packets, and results in a passive adversary directly recovering the key. Because 802.11 packets are encapsulated in 802.2 headers with a constant first byte, all that is needed is the collection of the packets.

All this shows that uneconomical WEP technology provides a sense of security, without useful security

4.5.1 Fixing WEP

Given the need to improve WEP before all of the hardware is redesigned and redeployed in new wireless cards a new enhanced protocol named *Temporal Key Integrity Protocol (TKIP)* is introduced by IEEE organization.

TKIP uses the existing API on the card—namely, RC4 with publicly visible IVs—and plays around with the keys so that packets are dynamically keyed. In TKIP, keys are changed often (on the order of hours), and IVs are forced to change with no opportunity to wrap around. Also, the checksum on packets is a cryptographic MAC, rather than the CRC used by WEP.

It is a reasonable workaround, given the legacy issues involved. The next generation of hardware is designed to support the *Advanced Encryption Standard (AES)*, and is being scrutinized by the security community.

It is not clear that the link layer is the right one for security. Perhaps link-layer security makes some sense in a home, where you control both the access point

and the wireless machines. It is recommended, that in commercial places, end-to-end security at the network layer or in the applications must be deployed.

II. Role of the Upper Layers

If we examine Figure 4.1 we will notice that the hourglass gets wider at the top part. In this part of the chapter we are going to discuss about the protocols and applications that run on the upper layers.

1. Messaging
2. Internet Telephony
3. RPC based protocols
4. File transfer protocols
5. Remote login
6. Simple Network Management Protocol (SNMP)
7. Network time Protocol
8. Peer to Peer networking

4.6 Messaging

In this section, we deal with mail transport protocols.

4.6.1 SMTP

One of the most popular Internet services is electronic mail. Though several services can move mail on the net, by far the most common is *Simple Mail Transfer Protocol (SMTP)*. Traditional SMTP transports 7-bit ASCII text characters using a simple protocol, shown below. Here's a log entry from a sample SMTP session (the arrows show the direction of data flow):

```
<--- 220 fg.net SMTP
---> HELO sales.mymegacorp.com
<--- 250 fg.net
---> MAIL FROM:<Anthony.Stazzone@sales.mymegacorp.com>
<--- 250 OK
---> RCPT TO:<ferd.berfle@fg.net>
<--- 250 OK
---> DATA
<--- 354 Start mail input; end with <CRLF>.<CRLF>
---> From: A.Stazzone@sales.mymegacorp.com
---> To: ferd.berfle@fg.net
---> Date: Thu, 27 Jan 94 21:00:05 EST
```

```
--->
---> Meet you for lunch after I buy some power tools.
--->
---> Anthony
---> .
--->
<--- 250 OK
.... sales.mymegacorp.com!A.Stazzone sent 273 bytes to fg.net!ferd.berfle
---> QUIT
<--- 221 sales.mymegacorp.com Terminating
```

Here, the remote site, SALES.MYMEGACORP.COM, is sending mail to the local machine, FG.NET. It is a simple protocol. Postmasters and hackers learn these commands and occasionally type them by hand.

Requirements under which SMTP works and advantages

An organization needs at least one mail guru to concentrate the mailer expertise at a gateway, even if the inside networks are fully connected to the Internet. This way, administrators on the inside need only get their mail to the gateway mailer.

The gateway can ensure that outgoing mail headers conform to standards. The organization becomes a better network citizen when there is a single, knowledgeable contact for reporting mailer problems.

The mail gateway is also an excellent place for corporate mail aliases for every person in a company until it is kept safe. If not kept safe the mail aliases can provide the hacker with some useful information. Commands such as

```
VRFY <postmaster>
VRFY <root>
```

often translate the mail alias to the actual login name. This can provide clues about who the system administrator is and which accounts might be most profitable if successfully attacked.

A useful technique is to have the alias on the well-known machine point to an inside machine, not reachable from the outside, so that the expansion can be done there without risk.

For this reason privileged programs should be as small and modular as possible so that SMTP daemon does not need to run as *root*.

Regardless of which mailer we run using SMTP, we should configure it so that it

will only accept mail that is either from one of our networks, or to one of our users. If we need to support road warriors, we can use SMTP Authentication. This is best used in conjunction with encryption of the SMTP session. The purpose of SMTP authentication is to avoid having an open relay; open relays attract spammers, and can result in site being added to a "reject all mail from these clowns" list. This use of SMTP is sometimes known as "mail submission," to distinguish it from more general mail transport.

4.6.2 MIME

Multipurpose Internet Mail Extension (MIME) is a protocol used on the internet for sending electronic mail. The main aims of this protocol are to check the state of the mail and its kind so that it is decoded to its original shape.
This protocol is not used now because of some disadvantages.

Disadvantages

It cannot specify that the content of the mail can pose dangers or not. Apart from possible bugs in the receiving machine's mailer, automated execution of *Multipurpose Internet Mail Extensions (MIME)* encoded messages is potentially quite dangerous. The structured information encoded in them can indicate actions to be taken.

One MIME type permits a single e-mail message to be broken up into multiple pieces. Judicious fragmentation can be used to evade the scrutiny of gateway-based virus checkers.

4.6.3 POP version 3

POP3, the Post Office Protocol is used by simple clients to obtain their mail. Their mail is delivered to a mailbox on a spooling host, perhaps provided by an ISP. When a client runs its mailer, the mailer downloads the waiting messages into the client. The mail is typically removed from the server. While online, the mailer may poll the server at regular intervals to obtain new mail. The client sends mail using SMTP, perhaps directly or through a different mail server. (A number of sites use the POP3 authentication to enable mail-relaying via SMTP, thus blocking spammers. The server caches the IP address of the machine from which the successful POP3 session came; for

a limited time thereafter, that machine is allowed to do SMTP relaying.). The protocol is quite simple, and has been around for a while. The server can implement it quite easily

Disadvantages

For using a POP3 on UNIX operating system it is required that a user must have a data base for authentication purposes. If user passes through each step it means he/she has all the access to the server which is not a good advice because users are bad security risks. To overcome this problem POP3 servers should be used that contains their database especially designed for the users.

The benefits of POP3 include the simplicity of the protocol (if only network telephony were this easy!) and the easy implementation on the server. It is limited, however users generally must read their mail from one host, as the mail is generally delivered to the client.

4.6.4 IMAP Version 4

IMAP version 4 offers remote access to mailboxes on a server. It enables the client and server to synchronize state, and supports multiple folders. As in POP3, mail is still sent using SMTP.

A typical UNIX IMAP4 server requires the same access as a POP3 server, plus more to support the extra features. The IMAP protocol does support a suite of authentication methods, some of which are fairly secure. The challenge/response authentication is a step in the right direction, but it is not as good as it could be. A shared secret is involved, which again must be stored on the server. It would be better if the challenge/response secret were first hashed with a domain string to remove some password equivalence. (Multiple authentication options always raise the possibility of *version-rollback attacks*, forcing a server to use weaker authentication or cryptography.)

Our biggest reservation about IMAP is the complexity of the protocol, which of course requires a complex server. *If* the server is implemented properly, with a small, simple authentication module as a front end to an unprivileged protocol engine, this may be no worse than user logins to the machine, but you need to verify the design of your

server.

4.7 Internet Telephony

One of the application areas gathering the most attention is Internet telephony. The global telephone network is increasingly connected to the Internet; this connectivity is providing signaling channels for phone switches, data channels for actual voice calls, and new customer functions, especially ones that involve both the Internet and the phone network.

Two main protocols are used for voice calls, the *Session Initiation Protocol (SIP)* and H.323. Both can do far more than set up simple phone calls. At a minimum, they can set up conferences (Microsoft's NetMeeting can use both protocols); SIP is also the basis for some Internet/telephone network interactions, and for some instant messaging protocols.

4.7.1 H.323

H.323 is the ITU's Internet telephony protocol. The design of this protocol is based on Q931, the ISDN signaling protocol.

The actual call traffic is carried over separate UDP ports. In a firewalled world, this means that the firewall has to parse the ASN.1 messages to figure out what port numbers should be allowed in. This isn't an easy task, and we worry about the complexity of any firewall that is trying to perform it.

H.323 calls are not point-to-point. At least one intermediate server, a telephone company, is needed; depending on the configuration and the options used, many more may be employed.

4.7.2 SIP

SIP, though rather complex, is significantly simpler than H.323. Its messages are ASCII; they resemble HTTP, and even use MIME and S/MIME for transporting data. SIP phones can speak peer-to-peer; however, they can also employ the same sorts of proxies as H.323. Generally, in fact, this will be done. Such proxies can simplify the process of passing SIP through a firewall, though the actual data transport is usually direct between the two (or more) endpoints. SIP also has provisions for very strong security perhaps too strong, in some cases, as it can interfere with attempts by the firewall to rewrite the messages to make it easier to pass the voice traffic via an application-level gateway.

