# Introduction to Network Security

Matt Curtin*

March 1997
Reprinted with the permission of Kent Information Services, Inc.

**Abstract**

Network security is a complicated subject, historically only tackled by well-trained and experienced experts. However, as more and more people become "wired", an increasing number of people need to understand the basics of security in a networked world. This document was written with the basic computer user and information systems manager in mind, explaining the concepts needed to read through the hype in the marketplace and understand risks and how to deal with them.

Some history of networking is included, as well as an introduction to TCP/IP and *internetworking*. We go on to consider risk management, network threats, firewalls, and more special-purpose secure networking devices.

This is not intended to be a "frequently asked questions" reference, nor is it a "hands-on" document describing how to accomplish specific functionality.

It is hoped that the reader will have a wider perspective on security in general, and better understand how to reduce and manage risk personally, at home, and in the workplace.

*This work completed while at Megasoft Online, for Kent Information Services.

# Contents

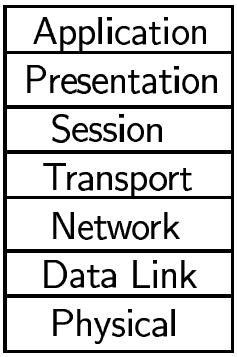


Figure 1: The ISO/OSI Reference Model

# 1 Introduction to Networking

A basic understanding of computer networks is requisite in order to understand the principles of network security. In this section, we'll cover some of the foundations of computer networking, then move on to an overview of some popular networks. Following that, we'll take a more in-depth look at TCP/IP, the network protocol suite that is used to run the Internet and many intranets.

Once we've covered this, we'll go back and discuss some of the threats that managers and administrators of computer networks need to confront, and then some tools that can be used to reduce the exposure to the risks of network computing.

## 1.1 What is a Network?

A "network" has been defined[1] as "any set of interlinking lines resembling a net, *a network of roads* || an interconnected system, *a network of alliances*." This definition suits our purpose well: a computer network is simply a system of interconnected computers. *How* they're connected is irrelevant, and as we'll soon see, there are a number of ways to do this.

## 1.2 The ISO/OSI Reference Model

The *International Standards Organization* (ISO) *Open Systems Interconnect* (OSI) Reference Model defines seven layers of communications types, and the interfaces among them. (See Figure 1.) Each layer depends on the services provided by the layer below it, all the way down to the physical network hardware, such as the computer's network interface card, and the wires that connect the cards together.

An easy way to look at this is to compare this model with something we use daily: the telephone. In order for you and I to talk when we're out of earshot, we need a device like a telephone. (In the ISO/OSI model, this is at the application layer.) The telephones, of course, are useless unless they have the ability to translate the sound into electronic pulses that can be transferred over wire and back again. (These functions are provided in layers below the application layer.) Finally, we get down to the physical connection: both must be plugged into an outlet that is connected to a switch that's part of the telephone system's network of switches.

If I place a call to you, I pick up the receiver, and dial your number. This number specifies which central office to which to send my request, and then which phone from that central office to ring. Once you answer the phone, we begin talking, and our session has begun. Conceptually, computer networks function exactly the same way.

It isn't important for you to memorize the ISO/OSI Reference Model's layers; but it's useful to know that they exist, and that each layer cannot work without the services provided by the layer below it.

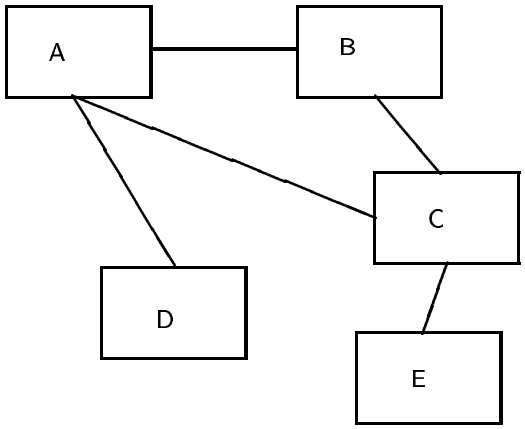


Figure 2: A Sample UUCP Network

## 1.3 What are some Popular Networks?

Over the last 25 years or so, a number of networks and network protocols have been defined and used. We're going to look at two of these networks, both of which are "public" networks. Anyone can connect to either of these networks, or they can use types of networks to connect their own *hosts* (computers) together, without connecting to the public networks. Each type takes a very different approach to providing network services.

### 1.3.1 UUCP

*UUCP* (Unix-to-Unix CoPy) was originally developed to connect Unix (surprise!) hosts together. UUCP has since been ported to many different architectures, including PCs, Macs, Amigas, Apple IIs, VMS hosts, everything else you can name, and even some things you can't. Additionally, a number of systems have been developed around the same principles as UUCP.

**Batch-Oriented Processing.** UUCP and similar systems are *batch-oriented* systems: everything that they have to do is added to a queue, and then at some specified time, everything in the queue is processed.

**Implementation Environment.** UUCP networks are commonly built using *dial-up* (modem) connections. This doesn't have to be the case though: UUCP can be used over any sort of connection between two computers, including an Internet connection.

