PART I

NETWORK SECURITY BACKGROUND

CHAPTER 1

UNDERSTANDING TCP/IP

TCP/IP is a set of data communications protocols. These protocols allow for the routing of information from one machine to another, the delivery of e-mail and news, even the use of remote login capabilities.

The name TCP/IP refers to the two major protocols, Transmission Control Protocol and Internet Protocol. While there are many other protocols that provide services that operate over TCP/IP, these are the most common.

THE HISTORY OF TCP/IP

Internetworking with TCP/IP has been around for many years—almost as many years as Unix has been available. TCP/IP, or Transmission Control Protocol/Internet Protocol, grew out of the work that was done with the Defense Advanced Research Projects Agency, or DARPA. In 1969, DARPA sponsored a project that became known as the ARPANET. This network mainly provided high-bandwidth connectivity between the major computing sites in government, educational, and research laboratories.

The ARPANET provided those users with the ability to transfer e-mail and files from one site to another, while DARPA provided the research funding for the entire project. Through the evolution of the project, it became clear that a wide range of benefits and advantages were available, and that it was possible to provide cross-country network links.

During the 1970s, DARPA continued to fund and encourage research on the ARPANET, which consisted chiefly of point-to-point leased line interconnections. DARPA also started pushing for research into alternate forms of communication links, such as satellites and radio. It was during this time that the framework for a common set of networking technologies started to form. The result was TCP/IP. In an attempt to increase acceptance and use of these protocols, DARPA provided a low-cost implementation of them to the user community. This implementation was targeted chiefly at the University of California at Berkeley's BSD Unix implementation.

DARPA funded the creation of the company Bolt Beranek and Newman Inc. (BBN) to develop the implementation of TCP/IP on BSD Unix. This development project came at the time when many sites were in the process of adopting and developing local area network technologies, which were based closely on extensions of the previous single computer environments that were already in use. By January 1983, all the computers connected to the ARPANET were running the new TCP/IP protocols. In addition, many sites that were not connected to the ARPANET also were using the TCP/IP protocols.

Because the ARPANET generally was limited to a select group of government departments and agencies, the National Science Foundation created the NSFNet that also was using the successful ARPANET protocols. This network, which in some ways was an extension of the ARPANET, consisted of a backbone network connecting all the super-computer centers within the United States and a series of smaller networks that were then connected to the NSFNet backbone.

Because of the approaches taken with NSFNet, numerous network topologies are available, and TCP/IP is not restricted to any single one. This means that TCP/IP can run on token ring, Ethernet and other bus topologies, point-to-point leased lines, and more. However, TCP/IP has been closely linked with Ethernet—so much so that the two were used almost interchangeably.

Since that time, the use of TCP/IP has increased at a phenomenal rate, and the number of connections to the Internet, or this global network of networks, has also increased at an almost exponential rate. A countless number of people are making a living off the Internet, and with the current trends in information dissemination, it likely will touch the lives of every person in the developed world at some time.

TCP/IP, however, is not a single protocol. In fact, it consists of a number of protocols, each providing some very specific services. The remainder of this chapter examines how addressing is performed in TCP/IP, network configuration, the files controlling how TCP/IP can be used, and many of the various administrative commands and daemons.

NOTE

A daemon is a program that runs to perform a specific function. Unlike many commands that execute and exit, a daemon performs its work and waits for more. For example, sendmail is a daemon. It remains active even if there is no mail to be processed.

EXPLORING ADDRESSES, SUBNETS, AND HOST NAMES

Each machine on the Internet must have a distinctly different address, like your postal address, so that information destined for it can be successfully delivered. This address scheme is controlled by the Internet Protocol (IP).

NOTE

As of the time of this writing, the Internet Engineering Task Force (IETF) has almost completed the specification for the next version of IP, which will include a much larger address space. The larger address space is needed in response to the large number of computers that now are being connected to the Internet.

Each machine has its own IP Address, and that IP address consists of two parts: the network portion and the host portion. The network part of the address is used to describe the network on which the host resides, and the host portion is used to identify the particular host. To ensure that network addresses are unique, a central agency is responsible for the assignment of those addresses.

Because the original Internet designers did not know how the Internet would grow, they decided to design an address scheme flexible enough to handle a larger network with many hosts or a smaller network with only a few hosts. This addressing scheme introduces address classes, of which there are four.

IP addresses can be expressed in several different forms. First is the dotted decimal notation, which shows a decimal number with each byte separated by a period, as in 192.139.234.102. Alternatively, this address also can be expressed as a single hexadecimal number such as 0xC08BEA66. The most commonly used address format, however, is the dotted decimal notation.

ADDRESS CLASSES

As mentioned, there are four major address classes: class A, B, C, and D. Classes A, B, and C are used to identify the computers that share a common network. A class D, or multicast address, is used to identify a set of computers that all share a common protocol. Because the first three classes are more commonly used, this chapter focuses on them. Regardless of the address class, each address consists of 32 bits, or 4 bytes. Each byte is commonly referred to as an octet, so an IP address consists of four octets.

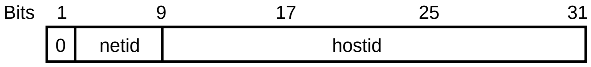
Each octet can have a value from 0 to 255. Certain values, however, have a special meaning that is shown in table 1.1 later in this chapter.

CLASS A ADDRESSES

In a class A address, the first octet represents the network portion, and the remaining three identify the host (see fig. 1.1).

FIGURE 1.1

The class A address format.



This address class means that this network can have millions of hosts because there are 24 bits available to specify the host address. In figure 1.1, you see that the first bit of the first octet is set to 0. This means that the network portion of the address must be less than 128. Actually, the network portion of a class A address ranges from 1 to 127.

CLASS B ADDRESSES

A class B address is similar in structure to a class A, with the exception that a class B address uses two octets for the network portion and two octets for the host portion (see fig. 1.2). This means that there can be more class B networks, each with thousands of hosts.

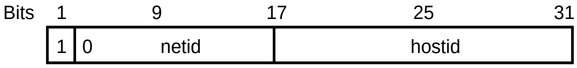


FIGURE 1.2

The class B address format.

As illustrated in figure 1.2, the configuration of the class B address is such that each portion shares the same amount of the address. The first two bits of the network address are set to 1 and 0, meaning that the network address ranges from 128 to 191. With this format, each network can have thousands of hosts.

CLASS C ADDRESSES

A class C address uses three octets for the network portion, and one octet for the host. The result is that there can be more class C networks, each with a small number of hosts. Because the maximum value of a single octet is 255, and there are two reserved values, there can be 253 hosts for a class C network. This network format is illustrated in figure 1.3.

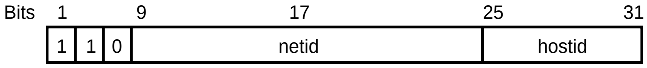


FIGURE 1.3

The class C address format.

As illustrated in figure 1.3, the first two bits of the network address are set to one. This means that the network address for a class C network ranges from 192 to 223. The remaining values from 224 to 255 are used in the fourth address class.

SPECIAL ADDRESSES

It has been mentioned that there are several different addresses reserved for special purposes. These addresses are listed in table 1.1.

TABLE 1.1

Reserved Addresses

Dotted Decimal Address	*Explanation*
0.0.0.0	All hosts broadcast address for old Sun networks.
num.num.num.0	Identifies the entire network.
num.num.num.255	All hosts on the specified network. (Broadcast Address)
255.255.255.255	All hosts broadcast for current networks.

These reserved addresses cannot be used to address any host or network. They have been specifically reserved. There can be other reserved addresses depending upon other factors, which you will see later in this chapter.

SUBNETS

Each host on a network has a specific IP address to enable other hosts to communicate with it. Depending upon the class of network, there can be anywhere from 253 to millions of hosts on a network. It would not be practical, however, for a class A or class B address to be restricted to one network with thousands or millions of hosts. To solve this problem, subnets were developed to split the host portion of the address into additional networks.

Subnets work by taking the host portion of the address and splitting it through the use of a netmask. The netmask essentially moves the dividing line between the network and the hosts from one place to another within the address. This has the effect of increasing the number of available networks, but reduces the number of hosts that can be connected to each network.

The use of subnets does provide advantages. Many smaller organizations can only obtain a class C address, yet they have several distinct offices that must be linked together. If they only have one IP address, a router will not connect the two locations because the router requires that each network have a distinct address. By splitting the network into subnets, they can use a router to connect the two networks because they now have distinctly different network addresses.

The subnet is interpreted through the netmask, or subnet mask. If the bit is on in the netmask, that equivalent bit in the address is interpreted as a network bit. If the bit is off, it is considered part of the host address. It is important to note that the subnet is known only locally; to the rest of the Internet, the address looks like a standard IP address.

As noted in table 1.2, each class of IP addresses has a default netmask associated with it.

TABLE 1.2
Standard Netmasks

Address Class	*Default Netmask*
A	255.0.0.0
B	255.255.0.0
C	255.255.255.0

In order to understand fully and appreciate how this works, you need to consider an example. Assume that you have a network address of 198.53.64.0, and you want to break this up into subnets. To further subdivide this class C network, it is necessary for you to use some of the bits in the host portion, or last byte, of the address as part of the network portion. While this increases the number of networks you can have, it decreases the number of hosts that can be on each subnet.

The Internet RFC 950 also requires that the first and last division of each subnet be reserved. This means that the actual number of useable subnets is two less than the total number of divisions. For example, if you want to split your class C network into two divisions, you cannot connect any hosts! If you want to have six subnets, then you must split your network into eight divisions.

The following example illustrates how the bits in the last octet are set, and how many subnets and hosts can be created for each. The variable portion that represents the bits used for the host portion is identified by the letter V. The fixed portion of the address is identified by the letter F.

```
8  7  6  5  4  3  2  1  Divisions   Subnets   Hosts/Subnets
----------------------------------------------------------
F  V  V  V  V  V  V  V      2          0           0
F  F  V  V  V  V  V  V      4          2          62
F  F  F  V  V  V  V  V      8          6          30
F  F  F  F  V  V  V  V     16         14          14
F  F  F  F  F  V  V  V     32         30           6
F  F  F  F  F  F  V  V     64         62           2
F  F  F  F  F  F  F  V    128        126           0
```

The preceding example shows that you can effectively only use a minimum division of four with two subnets and 62 hosts per net, or a maximum of 64 divisions which results in 62 subnets of two hosts each. The first example could be used for two separate ethernets, while the second could be used for a series of point-to-point protocol links.

However, the selection of the type of subnets that should be chosen is determined by the maximum number of users that will be required on any subnet, and the minimum number of subnets required.

The possible network portions formed in the development of your divisions are formed by evaluating the values of the fixed portion of the last byte. Looking back to the last example, you see that to split our class C address into eight divisions, or 6 subnets, you need to fix the first three bits in the last octet. The network portions are formed through the evaluation of the non-fixed portion of the last byte. Consider the following example, which lists the bit combinations and illustrates how the Class address is split into the subnets.

```
Network   Host
8  7  6 ¦ 5  4  3  2  1       Decimal Values
----------------------------------------------
0  0  1 ¦ 0  0  0  0  0             32
0  1  0 ¦ 0  0  0  0  0             64
0  1  1 ¦ 0  0  0  0  0             96
1  0  0 ¦ 0  0  0  0  0            128
1  0  1 ¦ 0  0  0  0  0            160
1  1  0 ¦ 0  0  0  0  0            192
```

As shown in the preceding example, the top three bits—8, 7, and 6—are fixed in that they are used as part of the host address. This means that the available networks become the following, where N, O, and P represent the first three octets in the address respectively:

Network
N.O.P.32
N.O.P.64
N.O.P.96
N.O.P.128
N.O.P.160
N.O.P.192

The standard netmask for a class C address is 255.255.255.0. For our subnetted network, the first three bytes remain the same. The fourth byte is created by setting the network portion to 1s and the host portion to 0. Looking back at the preceding example, you see what the network addresses will be. You use the same format for determining the netmask. This means that the netmasks for these subnets are the following:

Network	*Broadcast*	*Netmask*
N.O.P.32	N.O.P.31	255.255.255.32
N.O.P.64	N.O.P.63	255.255.255.64
N.O.P.96	N.O.P.95	255.255.255.96
N.O.P.128	N.O.P.127	255.255.255.128
N.O.P.160	N.O.P.159	255.255.255.160
N.O.P.192	N.O.P.191	255.255.255.192

The end result is that you have split this class C address into six subnetworks, thereby increasing your available address space without having to apply for an additional network address.

When looking at the netmask, it is easy to see why many administrators stick with byte-oriented netmasks—they are much easier to understand. By using a bit-oriented approach to the netmask, however, many different configurations can be achieved. Using a netmask of 255.255.255.192 on a class C address, for example, creates four subnets. The same netmask on a class B address, however, creates more than a thousand subnets!

In an attempt to help control the assignment of IP Addresses to organizaitons that were not connected to the Internet, a series of Private Network Addresses were reserved. This was accomplished in RFC1597.

RFC1597, which is included on the CD-ROM that accompanies this book, provides for three network addresses to be reserved for the use of organizations that are not connected to the Internet or who will be installing a firewall. These networks are a Class A at 10.0.0.0 to 10.255.255.255, a Class B at 172.16.0.0 to 172.31.255.255, and a Class C. at 192.168.1.90 to 192.168.254.0. By using the Private Network Addresses, the available public address space can be used more effectively.

Classless Addresses and CIDR

With the rapid expansion of the Internet, and more and more organizations wanting to have a TCP/IP-based network, a problem has begun to emerge. Aside from the rapid use of available network address space, a more ominous problem is the size of the global routing tables for routing information throughout the world.

The problem with the global routing tables is that each network requires its own individual routing entry, which causes the routing tables to be very large, particularly in the backbone network. This problem is made even worse by the fact that many Internet Service Providers advertise the route to a PPP host whenever that host dials in. This causes the routes to change very quickly, which results in excessive load in the backbone.

Classless Inter-Domain Routing (CIDR) was introduced to solve this problem. The purpose of CIDR is to change our view of addresses from “classful” to “classless.” This method of viewing network addresses also is known as *aggregation*. CIDR eliminates the concept of class A, B, and C networks and replaces this with a generalized IP prefix.

> The concept of CIDR was first presented back in 1992, at which time it was known as *Supernetting*. The name CIDR was adopted in 1993.

Note

The IP prefix refers to the number of bits of the network part of the IP address. A former class B address may appear as 172.50.0.0/16, which is the same as 256 class Cs, which can appear as 192.200.0.0/16. A single class C appears as 192.201.1.0/24. The IP prefix consists of an IP address and a mask length. The mask length specifies the number of leftmost contiguous significant bits in the corresponding IP address. Thus, an IP prefix with a prefix length of 15 (denoted /15) covers the address space of 128,000 IP addresses, and a /17 covers the address space of 32,000 IP addresses.

CIDR lessens the local administrative burden of updating external routing, saves routing table space in all backbone routers, and reduces route flapping (rapid changes in routes), and thus CPU load, in all backbone routers. The benefit of using aggregation in advertising network routes is felt by the entire Internet. Aggregation keeps the local routing changes local to the Internet Service Provider: the rest of the Internet does not see a new route go up or down. One Internet Service Provider was broadcasting approximately 12,000 routes before moving to aggregation. That provider is now advertising approximately 6,000 and expects to decrease this number even more. CIDR also allows delegation of pieces of what used to be called *network numbers* to customers, and therefore makes it possible to utilize available address space more efficiently.

Internet service providers are now responsible for assigning IP addresses or network numbers to their customers. These addresses are part of the CIDR block assigned to each individual provider.

There are a number of routing protocols in use on the Internet, including RIP, OSPF, IGRP, and EGRP. The only one that supports the use of CIDR, however, is BGP4. More information on CIDR can be obtained from the CIDR FAQ found at `http://www.rain.net/faqs/cidr.faq.html`.

HOST NAMES

Each device connected to the Internet must be assigned a unique IP address, but IP addresses can be difficult to remember. Consequently, each device is generally assigned a host name, which is used to access that device. The network does not require the use of names, but they do make the network easier to use.

For TCP/IP to work properly, the host name must be translated into the corresponding IP address. This can be accomplished through several different methods, including looking up the host name in a file called the host table or resolving it through the use of the Domain Name Service (DNS).

NOTE

> Methods for translating the host name into the corresponding IP address and DNS are discussed later in this chapter.

Within each organization, the host name must be unique. The host name consists of two pieces: the actual host name and the TCP/IP domain. The domain is assigned by a central registry depending on the country you are in and the type of organization you are registering. The most commonly used domains are .com, .edu, and .gov for commercial, educational, and government institutions within the United States. While it is possible to obtain a domain using these outside the United States, it is best not to.

For organizations outside the United States, there may be other rules governing how domains are assigned. For example, a company in Canada named Widgets Inc. could apply for widgets.ca, where .ca denotes that the organization is in Canada. If the same company was in the United Kingdom, then the domain would likely be widgets.co.uk, indicating that it is a commercial organization within the United Kingdom.

Regarding the actual names for the given hosts, the Internet Request for Comments (RFC) number 1178 provides some excellent guidelines regarding how to name systems. Here are some guidelines you should remember:

- Use real words that are short, easy to spell, and easy to remember. The point of using host names instead of IP addresses is that they are easier to use. If host names are difficult to spell and remember, they defeat their own purpose.
- Use theme names. All hosts in a group could be named after human movements such as fall, jump, or hop, or cartoon characters, foods, or other groupings. Theme names are much easier to think up than unrestricted names.
- Avoid using project names, personal names, acronyms, or other such cryptic jargon. This type of host name typically is renamed in the future, which can sometimes be more difficult than it sounds.

NOTE

> The only requirement is that the host name be unique within the domain. A well-chosen name, however, can save future work and make the user community happier.

The host name of your computer can be determined by using the hostname command, as shown in the following:

```
$ hostname
oreo.widgets.ca
$
```

On some TCP/IP implementations, the hostname command does not print the information as shown above, but only prints the actual name of the system. The output of hostname is the name of the system and the TCP/IP domain name.

Working with Network Interfaces

Each device that is to be connected to a network must have a network interface. This network interface must be consistent with the media on which the network is running. A network card for token ring, for example, cannot be connected to a thin coaxial cable network.

The following are the commonly used media types:

Token Ring
Thinnet (RG-58U coaxial cable)
Ethernet (RG-8U coaxial cable)
Twisted-pair cable
Fiber optics

Each network interface has a name for the device and an IP address. If there is more than one network interface in a device, each network interface must be part of a different network. That is, the IP addresses must be different, as shown in figure 1.4.

The exact name used for each device is vendor implemented, and often is different depending upon the type of interface that is in use. Table 1.3 lists some of the network interface names that are used, and on what systems those names are found.

Table 1.3

Network Interface Names

Interface Name	*Operating System*
le0, ie0	SunOS 4.1.3, 4.1.4
wdn0,e3a,sl0,ppp1	SCO Unix
du0	DEC Ultrix
le0	Linux
we0	BSD/OS Version 2.0
lan0	HP HP/UX 9.0

Bear in mind that these are only examples of a small collection of operating systems. There are many other network device names that are dependent upon the type of hardware installed on your system.

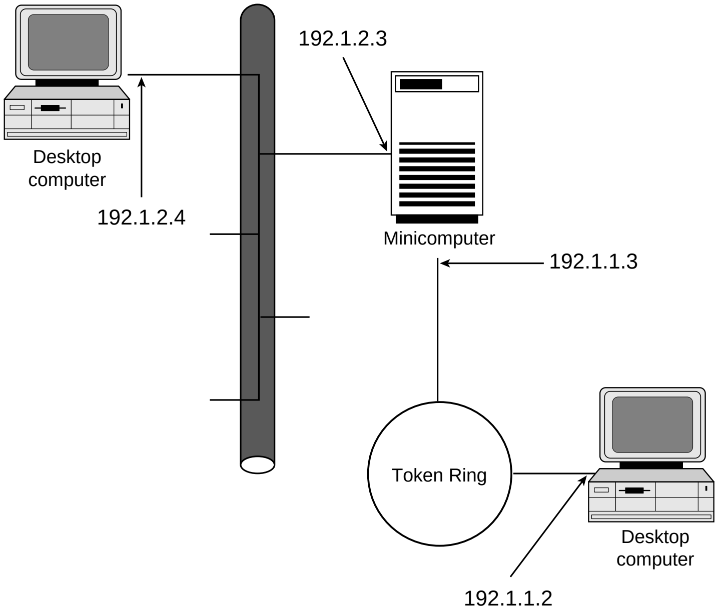


FIGURE 1.4

Network interfaces.

Consequently, as the system is configured, the network administrator must decide what the name of the device is, or must understand how to query the system to determine the name of the device. Having this information is essential to successfully configuring the network interface.

CONFIGURATION USING IFCONFIG

Except for Serial Line Internet Protocol (SLIP) and Point-to-Point Protocol (PPP) interfaces, the ifconfig command is used to configure the interface, including its IP address, broadcast address, netmask, and whether or not the interface is operational. There are some variations to this command, so it is wise to check out your system's documentation when setting up your interface.

Normally ifconfig is used at boot time to configure the interface. ifconfig also can be used after the system is running to change the IP address, or other interface configuration information.

The command syntax for ifconfig is the following:

```
ifconfig interface address-family address destination-address parameters
```

The interface value identifies the name of the interface that is being configured—wdn0, for example. The address family identifies the type of addressing used for this interface. Currently, the only value supported for this argument is inet.

The address value can consist of a host name that is found in /etc/hosts, or an Internet address expressed in dot notation. If the name form is used and the host name is not found in the /etc/hosts file, an error is returned.

Table 1.4 lists the commonly available parameters that can be configured with ifconfig. This list of options should by no means be considered exhaustive.

TABLE 1.4
ifconfig Commands

Command	*Function*
up	This marks the interface as being up, or operational. When the first address of the interface is configured, the interface is marked as up. It also can be used to reset an interface after it was previously marked down.
down	This marks an interface down. When the interface is marked down, the system does not attempt to transmit messages through that interface. If possible, the interface will be reset to prevent the reception of incoming packets as well. Use of this command does not automatically disable the routes that use this interface.
trailers	This requests the use of a trailer-link-level encapsulation when transmitting. If the interface is capable of supporting trailers, the system encapsulates the outgoing messages in a manner that minimizes the number of memory-to-memory copy operations performed by the receiver.
-trailers	Disables the use of trailer encapsulation.
arp	This enables the use of the Address Resolution Protocol in mapping between network-level addresses and link-level addresses.
-arp	This disables the use of the Address Resolution Protocol.

Command	*Function*
metric N	This sets the routing metric for this interface, which is by default 0. The routing metric is used by the route daemon, routed. The higher the metric, the less favorable the route is.
debug	This enables network-driver-level debugging.
-debug	This disables driver-dependent debugging code.
netmask MASK	This specifies how much of the address is used for the division of a network into subnets. The netmask contains 1s for the bit positions that are used for the network and subnet parts, and 0s for the host portion.
dest-address	This specifies the destination address of the correspondent on the other end of a point-to-point link.
broadcast	Specifies the address to use when sending a packet to all of the hosts on a network. The default value is the network portion and all 1s for the host portion. If the network portion is 192.139.234, for example, then the broadcast address is 192.139.234.255.

The following illustrates using ifconfig on an SCO system that has only one interface.

```
ifconfig lo0 localhost
ifconfig wdn0 198.73.138.2 -trailers netmask 255.255.255.0 broadcast
➥198.73.138.255
```

The preceding code has two lines. The first illustrates defining the localhost loopback interface, and the second defines an interface named wdn0 using an IP address of 198.73.138.2. The trailer encapsulation option is turned off (-trailers), the netmask is 255.255.255.0, and the broadcast address is the default, using all 1s for the host portion.

The following code illustrates using ifconfig on a SunOS 4.1.3 system.

```
ifconfig le0 198.73.138.6 -trailers netmask 0xffffff00 broadcast 198.73.138.255
```

The options used on the SunOS system are the same as with SCO systems, except that the netmask defined on the Sun system can use either a hexadecimal notation or the dot notation.

NOTE

> The use of the ifconfig is restricted to the super-user when used to configure the interface. A normal user can use the ifconfig command to query the status of the interface.

REVIEWING THE NETWORK CONFIGURATION FILES

A large number of files assist in the configuration and control of TCP/IP on the system. Next, this chapter examines those files, their use, and their formats. Understanding the services that are controlled from these files is essential to locate hidden security problems later. Some of these files also have inherent security problems, which will also be discussed.

THE /ETC/HOSTS FILE

The purpose of the /etc/hosts file is to provide simple host name to IP address resolution. Remember that TCP/IP only requires the use of IP addresses. The use of host names is for your convenience and ease of use. When a host name is used, TCP/IP examines the contents of the /etc/hosts file (assuming that Domain Name Service is not in use) to find the IP address for the host.

The format of an entry in the /etc/hosts file is as follows:

```
address     official name     alias ...
```

The columns refer to the IP address, the official or fully qualified domain name (FQDN), and any aliases for the machine. This is illustrated in the following sample hosts file:

```
# IP ADDRESS       FQDN                      ALIASES
127.0.0.1          localhost
192.139.234.50     gateway.widgets.ca         gateway
142.77.252.6       gateway.widgets.ca         router
142.77.17.1        nb.ottawa.uunet.ca
198.73.137.1       gateway.widgets.ca         ppp1
198.73.137.2       newton.widgets.ca          newton
198.73.137.50      gateway.widgets.ca         net2
```

Notice that the machine gateway has four IP addresses assigned to it. This implies that there are four network interfaces attached to this machine. The piece missing here is the netmask, making it reasonable to assume that two of the interfaces are on the same network.

The aliases include the short form of the host name, as well as any other names for the host. The routines that search this file skip text that follows a “#”, which represents a comment, as well as blank lines.

The network configuration files all support the use of comments with the "#" symbol. This allows the network administrator to document changes and notes.

THE /ETC/ETHERS FILE

After the IP address is known, TCP/IP converts this to the actual ethernet hardware address when the host is on the local network. This can be done by using the Address Resolution Protocol (ARP), or by creating a list of all of the ethernet addresses in the file /etc/ethers. The format of this file is the ethernet address followed by the official host name, as illustrated here:

```
# Ethernet Address      Host Name
8:0:20:0:fc:6f          laidbak
2:7:1:1:18:27           grinch
0:aa:0:2:30:55          slaid
e0:0:c0:1:85:23         lancelot
```

The information in this file actually is used by the Reverse Address Resolution Protocol daemon, rarpd, which is explained later in this chapter. The ethernet address notation used is x:x:x:x:x:x, where x is a hexadecimal number representing one byte in the address. The address bytes are always in network order, and there should be an entry in the hosts file for each device in this file.

THE /ETC/NETWORKS FILE

This file provides a list of IP addresses and names for networks on the Internet. Each line provides the information for a specific network, as follows:

```
# NETWORK NAME             IP ADDRESS
loopback                   127
Ottawa.widgets.ca          192.139.234
Toronto.widgets.ca         192.139.235
WAN.widgets.ca             198.73.137
Lab.widgets.ca             198.73.138
Montreal.widgets.ca        198.73.139
```

Each entry in the file consists of the network IP address, the name for the network, any aliases, and comments.

THE ETC/PROTOCOLS FILE

The /etc/protocols file provides a list of known DARPA Internet protocols. This file should not be changed, as it gives the information provided by the DDN Network Information Center. As shown here, each line contains the protocol name, the protocol number, and any aliases for the protocol.

```
# Internet (IP) protocols
#
ip       0      IP      # internet protocol, pseudo protocol number
icmp     1      ICMP    # internet control message protocol
ggp      3      GGP     # gateway to gateway protocol
tcp      6      TCP     # transmission control protocol
egp      8      EGP     # Exterior Gateway Protocol
pup      12     PUP     # PARC universal packet protocol
udp      17     UDP     # user datagram protocol
hello    63     HELLO   # HELLO Routing Protocol
```

THE ETC/SERVICES FILE

The /etc/services file provides a list of the available services on the host. For each service, a line in the file should be present that provides the following information:

Official service name

Port number

Protocol name

Aliases

As with the other files, each entry is separated by a space or tab. The port number and protocol name are considered a single item, as a slash (/) is used to separate them. A portion of the /etc/services file is shown in the following:

```
#
# Network services, Internet style
#
echo      7/tcp
echo      7/udp
discard   9/tcp      sink     null
discard   9/udp      sink     null
systat    11/tcp     users
ftp       21/tcp
telnet    23/tcp
smtp      25/tcp     mail
time      37/tcp     timserver
time      37/udp     timserver
rlp       39/udp     resource       # resource location
```

```
whois     43/tcp     nicname
domain    53/tcp     nameserver      # name-domain server
domain    53/udp     nameserver
```

It's obvious that this file relies upon information from /etc/protocols to function. If the service is not available, or you want to remove support for a specific service, then the appropriate line can be commented out using the comment symbol. In many cases, however, the file /etc/inetd.conf also has to be updated to disable support for a given protocol.

THE ETC/INETD.CONF FILE

The inetd.conf file is used to provide the information to the inetd command. As discussed later in the chapter, inetd is the Internet super-server. It listens on a specified TCP/IP port and starts the appropriate command when a connection is requested on that port. This saves system resources by only starting the daemons when they are needed.

The following illustrates an inetd.conf file from an SCO system, along with the file format:

```
#
ftp         stream     tcp     nowait     NOLUID     /etc/ftpd -l -v     ftpd
telnet      stream     tcp     nowait     NOLUID     /etc/telnetd        telnetd
shell       stream     tcp     nowait     NOLUID     /etc/rshd           rshd
login       stream     tcp     nowait     NOLUID     /etc/rlogind        rlogind
exec        stream     tcp     nowait     NOLUID     /etc/rexecd         rexecd
# finger    stream     tcp     nowait     nouser     /etc/fingerd        fingerd
```

The SCO inetd.conf file differs from the format of most systems because of the C2 security components found in the SCO Unix operating system. Specifically, SCO requires the Login UID, or LUID, to be set for each user who accesses the system. Because setting the LUID should not be done until a user actually logs in, the LUID must not be set when the daemon starts up. This is accomplished by using the NOLUID parameter in the file.

The following illustrates the standard inetd.conf file that is found on most other Unix systems.

```
#
ftp         stream     tcp     nowait     root       /usr/etc/in.ftpd       in.ftpd
telnet      stream     tcp     nowait     root       /usr/etc/in.telnetd    in.telnetd
shell       stream     tcp     nowait     root       /usr/etc/in.rshd       in.rshd
login       stream     tcp     nowait     root       /usr/etc/in.rlogind    in.rlogind
exec        stream     tcp     nowait     root       /usr/etc/in.rexecd     in.rexecd
# finger    stream     tcp     nowait     nobody     /usr/etc/in.fingerd    in.fingerd
```

UNDERSTANDING THE NETWORK ACCESS FILES

Two files can have a significant impact on the security of your system. These files are the /etc/hosts.equiv and the .rhosts files.

THE /ETC/HOSTS.EQUIV FILE

The /etc/hosts.equiv file contains a list of trusted hosts. This file is used by the r* commands rlogin, rcp, rcmd, and rsh, and is discussed in detail later in this chapter. The format of this file consists of a list of machine names, one per line, as illustrated here:

```
localhost
oreo
wabbit.widgets.ca
chare
chelsea
widgets
pirate
```

NOTE

> It’s a good habit to use a fully qualified name, but if the domain is omitted, TCP/IP adds it to the host name when validating the remote system.

THE .RHOSTS FILE

The .rhosts file that is used in the users HOME directory accomplishes a similar purpose to /etc/hosts.equiv. The file format is the same, with one notable exception. The hosts.equiv file is used to provide equivalence between hosts, while the .rhosts file is used to provide equivalence between users. The .rhosts file is appended to the information found in /etc/hosts.equiv when checking for equivalence.

NOTE

> The hosts.equiv file is not used for the root user. The only file processed in this case is /.rhosts.

USER AND HOST EQUIVALENCY

Host Equivalency, or Trusted Host Access, is configured by the system administrator by using the file /etc/hosts.equiv. This file consists of host names, one per line.

> It is a good idea to document in the file who the network administrator is, as comments can be included by using the comment symbol (#).

TIP

Each entry in the hosts.equiv file is trusted. That is, users on the named machine can access their equivalent accounts on this machine without a password. This is not applicable for root, however, as is explained later.

The two machines shown in figure 1.5, oreo and wabbit, both have a user named chare. If the user chare currently is logged into wabbit, and issues the command

```
$ rlogin oreo
```

with host equivalency established, then chare will be logged into oreo without being asked for his password. If host equivalency is not there, chare will be asked for his password on the remote machine.

The following must be considered with /etc/hosts.equiv:

- It assumes that you trust ALL the users on the remote machine.
- Root is never trusted through the use of this file.

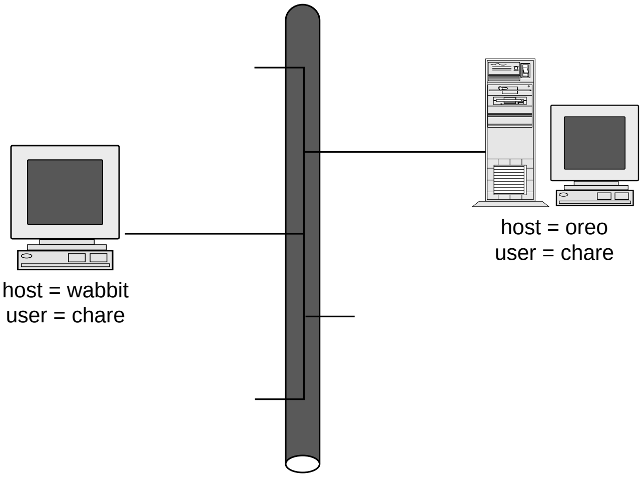


FIGURE 1.5

A sample network.

There is a second format for the hosts.equiv file, known as .rhosts, which was shown previously. This format lists a system name and a user name. With the addition of the user name, the user is allowed to log in with any user name found in /etc/passwd.

User equivalence is a mechanism in which the same user is known to all of the machines in the network. This makes the network administrator's job easier in the long run. It should be considered absolutely necessary for environments where NFS is used or is planned.

To configure user equivalence, the user creates a file in his home directory called .rhosts. This file must be writeable only by the owner of the file. If it is not, then the file is ignored for validation purposes. As with the hosts.equiv file, this file contains a system name per line, but generally also includes the name of the user who is being equivalenced. The issue of host and user equivalence will be presented again in more detail in Chapter 2, "Security."

EXAMINING TCP/IP DAEMONS

Because of the varying implementations of TCP/IP that are available, a wide range of daemons can comprise the system. As many as possible are listed here along with a brief explanation of what they do. If they are operating system specific, the operating system version information also is included.

Many of the TCP/IP daemons are named with the name of the service they provide followed by the letter "d," as in bootpd. This convention is used to indicate that this command is a daemon.

THE SLINK DAEMON

The slink daemon provides the necessary components to link the STREAMS modules required for streams TCP/IP. When the system is started, a configuration file, typically /etc/strcf, is processed, thus establishing the links between STREAMS modules for each of the network devices present in the system.

This daemon is only found on versions of Unix that use STREAMS-based TCP/IP, such as most System V derivatives.

THE LDSOCKET DAEMON

The ldsocket command initializes the System V STREAMS TCP/IP Berkeley networking compatibility interface. This daemon also is found only on System V-based implementations of TCP/IP, as the BSD-based versions do not use a STREAMS-based implementation. As the ldsocket program is loaded, a file, generally /etc/sockcf, is processed, and the streams modules are loaded and configured to provide the socket style interface.

The cpd Daemon

This is a copy protection daemon that is specific to the Santa Cruz Operation versions of TCP/IP. When TCP/IP starts, it registers with the copy protection daemon. When the cpd receives a datagram from a remote system with the same serial number, a warning message is printed advising the system administrator of the problem. SCO is the only system with this feature.

The Line Printer Daemon (lpd)

The lpd is the line printer daemon, or spool area handler, and is executed at boot time. It accepts incoming print jobs on a specific TCP/IP port, and queues the print job for printing on the local or remote system. The printer configuration information is stored in the file /etc/printcap, and the access control to the printer is maintained through the file /etc/hosts.lpd.

The SNMP Daemon (snmpd)

The SNMP daemon is an implementation of the Internet Simple Network Management Protocol, as defined in RFCs 1155-1157, 1213, and 1227. While this daemon is capable of receiving information from SNMP agents on other systems, many systems do not include SNMP Management software.

The RARP Daemon (rarpd)

The RARP command is a daemon that responds to Reverse Address Resolution Protocol (RARP) requests. Other systems typically use RARP at boot time to discover their (32-bit) IP address given their (48-bit) Ethernet address. The booting machine sends its Ethernet address in an RARP request message. For the request to be answered, the system running rarpd must have the machine's name-to-IP-address entry in the /etc/hosts file or must be available from the domain name server and its name-to-Ethernet-address entry must exist in the /etc/ethers file. Using the above two sources, rarpd maps this Ethernet address into the corresponding IP address.

THE BOOTP DAEMON (BOOTPD)

The BOOTP daemon implements an Internet Boot Protocol server as defined in RFC 951 and RFC 1048. The bootpd daemon is started by the inetd super-server when a boot request arrives. If bootpd does not receive another boot request within 15 minutes of the last one it received, it exits to conserve system resources. The Internet Boot Protocol server is designed to provide network information to the client. This information can include, but is not restricted to, the client's IP address, netmask, broadcast address, domain server address, router address, etc.

THE ROUTE DAEMON (ROUTED)

The route daemon is invoked at boot time to manage the Internet Routing Tables. The routed daemon uses a variant of the Xerox NS Routing Information Protocol to maintain up-to-date kernel Routing Table entries. In normal operation, routed listens on the UDP socket 520 to provide the route service for routing information packets. If the host is an internetwork router, it periodically supplies copies of its routing tables to any directly connected hosts and networks.