Some data can be carried in the SIP messages themselves, but as a rule, the actual voice traffic uses a separate transport. This can be UDP, probably carrying *Real-Time Transport Protocol (RTP)*, TCP, or SCTP.

4.8 RPC-Based Protocols

4.8.1 RPC and Rpcbind

Sun's *Remote Procedure Call (RPC)*, under-lies a few important services. Unfortunately, many of these services represent potential security problems. RPC is used today on many different platforms, including most of Microsoft's operating systems.

A thorough understanding of RPC is vital. The basic concept is simple enough. The person creating a network service uses a special language to specify the names of the external entry points and their parameters. A pre-compiler converts this specification into *stub* or glue routines for the client and server modules. With the help of this glue and a bit of boilerplate, the client can make seemingly ordinary subroutine calls to a remote server. Most of the difficulties of network programming are masked by the RPC layer.

RPC can live on top of either TCP or UDP. Most of the essential characteristics

of the transport mechanisms show through. RPC messages begin with their own header. It includes the *program number*, the *procedure number* denoting the entry point within the procedure, and some version numbers.

4.8.2 NIS

One dangerous RPC application is the *Network Information Service (NIS)*, formerly known as YP (yellow pages). NIS is used to distribute a variety of important databases from a central server to its clients. These include the password file, the host address table, and the public and private key databases used for Secure RPC. Access can be by search key, or the entire file can be transferred.

Disadvantages with NIS

Many of the risks are obvious. An intruder who obtains your password file has a precious thing indeed. The key database can be almost as good; private keys for individual users are generally encrypted with their login passwords.
NIS clients need to know about backup servers, in case the master is down. In some versions, clients can be told remotely to use a different, and possibly fraudulent, NIS server. This server could supply bogus /etc/passwd file entries, incorrect host addresses, and so on.

Some versions of NIS can be configured to disallow the most dangerous activities. Obviously, you should do this if possible. Better still, do not run NIS on exposed machines; the risks are high, and for gateway machines the benefits very low.

4.8.3 NFS

The *Network File System (NFS)*,is the mostly widely used file system these days. It is a vital component of most workstations, and it is not likely to go away any time soon. For robustness, NFS is based on RPC, UDP, and stateless servers.

NFS generally relies on a set of numeric user and group identifiers that must be consistent across the set of machines being served. While this is convenient for local

use, it is not a solution that scales. Some implementations provide for a map function. NFS access by *root* is generally prohibited, a restriction that often leads to more frustration than protection.

Problems and dangers associated with NFS

Normally, NFS servers live on port 2049. The choice of port number is problematic, as it is in the "unprivileged" range, and hence is in the range assignable to ordinary processes. Packet filters that permit UDP conversations *must* be configured to block inbound access to 2049; the service is too dangerous.
Furthermore, some versions of NFS live on random ports, with *rpcbind* providing addressing information.

NFS poses risks to client machines as well. Someone with privileged access to the server machine—or someone who can forge reply packets—can create setuid programs or device files, and then invoke or open them from the client

A more subtle problem with browsing archives via NFS is that it's too easy for the server machine to plant booby-trapped versions of certain programs likely to be used, such as *ls*. If the user's $PATH has the current directory first, the phony version will be used, rather than the client's own *ls* command. This is always poor practice: If the current directory appears in the path, it should always be the last entry. The NFS best defense here would be for the client to delete the "execute" bit on all imported files.

4.8.4 Andrew

The *Andrew File System (AFS)* is another network file system that can, to some extent, interoperate with NFS. Its major purpose is to provide a single scalable, global, location-independent file system to an organization, or even to the Internet as a whole.
AFS enables files to live on any server within the network, with caching occurring transparently, and as needed. AFS uses Kerberos authentication and a Kerberos-based user identifier mapping scheme. It thus provides a considerably higher degree of safety than do simpler versions of NFS.

4.9 File Transfer Protocols

4.9.1 TFTP

The *Trivial File Transfer Protocol (TFTP)* is a simple UDP-based file transfer mechanism. It has no authentication in the protocol. It is often used to boot routers, diskless workstations, and X11 terminals.

A properly configured TFTP daemon restricts file transfers to one or two directories, typically /usr/local/boot and the X11 font library. TFTP is used to load either executable images or configuration files.

4.9.2 FTP

The *File Transfer Protocol (FTP)* supports the transmission and character set translation of text and binary files. Figure 4.2 illustrates the typical session, the user's *ftp* command opens a control channel to the target machine. Various commands and responses are sent over this channel. The server's responses include a three-digit return code at the beginning of each line. A second data channel is opened for a file transfer or the listing from a directory command.

The FTP protocol specification suggests that a single channel be created and kept open for all data transfers during the session.

The data channel can be opened from the server to the client, or the client to the server. This choice can have important security implications, discussed below. In the older server-to-client connection, the client listens on a random port number and informs the server of this via the PORT command. In turn, the server makes the data connection by calling the given port.

```
$ ftp -d research.att.com
220 inet FTP server (Version 4.271 Fri Apr 9 10:11:04 EDT 1993) ready.
---> USER anonymous
331 Guest login ok, send ident as password.
---> PASS guest
230 Guest login ok, access restrictions apply.
---> SYST
215 UNIX Type: L8 Version: BSD-43
Remote system type is UNIX.
---> TYPE I
200 Type set to I.
```

```
Using binary mode to transfer files.
ftp> ls
---> PORT 192,20,225,3,5,163
200 PORT command successful.
---> TYPE A
200 Type set to A.
---> NLST
150 Opening ASCII mode data connection for /bin/ls.
bin
dist
etc
ls-lR.Z
netlib
pub
226 Transfer complete.
---> TYPE I
200 Type set to I.
ftp> bye
---> QUIT
221 Goodbye.
$
```

Figure 4.3: A sample FTP session using the PORT command. The lines starting with ---> show the commands that are actually sent over the wire; responses are preceded by a three-digit code.

4.9.3 SMB Protocol

The *Server Message Block (SMB)* protocols have evolved slowly, and now appear to be drifting toward the *Common Internet File System (CIFS)*, a new open file-sharing protocol promoted by Microsoft.

SMB is transported on various network services; these days, TCP/IP-based mechanisms are the most interesting

4.10 Remote Login

4.10.1 Telnet

Telnet provides simple terminal access to a machine. The protocol includes provisions for handling various terminal settings such as raw mode, character echo, and so on. As a rule, *telnet* daemons call *login* to authenticate and initialize the session. The caller supplies an account name and usually a password to *login*.

Problems within Telnet

Most *telnet* sessions come from un trusted machines. Neither the calling program, the calling operating system, nor the intervening networks can be trusted. *The password and the terminal session are available to prying eyes.* The local *telnet* program may be com-promised to record username and password combinations or to log the entire session. This is a common hacking trick.

Traditional passwords are not reliable when any part of the communications link is tapped. *So it is strongly recommended that we use a one-time password scheme* The authenticators can secure a login nicely, but they do not protect the rest of a session. If the *telnet* command has been tampered with, it could insert unwanted commands into your session or retain the connection after you think you have logged off.

4.10.2 The "*r* " Commands

To the first order, every computer in the world is connected to every other computer.
BOB MORRIS

The "r" commands rely on the BSD authentication mechanism. One can *rlogin* to a remote machine without entering a password if the authentication's criteria are met. These criteria are as follows:

- The call must originate from a privileged TCP port. On other systems (like PCs) there are no such restrictions, nor do they make any sense. A corollary of this is that *rlogin* and *rsh* calls should be permitted only from machines on which this restriction is enforced.
- The calling user and machine must be listed in the destination machine's list of trusted partners (typically /etc/hosts.equiv) or in a user's .rhosts file.
- The caller's name must correspond to its IP address.

This protocol uses rlogind daemon which has the capability to stop users from

accessing and taking control of the file system.

4.10.3 *Ssh*

A variety of encryption and authentication methods are available. *Ssh* can supplement or replace traditional host and password authentication with RSA- or DSA-keyed and challenge response authentication.

4.11 Simple Network Management Protocol—SNMP

The *Simple Network Management Protocol (SNMP)* is used to control routers bridges, and other network elements. It is used to read and write an astonishing variety of information about the device: operating system, version, routing tables, default TTL, traffic statistics, interface names, ARP tables, and sensitive data.