Building a UUCP network is a simple matter of configuring two hosts to recognize each other, and know how to get in touch with each other. Adding on to the network is simple; if hosts called `A` and `B` have a UUCP network between them, and `C` would like to join the network, then it must be configured to talk to `A` and/or `B`. Naturally, anything that `C` talks to must be made aware of `C`'s existence before any connections will work. Now, to connect `D` to the network, a connection must be established with at least one of the hosts on the network, and so on. Figure 2 shows a sample UUCP network.

In a UUCP network, users are identified in the format `host!userid`. The "!" character (pronounced "bang" in networking circles) is used to separate hosts and users. A `bangpath` is a string of host(s) and a userid like `A!cmcurtin` or `C!B!A!cmcurtin`. If I am a user on host `A` and you are a user on host `E`, I might be known as `A!cmcurtin` and you as `E!you`. Because there is no direct link between your host (`E`) and mine (`A`), in order for us to communicate, we need to do so through a host (or hosts!) that has connectivity to both `E` and `A`. In our sample network, `C` has the connectivity we need. So, to send me a file, or piece of email, you would address it to `C!A!cmcurtin`. Or, if you feel like taking the long way around, you can address me as `C!B!A!cmcurtin`.

The "public" UUCP network is simply a huge worldwide network of hosts connected to each other.

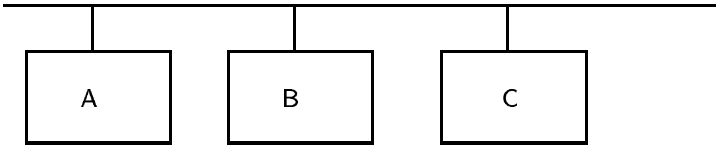


Figure 3: A Simple Local Area Network

**Popularity.** The public UUCP network has been shrinking in size over the years, with the rise of the availability of inexpensive Internet connections. Additionally, since UUCP connections are typically made hourly, daily, or weekly, there is a fair bit of delay in getting data from one user on a UUCP network to a user on the other end of the network. UUCP isn't very flexible, as it's used for simply copying files (which can be netnews, email, documents, etc.) Interactive protocols (that make applications such as the World Wide Web possible) have become much more the norm, and are preferred in most cases.

However, there are still many people whose needs for email and netnews are served quite well by UUCP, and its integration into the Internet has greatly reduced the amount of cumbersome addressing that had to be accomplished in times past.

**Security.** UUCP, like any other application, has security tradeoffs. Some strong points for its security is that it is fairly limited in what it can do, and it's therefore more difficult to trick into doing something it shouldn't; it's been around a *long* time, and most its bugs have been discovered, analyzed, and fixed; and because UUCP networks are made up of occasional connections to other hosts, it isn't possible for someone on host `E` to directly make contact with host `B`, and take advantage of that connection to do something naughty.

On the other hand, UUCP typically works by having a system-wide UUCP user account and password. Any system that has a UUCP connection with another must know the appropriate password for the `uucp` or `nuucp` account. Identifying a host beyond that point has traditionally been little more than a matter of trusting that the host is who it claims to be, and that a connection is allowed at that time. More recently, there has been an additional layer of authentication, whereby both hosts must have the same *sequence number*, that is a number that is incremented each time a connection is made.

Hence, if I run host `B`, I know the `uucp` password on host `A`. If, though, I want to impersonate host `C`, I'll need to connect, identify myself as `C`, hope that I've done so at a time that `A` will allow it, and try to guess the correct sequence number for the session. While this might not be a trivial attack, it isn't considered very secure.

### 1.3.2 The Internet

Internet: This is a word that I've heard *way* too often in the last few years. Movies, books, newspapers, magazines, television programs, and practically every other sort of media imaginable has dealt with the Internet recently.

**What is the Internet?** The Internet is the world's largest network *of networks*. When you want to access the resources offered by the Internet, you don't really connect to *the* Internet; you connect to a network that is eventually connected to the *Internet backbone*, a network of extremely fast (and incredibly overloaded!) network components. This is an important point: the Internet is a network of *networks* — not a network of hosts.

A simple network can be constructed using the same protocols and such that the Internet uses without actually *connecting* it to anything else. Such a basic network is shown in Figure 3.

I might be allowed to put one of my hosts on one of my employer's networks. We have a number of networks, which are all connected together on a *backbone*, that is a network of our networks. Our backbone is then connected to other networks, one of which is to an *Internet Service Provider* (ISP) whose backbone is connected to other networks, one of which is the Internet backbone.