The netstat command, which is discussed later in this chapter, is used to print the routing tables on a host. The netstat command is shown here:

```
$ netstat -r
Routing tables
Destination        Gateway             Flags  Refs  Use      Interface
nb.ottawa.uunet.   gateway             UH     0     1        du0
localhost.0.0.12   localhost.0.0.127.  UH     3     0        lo0
topgun             gateway             UH     1     3218     du1
default            gateway             UG     1     669360   du0
Lab.widgets.ca     gateway             U      8     3340413  wdn0
Ottawa.widgets.ca  gateway             U      10    2083505  iat0
$
```

The list identifies the gateway that is used to reach a specific destination network, along with the status of the route (flags). It also includes how many connections are in use through that gateway, the number of packets through the gateway, and the name of the interface in this machine that connects the machine to the network.

Most systems are capable of handling dynamic and static routes. The dynamic routes are handled by the routed daemon. As the routes change, the routed daemon updates the tables and informs other hosts as needed. The static routes generally are manipulated by hand using the route command, and generally are not controlled by the routed daemon.

Bear in mind that routed can handle only the RIP protocol; it cannot handle CIDR. For enhanced routing protocol handling, the gated daemon is a better choice.

THE DOMAIN NAME SERVER (NAMED)

named is the Internet Domain Name Server, and it is the second mechanism available to provide host name to IP address resolution. The daemon can serve in a variety of roles, including primary, secondary, caching, and as a slave, depending upon the requirements of the network administrator. If the /etc/hosts file is not used, and domain name service (DNS) is configured, then the system makes requests to the DNS to provide the IP address for a host name. If the local DNS does not know the IP address for the specified host, it queries other name servers until it obtains the address.

The user command nslookup is used to query the DNS server for a given piece of information. The following illustrates using the nslookup command to find the IP address for the host name gatekeeper.dec.com.

```
$ nslookup gatekeeper.dec.com
Server:  gateway.widgets.ca
Address:  192.139.234.50
Non-authoritative answer:
Name:    gatekeeper.dec.com
Address:  16.1.0.2

$
```

In this output, the domain name server gateway.widgets.ca cannot provide an authoritative response because it is not the authoritative master for the dec.com domain. The end result is that you learn the IP address for gatekeeper.dec.com is, in fact, 16.1.0.2.

Systems that are not running a DNS server themselves must use a resolv.conf file to indicate where DNS queries should be addressed. The format of the resolv.conf file is as follows:

```
domain widgets.ca

nameserver 192.168.1.1
```

This format indicates the domain in which the current machine resides, which is used when only a hostname is provided, and the IP address of the server to which to send the request. Multiple nameservers can be specified by adding the additional IP addresses separated by commas.

THE SYSTEM LOGGER (SYSLOGD)

This daemon is responsible for logging various system messages in a set of files described by the syslogd configuration file /etc/syslog.conf. Each message is saved on a single line in the file and can contain a wide variety of information. The syslog daemon receives information sent to it and saves the messages in its log file. Information can consist of informational, error, status, and debug messages. Each message also can have a level of severity associated with it.

INETD—THE SUPER-SERVER

The inetd super-server listens on multiple TCP/IP ports for incoming connection requests. When the request is received, it spawns the appropriate server. The use of a super-server allows other servers to spawn only when needed, thereby saving system resources. When the connection is terminated, the spawned server terminates.

Typically, servers that are started through inetd include fingerd, ftpd, rexecd, rlogind, and others. inetd, however, cannot be used for servers like named, routed, rwhod, sendmail, or any RFS or NFS server.

THE RWHO DAEMON (RWHOD)

The RWHO daemon maintains the database used by the rwho and ruptime commands. Its operation is predicated by its capability to broadcast messages on a network. rwho operates by periodically querying the state of the system and broadcasting that information on the network. It also listens for rwho messages produced by other systems, so it can update its database of remote server information.

It is important to note that this service takes up more and more bandwidth as the number of hosts grows. For large networks, the cost in network traffic becomes prohibitive.

EXPLORING TCP/IP UTILITIES

TCP/IP commands can be split into three categories: those that are used to administer the TCP/IP network at one level or another, user commands that can be considered applications unto themselves, and third-party applications that have been implemented by using one or more of the services provided by TCP/IP, such as client-server databases.

ADMINISTRATION COMMANDS

This section examines some of the commands that are used to administer the TCP/IP services provided in a system. Many of the commands can be executed by either a regular user or the super-user, but some features are restricted due to the nature of the command. An understanding of the commands available to administer TCP/IP, however, is important for the administrator and helpful for the user.

THE PING COMMAND

The ping command is used to send Internet Control Message Protocol (ICMP) packets from one host to another. ping transmits packets using the ICMP ECHO_REQUEST command and expects to get an ICMP ECHO_REPLY in response to each transmitted packet. The name ping comes from the sonar detection device that uses a sound pulse resembling a ping to locate targets in the surrounding area. In this case, the sound pulses are ICMP packets to a target host.

ICMP is a protocol suite of its own that is used to report errors encountered in processing packets and to perform other Internet layer functions, such as supplying the protocol for the ping command.

ICMPv6 messages are grouped into two classes: error messages and informational messages. ICMP messages are described in table 1.5.

TABLE 1.5

ICMP Messages

Message	*Type*	*Description*
`Destination Unreachable`	Error	Normally generated by a router or by the IP layer in the originating node, in response to a packet that cannot be delivered to its destination address for reasons other than congestion.
`Packet Too Big`	Error	Sent by a router in response to a packet that it cannot forward because the packet is larger than the MTU of the outgoing link.
`Time Exceeded`	Error	If a router receives a packet with a hop limit of zero, or a router decrements a packet's hop limit to zero, the router discards the packet and sends an ICMP `Time Exceeded` message with Code 0 to the source of the packet. This message indicates either a routing loop or too small an initial hop limit value.
`Parameter Problem`	Error	If a node processing a packet finds a problem with a field in the IP header or extension headers such that it cannot

continues

TABLE 1.5, CONTINUED
ICMP Messages

Message	*Type*	*Description*
		completely process the packet, it discards the packet and sends an ICMP `Parameter Problem` message to the packet's source, indicating the type and location of the problem.
`Echo Request`	Information	Every node implements an ICMP Echo responder function that receives Echo Requests and sends corresponding Echo Replies. A node normally also implements an Application layer interface for sending Echo Requests and receiving Echo Replies for diagnostic purposes.
`Echo Reply`	Information	The response generated when an Echo Request packet is transmitted.
`Group Membership Query`, `Group Membership Report`, `Group Membership Reduction`	Information	The ICMP Group Membership messages are used to convey information about multicast group membership from nodes to their neighboring routers.

The following illustrates using ping with a responding host and using ping with a nonresponding host. Under normal circumstances, ping does not terminate, but broadcasts packets until the user stops it, typically through an interrupt signal such as Ctrl+C.

```
$ ping shylock
PING shylock (192.139.234.12): 56 data bytes
64 bytes from shylock (192.139.234.12): icmp_seq=0 ttl=254 time=10 ms
64 bytes from shylock (192.139.234.12): icmp_seq=1 ttl=254 time=10 ms
64 bytes from shylock (192.139.234.12): icmp_seq=2 ttl=254 time=10 ms
64 bytes from shylock (192.139.234.12): icmp_seq=3 ttl=254 time=10 ms

-- shylock ping statistics --
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max = 10/10/10 ms
$
```

The ping command has a wide variety of options that can be used to help locate potential problems in the connections. These options and their explanation are shown in table 1.6.

TABLE 1.6
ping Command Options

Option	*Description*
-c count	This instructs ping to continue sending packets until count requests have been sent and received.
-d	This option turns on the debug option for the socket being used.
-f	This is a flood ping. It causes ping to output packets as fast as they come back from the remote host or 100 times per second, whichever is faster. In this mode, each request is shown with a period, and for each response, a backspace is printed. Only the super-user can use this option. For obvious reasons, this can be very hard on a network and should be used with caution.
-i seconds	This option instructs ping to wait the specified number of seconds between transmitting each packet. This option cannot be used with the -f option.
-n	Numeric mode only. Normally ping attempts to resolve the IP address for a host name. This option instructs ping to print the IP addresses and not look up the symbolic names. This is important if for some reason the local name server is not available.
-p pattern	This enables the user to specify up to 16 pad bytes to be added to the packet. This is useful for diagnosing data-dependent problems in a network. Using -p ff, for example, causes the transmitted packet to be filled with all 1s.
-q	Normally, ping reports each response received. This option puts ping into quiet mode. The result is that it prints the summary information at startup and completion of the command.
-R	This adds the ICMP RECORD_ROUTE option to the ECHO_REQUEST packet. This asks for the route to be recorded in the packet, which ping then prints when the packet is returned. There is only room for nine routes in each packet, and many hosts ignore or discard this option.
-r	This causes ping to bypass the normal routing tables that would be used to transmit a packet. For this to work, the host must be on a directly attached network. If the target host is not, an error is printed by ping.

continues

Table 1.6, Continued
ping Command Options

Option	*Description*
-s packetsize	This enables the user to specify the number of data bytes that are to be sent. The default is 56 bytes, which translates into 64 ICMP data bytes when combined with the 8 bytes of ICMP header of data.
-v	This puts ping into verbose mode. It instructs ping to print all ICMP packets returned other than ECHO_RESPONSE packets.

The following demonstrates the -q option for ping. With this example, ping prints only the startup and summary information.

```
$ ping -q ftp.widgets.ca
PING chelsea.widgets.ca (198.73.138.6): 56 data bytes

-- chelsea.widgets.ca ping statistics --
7 packets transmitted, 7 packets received, 0% packet loss
round-trip min/avg/max = 0/0/0 ms
$
```

These examples are not representative of all implementations of ping. The following illustrates the output of ping on BSD versions of Unix:

```
% /usr/etc/ping gateway
gateway.widgets.ca is alive
%
```

For the BSD Unix users, the ping command generally does not print the information that was illustrated in preceding code. The preceding example serves to illustrate that even different versions of TCP/IP have been implemented differently.

When ping is used for fault isolation, it should first be run on the local host to ensure that the network interface is up and running. The ping program is intended for use in network testing, measurement, and management. Because of the load it can impose on the network, however, it is not wise to use ping during normal working hours or from automated test scripts.

The ruptime and rwho Commands

The ruptime command uses the facilities of the rwhod server to show the status of the local machines on the network. It prints a status line for each host in its database. This database is built by using broadcast rwho packets, once every one to three minutes. Machines that have not reported a status for five minutes are reported as down. The output of ruptime is shown as follows:

```
$ ruptime
chelsea       up 17+01:28,     0 users,  load 0.00, 0.00, 0.00
daffy         up 29+06:11,     0 users,  load 1.00, 1.00, 1.00
gateway       up 16+14:34,     1 user,   load 1.00, 1.05, 1.02
mallow        up  5+12:46,     0 users,  load 0.00, 0.00, 0.00
oreo          up 19+13:13,     1 user,   load 2.00, 2.00, 1.33
ovide         up  4+04:54,     1 user,   load 1.14, 1.16, 1.17
wabbit      down 107+01:33
$
```

If the rwhod server is not running on any of the hosts in your network, then ruptime reports the status message `no hosts!!!` and exits. In the preceding example, the system wabbit appears to be down. This might be accurate, but it also might be that the rwhod server has exited or is no longer running on the system.

Normally, the ruptime output is sorted by host name, but the following options alter the output format of ruptime.

TABLE 1.7

ruptime Options

Option	*Description*
-a	Includes all users in ruptime output. (Users idle for more than an hour are not usually counted in the user list.)
-l	Sorts the output by load average.
-t	Sorts the output by uptime.
-u	Sorts the output by the number of users.
-r	Reverses the sort order.

The rwho command lists the users who currently are logged in on each of the servers in the network. The rwho command reports the users who are logged in on hosts that are responding to and transmitting rwhod packets. The output of the rwho command is shown here:

```
$ rwho
chare    oreo:ttyp0    Oct  9 14:58 :01
root     ovide:tty08   Oct  4 18:05
topgun   gateway:tty2A Oct  9 13:54
$
```

In this output, the name of the user is shown as well as the system name, the port he is logged in on, and the date he logged in to the system. This looks like the output from the who command, except the system name is included in the port information.

The ifconfig Command

The ifconfig command has been presented in some detail, but this section illustrates some additional uses for it. By using ifconfig, for example, it is possible to query the interface to find out how it has been configured, as shown here:

```
$ /etc/ifconfig wdn0
wdn0: flags=23<UP,BROADCAST,NOTRAILERS>
inet 198.73.138.2 netmask ffffff00 broadcast 198.73.138.255
$
```

This output shows that the interface is up, it does not use trailer encapsulation, and it identifies the addresses and netmask currently used by the interface. Any interface that is configured on the system can be queried in this manner.

The following illustrates marking an interface down, verifying that information, and then marking the interface up:

```
# ifconfig du0
du0: flags=51<UP,POINTOPOINT,RUNNING>
        inet 142.77.252.6 --> 142.77.17.1 netmask ffff0000
# ifconfig du0 down
# ifconfig du0
du0: flags=50<POINTOPOINT,RUNNING>
        inet 142.77.252.6 --> 142.77.17.1 netmask ffff0000
# ping toradm
PING toradm.widgets.ca (142.77.253.13): 56 data bytes
ping: sendto: Network is unreachable
ping: wrote toradm.widgets.ca 64 chars, ret=-1
ping: sendto: Network is unreachable
ping: wrote toradm.widgets.ca 64 chars, ret=-1
ping: sendto: Network is unreachable
ping: wrote toradm.widgets.ca 64 chars, ret=-1

-- toradm.widgets.ca ping statistics --
3 packets transmitted, 0 packets received, 100% packet loss
# ifconfig du0 up
# ifconfig du0
du0: flags=51<UP,POINTOPOINT,RUNNING>
inet 142.77.252.6 --> 142.77.17.1 netmask ffff0000
# ping toradm
PING toradm.widgets.ca (142.77.253.13): 56 data bytes
64 bytes from toradm.widgets.ca (142.77.253.13): icmp_seq=0 ttl=251 time=610
➥ms
64 bytes from toradm.widgets.ca (142.77.253.13): icmp_seq=1 ttl=251 time=630
➥ms

-- toradm.widgets.ca ping statistics --
3 packets transmitted, 2 packets received, 33% packet loss
round-trip min/avg/max = 610/620/630 ms
#
```

In this example, the interface being affected is a point-to-point protocol link, which is illustrated in the output of ifconfig. When the interface is marked down, packets will not be transmitted on that link, as shown using ping. When the interface is later marked up, traffic once again flows on that link.

The use of ifconfig to configure an interface is restricted to the super-user. Any user on the system, however, can use ifconfig to query the interface for its current operating statistics.

THE FINGER COMMAND

By default, finger lists the login name, full name, terminal name and terminal write status (as a "*" before the terminal name if write permission is denied), idle time, login time, office location, and phone number (if known) for each current user.

NOTE

> Idle time is minutes if it is a single integer, hours and minutes if a colon (:) is present, or days and hours if a "d" is present.

Longer format also exists and is used by finger whenever a list of names is given. (Account names as well as first and last names of users are accepted.) This is a multiline format; it includes all the information described earlier as well as the user's home directory, login shell, any plan the user has placed in the .plan file in her home directory, and the project on which she is working from the .project file that is also in her home directory. The output of finger is illustrated here:

```
$ finger chare
Login name: chare       (messages off)  In real life: Chris Hare
Directory: /u/chare                     Shell: /bin/ksh
On since Oct  8 22:06:31 on ttyp0
Project: Not assigned to one (yet).
Plan:
To complete the currently assigned tasks.
```

NOTE

> Many people do not want the output of finger to be generally available; therefore it is either turned off, or restricted versions are implemented. The issue is that finger can provide the cracker with information about the users on the system, such as their login names, and some ideas about what their passwords might be.

In the preceding code, the output from this finger command is for a user who is currently logged into the system. Notice the `(messages off)` text. This indicates that any attempts to contact this user with the write command will fail because the user does not allow writes to her terminal. When the user is not logged in, the output is different, as shown here:

```
$ finger andrewg
Login name: andrewg                     In real life: Andrew Goodier
Directory: /u/andrewg                   Shell: /bin/ksh
Last login Sun Sep 18 22:08
No Plan.
$
```

Table 1.8 lists the options that typically are available on the finger command.

TABLE 1.8
The Options for the finger Command

Option	*Description*
-b	Briefer output format
-f	Suppresses the printing of the header line (short format)
-i	Provides a quick list of users with idle times
-l	Forces long output format
-p	Suppresses printing of the .plan files
-q	Provides a quick list of users
-s	Forces short output format
-w	Forces narrow format list of specified users

It is important for you to recognize that the finger command allows the distribution of valuable user information, such as user names and home directories. For this reason, many sites choose to disable the finger daemon and remove the finger command entirely.

THE NETSTAT COMMAND

The netstat command is used to query the network subsystem regarding certain types of information. netstat, for example, can be used to print the routing tables, active connections, streams in use (on those systems that use streams), and more. netstat prints the information in a symbolic format that is easier for the user to understand. The options for netstat are listed in table 1.9.

Table 1.9

netstat Options

Option	*Description*
-A	Shows the addresses of any associated protocol control blocks. This option is primarily used for debugging only.
-a	Instructs netstat to show the status of all sockets. Normally, the sockets associated with server processes are not shown.
-i	Shows the state of the interfaces that have been autoconfigured. Those interfaces that have been configured after the initial boot of the system are not shown in the output.
-m	Prints the network memory usage.
-n	Causes netstat to print the actual addresses instead of interpreting them and displaying a symbol such as a host or network name.
-r	Prints the routing tables.
-f address-family	Causes netstat to print only the statistics and control block information for the named address family. Currently, the only address family supported is inet.
-I interface	Shows the interface state for only the named interface.
-p protocol-name	Limits the statistics and protocol control block information to the named protocol.
-s	Causes netstat to show the per protocol statistics.
-t	Replaces the queue length information with timer information in the output displays.

The output from netstat in the following code illustrates the retrieval of interface statistics from the interfaces on the system:

```
$ netstat -i
Name Mtu  Net/Dest     Address      Ipkts   Ierrs Opkts   Oerrs Collis Queue
le0  1500 198.73.138.0 chelsea      2608027 26    1421823 1     2632   0
lo0  1536 loopback     127.0.0.1    765364  0     765364  0     0      0
$ netstat -in
Name Mtu  Net/Dest     Address      Ipkts   Ierrs Opkts   Oerrs Collis Queue
le0  1500 198.73.138.0 198.73.138.6 2608082 26    1421862 1     2632   0
lo0  1536 127.0.0.0    127.0.0.1    765364  0     765364  0     0      0
$
```

In the second invocation of netstat in the preceding code, the use of the -n option is employed. This causes netstat to print the address instead of the symbolic name that was printed in the first invocation of netstat. This information is dependent upon the link level driver for the interface. If that driver does not attach itself to the ifstats structure in the kernel, then the phrase `No Statistics Available` is printed.

In the output of netstat shown in the preceding example, columns of information are shown. These columns and their meanings are listed in table 1.10.

Table 1.10

netstat Column Headings

Column	*Description*
Name	The name of the configured interface
Mtu	The maximum transmission unit for the interface
Net/Dest	The network that this interface serves
Address	The IP Address of the interface
Ipkts	The number of received packets
Ierrs	The number of packets that have been mangled when received
Opkts	The number of transmitted packets
Oerrs	The number of packets that were damaged when transmitted
Collisions	The number of collisions recorded by this interface on the network

Keep in mind that the notion of errors is somewhat ill-defined according to many of the manual pages for netstat, calling into question the validity of the values in the error columns. In addition, with the tables always being updated, the information presented is, like the output of ps, only a snapshot of the status at any given interval.

One of the common uses of netstat is to find out if there are any network memory allocation problems. This is achieved using the command netstat -m, as shown here:

```
$ netstat -m
streams allocation:
config   alloc    free   total     max    fail
streams                 292      93     199   53882     112       0
queues                 1424     452     972  122783     552       0
mblks                  5067     279    478820190677     706       0
dblks                  4054     279    377515804030     706       0
class 0,    4 bytes     652      55     597  475300     277       0
class 1,   16 bytes     652       8     644 2404108      62       0
class 2,   64 bytes     768      22     746 9964817     232       0
class 3,  128 bytes     872     138     734 1223784     386       0
class 4,  256 bytes     548      34     514  230688      75       0
class 5,  512 bytes     324      12     312   92565      76       0
class 6, 1024 bytes     107       0     107 1226009      49       0
class 7, 2048 bytes      90       0      90  182978      67       0
class 8, 4096 bytes      41      10      31    3781      13       0
total configured streams memory: 1166.73KB
streams memory in use: 98.44KB
maximum streams memory used: 409.22KB
$
```

This output is from an SCO Unix 3.2 version 4.2 system. If there are any non-zero values in the fail column, then it is important to readjust the number configured. When the configured number of data blocks is reached, a failure is generated. This means that a TCP/IP application or service could not get the needed resources. The only way to correct this problem in the short term is to reboot the machine. Over the long run, the only way to prevent these failures is to adjust the values and relink the kernel. The output of netstat -m on a SunOS system is similar in content to the SCO systems.

The netstat command also can be used to list all the sockets that are on the system using the -a option. This option is illustrated here:

```
$ netstat -a
Active Internet connections (including servers)
Proto Recv-Q Send-Q  Local Address       Foreign Address        (state)
ip          0      0  *.*                 *.*
tcp         0  28672  oreo.20             topgun.4450            ESTABLISHED
tcp         0    286  oreo.telnet         topgun.4449            ESTABLISHED
tcp         0      0  oreo.ftp            topgun.4438            ESTABLISHED
tcp         0      0  oreo.1725           gateway.telnet         ESTABLISHED
tcp         0      0  *.printer           *.*                    LISTEN
tcp         0      0  *.pop               *.*                    LISTEN
tcp         0      0  *.smtp              *.*                    LISTEN
tcp         0      0  *.finger            *.*                    LISTEN
tcp         0      0  *.exec              *.*                    LISTEN
tcp         0      0  *.login             *.*                    LISTEN
tcp         0      0  *.shell             *.*                    LISTEN
tcp         0      0  *.telnet            *.*                    LISTEN
tcp         0      0  *.ftp               *.*                    LISTEN
udp         0      0  *.snmp              *.*
udp         0      0  *.who               *.*
$
```

This output shows the status of the currently connected sockets and to what they are connected. For the TCP sockets, the status of the socket is reported in the output. The state is one of the following listed in table 1.11.

TABLE 1.11

TCP Socket Explanations

State	*Meaning*
CLOSED	The socket is not being used.
LISTEN	The socket is listening for an incoming connection.
SYN_SENT	The socket is actively trying to establish a connection.
SYN_RECEIVED	The initial synchronization of the connection is underway.
ESTABLISHED	The connection has been established.
CLOSE_WAIT	The remote has shut down: we are waiting for the socket to close.
FIN_WAIT_1	The socket is closed, and the connection is being shut down.
CLOSING	The socket is closed, and the remote is being shut down. The acknowledgment of the close is pending.
LAST_ACK	The remote has shut down and closed. They are waiting for us to acknowledge the close.
FIN_WAIT_2	The socket is closed, and we are waiting for the remote to shut down.
TIME_WAIT	The socket is waiting after the close for the remote shutdown transmission.

With this information, it is easy to tell what state the connection is in and how to trace the connection through the various stages of operation.

THE TRACEROUTE COMMAND

The traceroute command is used to trace the route that a packet must take to reach the destination machine. This command works by utilizing the time-to-live (TTL) field in the IP packet to elicit an ICMP TIME_EXCEEDED response from each gateway along the path to the remote host. The following code uses the traceroute command.

```
# traceroute toradm.widgets.ca
traceroute to toradm.widgets.ca (142.77.253.13), 30 hops max, 40 byte packets
 1  gateway (198.73.138.50)  10 ms  10 ms  10 ms
 2  nb.ottawa.uunet.ca (142.77.17.1)  260 ms  300 ms  270 ms
 3  gw.ottawa.uunet.ca (142.77.16.3)  240 ms  240 ms  270 ms
 4  wf.toronto.uunet.ca (142.77.59.1)  280 ms  260 ms  310 ms
 5  alternet-gw.toronto.uunet.ca (142.77.1.202)  250 ms  260 ms  250 ms
 6  nb1.toronto.uunet.ca (142.77.1.201)  260 ms  250 ms  260 ms
 7  toradm (142.77.253.13)  880 ms  720 ms  490 ms
#
```

As in the preceding example, the traceroute command attempts to trace the route that an IP packet would follow to some Internet host. The command works by sending probes until the maximum number of probes has been sent, or the remote responds with a `DESTINATION UNREACHABLE` message. The `DESTINATION UNREACHABLE` message is sent if the hop count to the remote site becomes zero, which implies that the TTL of the packet has been exceeded.

In the output of the traceroute command in the preceding example, the times following the host name are the round trip times for the probe. From this output, you can see that for a packet to travel from the originating host (oreo.widgets.ca), it must travel through seven hosts to reach the destination system, toradm.widgets.ca. The following illustrates another invocation of traceroute.

```
# traceroute gatekeeper.dec.com
traceroute to gatekeeper.dec.com (16.1.0.2), 30 hops max, 40 byte packets
 1  gateway (198.73.138.50)  10 ms  10 ms  10 ms
 2  nb.ottawa.uunet.ca (142.77.17.1)  250 ms  240 ms  240 ms
 3  gw.ottawa.uunet.ca (142.77.16.3)  270 ms  220 ms  240 ms
 4  wf.toronto.uunet.ca (142.77.59.1)  260 ms  270 ms  250 ms
 5  alternet-gw.toronto.uunet.ca (142.77.1.202)  250 ms  260 ms  260 ms
 6  Falls-Church1.VA.ALTER.NET (137.39.7.1)  470 ms  960 ms  810 ms
 7  Falls-Church4.VA.ALTER.NET (137.39.8.1)  760 ms  750 ms  830 ms
 8  Boone1.VA.ALTER.NET (137.39.43.66)  910 ms  810 ms  760 ms
 9  San-Jose3.CA.ALTER.NET (137.39.128.10)  930 ms  870 ms  850 ms
 10  * * Palo-Alto1.CA.ALTER.NET (137.39.101.130)  930 ms
 11  gatekeeper.dec.com (16.1.0.2)  830 ms  910 ms  830 ms
#
```

In this case, hop 10 did not report right away, but rather printed two asterisks before printing the gateway name and the round trip time. When traceroute does not receive a response within three seconds, it prints an asterisk. If no response from the gateway is received, then three asterisks are printed.

> Because of the apparent network load that traceroute can create, it should only be used for manual fault isolation or troubleshooting. This command should not be executed from cron or from within any automated test scripts.

NOTE

THE ARP COMMAND

The arp command displays and modifies the Internet-to-Ethernet address translation table, which normally is maintained by the address resolution protocol (ARP). When a host name is the only argument, arp displays the current ARP entry for that host. If the host is not in the current ARP table, then arp displays a message to that effect. The following illustrates using arp to find the Ethernet address for a specific host.

```
$ arp gateway
gateway (198.73.138.50) at 0:0:c0:11:57:4c
$ arp ovide
ovide (198.73.138.101) -- no entry
```

This illustrates the behavior of arp when no arguments are present. arp behaves a little differently, however, when options are combined. The available options for arp are defined in table 1.12.

TABLE 1.12
arp Command-Line Options

Option	*Description*
-a	Lists all the entries on the current ARP table.
-d host	Deletes the corresponding entry for host from the ARP table.
-s host address [temp] [pub] [trail]	Creates an entry in the ARP table for the named host, using an Ethernet address. If the keyword [temp] is included, the entry is temporary. Otherwise, the entry is permanent. The [pub] keyword indicates that the ARP entry will be published. Use of the [trail] keyword implies that trailer encapsulation is to be used.
-f file	Instructs arp to read the named file and create ARP table entries for each of the named hosts in the file.

The most commonly used option with arp is -a, which prints the entire ARP table, and is illustrated here:

```
$ arp -a
ovide.widgets.ca (198.73.138.101) at 0:0:c0:c6:4f:71
gateway.widgets.ca (198.73.138.50) at 0:0:c0:11:57:4c
chelsea.widgets.ca (198.73.138.6) at 8:0:20:2:94:bf
fremen.widgets.ca (198.73.138.54) at 0:0:3b:80:2:e5$
```

ARP is most commonly used to help debug and diagnose network connection problems. arp can help in that regard by assigning the Ethernet address for a given host. This is done by using the -s option, as shown here:

```
$ arp gateway
gateway (198.73.138.50) at 0:0:c0:11:57:4c
# arp -s ovide 0:0:c0:c6:4f:71
# arp -a
ovide.widgets.ca (198.73.138.101) at 0:0:c0:c6:4f:71 permanent
gateway.widgets.ca (198.73.138.50) at 0:0:c0:11:57:4c
#
```

This example illustrates adding an entry to the arp table. If you could not communicate with the remote host before the arp table entry was created, then you might have an addressing problem. If you still cannot communicate with the remote host after establishing the arp entry, then the problem is more likely to be hardware.

THE DIG COMMAND

The Domain Information Groper, dig, is a flexible command line tool that can be used to gather information from the Domain Name System servers. The dig tool can operate in simple interactive mode, where it satisfies a single query, and a batch mode, in which multiple requests are satisfied.

The dig tool requires a slightly modified version of the BIND resolver library to gather count and time statistics. Otherwise, it is a straightforward effort of parsing arguments and setting appropriate parameters. The output of dig can be rather convoluted, as shown here:

```
# dig gatekeeper.dec.com
; <<>> DiG 2.0 <<>> gatekeeper.dec.com
;; ->>HEADER<<- opcode: QUERY , status: NOERROR, id: 6
;; flags: qr rd ra ; Ques: 1, Ans: 1, Auth: 2, Addit: 2
;; QUESTIONS:
;;      gatekeeper.dec.com, type = A, class = IN

;; ANSWERS:
gatekeeper.dec.com.    150369  A       16.1.0.2

;; AUTHORITY RECORDS:
DEC.com.       166848  NS      GATEKEEPER.DEC.COM.
DEC.com.       166848  NS      CRL.DEC.COM.

;; ADDITIONAL RECORDS:
GATEKEEPER.DEC.COM.    150369  A       16.1.0.2
CRL.DEC.COM.   166848  A       192.58.206.2

;; Sent 1 pkts, answer found in time: 400 msec
;; FROM: oreo.widgets.ca to SERVER: default -- 192.139.234.50
;; WHEN: Mon Oct 10 15:07:41 1994
;; MSG SIZE  sent: 36  rcvd: 141

#
```

In the output shown here, the dig command searches the Domain Name Server records looking for gatekeeper.dec.com. A DNS record is found and reported to the user. Consequently, the dig command can be used to help resolve difficult name server problems. When there is some suspicion about how the resolution is being performed when done from nslookup, dig can be used to examine the problem. If, for example, nslookup is pulling the answer from the nameserver cache, it will not be evident from nslookup itself. dig, however, would expose the source of the information.

USER COMMANDS

Just as there are a number of commands to assist the system administrator in the management of the system and network, there are a number of commands that are used by the users to get the information they want and to perform the tasks they need. While a user can execute some of the administration commands you have seen, the real work is done with commands you are about to examine: telnet, ftp, and the Berkeley r-commands.

THE BERKELEY R-COMMANDS

The first of the commands examined here fall into the set called the Berkeley r-commands. These are the rlogin, rcp, and rsh/rcmd commands. They are called the Berkeley r-commands because they all start with the letter r, and they originated from the University of California at Berkeley. The successful use of the command in this section is dependent upon user and host equivalency being properly configured. Most users have difficulty with these commands because their network administrators have not properly configured the host and user equivalency.

NOTE

Host and user equivalency is accomplished with the /etc/hosts.equiv and .rhosts files, which were explained in the section "Understanding the Network Access Files," earlier in this chapter.

rlogin

The rlogin command connects your local session to a remote session on a different host. To initiate a remote terminal session, use the following command:

```
rlogin remote
```

This command starts a connection to the rlogind server on the remote host, as illustrated here:

```
$ rlogin gateway
Last   successful login for chare: Sun Oct 09 16:16:03 EDT 1994 on ttyp1
Last unsuccessful login for chare: Tue Sep 27 07:18:54 EDT 1994 on ttyp0
SCO UNIX System V/386 Release 3.2
```

```
Copyright (C) 1976-1989 UNIX System Laboratories, Inc.
Copyright (C) 1980-1989 Microsoft Corporation
Copyright (C) 1983-1992 The Santa Cruz Operation, Inc.
All Rights Reserved
gateway

Terminal type is dialup
$
```

The terminal type of the remote connection is the same as the terminal type that is in use for the current connections, unless modified by the user’s shell startup files. All of the character echoing is done at the remote site, so except for delays, the use of the rlogin is transparent to the user. Termination of the connection is made either by logging out of the remote host, or through the termination character, which is ~. (tilde period).

rcp

The rcp, or remote copy, command enables the user to copy a file from one host to another. rcp copies files between two machines. Each file or directory argument is either a remote file name of the form “rhost:path”, or a local file name (containing no ‘:’ characters, or a ‘/’ before any ‘:’).

The syntax of the command is as follows:

```
rcp [ -p ] file1 file2
    rcp [ -p ] [ -r ] file ... directory
```

The remote file must be specified using the following syntax:

```
hostname:filename
```

The named file is copied to or from the remote system depending upon whether the source or destination file is remote. The following illustrates copying a file from the local host to the remote.

```
$ rcp test.new chelsea:test.new
$
```

When the filename, as illustrated in the preceding example, does not begin with a slash (/), the file is copied in a directory relative to your home directory on the remote system. The rcp command behaves like the cp command in that the file could be called by a different name on the remote system.

If the -r option is specified and any of the source files are directories, rcp copies each subtree rooted at that name; in this case, the destination must be a directory. By default, the mode and owner of file2 are preserved if the file already existed; otherwise, the mode of the source file modified by the umask on the destination host is used.

The -p option causes rcp to attempt to preserve (duplicate) in its copies the modification times and modes of the source files, ignoring the umask. The following illustrates using rcp with the -r option to copy a directory tree.

```
$ pwd
/u/chare
$ lc tmp
arp.ADMN     bootpd       dig.new      route.new
arp.new      bootpd.ADMN  rarpd.new    routed.new
$rcp -r chelsea:/tmp tmp
$ lf tmp
arp.ADMN      bootpd        dig.new       route.new     test.new
arp.new       bootpd.ADMN   rarpd.new     routed.new    tmp/
$
```

After executing the rcp command in the preceding example, a new directory is created called tmp in /u/chare/tmp. This directory contains the contents of the /tmp directory on host chelsea.

rsh, remsh, and rcmd

These three commands all perform a similar function, which is to execute a command on a remote system. Interactive commands are not good candidates for this type of execution.

The rsh implementation of remote execution is not to be confused with the restricted shell (rsh) that exists on System V Unix systems. Likewise, some System V Unixes use remsh instead of rsh also. Typically, the systems that use rsh for remote execution are BSD-based Unix systems. rsh works by connecting to the specified host name and executing the specified command. rsh copies its standard input to the remote command, the standard output of the remote command to its standard output, and the standard error of the remote command to its standard error. Interrupt, quit, and terminate signals are propagated to the remote command; rsh normally terminates when the remote command does.

The command syntax of rsh is as follows:

```
rsh [ -l username ] [ -n ] hostname [ command ]
rsh hostname [ -l username ] [ -n ] [ command ]
```

The execution of a command involves entering the name of the host where the command is to be executed and the name of the command. Running rsh with no command argument has the effect of logging you into the remote system by using rlogin. The following example illustrates using rsh to execute commands:

```
% rsh oreo date
Mon Oct 10 17:23:43 EDT 1994
% rsh oreo hostname
oreo.widgets.ca
%
```

There are only two options to rsh, as shown in table 1.13.

TABLE 1.13
rsh Command-Line Options

Option	*Description*
-l username	Use username as the remote username instead of your local username. In the absence of this option, the remote username is the same as your local username.
-n	Redirect the input of rsh to /dev/null. You sometimes need this option to avoid unfortunate interactions between rsh and the shell that invokes it. If, for example, you are running rsh and start an rsh in the background without redirecting its input away from the terminal, it will block even if no reads are posted by the remote command. The -n option prevents this.

Virtually any command on the remote system can be executed. Commands that rely upon terminal characteristics or a level of user interaction, however, are not good candidates for the use of rsh.

The rcmd command is virtually identical to the rsh except that it typically is found on System V systems. Actually, the rcmd has the same options and operates the same fashion as the rsh command under BSD Unix. The following illustrates rcmd accessing a remote system by not specifying a command when starting rcmd.

```
$ rcmd chelsea
Last login: Mon Oct 10 17:18:10 from oreo.widgets.ca
SunOS Release 4.1 (GENERIC) #1: Wed Mar 7 10:59:35 PST 1990
%
```

The use of rsh/rcmd can be of value when you want to run a command on the remote system without having to log into that system. Some system administrators use it to see what processes are running on a remote system, as shown here:

```
$ rcmd gateway ps -ef ¦ more
     UID   PID  PPID  C    STIME  TTY    TIME COMMAND
    root     0     0  0  Sep 22   ?      0:00 sched
    root     1     0  0  Sep 22   ?     23:36 /etc/init -a
    root     2     0  0  Sep 22   ?      0:00 vhand
    root   221     1  0  Sep 22   ?      0:00 strerr
    root   150     1  0  Sep 22   ?      7:51 /etc/cron
    root   212     1  0  Sep 22   ?      0:35 cpd
    root   156     1  0  Sep 22   ?      3:21 /usr/lib/lpsched
    root   214     1  0  Sep 22   ?      0:00 slink
    root   317   315  0  Sep 22   ?      0:00 nfsd 4
    root   256     1  0  Sep 22   ?      0:00 /usr/lib/lpd start
  topgun  5740     1  0  10:30:51 2A     0:19 /etc/pppd 198.73.137.101: log /u
sr/lib/ppp/ppp-users/topgun/log debug 2 nolqm
    root   306     1  0  Sep 22   ?      0:00 pcnfsd
    root 17008   234  0  Sep 29   ?      0:07 telnetd
    root 17009 17008  0  Sep 29   p0     0:02 -sh
    root   286     1  0  Sep 22   ?      0:05 snmpd
$
```

Terminal Emulation Using telnet

The rlogin command allows for a connection from one system to another. rlogin, however, requires the user to have an account on the remote machine and host equivalency to have been configured. telnet, on the other hand, does not need either of those things.