The data object is defined in a *management information base (MIB)*. MIB entries are in turn encoded in *ASN.*1, a data specification language of some complexity. To obtain a piece of information from a router, one uses a standard MIB, or perhaps downloads a special MIB entry from the manufacturer but unfortunately these MIBS are not always well tested for security issues.

The SNMP protocol itself comes in two major versions, numbers one and three. The most widely deployed is version 1and is the least secure.

SNMP version.3 has much better security, cryptographic authentication, optional encryption, and most important, the ability to grant different access rights to portions of the MIB to different users. The crypto authentication can be expensive, and routers typically have weak CPUs, so it may be best to restrict access to these services as well.

4.12 The Network Time Protocol

The *Network Time Protocol (NTP)* is a valuable adjunct to gateway machines. As its name implies, it is used to synchronize a machine's clock with the outside world. It is not a voting protocol; rather, NTP supports the notion of absolute correct time. Each machine talks to one or more neighbors; the machines organize themselves into a

directed graph, depending on their distance from an authoritative time source.

Comparisons among multiple sources of time information enable NTP servers to discard erroneous inputs; this provides a high degree of protection against deliberate subversion as well. The *Global Positioning System (GPS)* receivers can supply very cheap and accurate time in-formation to a master host running *ntp*. Sites concerned with security should have a source of accurate time.

The time-keeping ability of NTP is so good that one can easily use it to determine the relative timings of probes to different machines, even when they occur nearly simultaneously. Such information can be very useful in understanding the attacker's technology.

4.13 Peer-to-Peer Networking

Peer-to-peer networking presents some unique challenges. The basic behavior is exactly what its name implies: all nodes are equal, rather than some being clients and some servers.

The precise problem is: many different nodes act as servers. This means that trying to secure just a few machines doesn't work anymore—*every* participating machine is offering up resources, and must be protected. That problem is compounded if you're trying to offer the service through a firewall.

The biggest issue, of course, is bugs in the p2p software or configuration. Apart from the usual plague of buffer overflows, there is the significant risk of offering up the wrong files. Here, you have to find and fix the problem on many different machines. In fact, you may not even know which machines are running that software.
Beyond that, there are human interface issues, similar to those that plague some mailers. Is that really a .doc file you're clicking on, or is it a .exe file with .doc embedded in the name?

If you or your users are file-sharing, you have more problems, even without considering the copyright issue (e.g. file theft in our c and c++ lab sessions).

Table 1. Sensor network layers and denial-of-service defenses

Network layer	Attacks	Defenses
Physical	Jamming	Spread-spectrum, priority messages, lower duty cycle, region mapping, mode change
	Tampering	Tamper-proofing, hiding
Link	Collision	Error-correcting code
	Exhaustion	Rate limitation
	Unfairness	Small frames
Network and routing	Neglect and greed	Redundancy, probing
	Homing	Encryption
	Misdirection	Egress filtering, authorization, monitoring
	Black holes	Authorization, monitoring, redundancy
Transport	Flooding	Client puzzles
	Desynchronization	Authentication

Table 4.1showing the problems existing within these layers.

5. SHORT COMING OF FIREWALLS AND ITS SOLUTIONS

5.1 Overview of Firewalls

A *firewall* is a collection of components, interposed between two networks, that filters traffic between them according to some security policy. Conventional firewalls rely on network topology restrictions to perform this filtering. Furthermore, one key assumption under this model is that everyone on the protected network(s) is trusted (since internal traffic is not seen by the firewall, it cannot be filtered); if that is not the case, then additional, internal firewalls have to be deployed in the internal network.
While this model worked well for small to medium size networks, several trends in networking threaten to make it obsolete:

Due to the increasing line speeds and the more computation intensive protocols that a firewall must support (especially IPsec), firewalls tend to become congestion points. This gap between processing and networking speeds is likely to increase, at least for the foreseeable future; while computers(and hence firewalls) are getting faster, the combination of more complex protocols and the tremendous increase in the amount of data that must be passed through the firewall has been and likely will continue to outpace Moore's Law.

There exist protocols, and new protocols are designed, that are difficult to process at the firewall, because the latter lacks certain knowledge that is readily available at the endpoints.
FTP and RealAudio are two such protocols. Although there exists application-level proxies that handle such protocols, such solutions are viewed as architecturally "unclean" and in some cases too invasive.

The assumption that all insiders are trusted too has not been valid for a long time. Specific individuals or remote networks may be allowed access to all or parts of the protected infrastructure (extranets, telecommuting, *etc.*). Consequently, the

traditional notion of a security perimeter can no longer hold unmodified; for example, it is desirable that telecommuters' systems comply with the corporate security policy.

Worse yet, it has become trivial for anyone to establish a new, unauthorized entry point to the network without the administrator's knowledge and consent. Various forms of tunnels, wireless, and dial-up access methods allow individuals to establish backdoor access that bypasses all the security mechanisms provided by traditional firewalls. While firewalls are in general not intended to guard against misbehavior by insiders, there is a tension between internal needs for more connectivity and the difficulty of satisfying such needs with a centralized firewall.

- Large (and even not-so-large) networks today tend to have a large number of entry points (for performance, failover, and other reasons). Furthermore, many sites employ internal firewalls to provide some form of compartmentalization. This makes administration particularly difficult, both from a practical point of view and with regard to policy consistency, since no unified and comprehensive management mechanism exists.
- End-to-end encryption can also be a threat to firewalls, as it prevents them from looking at the packet fields necessary to do filtering. Allowing end-to-end encryption through a firewall implies considerable trust to the users on behalf of the administrators.
- Finally, there is an increasing need for finer-grained (and even application-specific) access control which standard firewalls cannot readily accommodate without greatly increasing their complexity and processing requirements.

Despite their shortcomings, firewalls are still useful in providing some measure of security. The key reason that firewalls are still useful is that they provide an obvious, mostly hassle-free, mechanism for *enforcing* network security policy. For legacy applications and networks, they are the only mechanism for security. While newer protocols typically have some provisions for security, older protocols (and their implementations) are more difficult, often impossible, to secure. Furthermore, firewalls provide a convenient first-level barrier that allows quick responses to newly-discovered bugs.

5.2 Concepts of Distributed firewalls and feasibility

To address the shortcomings of firewalls while retaining their advantages, We propose the concept of a *distributed firewall.* In distributed firewalls, *security policy is defined centrally but enforce at each individual network endpoint* (hosts, routers, *etc.*). The system propagates the central policy to all endpoints. Policy distribution may take various forms. For example, it may be pushed directly to the end systems that have to enforce it, or it may be provided to the users in the form of credentials that they use when trying to communicate with the hosts, or it may be a combination of both. The extent of mutual trust between endpoints is specified by
the policy.

To implement a distributed firewall, three components are necessary:

- A language for expressing policies and resolving requests. In their simplest form, policies in a distributed firewall are functionally equivalent to packet filtering rules. However, it is desirable to use an extensible system (so other types of applications and security checks can be specified and enforced in the future). The language and resolution mechanism should also support credentials, for delegation of rights and authentication purposes.
- A mechanism for safely distributing security policies. This may be the IP sec key management protocol when possible, or some other protocol. The integrity of the policies transferred must be guaranteed, either through the communication protocol or as part of the policy object description (*e.g.,* they may be digitally signed).
- A mechanism that applies the security policy to incoming packets or connections, providing the enforcement part.

This is by no means a universal trait, and even today there are protocols designed with no security review. Our prototype implementation uses the KeyNote trust-management system, which provides a single, extensible language for expressing policies and credentials. Credentials in KeyNote are signed, thus simple file-transfer protocols may be used for policy distribution.

We also make use of the IPsec stack in the OpenBSD system to authenticate

users, protect traffic, and distribute credentials. The distribution of credentials and user authentication occurs are part of the Internet Key Exchange (IKE) negotiation.

Alternatively, policies may be distributed from a central location when a policy update is performed, or they may be fetched as-needed (from a web server, X.500 directory, or through some other protocol).

Since Key Note allows delegation, decentralized administration becomes feasible (establishing a hierarchy or web of administration, for the different departments or even individual systems). Users are also able to delegate authority to access machines or services they themselves have access to. Although this may initially seem counter-intuitive (after all, firewalls embody the concept of centralized control), in our experience users can almost always bypass a firewall's filtering mechanisms, usually by the most insecure and destructive way possible (*e.g.*, giving away their password, setting up a proxy or login server on some other port, *etc.*). Thus, it is better to allow for some flexibility in the system, as long as the users follow the overall policy. Also note that it is possible to "turn off" delegation.

Thus, the overall security policy relevant to a particular user and a particular end host is the composition of the security policy "pushed" to the end host, any credentials given to the user, and any credentials stored in a central location and retrieved on-demand.

