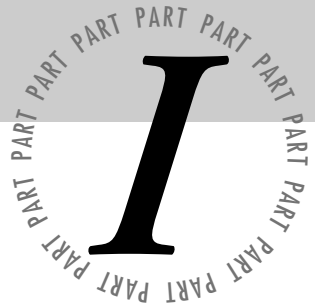


Managing Internet Security

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. Although 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 Hostnames

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).

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).

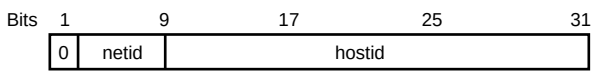


Figure 1.1
The class A address format.

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.

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.

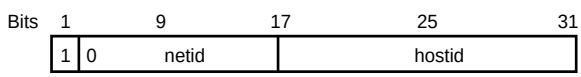


Figure 1.2
The class B address format.

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.

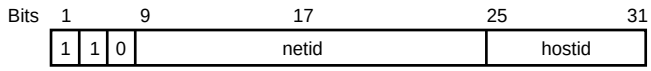


Figure 1.3
The class C address format.

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 has 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 the following table, 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

To fully understand and appreciate how this works, 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, you must use some of the bits in the host portion, or last byte, of the address as part of the network portion. Although 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.

```
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:

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 zero. 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!

Hostnames

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

Note Methods for translating the hostname into the corresponding IP address and DNS are discussed later in this chapter.

Within each organization, the hostname must be unique. The hostname consists of two pieces: the actual hostname 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 hostnames instead of IP addresses is that they are easier to use. If hostnames 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 hostname typically is renamed in the future, which can sometimes be more difficult than it sounds.

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

The hostname 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 network types:

Media Type
Token Ring
Thinnet (RG-58U Coax 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.

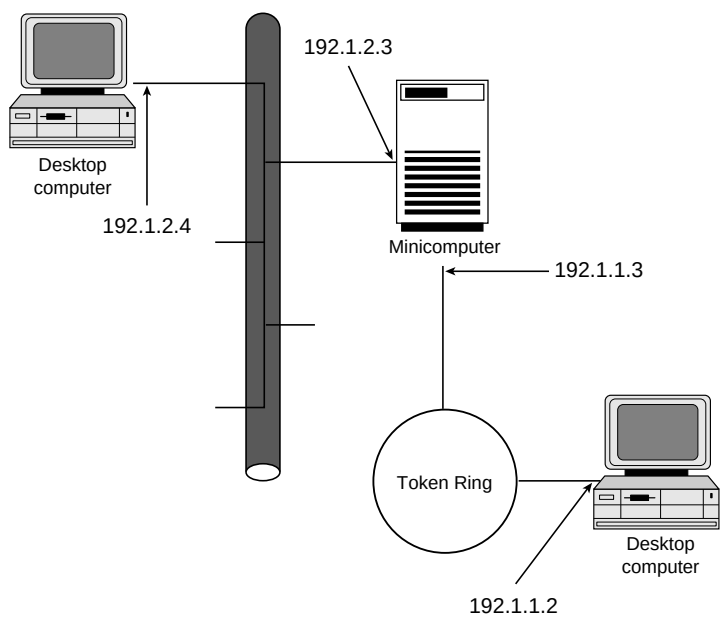


Figure 1.4
Network interfaces.

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	SunOS 4.1.3
wdn0,e3a,sl0,ppp1	SCO Unix
du0	DEC Ultrix

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 as follows:

```
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 hostname that is found in /etc/hosts, or an Internet address expressed in dot notation. If the name form is used and the hostname 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.

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.
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 198.73.138.6
```

The options used on the SunOS system are the same as with SCO systems, except that the netmask defined on the Sun system uses a hexadecimal notation rather than 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 a simple hostname to IP address resolution. Remember that TCP/IP only requires the use of IP addresses. The use of hostnames is for your convenience and ease of use. When a hostname 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:

```
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 sample hosts file shown here:

```
# 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
```

The aliases include the short form of the hostname, 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.

Note 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 hostname, as illustrated here:

```
# Ethernet Address      Hostname
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 shown here:

```
# 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/service 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.telentd   in.telentd
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.