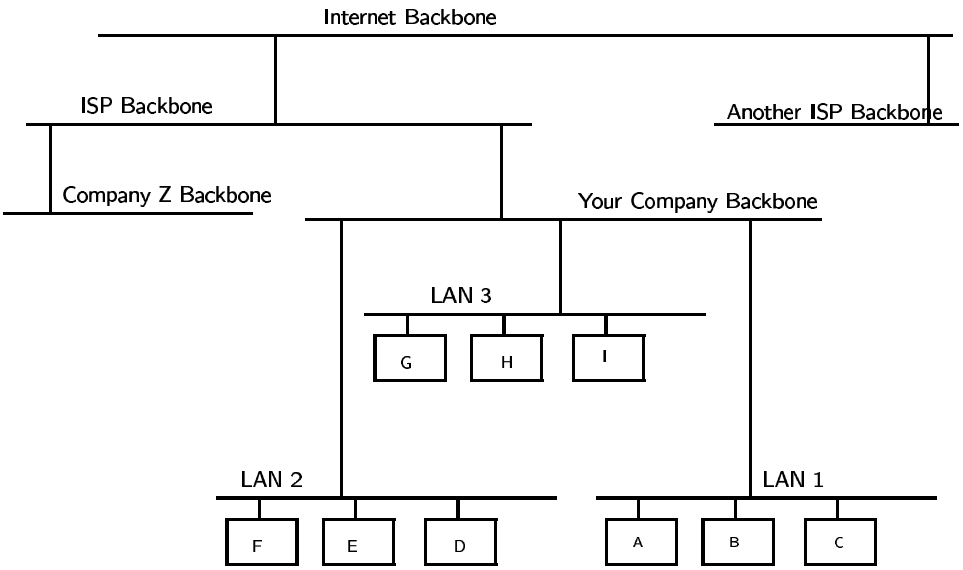


Figure 4: A Wider View of Internet-connected Networks

If you have a connection "to the Internet" through a local ISP, you are actually connecting your computer to one of their networks, which is connected to another, and so on. To use a service from my host, such as a web server, you would tell your web browser to connect to my host. Underlying services and protocols would send *packets* (small datagrams) with your query to your ISP's network, and then a network they're connected to, and so on, until it found a path to my employer's backbone, and to the exact network my host is on. My host would then respond appropriately, and the same would happen in reverse: packets would traverse all of the connections until they found their way back to your computer, and you were looking at my web page.

In Figure 4, the network shown in Figure 3 is designated "LAN 1" and shown in the bottom-right of the picture. This shows how the hosts on that network are provided connectivity to other hosts on the same LAN, within the same company, outside of the company, but in the same ISP *cloud*, and then from another ISP somewhere on the Internet.

The Internet is made up of a wide variety of hosts, from supercomputers to personal computers, including every imaginable type of hardware and software. How do all of these computers understand each other and work together?

# 2 TCP/IP: The Language of the Internet

*TCP/IP* (Transport Control Protocol/Internet Protocol) is the "language" of the Internet. Anything that can learn to "speak TCP/IP" can play on the Internet. This is functionality that occurs at the Network (IP) and Transport (TCP) layers in the ISO/OSI Reference Model. Consequently, a host that has TCP/IP functionality (such as Unix, OS/2, MacOS, or Windows NT) can easily support applications (such as Netscape's Navigator) that uses the network.

## 2.1 Open Design

One of the most important features of TCP/IP isn't a technological one: The protocol is an "open" protocol, and anyone who wishes to implement it may do so freely. Engineers and scientists from all over the world participate in the *IETF* (Internet Engineering Task Force) working groups that design the protocols that make the Internet work. Their time is typically donated by their companies, and the result is work that benefits everyone.

## 2.2 IP

As noted, IP is a "network layer" protocol. This is the layer that allows the hosts to actually "talk" to each other. Such things as carrying datagrams, mapping the Internet address (such as 10.2.3.4) to a physical network address (such as 08:00:69:0a:ca:8f), and routing, which takes care of making sure that all of the devices that have Internet connectivity can find the way to each other.

### 2.2.1 Understanding IP

IP has a number of very important features which make it an extremely robust and flexible protocol. For our purposes, though, we're going to focus on the security of IP, or more specifically, the lack thereof.

### 2.2.2 Attacks Against IP

A number of attacks against IP are possible. Typically, these exploit the fact that IP does not perform a robust mechanism for *authentication*, which is proving that a packet came from where it claims it did. A packet simply claims to originate from a given address, and there isn't a way to be sure that the host that sent the packet is telling the truth. This isn't necessarily a weakness, *per se*, but it is an important point, because it means that the facility of host authentication has to be provided at a higher layer on the ISO/OSI Reference Model. Today, applications that require strong host authentication (such as cryptographic applications) do this at the application layer.

**IP Spoofing.** This is where one host claims to have the IP address of another. Since many systems (such as router access control lists) define which packets may and which packets may not pass based on the sender's IP address, this is a useful technique to an attacker: he can send packets to a host, perhaps causing it to take some sort of action.

Additionally, some applications allow login based on the IP address of the person making the request (such as the Berkeley *r-commands*)[2]. These are both good examples how trusting untrustable layers can provide security that is — at best — weak.

**IP Session Hijacking.** This is a relatively sophisticated attack, first described by Steve Bellovin [3]. This is very dangerous, however, because there are now toolkits available in the underground community that allow otherwise unskilled bad-guy-wannabes to perpetrate this attack. IP Session Hijacking is an attack whereby a user's session is taken over, being in the control of the attacker. If the user was in the middle of email, the attacker is looking at the email, and then can execute any commands he wishes as the attacked user. The attacked user simply sees his session dropped, and may simply login again, perhaps not even noticing that the attacker is still logged in and doing things.