Finally, we implement the mechanism that enforces the security policy in a TCP-connection granularity. In our implementation, the mechanism is split in two parts, one residing in the kernel and the other in a user-level process.

5.3. The Distributed Firewalls

A distributed firewall, of the type described above, uses a central policy, but pushes enforcement towards the edges. That is, the policy defines what connectivity, inbound and outbound, is permitted; this policy is distributed to all endpoints which enforce it.

In the full-blown version, endpoints are characterized by their IPsec identity,

typically in the form of a certificate. Rather than relying on the topological notions of "inside" and "outside", as is done by a traditional firewall, a distributed firewall assigns *certain rights* to whichever machines own the private keys corresponding to certain public *keys*. Thus, the right to connect to the `http` port on a company's internal Web server might be granted to those machines having a certificate name of the form `*.goodfolks.org`, rather than those machines that happen to be connected to an internal wire. A laptop directly connected to the Internet has the same level of protection as does a desktop in the organization's facility. Conversely, a laptop connected to the corporate net by a visitor would not have the proper credentials, and hence would be denied access, even though it is topologically "inside".

5.3.1 Implementation of distributed firewalls

To implement a distributed firewall, we need a security policy language that can describe which connections are acceptable, an authentication mechanism, and a policy distribution scheme. As a policy specification language, we use the KeyNote trust-management system, further described in coming section.

As an authentication mechanism, we decided to use end to end IPsec for traffic protection and user/host authentication. Each incoming packet can be associated with a certificate; the access granted to that packet is determined by the rights granted to that certificate. If the certificate name is different, or if there is no IPsec protection, the packet will be dropped as unauthorized.

We note that the necessary filtering is prescribed by the IPSEC architecture. Specifically, the inbound *Security Policy Database* (SPD) is used to reject illegal input packets, while the outbound SPD can be used to control outgoing connections. An informal survey showed that most commercial IPsec implementations either support port number-granularity security associations or will in the near future.

Application-level protection can be achieved by distributing application-specific policy files. Thus, web browsers can be told, by the central site, to reject, for example, all ActiveX controls or Java applets. Note that this is a hard problem for conventional firewalls; doing it on the end hosts is *more* secure, if the policy distribution problem can

be solved.

The hardest problem in firewalls is handling protocols such as FTP without touching the application. That is done most easily with per-process keying for IPsec.. For example, a policy rule for FTP would indicate that outbound connections to port 21 must be protected with IPsec, and that all other TCP connections protected by that security association are legal. Since only that process can use that SA, and it would only have the FTP data channel open, an adequate level of protection can be achieved.

Furthermore, we would then depend on the good behavior of the very applications we are trying to protect. Finally, it would be impossible to secure legacy applications with inadequate provisioning for security.

When it comes to policy distribution, we have a number of choices:

- We can distribute the KeyNote (or other) credentials to the various end users. The users can then deliver their credentials to the end hosts through the IKE protocol. The users do not have to be online for the policy update; rather, they can periodically retrieve the credentials from a repository (such as a web server). Since the credentials are signed and can be transmitted over an insecure connection, users could retrieve their new credentials even when the old ones have expired. This approach also prevents, or at least mitigates, the effects of some possible denial of service attacks.
- The credentials can be pushed directly to the end hosts, where they would be immediately available to the policy verifier. Since every host would need a large number, if not all, of the credentials for every user, the storage and transmission bandwidth requirements are higher than in the previous case.
- The credentials can be placed in a repository where they can be fetched as needed by the hosts. This requires constant availability of the repository, and may impose some delays in the resolution of request (such as a TCP connection establishment).

While the first case is probably the most attractive from an engineering point of view, not all IKE implementations support distribution of KeyNote credentials.

Furthermore, some IPsec implementations do not support connection-grained security. Finally, since IPsec is not (yet) in wide use, it is desirable to allow for a policy-based filtering that does not depend on IPsec. Thus, it is necessary to provide a *policy resolution* mechanism that takes into consideration the connection parameters, the local policies, and any available credentials (retrieved through IPsec or other means), and determines whether the connection should be allowed. We describe our implementation of such a mechanism for the OpenBSD system latter in this chapter..

5.4 KeyNote

Trust Management is a relatively new approach to solving the authorization and security policy problem. Making use of public key cryptography for authentication, trust management dispenses with unique names as an indirect means for performing access control. Instead, *it uses a direct binding between a public key and a set of authorizations*, as represented by a safe programming language. This results in an inherently decentralized authorization system with sufficient expressibility to guarantee flexibility in the face of novel authorization scenarios.

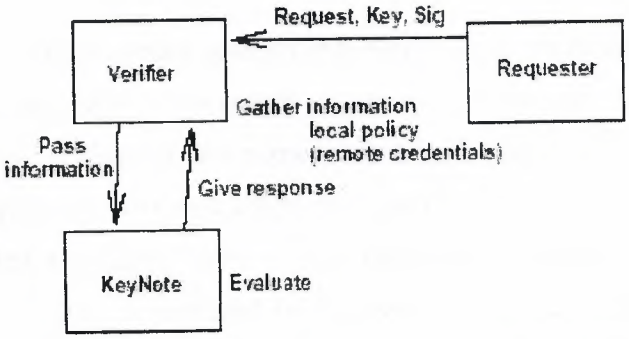


Figure 5.1: Application Interactions with KeyNote. The Requester is typically a user that authenticates through some application-dependent protocol, and optionally provides credentials. The Verifier needs to determine whether the Requester is allowed to perform the requested action. It is responsible for providing to KeyNote all the necessary information, the local policy, and any credentials. It is also responsible for acting upon KeyNote's response.

One instance of a trust-management system is KeyNote. KeyNote provides a simple notation for specifying both local security policies and credentials that can be sent over an untrusted network. Policies and credentials contain predicates (part of the

statement which says something about subject) that describe the trusted actions permitted by the holders of specific public keys (otherwise known as principals).

Signed credentials, which serve the role of "certificates," have the same syntax as policy assertions, but are also signed by the entity delegating the trust. Applications communicate with a "KeyNote evaluator" that interprets (explain) KeyNote assertions and returns results to applications, as shown in Figure 5.1. However, different hosts and environments may provide a variety of interfaces to the KeyNote evaluator.

A KeyNote evaluator accepts as input a set of local policy and credential assertions, and a set of attributes, called an "action environment," that describes a proposed trusted action associated with a set of public keys (the requesting principals). The KeyNote evaluator determines whether proposed actions are consistent with local policy by applying the assertion predicates to the action environment.

The KeyNote evaluator can return values other than simply true and false, depending on the application and the action environment definition.

An important concept in KeyNote (and, more generally, in trust management) is *"monotonicity"*. This simply means that given a set of credentials associated with a request, if there is any subset that would cause the request to be approved then the complete set will also cause the request to be approved. This greatly simplifies both request resolution (even in the presence of conflicts) and credential management. Monotonicity is enforced by the KeyNote language (it is not possible to write non-monotonic policies).

It is worth noting here that although KeyNote uses cryptographic keys as principal identifiers, other types of identifiers may also be used. For example, usernames may be used to identify principals inside a host. In this environment, delegation must be controlled by the operating system (or some implicitly trusted application), similar to the mechanisms used for transferring credentials in UNIX or in capability-based systems. Also, in the absence of cryptographic authentication, the identifier of the principal requesting an action must be securely established. In the

example of a single host, the operating system can provide this information.

```
KeyNote-Version: 2
Authorizer: "POLICY"
Licensees: "rsa-hex:1023abcd"
Comment: Allow Licensee to connect to local port 23 (telnet) from
         internal addresses only, or to port 22 (ssh) from anywhere.
         Since this is a policy, no signature field is required.
Conditions: (local_port == "23" && protocol == "tcp" &&
             remote_address > "158.130.006.000" &&
             remote_address < "158.130.007.255") -> "true";
            local_port == "22" && protocol == "tcp" -> "true";

KeyNote-Version: 2
Authorizer: "rsa-hex:1023abcd"
Licensees: "dsa-hex:986512a1" || "x509-base64:19abcd02=="
Comment: Authorizer delegates SSH connection access to either
         of the Licensees, if coming from a specific address.
Conditions: (remote_address == "139.091.001.001" &&
             local_port == "22") -> "true";
Signature: "rsa-md5-hex:f00f5673"
```

Figure 5.2: Example KeyNote Policy and Credential. The local policy allows a particular user (as identified by their public key) connect access to the telnet port by internal addresses, or to the SSH port from any address. That user, then delegates to two other users (keys) the right to connect to SSH from one specific address. Note that the first key can effectively delegate at most the same rights it possesses. KeyNote does not allow *rights amplification*; any delegation acts as *refinement*.