/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
```

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

The .rhosts File

The .rhosts file that is used in the user's 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.

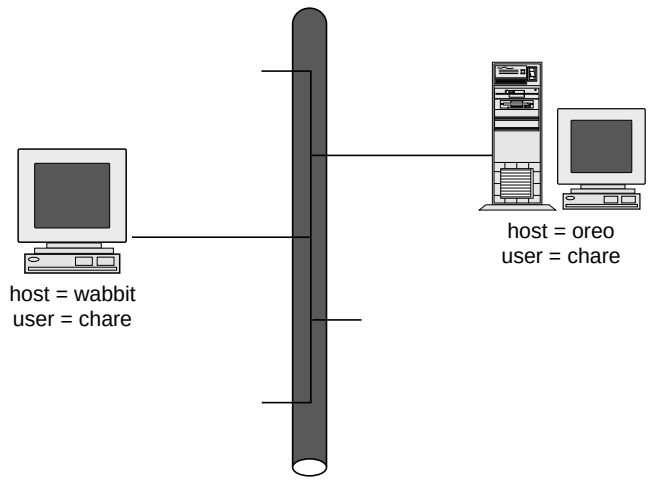
User and Host Equivalency

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

Tip 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 (#).

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 will be explained later. Figure 1.5 will be used in the discussion of user equivalency.

Figure 1.5
A sample network.



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.

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.

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 found only 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 routed 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.

The Domain Name Service Daemon (named)

named is the Internet Domain Name Server, and it is the second mechanism available to provide hostname 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 hostname. 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 hostname 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.

The System Logger Daemon (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
- 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.

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 Control+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 1.5.

Table 1.5
ping 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.

continues

Table 1.5, Continued
ping Options

Option	Description
	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 hostname. 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.
-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 here.

```
$ 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 hostname, but the following options alter the output format of ruptime.

Table 1.6
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.
```

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.
$
```

The following table lists the options that typically are available on the finger command.

Table 1.7
finger Options

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 1.8.

Table 1.8
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 1.9.

Table 1.9
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   478820     190677      706      0
dblks                     4054     279   377515     804030      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 1.10.

Table 1.10
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_RECIEVED	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 shutdown. The acknowledgment of the close is pending.
LAST_ACK	The rmote 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 an ICMP PORT UNREACHABLE message.

In the output of the traceroute command in the preceding example, the times following the hostname 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.

Note 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.

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 hostname 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 1.11.

Table 1.11
arp 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	Creates an entry in the ARP table for the named [temp] [pub] [trail]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.

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. Although 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 falls 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.

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 filename of the form "rhost:path", or a local filename (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 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 hostname 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 1.12.

Table 1.12
rsh 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 /usr/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 hostname. When telnet is started with a hostname, 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 Control+], 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 log out 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.
$
```

Although 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 usernames, 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 because 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:

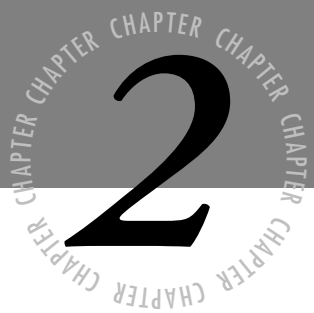
```
$ ftp yosemite.widgets.ca
Connected to yosemite.widgets.ca.
220 yosemite.widgets.ca FTP server (Version 5.60 #1) ready.
Error - .netrc file not correct mode.
Remove password or correct mode.
Remote system type is Unix.
Using binary mode to transfer files.
ftp> quit
221 Goodbye.
$ ls -l .netrc
-rw-r--r--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

Tip 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.



Understanding and Creating Daemons

The most secretive, yet most productive, application or service on a Unix system is the daemon process. A daemon, pronounced "demon," process is secretive because it runs in the background, and often does not indicate its presence in any significant way. Without it, most Unix systems would cease to function. Programmers write daemons to carry out a function with little or no intervention by users or system administrators. In fact, many daemons require no intervention at all!