The telnet command uses the TELNET protocol to establish a connection from the client to a telnetd server on the remote system. Unlike rlogin, telnet has a host mode where it is connected to the remote system, and command mode where the user can enter commands and interact with the TELNET protocol to change how the connection is handled.

To create a telnet connection, the user enters the telnet command, with or without a host name. When telnet is started with a host name, a connection to the remote host is established. After the connection is established, the user must then provide a login name and password to access the remote system. This is illustrated in the following:

```
$ telnet chelsea
Trying 198.73.138.6...
Connected to chelsea.widgets.ca.
Escape character is '^]'.


SunOS Unix(chelsea.widgets.ca)

login: chare
Password:
Last login: Mon Oct 10 17:33:35 from oreo.widgets.ca
SunOS Release 4.1 (GENERIC) #1: Wed Mar 7 10:59:35 PST 1990
%
```

Command mode is entered either by starting telnet with no arguments, or by entering Ctrl+], which is the telnet 'escape' key. This control key instructs telnet to enter command mode, as shown here:

```
chelsea.widgets.ca%
telnet> ?
Commands may be abbreviated. Commands are:

close           close current connection
```

```
logout          forcibly logout remote user and close the connection
display         display operating parameters
mode            try to enter line or character mode ('mode ?' for more)
open            connect to a site
quit            exit telnet
send            transmit special characters ('send ?' for more)
set             set operating parameters ('set ?' for more)
unset           unset operating parameters ('unset ?' for more)
status          print status information
toggle          toggle operating parameters ('toggle ?' for more)
slc             change state of special characters ('slc ?' for more)
z               suspend telnet
!               invoke a subshell
environ         change environment variables ('environ ?' for more)
?               print help information
telnet>
```

In the preceding example, the switch to command mode is performed and is indicated by the telnet> prompt. Once in command mode, there are a number of commands that can be used to alter or reconfigure the current session. The actual number of commands available in command mode is far too numerous to be discussed here.

telnet, however, has another useful feature. It is to allow the connection to a specific TCP port on a system, which may or may not be remote. The following example illustrates a connection to the SMTP port, port 25, on the local system.

```
$ telnet localhost 25
Trying 127.0.0.1...
Connected to localhost.widgets.ca.
Escape character is '^]'.
220 oreo.widgets.ca Server SMTP (Complaints/bugs to:  postmaster)
helo
250 oreo.widgets.ca - you are a charlatan
help
214-The following commands are accepted:
214-helo noop mail data rcpt help quit rset expn vrfy
214-
214 Send complaints/bugs to:  postmaster
quit
221 oreo.widgets.ca says goodbye to localhost.0.0.127.in-addr.arpa at Mon Oct
➥10
20:35:24.
Connection closed by foreign host.
$
```

While this is a useful feature to have when debugging connection problems, it also enables a user to forge e-mail by giving it directly to the SMTP or sendmail daemon. Actually, most TCP/IP daemons can be connected to by using telnet with the port number, which might allow for other security mechanisms to be breached, particularly with sendmail.

FILE TRANSFERS WITH FTP

FTP is the ARPANET File Transfer Program that uses the File Transfer Protocol to allow for the verified transfer of a file from one PC to another. To reduce the chance of confusion, ftp usually refers to the program, while FTP refers to the protocol that is used to transfer the files.

The client host that ftp is to communicate with is normally provided on the command line. If so, ftp will immediately try to connect with the ftp server on that system. If a connection is established, then the user must log in to access the system. Logging in can be achieved either by having a valid account on the system, or through accessing a server that allows anonymous ftp access. Accessing an ftp server through anonymous mode is illustrated in the following:

```
$ ftp ftp.widgets.ca
Connected to chelsea.widgets.ca.
220 chelsea.widgets.ca FTP server (SunOS 4.1) ready.
Name (ftp.widgets.ca:chare): anonymous
331 Guest login ok, send ident as password.
Password:
230 Guest login ok, access restrictions apply.
ftp> quit
221 Goodbye.
$
```

NOTE

> When configuring a server for anonymous ftp access, be sure to create the file /etc/ftpusers. This file contains a list of user names, one per line, who are not allowed to access the ftp server. On any ftp server that supports anonymous ftp, access to the server as the root user should not be permitted.

By not restricting access through certain accounts, anyone, once one machine is compromised, can gain access to the anonymous ftp server and complete the transaction shown in the following example:

```
$ ftp ftp.widgets.ca
Connected to chelsea.widgets.ca.
220 chelsea.widgets.ca FTP server (SunOS 4.1) ready.
Name (ftp.widgets.ca:chare): root
331 Password required for root.
Password:
230 User root logged in.
ftp> cd /etc
250 CWD command successful.
ftp> lcd /tmp
Local directory now /tmp
```

```
ftp> get passwd passwd.ccca
local: passwd.ccca remote: passwd
200 PORT command successful.
150 ASCII data connection for passwd (198.73.138.2,1138) (736 bytes).
226 ASCII Transfer complete.
753 bytes received in 0.01 seconds (74 Kbytes/s)
ftp> quit
221 Goodbye.
$
```

The user who made this connection now has your password file. This type of connection can be prevented by creating the /etc/ftpusers file, as shown in the following:

```
# cd /etc
# s -l ftpusers
-rw-r--r--  1 root          10 Oct 10 20:53 ftpusers
# cat ftpusers
root
uucp
#
```

Now when a user tries to access the system by using the root account, he does not get the chance to enter a root password because ftp informs him that root access through ftp is not allowed, as shown in the following:

```
$ ftp ftp.widgets.ca
Connected to chelsea.widgets.ca.
220 chelsea.widgets.ca FTP server (SunOS 4.1) ready.
Name (ftp.widgets.ca:chare): root
530 User root access denied.
Login failed.
ftp> quit
221 Goodbye.
$
```

Another problem with ftp is the .netrc file that enables users to automate a file transfer. The reason this file is a problem is because users can insert login and password information in the file. The ftp client aborts the use of the file if it finds that it is readable by anyone other than the owner, but even that is not enough as the file can still leave security holes wide open.

The .netrc file resides in the user's home directory and can contain information for accessing more than one system. Consider the sample .netrc file shown here:

```
$ cat .netrc
machine yosemite.widgets.ca login chare password yippee
default login anonymous password chare@widgets.ca
```

The file format of the .netrc file is to include the information for each machine on a single line. The first entry of this file, for example, shows the connection to a machine called yosemite.widgets.ca. When this machine name is provided as an argument to ftp, the .netrc file is checked, and the login information here is used to access the system. The second entry is used as the default. If the system is not found explicitly, then use the anonymous entry to allow for anonymous access to the ftp site.

As mentioned, the ftp command does perform a security check on the .netrc. If the file is readable by anyone other than the owner, the connection is not established. This is illustrated in the following:

```
$ ftp yosemite.widgets.ca
Connected to yosemite.widgets.ca.
220 yosemite.widgets.ca FTP server (Version 5.60 #1) ready.
Error - .netrc file not correct mode.
Remove password or correct mode.
Remote system type is Unix.
Using binary mode to transfer files.
ftp> quit
221 Goodbye.
$ ls -l .netrc
-rw-r--r--   1 chare    group        103 Oct 10 21:16 .netrc
$
```

In the preceding example, the connection to yosemite is not made because the permissions on the .netrc file are incorrect. After the permissions are changed, the connection can be established without incident, as shown in the following:

```
$ ls -l .netrc
-rw-r--r--   1 chare    group        103 Oct 10 21:16 .netrc
$ chmod 600 .netrc
$ ls -l .netrc
-rw------   1 chare    group        103 Oct 10 21:16 .netrc
149$ ftp gateway.widgets.ca
Connected to gateway.widgets.ca.
220 gateway.widgets.ca FTP server (Version 5.60 #1) ready.
331 Password required for chare.
230 User chare logged in.
Remote system type is Unix.
Using binary mode to transfer files.
ftp>
```

NOTE

It's a good idea to teach users who want to use the .netrc file about security. By improperly setting the permissions on the file, users can prevent themselves from accessing the remote machine using the auto-login features, but can still allow someone else access by giving that person their login name and password.

SUMMARY

This chapter has introduced TCP/IP, addressing, the TCP/IP daemons, and user level commands. It has examined some of the commands that are available to both the user and the system administrator, and outlined how TCP/IP can be used to access and manipulate information.

Like anything else, the ability to utilize information presented here comes from experimenting and examining your own system. Keep in mind that most System V systems use STREAMS as their transport layer, while BSD systems use SOCKETS. TCP/IP provides a wide array of protocols and services to allow users to transport data from one place to another.

CHAPTER 2

SECURITY

UNLESS THE COMPUTER YOU are trying to protect is in a locked room where access is controlled and no connections to this computer from outside the room exist, then your computer is at risk. Break-ins and security violations occur almost daily around the world. These violators are not just Internet vandals, they include the employee down the hall who steals computer time or services for his own personal or malicious use.

This chapter examines computer security in some detail. You learn what computer security is, how you can protect yourself, and what is available to help you perform these security tasks. Because Unix is the predominant operating system on the Internet, this chapter focuses on Unix. This does not mean, however, that other operating systems do not have their security problems. Regardless of the manufacturer of your hardware and operating system, you must have a thorough understanding of the risks in your situation.

EXAMINING SECURITY LEVELS

According to the computer security standards developed by the United States Department of Defense, the Trusted Computer Standards Evaluation Criteria—the Orange Book, several levels of security are used to protect the hardware, software, and stored information from attack. These levels all describe different types of physical security, user authentication, trustedness of the operating system software, and user applications. These standards also impose limits on what types of other systems can be connected to your system.

NOTE

> The Orange Book has remained unchanged since it became a Department of Defense standard in 1985. It has been the primary method of evaluating the security of multi-user mainframe and mini operating systems for many years. Other subsystems, such as databases and networks, have been evaluated through interpretations of the Orange Book, such as the Trusted Database Interpretation and the Trusted Network Interpretation.

LEVEL D1

Level D1 is the lowest form of security available. This standard states that the entire system is untrusted. No protection is available for the hardware; the operating system is easily compromised; and there is no authentication regarding users and their rights to access information stored on the computer. This level of security typically refers to operating systems like MS-DOS, MS-Windows, and the Apple Macintosh System 7.*x*.

These operating systems do not distinguish between users and have no defined method of determining who is typing at the keyboard. These operating systems also do not have any controls regarding what information can be accessed on the hard drives in the computer.

LEVEL C1

Level C has two sublevels of security—C1 and C2. Level C1, or the Discretionary Security Protection system, describes the security available on a typical Unix system. Some level of protection exists for the hardware because it cannot be as easily compromised, although it is still possible. Users must identify themselves to the system through a user login name and password. This combination is used to determine what access rights to programs and information each user has.

These access rights are the file and directory permissions. These Discretionary Access Controls enable the owner of the file or directory, or the system administrator, to prevent certain people, or groups of people, from accessing those programs or information. However, the system administration account is not prevented from performing any activity. Consequently, an unscrupulous system administrator can easily compromise the security of the system without anyone's knowledge.

In addition, many of the day-to-day system administration tasks only can be performed by the user login known as *root*. With the decentralization of computer systems today, it is not uncommon to enter an organization and find more than two or three people who know the root password. This itself is a problem because no way exists to distinguish between the changes Doug or Mary made to the system yesterday.

LEVEL C2

The second sub-level, C2, was meant to help address these issues. Along with the features of C1, level C2 includes additional security features that create a controlled-access environment. This environment has the capability to further restrict users from executing certain commands or accessing certain files based not only upon the permissions, but upon authorization levels. In addition, this level of security requires that the system be audited. This involves writing an audit record for each event that occurs on the system.

Auditing is used to keep records of all security-related events, such as those activities performed by the system administrator. Auditing requires additional authentication because without it, how can you be sure that the person who executes the command really is that person. The disadvantage to auditing is that it requires additional processor and disk subsystem resources.

With the use of additional authorizations, it is possible for users on a C2 system to have the authority to perform system management tasks without having the root password. This enables improved tracking of system administration-related tasks because the individual user performs the work and not the system administrator.

These additional authorizations are not to be confused with the SGID and SUID permissions that can be applied to a program. Rather, they are specific authorizations that allow a user to execute specific commands or access certain kernel tables. Users who do not have the proper authority to view the process table, for example, see only their processes when they execute the ps command.

LEVEL B1

B-level security contains three levels. The B1 level, or Labeled Security Protection, is the first level that supports multilevel security, such as secret and top secret. This level states that an object under mandatory access control cannot have its permissions changed by the owner of the file.

LEVEL B2

The B2 level, known as Structured Protection, requires that every object be labeled. Devices such as disks, tapes, or terminals might have a single or multiple level of security assigned to them. This is the first level that starts to address the problem of an object at a higher level of security communicating with another object at a lower level of security.

LEVEL B3

The B3, or Security Domains level, enforces the domain with the installation of hardware. For example, memory management hardware is used to protect the security domain from unauthorized access or modification from objects in different security domains. This level also requires that the user's terminal be connected to the system through a trusted path.

LEVEL A

Level A, or the Verified Design level, is currently the highest level of security validated through the Orange Book. It includes a stringent design, control, and verification process. For this level of security to be achieved, all the components of the lower levels must be included; the design must be mathematically verified; and an analysis of covert channels and trusted distribution must be performed. *Trusted distribution* means that the hardware and software have been protected during shipment to prevent tampering with the security systems.

CANADIAN SECURITY

The Canadian government has undertaken its own design of trusted computing standards. These standards consist of two components: The Canadian Trusted Computer Product Evaluation Criteria (CTCPEC) and the Common Criteria.

The CTCPEC addresses both functionality and assurance of the product under development or evaluation. Functionality addresses the areas of confidentiality, integrity, availability, and accountability; and assurance focuses on the degree of confidence with which a product implements the organization's security policy.

The Common Criteria is a collaborative effort to pool the investigations and research of the United States, Canada, France, Germany, and the United Kingdom. This document contains the requirements for the selection of appropriate IT security measures.

There are seven levels of assurance within the Common Criteria: EAL-1 to EAL-7. These criteria are defined in the following sections.

Level EAL-1

EAL-1 is the lowest level of assurance that is meaningful to the developer and the consumer. It defines the minimum level of assurance and is based on an analysis of the security functions of the product using the functional and interface designs, as presented by the developer, to understand the security behavior.

Level EAL-2

EAL-2 is the highest level of assurance that can be granted without imposing on the developer of the product tasks additional to those required for EAL-1. An analysis of the functional and interface specifications is performed, along with a high-level design review of the subsystems of the product.

Level EAL-3

EAL-3 describes a moderate level of independently assured security, meaning security validated by an outside source. This level permits maximum assurance from the design stage with few alterations to the testing process. Maximum assurance implies that the product has been designed with security in mind instead of security being implemented after design. The developer must provide evidence of testing including vulnerability analyses, which are then selectively verified. This is the first level to also include elements of configuration management evaluation.

Level EAL-4

EAL-4 is the highest level of assurance that is feasible to retrofit an existing product line. An EAL-4-assured product is a methodically designed, tested, and reviewed product, which provides the consumer the highest level of security based on sound commercial software development practices. The use of these practices assist in eliminating the common software design problems that can plague a project.

Aside from the elements of Level EAL-3, EAL-4 also includes an independent search for vulnerabilities in the product.

Level EAL-5

EAL-5 level is not easily achieved for existing products, as the product must be designed and built with the intention of achieving this rating. Here the developer must apply commercial software development practices as well as specialized security engineering techniques. This level is suited for those developers and users who require a high level of assurance through a rigorous development approach. At this level, the developer also must present the design specifications and how these specifications are implemented functionally in the product.

Level EAL-6

Level EAL-6 consists of a semi-formal verified design and test component. This level includes all the elements of the Level EAL-5, with the requirement of a structured presentation of the implementation. Additionally, the product undergoes a high and low design review, and must ensure a high degree of resistance to attack. Level EAL-6 also assures that a structured development process, development controls, and configuration management controls are in place through the design cycle.

Level EAL-7

Level EAL-7 is intended for only the highest level of security applications where high risks of security breaches justify the cost of development and, in turn, the price paid by the consumer. This level consists of a complete independent and formal design review, with a verified design and test stage. The developer must test every facet of the product, looking for obvious and non-obvious vulnerabilities, which are then completely verified by an independent source. This is an exhaustive process, and the evaluating agency must be involved from the conception of the idea to the completion of the product.

Examining Local Security Issues

Aside from the security products or regulations developed outside your organization, you must work to resolve security issues that may be local or restricted to your organization or to a subgroup within your organization. These local security issues include security policies and password controls.

SECURITY POLICIES

Two major stances can be adopted when developing the policies that reflect security at your site. These major statements form the basis of all other security-related policies and regulate the procedures put into place to implement them.

That which is not expressly permitted is prohibited, is the first approach to security. This means that your organization provides a distinct and documented set of services, and anything else is prohibited. For example, if you decide to allow anonymous FTP transfers to and from a particular machine, but deny telnet services, then documenting support for FTP and not telnet illustrates this approach.

The alternative train of thought is *That which is not expressly prohibited is permitted.* This means that unless you expressly indicate that a service is not available, then all services are available. For example, if you do not expressly say that telnet sessions to a given host are prohibited, then they must be permitted. (You can still prevent the service, however, by not allowing a connection to the TCP/IP port.)

Regardless of which train of thought you follow, the reason behind defining a security policy is to determine what action should be taken in the event that the security of an organization is compromised. The policy is also intended to describe what actions will be tolerated and what will not.

THE PASSWORD FILE

The first line of defense against unauthorized access to the system is the /etc/password file. Unfortunately, it is also often the weakest link! The password file consists of lines or records in which each line is split into seven fields with a colon, as illustrated in the following example:

```
chare:u7mHuh5R4UmVo:105:100:Chris Hare:/u/chare:/bin/ksh
username:encrypted password:UID:GID:comment:home directory:login shell
```

Table 2.1 explains the contents and typical values for each file.

TABLE 2.1

The /etc/passwd File

Field Name	*Description*	*Typical Values*
username	This field identifies the user to the system. It is primarily for human benefit.	chare rceh brb

continues

TABLE 2.1, CONTINUED

The /etc/passwd File

Field Name	*Description*	*Typical Values*
	They must be unique on a given machine and, ideally, unique within an organization.	markd
encrypted password	This field contains, or can contain, the encrypted password. How the passwords are encrypted is explained later in this chapter.	u7mHuh5R4UmVo x * NOLOGIN
UID	This is the numerical representation of the user on the system.	0—60,000
GID	This is the numerical representation of the login group to which the user belongs.	0—60,000
comment	This contains information regarding the user. It often lists a user's full name, phone number, or other information.	Chris Hare Ops Manager, x273
home directory	This is the directory where the user is placed upon login.	/u/chare /usr/lib/ppp/ppp-users
login shell	This is the login shell that is started for users to enable them to interact with the system. Remember, not all login shells are interactive.	/bin/sh /bin/csh /bin/ksh /bin/tcsh /usr/lib/uucp/uucico /usr/lib/ppp/ppd/

Table 2.1 provides some additional information worth mentioning. It is not necessary that the actual encrypted password be placed in the encrypted password field. This field can contain other values. For example, to prevent someone from logging in using a specific account, that password can be changed to a non-matchable value, such as NOLOGIN. This field, however, typically contains an x or an asterisk (*), indicating that the password is stored either in the Trusted Computing Base (TCB), which is explained later in this chapter, or in the shadow password file.

The permissions on the password file are read-only, which means that no one can edit the file. Similarly, the shadow password file and the files in the TCB are generally also read-only.

The password information in these files are updated by the password command. The permissions on the password command include the SUID (Set User ID) bit, which makes the user executing the program look like the owner of the program, or in this case, root. Because of the application of the SUID bit, the user can change the password even though he does not have the authority to edit the file.

THE SHADOW PASSWORD FILE

The versions of Unix that do not include the advanced security options from SecureWare can support the shadow password file. When you see the character x in the password field, then the user's actual password is stored in the shadow password file, /etc/shadow. Unlike the /etc/passwd file, the shadow password file must be readable only by root, as illustrated in the following example:

```
# ls -l /etc/shadow
-r--------   1 root     auth        861 Oct 24 11:46 /etc/shadow
#
```

NOTE

SecureWare is a software company specializing in network security, secure Web servers, and privacy-enhanced mail. SecureWare wrote the code for SCO needed to bring the SCO Unix operating system to C2 compliancy. More information on SecureWare can be found at the URL `http://www.secureware.com`.

The format of the /etc/shadow file is identical to the /etc/passwd file in that it also has seven colon-delimited fields. However, the /etc/shadow file only contains the user name, encrypted password, and password aging information, as illustrated in the following example:

```
# cat /etc/shadow
root:DYQC9rXCioAuo:8887:0:0::
daemon:*::0:0::
```

```
mmdf:MZ74AeYMs4sn6:8842:0:0::
ftp:b80lug/92lMeY:8363::::
anyone::8418::::
chare:7xqmj9fj3bVw2:9009::::
pppdemo:4QYrWsJEHc8IA:9062::::
#
```

The /etc/shadow file is created by the command pwconv on both SCO and SVR4 systems. Only the superuser can create the shadow password file. On SCO systems, the information in the /etc/passwd and the protected password database are used to create the shadow password files. On SVR4 systems, the information from /etc/passwd is used.

The primary advantage to using the shadow password file is that it puts the encrypted passwords in a file not accessible to normal users, thereby decreasing the chances that a vandal will be able to steal the encrypted password information.

THE DIALUP PASSWORD FILE

The issue of supporting dialup access directly on a Unix system is open to debate. Many people still do it, but it generally leads to problems. Allowing a vandal the opportunity to "hack" on the system might result in the system's compromise. Many Unix systems offer anonymous UUCP access, which could result in the loss of the password file and would be detrimental to the organization.

An alternative to offering dialup access on a Unix system is to provide access through a terminal server that supports authentication. In this manner, the user must validate to the terminal server before being permitted network access. Because there is no password file to steal from the terminal server, the job of attacking the terminal server becomes more difficult.

In those situations where you must provide dialup access on the Unix server, you can take some steps to better protect yourself from unauthorized network access.

The capability to install additional passwords on serial port lines is unfortunately not present in all versions of Unix. They have become most prominently found on SCO-based systems. The dialup password protection for these serial lines is controlled through two files: /etc/dialups and /etc/d_passwd. The /etc/dialups file contains a list of the serial ports protected by a dialup password. The file is illustrated in the following example:

```
# cat dialups
/dev/tty2A
#
```

The file /etc/d_passwd is used to identify the login shell that a given password is for. Unlike the regular password, which uses the user's login name, the dialup password is tied to the login shell that a given user is using. This means that every user who uses the Bourne shell has the same dialup password. The following example illustrates the typical dialup password.

```
# cat /etc/d_passwd
/bin/sh::
/usr/lib/uucp/uucico::
#
```

Each line or record in the file consists of two colon-delimited fields. The first field identifies the shell that the given password is for, and the second field lists the password. In the preceding example, neither shell has a password. This means that the user will not be prompted for a password when he logs in.

The dialup password is added through the use of the -m option on the passwd command. This option informs passwd that the password being collected is for a specific shell in the dialup passwords file. The syntax for this form of the command is as follows:

```
passwd -m shell_name
```

The execution of this command is illustrated in the following example:

```
# passwd -m /bin/s:
Setting modem password for login shell: /bin/sh
Please enter new password:
Modem password: testing
Re-enter password: testing
#
```

In the preceding example, the system administrator is adding a dialup password to the /bin/sh shell. This means that any users who log in to the system and have the Bourne shell as their login shell will be prompted for the dialup password. The password they must enter is shown in the preceding example as the system administrator would have typed it. Like the normal passwd command, the actual password is not displayed as it is typed. It is shown here only for illustration purposes. Note that only the system administrator can change the dialup password for a shell.

The passwd command modifies the contents of the d_passwd file to include the new password, as illustrated in the following example:

```
# cat d_passwd
/bin/sh:ORa691.Na1jjQ:
/usr/lib/uucp/uucico::
#
```

As shown in the preceding example, the Bourne shell users now have to provide the additional password when logging in on the terminal ports specified in the file /etc/dialups.

The following example illustrates the process that a user now experiences when he logs in with the dialup password:

```
gateway
gateway!login: chare
Password:
Dialup Password:
Last   successful login for chare: Fri Oct 28 22:52:02 EDT 1994 on tty2a
Last unsuccessful login for chare: Tue Oct 18 16:27:56 EDT 1994 on ttyp1
SCO Unix System V/386 Release 3.2
Copyright (C) 1976-1989 UNIX System Laboratories, Inc.
Copyright (C) 1980-1989 Microsoft Corporation
Copyright (C) 1983-1992 The Santa Cruz Operation, Inc.
All Rights Reserved
gateway
TERM = (ansi)
#
```

As illustrated, the user goes through the normal login procedure until he enters the user name and password. After these are validated, the dialup password control file, /etc/dialups, is checked. When the terminal on which the user is connecting is located in the file and the login shell is the Bourne shell, the user is prompted to enter the dialup password. If the dialup password is entered correctly, the user is granted access to the system, as illustrated in the preceding example. If the dialup password is incorrect, however, the login is aborted, and the user is forced to start over again. This is illustrated in the following example:

```
gateway
gateway!login: chare
Password:
Dialup Password:
Login incorrect
Wait for login retry: .
login: chare
Password:
Dialup Password:
Last   successful login for chare: Fri Oct 28 23:28:28 EDT 1994 on tty2a
   1 unsuccessful login for chare: Fri Oct 28 23:30:55 EDT 1994 on tty2a
SCO Unix System V/386 Release 3.2
Copyright (C) 1976-1989 UNIX System Laboratories, Inc.
Copyright (C) 1980-1989 Microsoft Corporation
Copyright (C) 1983-1992 The Santa Cruz Operation, Inc.
All Rights Reserved
gateway
TERM = (ansi)
$
```

THE GROUP FILE

The /etc/group file is used to control access to files not owned by the user. Recall how permissions work: If the user is not the owner of the file, then the group(s) to which the user belongs is checked to see if the user is a member of the group that owns the file. The group membership list is stored in /etc/group. The format of the file is shown in the following:

```
tech:*:103:andrewg,patc,chare,erict,lorrainel,lens
group name:password:GID:userlist
```

Table 2.2 explains each of the fields contained in the /etc/group file.

TABLE 2.2
The /etc/group File

Field Name	*Description*	*Examples*
group name	This is the name of the group. This name is primarily used for human purposes.	tech sales group
password	This is the password to use when switching to this group.	*
GID	This is the numerical group ID number stamped on all files.	0–30,000
userlist	This is a comma-separated list of users that are members of the group.	chare, andrewg

The password for the group file is not used, and no easy mechanisms are available to install a password in the file. If a user attempts to switch to a group that they are not a member of, then they are greeted by a prompt to enter a password, as illustrated in the following example:

```
$ newgrp tech
newgrp: Password
newgrp: Sorry
$ grep tech /etc/group
tech:*:103:andrewg,patc
$
```

If the user who executed this command had been a member of the group tech, then the newgrp command would have been successful. Many current versions of Unix, however, enable a user to be a member of more than one group at a time, thereby reducing or eliminating the need for the newgrp command.

It is important to consider that Berkeley Unix further protects the root account with the wheel group. Only the users who are also members of the wheel group can use the su command to become root.

TIP

To provide access to the root login without providing the actual password, the system administrator should consider the use of the sudo command. More information on sudo, including configuration details, can be found at `http://www.cs.colorado.edu/~millert/sudo/`.

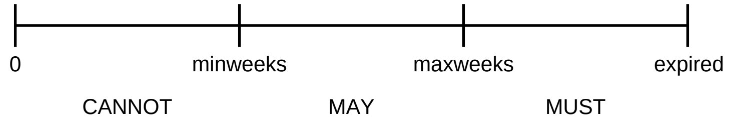
PASSWORD AGING AND CONTROL

Many versions of Unix provide for password aging. This mechanism controls when users can change their passwords by inserting a value into the password file after the encrypted password. This value defines the minimum period of time that must pass before users can change their passwords, and the maximum period of time that can elapse before the password expires.

This explanation becomes clearer by thinking of a timeline, as shown in figure 2.1.

FIGURE 2.1

Password aging and lifetime.



The password aging control information is stored along with the encrypted password, as a series of printable characters. The controls are included after the password, preceded by a comma. Typically a number of characters after the comma represents the following information:

- The maximum number of weeks the password is valid
- The minimum number of weeks that must elapse before the user can change his or her password again
- When the password was most recently changed

The password aging values are defined in table 2.3.

TABLE 2.3

Password Aging Values

Character	*Value in Weeks*	*Character*	*Value in Weeks*
.	0	/	1
0	2	1	3

Character	*Value in Weeks*	*Character*	*Value in Weeks*
2	4	3	5
4	6	5	7
6	8	7	9
8	10	9	11
A	12	B	13
C	14	D	15
E	16	F	17
G	18	H	19
I	20	J	21
K	22	L	23
M	24	N	25
O	26	P	27
Q	28	R	29
S	30	T	31
U	32	V	33
W	34	X	35
Y	36	Z	37
a	38	b	39
c	40	d	41
e	42	f	43
g	44	h	45
i	46	j	47
k	48	l	49
m	50	n	51
o	52	p	53
q	54	r	55

continues

TABLE 2.3, CONTINUED
Password Aging Values

Character	*Value in Weeks*	*Character*	*Value in Weeks*
s	56	t	57
u	58	v	59
w	60	x	61
y	62	z	63

Using this information and looking at a password that has password aging in place, you can decipher the meaning of the following example:

```
chare:2eLNss48eJ/GY,C2:215:100:Chris Hare:/usr1/chare:/bin/sh
```

In the preceding example, the password has been set to age using the value of "C" to define the maximum number of weeks before the password expires, and "2" to define the minimum number of weeks that must pass before the user can change the password again. Each time the user logs in, the expiration time of his password is checked by comparing the last changed value with the maximum. If the user must change his password every three weeks, for example, and it was changed three weeks ago, then the user must change his password. When the user changes his password, the last changed value is updated to reflect when it was last changed.

Two special conditions are recognized by the aging control mechanisms: One forces a user to change his password on the next login, and one prevents a user from being able to change it.

To force a user to change his password, such as for a new user, the password field for that user would be modified to include a comma, followed by two periods for the maximum and minimum time periods. In this case, the user is forced to change his password on the next login. Once changed, the "force" control information is removed from the password entry. An example of the forced entry in the password is shown in the following:

```
chare:2eLNss48eJ/GY,./:215:100:Chris Hare:/usr1/chare:/bin/sh
```

The second special case prohibits a user from being able to change his password. This condition is established by setting the maximum value to be less than the minimum value (that is, first < second). In this case, the user is informed that his password cannot be changed and is illustrated in the following example:

```
chare:,..:215:100:Chris Hare:/usr1/chare:/bin/sh
```

With newer, more secure versions of Unix currently on the market, you may hear the term *password lifetime*. This is a grace period after the maximum time period in which the user can still log in to his account using the expired password. When the lifetime has been reached, the account is disabled. When the user attempts to log in to a system using a disabled account, he is informed that the account has been disabled and to see the system administrator.

The password aging mechanism doesn't prevent a user from changing his password and then changing it back to the old one later. Only a few Unix system versions keep track of what passwords a user has used. The actual process of implementing password aging is version-dependent. To implement it on your system, consult your system documentation.

The program in Listing 2.1, pwexp.pl, is a PERL language program that advises users when their passwords are going to expire so that they can be prepared for the day when the system informs them that their passwords have expired. Note, however, that this version of the program is for standard System V Unix in which a shadow password file is not used.

NOTE

> PERL (Practical Extraction and Report Language) is a freely available and widely used programming language. It combines elements of many of the Unix commands into one neat package. Some of the features include pattern matching, arrays, conditional programming, and subroutines. PERL can be obtained from many anonymous FTP sites including `uunet.uu.net` and `netlabs.com`.

This program addresses the aging mechanism implemented in versions of Unix up to System V Release 3.2. At this level, some variations took place in the interest of increasing system security. The AT&T/USL versions moved to using the /etc/shadow file for storing the password information, whereas the SCO implementations moved to using the Trusted Computing Base facilities from SecureWare. In SCO's Unix 3.2v4 product, the aging control information may be in one of /etc/passwd, /etc/shadow, or the Trusted Computing Base files, depending upon which level of security you configure. It also is important to note in SCO, that both the Protected Password Database and the /etc/shadow database might be in use simultaneously.

VANDALS AND PASSWORDS

The term *hacker* did not always have such a negative connotation. Rather, it was a term denoting someone who was persistent, who tried to break things and figure out how they worked. As a result of this reputation, and because most of the people doing hacking were computer science wizards, the term hacker developed a negative connotation. However, for the purpose of this discussion, and because I know "good" hackers, the bad guys are referred to as *vandals*.

A vandal wants to get access to your system for one reason or another. Some of those reasons can include the following:

- Just for the fun of it
- To look around
- To steal computer resources like CPU time
- To steal trade secrets or other proprietary information

Note that not every attempt to access your system is intended to do harm. However, in most cases they should be treated that way. Regardless of the reasons behind the attack, the piece of information most wanted by a vandal is the /etc/passwd file. When the vandal has a list of valid user account names, it is trivial to create a program to simply guess passwords. However, many modern login programs include a time delay between login prompts that gets longer with each unsuccessful attempt. They also might include program code to disable the access port should too many unsuccessful attempts be recorded.

The primary source of protection is to use /etc/shadow or the Protected Password Database because these files and directories require root access to view them. This makes it harder for the vandal to obtain encrypted password information. However, recall that the password information in /etc/passwd is encrypted. How can the vandal even hope to gain access when he cannot decrypt the passwords as they are stored?

UNDERSTANDING HOW VANDALS BREAK PASSWORDS

It is correct to say that after the password is encrypted it cannot be decrypted. But this doesn't mean that the password is still safe. A password *cracker* is a program used by a vandal that attempts to "guess" the passwords in the /etc/passwd file by comparing them to words in a dictionary. The success of the cracker program is dependent upon the CPU resources, the quality of the dictionary, and the fact that the user has a copy of /etc/passwd.

Password crackers are fairly easy to write. A simple, although less-effective one, can be written in approximately 60 lines of C code or 40 lines of PERL code. That's it. In attempting to provide for better passwords on your system, a system administrator may well give consideration to writing one. Be warned, however, that you might be inviting disaster. If the program is efficient enough, or detects some deficiencies in the passwords on your system, you may have further defeated the security of your machine! An efficient cracker program may be stolen and subsequently used to gain access to other machines. Other ways to better protect the system are available using the logical security tools.

Furthermore, because of the possibility that your cracking program could be stolen, other legal issues could be involved should damage be a direct result of the use of the program. This issue is not to be taken lightly; it is a serious example of how successful these types of programs can be.

For example, a system administrator once had the unfortunate circumstance of not knowing the root password of a machine and no distribution media to reinstall. In this case, he was a friendly vandal in that he owned the machine. He retrieved the /etc/passwd file through anonymous UUCP (Unix to Unix Copy). (So much for security.) After he had the file, he sent it to a contact who tried his password cracking program on it. With no success, he sent it to another machine where a supercomputer chewed on the file for about 18 hours before finally cracking the root password. The ironic part about it was that the password was the name of the company he was working for at the time! Sometimes the simple practice is so effective.

Over the last few years, several password cracking, or guessing, programs have been posted to the Internet. This does not mean that you should run out to get one. In fact, it may be the one program you do not want to have on your system. The authors of these programs clearly state in their documentation that they assume no responsibility for the use of these programs, yet they claim that the programs are highly efficient.

It is highly recommended that you have copies of any password cracking programs on a tightly secured machine and make use of them regularly. However, make sure that the company's security policy allows you, the system administrator, to use the files, and at the same time prohibits everyone else from doing so.

A number of password cracking tools are available; they are listed in Appendix B, "Sources of Information."

The moral of the story is to treat programs that try to guess passwords as dangerous and not worth the potential problems that they could be used to overcome.

WARNING

If you choose to have security programs available for auditing and other evaluations, it is essential that they are not contained on a network-accessible system. Making them available on a network may cause you more grief than you care to have.

C2 SECURITY AND THE TRUSTED COMPUTING BASE

The *Trusted Computing Base* (TCB) is part of the security system for C2-rated Unix systems. It adds a significant level of complexity to the operation of the system and to the administration of it. This system works by moving bits of the /etc/passwd file to other places, as well as adding additional information into the original information. The files that make up the databases for the trusted computing base are scattered in several different directory hierarchies. It is not a good idea to edit these files, however, because serious damage can result to your system.

On a system that uses the TCB, an asterisk is placed in the password field of /etc/passwd. This is because the actual user password is stored along with other user information in the Trusted Computing Base. Using the TCB doesn't change the operation of the system so much as how Unix provides the same services using TCB. On some versions of Unix, such as SCO Unix, even if you are not using C2 security, the Trusted Computing Base is still being used to provide the security services.

Table 2.4 shows the six components to the Trusted Computing Base.