In our prototype, end hosts (as identified by their IP address) are also considered *principals* when IPsec is not used to secure communications. This allows local policies or credentials issued by administrative keys to specify policies similar to current packet filtering rules. Naturally, such policies or credentials implicitly trust the validity of an IP address as an identifier. In that respect, they are equivalent to standard packet filtering. The only known solution to this is the use of cryptographic protocols to secure communications.

Since KeyNote allows multiple policy constraints (to compel), potentially for different applications, to be contained in the same assertion, it is trivial to support application-specific credentials. Credentials that specify, *e.g.*, Java applet permissions, could be delivered under any of the distribution schemes and made available to the end application through some OS-specific mechanism (*e.g.*, `getsockopt(2)` calls).

In the context of the distributed firewall, KeyNote allows us to use the same, simple language for both policy and credentials. The latter, being signed, may be distributed over an insecure communication channel. In KeyNote, credentials may be

considered as an extension, or refinement, of local policy; *the union of all policy and credential assertions is the overall network security policy.* Alternately, credentials may be viewed as parts of a hypothetical access matrix. End hosts may specify their own security policies, or they may depend exclusively on credentials from the administrator, or do anything in between these two ends of the spectrum. Perhaps of more interest, it is possible to "merge" policies from different administrative entities and process them unambiguously, or to layer them in increasing levels of refinement. This merging can be expressed in the KeyNote language, in the form of intersection (conjunction) and union (disjunction) of the component sub-policies.

Although KeyNote uses a human-readable format and it is indeed possible to write credentials and policies that way, our ultimate goal is to use it as an interoperability-layer language that "ties together" the various applications that need access control services. An administrator would use a higher-level language like GUI to specify correspondingly higher-level policy and then have this compiled to a set of KeyNote credentials. This higher-level language would provide grouping mechanisms and network-specific abstractions (for networks, hosts, services, *etc.*) that are not present in KeyNote.

Using KeyNote as the middle language offers a number of benefits:

- It can handle a variety of different applications (since it is application-independent but customizable), allowing for more comprehensive and mixed-level policies (*e.g.*, covering email, active code content, IPsec, *etc.*).
- Provides built-in delegation, thus allowing for decentralized administration.
- Allows for incremental or localized policy updates (as only the relevant credentials need to be modified, produced, or revoked).

Figure 5.2 shows two sample KeyNote assertions, a policy and a (signed) credential. Figure 5.3 shows an example of a key delegating to an IP address.

5.5 Implementation

For our development platform we decided to use the OpenBSD operating system. OpenBSD provides an attractive platform for developing security applications

because of the well-integrated security features and libraries (an IPsec stack, SSL, KeyNote, *etc.*). However, similar implementations are possible under other operating systems.

Our system is comprised of three components: a set of kernel extensions, which implement the enforcement mechanisms, a user level daemon process, which implements the distributed firewall policies, and a device driver, which is used for two-way communication between the kernel and the policy daemon.

Our prototype implementation totals approximately 1150 lines of C code; each component is roughly the same size. Figure 5.4 shows a graphical representation of the system, with all its components.

```
KeyNote-Version: 2
Authorizer: "rsa-hex:1023abcd"
Licensees: "IP:158.130.6.141"
Conditions:  (@remote_port < 1024 &&
              @local_port == 22) -> "true";
Signature: "rsa-sha1-hex:bee11984"
```

Figure 5.3: An example credential where an (administrative) key delegates to an IP address. This would allow the specified address to connect to the local SSH port, if the connection is coming from a privileged port. Since the remote host has no way of supplying the credential to the distributed firewall through a security protocol like IPsec, the distributed firewall must search for such credentials or must be provided with them when policy is generated/updated.

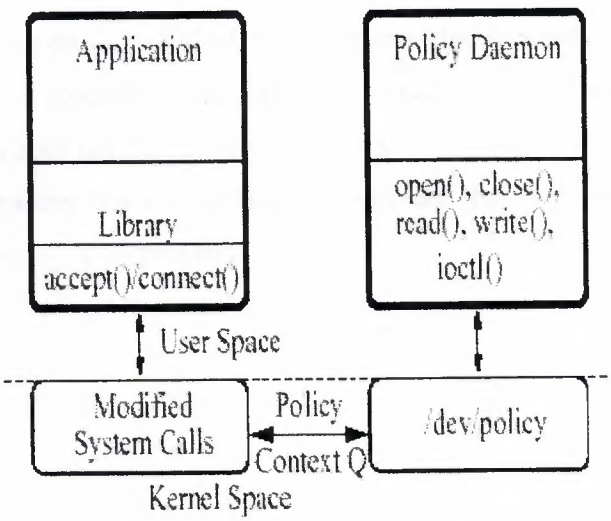


Figure 5.4: The Figure shows a graphical representation of the system, with all its

components. The core of the enforcement mechanism lives in kernel space and is comprised of the two modified system calls that interest us, `connect(2)` and `accept(2)`. The policy specification and processing unit lives in user space inside the policy daemon process. The two units communicate via a loadable pseudo device driver interface. Messages travel from the system call layer to the user level daemon and back using the *policy context queue*.

In the following three subsections we describe the various parts of the architecture, their functionality, and how they interact with each other.

5.5.1 Kernel Extensions

For our working prototype we focused our efforts on the control of the TCP connections. Similar principles can be applied to other protocols; for unreliable protocols, some form of reply caching is desirable to improve performance.

In the UNIX operating system users create outgoing and allow incoming TCP connections using the `connect(2)` and `accept(2)` system calls respectively. Since any user has access to these system calls, some "filtering" mechanism is needed. This filtering should be based on a policy that is set by the administrator.

Filters can be implemented either in user space or inside the kernel. Each has its advantages and disadvantages.

A user level approach, as depicted in Figure 3.5, requires each application of interest to be linked with a library that provides the required security mechanisms, *e.g.*,, a modified `libc`. This has the advantage of operating system-independence, and thus does not require any changes to the kernel code. However, such a scheme does not guarantee that the applications *will* use the modified library, potentially leading to a major security problem.

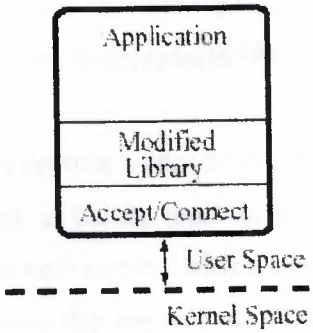


Figure 5.5: Wrappers for filtering the `connect(2)` and `accept(2)` system calls are added to a system library. While this approach offers considerable flexibility, it suffers from its inability to guarantee the enforcement of security policies, as applications might not link with the appropriate library.

A kernel level approach, as shown in the left side of Figure 5.4, requires modifications to the operating system kernel. This restricts us to open source operating systems like BSD and Linux. The main advantage of this approach is that the additional security mechanisms can be enforced transparently on the applications.

As we mentioned previously, the two system calls we need to filter are `connect(2)` and `accept(2)`. When a `connect(2)` is issued by a user application and the call traps into the kernel, we create what we call a *policy context* (see Figure 5.6), associated with that connection. The policy context is a container for all the information related to that specific connection. We associate a sequence number to each such context and then we start filling it with all the information the *policy daemon* will need to decide whether to permit it or not. In the case of the `connect(2)`, this includes the ID of the user that initiated the connection, the destination address and port, *etc.* Any credentials acquired through IPsec may also be added to the context at this stage. There is no limit as to the kind or amount of information we can associate with a context. We can, for example, include the time of day or the number of other open connections of that user, if we want them to be considered by our decision–making strategy.

Once all the information is in place, we *commit* that context. The commit

operation adds the context to the list of contexts the policy daemon needs to handle. After this, the application is blocked waiting for the policy daemon reply.

Accepting a connection works in a similar fashion. When `accept(2)` enters the kernel, it blocks until an incoming connection request arrives. Upon receipt, we allocate a new context which we fill in similarly to the `connect(2)` case. The only difference is that we now also include the source address and port. The context is then enqueued, and the process blocks waiting for a reply from the policy daemon.

In the next section we discuss how messages are passed between the kernel and the policy daemon.

```
typedef struct  policy_mbuf     policy_mbuf;
struct  policy_mbuf     {
        policy_mbuf     *next;
        int              length;
        char             data[POLICY_DATA_SIZE];
};

typedef struct  policy_context  policy_context;
struct  policy_context  {
        policy_mbuf     *p_mbuf;
        u_int32_t        sequence;
        char            *reply;
        policy_context  *policy_context_next;
};

policy_context  *policy_create_context(void);
void            policy_destroy_context(policy_context *);
void            policy_commit_context(policy_context *);
void            policy_add_int(policy_context *, char *, int);
void            policy_add_string(policy_context *, char *, char *);
void            policy_add_ipv4addr(policy_context *, char *, in_addr_t *);
```

Figure 5.6: The `connect(2)` and `accept(2)` system calls create *contexts* which contain information relevant to that connection. These are appended to a queue from which the policy daemon will receive and process them. The policy daemon will then return to the kernel a decision on whether to accept or deny the connection.