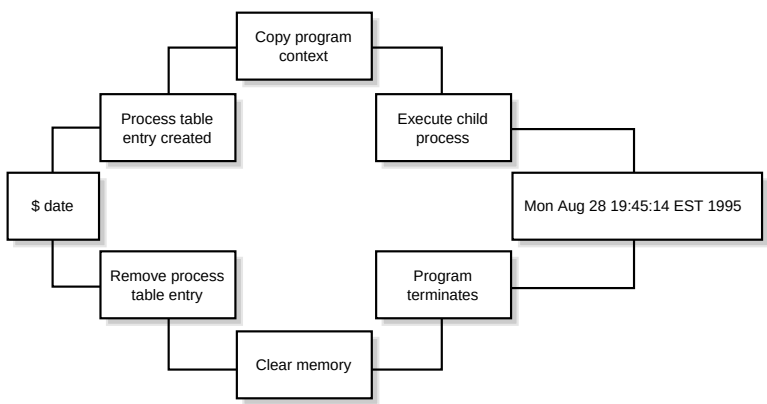
The services offered by daemon processes are important to understand, because the potential security violation may be through a program that masquarades as a daemon.

What Is a Daemon?

A *daemon process* is a process that is not associated with a user, but performs system-wide functions, such as administration and control, network services, execution of time-dependent activities, and print services. To qualify as a daemon process, several criteria must be met: the process must not be associated with a user's terminal session; and it must continue after the user logs off.

From the rudimentary process management knowledge you have read about so far, you know that each process a user starts is terminated by the init program when the user exits. The init program is the most famous of all system daemons. This approach, illustrated in figure 2.1, allows for proper management of the process table.

Figure 2.1
The process life cycle.



Although daemon processes are almost completely invisible, they do provide some level of service to users. Daemon processes accept user requests and process them; they also respond to various events and conditions. They are often inactive, however, and are designed to be called into service only when required. By using a daemon instead of starting a new process for every instance, system load is reduced, and large programs that take time to get started will not slow down the user or the operation.

A daemon can be distinguished from other programs on the system by examining the process table—the ps command displays this table. The distinguishing characteristic of a daemon is that the TTY column does not reflect the controlling terminal name. The following portion of the process table shows this difference:

```
nms# ps -aux ¦ more
USER       PID %CPU %MEM   SZ  RSS TT STAT START  TIME COMMAND
root       257  7.9  0.0   12    8 ?  S    Aug 22 47:24 update
root         1  0.0  0.0   52    0 ?  IW   Aug 22  0:02 /sbin/init -
root       289  0.0  0.0   40    0 ?  IW   Aug 22  0:00 - sxp.9600 ttya (getty)
root        79  0.0  0.0   16    0 ?  I    Aug 22  0:00  (biod)
root         2  0.0  0.0    0    0 ?  D    Aug 22  0:00 pagedaemon
```

```
root        51  0.0  0.0   68    0 ?  IW   Aug 22  0:25 portmap
root        56  0.0  0.7   84  212 ?  S    Aug 22  1:15 ypserv
root       288  0.0  0.0   40    0 co IW   Aug 22  0:00 - cons8 console (getty)
bin         58  0.0  0.0   36    0 ?  IW   Aug 22  0:00 ypbind
root         0  0.0  0.0    0    0 ?  D    Aug 22  1:31 swapper
root        60  0.0  0.0   40    0 ?  IW   Aug 22  0:00 rpc.ypupdated
root        73  0.0  0.5   48  140 ?  S    Aug 22  1:01 in.routed
root        76  0.0  0.4  216  128 ?  S    Aug 22  0:38 in.named
root       120  0.0  0.0   28    0 ?  I    Aug 22  0:00  (nfsd)
root        93  0.0  0.4   68  120 ?  S    Aug 22  1:14 syslogd
root       101  0.0  0.0  160    0 ?  IW   Aug 22  0:02 /usr/lib/sendmail -bd -q
root        62  0.0  0.0   40    0 ?  IW   Aug 22  0:00 keyserv
root       119  0.0  0.0   72    0 ?  IW   Aug 22  0:00 rpc.lockd
```

The daemon is the process with a question mark "?" as the controlling terminal name. The controlling terminal is identified in the "TT" or "TTY" column of the ps output. Whenever this is found in a process entry, the process is a daemon. You can see that most of the processes in this part of the process table are in fact daemon processes.

Daemon processes usually do not accumulate very much CPU in the short run, unless they have a lot of processing to do when they start. It usually takes a tremendous amount of time for these daemon processes to equal the CPU requirements that many other processes accumulate in a minute or two.

The daemon processes shown in the ps output were likely started as part of the system's boot process. The files required to boot the system and start these daemons for the SunOS 4.1.3 and SunOS 4.1.4 systems are listed in table 2.1.