For the description of the attack, let's return to our large network of networks in Figure 4. In this attack, a user on host `A` is carrying on a session with host `G`. Perhaps this is a `telnet` session, where the user is reading his email, or using a Unix shell account from home. Somewhere in the network between `A` and `B` sits host `H` which is run by a naughty person. The naughty person on host `H` watches the traffic between `A` and `G`, and runs a tool which starts to impersonate `A` to `G`, and at the same time tells `A` to shut up, perhaps trying to convince it that `G` is no longer on the net (which might happen in the event of a crash, or major network outage). After a few seconds of this, if the attack is successful, naughty person has "hijacked" the session of our user. Anything that the user can do legitimately can now be done by the attacker, illegitimately. As far as `G` knows, nothing has happened.

This can be solved by replacing standard `telnet`-type applications with encrypted versions of the same thing. In this case, the attacker can still take over the session, but he'll see only "gibberish" because the session is encrypted. The attacker will not have the needed cryptographic key(s) to decrypt the data stream from `G`, and will, therefore, be unable to do anything with the session.

## 2.3 TCP

TCP is a transport-layer protocol. It needs to sit on top of a network-layer protocol, and was designed to ride atop IP. (Just as IP was designed to carry, among other things, TCP packets.) Because TCP and IP

were designed together and wherever you have one, you typically have the other, the entire suite of Internet protocols are known collectively as "TCP/IP." TCP itself has a number of important features that we'll cover briefly.

### 2.3.1 Guaranteed Packet Delivery

Probably the most important is guaranteed packet delivery. Host `A` sending packets to host `B` expects to get acknowledgments back for each packet. If `B` does not send an acknowledgment within a specified amount of time, `A` will resend the packet.

Applications on host `B` will expect a data stream from a TCP session to be complete, and in order. As noted, if a packet is missing, it will be resent by `A`, and if packets arrive out of order, `B` will arrange them in proper order before passing the data to the requesting application.

This is suited well toward a number of applications, such as a `telnet` session. A user wants to be sure every keystroke is received by the remote host, and that it gets every packet sent back, even if this means occasional slight delays in responsiveness while a lost packet is resent, or while out-of-order packets are rearranged.

It is not suited well toward other applications, such as streaming audio or video, however. In these, it doesn't really matter if a packet is lost (a lost packet in a stream of 100 won't be distinguishable) but it *does* matter if they arrive late (i.e., because of a host resending a packet presumed lost), since the data stream will be paused while the lost packet is being resent. Once the lost packet is received, it will be put in the proper slot in the data stream, and then passed up to the application.

## 2.4 UDP

*UDP* (User Datagram Protocol) is a simple transport-layer protocol. It does not provide the same features as TCP, and is thus considered "unreliable." Again, although this is unsuitable for some applications, it does have much more applicability in other applications than the more reliable and robust TCP.

### 2.4.1 Lower Overhead than TCP

One of the things that makes UDP nice is its simplicity. Because it doesn't need to keep track of the sequence of packets, whether they ever made it to their destination, etc., it has lower overhead than TCP. This is another reason why it's more suited to streaming-data applications: there's less screwing around that needs to be done with making sure all the packets are there, in the right order, and that sort of thing.

# 3 Risk Management: The Game of Security

It's very important to understand that in security, one simply cannot say "what's the best firewall?" There are two extremes: absolute security and absolute access. The closest we can get to an absolutely secure machine is one unplugged from the network, power supply, locked in a safe, and thrown at the bottom of the ocean. Unfortunately, it isn't terribly useful in this state. A machine with absolute access is extremely convenient to use: it's simply there, and will do whatever you tell it, without questions, authorization, passwords, or any other mechanism. Unfortunately, this isn't terribly practical, either: the Internet is a bad neighborhood now, and it isn't long before some bonehead will tell the computer to do something like self-destruct, after which, it isn't terribly useful to you.

This is no different from our daily lives. We constantly make decisions about what risks we're willing to accept. When we get in a car and drive to work, there's a certain risk that we're taking. It's possible that something completely out of control will cause us to become part of an accident on the highway. When we get on an airplane, we're accepting the level of risk involved as the price of convenience. However, most people have a mental picture of what an acceptable risk is, and won't go beyond that in most circumstances. If I happen to be upstairs at home, and want to leave for work, I'm not going to jump out the window. Yes, it would be more convenient, but the risk of injury outweighs the advantage of convenience.

Every organization needs to decide for itself where between the two extremes of total security and total access they need to be. A policy needs to articulate this, and then define *how* that will be enforced with practices and such. Everything that is done in the name of security, then, must enforce that policy uniformly.

# 4 Types And Sources Of Network Threats

Now, we've covered enough background information on networking that we can actually get into the security aspects of all of this. First of all, we'll get into the types of threats there are against networked computers, and then some things that can be done to protect yourself against various threats.