TABLE 2.4

The Trusted Computing Base

Component	*Description*
/etc/passwd	The System Password File
/tcb/auth/files/	The Protected Password Database
/etc/auth/systems/ttys	The Terminal Control Database
/etc/auth/systems/files	The File Control Database
/etc/auth/subsystems/	The Protected Subsystem Database
/etc/auth/system/default	The System Defaults Database

When reference is made to the Trusted Computing Base, it points to all the components of table 2.4 collectively and not any single component.

The operation of a trusted system brings with it some concepts that must be understood to prevent putting the system in jeopardy.

The trusted computing base is comprised of a collection of software including the Unix kernel and the utilities that maintain the trusted computing base. These utilities include authck for verifying and correcting problems in the password database and integrity for verifying the accuracy of the system files. The trusted computing base implements the security policy on the system. This policy is a set of operating rules that governs the interaction between *subjects* such as processes, and *objects* such as files, devices, and interprocess communication objects.

Accountability for an action is defined only if the action can be traced back to a single person. On traditional Unix systems in which more than one person knows the root password, it is difficult, if not impossible, for the action to be traced to any single person. The pseudo-accounts like cron and lp run anonymously—their action can only be traced after the change of system information. This is corrected on a trusted Unix system because each account is associated with a real user, each action is audited, and every action is associated to a specific user.

Little Identification and Authentication (I&A) is performed on the traditional Unix system. On traditional Unix, the user logs in by providing a login name and password combination, which is validated by a search in the /etc/passwd file. If the login name and password are correct, then the user is allowed access to the system. In a trusted system, some additional rules are used to improve upon the standard I&A techniques. For example, new procedures for changing and generating passwords have been established, providing better protection for parts of the password database to prevent prying eyes from accessing it.

The following example illustrates a typical user entry in the trusted computing base on an SCO system that details the information for a specific user. This information should never be hand-edited because doing so could leave your system in a state of uselessness.

```
chare:u_name=chare:\                 # Actual user name
     :u_id#1003:\                    # User ID
     :u_pwd=MWUNe/9lrPqck:\          # Encrypted password
     :u_type=general:\               # User Type
     :u_succhg#746505937:\           # Last Successful Password Change
     :u_unsucchg#746506114:\         # Last Unsuccessful Password Change
     :u_pswduser=chare:\             #
     :u_suclog#747066756:\           # Last successful login
     :u_suctty=tty02:\               # Last successful login on tty
     :u_unsuclog#747150039:\         # Last unsuccessful login
     :u_unsuctty=tty04:\             # Last unsuccessful login on tty
     :u_numunsuclog#1:\              # Number of unsuccessful logins
     :u_lock@:\                      # Lock Status
     :chkent:                        #
```

The preceding example has been modified to insert the comments on the entry. The "# text" does NOT appear in the file. This example is included here to illustrate that in the TCB, other information is, in fact, being tracked. This is not all of the information, but that which is contained in one file. For each user, a file named with the user name is stored and contains the information shown in the preceding example.

In the preceding illustration, the entry for u_succhg shows a value of 746505937. This is how Unix keeps track of time. The value is the number of seconds since January 1, 1970. The value can be given to a function in the Unix kernel that can convert it to the actual date and time.

Traditional Unix systems keep a limited amount of information regarding system activity, and in some cases, only when they have been configured to do so. In trusted Unix, auditing is a major element to ensure that the actions taken are associated with a specific user. The audit system writes a series of records for each action to generate an audit trail of the events that have occurred on a system. This trail consists of every action between a subject and an object that is either successful or unsuccessful in regards to object access, changes made to subject or object, and system characteristics. Consequently, the audit system provides the audit administrator with an extensive history of system actions. This helps the administrator determine what happened, who did it, and when it occurred.

UNDERSTANDING NETWORK EQUIVALENCY

The two primary types of network equivalency are trusted host access and trusted user access. Throughout this discussion, keep in mind that many network administrators prefer to use these facilities to deliver services throughout their organizations without understanding how they operate and what impact these facilities have on host and network security.

HOST EQUIVALENCY

Host equivalency, or *trusted access*, enables the users of a system to access their accounts on remote systems without having to use their login names and passwords. Through the use of the /etc/hosts.equiv file, the system administrator can list all the trusted systems. If no user names are identified for the machine entry, then all the users are trusted. Similarly, if the system administrator does not specify a particular machine for all users, each individual user can use the.rhosts file in their home directory to list machines to which they want trusted access.

Each entry in the hosts.equiv file is trusted. This means that users on the specified machine can access their equivalent accounts on the local machine without a password. This is not applicable for root because this would be a major security problem. For root, and for the user who wants to provide access to systems not included in the system-wide list, use of the .rhosts file in the user's home directory is required.

Consider the following sample /etc/hosts.equiv file:

```
# cat /etc/hosts.equiv
macintosh.mydomain.com
delicious.mydomain.com
#
```

In the preceding example, the entries enable any user on these machines to log into the local system using the same account name without using a password. However, as the following example shows, it is possible to list account names in this file.

```
# cat /etc/hosts.equiv
macintosh.mydomain.com andrewg
delicious.mydomain.com
#
```

As the preceding example illustrates, the user andrewg on that system can log in using any account name other than root on the local system. Consequently, this creates the potential for problems because andrewg's account may be compromised, and then a vandal can access the local system from that machine. As a result, use of account names in the /etc/hosts.equiv and .rhosts files is discouraged.

The security issues regarding host equivalency include root equivalency and file permissions. It is very dangerous to allow root equivalency between systems. Although root equivalency makes the administration tasks easier to complete, it also makes vandalization of the network easier. If security is compromised on one node that has root equivalency with other nodes, it takes the vandal seconds to determine this and access the other machines. Consequently, it is the author's recommendation not to allow root equivalency in your network environment.

Another problem is the permissions on the network access files /etc/hosts.equiv and .rhosts. The /etc/hosts.equiv file must only be writable by root, although other users can read the file. The .rhosts file must only be writable by the owner of the file. Having the file world writable—and even world readable—can create problems such as enabling users to edit the file and add in other systems.

Some implementations of the Berkeley r-commands that use the .rhosts and /etc/host.equiv files check the permissions on them and refuse to use them if the permissions are set inappropriately. However, it wouldn't take too much effort to write a shell program to look for inappropriately authorized .rhosts files.

USER EQUIVALENCY

Trusted user access is much easier to configure but can be very difficult if it is being installed after a list of users has already been configured. User equivalence is a simple concept that is quite different from trusted host access. Trusted host access is not required for user equivalence to work. For NFS (Network File System) use, user equivalence becomes mandatory to prevent access problems.

User equivalence is configured by giving each user on your network, not just the machine, a unique user name and numerical UID (User ID). This means that on each machine, the user has an account with the same UID. If you do not want to allow a user to access a specific machine, then you do not have to provide an account. Another way to explain this is to say that all the /.etc/passwd and /etc/group files on the machines in the network will be the same.

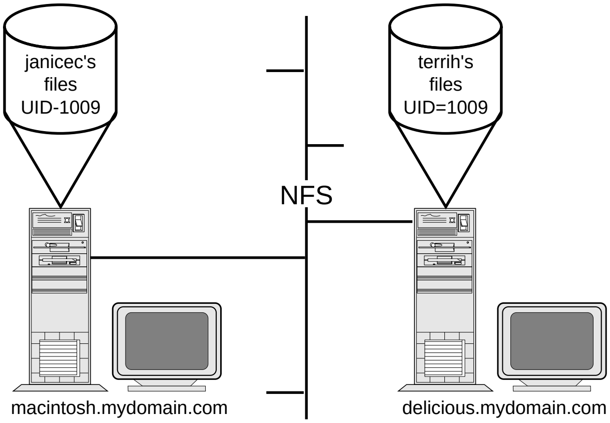
As you will see, not using user equivalence in your networks can put your file systems and data at risk by allowing unauthorized access to them. Consider the following user list:

Username	*UID*
chare	003
janicec	1009
terrih	1009
andrewg	1004

The users in the preceding list each have a unique user name, but their UID numbers are not unique. In figure 2.2, you see that janicec has her account on `macintosh.mydomain.com`, and terrih has her account on `delicious.mydomain.com`.

FIGURE 2.2

A user equivalence example.



The file system on which terrih has her account is part of an exported file system to the users working on macintosh.mydomain.com. When janicec accesses the files in terrih's directory and lists them, the ls -l command lists janicec as the owner of the files because the UID numbers are the same. This means that janicec can erase or modify the information in those files as if she were the rightful owner. In this case, it does not matter that the user names are different. The key with NFS and user equivalence is the UID.

Another type of problem can occur when user equivalence is not properly configured. Consider the following user list:

Username	*Home System*
efudd (Elmer)	delicious
efudd (Elizabeth)	macintosh

In the preceding list, two users named efudd exist. The user efudd on the system named delicious is Elmer Fudd, whereas Elizabeth Fudd uses the account name efudd on the system known as macintosh. In an effort to make things easier, the system administrator on delicious has configured trusted host access, or host equivalency, for all the other systems in the network. One day Elizabeth uses the rlogin command to access macintosh and finds that she can log in without having to provide a password. She can see all the files that belong to Elmer, and she has access to all of them. That is because as far as macintosh is concerned, Elizabeth is really Elmer. In this situation, the user name is the determining factor—not the UID.

As you can see, both types of network equivalency, although creating a more productive and user-friendly environment, can actually create a list of security problems that are difficult to track down and correct.

DEFINING USERS AND GROUPS

As you saw in the preceding discussion on network equivalency, placing some forethought into how you will handle the addition of users in your network is essential to your defense. The comment that your /etc/passwd and /etc/group files will be the same across all the systems is an accurate one. However, a strategy for assigning UID numbers and ensuring their uniqueness is more the issue.

This can be done in many ways. You can assign them in sequential order, allocate number blocks to departments or categories of users, or allocate number blocks to offices. Regardless of the method, it is critical that it be the standard for adding users.

UNDERSTANDING PERMISSIONS

The permissions that Unix uses on each file determine how access to the files and directories is controlled. Many situations can be prevented through the correct application of this simple, yet powerful mechanism. The next section reviews how permissions are handled in the Unix environment.

A REVIEW OF STANDARD PERMISSIONS

The permissions applied to a file are based upon the UID and GID (Group ID) stamped on the file and the UID or GID of the user trying to access the file. The three sets of permissions are as follows:

- Those applicable to the owner of the file
- The users who have the same GID as that on the file
- All other users

For each category of users, there are three permission bits: read, write, and execute. This means that for each file or directory there are nine permission bits. These bits are represented in the following example:

```
$ ls -l output
-rw-r--r--  1 chare   users       236 Aug 24 20:13 output
$
```

In the output in the preceding example, the permissions are as follows:

```
-rw-r--r--
```

The first hyphen represents the file type, which can be one of the types listed in table 2.5, and the remaining characters represent the permissions.

TABLE 2.5
Unix File Types

Symbol	*Description*	*Explanation*
-	Regular file	A file that contains a program, data, or text
d	Directory	A special type of file that contains a list of files and the index to their location on the disk
b	Block device file	A file that allows access to devices such as disk drives

Symbol	*Description*	*Explanation*
c	Character device file	A file that allows access to devices such as terminals and modems
l	Symbolic link	A pointer to another file that can be on this or another filesystem
p	Named pipe	A method of communicating between two processes

In the permissions, the symbols r, w, and x represent the read, write, and execute permissions respectively. Read permission means that the requesting user can view the contents of the file or directory. Write permission grants the user the authority to modify files or create new files in a directory. Finally, execute permission allows the file to be executed like a command. Table 2.6 summarizes these permissions and their impact on a file or directory.

TABLE 2.6
File and Directory Permissions

Permissions Symbol	*Meaning*	*Impact on Files*	*Impact on Directories*
r	read	View the file with cat or more	List the directory with ls
w	write	Modify the file with vi	Create or delete files
x	execute	Run the file as a command	Search the directory looking for a file; use the cd command

When the permissions on files or directories are not set appropriately, it creates pathways into the system for vandals and also offers the potential for any user to make a mistake and invite disaster. The following example shows some problems involving permissions.

This example illustrates a directory that has no write permission, thereby meaning that file cannot be removed with rm.

```
$ id
uid=1009(terrih)  gid=101(users)
$ ls -l
total 4
dr--r--r--  2 chare   users        48 Aug 24 21:09 a
dr-xr-xr-x  2 chare   users        48 Aug 24 21:12 a2
drwxr--xr--x  2 chare   users        32 Aug 24 21:08 micro_light
$ ls -ld a2
dr-xr-xr-x  2 chare   users        48 Aug 24 21:12 a2
$ls a2
output
$ rm a2/output
rm: a2/output not removed. Permission denied
$ ls -l a2
total 1
-rw-rw-r--  1 chare   users       133 Aug 24 21:12 output
$ date > a2/output
$ ls -l a2
total 1
-rw-rw-r--  1 chare   users        29 Aug 24 21:14 output
```

As the user finds out, he cannot remove the file with the rm command because the directory does not have write permission. Remember, a directory is only a special type of file. As the user checks the permission on the file output itself, he sees that the file has its group write permission turned on. This means that the contents of the file can be changed, as shown, by redirecting the output of the date command into the file.

What has not been mentioned yet is the fact that the user executing these commands is not the owner of the file. Here, through a common error, chare didn't protect his files from terrih because the contents of output could be erased even though he thought he had taken the needed precautions.

Such are the woes of the system administrator. Many attacks are successful not because the system administrator isn't doing his job, but because despite the best of intentions, the users leave these types of holes around. Often users change the value of their umask, for example, without understanding it properly. Some facilities, such as ftp and the r-commands, check the permissions on the files used by these commands and prevent their use if the file permissions are not correct.

NOTE

The umask is a bit pattern that is applied against the default permissions to achieve the permissions assigned to a file. Say, for example, the default permissions are 777. When a umask of 022 is applied, the resulting permissions are 755, or rwxr-xr-x.

The Set User ID (SUID) and Set Group ID (SGID) are a major part of the security on a Unix system. The SUID and SGID bits allow a user to assume another identity while executing the program. For example, the passwd command uses the SUID bit to allow users to become root in order to change the contents of the password file.

In the case of the SUID bit, the user who executes the program assumes the identity of the owner of the program. For SGID, the user who executes the program becomes a member of the same group that owns the program.

ROOT AND NFS

When you think about root, you think of uncontrolled access to the files, directories, programs, and devices on a system. However, when root attempts to access files on a remote system through NFS, the root user has little or no permission. This is due to the security features built into NFS. This security feature looks for a UID value of zero. When it finds that value, it knows that this is root, and it remaps the UID value to 65534, or -2. This means that over NFS, root falls into the other category of user.

The advantage here is that if you have no root equivalency between your network machines and one becomes compromised, it becomes harder for the vandal to propagate through your filesystems.

EXPLORING DATA ENCRYPTION METHODS

The opportunity to encrypt information to provide a higher level of security for your system and its data is of interest to users and system administrators everywhere. However, even encrypted data can be at risk without the proper monitoring and training for the users who want to use these facilities.

HOW PASSWORDS ARE ENCRYPTED

At one time, the passwords were stored in a plain text format, and only the administrator and system software had access to the file. However, numerous problems occurred when the password file /etc/passwd was being edited. Most editors create a temporary file, which is the file actually edited. At this point, the file would be world readable, giving away the passwords for all the accounts.

As a result, a method of encrypting the passwords using a one-way encryption algorithm was developed. The encrypted values were then stored instead of the clear text. However, the security of the system is only as good as the encryption method chosen.

When a user logs in to a Unix system, the getty program prompts the user for his user name and then executes the login program. The login program prompts for the password but does not decrypt it. In fact, the login program encrypts the password and then compares the newly encrypted value to the one stored in /etc/passwd. If they match, then the user supplied the correct one.

The Unix encryption method for password encryption is accessed through a kernel mechanism named *crypt(3)*. Because of United States Federal licensing issues, the crypt routines might not be available on your machine. The issue is that while the needed routines to encrypt information are available, those programs that decrypt information are not available outside the United States.

The actual password value stored in /etc/passwd is the result of using the user's password to encrypt a 64-bit block of zeroes using the crypt(3) call. The *clear text* is the user's password, which is the key to the encryption operation. The text being encrypted is 64 bits of zeroes, and the resulting *cipher text* is the encrypted password.

This crypt(3) algorithm is based upon the Data Encryption Standard (DES) developed by the National Institute of Standards and Technology or NIST. In normal operation according to the DES standard, a 56-bit key, such as eight 7-bit characters, is used to encrypt the original text, which is commonly called clear text. This clear text is typically 64 bits in length. The resulting cipher text cannot easily be decrypted without knowledge of the original key.

The Unix crypt(3) call uses a modified version of this method by stating that the clear text is encrypted in a block of zeroes. The process is complicated by taking the resulting cipher text and encrypting it again with the user's password as the key. This process in performed 25 times! When complete, the resulting 64 bits are split into 11 printable characters and then saved in the password file.

Despite the fact that the source for crypt can be obtained from many vendors, although the commercial distribution of this is limited outside the United States (you can find public versions of the code on the Internet), no known method is available to translate the cipher text or encrypted value back to its original clear text.

Robert Morris, Sr. and Ken Thompson, who originally implemented the Unix crypt(3) technology, were afraid that with the advent of hardware DES chips, the security of the Unix system could be easily bypassed. By using a "grain of salt," they managed to bypass this threat.

The "grain of salt," which has become known as salt, is a 12-bit number that is used to modify the result of the DES function. The value of this 12-bit number ranges from 0 to

4,095. So for each possible password, 4,096 ways exist that each password could be encrypted and stored in the password file. It is possible for multiple users on the same machine to use the same password, and no one, including the system manager, would be the wiser.

When a user runs the /bin/passwd program to establish a new password, the /bin/passwd program picks a salt based upon the time of day. This salt is used to modify the user's password.

The problem comes later in encrypting the password the next time the user logs in. It is possible, but unlikely, that the user will log in and the salt will be the same. For things to work correctly, Unix stores the salt in /etc/passwd as well. It, in fact, makes up the first two characters of the encrypted password. The following is a sample encrypted value.

```
2eLNss48eJ/GY
```

In the preceding example, the initial two characters—2e—are the salt for this password. When the user logs in to the system, the login program collects the salt from the stored password and uses it to encrypt the password provided by the user. If the newly encrypted password and the stored password match, then the password entered by the user is correct, and the user is logged in to the system. If the values do not match, then the user is greeted by a Login incorrect message, and the user must try to login again.

ENCRYPTING FILES

As you have seen, the encryption of passwords using a mechanism that is not easily decrypted provides a relatively secure method of preventing unauthorized users from having access to the system. But how can users prevent unauthorized access to their files? This can be accomplished through the use of the command crypt(1). This is a relatively unsecured method of encryption, however. Interestingly enough, some Unix commands support the direct manipulation of these encrypted files without having to decrypt them first.

Encrypting files using the crypt command is relatively simple. If no arguments are provided on the command line, crypt prompts for the key, reads the data to encrypt from standard input, and prints the encrypted information on standard output. Ideally however, the information to use is provided on the command line, as illustrated in the following example:

```
crypt key < clear > cipher
```

The preceding command reads from the file clear, encrypts the test using the password key, and saves the resulting encrypted text in the file cipher. Encrypted files can be viewed or decrypted by using a similar command line, as shown in the following:

```
crypt key < cipher > clear
crypt key < cipher ¦ pr ¦ lp
```

In the first command of the preceding example, the encrypted text in cipher is decrypted using key and saved in the file called clear. The second command-line example uses crypt to decrypt the text and sends the resulting clear text to pr for formatting and then to lp to be printed. The files generated by crypt can be edited by ed or vi, provided the version of ed or vi you have on your system supports editing encrypted files.

The exact mechanism used by crypt is well documented, and many versions of this command are publicly available on the Internet. crypt(1) does not use the same encryption routines as crypt(3), which is used for password encryption. The mechanism is a one-rotor encryption machine designed along the lines of the German Enigma, but using a 256-element rotor. Although the methods of attack on such types of encryption machines are known, the amount of work required may be large.

The encryption key used is the limiting factor to determine the level of effort to decrypt the data. The longer the password, the more complex the encryption pattern, and the longer it takes to transform the key to the internal settings used by the machine. For example, the transformation process is meant to take close to a second, but if the key is restricted to three lowercase letters, encrypted files can be read by expending only a substantial fraction of five minutes of real machine time!

Because the key given to crypt might be seen by prying eyes using ps, crypt destroys any record of the actual key immediately upon entry. Consequently, like any other security system, the password used to encrypt the data is the most critical component—and the most suspect.

NOTE

This encryption mechanism is not unbreakable and can be figured out given enough time. One method of improving your chances of protecting your data is to compress the file prior to encrypting it.

EXAMINING KERBEROS AUTHENTICATION

The Kerberos Authentication system was developed by the Massachusetts Institute of Technology's Project Athena. Since that time, Kerberos has been adopted by other organizations for their own purposes. Many third-party application developers include support for the Kerberos Authentication system in their products.

UNDERSTANDING KERBEROS

Kerberos is an authentication system. Simply put, it is a system that validates a principal's identity. A *principal* can be either a user or a service. In either case, the principal is defined by the following three components:

- Primary name
- Instance
- Realm

In Kerberos terminology, this is called a three-tuple and is illustrated by the following example:

`<primaryname, instance, realm>`

The *primaryname*, in the case of a genuine person, is the login identifier. The *instance* is either null or contains particular information regarding the user. For a service, the primaryname is the name of the service, and the machine name is used as the instance, as in rlogin.mymachine. In either case, the *realm* is used to distinguish between different authentication domains. By using the realm, it is possible to have a different Kerberos server for each small unit within an organization instead of a single large one. The latter situation would be a prime target for vandals because it would have to be universally trusted throughout the organization. Consequently, it is not the best configuration choice.

Kerberos principals obtain tickets for services from a special server known as a *ticket-granting server*. Each ticket consists of assorted information identifying the principal that is encrypted in the private key for that service. Because only Kerberos and the service know the private key, it is considered to be authentic. The ticket granted by the ticket-granting server contains a new private session key that is known to the client as well. This key is often used to encrypt the transactions that occur during the session.

The major advantage to the Kerberos approach is that each ticket has a specific lifetime. After the lifetime is reached, a new ticket must be applied for and issued from the ticket-granting server.

DISADVANTAGES OF KERBEROS

As mentioned earlier, the Kerberos system was originally developed at MIT during the development of Project Athena. The disadvantage here is that the configuration of the environment at MIT is unlike any other organization. Simply speaking, Kerberos is designed to authenticate the end user—the human being sitting at the keyboard—to some number of servers. Kerberos is not a peer-to-peer system, nor was it meant for one computer system's daemons to contact another computer.

Several major issues concern the Kerberos authentication system. First and foremost, the majority of computer systems do not have a secure area in which to save the keys. Because a plain text key must be stored in the initial dialog to obtain a ticket-granting ticket, there must be a secure place to save this information. In the event that this plain text key is obtained by an unauthorized user, the Kerberos authentication server in that realm can be easily compromised.

The next problem is how Kerberos handles keys on multiuser computers. In this case, the cached keys can be obtained by other users logged into the system. In a single-user workstation environment, only the current user has access to system resources, so there is little or no need to enable remote access to the workstation. However, if the workstation supports multiple users, then it is possible for another user on the system to obtain the keys.

Other weaknesses also exist in the Kerberos protocol, but these are difficult to discuss without a thorough understanding of how the protocol works and is implemented.

UNDERSTANDING IP SPOOFING

IP Spoofing is a potentially dangerous threat to any network connected to the Internet. On a network, a host allows other "trusted" hosts to communicate with it without requiring authentication. This is accomplished by setting up the .rhost and other files. Any communications coming from sources other than those defined as trusted must provide authentication before they are allowed to establish communication links.

With IP Spoofing, a host not connected to the network connects and makes it look as if they are one of the trusted hosts on the network. This is accomplished by changing their IP number to one of those on the network. In essence, the intruding host impersonates a local system's IP address and fools other hosts into not prompting for authentication.

Security measures to avoid being hit by IP Spoofing include avoiding any IP address based authentication. Require passwords every way you can, and implement encryption based authentication. Many firewalls are also capable of checking the IP source address against the physical location of origin and ascertaining whether or not the data is coming from a real host.

SUMMARY

This chapter examined some of the areas that both the user and system administrator must be concerned with to protect the system. It is vitally important that all users on a machine understand that the security of the machine is not up to the system administrator alone, but also to the users.

For example, the use of good-quality passwords helps to make the system less accessible. The fact of the matter is, though, that few users choose good passwords. Consequently, the system administrator can help achieve a higher level of security through the following mechanisms:

- Periodic examination of the system for common problems
- Use of a security probing tool like COPS (Computer Oracle and Password System) or TAMU (Texas A&M University)
- Education of system users that they each hold a key to the front door of the system

ACKNOWLEDGMENTS

Some of the text and examples in this chapter originally appeared in *Sys Admin Magazine*, Volume 1, No 1, “How Unix Password Controls Work” by Chris Hare; Volume 2, No 4, “More Network Security: Equivalency;” Volume 2, No. 6, “C2 Security: Is Big Brother Watching?”. *Sys Admin Magazine* is published bimonthly by R&D Publications.

A SAMPLE PROGRAM

The sample program pwexp.pl reports the number of days until the user’s current password expires. This program could be called from the /etc/profile file and if password aging is enabled for that user, it will report the number of days before the password expires.

Worth noting about this program is that the getpwuid command provides more fields than are actually in the /etc/passwd file. This is because the command puts the password aging information into a separate variable instead of having the programmer split the aging data from the password.

LISTING 2.1—PWEXP.PL

```
#! perl
eval '(exit $?0)' && eval 'exec perl -S $0 ${1+"$@"}'
& eval 'exec perl -S $0 $argv:q'
if 0;
#
# pwexp.pl - PERL program to check for password expiration times
#
#
#
# get the passwd file entry for this account. $< is the numerical
# representation of our REAL UID
#
```

```
 ( $username, $passwd, $uid, $gid, $pwage,
   $comment, $gcos, $dir, $shell ) = getpwuid($<);
#
# If passwd aging value is defined
#
if ( $pwage ne "" )
   {
   #
   # extract the maxweeks value
   #
   $maxweeks = &a64l( substr( $pwage, 0, 1 ) );
   #
   # extract the minweeks value
   #
   $minweeks = &a64l( substr( $pwage, 1, 1 ) );
   #
   # extract the last changed value
   #
   $lastchange = &a64l( substr( $pwage, 2 ) );
   #
   # what is NOW?
   #
   $now = time / 604800;
   #
   # If maxweeks < minweeks, the user can't change his passwd
   #
   if ( $maxweeks < $minweeks )
          {
          printf "You cannot change your password. Ever. \n";
          }
   #
   # The special case where the password must be changed
   #
   elsif ( ( $minweeks == 0 ) && ( $maxweeks == 0 ) )
          {
          printf "You must change your password. Now.\n";
          }
   #
   # if lastchanged is > now, then expired
   # if now > lastchanged + maxweeks, then expired
   #
   elsif ( $lastchange > $now  ¦¦
                 ( $now > $lastchange + $maxweeks ) &&
                 ( $maxweeks > $minweeks ) )
          {
          printf "Your password has expired.\n";
          }
   #
   # tell the user when his password expires
   #
   else
```

```
          {
          printf "Your password expires in %d weeks.\n",
                 $lastchange + $maxweeks - $now;
          printf "Please start thinking of a new one.\n";
          }
   }
else
   {
   printf "Password aging is not enabled.\n";
   }
exit(0);
#
# the a64l routine was written by Randall Schwartz after a call for help
# in the comp.lng.perl newsgroup. Thanks Randall!
#
sub a64l {
        local($_) = @_; # arg into $_
        die "a64l: illegal value: $_" unless m#^[./0-9A-Za-z]{0,6}$#;
        unless (defined %a64map) {
                @a64map{'.','/',0..9,'A'..'Z','a'..'z'} = 0..63;
        }
        local($result) = 0;
        for (reverse split(//)) {
                $result *= 64;
                $result += $a64map{$_};
        }
        $result;
}
```

CHAPTER 3

DESIGNING A NETWORK POLICY

BEFORE BUILDING A FIREWALL in preparation for connecting your network to the rest of the Internet, it is important to understand exactly what network resources and services you want to protect. The *Network Policy* is a document that describes an organization's network security concerns. This document becomes the first step in building effective firewalls.

This chapter discusses this first step of designing a Network Policy for your Internet. Among the issues explored are overall network security planning, site security policy, and risk analysis. This chapter discusses identification of resources and threats, network use and responsibilities, and action plans when the security policy is violated. It helps you monitor system use, mechanisms, and schedules; it also covers account and configuration management procedures and recovery procedures. Finally, this chapter discusses using encryption to protect the network, using authentication systems, and using Kerberos, mailing lists, newsgroups, and security response teams.

NETWORK SECURITY PLANNING

It is important to have a well-conceived and effective network security policy that can guard the investment and information resources of your company. A network security policy is worth implementing if the resources and information your organization has on its networks are worth protecting. Most organizations have sensitive information and competitive secrets on their networks; these should be protected against vandalism in the same manner as other valuable assets such as corporate property and office buildings.

Most network designers commonly begin implementing firewall solutions before a particular network security problem has been properly identified. Perhaps one reason for this is that coming up with an effective network security policy means asking some difficult questions about what types of Internetwork services and resources you are going to allow your users to access, and which ones you will have to restrict because of security risks.

If your users currently enjoy unrestricted access to the network, it can be difficult to enforce a policy that restricts their access. You should also keep in mind that the network policy that you should use is such that it will not impair the functioning of your organization. A network security policy that prevents network users from effectively implementing their tasks can have an unwanted consequence: network users may find means of bypassing your network policy, rendering it ineffective.

An effective network security policy is something that all network users and network administrators can accept and are willing to enforce.

SITE SECURITY POLICY

An organization can have multiple sites with each site having its own networks. If the organization is large, it is quite probable that the sites have differing network administrations with different goals and objectives. If these sites are not connected through an internal network, each of these sites may have their own network security policies. However, if the sites are connected through an internal network, the network policy should encompass the goals of all the interconnected sites.

In general a site is any part of an organization that owns computers and network-related resources. Such resources include, but are not restricted to, the following:

- Workstations
- Host computers and servers
- Interconnection devices: gateways, routers, bridges, repeaters

- Terminal servers
- Networking and applications software
- Network cables
- Information in files and databases

The site security policy should take into account the protection of these resources. Because the site is connected to other networks, the site security policy should consider the security needs and requirements of all of the interconnected networks. This is an important point because it is possible to come up with a network security policy that safeguards your interests but can be harmful to others. An example of this could be a deliberate use of IP addresses, behind the firewall gateway, that are already used by someone else. In this case, attacks made against your network by spoofing your network's IP addresses will be deflected to the organization whose IP addresses you are using. This situation should be avoided, because it is in your own interests to be a "good citizen" of the Internet.

NOTE

> RFC 1244 discusses site security policy in considerable detail. Many of the security policy issues in this chapter are based on the issues raised in this RFC.

APPROACH TO SECURITY POLICY

Defining a network security policy means developing procedures and plans that safeguard your network resources against loss and damage. One possible approach to developing this policy is to examine the following:

- What resources are you are trying to protect?
- Which people do you need to protect the resources from?
- How likely are the threats?
- How important is the resource?
- What measures can you implement to protect your assets in a cost-effective manner and timely manner?
- Periodically examine your network security policy to see if your objectives and network circumstances have changed.

Figure 3.1 shows a worksheet that can be used to help you direct your thinking along these lines.

- The “Network Resources Number” column is an internal identification network number of the resources to be protected (if applicable).
- The “Network Resources Name” column is an English description of the resources. The importance of the resource can be on a numeric scale of 0 to 10, or in “fuzzy” natural-language expressions such as low, high, medium, very high, and so on.
- The “Type of users to protect resource from” column can have designations such as internal, external, guest, or group names such as accounting users, corporate assistants, and so on.
- The “Likelihood of a threat” column can be on a numeric scale of 0 to 10, or in “fuzzy” natural language expressions such as low, high, medium, very high, and so on.
- The “Measures to be implemented to protect network resource” column can have values such as “operating system permissions” for files and directories; “audit trail/alerts” for network services; “screening routers” and “firewall” for hosts and networking devices; or any other description of the type of security control.

In general, the cost of protecting your networks against a threat should be less than the cost of recovering should you be affected by the security threat. If you do not have sufficient knowledge of what you are protecting and the sources of the threat, reaching an acceptable level of security can be difficult. Don’t hesitate to enlist the help of others with specialized knowledge concerning your network’s assets and possible threats aginst it.

It is important to get the right type of people involved in the design of the network security policy. You might already have user groups who would consider implementation of a network security policy to be their specialty. These groups could include those that are involved with auditing/control, campus information systems groups, and organizations that deal with physical security. If you want universal support of the network security policy, it is important to involve these groups so that you can have their cooperation and acceptance of the network security policy.

Worksheet for Developing Security Approach

Network Resources			Type of users to protect resource from	Likelihood of a threat	Measures to be implemented to protect network resource
Number	Name	Importance of resource			

FIGURE 3.1 *A worksheet for developing a security approach.*

ENSURING RESPONSIBILITY FOR THE SECURITY POLICY

An important aspect of network security policy is ensuring that everyone knows their own responsibility for maintaining security. It is difficult for a network security policy to anticipate all possible threats. The policy can, however ensure that each type of problem does have someone who can deal with it in a responsible manner. There may be many levels of responsibility associated with a network security policy. Each network user, for example, should be responsible for guarding their password. A user who allows their account to be compromised increases the likelihood of compromising other accounts and resources. On the other hand, network administrators and system managers are responsible for maintaining the overall security of the network.

RISK ANALYSIS

When you create the network security policy, it is important to understand that the reason for creating a policy in the first place is to ensure that efforts spent on security are cost-effective. This means that you should understand which network resources are worth protecting, and that some resources are more important than others. You also should identify the threat source that you are protecting the network resource from. Despite the amount of publicity about external intruders breaking into a network, many surveys indicate that for most organizations, the actual loss from “insiders” is much greater.

Risk analysis involves determining the following:

- What you need to protect
- What you need to protect it from
- How to protect it

The risks should be ranked by the level of importance and severity of loss. You should not end up with a situation where you spend more to protect that which is of less value to you. Risk analysis involves determining the following two factors:

1. Estimating the risk of losing the resource *(Ri)*
2. Estimating the importance of the resource *(Wi)*

As a step towards quantifying the risk of losing a resource, a numeric value can be assigned. For example, the risk *(Ri)* of losing a resource can be assigned a value from 0 to 10, where 0 represents no risk, and 10 represents the highest risk. Similarly, the importance of a resource *(Wi)* can be assigned a value from 0 to 10, where 0 means no importance and 10 means highest importance. The overall weighted risk for the resource is then, the numeric product of the risk value and its importance (also, called the weight). This can be written as the following:

```
      WRi = Ri*Wi
WRi = Weighted risk of resource "i"
Ri = Risk of resource "i"
Wi = Weight (importance) of resource "i"
```

Consider figure 3.2, a simplified network with a router, a server, and a bridge. Assume that the network and system administrators have come up with the following estimates for the risk and importance of the network devices.

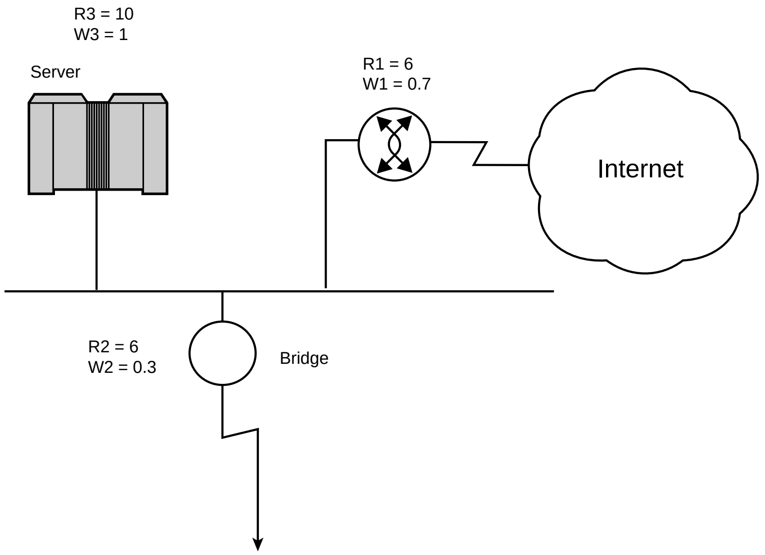


FIGURE 3.2

A simplified network layout with valued weights and risks.

Router:

R1 = 6

W1 = .7

Bridge:

R2 = 6

W2 = .3

Server:

R3 = 10

W3 = 1

The computation of the weighted risks of these devices is shown next:

Router:

WR1 = R1 * W1 = 6 * 0.7 = 4.2

Bridge:

WR2 = R2 * W2 = 6 * 0.3 = 1.8

Server:

WR3 = R3 * W3 = 10 * 1 = 10

Figure 3.3 shows a worksheet that you can use for recording the previous calculations.

- The "Network Resources Number" column is an internal identification network number of the resource (if applicable).
- The "Network Resources Name" column is an English description of the resources.
- The "Risk to Network Resources (*Ri*)" column can be on a numeric scale of 0 to 10, or in "fuzzy" natural language expressions such as low, medium, high, very high, and so on.
- Similarly, the "Weight (Importance)of Resource (*Wi*)" column can be on a numeric scale of 0 to 10, or in "fuzzy" natural language expressions such as "low," "high," "medium," "very high," and so on. If you are using numerical values for the risk and weight columns, you can compute the value in the "Weighted Risk (*Ri* * *Wi*)" column as the product of the Risk and Weight values.