5.5.2 Policy Device

To maximize the flexibility of our system and allow for easy experimentation, we decided to make the policy daemon a user level process. To support this architecture, we implemented a *pseudo device driver*, `/dev/policy` that serves as a communication path between the user-space policy daemon, and the modified system calls in the kernel. Our device driver supports the usual operations (`open(2)`,

`close(2)`, `read(2)`, `write(2)`, and `ioctl(2)`). Furthermore, we have implemented the device driver as a loadable module. This increases the functionality of our system even more, since we can add functionality dynamically, without needing to recompile the whole kernel.

If no policy daemon has opened `/dev/policy`, no connection filtering is done. Opening the device activates the distributed firewall and initializes data structures. All subsequent `connect(2)` and `accept(2)` calls will go through the procedure described in the previous section. Closing the device will free any allocated resources and disable the distributed firewall.

When reading from the device the policy daemon blocks until there are requests to be served. The policy daemon handles the policy resolution messages from the kernel, and writes back a reply. The `write(2)` is responsible for returning the policy daemons decision to the blocked connection call, and then waking it up. It should be noted that both the device and the associated messaging protocol are not tied to any particular type of application, and may in fact be used without any modifications by other kernel components that require similar security policy handling.

Finally, we have included an `ioctl(2)` call for "house–keeping". This allows the kernel and the policy daemon to re–synchronize in case of any errors in creating or parsing the request messages, by discarding the current policy context and dropping the associated connection.

5.5.3 Policy Daemon and its working

The third and last component of our system is the policy daemon. *It is a user level process responsible for making decisions, based on policies that are specified by some administrator and credentials retrieved remotely or provided by the kernel, on whether to allow or deny connections.*

Policies, as shown in Figure 5.2, are initially read in from a file. It is possible to remove old policies and add new ones dynamically. In the current implementation, such policy changes only affect new connections.

Communication between the policy daemon and the kernel is possible, as we mentioned earlier, using the `policy` device. The daemon receives each request from the kernel by reading the device. This can be explained in figure 5.7.

The request contains all the information relevant to that connection as described in Section 5.1 of this chapter. Processing of the request is done by the daemon using the KeyNote library, and a decision to accept or deny it is reached. Finally the daemon writes the reply back to the kernel and waits for the next request. While the information received in a particular message is application-dependent (in our case, relevant to the distributed firewall), the daemon itself has no awareness of the specific application. Thus, it can be used to provide policy resolution services for many different applications, literally without any modifications.

When using a remote repository (save storage) server, the daemon can fetch a credential based on the ID of the user associated with a connection, or with the local or remote IP address (such credentials may look like the one in Figure 5.3). A very simple approach to that is fetching the credentials via HTTP from a remote web server. The credentials are stored by user ID and IP address, and provided to anyone requesting them. If credential "privacy" is a requirement, one could secure this connection using IPsec or SSL.

To avoid potential deadlocks, the policy daemon is not subject to the connection filtering mechanism.

```
u_int32_t seq;     /* Sequence Number */
u_int32_t uid;              /* User Id */
u_int32_t N;      /* Number of Fields */
u_int32_t l[N]; /* Lengths of Fields */
char      *field[N];        /* Fields */
```

Figure 5.7: The request to the policy daemon is comprised of the following fields: a sequence number uniquely identifying the request, the ID of the user the connection request belongs to, the number of information fields that will be included in the request, the lengths of those fields, and finally the fields themselves.

```
KeyNote-Version: 2
Authorizer: "POLICY"
Licensees: ADMINISTRATIVE_KEY
```

Figure 5.8: End-host local security policy. In our particular scenario, the policy simply states that some administrative key will specify our policy, in the form of one or more credentials. The lack of a Conditions field means that there are no restrictions imposed on the policies specified by the administrative key.

5.6 Practical Use of Distributed Firewalls

To better explain the interaction of the various components in the distributed firewall, we discuss the course of events during two incoming TCP connection requests, one of which is IPsec–protected. The local host where the connection is coming is part of a distributed firewall, and has a local policy as shown in Figure 5.8.

In the case of a connection coming in over IPsec, the remote user or host will have established an IPsec Security Association with the local host using IKE. As part of the IKE exchange, a KeyNote credential as shown in Figure 5.9 is provided to the local host. Once the TCP connection is received, the kernel will construct the appropriate context as discussed in Section 4.1. This context will contain the local and remote IP addresses and ports for the connection, the fact that the connection is protected by IPsec, the time of day, *etc.*

This information along with the *credential acquired via IPsec* will be passed to the policy daemon. The *policy daemon will perform a KeyNote* evaluation using the local policy and the credential, and will determine whether the connection is authorized or not. In our case, Fig.5.9, the positive response will be sent back to the kernel, which will then permit the TCP connection to proceed. Note that more credentials may be provided during the IKE negotiation (for example, a chain of credentials delegating authority).

If KeyNote does not authorize the connection, the policy daemon will try to acquire relevant credentials by contacting a remote server where these are stored. In our current implementation, we use a web server as the credential repository. In a large-scale network, a distributed/replicated database could be used instead. The policy

daemon uses the public key of the remote user (when it is known, *i.e.,* when IPsec is in use) and the IP address of the remote host as the keys to lookup credentials with; more specifically, credentials where the user's public key or the remote host's address appears in the Licensees field are retrieved and cached locally (Figure 5.3 lists an example credential that refers to an IP address). These are then used in conjunction with the information provided by the kernel to re-examine the request. If it is again denied, the connection is ultimately denied.

```
inside_net = x509{name="*.example.com", ver>19990315};
 mail_gw = x509{name="mailgw.example.com"};
 time_server = IPv4{10.1.2.3};

 allow smtp(*, mail_gw);
 allow smtp(mail_gw, inside_net);
 allow ntp(time_server,inside_net);
 allow *(inside_net, *);
```

Figure 5.9 A policy configuration file with version information

5.7 Advantages and Threats using Distributed Firewalls

Distributed firewalls have both strengths and weaknesses when compared to conventional firewalls. These are

5.7.1 Service Exposure and Port Scanning

Both types of firewalls are excellent at rejecting connection requests for inappropriate services. Conventional firewalls drop the requests at the border; distributed firewalls do so at the host. A more interesting question is what is noticed by the host attempting to connect. Today, such packets are typically discarded, with no notification. A distributed firewall may choose to discard the packet, under the assumption that its legal peers know to use IPsec; alternatively, it may instead send back a response requesting that the connection be authenticated, which in turn gives notice of the existence of the host.

```
KeyNote-Version: 2
Authorizer: ADMINISTRATIVE_KEY
Licensees: USER_KEY
Conditions:
     (app_domain == "IPsec policy" &&
      encryption_algorithm == "3DES" &&
      local_address == "158.130.006.141")
           -> "true";
     (app_domain ==
           "Distributed Firewall" &&
      @local_port == 23 &&
      encrypted == "yes" &&
      authenticated == "yes") -> "true";
Signature: ...
```

Figure 5.10: A credential from the administrator to some user, authorizing that user to establish an IPsec Security Association (SA) with the local host and to connect to port 23 (telnet) over that SA. To do this, we use the fact that multiple expressions can be included in a single KeyNote credential. Since IPsec also enforces some form of access control on packets, we could simplify the overall architecture by skipping the security check for TCP connections coming over an IPsec tunnel. In that case, we could simply merge the two clauses (the IPsec policy clause could specify that the specific user may talk to TCP port 23 only over that SA).

Firewalls built on pure packet filters cannot reject some "stealth scans" very well. One technique, for example, uses fragmented packets that can pass through unexamined because the port numbers aren't present in the first fragment. A distributed firewall will reassemble the packet and then reject it.

On balance, against this sort of threat the two firewall types are at least comparable.

5.7.2 Application-level Proxies

Some services require an application-level proxy. Conventional firewalls often have an edge here; the filtering code is complex and not generally available on host platforms. As noted, a hybrid technique can often be used to overcome this disadvantage.

In some cases, of course, application-level controls can avoid the problem entirely. If the security administrator can configure all Web browsers to reject ActiveX, there is no need to filter incoming HTML via a proxy.

In other cases, a suitably sophisticated IPsec implementation will suffice. For

example, there may be no need to use a proxy that scans outbound FTP control messages for PORT commands, if the kernel will permit an application that has opened an outbound connection to receive inbound connections. This is more or less what such a proxy would do.