## 4.1 Denial-of-Service

*DoS* (Denial-of-Service) attacks are probably the nastiest, and most difficult to address. These are the nastiest, because they're very easy to launch, difficult (sometimes impossible) to track, and it isn't easy to refuse the requests of the attacker, without also refusing legitimate requests for service.

The premise of a DoS attack is simple: send more requests to the machine than it can handle. There are toolkits available in the underground community that make this a simple matter of running a program and telling it which host to blast with requests. The attacker's program simply makes a connection on some service port, perhaps forging the packet's header information that says where the packet came from, and then dropping the connection. If the host is able to answer 20 requests per second, and the attacker is sending 50 per second, obviously the host will be unable to service all of the attacker's requests, much less any legitimate requests (hits on the web site running there, for example).

Such attacks were fairly common in late 1996 and early 1997, but are now becoming less popular.

Some things that can be done to reduce the risk of being stung by a denial of service attack include

- Not running your visible-to-the-world servers at a level too close to capacity
- Using packet filtering to prevent obviously forged packets from entering into your network address space.

  Obviously forged packets would include those that claim to come from your own hosts, addresses reserved for private networks as defined in RFC 1918 [4], and the *loopback* network (127.0.0.0).
- Keeping up-to-date on security-related patches for your hosts' operating systems.

## 4.2 Unauthorized Access

"Unauthorized access" is a very high-level term that can refer to a number of different sorts of attacks. The goal of these attacks is to access some resource that your machine should not provide the attacker. For example, a host might be a web server, and should provide anyone with requested web pages. However, that host should not provide command shell access without being sure that the person making such a request is someone who should get it, such as a local administrator.

### 4.2.1 Executing Commands Illicitly

It's obviously undesirable for an unknown and untrusted person to be able to execute commands on your server machines. There are two main classifications of the severity of this problem: normal user access, and administrator access. A normal user can do a number of things on a system (such as read files, mail them to other people, etc.) that an attacker should not be able to do. This might, then, be all the access that an attacker needs. On the other hand, an attacker might wish to make configuration changes to a host (perhaps changing its IP address, putting a start-up script in place to cause the machine to shut down every time it's started, or something similar). In this case, the attacker will need to gain administrator privileges on the host.

### 4.2.2 Confidentiality Breaches

We need to examine the threat model: what is it that you're trying to protect yourself against? There is certain information that could be quite damaging if it fell into the hands of a competitor, an enemy, or the public. In these cases, it's possible that compromise of a normal user's account on the machine can be enough to cause damage (perhaps in the form of PR, or obtaining information that can be used against the company, etc.)

While many of the perpetrators of these sorts of break-ins are merely thrill-seekers interested in nothing more than to see a shell prompt for your computer on their screen, there are those who are more malicious, as we'll consider next. (Additionally, keep in mind that it's possible that someone who is normally interested in nothing more than the thrill could be persuaded to do more: perhaps an unscrupulous competitor is willing to hire such a person to hurt you.)

### 4.2.3 Destructive Behavior

Among the destructive sorts of break-ins and attacks, there are two major categories.

**Data Diddling.** The data diddler is likely the worst sort, since the fact of a break-in might not be immediately obvious. Perhaps he's toying with the numbers in your spreadsheets, or changing the dates in your projections and plans. Maybe he's changing the account numbers for the auto-deposit of certain paychecks. In any case, rare is the case when you'll come in to work one day, and simply know that something is wrong. An accounting procedure might turn up a discrepancy in the books three or four months after the fact. Trying to track the problem down will certainly be difficult, and once *that* problem is discovered, how can any of your numbers from that time period be trusted? How far back do you have to go before you think that your data is safe?

**Data Destruction.** Some of those perpetrate attacks are simply twisted jerks who like to delete things. In these cases, the impact on your computing capability — and consequently your business — can be nothing less than if a fire or other disaster caused your computing equipment to be completely destroyed.

## 4.3 Where Do They Come From?

How, though, does an attacker gain access to your equipment? *Through any connection that you have to the outside world.* This includes Internet connections, dial-up modems, and even physical access. (How do you know that one of the temps that you've brought in to help with the data entry isn't really a system cracker looking for passwords, data phone numbers, vulnerabilities and anything else that can get him access to your equipment?)

In order to be able to adequately address security, all possible avenues of entry must be identified and evaluated. The security of that entry point must be consistent with your stated policy on acceptable risk levels.

## 4.4 Lessons Learned

From looking at the sorts of attacks that are common, we can divine a relatively short list of high-level practices that can help prevent security disasters, and to help control the damage in the event that preventative measures were unsuccessful in warding off an attack.

### 4.4.1 Hope you have backups

This isn't just a good idea from a security point of view. Operational requirements should dictate the backup policy, and this should be closely coordinated with a disaster recovery plan, such that if an airplane crashes into your building one night, you'll be able to carry on your business from another location. Similarly, these can be useful in recovering your data in the event of an electronic disaster: a hardware failure, or a breakin that changes or otherwise damages your data.