One can compute the overall risk of the resources on the network by using the following formula:

```
WR = (R1*W1 + R2*W2 + .... + Rn*Wn)/(W1 + W2 + ... + Wn)
```

For the network in figure 3.2, the overall network risk would be as follows:

```
WR = (R1*W1 + R2*W2 + R3*W3)/(W1 + W2 + W3)
   = (4.2 + 1.2 + 10)/(0.7 + 0.3 + 1)
   = 15.4/2
   = 7.7
```

Worksheet for Network Security Risk Analysis

Network Resources		Risk to Network Resources (Ri)	Weight (Importance) of Resource (Wi)	Weighted Risk (Ri * Wi)
Number	Name			

FIGURE 3.3 *A sample worksheet for a network security analysis.*

NOTE

> A threat and risk assessment should not be a one-time activity; it should be carried out regularly, as defined in the site security policy. The United States Fish and Wildlife Service has documented the issues involved in performing a threat and risk assessment. The VRL for the threat and risk assessment document is `http://www.fws.gov/~pullenl/security/rpamp.html`.

Other factors to consider in estimating the risk to a network resource are its availability, integrity, and confidentiality. Availability of a resource is a measure of how important it is to have the resource available all the time. Integrity of a resource is a measure of how important it is that the resource or the resource's data be consistent. This is particularly important for database resources. Confidentiality applies to resources such as data files to which you wish to restrict access.

IDENTIFYING RESOURCES

In performing a risk analysis, you should identify all the resources that are at risk of a possible security breach. Resources such as hardware are fairly obvious to list in this estimate, but resources such as the people who actually use the systems are often overlooked. It is important to identify *all* network resources that could be affected by a security problem.

RFC 1244 lists the following network resources that should be considered in estimating threats to overall security:

1. **HARDWARE:** processors, boards, keyboards, terminals, workstations, personal computers, printers, disk drives, communication lines, terminal servers, routers.
2. **SOFTWARE:** source programs, object programs, utilities, diagnostic programs, operating systems, communication programs.
3. **DATA:** during execution, stored on-line, archived off-line, backups, audit logs, databases, in transit over communication media.
4. **PEOPLE:** users, people needed to run systems.
5. **DOCUMENTATION:** on programs, hardware, systems, local administrative procedures.
6. **SUPPLIES:** paper, forms, ribbons, magnetic media.

IDENTIFYING THE THREATS

Once the network resources requiring protection are identified, you should identify the threats to these resources. The threats can then be examined to determine what potential for loss exists. You should also identify from what threats you are trying to protect your resources.

The following sections describe a few of the possible threats.

DEFINING UNAUTHORIZED ACCESS

Access to network resources should only be permitted to authorized users. This is called *authorized access*. A common threat that concerns many sites is unauthorized access to computing facilities. This access can take many forms, such as use of another user's account to gain access to the network and its resources. In general, the use of any network resource without prior permission is considered to be *unauthorized access*.

The seriousness of an unauthorized access depends on the site and the nature of the potential loss. For some sites, the mere act of granting access to an unauthorized user may cause irreparable harm by negative media coverage.

NOTE

> Some sites, because of their size and visibility, may be more frequent targets than others. The Computer Emergency Response Team (CERT) has made the observation that in general, well-known universities, government sites, and military sites seem to attract more intruders. More information on CERT, as well as other similar organizations, can be found in the section, "Security Response Teams," later in this chapter.

RISK OF DISCLOSURE OF INFORMATION

Disclosure of information, whether it is voluntary or involuntary, is another type of threat. You should determine the value or sensitivity of the information stored on your computers. In the case of hardware and software vendors, source code, design details, diagrams, and product-specific information represent a competitive edge. Hospitals, insurance companies, and financial institutions maintain confidential information, the disclosure of which can be damaging to the clients and to the reputation of the company. Pharmaceutical labs may have patent applications and cannot risk loss due to theft.

At a systems level, disclosure of a password file on a Unix system may make you vulnerable to unauthorized access at a future date. For many organizations, a glimpse of a proposal or research project containing many years of research may give a competitor an unfair advantage.

NOTE

> People often make the assumption that network and computer break-ins are made by individuals working independently. This is not always so. The dangers of systematic industrial and governmental espionage are unfortunate realities of life.
>
> In addition, when a break-in is achieved, the information typically flows through the Internet in very short order. There are newsgroups and Internet Relay Chat (IRC) channels where users share break-in information.

DENIAL OF SERVICE

Networks link valuable resources such as computers and databases, and provide services that an organization depends on. Most users on such networks rely on these services for performing their jobs efficiently. If these services are not available, there is a corresponding loss in productivity. A classic example of this was the Internet Worm incident that took place on November 2–3, 1988, where a large number of computers on the network were rendered unusable.

It is difficult to predict the form in which the denial of service may come. The following are some examples of how a denial of service can affect a network.

- A network may be rendered unusable by a rogue packet.
- A network may be rendered useless by traffic flooding.
- A network may be partitioned by disabling a critical network component such as a router joining the network segments.
- A virus might slow down or cripple a computer system by consuming system resources.
- The actual devices that protect the network may be subverted.

NOTE

You should determine which services are absolutely essential, and for each of these services determine the effect of loss of this service. You should also have contingency policies on how you can recover from such losses.

NETWORK USE AND RESPONSIBILITIES

There are a number of issues that must be addressed when developing a security policy:

1. Who is allowed to use the resources?
2. What is the proper use of the resources?
3. Who is authorized to grant access and approve usage?
4. Who may have system administration privileges?
5. What are the user's rights and responsibilities?

6. What are the rights and responsibilities of the system administrator vs. those of the user?
7. What do you do with sensitive information?

These issues are discussed in the sections that follow.

IDENTIFYING WHO IS ALLOWED USE OF NETWORK RESOURCES

You must make a list of users who need access to network resources. It is not necessary to list every user on the network. Most network users fall into groups such as accounting users, corporate lawyers, engineers, and so on. You must also include a class of users called the external users. These are users who can access your network from elsewhere, such as stand-alone workstations or other networks. These external users can be users who are not employees, or who are employees accessing the network from their homes or while traveling.

IDENTIFYING THE PROPER USE OF A RESOURCE

After you determine which users are allowed access to network resources, you should provide guidelines for the acceptable use of these resources. The guidelines will depend on the class of user, such as software developers, students, faculty, external users, and so on. For each of these user classes, you should have separate guidelines. The policy should state what types of network usage are acceptable and unacceptable, and what type of usage is restricted. The policy that you develop is called the Acceptable Use Policy (AUP) for your network. If access to a network resource is restricted, you should consider the level of access the different user classes will have.

NOTE

Your AUP may clearly state that individual users are responsible for their actions. Each user's responsibility exists regardless of the security mechanisms that are in place. It makes no sense building expensive firewall security mechanisms if a user can disclose the information by copying files on disk or tape and making data available to unauthorized individuals.

Though it may seem obvious, the AUP should clearly state that breaking into accounts or bypassing security is not permitted. This can help avoid legal issues raised by employees who bypass network security and later claim they were not properly informed or trained about network policy. The following is a guideline of issues that should be covered when developing an AUP:

- Is breaking into accounts permitted?
- Is cracking passwords permitted?
- Is disrupting service permitted?
- Should users assume that a file being world-readable grants them the authorization to read it?
- Should users be permitted to modify files that are not their own, even if they happen to have write permission?
- Should users share accounts?

Unless you have unusual requirements, the answer to most of these questions for most organizations will be *No*.

Additionally, you may want to incorporate a statement in your policies concerning copyrighted and licensed software. In general, your network usage procedures should be such that it should be difficult for employees to download unauthorized software from the network. Copying software illegally is punishable by law in most countries in the West. Large organizations often have very strict policies concerning licensing because of the risk of lawsuits and damage caused by publicity of incidents.

NOTE

> Many licensed software products for networks determine their usage and restrict the number of users that can access the network. However, some licensing agreements may require that you monitor the license usage so that the license agreement is not violated.

You may want to include information concerning copyrighted or licensed software in your AUP. Examples of the points that you may want to address are the following:

- Copyrighted and licensed software may not be duplicated unless it is explicitly stated.
- Indicate methods of conveying information on the copyright/licensed status of software.
- Err on the side of caution. When in doubt, do not copy.

An AUP that does not clearly state what is prohibited can make it difficult to prove that a user has violated the policy. Exemptions to your policy could be members of so-called *tiger teams,* charged with the responsibility of probing the security weakness of your networks. The users comprising these tiger teams must be clearly identified. On occasion you may have to deal with users who are self-appointed members of the tiger team and want to probe the security weakness for research purposes or to make a point. Your AUP should address the following issues about security probing:

- Is user-level hacking permitted at all?
- What type of security probing activities are permitted?
- What controls must be in place to ensure that the security probing does not get out of control?
- What controls must be in place to protect other network users from being victims of security probing activities?
- Who should have the permission to do security probing, and what is the process for obtaining permission to conduct these tests?

If you want to permit legitimate security probing, you should have separate network segments and hosts on your network for these tests. In general, it is very dangerous to test worms and viruses. If you must perform these tests, it would be foolish to conduct them on a live network. You should, instead, physically isolate the hosts and network segments that are used for the test, and do a complete and careful reload of all the software at the end of each test.

Assessing the security weaknesses and taking proper measures can be effective in repelling hacker attacks. Some organizations use outside consultants to evaluate the security of their services. As part of this evaluation, they may legitimately perform “hacking.” Your policy should make allowances for these situations.

DETERMINING WHO IS AUTHORIZED TO GRANT ACCESS AND APPROVE USAGE

The network security policy should identify who is authorized to grant access to your services. You also should determine what type of access these individuals can grant. If you cannot control who is granted access to your system, it is difficult to control who is using your network. If you can identify the persons who are charged with granting access to the network, you can trace what type of access or control has been granted. This is helpful in identifying the cause of security holes as a result of users being granted excessive privileges.

You may want to consider the following factors in determining who will grant access to services on your networks:

- Will access to services be granted from a central point?
- What methods will you use for creating accounts and terminating access?

If the organization is large and decentralized, you may have several central points, one for each department, that is responsible for the security of its departmental network. In this case you must have global guidelines on what types of services are to be permitted for each class of user. In general, the more centralized the network administration is, the easier it is to maintain security. On the other hand, centralized administration can create problems when departments want greater control over their network resources. The correct amount of centralization or decentralization will depend on factors that are beyond the scope of this discussion.

The system administrators will, of course, need special access to the network, but other users may need certain privileges as well. The network security policy should address this issue. A universal policy restricting all special privileges, while being more secure, may prevent legitimate users from performing their work. A more balanced approach is needed.

NOTE

The challenge is to balance restricted access to special privileges in order to make the network more secure, versus giving people who need these privileges access so that they can perform their tasks. In general you should grant only enough privilege to accomplish the necessary tasks.

Some system administrators take the easy way out and assign more privileges than the user needs, so the users will not bother them again. Also, the system administrator may not properly understand the fine points in assigning security and may err on the side of giving more privileges. Training and education can help avoid these types of problems.

People holding special privileges should be responsible and also accountable to some authority identified within the security policy. Some systems may have custom audit trails that can be used so that privileged users cannot abuse their trust.

WARNING

If the people you grant privileges to are not accountable and responsible, you run the risk of creating security holes in the system and inconsistent granting of permissions to users. Such systems are invariably difficult to manage.

If there are a large number of network and system administrators, it is difficult to keep track of what permissions have been granted to network resources. A formalized way of granting requests can be used. After the user makes the request and the request is authorized by the user's supervisor, the system administrator must document the security restrictions or access the user has been granted.

Figure 3.4 shows a worksheet that can be used to keep a paper trail of what permissions have been granted to a user. The following is a description of the columns used in this worksheet.

- The network resource number column is an internal identification network number of each resource (if applicable).
- The resource name column is an English description of the resources.
- The type of access column can be used for a description of the resource, such as "read and execute access to directory."
- The operating system permissions column contains the operating system flags used to implement the security access. For Unix systems, these are the read(r), write(w), execute(x) flags. Other operating systems will use their own sets of flags.

You also should examine the procedure that you will be using to create user accounts and assign permissions. In the least restrictive case, the people who are authorized to grant access would be able to go into the system directly and create an account by hand or through vendor-supplied mechanisms. These mechanisms place a great deal of trust in the person running them, and the person running them usually has a large amount of privileges, such as the root user in Unix. Under these circumstances, you need to select someone who is trustworthy to perform this task.

You should develop specific procedures for the creation of accounts. Under Unix, there are several methods that can be used for creating accounts. Regardless of which procedure you decide to use, it should be well documented to prevent confusion and reduce mistakes.

TIP

Security vulnerabilities can easily occur as a result of mistakes made by the system administrator. If you have well-documented procedures, this will help ensure fewer mistakes.

Worksheet for Granting Access to System/Network Resources

Network Resources		Type of Access	Operating System Permission Unix: rwx
Number	Name		

FIGURE 3.4 *A sample worksheet for granting access for system/network resources.*

These procedures also enable future system administrators to be easily trained on the peculiarities of a particular system. Another issue to consider is to select a user account creation procedure that is the simplest and most easily understood. This ensures that fewer mistakes will be made, and that the system administrators are most likely to follow them (as they normally should).

You also should have a policy on the selection of an initial password. The granting of an initial password is a vulnerable time for the user account. Policies such as the initial password being the same as the user name, or being left blank, can leave the accounts wide open. Also, avoid setting the initial password as a function of the user name, or part of the user name, or some algorithmically generated password that can easily be guessed. The selection of an initial password should not be obvious.

> CERT estimates that 80 percent of all network security problems are created by insecure passwords.

NOTE

Some users do not use their account for a considerable period of time after their account is created; others never log in. In these circumstances, if the initial password is not secure, the account and the system are vulnerable. For this reason, you should have a policy for disabling accounts that have never been accessed for a period of time. The user is then forced into asking for enabling of their user account.

It is also a mistake to allow users to continue to use the initial password indefinitely. If the system permits, you should force users to change their password on their first login. Many systems have a password aging policy. This can be helpful in protecting the passwords. There are also Unix utilities such as password+ and npasswd that can be used to test the security of the password.

> passwdt is an application for analyzing passwords. It can be found at `ftp://ftp.dartmouth.edu/pub/security/passwd+.tar`.
>
> npasswd is plug-compatible replacement for the passwd command. It incorporates a password-checking system that disallows simple passwords. npasswd can be found at `ftp://ftp.uga.edu/pub/security/npasswd.tar.gz`.

NOTE

Determining User Responsibilities

The network security policy should define the users rights and responsibilities for using the network resources and services. The following is a list of issues that you may want to address concerning user responsibilities:

- Guidelines regarding network resource usage such as whether users are restricted and what the restrictions are.
- What constitutes abuse in terms of using network resources and affecting system and network performance.
- Are users permitted to share accounts or let others use their accounts?
- Should users reveal their passwords on a temporary basis to allow others working on a project to access their accounts?
- User password policy—How frequently should users change their passwords and any other password restrictions or requirements?
- Are users responsible for providing backups of their data, or is this the system administrator's responsibilities?
- Consequences for users disclosing information that may be proprietary. What legal action or other punishment may be implemented?
- Statement on electronic mail privacy (Electronic Communications Privacy Act).
- A policy concerning controversial mailing or postings to mailing lists or discussion groups.
- A policy on electronic communications such as mail forging.

The Electronic Mail Association (EMA) recommends that every site have a policy on the protection of employee privacy. Organizations should establish privacy policies that are not limited to electronic mail, but which encompass other media such as disks, tapes, and paper documents. The EMA suggests five criteria for evaluating any policy:

1. Does the policy comply with law and with duties to third parties?
2. Does the policy unnecessarily compromise the interest of the employee, the employer, or third parties?
3. Is the policy workable as a practical matter and likely to be enforced?
4. Does the policy deal appropriately with all different forms of communications and record keeping with the office?
5. Has the policy been announced in advance and agreed to by all concerned?

DETERMINING THE RESPONSIBILITIES OF SYSTEM ADMINISTRATORS

The system administrator often needs to gather information from files in users' private directories to diagnose system problems. The users, on the other hand, have a right to maintain their privacy. There is, therefore, a tradeoff between a user's right to privacy and the needs of the system administrators. When threats to the network security occur, the system administrator may have greater need to gather information on files on the system, including the users' home directories.

The network security policy should specify the extent to which system administrators can examine users' private directories and files for diagnosing system problems and for investigating security violations. If the network security is at risk, the policy should allow a greater flexibility for the system administrators to correct the security problems. Other related questions that you should address are the following:

- Can the system administrator monitor or read a user's files for any reason?
- Do network administrators have the right to examine network or host traffic?
- What are the users', system administrators', and organization's liabilities for gaining unauthorized access to other people's private data?

WHAT TO DO WITH SENSITIVE INFORMATION

You must determine what types of sensitive data can be stored on a specific system. From a security standpoint, extremely sensitive data such as payroll and future plans should be restricted to few hosts and few system administrators. Before granting users access to a service on a host, you should consider what other services and information are provided that a user can gain access to. If the user has no need to deal with sensitive data, the user should not have an account on a system containing such material.

You also should consider if adequate security exists on the system to protect sensitive data. In general, you do not want users to store very sensitive information on a system that you do not plan to secure very well. On the other hand, securing a system can involve additional hardware, software, and system administration cost, and it might not be cost-effective to secure data on a host that is not very important to the organization or the users.

The policy also should take into account the fact that you need to tell users who might store sensitive information what services are appropriate for the storage of this sensitive information.

PLAN OF ACTION WHEN SECURITY POLICY IS VIOLATED

Each time the security policy is violated, the system is open to security threats. If no change to the network security occurs when the security policy is violated, then the security policy should be modified to remove those elements that do not have security.

TIP

> A security policy and its implementation should be as non-obtrusive as possible. If the security policy is too restrictive, or improperly explained, it is quite likely to be violated.

Regardless of what type of policy is implemented, there is a tendency for some users to violate the security policy. Sometimes it is obvious when a security policy is broken; other times these infractions can go undetected. The security procedures that you implement should minimize the possibility of a security infraction going undetected.

When you detect a security policy violation, you should classify if the violation occurred due to an individual's negligence, an accident or mistake, ignorance about the current policy, or deliberate ignoring of the policy. In the latter case, the violation may be performed not by just one individual, but by a group of individuals who knowingly perform an act that is in direct violation of the security policy. In each of these circumstances, the security policy should provide guidelines on the immediate course of action.

NOTE

> An investigation should be performed to determine the circumstances surrounding the security violation, and how or why the violation occurred. The security policy should provide guidelines on the corrective action for a security breach. It is reasonable to expect that the type and severity of action will depend on the severity of the violation.

RESPONSE TO POLICY VIOLATIONS

When a violation takes place, the response can depend on the type of user responsible for the violation. Violations to policy may be committed by a wide variety of users. Some of these users may be local users and others may be external users. Local users are often called insiders and external users and called outsiders. The distinction between the two is usually based on network, administrative, legal, or political boundaries. The type of boundary determines what the response should be to the security violation. Examples of responses can range from a verbal reprimand or warning, a formal letter, or the pressing of legal charges.

You need to define actions based on the type of violation. These actions need to be clearly defined, based on the kind of user violating your computer security policy. The internal and external users of your network should be aware of your security policy. If you have outsiders who are using your computer network legally, it is your responsibility to verify that these individuals have a general awareness of the policies that you have set. This is particularly important if you need to take legal action against the offending parties. If a significant amount of loss was incurred, you may want to take more drastic action. If adverse publicity is involved, you may prefer to fix the security hole and not pursue legal action.

The security policy document also should include procedures for handling each security violation incident. A proper log of such security violations should be maintained, and should be periodically reviewed to observe trends and perhaps adjust security policy to take into account any new types of threats.

RESPONSE TO POLICY VIOLATIONS BY LOCAL USERS

You could have a security policy violation in which an internal user violates the security policy. This could occur in the following situations:

- A local user violates the local site security policy.
- A local user violates a remote site security policy.

In the first case, because your internal security policy is violated, you may have more control over the type of response to this security violation. In the second case, a local user has violated another organization's security policy. This could happen through a connection such as the Internet. This situation is complicated by the fact that another organization is involved, and any response you take will have to be discussed with the organization whose security policy your local user has violated. You also should consult your corporate attorneys or attorneys who specialize in computer legal security.

RESPONSE STRATEGIES

There are two types of response strategies to security incidents:

- Protect and Proceed
- Pursue and Prosecute

If the administrators of the security policy feel that the company is sufficiently vulnerable, they may choose the Protect and Proceed strategy. The goal of this policy is to immediately protect the network and restore it to its normal status so that the users can continue using the network. In order to do this, you may have to actively interfere with the intruder's actions and prevent further access. This should be followed with an analysis of the amount of damage done.

Sometimes it may not be possible to immediately restore the network to its normal operation; you may have to isolate network segments and shut down systems, with the objective of preventing further unauthorized access to the system. A disadvantage to this approach is that the intruders know that they have been detected and will take actions to avoid being traced. Also, the intruder could react to your protection strategy by attacking the site using a different strategy; at the very least, the intruder likely will live to pursue their hacking against another site.

The second approach—Pursue and Prosecute—adopts the strategy that the major goal is to allow the intruders to pursue their actions while monitoring their activities. This should be done as unobtrusively as possible so that the intruders are not aware of the fact that they are being monitored. The intruders' activities should be logged, so proof will be available in the Prosecute phase of the strategy. This approach is the one recommended by law enforcement agencies and prosecutors, because it yields proof these agencies can use in prosecuting the intruders. The disadvantage of this approach is that the intruder will continue stealing information or doing other damage, and you are still vulnerable to lawsuits resulting from damage to the system and loss of information.

One possible way of monitoring the intruders without causing damage to the system is to construct a "jail." A *jail*, in this case, defines a simulated environment for the intruders to use, so that their activities can be monitored. The simulated environment presents fake data, but the system is set up in such a way that the intruders' keystrokes and activities are monitored.

Figure 3.5 shows the general idea behind constructing a jail. To construct a jail, you need access to the source code for the operating system and in-house programming talent who can simulate this environment. It is safest to construct the jail using a sacrificial machine on an isolated network segment to minimize the risk of pollution of other network segments and systems by the intruders' activities. It is also possible to construct the jail using a software-simulated environment; however this approach is more difficult to set up.

On a Unix system the chroot mechanism can be very handy in setting up a jail. The chroot mechanism irrevocably confines a process to a single branch of the file system. For all practical purposes, the root of this branch of the file system appears as the root of the file system to the process. This mechanism prevents access to device files and the real password file (/etc/passwd).

If you do not want other users logged into the sacrificial machine, you will have to periodically update the utmp file that contains a log of the users logged in so that the Jail looks realistic. You should also remove access to utilities that can reveal that the jail is a simulated environment. Examples of these utilities are netstat, ps, who, w. Alternatively, you can provide fake versions of these utilities to make the simulated environment appear like a real one.

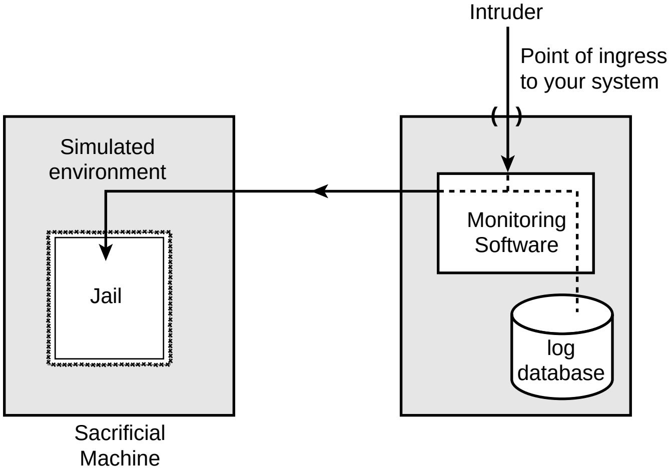


FIGURE 3.5

The general architecture of a jail .

Once you have sufficient evidence against the intruder, you may want to prosecute. However, prosecution is not always the best possible outcome. If the intruder is an internal user or a guest user such as a student, proper disciplinary actions can be equally effective without the additional cost of legal prosecution and the ensuing publicity. The network security policy should list these choices and provide guidelines on when they should be exercised.

> The following can be used as a guideline to help determine when the site should use a policy of Protect and Proceed or Pursue and Prosecute.

NOTE

The Protect and Proceed strategy can be used under the following conditions:

- If network resources are not well-protected against intruders.
- If continued intruder activity could result in great damage and financial risk.
- If the cost of prosecution is deemed too costly, or the possibility or willingness to prosecute is not present.
- If there is considerable risk to the existing users on the network.
- If the types of users for a large internal network are not known at the time of attack.
- If the site is vulnerable to lawsuits from users. This is true for insurance companies, banks, security forms, network providers, and so on.

The Pursue and Prosecute strategy can be used under the following conditions:

- If network resources and systems are well protected.
- If the risk to the network is outweighed by the disruption caused by present and potentially future intrusions.
- If this is a concentrated attack and has occurred before.
- If the site is highly visible and has been a target of past attacks.
- If not pursuing and prosecuting will invite further intrusions.
- If the site is willing to incur the risk to network resources by allowing the intruder to continue.
- If intruder access can be controlled.
- If the monitoring tools are sufficiently well-developed to create proper logs and gather evidence for prosecution.
- If you have in-house programming talent to construct specialized tools quickly.
- If the programmers, system and network administrators are sufficiently clever and knowledgeable about the operating system, system utilities, and systems to make the pursuit worthwhile.
- If there is willingness on the part of management to prosecute.
- If the system administrators know what kind of evidence would lead to prosecution, and can create proper logs of the intruders activities.
- If there is established contact with knowledgeable law enforcement.
- If there is a site representative versed in the relevant legal issues.
- If the site is prepared for possible legal action from its own users if their data or systems become compromised during the pursuit.
- If good backups are available.

DEFINING RESPONSIBILITIES OF BEING A GOOD CITIZEN ON THE INTERNET

The Internet is a cooperative venture, and the sites that have networks connected to it should be expected to follow rules of good behavior to other sites. This is similar to the workings of a successful modern society. Your security policy should include a statement that deliberate attempts at violating another site's networks is a violation of the company policy.

You should also define what types of information should be released. It may be more economical for you to publish documents about your organization on an FTP server. In this case, you should decide what type of information and how much should be released.

CONTACTS AND RESPONSIBILITIES TO EXTERNAL ORGANIZATIONS

The network security policy should define procedures for interacting with external organizations. External organizations could include law enforcement agencies, legal experts, other sites affected by the security violation incident, external response team organizations such as the CERT (Computer Emergency Response Team), CIAC (Computer Incident Advisory Capability), and if necessary, press agencies.

The people who are authorized to contact these organizations should be identified. You should identify more than one person for each area, to cover situations when the designated individuals may not be reachable. Issues that you should address can include the following:

- Identify “public relation types” who are versed in talking to the press.
- When should you contact local and federal law enforcement and investigative agencies?
- What type of information can be released?

During an investigation, certain rules regarding evidence handling must be followed. Failure to follow these rules might result in the loss of your case. Consequently, it is essential that you contact the authorities in your country responsible for computer crime and have them assist you in planning your investigation. Many law-enforcement agencies offer training in evidence-handling procedures, and there are companies that offer services in the investigation arena.

INTERPRETING AND PUBLICIZING THE SECURITY POLICY

It is important to identify the individuals who will interpret the policy. It is usually not a good idea to have only one individual involved, in case that person would be unreachable at the time of crisis. One can identify a committee, but it is also not a good idea to have too many members of such a committee. From time to time, the security policy committee will be called upon to interpret, review, and revise the document.

Once the site security policy has been written and agreed upon, the site must ensure that the policy statement is widely disseminated and discussed. Mailing lists can be used. The new policy can also be reinforced by internal education such as training seminars, group briefings, workshops, one-on-one meetings with administrators, or all of these depending on the size of the institution and the needs at hand.

Implementing an effective security policy is a collective effort. Therefore, the network users should be allowed to comment on the policy for a period of time. You may want to hold meetings to elicit comments and to ensure that the policy is correctly understood. This also will help you in clarifying the language of the policy and avoid ambiguities and inconsistencies in the policy.

The meetings should be open to all network users and higher-level management who may be needed to make global decisions when important questions come up. User participation and interest further ensures that the policy will be better understood and more likely to be followed.

TIP

> If users perceive that the policy reduces their productivity, they should be allowed to respond. If necessary, additional resources may have to be added to the network to ensure that the users can continue performing their jobs without loss in productivity. To create an effective network policy, you need to find a fine balance between protection and productivity.

Sometimes new programs are met with enthusiasm initially, when everyone is aware of the policy. Over a period of time, though, there is a tendency to forget the contents of the policy. Users need periodic reminders. Also, as new users are added to the network, they need to understand the security policy.

Periodic reminders (properly timed) and continual education on the policy, will increase the chances that the users will follow the security policy. New users should have the policy included as part of their network user information packet. Some organizations require a signed statement from every network user that they have read and understood the policy. If users later require legal action for serious security violations, the signed statement can help you pursue the legal action successfully.

IDENTIFYING AND PREVENTING SECURITY PROBLEMS

The security policy defines what needs to be protected, but the policy may not explicitly spell out how the resources should be protected and the general approach to handling security problems. A separate section on the security policy should discuss the general procedures to be implemented to prevent security problems. The security policy may refer to the site's system administrator's guide for additional details on implementing the security procedures.

Before establishing the security procedures, you must assess the level of importance of network resources and their degree of security risk. The earlier discussion in this chapter on "Approach to Security Policy" and "Risk Analysis," and the worksheets in figures 3.1 and 3.3 can be used as a guide in focusing attention on the more important network resources.

NOTE

> It is often tempting to start implementing procedures such as the following without first defining the network security policy:
>
> "Our site needs to provide users with telnet access to internal hosts and external hosts, prevent NFS access to internal hosts but deny this to external users, have smart cards for logging in from the outside, have dial-back modems ..."
>
> Without a proper understanding of the most important resources and those that are at the greatest risk, the previous approach could lead to some areas having more protection than they need, and other more important areas that do not have sufficient protection.

Developing an effective security policy requires considerable effort. It takes a determined effort to consider all the issues and a willingness to set the policies in paper and do what it takes to see that the policy is properly understood by network users.

Besides performing a risk analysis of the network resources, you must identify other vulnerabilities. The following list is an attempt to describe some of the more common problem areas. This list can point you in the right direction, but is by no means complete, because your site is likely to have a few unique vulnerabilities.

- Access points
- Improperly configured systems
- Software bugs
- Insider threats
- Physical security

A discussion of these issues is next.

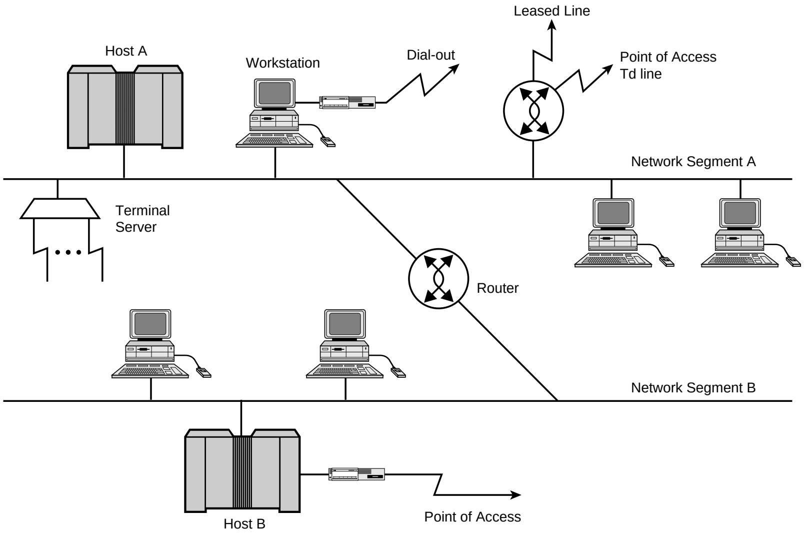
ACCESS POINTS

Access points are points of entry (also called *ingress*) by unauthorized users. Having many access points will increase the security risks to the network.

Figure 3.6 shows a simplified network for an organization in which there exists several points of ingress to the network. The access points are the terminal server and router on network segment A. The workstation on segment A has a private modem which is used for dial-out connections. Host B on network segment B is also an access point to this network segment. Because a router joins the two network segments, an intruder can use any of the access points on each network segment to reach the entire network.

FIGURE 3.6

Identifying access points in a network.



You may secure the listed access points in figure 3.6, but it is easy to forget the workstation on segment A which was going to be used for dial-out connections, perhaps to simple computer bulletin boards.

Consider the following situation: The user of the workstation on segment A may have an account with an Internet Access Provider. Suppose this user uses a SLIP (Serial Line Interface Protocol) or PPP connection to access this Internet Access Provider. If the TCP/IP software that the user is running on the workstation is also configured as a router, it is possible for an intruder to access the entire network. Also, if a routing protocol such as RIP (Routing Information Protocol) or OSPF (Open Shortest Path First) is enabled at the workstation, the workstation can expose the internal network to routing protocol-based attacks.

Note that the user may not deliberately enable the workstation as a router. The workstation operating system might be by default enabled as a router. This is true for many Unix systems as well as DOS/Windows TCP/IP packages. Even the workstation is properly configured by the networking staff, the user could plug in their laptop computer to the network and use a modem to dial out to the Internet Access Provider. If the user was using a dial-up (also called a shell account) where the user is running terminal emulation software at the workstation and not TCP/IP software, perhaps no harm can be done. However, if the user is using SLIP or PPP connection, the user has inadvertently created another access point which the network administration staff may not be aware of. This could represent a security risk to the entire network.

The situation in figure 3.6 can be prevented if the network security policy informs the user that private connections through individual workstations are prohibited. This situation also underscores the importance of having a network security policy that clearly delineates the AUP for the network.

If you want to connect to the Internet, you must have at least one network link to networks outside the organization. A network link can make available a large number of network services, both inside the network and outside the network, and each service has a potential to be compromised.

NOTE

Terminal servers can represent a security risk if not properly protected. Many terminal servers on the market do not require any kind of authentication. Check with your vendor about authentication capabilities of your terminal server. Intruders can use terminal servers to disguise their actions, dialing in to the terminal server and accessing your internal network.

If the terminal server permits this, an intruder can access the internal network from the terminal server, and then use telnet to go out again, making it difficult to trace them. Also, if the intruder uses this to attack another network, it may appear that the attack originated from your network.

Dialup lines, depending on their configuration, may provide access merely to a login port of a single system. If connected to a terminal server, the dialup line may give access to the entire network. As mentioned in the discussion in figure 3.6, a dialup line at a workstation running TCP/IP software can give access to the entire network.

IMPROPERLY CONFIGURED SYSTEMS

If intruders penetrate the network, they usually try to subvert the hosts on the system. Hosts that act as telnet servers are popular targets. If the host is improperly configured, the system can be easily subverted. Misconfigured systems account for a large number of security problems on networks.

Modern operating systems and their associated software have become so complex that understanding how the system works is not only a full-time job, but often requires specialized knowledge. Vendors may also be responsible for misconfigured systems. Many vendors ship systems with wide-open security. Passwords to critical accounts may not be set, or they use easily guessable login name and password combinations. The book *The Cuckoo's Egg,* by Cliff Stoll, which tells the true story of the global hunt for a computer spy, mentions how an intruder obtained access to systems by using login name/password combinations such as "system/manager," "field/service," and so on.

SOFTWARE BUGS

As the complexity of software increases, so does the number and complexity of bugs in any given system. Perhaps software will never be bug free unless revolutionary ways of creating software are developed. Publicly known security bugs are common methods of unauthorized entry. If a system's implementation is open and widely known (such as with Unix), an intruder can use weaknesses in the software code that runs in a privileged mode to gain privileged access to the system.

The system administrators must be aware of security weaknesses in their operating systems and be responsible for obtaining updates and implementing fixes when these problems are discovered. You also should have a policy to report bugs (when found) to the vendor so that a solution to the problem can be implemented and distributed.

INSIDER THREATS

Insiders usually have more direct access to the computer and network software than to the actual hardware. If an insider decides to subvert the network, he can represent a considerable threat to the security of the network. If you have physical access to the components of a system, the system is easier to subvert. For example, many workstations easily can be manipulated to grant privileged access. One can easily run protocol decode and capture software to analyze protocol traffic. Most standard TCP/IP application services such as telnet, rlogin, and ftp have very weak authentication mechanisms where passwords are sent in the clear. Access to these services using privileged accounts should be avoided because it can easily compromise the passwords for these accounts.

PHYSICAL SECURITY

If the computer itself is not physically secure, software security mechanisms easily can be bypassed. In the case of DOS/Windows workstations, there is not even a password level of protection. If a Unix workstation is left unattended, its physical disks can be swapped out, or if the workstation is left in a privileged mode, the workstation is wide open. Alternatively, the intruder can halt the machine and bring it back up in privileged mode, and then plant Trojan-horse programs or take any number of actions that can leave the system wide open to future attacks.

All critical network resources such as backbones, communications links, hosts, important servers, and key machines should be located in physically secure areas. The Kerberos authentication mechanism, for example, requires that the Kerberos server be physically secure. *Physically secure* means that the machine is locked in a room or placed in such a manner that restricts physical access to the data on the machine.

Sometimes it is not always easy to physically secure machines. In this case care should be taken not to trust those machines too much. You should limit access from non-secure machines to more secure machines. In particular, you should not allow access to hosts using trusted access mechanisms such as the Berkeley-r* utilities (rsh, rlogin, rcp, rwho, ruptime, rexec).

Even when the machine is physically secure, care should be taken about who has access to these machines. Using electronic "smart" cards to access the room in which the machines are secured can limit the number of people with access and also provide a log of the identity and time-of-day individuals accessed the room. You should also have a policy for your employees that other persons cannot tag along when the door to the secure room is opened, even when the identity of the individual is known. If you allow people to tag along, you will not have a proper log of who entered the room and when.