5.7.3 Denial of Service Attacks

It is impossible to lump all denial of service attacks into one basket. However, some statements can be made about particular known attacks.

The "smurf" attack primarily consumes the bandwidth on the access line from an ISP to the target site. Neither form of firewall offers an effective defense. If one is willing to change the topology, both can be moderately effective. Conventional firewalls can be located at the ISP's POP, thus blocking the attack before it reaches the low-bandwidth access line. Distributed firewalls permit hosts to be connected via many different access lines, thus finessing the problem. For now, a distributed intrusion detection systems would be useful.

It may be possible to chew up CPU time by bombarding the IKE process with bogus security association negotiation requests. While this can affect conventional firewalls, inside machines would still be able to communicate. Distributed firewalls rely much more on IKE, and hence are more susceptible. It is an open question if a different key exchange protocol will be needed to resist such attacks.

Conversely, any attack that consumes resources on conventional firewalls, such as many email attachments that must be scanned for viruses, can bog down such firewalls and affect all users. For that matter, too much legitimate traffic can overload a firewall. As noted, distributed firewalls do not suffer from this effect.

5.7.4 Intrusion Detection

Many firewalls detect attempted intrusions. If that functionality is to be provided by a distributed firewall, each individual host has to notice probes and forward them to some central location for processing and correlation.

The former problem is not hard; many hosts already log such attempts. One can make a good case that such detection should be done in any event. Collection is more problematic, especially at times of poor connectivity to the central site. There is also the risk of co-ordinated attacks in effect causing a denial of service attack against the central machine.

Our tentative conclusion is that intrusion detection is somewhat harder than with conventional firewalls. While more information can be gathered, using the same techniques on hosts protected by conventional firewalls would gather the same sort of data.

5.7.5 Insider Attacks

At first glance, the biggest weakness of distributed firewalls is their greater susceptibility to lack of cooperation by users. What happens if someone changes the policy files on their own?

Although there are technical measures that can be taken, as discussed earlier, these are limited in their ability to cope with serious misbehavior. Even conventional firewalls are easily subverted by an uncooperative insider. SSH can be used to tunnel TCP ports, external Web proxies such as `www.anonymizer.com` can bypass destination restrictions, end-to-end encryption can hide traffic, etc. In other words, an insider who wishes to violate firewall policy can do so, with either type of firewall.

On the other hand, distributed firewalls can reduce the threat of actual attacks by insiders, simply by making it easier to set up smaller groups of users. Thus, one can restrict access to a file server to only those users who need it, rather than letting anyone inside the company pound on it.

BIBLOGRAPHY.

1. **Firewall & Network Security** Bill Cheswick ches@research.att.com & Steven M. Bellovin smb@research.att.com
2. **Improved Firewalling** Peter M.Gleitz pmgleit@netscape.net& Steven M. Bellovin smb@research.att.comAT&T Labs Research.
3. **Implementing a Distributed Firewall** Sotiris Ioannidis Univ. Of Pennsylvania sotiris@dsl.cis.upenn.edu Angelos D. Keromytis Univ. of Pennsylvania adk@adk.gr Steve M. Bellovin AT&T Labs — Research smb@research.att.com Jonathan M. Smith Univ. of Pennsylvania jms@cis.upenn.edu.
4. **How Trustworth Is the Trusted Computing** Steven J. Vaughan-Nichols
5. **Network Security Management with Firewalls** Stephen P. Cooper SPCooper@LLNL.GOV.
6. **Firewalls and Internet Security, the Second Hundred (Internet) Years** by Frederick Avolio, Avolio Consulting, Inc. http://www.cisco.com/ipj.
7. **Firewalls and Virtual Private Networks** by Frederick Avolio Trusted Information Systems, Inc.
8. **Cryptography And Network Security** Principles and Practice: Third Edition by William Stalling.
9. **Net use Statistics** http://www.princeton.edu/~eszter/netuse.html
10. **Global Internet Statistics**: Sources & References http://global-reach.biz/globstats/refs.php3
11. George M. Henry, Robert A. Zerwekh, "**Academic Uses of the Internet**: Computer-Aided Language Learning", Tenth Annual Midwest Computer Conference, Chicago 1996
12. **Why is Internet Security so Important?** http://way2goal.com/internet/is.html
13. Sam Silverthorn, "**Internet Security**", http://www.cc.ndsu.nodak.edu/ndsu/draper/470/projects/ss.html
14. **What is hacking?** http://www.boloji.com/computing/security/020.htm

15. Andrew Odlyzko "**Internet Growth**: Myth and Reality, Use and Abuse" Imp magazine, November 2000
http://www.cisp.org/imp/november_2000/odlyzko/11_00odlyzko.htm

16. Charlie Kaufman, Radia Perlman, Mike Speciner: **Network Security – Private Communication in a Public World**, Prentice-Hall, 1995.

17. B.Schneier: **Applied Cryptography**, 2nd Ed (1995) John Wiley and Sons;

18. David Kahn: **The Codebreakers**, Revised Ed (1997) Simon & Schuster.

19. William Stallings: **Network Security Essentials**, (2000) Prentice-Hall.

CONCLUSION.

Network security is a complicated subject, historically only controlled by well trained and experienced experts. However, as more and more people became ``wired'', an increasing number of people need to understand the basics of security in a networked world.

As the humans invented the network for benifits so shall the humans are responsible for its destruction. These problems can be made accidently or may be created willingly to attack other neighbouring nodes or remote nodes. Today it is almost impossible to consider that a machine in the network is safe, from somewhere it can face any kind of attacking threats.

This reason, one must atleast needs to save his/her "important data" from outside attack by working on a secure network.To make such kind of network; few vital asspects must be implemented such as **'security policy', 'maximum attack resisting strategies', 'use of firewalls logic'** both invisible as well as visible, **'data encryption' and 'protecting passwords'** strategies.

Most of the problems usually comes from outside of the secure network boundries. These attacks occur due to misunderstanding of type of incomming data. For example , a virus program is downloaded in the system considering it as if it was an important locally usable file or program or in other way ,if the password is given to an *adversory accidently*, he/she can enter the netwok with all legal ways and can control inside everthing of a secure network.

To overcome such kinds of problems, different ***encyption techniques*** should be used so that the "***adversory or the hacker***" if possibly try his/her best to read the important data or the password he/she '***should fail atleast'*** to enter the network, this technique is called ***confidentiality*** and ***digital signature***. Secondly if someone on the other hand tries to send data (bombs) to attack the secure network, firewalls should prevent such unauthonticated (unauthorized) data from entering the VPN.

It is very important for a network administrator to enter ,solutions against such kinds of threats and working options ,to be programmed earlier while deploying a security policy, that is kind of *firewalls* and to *design a trusted computing* for a secure network.

Human life depends on give and achive information infastructure so does the computers. Every machine should communicate with each other for sending and recieving information through a "channel" with particular language that both machines understand called ***protocols***. Protocols play an important role in the field of security while working on OSI network layers. Different protocols have different requirements of layer model they use for packet sending and recieving.

There are many kinds of protocols for different reasons such as "mail sending and recieving" protocols etc. For a secure firewall, free sending and recieving information, a particular protocol should be used.

Now a days the most popular protocol used is the TCP/IP but there are more than a million ways to hack this popurly used protocol .The securest possible protocol to be used from research point of view for the network and for the firewalls are ***"IPv6"*** and the ***"IPsec"*** protocols.

Conventional firewalls rely on the notions of restricted topology and controlled entry points to function. More precisely, they rely on the assumption that everyone on one side of the entry point--the firewall--is to be trusted, and that anyone on the other side is, at least potentially, an enemy. The vastly expanded Internet connectivity in recent years has called that assumption into question. Where a distributed firewall preserves central control of access policy, while reducing or eliminating any dependency on topology.

Based on my research I would like to propose a ***"distributed firewalls"***, using ***IPsec*** *protocol,* a policy language, and an ***OpenBSD*** system because it is the most securest possible way to save information and data from unauthorized or illigal intrusions.

APPENDICES

IPsec

1. IPsec is a protocol suite, recently standardized by the IETF that provides network-layer security services such as packet confidentiality, authentication, data integrity, replay protection, and automated key management.
2. This is an artifact of firewall deployment: internal traffic that is not seen by the firewall cannot be filtered; as a result, internal users can mount attacks on other users and networks without the firewall being able to intervene. If firewalls were placed everywhere, this would not be necessary.