### 4.4.2 Don't put data where it doesn't need to be

Although this *should* go without saying, this doesn't occur to lots of folks. As a result, information that doesn't need to be accessible from the outside world sometimes is, and this can needlessly increase the severity of a break-in dramatically.

### 4.4.3 Avoid systems with single points of failure

Any security system that can be broken by breaking through any one component isn't really very strong. In security, a degree of redundancy is good, and can help you protect your organization from a minor security breach becoming a catastrophe.

### 4.4.4 Stay current with relevant operating system patches

Be sure that someone who knows what you've got is watching the vendors' security advisories. Exploiting old bugs is still one of the most common (and most effective!) means of breaking into systems.

### 4.4.5 Watch for relevant security advisories

In addition to watching what the vendors are saying, keep a close watch on groups like CERT[1] and CIAC[2]. Make sure that at least one person (preferably more) is subscribed to these mailing lists

### 4.4.6 Have someone on staff be familiar with security practices

Having at least one person who is charged with keeping abreast of security developments is a good idea. This need not be a technical wizard, but could be someone who is simply able to read advisories issued by various incident response teams, and keep track of various problems that arise. Such a person would then be a wise one to consult with on security related issues, as he'll be the one who knows if web server software version such-and-such has any known problems, etc.

This person should also know the "dos" and "don'ts" of security, from reading such things as the "Site Security Handbook."[5]

# 5 Firewalls

As we've seen in our discussion of the Internet and similar networks, connecting an organization to the Internet provides a two-way flow of traffic. This is clearly undesirable in many organizations, as proprietary information is often displayed freely within a corporate *intranet* (that is, a TCP/IP network, modeled after the Internet that only works within the organization).

In order to provide some level of separation between an organization's intranet and the Internet, *firewalls* have been employed. A firewall is simply a group of components that collectively form a barrier between two networks.

A number of terms specific to firewalls and networking are going to be used throughout this section, so let's introduce them all together.

**Bastion host.** A general-purpose computer used to control access between the internal (private) network (intranet) and the Internet (or any other untrusted network). Typically, these are hosts running a flavor of the Unix operating system that has been customized in order to reduce its functionality to only what is necessary in order to support its functions. Many of the general-purpose features have been turned off, and in many cases, completely removed, in order to improve the security of the machine.

**Router.** A special purpose computer for connecting networks together. Routers also handle certain functions, such as *routing*, or managing the traffic on the networks they connect.

[1] Computer Emergency Response Team. <http://www.cert.org/>
[2] Computer Incident Advisory Capability. <http://ciac.llnl.gov/ciac/>

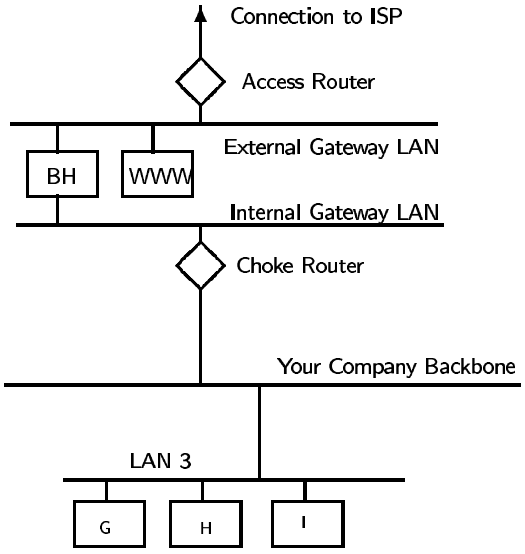


Figure 5: A sample application gateway

**Access Control List (ACL).** Many routers now have the ability to selectively perform their duties, based on a number of facts about a packet that comes to it. This includes things like origination address, destination address, destination service port, and so on. These can be employed to limit the sorts of packets that are allowed to come in and go out of a given network.

**Demilitarized Zone (DMZ).** The DMZ is a critical part of a firewall: it is a network that is neither part of the untrusted network, nor part of the trusted network. But, this is a network that connects the untrusted to the trusted. The importance of a DMZ is tremendous: someone who breaks into your network from the Internet should have to get through several layers in order to successfully do so. Those layers are provided by various components within the DMZ.

**Proxy.** This is the process of having one host act in behalf of another. A host that has the ability to fetch documents from the Internet might be configured as a *proxy server*, and host on the intranet might be configured to be *proxy clients*. In this situation, when a host on the intranet wishes to fetch the <`http://www.interhack.net/`> web page, for example, the browser will make a connection to the proxy server, and request the given URL. The proxy server will fetch the document, and return the result to the client. In this way, all hosts on the intranet are able to access resources on the Internet without having the ability to direct talk to the Internet.

## 5.1 Types of Firewalls

There are three basic types of firewalls, and we'll consider each of them.

### 5.1.1 Application Gateways

The first firewalls were application gateways, and are sometimes known as proxy gateways. These are made up of bastion hosts that run special software to act as a proxy server. This software runs at the *Application Layer* of our old friend the ISO/OSI Reference Model, hence the name. Clients behind the firewall must be *proxitized* (that is, must know how to use the proxy, and be configured to do so) in order to use Internet services. Traditionally, these have been the most secure, because they don't allow anything to pass by default, but need to have the programs written and turned on in order to begin passing traffic.