NOTE

Also remember that maintenance and building staff may have access to the secure rooms. Make sure you take this into account when designing your system security.

CONFIDENTIALITY

Confidentiality can be defined as the act of keeping things hidden or secret. This is an important consideration for many types of sensitive data.

Some of the situations in which information is vulnerable to disclosure are the following:

- When the information is stored on a computer system
- When the information is in transit to another system on the network
- When the information is stored on backup tapes

Access to information that is stored on a computer is controlled by file permissions, access control lists (ACLs), and other similar mechanisms. Information that is in transit can be protected by encryption or by firewall gateways. Encryption can be used to protect all three situations. Access to information stored on tapes can be controlled by physical security such as by locking them in a safe, or an inaccessible area.

Implementing Cost-Effective Policy Controls

Controls and protection mechanisms should be selected so that they can adequately counter the threats found during risk assessment. These controls should be implemented in a cost-effective manner. It makes little sense to spend large sums of money, and over-protect and restrict the use of a resource if the risk of exposure is very small.

Common sense is often a very effective tool to use to establish your security policy. While elaborate security schemes and mechanisms are impressive, they can be quite expensive. Sometimes the costs of these implementations are hidden. For example, you might implement a freely available software security solution but not take into account the cost of administering such a system and keeping it updated. Also if the security solution is very elaborate, it may be difficult to implement and administer. If the administration is a one-time installation experience, the commands to administer such a system can be easily forgotten.

TIP

You should also maintain a sense of perspective that no matter how elaborate is the solution, a weak password or stolen password can compromise the system.

The following are some guidelines for implementing cost-effective policy controls.

Selecting the Policy Control

The controls that you select are the first line of defense in the protection of your network. These controls should accurately represent what you intend to protect as outlined in the security policy. If a major threat to your system is external intrusions, it may not be cost-effective to use biometric devices to authenticate your internal users. If the major threat to your systems is unauthorized use of computing resources by internal users, you may want to establish good automated accounting procedures. If the major threat to your network is external users, you may want to build screening routers and firewall solutions.

USING FALLBACK STRATEGIES

If the risk analysis indicates that protecting a resource is critical to the security of the network, you may want to use multiple strategies to protect your network. Using multiple strategies gives you the assurance that if one strategy fails or is subverted, another strategy can come into play and continue protecting the network resource.

It might be more cost-effective and simpler to use several simple-to-implement, but nevertheless effective, strategies rather than use a single elaborate and sophisticated strategy. The latter is an all-or-nothing approach. If the elaborate mechanism is circumvented, there is no fallback mechanism to protect the network resource.

> Examples of simpler controls are dial-back modems that can be used in conjunction with traditional logon mechanisms. These can be augmented with smart cards or one-time hand-held authenticators.

DETECTING AND MONITORING UNAUTHORIZED ACTIVITY

If a break-in or attempted break-in takes place, it should be detected as soon as possible. You can implement several simple procedures to detect unauthorized uses of a computer system. Some procedures rely on tools provided with the operating system by the vendor. There are also tools that are publicly available on the Internet.

MONITORING SYSTEM USE

System monitoring can be done periodically by the system administrator. Alternatively, software written for the purposes of monitoring the system can be used. Monitoring a system involves looking at several parts of the system and searching for anything unusual. A few of the ways this can be done are outlined in this section.

Monitoring must be done on a regular basis. It is not sufficient to do this on a monthly or weekly basis, because this could leave a security breach undetected for a long time. Some security breaches may be detected a few hours after the fact by other means, in which case monitoring on a weekly or monthly basis will do little good. The goal of monitoring is to detect the security breach in a timely manner so that you can respond to it appropriately.

If you are using monitoring tools, you should regularly examine the output of these tools. If the logs are voluminous, you may want to use awk or perl scripts to analyze the output. These tools are also available for non-Unix systems.

MONITORING MECHANISMS

Many operating systems store information about logins in special log files. The system administrator should examine these log files on a regular basis to detect unauthorized use of the system. The following is a list of methods that can be used at your site.

- You can compare lists of currently logged-in users with past login histories. Most network users have regular working hours and log in and log out at about the same time every day. An account that shows login activity outside the "normal" hours for the user should be closely monitored. Perhaps this account is in use by an intruder. Users can also be alerted to watch for the last login message that appears when they first log in. If they see some unusual times, they should alert the system administrator.
- Many operating systems can use accounting records for billing purposes. These records can also be examined for any unusual usage patterns for the system. Such unusual accounting records may indicate illegal penetration of the system.
- The operating system may have system logging facilities, such as the syslog used in Unix. The logs produced by such tools should be checked for unusual error messages from system software. For instance, a large number of failed login attempts in a short period of time may indicate someone trying to guess passwords. You should also monitor the number of attempts at logging into sensitive accounts such as root, sysadm, and so on.
- Many operating systems have commands, such as the ps command under Unix, to list currently executing processes. These can be used to detect users running programs they are not authorized to use, as well as to detect unauthorized programs which have been started by an intruder.
- Firewall gateways can be used to produce a log of the network access. These should be monitored regularly. Firewalls are discussed in detail later in the book.

- If you have special resources that you want to monitor, you can construct your own monitoring tools using standard operating system utilities. For example, you can combine the Unix ls and find commands in a shell script to check for privileged file ownerships and permission settings. You can store the output of your monitoring activity in lists that can be compared and analyzed using ordinary Unix tools such as diff, awk, or perl. Differences in file permissions of critical files can indicate unauthorized modifications to the system.

MONITORING SCHEDULE

Regular and frequent monitoring should be done by system administrators throughout the day. If the monitoring is done at fixed times, it can become very irksome, but monitoring commands can be run at any time of the day during idle moments, such as when you're conducting business over the phone.

By running monitoring commands frequently, you quickly will become aware of the normal output of the monitoring tools. This can help in detecting unusual monitoring outputs. One can try to automate this process by running search tools on the output, and one can look for certain set patterns, but it is generally difficult to anticipate all the unusual outputs caused by an intrusion into the system. The human brain is still better than most programs in detecting subtle differences in the monitor log.

If you run various monitoring commands at different times throughout the day, it is difficult for an intruder to predict your actions. The intruder cannot guess when a system administrator might run the monitor command to display logged in users, and thus runs a greater risk of detection. On the other hand, if the intruder knows that at 6:00 p.m. daily the system is checked to see that everyone has logged off, they will wait for the system check to be completed before logging in.

WARNING

Monitoring is useful, but the monitoring process can itself be subverted. Some intruders may be aware of the standard logging mechanisms in use on your system, and may attempt to disable these monitoring mechanisms. Regular monitoring can detect intruders, but does not provide any guarantee that your system is secure. It is not an infallible method of detecting intruders.

REPORTING PROCEDURES

In the event that unauthorized access is detected, you should have procedures on how this access will be reported and to whom it will be reported. In addition, your security policy should cover the following:

- Account management procedures
- Configuration management procedures
- Recovery procedures
- Problem reporting procedures for system administrators

ACCOUNT MANAGEMENT PROCEDURES

When creating user accounts, you must exercise care that you do not leave any security holes. If the operating system is being installed from the distribution media, the password file should be examined for privileged accounts that you do not need.

WARNING

> Some operating system vendors provide accounts for their field service and system services engineers. These accounts either have no password or they may be common knowledge. If you need these accounts, you should give them new passwords, or else disable or delete them. There is generally no good reason to allow accounts that do not have a password set.

Accounts without passwords are dangerous even if they do not execute a command interpreter, such as accounts that exist only to see who is logged in to the system. If these are not set up correctly, system security can be compromised. For example, if the anonymous user account used by FTP (File Transfer Protocol) is not set up correctly, you could allow any users to access your system to retrieve files. If mistakes are made in setting up this account and write access to the file system is inadvertently granted, an intruder can change the password file or destroy the system.

TIP

> Some operating systems such as Unix System V provide a special /etc/shadow password file that is used for storing passwords. This file is accessible only to privileged users. If your system supports this facility, you should use it.

The shadow password facility was first introduced with System V, but other Unix systems such as SunOS 4.0 and above, and 4.3BSD Unix Tahoe, provide this feature. The shadow password file permits the encrypted form of the passwords to be hidden from non-privileged users. The intruder, therefore, cannot copy the password file and attempt to guess passwords.

Your policy also should include procedures for keeping track of who has privileged user accounts, such as root on Unix, and MAINT under VMS. Under Unix, if you know the root password you can use the su command to assume root privileges. If the password is inadvertently discovered, the user can log in with his own personal accounts and assume root privileges. You must therefore implement a policy that forces change of passwords for privileged user accounts at periodic intervals.

WARNING

> Also, when a privileged user leaves the organization, you should be alerted and the passwords for the privileged accounts should be changed. In addition the user accounts for those who have left the company should be changed.

Network services should undergo a close scrutiny. Many vendors provide default network permission files which implies that all outside hosts are to be trusted. This is not the case when connected to a network such as the Internet.

Intruders themselves collect information on the vulnerabilities of particular system versions. Sometimes they circulate their findings in underground magazines such as the following:

2600 Magazine
Phrack
Computer Underground Digest

Some system administrators subscribe to these journals to keep abreast of the intruders.

CONFIGURATION MANAGEMENT PROCEDURES

You should keep updated versions of the operating system and critical utilities. The security weaknesses of older systems are usually well known, and it is likely that the intruder is aware of the security problems. Unfortunately, new releases of software, while fixing old security problems, often introduce new ones. For this reason, it is important to weigh the risks of not upgrading to a new operating system release and leaving security holes unplugged, against the cost of upgrading to the new software.

NOTE

> Although most vendors can be trusted when shipping updates, many organizations rely on the wealth of publicly developed software for some activity within their company. Many software projects are shipping their software with PGP or other digital signatures to signify that the software has not been tampered with.
>
> Tripwire is a tool that aids system administrators and users in monitoring a designated set of files for any changes. Used with system files on a regular (that is, daily) basis, Tripwire can notify system administrators of corrupted or tampered files so damage control measures can be taken in a timely manner.
>
> Tripwire can be found at URL `ftp://ftp/nordu.net/networking/security/tools/tripwire/tripwire-1.2.tar.gz`.

Generally, most vendors can be trusted so that the new releases of software are more likely to fix old problems and not create bigger security problems. Another complication is that the new release may break existing application software that your users depend upon. You may have to coordinate your upgrade efforts with more than one vendor.

You also can receive fixes through network mailing lists. You must have competent personnel who can examine these bug fixes carefully and only implement them if they are safe.

TIP

> As a rule, you should not install a bug fix unless you know the consequences of a fix. It is always possible that the authors of the fix may have non-obvious code to allow them unauthorized access to your system.

RECOVERY PROCEDURES

Whenever you install a new version of the operating system, you should not only take a backup of the binary image of the operating system kernel but also the files that are used to compile and configure the operating system. The same also applies to all other applications and networking software.

File system backups are like an insurance policy. They not only protect you in the event of disk or other hardware failures, but also against accidental deletions and as a fallback measure if your system has been penetrated. If you suspect your system has been broken into, you might have to restore the system from a backup to protect yourself against changes made by the intruder. If you cannot detect when the unauthorized changes took place, you have to examine several backups. If you do not have a good copy of your system software, it is hard to determine what your system data and files are supposed to be.

Daily backups, such as incremental backups, can be useful in providing a history of the intruder's activities. By examining older backups, you can determine when the system was first penetrated. Even though the intruder's files have been deleted, you can see them on the backup tapes.

> When examining traces for intruders files, you should check for file names that normally would not show up in a directory listing. On Unix systems, some intruders like to save data in files beginning with a period (.), or files containing non-displayable characters. These files are harder to detect.

TIP

You must decide on a backup strategy. Backup strategies usually involve the combination of the following methods:

- Full backup
- Level 1 backup
- Level 2 backup
- Custom backup

In Unix systems a full backup is also called a level 0 backup. In Unix systems, a *level 1 backup* backs up all files that have been modified since the last level 0 backup. In general, a level *N backup* backs up all files modified since the last *N-1 backup.* In the case of backup utilities such as dump, a level *N backup* backs up all files modified since the last *N-1 backup* or lower.

One can use an arbitrary number of levels, but generally this does not make sense because it becomes difficult to keep track of the backups. The numeric backup levels are supported in BSD-style backup commands from levels 0 to 9, but the concept can be used on any system, and you may have to do some manual bookkeeping. On BSD Unix the backup program is dump, and the files that were backed up at a specific level are kept in the /etc/dumpdates file.

In *full backup* (level 0), all data is backed up, regardless of when it was last modified, or whether it has not been modified at all. An example of this is all directories and files in a file system. After the data is backed up, the archive bit is cleared for all files that are backed up.

NOTE

> The full backup strategy is the most comprehensive of all backup strategies, because it backs up all files regardless of the fact that they have been modified since the last back up or not. Because of the large volume of data that may need to get backed up, however, it is the slowest of the backup strategies.

A level 1 backup backs up all files that have been modified since the last full backup (level 0). This means that all files that were backed up in the first level 1 backup are also backed up in the second level 1 backup, together with any files that have been modified since the first level 1 backup. This process continues with each level 1 backup, and more files can be expected to be backed up with each level 1 backup.

There is an unfortunate confusion of terms to describe the level 1 backup. On Unix systems, the level 1 backup is called an *incremental* backup. On many non-Unix systems (DOS/Windows/PC LAN operations systems), the level 1 backup is called a *differential* backup. The term incremental backup on many non-Unix systems means something entirely different. To avoid confusion, your policy must state which definition you are using.

To obtain a complete record of the most updated versions of the files, you would have to start with the most recent full backup (level 1), and add to it the files in the most recent level 1 backup. That is,

$$\text{Most Recent Backup} = \text{Last Full Backup} + \Delta d$$

where Δd is the most recent level 1 backup.

Because the last level 1 backup contains all files that have been modified since the last full backup, you can restore data with just two tape backup sets: the backup set for the full backup and the backup set for the last level 1 backup.

If the data on one of the last level 1 backups is corrupt, you have to fall back on the next-to-the-last differential backup. On the other hand, if any data in another level 1 backup tape is corrupt, it does not matter as long as the data in the most recent differential backup is good.

If a full backup has not been done for some time, and there have been many changes to the file, the size of the data that needs to be backed up tends to grow, with each level 1 backup. If all files have been modified, the level 1 backup session is the same as the full backup sessions. This tends not to be the case, because most file systems contain a mix of programs and data, and program files are not usually modified.

In many non-Unix systems the term *incremental backup* is used to describe a backup of all files that have been modified since the last backup (level 0 or level 1). This is like a level 2 backup on Unix systems. Files that have not been modified are not backed up. To obtain a complete record of the most update versions of the files, you would have to start with the most recent full backup and add all the incremental changes recorded in each incremental backup session. That is,

$$\begin{aligned}\text{Most Recent Backup} &= \text{Last Full Backup} + \Delta 1 + \Delta 2 + \Delta n\\ &= \text{Last Full Backup} + \Delta i \ (i = 1 \text{ to } n)\end{aligned}$$

where each Δi is an incremental backup.

The incremental backup contains a sequential history of the files that have been modified. This means that to restore data, you need the last full backup and every incremental backup after it. If the data on one of the backup tapes is corrupt, you might not be able to restore data. The exception to this is situations in which later incremental backups have the files that were inaccessible on the corrupted tape. In this case, you could restore the data from a later tape.

Custom backup gives you complete control over what files to backup or not to backup. You can include or exclude parts of the directory structure to be backed up or select different type of data items to be backed up. Custom backups are useful if you want to selectively back up a few files and directories and not wait for a scheduled backup.

PROBLEM REPORTING PROCEDURES FOR SYSTEM ADMINISTRATORS

Earlier in this chapter, problem reporting procedures for users was discussed. System administrators should have a defined procedure for reporting security problems. In large network installations, this can be done by creating an electronic mailing list that contains the e-mail addresses of all system administrators in the organization. Some organizations set up a response team that provides a hotline service.

PROTECTING NETWORK CONNECTIONS

If the intruder attack is likely to take place through an externally connected network such as the Internet, you may want to protect your connections to the external network.

A firewall device can be used to provide a point of resistance to the entry of flames (intruders) into the network. Besides firewalls, screening routers can be used. These topics are the subject of discussion in the following chapters.

Some organizations' sites need to connect to other sites in the same organization and are prohibited from connecting to external networks. These networks are less susceptible to threats from outside the organizations' network. Unexpected intrusions can still occur through dial-up modems at users' desktop workstations.

An organization may require connections to their other sites through larger networks such as the Internet. If the protocols they are using are different from the Internet protocols, a technique called IP tunneling can be used. These sites are susceptible to external threats.

Many organizations require connections to the Internet because of the services it offers. The security risks of connecting to outside networks must be weighed against the benefits. You should limit the number of access points to the network. Moreover, you should connect to external networks through hosts that do not store sensitive material. Such hosts should also have removed software development tools and other privileged tools that could be used to probe your network. The idea is to provide a degree of isolation or a firewall between your network and the external network. Important services needed by the organization can be kept behind the isolated network segment.

You should seriously consider restricting the access to an external network through a single system. If all access to an external network is provided through a single host, this host acts as a firewall between you and the external network. The firewall system should be strictly controlled and password-protected. External users who need access to your internal network will have to pass through the firewall. The firewall host can properly screen the incoming calls.

NOTE

> The firewall system is not a guarantee against a successful intruder attack. If the intruder succeeds in compromising the security of the firewall, the intruder can gain access to your internal network behind the firewall.

USING ENCRYPTION TO PROTECT THE NETWORK

Encryption can be used to protect data in transit as well as data in storage. Some vendors provide hardware encryption devices that can be used to encrypt and decrypt data on point-to-point connections.

Encryption can be defined as the process of taking information that exists in some readable form and converting it into a form so that it cannot be understood by others.

If the receiver of the encrypted data wants to read the original data, the receiver must convert it back to the original through a process called *decryption*. Decryption is the inverse of the encryption process. In order to perform the decryption, the receiver must be in possession of a special piece of data called the key. The key should be guarded and distributed carefully.

NOTE

> The advantage of using encryption is that, even if other methods of protecting your data (Access Control Lists, file permissions, passwords, and so on) are overcome by an intruder, the data is still meaningless to the intruder.

There are several types of encryption packages in both hardware and software forms. The software encryption packages are available either commercially or as free software. Hardware encryption engines are usually built around dedicated processors and are much faster than the software equivalent. On the other hand, if the intruder has access to hardware, they can build hardware-based decryption schemes that can be used for a brute-force attack on your encrypted information.

Data in transit over a network may be vulnerable to interception. Some sites prefer to encrypt the entire file as a separate step before sending it. This is sometimes called end-to-end encryption. Others prefer encrypting the data dynamically as it reaches the network using hardware encryption engines creating a secure link.

If the entire packet is encrypted before being sent, as in the case of hardware encryption engines, the IP protcol routers that do not understand the encrypted packet will reject it. The Internet routers do not understand encrypted packets and will reject it. If you want to use encryption over the Internet, you must encrypt the data in a separate step and pass it to the application process.

The following is a brief discussion on the different types of encryption methods. In your network security policy, you must specify which, if any, of these encryption techniques should be used.

DATA ENCRYPTION STANDARD (DES)

DES is a very widely used data encryption mechanism. There are many hardware and software implementations of DES. DES transforms plain text information into encrypted data called *ciphertext* by means of a special algorithm and *seed* value called a key. If the key is known to the receiver, it can be used to convert from the ciphertext the original data.

A potential weakness of all encryption systems is the need to remember the key under which any data was encrypted. In this regard, it is similar to the problem of remembering the password. If the key is written and becomes known to an unauthorized party, they can read your original data. If the key is forgotten, then you are unable to recover the original data.

Many systems support a DES command, or utilities and code libraries that can be used for DES.

CRYPT

On Unix systems the crypt command also can be used to encrypt data. The algorithm used by crypt based on the World War II *Enigma* device and is very insecure. Files encrypted with crypt can be decrypted easily by brute-force approach in a matter of a few hours. For this reason, the crypt command should be avoided for sensitive data. It can be used for trivial encryption tasks. Chapter 2, “Security,” discusses the use of the crypt command.

PRIVACY ENHANCED MAIL (PEM)

E-mail is usually sent on the Internet using SMTP (Simple Mail Transfer Protocol). This protocol is very simple and transmits data in the clear. Moreover, it can be used for transmitting ASCII text data only. If you want to send an encrypted message you have to use indirect means. You have to first encrypt the message. This converts the message into a binary file. Because SMTP cannot be used to transmit binary data—it transmits text data only—you have to encode the binary data as text.

A popular way of doing this on the Internet is to use a utility called uuencode. The recipient of the e-mail has to use a utility called uudecode to convert the text message back to the original encrypted binary form. If the recipient knows the key, he can decrypt the message. While it is possible to secure mail by using the method just outlined, it is cumbersome and laborious. Also, there is the problem of distributing the key to the recipients of the message. You should consider whether this should be done through the Internet, or by some other distribution methods.

Another approach that has attracted a great deal of interest is Privacy Enhanced Mail (PEM). PEM provides a means to automatically encrypt e-mail messages before sending. There are no separate procedures one has to invoke to encrypt the mail message. Therefore, even if the mail is intercepted at a mail distribution host, the interceptor cannot read the encrypted mail.

PRETTY GOOD PRIVACY

Pretty Good Privacy, or PGP, was written by Phil Zimmerman to address the issue of public-key, or asymmetric, file encryption and digital signatures. PGP provides a strong form of cryptographic protection not previously available. PGP is used to protect e-mail, files, and digitally signed documents and is available in commercial and non-commercial forms.

Residents of the U.S. and Canada can obtain PGP through MIT: `http://web.mit.edu/network/pgp-form.html`. It also can be found in other places on the Internet.

NOTE

There are a variety of digital signature formats and programs. Some of these programs are publicly available, meaning should you have a problem, you must deal with it on your own or hope someone in the public can help you. An alternative is to use a commercial product that supports file encryption and digital signatures, such as Northern Telecom Secure Networks' Entrust product (URL `http://www.entrudt.com`) or the commercial version of PGP from Viacrypt ((602) 944-0773). The advantage of using a commercial product is that problems can be reported to and resolved by the manufacturer. You can obtain user support and updates to software and documentation as they are released, ensuring that new bugs have not affected the application.

ORIGIN AUTHENTICATION

When an e-mail is received, the header of the e-mail indicates the originator of the message. Most users of Internet mail take it for granted that the header of the e-mail message truly indicates the sender of the message. It is possible, if one is clever enough, to forge the header so that it indicates a message sent from another e-mail address. This is called e-mail address *spoofing*. To prevent this type of forgery, a technique called origin authentication can be used.

Origin authentication provides a means to ascertain that the originator of a message is indeed who he claims to be. You might think of origin authentication as an electronic notary service similar to the human notary public who verifies signatures on legal documents. Origin authentication is commonly implemented by a public key cryptosystem.

A *cryptosystem* uses two keys. The keys are independent in the sense that one key cannot be derived from the other key using any mathematical or algorithmic procedures. One of the keys is a *public key*, which means that it can be easily found out by anyone and there is no attempt made to hide it. The other key is called a *private key* which means that this key is known only to the party who owns the key. The private key must be guarded very carefully.

In a public key cryptosystem, the originator uses a private key to encrypt the message. The recipient uses a public key obtained from the originator of the message to decrypt the message. The public key is used to authenticate that only the originator could have used their private key. There are several public cryptosystems that are available.

NOTE

The most widely known implementation of the public key cryptosystem is the RSA (Rivest Shamir Adleman) system. The Internet standard for privacy enhanced mail makes use of the RSA system.

INFORMATION INTEGRITY

When a file or document is sent on the network, you should have some means of verifying that the file or document has not been altered. This is called information integrity, and it refers to the process of verifying that the information that was sent is complete and unchanged from the last time it was verified. Information integrity is important for military, government, and financial institutions. It may also be important that classified information be undisclosed, whether it is modified or not modified. Information that is maliciously modified can create misunderstandings, confusion, and conflict.

If the information is sent in electronic form across the network, one way of ensuring that the information is not modified is to use *checksums*. Any form of encryption also provides information integrity, because an interceptor would have to first decrypt the message before modifying it.

USING CHECKSUMS

Checksums are a very simple and effective mechanism to verify the integrity of a file. A simple checksum procedure can be used to compute a value for a file and then compare it with the previous value. If the checksums match, the file is probably unchanged. If the checksums do not match, the file has been altered. Many compression and decompression utilities that can be used to conserve disk space and reduce transmission costs for files generate internal checksums to verify their compression/decompression algorithms.

Arithmetic checksums are simple to implement. They are formed by adding up 16-bit or 32-bit elements of a file to arrive at the checksum number. Though simple to implement, arithmetic checksums are weak from a security point of view. A determined attacker can modify and add data to the file so that the arithmetic checksum computes to the correct value.

The CRC (Cyclic Redundancy Checksum), also called the *polynomial checksum*, is more secure than the arithmetic checksum. Its implementation is fairly simple. However, like the arithmetic checksum, it too can be compromised by a determined interceptor.

NOTE

Checksums make it difficult for the interceptor to alter the information and be undetected. They cannot, however, guard against changes being made. You may want to use other mechanisms such as the operating system access controls and encryption. The operating system access controls can only guard the data when it is stored in a file system. It cannot protect data while it is being transmitted in a network.

Cryptographic checksums provide improvements over arithmetic checksums and CRC checksums. This type of checksum is discussed next.

CRYPTOGRAPHIC CHECKSUMS

In cryptographic checksums, also called *cryptosealing*, the data is divided into smaller sets and a CRC checksum is calculated for each data set. The CRCs of all the data sets are then added together.

This method makes it difficult to alter the data, because the interceptor does not know the sizes of the data sets that are used. The data set size can be variable and computed using pseudo-random techniques, thus making it extremely difficult for the interceptor to alter the data. A disadvantage of this mechanism is that it is sometimes computationally intensive.

Another method, called the *Manipulation Detection Code* (MDC) or one-way hash function, can be used to detect modifications to a file. The one-way hash function is so called because no two inputs can produce the same value. The data in the file is used as the input to the one-way hash function to produce a hash value. If the data in the file is modified, it will have a different hash value. One-way hash functions can be implemented quite efficiently and they make unbreakable integrity checks possible. Examples of a one-way hash function are the MD2 (Message Digest 2) and MD5 (Message Digest 5) functions, described in RFC 1319 and RFC 1321, respectively.

USING AUTHENTICATION SYSTEMS

Authentication can be defined as the process of proving a claimed identity to the satisfaction of some permission-granting authority.

On most systems, the user has to specify a password to their user account before they are allowed to log in. The purpose of the password is to verify that the user is who they claim to be. In other words, the password acts as a mechanism that authenticates the user. However, passwords can be stolen, and someone else can impersonate the user. Because adequate measures are not taken as often as they should be, stolen passwords are the cause of a large number of security breaches on the Internet.

Authentication systems are a combination of hardware, software, and procedural mechanisms that enable a user to obtain access to computing resources. Your site policy must state what type of authentication mechanism you should adopt. If users are to log in to their accounts from an external site, you should use stronger authentication mechanisms than passwords.

NOTE

> Authentication mechanisms range from smart cards to biometric devices such as fingerprint readers, voice print readers, and retina scan devices.

Authentication mechanisms can be augmented by challenge/response mechanisms. Challenge/response mechanisms ask the user to supply some piece of information shared by both the computer and the user, such as the user's mother's maiden name or some special information known to the user and the system.

USING SMART CARDS

A smart card is a hand-held portable (HHP) device that has a microprocessor, input-output ports, and a few kilobytes of non-volatile memory. The user must have one of these devices in their possession to be able to log on to the system. This authentication is based on "something you know." The host computer prompts the user for a value obtained from a smart card when asked for a password by the computer. Sometimes, the host machine gives the user some piece of information that the user has to enter into the smart card. The smart card then displays a response that must then be entered into the computer. If the response is accepted, the session will be established. Some smart cards display a number that changes over time, but is synchronized with the authentication software on the computer.

USING KERBEROS

Many systems can be modified to use the Kerberos authentication mechanism. Kerberos, named after the dog who in Greek mythology is said to stand at the gates of Hades, is a collection of software used in a large network to establish a user's claimed identity. Developed at the Massachusetts Institute of Technology (MIT), it uses a combination of encryption and distributed databases so that a user at a campus facility can log in and start a session from any computer located on the campus.

This mechanism was briefly discussed in Chapter 2.

KEEPING UP-TO-DATE

Your security policy, besides identifying the agencies with which you should establish contact in case of security incidents, should also designate individuals who should keep up-to-date with security issues and problems.

If your organization is connected to the Internet, you might want to join mailing lists or newsgroups that discuss security topics of interest to you.

NOTE

> Remember that it takes time to keep up with the information in mailing lists and newsgroups. Unless you identify the individuals who should keep up with this information, and make it part of their job description, your system administrators will probably not find any time to do so.

Mailing Lists

Mailing lists are maintained by list servers on the Internet. When you join a mailing list, you can communicate with users in this mailing list through electronic mail. To send your response or views on a topic, you can send e-mail to the mailing list. Everyone who is on the mailing list will receive your message. The request to join a mailing list is sent to another e-mail address. This address is *different* from the list e-mail address. You should send your subscription request to the e-mail request address and not the e-mail address of the list. The members of a list, which may number in the thousands, will not appreciate receiving requests to join the mailing list! Some mailing list administrators compile a special list of Frequently Asked Questions (FAQs). FAQs are often a good place to begin finding more information.

Mailing lists can be moderated or unmoderated. In moderated mailing lists, the list owner usually acts as a moderator and screens out mail responses that are not in keeping with the objectives of the mailing list.

NOTE

> A variety of mailing list managers are available. Some are automatic, and some are processed by hand. If you have any doubts about how to subscribe to or unsubscribe from a list, send the following command, where *listname* is the name of the list about which you want information:
>
> ```
> INFO <listname>
> ```

In unmoderated mailing lists, there is no screening process. Consequently, the signal-to-noise ratio can be very low. If you decide that the mailing list is not for you send an "unsubscribe" request to the e-mail request address for the mailing list (and not the mailing list itself!). This request should include the following in the mail body:

```
UNSUBSCRIBE  listname
```

NOTE

> The e-mail term *signal-to-noise ratio* borrows its meaning from the audio usage. *Signal* represents actual, relevant e-mail messages. *Noise* represents useless messages, such as subscribe and test messages, that hinder normal communications among others using the mailing list.

UNIX SECURITY MAILING LISTS

The goal of the Unix security mailing list is to notify system administrators of security problems before they become common knowledge, and to provide information on security related topics. Because this kind of information can be damaging if it falls into the wrong hands, the Unix security mailing list is a restricted-access list. This list is open only to people who can be verified as being principal system administrators of a site.

In order to join this list, the requests must originate from the site contact listed in the Defense Data Network's Network Information Center's (DDN NIC) WHOIS database, or from the root account on one of the major site machines. You must include the destination e-mail address you want on the list. You should also indicate if you want to be on the mail reflector list or receive weekly digests. You should also include the e-mail address and voice mail telephone number of the site contact.

The e-mail address to send the subscription request is as follows:

```
security-request@cpd.com
```

THE RISKS FORUM LIST

The Risks forum is a component of the ACM Committee on Computers and Public Policy. This is a moderated list and discusses risks to the public in computers and related systems. It also discusses security issues of topical interest, major international computer-related security incidents, problems in air and railroad traffic control systems, software engineering, and so on.

To join the mailing list, send an e-mail subscribe message to the following address:

```
risks-request@csl.sri.com
```

In the body of the list include the following line:

`subscribe risks` *`Firstname Lastname`*

If you want to receive a digest version, rather than individual e-mail responses, include the following line:

```
set risks digest
```

This Risks list is also available through the Usenet newsgroup by the following name:

```
comp.risks
```

The VIRUS-L List

The VIRUS-L list discusses computer virus experiences, protection software, and related topics. The list is open to the public and is implemented as a moderated digest. Most of the information is related to personal computers, although some of it may be applicable to larger systems. To subscribe, send e-mail to the following address:

```
listserv%lehiibm1.bitnet@mitvma.mit.edu
```

or

```
listserv@lehiibm1.bitnet
```

In the body of the list include the following line:

```
subscribe virus-L Firstname Lastname
```

If you want to receive a digest version, rather than individual e-mail responses, include the following line:

```
set virus-L digest
```

This list is also available through the Usenet newsgroup by the following name:

```
comp.virus
```

The Bugtraq List

The Bugtraq list discusses software bugs and security holes. This can be used to assess the security risk to your system. It also discusses how these security holes can be fixed, so that you can use this information to fix security holes in your system.

To subscribe, send e-mail to the following address:

```
bugtraq-request@crimelab.com
```

In the body of the list include the following line:

```
subscribe bugtraq-list Firstname Lastname
```

If you want to receive a digest version, rather than individual e-mail responses, include the following line:

```
set bugtraq-list digest
```

THE COMPUTER UNDERGROUND DIGEST

The Computer Underground Digest states its goal as "an open forum dedicated to sharing information among computerists and to the presentation and debate of diverse views."

The Computer Underground Digest contains discussions about privacy and other security-related topics. It can be reached at the following URL:

`http://sun.soci.nui.edu/~cudigest/`

To subscribe, send e-mail to the following address:

`cu-digest-request@weber.ucsd.edu`

In the body of the list include the following line:

```
SUB CuD
```

Also include the subject SUB CuD in your message.

This list is also available through the Usenet newsgroup by the following name:

`comp.society.cu-digest`

THE CERT MAILING LIST

The CERT (Computer Emergency Response Team) puts out advisories. CERT is discussed in this chapter in an earlier section on organizations you can contact for security-related help.

To subscribe, send e-mail to the following address:

`cert-request@cert.sei.cmu.edu`

In the body of the list include the following line:

```
subscribe cert Firstname Lastname
```

If you want to receive a digest version, rather than individual e-mail responses, include the following line:

```
set cert digest
```

THE CERT-TOOLS MAILING LIST

The CERT also maintains a CERT-TOOLS list for the purposes of exchanging of information on tools and techniques that increase the secure operation of Internet systems. The CERT/CC does not review or endorse the tools described on the list.

To subscribe, send e-mail to the following address:

```
cert-tools-request@cert.sei.cmu.edu
```

In the body of the list include the following line:

```
subscribe cert-tools Firstname Lastname
```

If you want to receive a digest version, rather than individual e-mail responses, include the following line:

```
set cert-tools digest
```

CERT's Web site contains a wealth of security information, including all of the CERT advisories. You can reach it at `http://www.cert.org`.

THE TCP/IP MAILING LIST

The TCP/IP mailing list is a discussion forum for developers and maintainers of implementations of the TCP/IP protocol suite. However, many of the questions that are received on the list are from users of various TCP/IP packages, or those seeking help on TCP/IP applications. This list also discusses network security problems.

To subscribe, send e-mail to the following address:

```
tcp-ip-request@nisc.sri.com
```

In the body of the list include the following line:

```
subscribe tcp-ip Firstname Lastname
```

If you want to receive a digest version, rather than individual e-mail responses, include the following line:

```
set tcp-ip digest
```

This list is also available through the Usenet newsgroup by the following name:

```
comp.protocols.tcp-ip
```

THE SUN-NETS MAILING LIST

The SUN-NETS list discusses issues related to networking on SUN Microsystems workstation systems. The discussion centers around networking and security issues dealing with NFS, NIS, and name servers.

To subscribe, send e-mail to the following address:

```
sun-nets-request@umiacs.umd.edu
```

In the body of the list include the following line:

```
subscribe sun-nets Firstname Lastname
```

If you want to receive a digest version, rather than individual e-mail responses, include the following line:

```
set sun-nets digest
```

NEWSGROUPS

Newsgroups are discussion groups that exchange information through special news reader programs. To join a newsgroup, you must use a special news reader program. These are programs such as nn and tin for Unix systems. DOS/Windows systems have many commercial shareware/freeware packages. Using a news reader, you have greater flexibility in handling messages.

Like mailing lists, newsgroups can be moderated or unmoderated.

The Usenet groups dealing with security-related issues are:

misc.security
alt.security
comp.security.announce

The misc.security is a moderated group and also includes discussions of physical security and locks. The alt.security is unmoderated. The comp.security.announce newsgroup contains mailings sent to the CERT mailing list.

Some of the mailing lists are also available via newsgroups. These were mentioned in the section on mailing lists and are listed here for your reference.

comp.risks
comp.virus
alt.society.cu-digest
comp.protocols.tcp-ip

SECURITY RESPONSE TEAMS

Some organizations have formed a group of security experts that deal with computer security problems. These teams gather information about possible security holes in systems. They disseminate this information and report it to the appropriate people. They can help in tracking intruders, and provide help and guidance in recovering from security violations. The teams may have electronic mail distribution lists and special telephone numbers that you can call for information or to report a problem. Some of the teams are members of the CERT System.

COMPUTER EMERGENCY RESPONSE TEAM

The Computer Emergency Response Team/Coordination Center (CERT/CC) was established in December 1988 by the Defense Advanced Research Projects Agency (DARPA). The goal of this team was to address computer security concerns of research users of the Internet. The CERT is coordinated by the U.S. National Institute of Standards and Technology (NIST), and exists to facilitate exchange of information between the various teams.

NOTE

A major motivation for the promotion of the CERT/CC team was to prevent and handle incidents such as the Internet Worm. This incident was mentioned earlier in this chapter.

CERT is operated by the Software Engineering Institute (SEI) at Carnegie Mellon University (CMU). The CERT team has the ability to immediately confer with experts to diagnose and solve security problems. They can also assist in establishing and maintaining communications with your site and government authorities.

When not responding to emergencies, the CERT/CC serves as a clearinghouse for identifying and repairing security vulnerabilities in major operating systems. They can also provide informal assessments of existing systems and guide you in improving your emergency-response capability. Because of this, they can help you indirectly in formulating an effective network security policy. The team has also been known to work with vendors of software systems in order to coordinate the fixes for security problems.