KeyNote

An instance of Trust Management system. KeyNote provides a simple notation for specifying both local security policies and credentials that can be sent over an un trusted network

OpenBSD,

OpenBSD provides an attractive platform for developing security applications because of the well-integrated security features and libraries

Observations about RSA algorithm:

- First method that could be used for digital signatures
- Used world-wide today
- Was patented in the USA by RSA Data Security Inc.; patent expired 20.9.2000
- Public key must be authentic – how to organize key distribution?
- f (and therefor also d) can only be computed with p and q. RSA relies on the difficulty of factorization of large numbers (NP complete problem, takes exponential time)
- There is no proof that there is no fast and general method for factorization; record today ~ 150 digits (155)
- Choice of exponent e is not critical, often e=3
- Key generation and encryption/decryption are computationally intensive

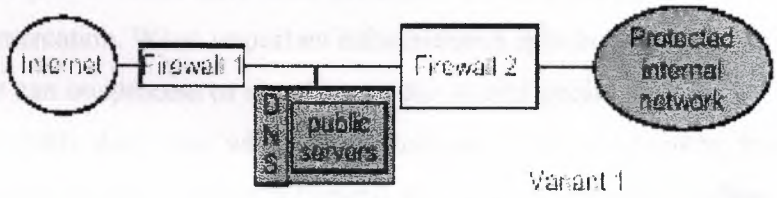
Access control by visible firewalls:

- Users use Internet exclusively from the firewall.
- All users shold have user account on the firewall.
- The firewalls terminates DNS, e=mail, http
- User authentication must be secure(cryptographically.

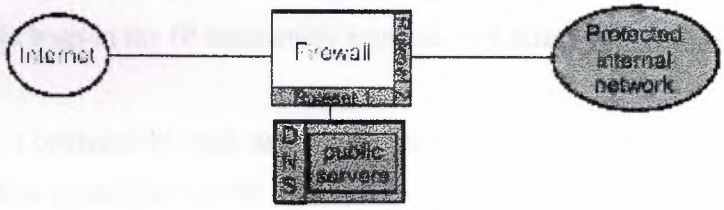
It is user unfriendly as it shold be.

Access control by visible firewalls:

- Termination of all storing and forwarding (DNS and email) with servers on the firewalls.
- Selective forarding of the connections.
- Authontication of external and internal peers.
- Logging and intrusion detection.
- Network Address Translation.
- Proxy Functions.



An example of visible firewall



An example of invisible firewalls, it uses only one unit.

List of Bombs

1. In fact, there is no guarantee that a packet was actually sent from the given source address. Any host can transmit a packet with any source address. Although many operating systems control this field and ensure that it leaves with a correct value, and although a few ISPs ensure that impossible packets do not leave a site, *you cannot rely on the validity of the source address, except under certain carefully controlled circumstances.*

   Therefore, authentication cannot rely on the source address field, although several protocols do just that. In general, attackers can send packets with faked return addresses: this is called *IP spoofing*. Authentication and security in general, must use mechanisms in higher layers of the protocol.

   A packet traveling a long distance will travel through many *hops*. Each hop terminates in a host or router, which forwards the packet to the next hop based on routing information. Along the way, a router is allowed to drop packets without notice if there is too much traffic.

2. Some packet filters have been breached by being fed packets with pathological fragmentation. When important information is split between two packets, the filter can misprocess or simply pass the second packet. Worse yet, the rules for reassembly don't say what should happen if two overlapping fragments have different content. Perhaps a firewall will pass one harmless variant, only to find that the other dangerous variant is accepted by the destination host. Most firewalls reassemble fragmented packets to examine their contents. This processing can also be a trouble spot. Fragment sequences have also been chosen to tickle bugs in the IP reassembly routines on a host, causing crashes.

3. There is considerable risk here if untrusted nodes have write access to the local net. Such a machine could emit phony ARP queries or replies and divert all traffic to itself; it could then either impersonate some machines or simply modify the data streams *en passant*.

   This is called ARP *spoofing* and a number of *Hacker Off-the-Shelf (HOTS)* packages implement this attack.

4. Many ICMP messages received on a given host are specific to a particular connection or are triggered by a packet sent by that machine. The hacker community is fond of abusing ICMP to tear down connections.

5. Worse things can be done with Redirect messages. As explained in the following section, anyone who can tamper with your knowledge of the proper route to a destination can probably penetrate your machine. The Redirect messages should be obeyed only by hosts, not routers, and only when a message comes from a router on a directly attached network.

   However, not all routers (or, in some cases, their administrators) are that careful; it is sometimes possible to abuse ICMP to create new paths to a destination. If that happens, you are in serious trouble indeed.

   There are a number of ways to attack the standard routing facilities. The easiest is to employ the IP *loose source route* option. With it, the person initiating a TCP connection can specify an explicit path to the destination, overriding the usual route selection process.

6. Another path attackers can take is to play games with the routing protocols themselves. For example, it is relatively easy to inject bogus *Routing Information Protocol (RIP)* packets into a network. Hosts and other routers will generally believe them. If the attacking machine is closer to the target than is the real source machine, it is easy to divert traffic.

   Many implementations of RIP will even accept host-specific routes, which are much harder to detect.

7. The separation between forward naming and backward naming can lead to trouble. A hacker who controls a portion of the inverse mapping tree can make it lie. That is, the inverse record could falsely contain the name of a machine your machine trusts. The attacker then attempts an *rlogin* to your machine, which, believing the phony record, will accept the call.

8. There is a more damaging variant of this attack. In this version, the attacker contaminates the target's cache of DNS responses prior to initiating the call. When the target does the cross-check, it appears to succeed, and the intruder gains access. A variation on this attack involves flooding the target's DNS server with phony responses, thereby confusing it. We've seen hacker's toolkits with simple programs for poisoning DNS caches.

9. The most important thing to know about IPv6 is that easy renumbering is one of the design goals. This means that any address-based access controls need to know about renumbering, and need to be updated at the right times. Of course, they need to know about *authentic* renumbering events; fraudulent ones should, of course, be treated with the proper mix of disdain and contempt (disobedience).

10. Notice that the caller specified a return address in the MAIL FROM command. At this level, there is no reliable way for the local machine to verify the return address. *You do not know for sure who sent you mail based on SMTP.* You must use some higher level mechanism if you need trust or privacy.

11. From a security standpoint, the basic SMTP by itself is fairly innocuous. It could, however, be the source of a *denial-of-service (DOS)* attack, an attack that's aimed at preventing legitimate use of the machine. Suppose we arrange to have 50 machines each mail you 1000 1 MB mail messages. Can your systems handle it? Can they handle the load? Is the spool directory large enough?

12. Password system failures are the biggest single problem.

13. Sequence number attacks can be used to subvert address-based authentication.

14. It is easy to spoof UDP packets.

15. ICMP packets can tear down all connections between a pair of hosts .

16. ICMP Redirect messages can subvert routing tables.

17. IP source routing can subvert address-based authentication.

18. It is easy to generate bogus RIP messages .

19. The inverse DNS tree can be used for name-spoofing.

20. The DNS cache can be contaminated to foil crosschecks.

21. Return addresses in mail aren't reliable.
22. *Sendmail* is a security risk.
23. Don't blindly execute MIME messages.
24. It is easy to wiretap *telnet* sessions.
25. You can subvert NTP in order to attack authentication protocols.
26. *Finger* discloses too much information about users.
27. Don't trust RPC's machine name field.
28. The *portmapper* can call RPC services for its caller.
29. NIS can often be persuaded to give out password files.
30. It is sometimes possible to direct machines to phony NIS servers.
31. It is hard to revoke NFS access.
32. If misconfigured, TFTP will hand out /etc/passwd.
33. Don't make *ftp*'s home directory writable by *ftp*.
34. Don't put a real password file in the anonymous *ftp* area.
35. FSP is often abused to give out files to those who should not have them.
36. Be careful about interpreting WWW format information .
37. WWW servers should be careful about file pointers.
38. Attackers can use *ftp* to create *gopher* control information.
39. Poorly written query scripts pose a danger to WWW servers.
40. The MBone can be used to route through some firewalls.
41. An attacker anywhere on the Internet can probe for X11 servers.
42. Don't believe port numbers supplied by outside machines.
43. It is all but impossible to permit most UDP traffic through a packet filter safely.
44. A tunnel can be built on top of almost any transport mechanism.
45 Firewalls can't block attacks at higher levels of the protocol stack.
46. Network monitoring tools can be very dangerous on an exposed machine.
47 Be careful about pointing *finger* at a subverted machine .
48. Watch out for booby-trapped file names.
49. Hackers plant silent password grabbers.
50. There are lots of ways to grab /etc/passwd.
51. Logging failed logins will often capture passwords.
52. You may be liable for a hacker's activities.