These are also typically the slowest, because more processes need to be started in order to have a request serviced. Figure 5 shows a application gateway.

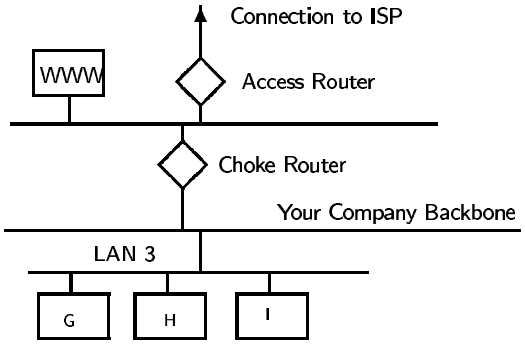


Figure 6: A sample packet filtering gateway

### 5.1.2 Packet Filtering

Packet filtering is a technique whereby routers have *ACLs* (Access Control Lists) turned on. By default, a router will pass all traffic sent it, and will do so without any sort of restrictions. Employing ACLs is a method for enforcing your security policy with regard to what sorts of access you allow the outside world to have to your internal network, and vice versa.

There is less overhead in packet filtering than with an application gateway, because the feature of access control is performed at a lower ISO/OSI layer (typically, the transport or session layer). Due to the lower overhead and the fact that packet filtering is done with routers, which are specialized computers optimized for tasks related to networking, a packet filtering gateway is often much faster than its application layer cousins. Figure 6 shows a packet filtering gateway.

Because we're working at a lower level, supporting new applications either comes automatically, or is a simple matter of allowing a specific packet type to pass through the gateway. (Not that the *possibility* of something automatically makes it a good idea; opening things up this way might very well compromise your level of security below what your policy allows.)

There are problems with this method, though. Remember, TCP/IP has absolutely no means of guaranteeing that the source address is really what it claims to be. As a result, we have to use layers of packet filters in order to localize the traffic. We can't get all the way down to the actual host, but with two layers of packet filters, we can differentiate between a packet that came from the Internet and one that came from our internal network. We can identify which network the packet came from with certainty, but we can't get more specific than that.

### 5.1.3 Hybrid Systems

In an attempt to marry the security of the application layer gateways with the flexibility and speed of packet filtering, some vendors have created systems that use the principles of both.

In some of these systems, new connections must be authenticated and approved at the application layer. Once this has been done, the remainder of the connection is passed down to the session layer, where packet filters watch the connection to ensure that only packets that are part of an ongoing (already authenticated and approved) conversation are being passed.

Other possibilities include using both packet filtering and application layer proxies. The benefits here include providing a measure of protection against your machines that provide services to the Internet (such as a public web server), as well as provide the security of an application layer gateway to the internal network. Additionally, using this method, an attacker, in order to get to services on the internal network, will have to break through the access router, the bastion host, and the choke router.

## 5.2 So, what's best for me?

Lots of options are available, and it makes sense to spend some time with an expert, either in-house, or an experienced consultant who can take the time to understand your organization's security policy, and can

design and build a firewall architecture that best implements that policy. Other issues like services required, convenience, and scalability might factor in to the final design.

## 5.3 Some Words of Caution

The business of building firewalls is in the process of becoming a commodity market. Along with commodity markets come lots of folks who are looking for a way to make a buck without necessarily knowing what they're doing. Additionally, vendors compete with each other to try and claim the greatest security, the easiest to administer, and the least visible to end users. In order to try to quantify the potential security of firewalls, some organizations have taken to firewall certifications. The certification of a firewall means nothing more than the fact that it *can* be configured in such a way that it can pass a series of tests. Similarly, claims about meeting or exceeding U.S. Department of Defense "Orange Book" standards, C-2, B-1, and such all simply mean that an organization was able to configure a machine to pass a series of tests. This doesn't mean that it was loaded with the vendor's software at the time, or that the machine was even usable. In fact, one vendor has been claiming their operating system is "C-2 Certified" didn't make mention of the fact that their operating system only passed the C-2 tests without being connected to any sort of network devices.

Such gauges as market share, certification, and the like are no guarantees of security or quality. Taking a little bit of time to talk to some knowledgeable folks can go a long way in providing you a comfortable level of security between your private network and the big, bad Internet.

Additionally, it's important to note that many consultants these days have become much less the advocate of their clients, and more of an extension of the vendor. Ask any consultants you talk to about their vendor affiliations, certifications, and whatnot. Ask what difference it makes to them whether you choose one product over another, and vice versa. And then ask yourself if a consultant who is certified in technology XYZ is going to provide you with competing technology ABC, even if ABC best fits your needs.

### 5.3.1 Single Points of Failure

Many "firewalls" are sold as a single component: a bastion host, or some other black box that you plug your networks into and get a warm-fuzzy, feeling safe and secure. *The term "firewall" refers to a number of components that collectively provide the security of the system.* Any time there is only one component paying attention to what's going on between the internal and external networks, an attacker has only one thing to break (or fool!) in order to gain complete access to your internal networks.