CERT operates a 24-hour hotline that you can call to report security problems such as someone breaking into your system. You can also call this number to obtain current information about rumored security problems. This 24-hour hotline number for CERT is (412)268-7090.

The CERT/CC sends out security advisories to the CERT-ADVISORY mailing list whenever appropriate. To join the CERT-ADVISORY mailing list, send a message to:

`cert-request@cert.sei.cmu.edu`

Security information that is sent to this list also appears in the following Usenet newsgroup:

`comp.security.announce`

Past security advisories are available for anonymous FTP from the host cert.sei.cmu.edu. The FTP server on the host cert.sei.cmu.edu maintains other useful information on security issues. The README file on this server can inform you on what is available. For more information, contact:

CERT
Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213-3890
(412)268-7090
`cert@cert.sei.cmu.edu`
`http://www.cert.org`

DDN SECURITY COORDINATION CENTER

For Defense Data Network (DDN) users, the Security Coordination Center (SCC) serves as a clearinghouse for host/user security problems and fixes, and works with the DDN Network Security Officer. In this regard, the DDN SCC provides a function similar to CERT.

The SCC publishes the DDN Security Bulletin. This bulletin discusses issues that relate to network and host security, security fixes, and concerns to security and management personnel at DDN facilities. The DDN Security Bulletin is available on-line, via Kermit downloads or anonymous FTP, from the host NIC.DDN.MIL. The security bulletins are in files with the following format:

`SCC:DDN-SECURITY-yy-nn.TXT`

The *yy* is the year and *nn* is the bulletin number.

The SCC provides assistance on DDN-related host security problems through the hotline number (800)235-3155 (6 a.m. to 5 p.m. Pacific Time).

To reach the SCC by e-mail, send a message to:

`SCC@NIC.DDN.MIL.`

For 24-hour coverage, you can call the MILNET Trouble Desk at (800)451-7413.

NIST COMPUTER SECURITY RESOURCE AND RESPONSE CLEARINGHOUSE

The National Institute of Standards and Technology (NIST), besides dealing with standards issues, also has responsibility within the U.S. government for computer science and technology activities. The NIST has played a major role in organizing the CERT System, and serves as the CERT System Secretariat.

NIST operates a Computer Security Resource and Response Clearinghouse (CSRC) that provides help and information regarding computer security events and incidents. They are also interested in raising awareness about computer security vulnerabilities. The CSRC team operates a 24-hour hotline at (301)975-5200.

The NIST provides on-line publications and computer security information that can be downloaded using anonymous FTP from the host csrc.nist.gov. The information also is available via the World Wide Web at the URL `http://crsc.nist.gov`.

In addition to the FTP server, NIST operates a personal computer bulletin board that contains information regarding computer viruses as well as other aspects of computer security. To access this bulletin board, use the following information:

Bulletin Board line: 301-948-5717
8 bits, no parity, 1 stop bit

When you first log in to the bulletin board, you must register your user name and address.

NIST also produces special publications related to computer security and computer viruses. These can be downloaded through the bulletin board or the FTP server. For additional information, you can contact NIST at the following address:

Computer Security Resource and Response Center
A-216 Technology
Gaithersburg, MD 20899
(301)975-3359
`csrc@nist.gov`
`http://csrc.nist.gov`

DOE COMPUTER INCIDENT ADVISORY CAPABILITY (CIAC)

The CIAC is the Department of Energy's Computer Incident Advisory Capability. CIAC was formed to provide a centralized response capability and technical assistance center for the DOE sites.

CIAC consists of a four-person team of computer scientists from Lawrence Livermore National Laboratory (LLNL). This group's primary responsibility is to assist DOE sites faced with computer security incidents such as intruder attacks, virus infections, worm attacks, and so on. CIAC keeps sites informed of current security-related events, and maintains liaisons with other response teams and agencies.

CIAC assists sites through direct technical assistance, by providing information, or referring inquiries to other technical experts. It also serves as a clearinghouse for information about security threats, known security incidents, and vulnerabilities. Additionally, it develops guidelines for security incident handling and develops software for responding to security incidents.

CIAC analyzes security events and trends, and conducts training and awareness activities to alert and advise sites about vulnerabilities and potential attacks.

The following are CIAC's phone number, e-mail address, and URL:

(415)422-8193
`ciac@tiger.llnl.gov`
`http://ciac.llnl.gov`

NASA AMES COMPUTER NETWORK SECURITY RESPONSE TEAM

The Computer Network Security Response Team (CNSRT) was formed by NASA Ames Research Center in August 1989. The team's primary goal is to provide help to Ames users, but it is also involved in assisting other NASA Centers and federal agencies.

The CNSRT is NASA's equivalent of CERT. The CNSRT maintains liaisons with the DOE's CIAC team and the DARPA CERT, and it is a charter member of the CERT System.

The CNSRT can be reached through 24-hour pager at (415)694-0571. CNSRT's e-mail address is:

```
cnsrt@ames.arc.nasa.gov
```

SUMMARY

This chapter discussed the factors and issues you need to take into account when designing a secure network. These factors are formalized into a Network Policy that helps you identify security threats, perform risk analysis, and determine how you are going to protect your network resources.

You need to formulate an effective network security policy before building a firewall for connecting your network to the rest of the Internet. It is important to understand exactly what resources and services that you want to protect on the network.

The Network Policy is a document that describes an organization's network security concerns. This document becomes the first step in building effective firewalls.

CHAPTER 4

THE ONE-TIME PASSWORD AUTHENTICATION SYSTEM

As is well known in the security community, two of the most common forms of attack on computing systems connected to the Internet are through the theft of the system password file and through eavesdropping on network connections to obtain user IDs and passwords of legitimate users. The captured user ID and password are, at a later time, used to gain access to the system; or in the case of the suitable password file, the encrypted passwords are converted to plain text through the use of a password cracker. Some systems no longer store the encrypted passwords in the typical password file for this reason. With the theft of a suitable password file, it is likely that a good password-cracking program will crack at least five percent of the passwords!

Alternatively, "sniffer" type systems that analyze the packets on the network looking for TCP connections can sniff out the first 100 bytes of user data, which will easily catch the user ID and associated password. However, if such a system is found on your network, it is likely that more than only passwords have been violated.

The one-time password system is designed to counter these types of attack and force a user to use a different password each time he or she logs in. This is accomplished by providing the user with a password that is different for each login, whether the login attempt is successful or not. As a result, it is not possible for the passwords to be re-used in a replay attack.

One-time passwords work by providing the user with a challenge. The *challenge* is a predetermined string of text, to which there is only one possible response. In this chapter, several implementations of one-time passwords are discussed.

WHAT IS OTP?

In general, the one-time password system, hereafter referred to as OTP, protects the secured system from external attacks on its authentication subsystem. OTP does not prevent the wily cracker from eavesdropping on your network and gaining access to sensitive information, however. This cannot even be achieved solely with the use of a firewall. Network eavesdropping can be prevented only by eliminating the ability to sniff packets where sensitive data passes. (For more information on firewalls, see Chapter 8, "Firewall Architecture and Theory.") OTP also does not protect the organization from "inside jobs" or against active attacks in which the potential intruder is able to intercept and modify the packet stream.

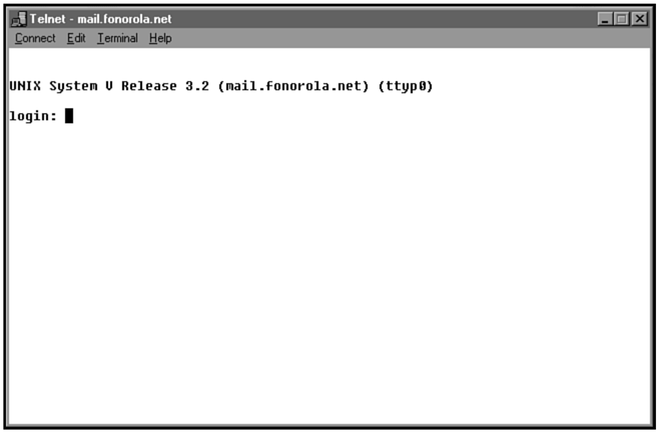
OTP is available in a number of different forms: Bellcore's S/KEY Version 1.0, Bellcore's Commercial S/KEY Version 2.0, the United States Naval Research Laboratory's (NRL) One Time Passwords In Everything (OPIE), and Wietse Venema's LogDaemon.

Before continuing the discussion of OTP, consider the authentication, or login, process for the typical Unix system. The typical Unix system presents an unsecured login prompt where the user enters a password. If the combination of user ID and password match, the user is allowed access to the system. An example of an unsecured login is shown in figure 4.1.

In this configuration, the user enters his user name and password. The password is sent in clear text across the local- or wide-area network, thereby making it easier for the cracker to steal the password. In this example, the user has a multi-use password: one that is used over and over again.

When the OTP system is in operation, only a single-use password (one that is used once, and not again) ever crosses the network. Furthermore, this one-time password consists of six

English words and therefore is not discernible from an ordinary Unix cleartext password. When using either the multi-use or single-use password, the text is sent in the clear across the network. This means that the text is not private.



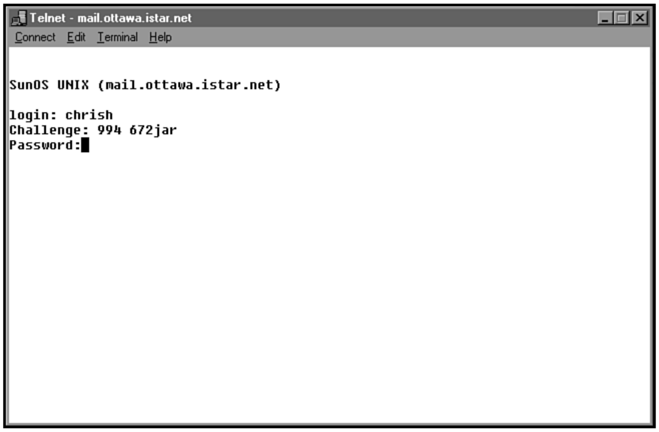
```
UNIX System V Release 3.2 (mail.fonorola.net) (ttyp0)

login: █
```

FIGURE 4.1

Unsecured login.

When using OTP, the user is prompted with a challenge: she must provide the answer to a question that only she can know. OTP makes use of a seed value, an iteration value, and a secret pass phrase that is known only to the user. The challenge is composed of the seed value and the iteration value. The response to the challenge is generated using a special program called a *calculator* on the user's workstation. Using the seed and iteration values, along with the user's secret pass phrase, the response to the challenge is generated. The user's secret pass phrase never crosses the network at any time, including during login or when executing other commands requiring authentication, such as the Unix commands passwd or su. This interaction with the OTP-protected system is illustrated in figure 4.2.



```
SunOS UNIX (mail.ottawa.istar.net)

login: chrish
Challenge: 994 672jar
Password:█
```

FIGURE 4.2

OTP-protected login.

The OTP-protected system responds to the user with a challenge. This challenge consists of the iteration, which is 994 in this case, and the seed, which is 672jar. These values are specific to each user and were configured when the user was added to the OTP database. Worth noting is that even if two users use the same iteration and seed value, their generated passwords will not be the same if they have chosen different pass phrases. The combination of these values and the user's secret pass phrase is passed through a hashing algorithm in the OTP calculator to derive the single-use password. Consequently, OTP is not vulnerable to either eavesdropping or password replay, or to theft or password file attacks.

The operation of the OTP one-time password system involves two sides: the client and the host. On the client side, the appropriate one-time password must be generated. On the host side, the server must verify the one-time password and permit the secure changing of the user's secret pass phrase. This chapter addresses the history of OTP, where to obtain the source and compile it, how to implement the server side, and how to put all the pieces together.

THE HISTORY OF OTP

The idea of using hash functions to generate one-time passwords is not new, having first been presented by Leslie Lamport in the early 1980s. Developed by the Bell Communications Research Center, commonly known as Bellcore, in 1991, S/KEY was the first implementation of a one-time password system. It was first proposed by Phil Karn with contributions from Neil Haller and John Walden.

Through activities involving TCP/IP and amateur radio, Phil Karn saw the problem of password eavesdropping that would be facing network users. To address this problem, Mr. Karn proposed using a one-time password scheme to provide a higher level of access security and to reduce, if not eliminate, the likelihood of password eavesdropping. A description of S/KEY can be found in RFC 1760, an HTML version of which is included on the CD-ROM that accompanies this book.

S/KEY is a one-time password authentication initially implemented using DES as the hashing algorithm. This process was slow on the 8088 systems available at that time, and an encrypted file of one-time passwords was used instead of computing them as needed. Using this encrypted file introduced other potential security problems, which were addressed by moving to a system based on the MD4 Message Digest algorithm, as documented in RFC 1320.

The MD4 algorithm takes as input a message of arbitrary length and produces as output a 128-bit "fingerprint" or "message digest" of the input. To produce two messages having the same message digest, or to produce any message having a given prespecified target message digest, is conjectured to be computationally unfeasible.

Since the initial development of S/KEY, a newer form of digest algorithm, known as *Message Digest 5*, or MD5, has been developed. The MD5 algorithm is documented as RFC 1321 and is an extension of the MD4 algorithm. The Bellcore S/KEY Version 1.0 is currently considered to be a reference implementation, and ongoing development of it is nonexistent. Bellcore is putting its efforts behind the commercial implementation. Recognizing the security limitations placed upon the MD4 implementation, several other OTP implementations have been developed using the MD5 hash algorithm.

> Both the client and the server must be using the same algorithm, either MD4 or MD5. An MD4 client cannot interact with an MD5 server.

NOTE

This chapter does not cover the mathematics discussed in RFC 1320 and RFC 1321; suffice it to say that MD5 was written to address some concerns in the MD4 design and the haste with which it had been adopted. Because MD4 was considered exceptionally fast, designers had concerns about the risks associated with successful cryptoanalytic attack. MD5 designers addressed this concern by making MD5 somewhat slower in computation and also by making some fundamental changes to those computations. The end result is that, unlike MD4, people consider MD5 to be sufficient for use in high and very high security applications.

Starting in April 1995, the Internet Engineering Task Force, known as the IETF, undertook an effort to write a standard for one-time passwords. Neil Haller became one of the cochairs of this working committee, which chose *One Time Password System* (OTP) as the name of the system. This name was chosen because of concern over the use of the name S/KEY, which is a trademark of Bellcore. The current plan of the IETF is to have an RFC assigned and OTP declared a standard in 1996. Ongoing information about and status of the Internet draft and the working group can be found at `ftp.bellcore.com:/pub/ietf-otp/archive`.

Now that you have an overview of why and how OTP developed and matured, you can turn your focus to implementing OTP in your environment.

IMPLEMENTING OTP

OTP works by combining publicly available information, the *challenge*, with a secret known only to the user to produce the proper response. This means that OTP is implemented using one of the major security concepts, Something You Know (SYK). The secret is called a *pass* *phrase*, and when it is combined with the challenge in either the MD4 or MD5 algorithm, it produces the *password* or response. The challenge is in the form of an iteration number and a string of characters called a *seed*. Your seed is not considered a secret, but having all your seeds be unique between systems is important. If you choose your dog's name as a seed on one system, you should not reuse this seed on another system. Your secret (your password) can be the same on all systems, provided that your seed is different.

Different hardware architectures and OS versions have a variety of OTP implementations. When you connect to a system running OTP, you are prompted for your user name. The system responds with an iteration number and your seed. You must then calculate the required response by typing the iteration number, your seed, and your secret password at a trusted piece of hardware, such as a calculator that supports MD4/MD5 or a piece of calculator software.

NOTE

> You can access the system running OTP without being challenged. This can occur on service ports such as SMTP and POP. It is permitted so that new clients are not required.

The calculator computes the required response, and you type that response as your password. The challenge and required response change every time you successfully log in. This mechanism is demonstrated in figure 4.3.

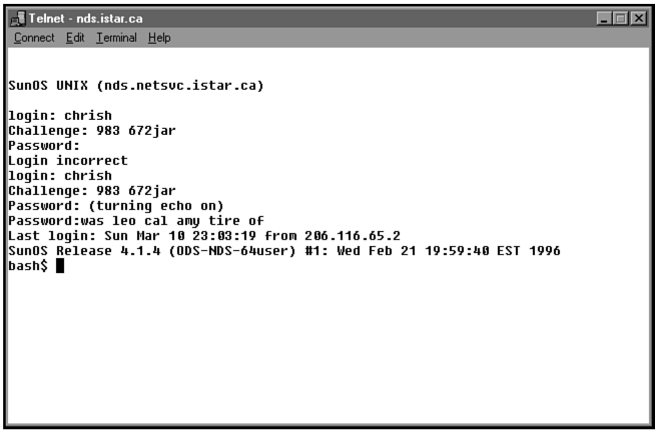


FIGURE 4.3

An OTP login.

```
SunOS UNIX (nds.netsvc.istar.ca)

login: chrish
Challenge: 983 672jar
Password:
Login incorrect
login: chrish
Challenge: 983 672jar
Password: (turning echo on)
Password:was leo cal amy tire of
Last login: Sun Mar 10 23:03:19 from 206.116.65.2
SunOS Release 4.1.4 (ODS-NDS-64user) #1: Wed Feb 21 19:59:40 EST 1996
bash$ █
```

Figure 4.3 shows a sample login to a system protected by OTP. When the user connects to the remote system, the login program provides the iteration number and seed for this user. Using the seed and iteration value supplied in the challenge, the user must then calculate his or her one-time password and log in. In this example, password echoing was enabled so that the password that was provided could be seen.

You must take several actions in order to secure a system with OTP. These are the following:

- Get the OTP code for the system.
- Get the needed calculators for the clients.
- Compile and install the OTP components.
- Have users initialize their keys.
- Enable OTP.

Now you can turn your focus to the steps involved in installing OTP.

Deciding Which Version of OTP to Use

A number of free and commercial versions of OTP are available on the market. These versions are listed in table 4.1.

Table 4.1
OTP Implementations

Version	*Developed By*	*Status*	*FTP Site*	*Notes*
S/KEY 1.0	Bellcore	Free	`ftp.bellcore.com`	This version is more of a reference and is outdated at this point. It is based on MD4.
S/KEY	Bellcore	Commercial		This is a commercial implementation. It supports both MD4 and MD5.

continues

TABLE 4.1, CONTINUED
OTP Implementations

Version	*Developed By*	*Status*	*FTP Site*	*Notes*
OPIE	NRL	Free	`ftp.nrl.navy.mil`	Version 2.1 is in production, and 2.2 is in Beta (as of March 22, 1996). OPIE favors MD5 but does support MD4.
LogDaemon	Wietse	Free	`ftp.win.tue.nl`	The current version is 5.3 and it supports both MD4 and MD5.

Unfortunately, choosing one version or the other can be a difficult task. Each of these versions has its own strengths and weaknesses (such as installation issues, support for only one algorithm, ease of ongoing administration, or operating system support) and differing implementations. The result is that not one of these implementations has everything that the user wants. Consequently, you might see security administrators mix components from all the freely available tool sets.

HOW S/KEY AND OPIE WORK

The S/KEY system derives its strength from the interdependence of several parts. These parts consist of the server application, the encryption system, and the calculator. The server application is the component that resides on the server system and prompts the user to authenticate using S/KEY. For example, the /bin/login program, when appropriately modified, constitutes a server program.

The encryption mechanism is the component that performs the encryption and one-time password generation. Combined with the calculator, this mechanism results in a password that is provided by the user to the server program and, if the generated password is correct, results in successful authentication.

THE ITERATION, SEED, AND PASS PHRASE

The initialization of a user ID to authenticate through S/KEY involves several parts. These are the iteration value, the seed, and the pass phrase. The *iteration value* is a counter that is decreased each time the user authenticates off a given server. The *seed* is a phrase that is provided to the server; the seed is given to the user on each authentication attempt. For example, consider the output of figure 4.3.

In figure 4.3, the iteration value is 983, and the seed is 672jar. As you will see later in this chapter, these values were initially provided when the user account was configured to authenticate through OTP.

The pass phrase is the one piece of information that the user must know. The iteration and seed values are provided by the system. With these values and the appropriate pass phrase, users can generate the correct challenge password/response with their calculators. This process is illustrated in figure 4.4.

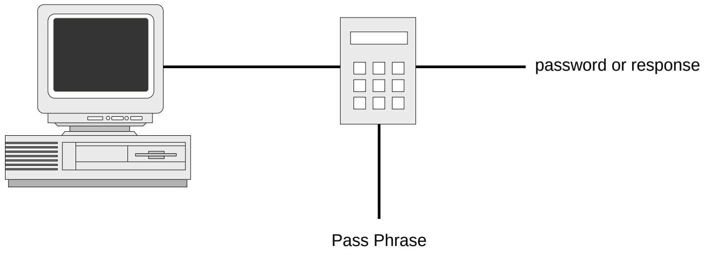


FIGURE 4.4

OTP calculation.

On each successful login, the iteration value is decreased by one. This is how a different password is used on each login attempt. The iteration value should not fall below a value of five because those last five logins will occur very quickly and, when users reach zero, they have no way of logging in to reinitialize their OTP passwords. At that point, only the system administrator can restore access to the system for them.

BELLCORE S/KEY VERSION 1.0

The Bellcore S/KEY distribution, which is available free of charge, is for reference purposes only, according to Bellcore. It no longer receives any real development cycles because S/KEY is also a commercial product. As noted previously, the use of S/KEY for the newly undertaken versions of this software is a violation of the trademark held by Bellcore. If you desire, however, you can use the Bellcore S/KEY distribution without compromising security, considering that this implementation is based on MD4, which is not rated for high security applications.

Bearing in mind the lack of ongoing development of the Bellcore S/KEY distribution, the focus of this chapter is on the two other major development efforts to date: OPIE and LogDaemon.

U. S. NAVAL RESEARCH LABORATORIES OPIE

The United States Naval Research Laboratories produced the One-Time Passwords in Everything (OPIE) Version 2.1 software distribution. OPIE provides a one-time password system for POSIX-compliant Unix-like operating systems. The system should be secure against the passive attacks that are now commonplace on the Internet, as described in RFC 1704, which is included on the accompanying CD-ROM. Despite the use of OPIE, and OTP in general, the system is vulnerable to active dictionary attacks, although these are not widespread at present and can be detected through proper use of system audit software. The NRL OPIE software is derived in part from and is backward-compatible with the Bell Communications Research (Bellcore) S/KEY Version 1 software distribution.

Although the OPIE Version 2 code is backward-compatible with Bellcore S/KEY Version 1.0, it is more tightly based upon the Internet Draft standard for OTP than on S/KEY. Consequently, anyone who has seen or worked with S/KEY Version 1.0 will find the presentation of OPIE somewhat different.

NOTE

> During the preparation of this book, the U.S. Naval Research Laboratories released OPIE 2.2. This release incorporates minor bug fixes and adds support for the development of OTP clients using the OPIE library. You can find OPIE 2.2 at the following URL:
>
> `ftp://ftp.nrl/navy.mil/pub/security/nrl-opie/opie-2.22.tar.gz.`

OBTAINING THE OPIE SOURCE CODE

The location of the OPIE source code is defined in table 4.1, earlier in this chapter. To obtain the source code, select the site for the version you want to download and then retrieve that version via FTP. When you attempt to contact the United States Naval Research Laboratory, you might experience a rough ride. Because it is a busy site during work hours, anonymous access can be difficult to achieve. The best time to get this code is during off-peak hours.

After you have recovered the most recent code, which, at the time of this writing is opie-2.11, you must extract the source code from the tar image. This creates a new subdirectory named opie-2.11, which is where the source code is extracted.

Reading the README and INSTALL files included with the software distribution in the order mentioned is essential. Those documents are supplemented with further explanation in this text. The README file lists all the changes made from the last release to the current and identifies the systems that have been evaluated for properly running OPIE. The list of platforms tested for the 2.11 release is shown in table 4.2.

TABLE 4.2

Tested OPIE Platforms

Hardware	*Software*	*Referred To As*	*System*
Sun SPARCStation 20	Solaris 2.4+SunPro C	Solaris	solaris
Sun 4/300	SunOS 4.1.3+GNU C	SunOS	sunos
Sun SPARCStation 2	4.4BSD-Encumbered	4.4BSD	44bsd
486/66 PC	BSDI BSD/OS 1.1 & 2.0	BSD/OS	bsdos
486/66 PC	Slackware Linux 2.1	Linux	linux
SGI Indigo^2	IRIX 5.2	IRIX	irix
HP 9000/750	HP-UX 9.01+GCC	HP-UX9	hpux9
HP 9000/755	HP-UX 10.0+GCC	HP-UX10	hpux10
IBM RS/6000 550	AIX 3.2.5	AIX	aix

Additionally, information provided to NRL from beta testers suggests that OPIE will work on the platforms listed in table 4.3:

TABLE 4.3

Operating Systems Reported to Support OPIE

Hardware	*Software*	*Referred To As*	*System*
486 PC	FreeBSD	FreeBSD	freebsd
486 PC	NetBSD	NetBSD	netbsd
Macintosh IIfx	A/UX 3.0	A/UX	aux
Sun 3/50	SunOS 4.1	SunOS	sunos

If the Unix system you use appears on either of these tables, you stand a good chance of putting OPIE into operation with little extra effort.

COMPILING THE OPIE CODE

Compiling the OPIE code involves running the autoconfigure script and building the executables for your particular system. If your system is listed in either table 4.2 or 4.3, then you should not have a problem when you compile OPIE for your system. The OPIE developers caution that if you experience a problem when using OPIE, you should defer to the full installation as listed in the INSTALL file. Regardless of which path you choose, most of the steps are still the same.

If you choose to run the autoconfigure script, you will see output similar to that shown in listing 4.1. This example was generated from a Linux Slackware 1.2 system.

LISTING 4.1—SAMPLE AUTOCONFIGURE SCRIPT

```
reliant:/home/opie-2.11# sh configure
creating cache ./config.cache
checking for gcc... gcc
checking whether we are using GNU C... yes
checking whether gcc accepts -g... yes
checking how to run the C preprocessor... gcc -E
checking whether ln -s works... yes
checking for ranlib... ranlib
checking for bison... bison -y
checking for AIX... no
checking for POSIXized ISC... no
checking for minix/config.h... no
checking for chown... /bin/chown
checking for su... /bin/su
checking for su... no
checking for scheme... no
checking for login... /bin/login
checking for ftpd... no
checking for in.ftpd... no
checking for default PATH entries... /usr/bin:/bin:/usr/sbin:/sbin
checking for test -e flag... yes
checking for mkdir -p flag... yes
checking for /etc/default/login... no
checking for /etc/securetty... yes
checking for /etc/logindevperm... no
checking for /etc/fbtab... no
checking mail spool location... /usr/spool/mail
checking whether the system profile displays the motd... no
checking whether the system profile checks for mail... no
checking for -lcrypt... no
checking for -lnsl... no
checking for -lposix... no
checking for -lsocket... no
checking for dirent.h that defines DIR... yes
checking for -ldir... no
checking for sys/wait.h that is POSIX.1 compatible... yes
checking for crypt.h... no
checking for sigemptyset... yes
checking for sigaddset... yes
checking for sigprocmask... yes
checking for getspent... no
checking for endspent... no
updating cache ./config.cache
creating ./config.status
creating configure.munger
creating Makefile.munge
creating config.h

Binaries are going to be installed into /usr/local/bin
Manual pages are going to be installed into /usr/local/man.

creating Makefile

Have you read the README file?
reliant:/home/opie-2.11#
```

If you choose not to use the autoconfigure script, you must edit the Makefile by hand to suit the configuration of your system. After the Makefile is generated, you can run a “make” to build the executables. The make process builds each of the executables and, upon completion, the OPIE system is ready to be tested before being placed into operation.

The Makefile has several command-line arguments. These options select the system for which the commands will be built, and what those commands are. The command used to build the programs is as follows:

```
make system target
```

The system options are listed in table 4.2 earlier in this chapter, and the target options are shown in table 4.4.

WARNING

> The make command is a powerful command generator that is used to build a sequence of commands to be executed by the Unix shell. If you are experienced at using make, then you should use the supplied Makefile or one generated by autoconfigure.

TABLE 4.4
Makefile Target Options

Option	*Action*
client	Builds the opiekey(1) client only
client-install	Builds the opiekey(1) client; installs it and the associated man pages
server	Builds the server programs only
server-install	Builds the server programs and installs the programs and associated man pages
all	Builds everything
install	Builds both the client and server programs; also installs them, opiekey's aliases, and the associated man pages

WARNING

> Do not attempt to install the newly compiled programs until they have been tested. Furthermore, unless you are an experienced system administrator, do not install them at all. Installing programs to replace the operating system versions could render the system unusable.

Typing the command **make system** is the same as using the command **make system all**. The authors of the Makefile have written it so that if you simply type **make**, the Makefile asks you whether you know what you are doing. It is recommended that you build all the programs and then install one after you verify that they are functioning properly.

TESTING THE COMPILED PROGRAMS

Before putting OPIE into production, you should perform some very thorough testing to validate the compile. This testing is important; if you do not do it, and you place these untested programs into production, you could end up with an unusable machine. Also, before you do anything else, back up your system and make sure you have a set of emergency boot disks to restart. To do this, you need two accounts on the system: one with superuser access privileges and one ordinary user account.

Log into the system as root on the system console. The installation cannot be done from any other device than the system console. You must not log out during this testing and installation process because, if the OPIE software is not working properly, logging out might leave your system in a state that prevents you from logging in again. You also must have a non-root account that can access the newly compiled OPIE programs.

Run the opiepasswd command to set the test user's OPIE password. To do this, run the following command:

```
./opiepasswd -c <username>
```

The <*username*> parameter specifies the name of the normal account that you will use to test OPIE. The following list shows the execution of this command:

```
reliant:/home/opie-2.11# ./opiepasswd -c chrish
Adding chrish:
Reminder - Only use this method from the console; NEVER from remote. If you
are using telnet, xterm, or a dial-in, type ^C now or exit with no password.
Then run opiepasswd without the -c parameter.
Adding chrish:
Using MD5 to compute responses.
Enter new secret pass phrase:  <pass phrase typed here>
Again new secret pass phrase:  <pass phrase typed here>
ID chrish OTP key is 499 re1192
HEBE FORD VAN BEAR BEAK LEND
reliant:/home/opie-2.11#
```

The -c option tells opiepasswd that this work is being done on the console.

WARNING

> The warning notice printed by opiepasswd tells you to abort the command if you are using telnet, xterm, or a dial-in session. The same is true if your connection is through rlogin, even though it is not explicitly listed here.

If you don't use the -c option, users must enter their pass phrase through an OPIE-compatible calculator. Because you are testing the code at this point, you do not have to use a calculator. The pass phrase must be between 10 and 127 characters long.

The output of opiepasswd consists of the user identification information (ID chrish), the type of authentication (OTP), and the iteration and seed value. The iteration and seed value are used by the OPIE calculator to generate the password. The group of six words that follow the iteration and seed value is the password that is to be used to access the system. From the previous example showing the use of opiepasswd, the output generated is illustrated here:

```
ID chrish OTP key is 499 re1192
HEBE FORD VAN BEAR BEAK LEND
```

The output is the ID (chrish), the iteration (499), and the seed (rel192). The second line is the actual password used to access the remote system.

The next step is to test the output generated by the opiepasswd command. You perform this test using the opiekey command. To use opiekey, you must supply the iteration number and seed that were provided by opiepasswd. Using the information from the previous example, the execution of opiekey would appear as follows:

```
reliant:/home/opie-2.11# ./opiekey 499 re1192
Using MD5 algorithm to compute response.
Reminder: don't use opiekey from telnet or dial-in session.
Enter secret pass phrase: <pass phrase typed here>
HEBE FORD VAN BEAR BEAK LEND
reliant:/home/opie-2.11#
```

When opiekey prompts for the secret pass phrase, you must use the same one that was used for opiepasswd. If the generated six-word response is the same as the one generated by opiepasswd, then the OPIE software is working properly, and you can continue your testing. If they do not match, you might have misconfigured some piece of software.

NOTE

The typical problem with the software is the byte ordering in the MD4/5 calculations. The OPIE software is supposed to take care of this, but you should check out the generated Makefile if you think there is a problem.

Another, more common, problem is an inconsistent use of the MD4 or MD5 algorithm for the client and server sides. To check, find the lines in the Makefile that read

```
MDX=5

#MDX=4
```

If you decide to use MD4, then set the value of MDX to MD4 and recompile the OPIE programs. If this doesn't fix the problem, then contact one of the OTP mailing lists to discuss your situation. Information on the various OTP mailing lists is found at the end of this chapter.

At this point, you have reason to have confidence in the compile of the OPIE software. Now you need to run the make system-test command to install the OPIE software into your local directories. When you perform this test, the system binaries for login, su, and ftpd are not yet replaced. That replacement occurs later, after you are sure that everything is working as it should. The following is the output of your make:

```
reliant:/home/opie-2.11# make linux-test
make CHOWN="/bin/chown" EXISTS="-e" MKDIR="mkdir -p" RANLIB="ranlib"
➥LOCALBIN="/usr/local/bin"
LOCALMAN="/usr/local/man" SU="/usr/bin/su" LOGIN="/bin/login" DEFAULT_PATH=
➥"/usr/bin:/bin"
FTPD="/usr/sbin/in.ftpd" OPTIONS="-DDOSECURETTY=1" YACC="yacc" test
```

```
make[1]: Entering directory '/home/opie-2.11'
Installing OPIE server software...
Copying OPIE user programs
Changing ownership
Changing file permissions
Preparing opiesu and opielogin for testing
make[1]: Leaving directory '/home/opie-2.11'
reliant:/home/opie-2.11#
```

At this point, you can continue testing your OPIE software. The make output shows that the OPIE binaries have been placed in /usr/local/bin.

You should now run a test to ensure that, with the files in the correct directories, everything still works properly. You perform this test by running opiekey from the newly installed directory. If you receive a message indicating that the opiekey command could not be found, check to make sure that the directory where it was installed is in the PATH. When you run opiekey at this point, an additional option is used because you will need some slightly different information to continue testing. Use the command

```
opiekey -n 7 499 <seed>
```

in which <*seed*> is the same as the seed you used previously. The -n 7 option instructs opiekey to generate seven passwords. The output follows:

```
reliant:/home/opie-2.11# opiekey -n 7 re1192
Using MD5 algorithm to compute response.
Reminder: don't use opiekey from telnet or dial-in session.
Enter secret pass phrase: <pass phrase typed here>
493: GLOM IO OLGA HASH BAH LILY
494: DEAF DUEL SUM MY FOLD ANN
495: GREG ROY BIDE DEAD DAM LOIS
496: FLO BONN SON SKEW GLIB FOLD
497: LUSH OWNS TOOK YALE AFRO HIKE
498: BOP SELL MAYO ORB MID TEN
499: HEBE FORD VAN BEAR BEAK LEND
reliant:/home/opie-2.11#
```

Here, opiekey provides seven different passwords. In this manner, opiekey can be used to generate lists of passwords for future use. However, you should omit the leading ./ from the opiekey command at this stage to make sure that the copy from the binary directory is used. Write down the output from opiekey; you need it to complete the installation and testing instructions.

Type **./opiesu <*username*>**, where <*username*> is the same user name you used when you ran opiepasswd. You are not asked for a password at this point. Again, type the command **./opiesu <*username*>**. opiesu should now ask you for a password. Press Enter once. You should receive a message saying `(echo on)` and asking you for a password again. This message is shown in the following example:

```
reliant:/home/opie-2.11# ./opiesu chrish
reliant:/home/opie-2.11$ ./opiesu chrish
otp-md5 498 re1192
(OTP response required)
chrish's password:  (echo on)
chrish's password: BOP SELL MAYO ORB MID TEN
reliant:/home/opie-2.11$ id
uid=502(chrish) gid=100(users) groups=100(users)
reliant:/home/opie-2.11$
```

Notice that opiesu prompts with the iteration number and seed value. The seed value is the same as in the previous examples, but the iteration value has decreased from 499 to 498. Remember, for each successful login attempt, the iteration value is decremented by one. You must be aware of this value to provide the correct password in response to the challenge.

When you press Enter once and turn on the password echo feature, you can easily see the six words as you type them, thereby reducing your chances of making a mistake. To turn password echo on, simply press Enter when asked for your password. When password echo is turned on, you will see the words `echo on`, and your text will start on a new line.

NOTE

> Having a password visible on the screen is not a common practice because common sense tells us to keep our passwords secret. However, so long as the pass phrase is divulged, the passwords themselves can be echoed. This enables the user to see the words he is typing and eliminates the likelihood of a typing error.

Enter the six words (and only the six words) on the line starting with 498 that you got when you ran the opiekey command. If you receive the message `Sorry`, verify that you are using the correct password and repeat the password once. If the message is still `Sorry`, the OPIE software is not working properly on your machine.

You now need to use the opieinfo command to verify the seed that OPIE thinks is to be used. The output provided by opieinfo, as follows, indicates the iteration number and the current seed:

```
reliant:/home/opie-2.11$ opieinfo
497 re1192
reliant:/home/opie-2.11$
```

The final step in the testing process is to test the operation of the replacement login command. Type the command

```
./opielogin <username>
```

where <*username*> is the user name that you have been using. The program should now ask you for a password. Press Enter once. You should see the message `(echo on)`, along with a request for a password again. Enter the six words (and only the six words) on the line starting with 497 that you received from the opiekey command earlier. The following list shows this sequence:

```
reliant:/home/opie-2.11$ ./opielogin chrish
otp-md5 497 re1192
Password:
Linux 1.1.18. (POSIX).
Welcom to Reliant
reliant:~$
```

If opielong responds with `Login incorrect`, repeat the preceding sequence once. If the message is still `Login incorrect`, the problem is related to either byte-ordering or an inconsistency in the MD4/MD5 algorithm specification between the calculator and the server.

If the software works but displays your message of the day twice, you need to change the setting for -DDOMOTD to zero in the Makefile and start over. If everything seems to be functioning properly, then it is safe to install OPIE and start using it.

INSTALLING OPIE

Installing OPIE involves replacing some of your system binaries and reevaluating your system. Remember to make sure you have a backup or some way of restarting and reinstalling your system. To complete the installation of the programs, use the following command:

```
make install
```

This command installs the OPIE replacements for login, su, and ftpd. The installation process will try to rename your old programs to their original names with the extension opie.old.

WARNING

> If programs or files with the same name (that is filename.opie.old) already exist, then your old programs will not be backed up.

Listing 4.2 shows the installation being run.

LISTING 4.2—INSTALLING THE OPIE SOFTWARE

```
reliant:/home/opie-2.11# make linux-install
make CHOWN="/bin/chown" EXISTS="-e" MKDIR="mkdir -p" RANLIB="ranlib"
➥LOCALBIN="/usr/local/bin"
LOCALMAN="/usr/local/man" SU="/usr/bin/su" LOGIN="/bin/login" DEFAULT_PATH=
➥"/usr/bin:/bin"
FTPD="/usr/sbin/in.ftpd" OPTIONS="-DDOSECURETTY=1" YACC="yacc" install
make[1]: Entering directory '/home/ftp/security/opie-2.11'
Installing OPIE client software...
Copying OPIE key-related files
Changing file permissions
Symlinking aliases to opiekey
Installing manual pages
Installing OPIE server software...
Copying OPIE user programs
Changing ownership
Changing file permissions
Preparing opiesu and opielogin for testing
Clearing testing permissions on opiesu and opielogin
Installing OPIE system programs...
Renaming existing /bin/login to /bin/login.opie.old
Clearing permissions on old /bin/login
Copying opielogin to /bin/login
Changing ownership of /bin/login
Changing file permissions of /bin/login
Renaming existing su to su.opie.old
Clearing permissions on old su
Copying opiesu to su
Changing ownership of su
Changing file permissions of su
Renaming existing ftp daemon to /usr/sbin/in.ftpd.opie.old
Clearing permissions on old ftp daemon
Copying OPIE ftp daemon
Changing ownership of ftpd
Changing file permissions of ftpd
Creating OPIE key file
Changing permissions of OPIE key file
Changing ownership of OPIE key file
Installing manual pages
REMEMBER to run opiepasswd on your users immediately.
make[1]: Leaving directory '/home/ftp/security/opie-2.11'
reliant:/home/opie-2.11#
```

Checking the operation of some components before exiting and restarting the system is important. First, evaluate the operation of the replacement FTP daemon. Type **ftp localhost**. At the prompt, enter the user name you have been using. You should receive an OTP challenge. Log in using the user that you configured earlier in this process, and use the correct password for the current iteration. This is illustrated in the following example:

```
reliant:/home/opie-2.11#ftp localhost
Connected to reliant.unilabs.org.
```

```
220 reliant FTP server ready.
User (reliant.unilabs.org:(none)): chrish
331 OTP response otp-md5 496 re1192 required for chrish.
Password:
230 User chrish logged in.
ftp>
```

If you do not see a line that starts with `331 OTP response`…, then you either did not install the OPIE replacement program in the proper directory (in which case you need to change the value in the Makefile and start over), or you are using a client program that will not allow users to see challenges, in which case you need to contact the author of your client for an updated version that fixes this deficiency.

> The default configuration of the OPIE FTP Server is not to allow anonymous connections. To allow anonymous connections, add the DDOANONYMOUS option to your system's options entry and rebuild. This is further explained in the later section entitled "opieftpd."

The last step is to test the operation of the telnet program. To test the program, execute the command **telnet localhost** and log in using the previously defined non-root user and the appropriate challenge response, as follows:

```
reliant:/home/opie-2.11# telnet localhost
Trying 127.0.0.1 ].
Connected to localhost
Escape character is "^]"
Linux 1.1.18 (reliant.unilabs.org) (ttyp1)
login: chrish
otp-md5 494 re1192
(OTP response required)
Password: (echo on)
Password:DEAF DUEL SUM MY FOLD ANN
Linux 1.1.18. (POSIX).
reliant:~$
```

If the connection is configured correctly and responds as it should, then your system is operational and successfully OPIE-ized. You must now configure the users by running opiekey for each account. At this point, you should reboot your system to make sure it comes up as it should. If you experience any problems, you might need to reinstall and recover your system from a backup taken prior to the OPIE configuration.

THE OPIE COMPONENTS

The OPIE implementation of OTP consists of several components, or programs. These programs are as follows:

- opieftpd
- opieinfo
- opiekey
- opielogin
- opiepasswd
- opiesu

OPIEFTPD

This FTP server daemon is a full-featured FTPD implementation. It is not a stripped-down version, yet it does not have all the functionality of the Washington University Archive FTP server. If you did not choose to install the FTP server during the initial installation of OPIE, then you can do so by changing the name of the existing FTP server and copying the OPIE FTP server into the same directory.

The opieftp daemon does not, by default, support anonymous FTP logins. Anonymous access would allow access to the server through one of the most attacked points—not to mention the fact that anonymous FTP sites often are used to store pirated applications. If you require the use of anonymous FTP, however, then you must set it up according to the following instructions.

Look in the Makefile for the OPIE software to find the configuration options. This section is shown in listing 4.3:

LISTING 4.3—MAKEFILE OPTIONS SECTION

```
# * Vendor-compatible "features"
#
# -DDOUTMPX=1           If your system uses a utmpx file along with a utmp
# -DDOSECURETTY=1       If you want to use an /etc/securetty file to control
#                       which terminals root can log in from
# -DPERMSFILE="<file>"  Change the permissions of certain devices on login,
#                               as specified in <file>
# -DDOWHEEL=1           Implement the BSD "wheel group" su restriction
#                               (only members of group 0 can su)
# -DDOTITLE=1          Change the process info of ftpd so that ps listings
#                              will show status information
# -DDOMOTD=0           If your system's login program *doesn't* display
#                      /etc/motd and check for mail (i.e., it is done in
#                      shell scripts like /etc/profile and /etc/.login)
#
# * Miscellaneous
#
# -DDOANONYMOUS=1      If you want ftpd to support anonymous logins
#                      whenever an "ftp" account exists in /etc/passwd.
#
# -DSYS_FCNTL_H=1      Use <sys/fcntl.h> instead of <fcntl.h>.
# -DMJR=1              Support Marcus J. Ranum's scheme to prevent
```

To allow for anonymous user access, find the OPTIONS line that corresponds to your system and add the -DDOANONYMOUS option, as shown here:

```
# Linux
OPTIONS=-DDOANONYMOUS=1
```

The operating-system-specific OPTION lines are found in the Makefile following the text shown in the listing 4.3. You must then recompile the FTPD server and install that version. In addition, the standard configuration issues for anonymous logins to the FTPD server must be observed. In this situation, however, performing those configuration steps is very important; without them, you might be invalidating the OPIE configuration steps you have just taken.

OPIEINFO

opieinfo retrieves and prints the current iteration and seed from the OPIE database, either for the current user or for the specified user. opieinfo will retrieve the next iteration value and the seed for the user. If no user is specified on the command line, then opieinfo looks for the information on the user who is running the program. The following example shows the output of opieinfo:

```
reliant:/home/ftp/security/opie-2.11$ opieinfo
488 re1192
reliant:/home/ftp/security/opie-2.11$ opieinfo chrish
488 re1192
reliant:/home/ftp/security/opie-2.11$ opieinfo terrih
terrih not found in database.
reliant:/home/ftp/security/opie-2.11$
```

In this example, chrish is found in the OPIE database, so his information is printed by opieinfo. When chrish attempts to look up the iteration and seed for terrih, he is told that terrih is not in the database. This means that terrih has not been initialized with opiepasswd yet. That initialization is demonstrated later in this chapter.

OPIEKEY

opiekey is just one of many names used for the same program. opiekey is also known as opie-md4, opie-md5, otp-md4, otp-md5, and opie-des. opiekey takes the optional count of the number of responses to print along with a (maximum) sequence number and seed as command-line arguments. It prompts for the user's secret password twice and produces an OPIE response as six words. Using this, you can generate a list of OPIE passwords in advance, so if you are away from a calculator, you still have some means of logging in. This could be rather dangerous, though, because you could lose the paper and unintentionally allow someone to gain access to your systems.

The operation of opiekey is illustrated in listing 4.4:

LISTING 4.4—SAMPLE OPIEKEY EXECUTION

```
reliant:/home/ftp/security/opie-2.11$ opieinfo
488 re1192
reliant:/home/ftp/security/opie-2.11$ opiekey -n 10 488 re1192
Using MD5 algorithm to compute response.
Reminder: Don't use opiekey from telnet or dial-in sessions.
Enter secret pass phrase:
479: MAD EM SELF TOTE FORD SLAY
480: HUFF VEAL RUST DAYS FEAR GIL
481: BUSY FEAR LAY MONA SINK CROW
482: YELL RANT HIDE KENT HE PO
483: TAB ITEM COAL BROW HAAG FUR
484: PUP SKID HALF BALD TELL HASH
485: INN AT LARK KEN LONG KEEN
486: HILL REAR ADAM MESS FOUL EWE
487: BANG ANA NEWT MADE OMAN OTTO
488: LOS RAYS NAME VASE FOOT RIFT
reliant:/home/ftp/security/opie-2.11$
```

This example shows how to use opieinfo to obtain the current iteration and seed, which are required by opiekey. In this example, 10 passwords are printed for future use. As already mentioned, this is valuable information and must be handled with care. Anyone who obtains this information and your user name will be able to access your account.

OPIELOGIN

The opielogin command is a straight replacement for the standard login command. This command is used to authenticate users when they want to access the system. Most people see the output of login as the login prompt. In fact, the login command is also responsible for authenticating users before their admittance to the system.

To have the data it needs to authenticate the user, opielogin uses the file /etc/opiekeys, which contains the user authentication information. If you choose to live dangerously, you might also use the ./opiealways file to determine which users can log in with clear text and which ones must have the OTP challenge. This is a major security hole and should be used only if the site security administrator determines that no other way exists.

OPIEPASSWD

opiepasswd is the command that initializes the user into the OPIE database, /etc/opiekeys. Before this initialization, the user will not be able to log in to an OPIE-secured system. The initialization of the user can take place in one of several ways.

First, the opiepasswd command can be executed from a secure terminal or console, which allows users to provide their own secret pass phrase. If opiepasswd is not executed from a secure terminal, users must have access to an OPIE-capable calculator. Finally, users on the console, perhaps root, can assign the initial iteration and seed value for each user.

opiepasswd can generate a default number of iterations, which is 499, or a user-specified value. Furthermore, opiepasswd can generate a random seed value or use one provided by the user.

OPIESU

opiesu is a replacement program for the su command. This application is used to switch to another userid, with the appropriate OTP challenge being issued. If the correct response is provided, then the request is granted, and the user identity is changed. Otherwise, the failure is logged and reviewed later.

LOGDAEMON 5.0

The LogDaemon 5.0 tools are a project of Wietse Venema of Eindhoven University of Technology in the Netherlands. Venema is well known for his TCP Wrapper and SATAN projects. The LogDaemon programs were originally developed in 1990 after Eindhoven University experienced a case of severe hacking after it was connected to the Internet. Wietse immediately wrote an early version of the TCP Wrapper programs in an attempt to gain control over the hacking problem. At that time, Wietse was preparing for the worst possibility: an intruder penetrating one of the university's systems. The resulting LogDaemon programs were intended to be direct plug-in replacement programs with built-in keystroke logging. The official distribution does not have keystroke logging, however, so don't look for it in the downloaded implementation. Wietse added OTP support to LogDaemon during the SATAN development project with Dan Farmer because he did not want to type his password over an unsecured link from San Francisco. This project resulted in the current implementation of the LogDaemon tools that exist today.

The LogDaemon kits are a variation of the Bellcore S/KEY package and include not only support for MD4 and MD5 but also a large array of server-side daemons that were modified to include support for OTP. These server daemons, whose transformation occurred over a number of years, are based on the BSD sources.

The rsh and rlogin daemons log the remote user name and perform logging and access control in TCP/IP wrapper style. To improve the level of security that is generally lacking in the r* commands, these daemons, by default, do not accept wild cards in either the hosts.equiv or .rhosts files. Both daemons have an '-l' option to disable user .rhosts files. The rshd command has a compile time that will cause it to log the user command.

The ftpd, rexecd, and login software supports considerable login failure logging, with optional support for S/KEY one-time passwords. The rexecd daemon disallows root logins, which is a popular back door for the interloper. The support for S/KEY one-time passwords is optional and completely invisible to users that do not need it. Unix passwords are still permitted by default. Like rshd, the rexecd command can be compiled to log executions of the user command.

For those network administrators who want an S/KEY login shell but, for whatever reason, cannot replace the standard login program, the S/KEY shell is provided. The user first logs in to a dummy account that does not require a password. The S/KEY login shell prompts for the user's real account name and presents the corresponding S/KEY challenge.

With Wietse's knowledge and visibility in the network security field, this code is very popular and is often used in conjunction with applications from other packages.

Obtaining the LogDaemon Code

The LogDaemon code can be found at `ftp.win.tue.nl` in the /pub/security directory, or through the URL `ftp://ftp.win.tue.nl/pub/security/logdaemon-5.3.tar.gz`. To get the code, download the file logdaemon-5.3.tar.gz (or later) from this directory. In fact, if you are looking for security-related code, you will find some good packages here, including TCP Wrapper, Portmap, Crack, COPS, Tiger, and SATAN.

NOTE

> If you plan to build the rsh and rlogin programs, you might as well download the latest release of the TCP Wrapper code in addition to the LogDaemon code. The rshd and rlogind compiles depend on a library in the TCP Wrapper package.

The LogDaemon and TCP Wrapper sources as downloaded are GNU gzip compressed files, so you must have GNU gzip to decompress them. After they are decompressed, untar the images, which results in the creation of a directory called logdaemon-5.0 and the installation of the source files installed under it.

NOTE

> You can find many other high-quality, interesting security programs here that are also worth investigating. Some of them have been discussed in *Actually Useful Internet Security Techniques* (ISBN: 1-56205-508-9) and the *Internet Security Professional Reference* (ISBN: 1-56205-557-7), both of which were published by New Riders.

COMPILING THE LOGDAEMON CODE

With the source code now unpacked on your system, you must configure and compile it. The sources are based on the original BSD Unix sources and have been tested on a number of different platforms. These platforms are as follows:

Sun OS 4.1*x*, 5
Sunsoft Solaris 1.1.2, 2.4
DEC Ultrix 4.*x*
IRIX 5.*x*
HP/UX 9.*x*
DEC OSF 1.*x*
Sony NewsOS 4.*x*

These are the only currently supported platforms. The LogDaemon code was evaluated also on Linux, but only parts of the tools could be compiled. Therefore, if you are not running one of these versions of the Unix operating system, you can expect to have some porting problems.

The LogDaemon programs require the libwrap.a library from the TCP Wrapper 7.3 distribution. Consequently, you must compile the TCP Wrapper distribution first. You do this by simply running the make command identifying the named target. The available targets for TCP Wrapper are identified in the following list:

generic
386bsd
aix
alpha
apollo
convex-ultranet
dell-gcc
dgux dgux543
dynix
epix
esix
freebsd
hpux
irix4 irix5
isc (untested)
iunix
linux
machten
mips (untested)
ncrsvr4
netbsd
next
osf
ptx-2.x ptx-generic
pyramid
sco sco-nis sco-od2 sco-os5
sunos4 sunos40 sunos5
sysv4
ultrix
unicos (untested)
unixware1 unixware2
uxp

To compile TCP Wrapper, you need only to set the value of REAL_DAEMON_DIR in the Makefile. You can do this on the make command line, as follows:

```
$ make REAL_DAEMON_DIR='pwd' sunos4
```

In this example, the REAL_DAEMON_DIR is set to the current directory. Normally, it would be set to the location in which the daemons are actually installed. In this case, however, all you want is the libwrap.a library, so the actual value of the REAL_DAEMON_DIR is less important.

After the TCP Wrapper libwrap.a library archive is built, the location of this library must be saved in the user environment to be used when you are compiling the actual LogDaemon tools. You save the location of this library in the user environment by using one of the following commands:

For C-shell:

```
setenv LOG_TCP path to librwap.a file
```

For Bourne and Korn shells:

```
export LOG_TCP=path to libwrap.a file
```

Now go back to the logdaemon directory and run the command

```
make target
```

where *target* is one of the supported platforms previously listed.

> **NOTE**
>
> Your success with nonsupported platforms might be very limited. The author attempted to compile the LogDaemon tools under Linux, but found considerable problems with some of the tools—most notably the FTP daemon.

With the LogDaemon programs successfully compiled, it is necessary to test them before placing them into operation.

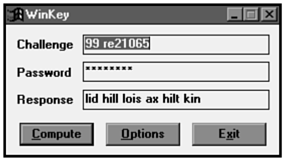
TESTING THE COMPILED PROGRAMS

The test procedures for the LogDaemon tools are not as well defined as for the OPIE tools. Bear in mind, however, that this tool set, like OPIE, replaces a number of system programs. You must be ready to and capable of recovering your system should these new replacement programs not function properly. The compiled programs are best tested before moving them into their target locations.

First, test the creation of users in the OTP database. Remember, although the term *OTP* is used, the LogDaemon programs use a modified version of the S/KEY Version 1.0 implementation. Creation of users in the OTP database is performed with keyinit, which is illustrated here:

```
reliant:/home/ftp/security/logdaemon-5.0/skey# pwd
/home/ftp/security/logdaemon-5.0/skey
reliant:/home/ftp/security/logdaemon-5.0/skey# ./keyinit terri
Adding terri:
Reminder - Only use this method if you are directly connected.
If you are using telnet or rlogin exit with no password and use keyinit -s.
Enter secret password: <type pass phrase here>
Again secret password: <type pass phrase here>
ID terri s/key is 99 re21065
LID HILL LOIS AX HILT KIN
reliant:/home/ftp/security/logdaemon-5.0/skey#
```

The preceding example uses the keyinit command to add the user terri to the OTP database, which is in /etc/skeykeys. You can verify this key by testing it against another calculator that also is configured appropriately. For example, you can test it with the Windows calculator, WinKey, as shown in figure 4.5.



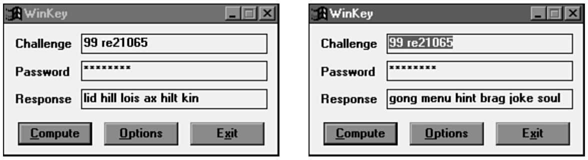
Challenge: 99 re21065
Password: ********
Response: lid hill lois ax hilt kin

FIGURE 4.5

The Microsoft Windows OTP calculator.

As you can see, the calculated response in WinKey is the same as that shown in the same execution of keyinit. During testing, bear in mind that it is possible to confuse the encryption types. For example, if you expect to be using MD5, and the answer doesn't match in an MD5 calculation, try the MD4 method.

The only difference between the two generated passwords is that one will work and one will not. Both the MD4 and MD5 algorithms generate a six-word password, but because the algorithm is different, the series of words will not be the same. Often the case is that the algorithm does not match, as shown in figure 4.6.



Challenge: 99 re21065
Password: ********
Response: lid hill lois ax hilt kin

Challenge: 99 re21065
Password: ********
Response: gong menu hint brag joke soul

FIGURE 4.6

MD4 and MD5 generated password differences.

In figure 4.6, the WinKey calculator has two different answers for the same challenge and pass phrase, but only one is correct. You know from the previous calculation that the answer shown on the left is the correct one. This is a common problem when you set up the LogDaemon tools, however, because they understand both MD4 and MD5. Incidentally, the WinKey on the left in figure 4.6 is MD4, and the one on the right is MD5.

If you want to use the MD5 algorithm but got MD4 instead, now is the time to change it. To make the correction, edit Makefile in the skey directory and then change the following lines to read MD5:

```
CFLAGS  = -O $(XFLAGS) -DPERMIT_CONSOLE -DKEYACCESS=\"$(KEYACCESS)\" \
        -DKEYFILE=\"$(KEYFILE)\" -DMD4
```

You then change to the top-level directory, type **make**, and rebuild the components. You must rebuild the entire structure because the libskey.a archive will be rebuilt, and some of the other components rely upon this library.

Having proven the operation of the keyinit program by using an external calculator, you can test the key program to make sure that the local calculator can generate the correct response. The key command will print the password for the supplied iteration and seed value. The user is asked to enter his secret pass phrase, and the password is then generated, as shown here:

```
reliant:/home/ftp/security/logdaemon-5.0/skey# ./key 99 re21065
Reminder - Do not use this program while logged in via telnet or rlogin.
Enter secret password: <enter secret pass phrase>
LID HILL LOIS AX HILT KIN
reliant:/home/ftp/security/logdaemon-5.0/skey#
```

As illustrated, the generated password is the same as that previously shown. Consequently, you can assume that the keyinit and key programs are functioning properly.

Now that you have tested the basic OTP functionality, you need to examine the other tools. You don't necessarily have to use any of these other tools. Some people have retrieved the tools for only part of the package's functionality. In fact, although the LogDaemon tools include the BSD r* commands, many sites disable these commands because of their inherent dangers. The purpose of these tools, however, is to allow users the freedom of using these facilities.

Installing LogDaemon

The LogDaemon tools are installed by simply copying the newly compiled programs into the same places as their existing versions. Remember to save the system-provided implementation in case the newly compiled version has a problem.

> **TIP**
>
> Before you install any of these tools over system-provided copies, make a backup of the existing system so that you can recover it in the event of a disaster.

After the new programs have been installed into the proper directories, your system is ready to have its users configured with keyinit. Putting OTP into practice is discussed in a later section of this chapter.

The LogDaemon Components

As mentioned previously, the LogDaemon package consists of a collection of commands to replace the operating system versions and to provide additional support. This section provides a brief overview of each of the components in the LogDaemon package.

FTPD

The ftpd program is a direct plug-in replacement for the system implementation. It is, however, a much less impressive version than the wu-ftpd from Washington University, or even the implementation provided with OPIE. It does support extensive failure logging, one-time passwords, and the SecurID card, however.

LOGIN

The login program is a direct replacement with extensive logging and support for one-time passwords. You should note that support for OTP is not a requirement. Those sites that cannot or will not replace the login program can create a generic access account and then use skeysh to authenticate a specific user. Creating this account is illustrated in the later section "Putting OTP into Practice."

REXECD

The rexecd program is the server program for the rcmd command. To place this program into operation, review the /etc/inetd.conf file for an entry for rexecd. Check this line to make sure it is not commented out, as it is in the following example:

```
#exec   stream  tcp     nowait  root /usr/sbin/in.rexecd
```

Replace the named program (in.rexecd) with the newly compiled rexecd. This new command supports OTP and extensive logging. By default, the logging records are written using the syslog facility LOG_AUTH. This can be changed by altering the following line in the rexecd/Makefile and building the executable again:

```
CFLAGS  = -I$(UTIL) -O -DFACILITY=LOG_AUTH $(XFLAGS) # -DLOG_COMMANDS
```

RLOGIND

The rlogind program is the server side of the rlogin program. It supports extensive logging and access control capability. It logs regular access using the syslog priority DAEMON.INFO and other messages using DAEMON.WARN.

RSHD

The rshd program is the server program for the rcmd and rsh client programs. It supports OTP, and extensive logging Regular access is logged (by default) with priority DAEMON.INFO.

SKEYSH

The skeysh program is a login shell that understands S/KEY for those sites that cannot replace the login program. The solution is to create a dummy account with skeysh as the login shell, because skeysh is nothing but a stripped-down skey-only login program. The solution to the problem using skeysh is to first have users log in to a dummy user account. Doing so drops them into skeysh, which prompts them for their real account name and presents the corresponding S/KEY challenge.

USING THE S/KEY AND OPIE CALCULATORS

As you have seen thus far, a calculator or list of pregenerated passwords is required before a user can access the OTP-protected system. The calculator can take a number of forms. It can be an application that runs on an OTP-protected system to allow access to other remote OTP systems. It can be an application that runs on the user's desktop system for use when needed. It can be a list of printed pregenerated passwords, or even an external calculator. This section covers the available hardware calculators and the software calculators available for different operating systems.

The critically important point to understand here is that you use only a calculator that you trust to be secure. This can be in the form of a hand-held calculator, an application running on your Mac or PC, or a bit of paper having some precomputed pass phrases. You should never type your secret password over the network.

UNIX

Each of the OTP systems comes with its own OTP password program for the Unix operating system. The names of these programs and the relevant package are listed in table 4.5.

TABLE 4.5

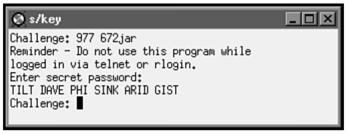
Unix OTP Calculators

Package	*Key Generator Name*
S/KEY	key
OPIE	opiekey
LogDaemon	winkey

These programs run on the Unix host itself. They do not allow access to the individual host where they are executed, because it is likely to be OTP-protected also. To use these calculators, the user must execute them with the iteration and seed information so that a key can be generated. Figure 4.7 illustrates using the LogDaemon implementation, winkey, to generate a password for a specific iteration and seed value.

FIGURE 4.7

The LogDaemon S/KEY calculator.



```
Challenge: 977 672jar
Reminder - Do not use this program while
logged in via telnet or rlogin.
Enter secret password:
TILT DAVE PHI SINK ARID GIST
Challenge: █
```

This example illustrates using the winkey program that is part of the LogDaemon package. winkey is an S/KEY calculator that runs in an X Window and generates the required keys.

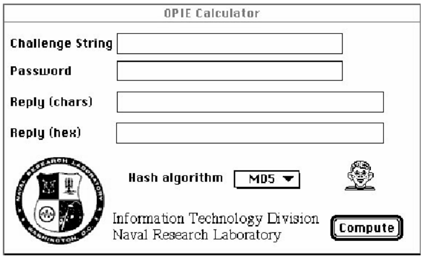
Figure 4.7 illustrates the LogDaemon version. The OPIE implementation, opiekey, appeared in earlier examples.

MACINTOSH

The most popular Macintosh calculator is found at `ftp.nrl.navy.mil` in /pub/security/nrl-opie/OPIEcalc.sit.hqx. Like the other calculators, it is used to calculate the response to the OTP challenge provided by the remote system. The calculator computes a response based on the iteration and seed value provided by the remote system, and on the user's secret pass phrase. Figure 4.8 illustrates the most popular Macintosh calculator.

FIGURE 4.8

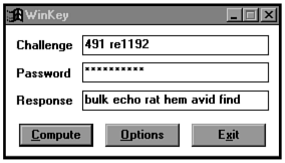
The NRL OPIE calculator for the Macintosh.



This file is a self-extracting archive and contains a fat binary, meaning it will run on both the 68-KB and Power Macintosh systems. It supports both S/KEY and OPIE using MD4 or MD5.

MICROSOFT WINDOWS

The Microsoft Windows calculator for Windows 3.1 and Windows 95 can be found at either `ftp.nrl.navy.mil` in /pub/security/nrl-opie/opie.contrib/winkey11.zip, or at `ftp.tlogic.com/pub/skey`. This is a DOS zip file that requires you to have an unzip program to extract the files. The contents of the archive include the compiled application winkey.exe and all the source code. Figure 4.9 illustrates the WinKey application.



Challenge 491 re1192

Password ***********

Response bulk echo rat hem avid find

FIGURE 4.9

Generating passwords with WinKey.

WinKey has a number of options that increase its usability. For example, it can be configured to read the challenge from the Clipboard and to paste the computed value back into the Clipboard. It generates both S/KEY MD4 and OPIE MD5 passwords. Generating both passwords is important because the application keyapp.exe, which is part of the Bellcore S/KEY reference set, can generate only MD4 passwords.

If you use WinKey on Windows 95 and find that the application is not visible on-screen when you start it, the problem is the winkey.ini file in the Windows directory. You can either remove the winkey.ini file, or edit it and change the X and Y values to 0. This will start WinKey in the top left corner of the screen. It is a nonissue for the author of WinKey, who will one day get around to fixing it.

EXTERNAL CALCULATORS

You can use external calculators that hold the necessary program code. For example, the Hewlett-Packard HP48 series hand-held calculator has the capability to accept a downloaded program, and it will generate a key given the iteration, seed, and password for the user. This is a highly trusted method, because a cracker cannot gain access to your external calculator and therefore your password unless he or she is sitting right beside you. If you choose this route, note that the G, GX, S, and SX versions of the HP48 all can run this code. It requires 16 KB of memory, so the low-end versions might need memory upgrades.

To use an external calculator, you must type the iteration, seed, and then your password into the calculator. The calculator performs the computation, and you must then type the generated pass phrase into the remote system in order to complete the authentication. The downside is that users can make an error when they type the authenticated pass phrase, thereby delaying the login and raising the users' frustration level.

PUTTING OTP INTO PRACTICE

With your OTP software installed, you need to look at how to put OTP into practice. Your system is protected by OTP; now you need to configure the users. You accomplish this configuration with the keyinit/opieinit program. If the command is being run from a secure console, then you will not require the use of an OTP calculator in the process of configuring OTP for each user. If the configuration cannot be done over a secure terminal, then an OTP calculator will be required to generate a password to be sent across the network.

The first step is to initialize the OTP password for the user, which is illustrated as follows using the command keyinit. Keyinit asks you for the sequence number or iteration value, the key (or seed), and an S/KEY access password. The S/KEY password is generated using the same iteration and seed value, and a pass phrase.

```
host.mycorp.com >   keyinit
Adding chrish:
Reminder you need the 6 english words from the skey command.
Enter sequence count from 1 to 9999: 1000
Enter new key [default nd12043]: heather
s/key 1000 heather
s/key access password: ORB HURD LENT CRAM MELD HOSE
ID chrish s/key is 1000 heather
ORB HURD LENT CRAM MELD HOSE
```

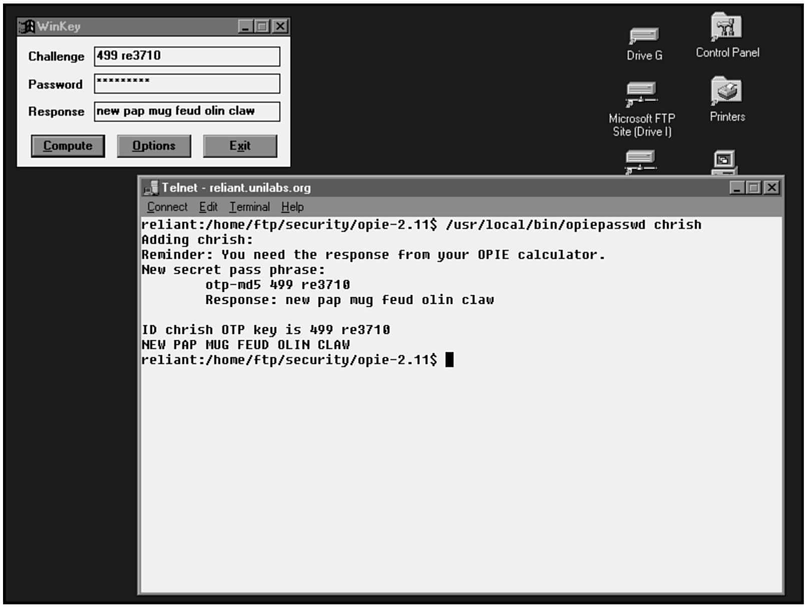
In this example, the keyinit program informs you that the six English words from the skey program, from the OTP calculator, are required. This is because the keyinit program was not started with the -s flag indicating that it is being executed from a secure terminal. Had keyinit been started with the -s flag, it would have assumed that it was being executed on a secure terminal, and your pass phrase would have been requested rather than the S/KEY password.

This means that in order to complete the process, the skey command will have been run previously from a secure terminal, or the user will have access to an OTP calculator on the local machine. This instance is illustrated in figure 4.10.

Figure 4.10 illustrates using the OPIE commands to initialize the user chrish. Notice that the local WinKey calculator was available to provide the six English words.

WARNING

As the administrator of the OTP service, you must inform your users that they should not let their iteration value drop below five. Doing so could lock them out of the system. Getting locked out normally occurs when the iteration value reaches zero, and that value can be reset only by the super user.



Challenge 499 re3710
Password *********
Response new pap mug feud olin claw

```
reliant:/home/ftp/security/opie-2.11$ /usr/local/bin/opiepasswd chrish
Adding chrish:
Reminder: You need the response from your OPIE calculator.
New secret pass phrase:
        otp-md5 499 re3710
        Response: new pap mug feud olin claw

ID chrish OTP key is 499 re3710
NEW PAP MUG FEUD OLIN CLAW
reliant:/home/ftp/security/opie-2.11$ █
```

FIGURE 4.10

Initializing a user with opiepasswd.

Regardless of how good your password is, you should assume that it is known. Do not use the same Unix login password as your OTP password. Change your Unix password after you initialize your OTP passwords to completely eradicate any possibility that a compromised password exists.

Select the number of one-time passwords that you would like to set up, and use your calculator to predetermine the Nth corresponding response. This is illustrated in figure 4.10, which shows the opiepasswd program prompting with the iteration and seed. These values can be user configurable. In this example, however, the opiepasswd program was permitted to use its default values of 499 and a randomly generated seed value.

> It is imperative to note that none of the current OTP calculators detect any typing mistake that you might make. Consequently, it is recommended that, at this state, you calculate the six English words twice to ensure that the value you provided as your pass phrase is correct.

In the example in figure 4.10, opiepasswd initialized an iteration value of 499. Before users reach that limit, they must reinitialize their OTP configuration using a different and unique seed. They can keep the same password, however, if they want. In an earlier example, “heather” was used as the seed; whereas, as figure 4.10 illustrates, using the opiepasswd provided a random seed. Bear in mind that a disadvantage of using system-generated seeds is that they can be difficult to type, creating a higher login failure rate.

SECURITY NOTES REGARDING /BIN/LOGIN

Although an intruder can log into a system, then log in again by running the login command from a shell, most people remain unaware of this almost universal “feature.” Because the second login is from the local host, the utmp entry will not show a remote login host any more. This technique is used by hackers to trick the system administrator into thinking the hacker is a local user, when in fact he is logged in from another site.

The OPIE replacement for /bin/login currently carries on this behavior for compatibility reasons. To prevent this from happening, change the permissions of /bin/login from 4511 to 0100, thus preventing unprivileged users from executing it. This fix should work on non-OPIE /bin/login programs as well. This does not restrict access to the system, nor prevent users from logging in. It only means that once they have logged in, they cannot run the login program.

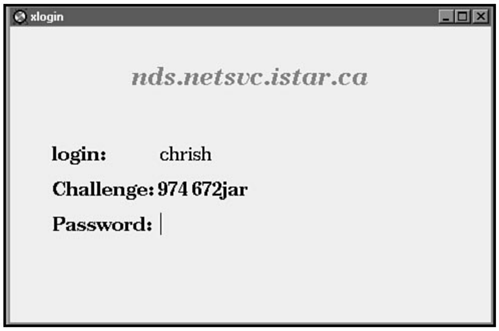
USING OTP AND X WINDOWS

The major concern for users of the X Window environments is how to integrate OTP into their environments. The typical X Display Manager does not know how to interact with the OTP challenge/response format during the login session. An implementation of XDM with the OTP capability has been produced, however. The complete source code for the Release 6/Fix 13 version of XDM with S/KEY capability is available from the following location:

`ftp://cs.anu.edu.au/pub/people/Hugh.Fisher/skey-xdm.tar.Z`

Figure 4.11 illustrates how this implementation of XDM looks when it is in operation.

The implementation is robust and, like the login program, it supports password echoing. When you press the Enter key, you activate character echoing, which enables you to see the password as it is typed. This implementation was produced by Hugh Fisher of the Computer Science Department, Australian National University, and is freely available.



nds.netsvc.istar.ca

login: chrish

Challenge: 974 672jar

Password:

FIGURE 4.11

Adding OTP to the XDM login process.

GETTING MORE INFORMATION

Other sources of ongoing information and mailing lists that discuss S/KEY and its continuing development are available. One mailing list is sponsored directly by Bellcore. Bellcore uses this mailing list to announce its own changes to the S/KEY code, as well as for bug reporting and any general discussions related to S/KEY authentication.

If you want to be added or deleted from this list, send mail to the following address:

`skey-users-request@thumper.bellcore.com`

To send a message to the mailing list, send mail to the following address:

`skey-users@thumper.bellcore.com`

This mailing list is manually administered. You will not find a listserv or majordomo on the receiving end of your message. This means that the traditional listserv or majordomo command such as SUBSCRIBE does not work with this list. The list itself is for implementation issues and general discussion surrounding S/KEY.

Another mailing list, `Ietf-otp@bellcore.com`, is used for following the discussions of the IETF group working on one-time passwords. If you want to be added to or removed from the list, send a message to the following address:

`ietf-otp-request@bellcore.com`

To send a message to the list, use this address:

`ietf-otp@bellcore.com.`

SUMMARY

The development of OTP and the ongoing maintenance and support for a freely available product of this nature has made a difference in network security. After OTP is implemented, it is virtually impossible for a hacker to attack through the password security system. In fact, one company that uses OTP has field personnel access equipment using a one-time password that is generated for them from the central site. The field personnel do not know the pass phrase. In this situation, the field personnel contact their central office, at which time the OTP password is given to them.

The OTP-secured system is considered safe for high-security applications according to the current Internet Drafts and the RFC documents on the MD4 and MD5 encryption methods. Implementing OTP is a wise practice, and the success of the implementation might be dependent upon the use of tools from the several different tool sets that have been presented in this chapter.