# 6 Secure Network Devices

It's important to remember that the firewall only one entry point to your network. Modems, if you allow them to answer incoming calls, can provide an easy means for an attacker to sneak *around* (rather than *through*) your front door (or, firewall). Just as castles weren't built with moats only in the front, your network needs to be protected at all of its entry points.

## 6.1 Secure Modems; Dial-Back Systems

If modem access is to be provided, this should be guarded carefully. The *terminal server*, or network device that provides dial-up access to your network needs to be actively administered, and its logs need to be examined for strange behavior. Its password need to be strong — not ones that can be guessed. Accounts that aren't actively used should be disabled. In short, it's the easiest way to get into your network from remote: guard it carefully.

There are some remote access systems that have the feature of a two-part procedure to establish a connection. The first part is the remote user dialing into the system, and providing the correct userid and password. The system will then drop the connection, and call the authenticated user back at a known telephone number. Once the remote user's system answers that call, the connection is established, and the user is on the network. This works well for folks working at home, but can be problematic for users wishing to dial in from hotel rooms and such when on business trips.

Other possibilities include one-time password schemes, where the user enters his userid, and is presented with a "challenge," a string of between six and eight numbers. He types this challenge into a small device that he carries with him that looks like a calculator. He then presses enter, and a "response" is displayed on the LCD screen. The user types the response, and if all is correct, he login will proceed. These are useful devices for solving the problem of good passwords, without requiring dial-back access. However, these have their own problems, as they require the user to carry them, and they must be tracked, much like building and office keys.

No doubt many other schemes exist. Take a look at your options, and find out how what the vendors have to offer will help you *enforce your security policy effectively.*

### 6.2 Crypto-Capable Routers

A feature that is being built into some routers is the ability to session encryption between specified routers. Because traffic traveling across the Internet can be seen by people in the middle who have the resources (and time) to snoop around, these are advantageous for providing connectivity between two sites, such that there can be secure routes.

See the Snake Oil FAQ [6] for a description of cryptography, ideas for evaluating cryptographic products, and how to determine which will most likely meet your needs.

### 6.3 Virtual Private Networks

Given the ubiquity of the Internet, and the considerable expense in private leased lines, many organizations have been building *VPNs* (Virtual Private Networks). Traditionally, for an organization to provide connectivity between a main office and a satellite one, an expensive data line had to be leased in order to provide direct connectivity between the two offices. Now, a solution that is often more economical is to provide both offices connectivity to the Internet. Then, using the Internet as the medium, the two offices can communicate.

The danger in doing this, of course, is that there is no privacy on this channel, and it's difficult to provide the other office access to "internal" resources without providing those resources to everyone on the Internet.

VPNs provide the ability for two offices to communicate with each other in such a way that it looks like they're directly connected over a private leased line. The session between them, although going over the Internet, is private (because the link is encrypted), and the link is convenient, because each can see each others' internal resources without showing them off to the entire world.

A number of firewall vendors are including the ability to build VPNs in their offerings, either directly with their base product, or as an add-on. If you have need to connect several offices together, this might very well be the best way to do it.

## 7 Conclusions

Security is a very difficult topic. Everyone has a different idea of what "security" is, and what levels of risk are acceptable. The key for building a secure network is to *define what security means to your organization.* Once that has been defined, everything that goes on with the network can be evaluated with respect to that policy. Projects and systems can then be broken down into their components, and it becomes much simpler to decide whether what is proposed will conflict with your security policies and practices.

Many people pay great amounts of lip service to security, but do not want to be bothered with it when it gets in their way. It's important to build systems and networks in such a way that the user is not constantly reminded of the security system around him. Users who find security policies and systems too restrictive will find ways around them. It's important to get their feedback to understand what can be improved, and it's important to let them know *why* what's been done has been, the sorts of risks that are deemed unacceptable, and what has been done to minimize the organization's exposure to them.

Security is everybody's business, and only with everyone's cooperation, an intelligent policy, and consistent practices, will it be achievable.

# References


[1] *The New Lexicon Webster's Encyclopedic Dictionary of the English Language.* New York: Lexicon.

[2] R.T. Morris, 1985. *A Weakness in the 4.2BSD Unix TCP/IP Software.* Computing Science Technical Report No. 117, AT&T Bell Laboratories, Murray Hill, New Jersey.

[3] S.M. Bellovin. *Security Problems in the TCP/IP Protocol Suite.* Computer Communication Review, Vol. 19, No. 2, pp. 32–48, April 1989.

[4] Y. Rekhter, R. Moskowitz, D. Karrenberg, G. de Groot, E. Lear, "Address Allocation for Private Internets." RFC 1918.

[5] J.P. Holbrook, J.K. Reynolds. "Site Security Handbook." RFC 1244.

[6] M. Curtin, "Snake Oil Warning Signs: Encryption Software to Avoid." USENET <sci.crypt> Frequently Asked Questions File.