Table 2.1
SunOS 4.1.x Startup Daemons

File Name	Daemon	Description
/etc/rc	update	Periodically updates the super block
	cron	Executes commands at specified dates and times
	in.rwhod	System status server
	inetd	Internet services daemon
	lpd	Printer daemon
/etc/rc.local	portmap	TCP/IP port to RPC program number mapper
	ypserv	NIS server
	ypxfrd	NIS transfer server
	rpc.ypupdated	NIS update server
	ypbind	NIS domain binding agent
	keyserv	Server for storing public and private keys
	in.routed	Network routing daemon
	in.named	Internet domain name server

continues

Table 2.1, Continued
SunOS 4.1.x Startup Daemons

File Name	Daemon	Description
	biod	Asynchronous block I/O daemons
	syslogd	Logs system messages
	auditd	Controls the generation and location of audit trail files
	sendmail	Sends mail over the Internet
	ndbootd	ND boot block server
	nfsd	Client file system requests
	rpc.mountd	NFS mount request server
	rarpd	TCP/IP Reverse Address Resolution Protocol server
	bootparamd	Boot parameter server
	rpc.statd	Network status monitor
	rpc.lockd	Network lock daemon
	automount	Automatically mounts NFS file systems
	snmpd	Daemon that responds to SNMP requests

As you can see in the SunOS 4.1.x configuration, an extensive list of daemons are started to service users and provide system-level services. Notice that these daemons are started from only two startup files. This setup makes the maintenance of the SunOS system easier because the startup routines aren't scattered all over the place. Obviously, this is an important benefit when you need to make changes to the startup process, such as add new daemons, modify the operation of existing ones, or remove unneeded daemons.

It is important to consider that the startup procedures of the various Unix flavors often are very different depending upon the heritage. SunOS 4.1.x, for example, is derived from the Berkeley Software Distribution (BSD) code and as such bears little or no resemblence to the startup procedure seen in Solaris 2.x, which is based upon the Unix System Laboriatories Unix System V Release 4.

The same is true when comparing Unix System V Release 3.2 and Release 4.0 These differences are important to note, because they make it easier to hide inconspicuous programs for later action.

Table 2.2 lists daemons that are used to start and operate the HP-UX operating system on a Hewlett-Packard HP9000 Series 700 workstation.

Table 2.2
HP-UX Startup Daemons

File Name	Daemon	Description
/etc/rc	lpsched	Printer daemon
	cron	Executes commands at specified dates and times
	vtdaemon	Responds to vt requests
	envd	System physical environment daemon
	rbootd	Remote boot server
	syslogd	Logs system messages
/etc/netlinkrc	nettl	Controls network tracing and logging
	inetd	Internet services daemon
/etc/audiorc	Aserver	Audio server
/etc/netbsdsrc	gated	Network routing daemon
	named	Internet domain name server
	rwhod	System status server
	sendmail	Sends mail over the Internet
/etc/netnfsrc	portmap	TCP/IP port to RPC program number mapper
	ypserv	NIS server
	ypbind	NIS domain binding agent
	rpc.mountd	NFS mount request server
	nfsd	Client file system requests
	biod	Asynchronous block I/O daemons
	pcnfsd	(PC)NFS authentication and print request server
	rpc.lockd	Network lock daemon
	rpc.statd	Network status monitor
/etc/netnmrc	snmpd	Daemon that responds to SNMP requests

The HP-UX startup sequence makes use of a large number of files, each of which are tightly linked to a given subsystem. For example, the file netlinkrc is used to start the network processes. With this type of startup file layout, it is much harder to locate the daemons and to modify the system startup procedure.

Regardless of the Unix implementation being considered, the use of the /etc/rc file to start the system is common. Consider the list of files required to start the daemons on an SCO OpenServer 5.0 system. Table 2.3 lists the daemons and their start up file locations.

SCO Unix products use a file system structure that is grouped by the desired run level. Run levels, their meanings, and how to switch between them are discussed in the section, “Unix Run Levels.”

Table 2.3
SCO Unix Startup Daemons

Filename	Daemon	Description
/etc/rc2.d//01MOUNTFSYS	uditd	Reads audit collection files generated by the audit subsystem and compact the records
/etc/rc2.d/P75cron	cron	Executes commands at specified dates and times
/etc/rc2.d/P86mmdf	deliver	Handles the management of all mail delivery
/etc/rc2.d/S80lp	lpsched	Printer daemon
/etc/rc2.d/S84rpcinit	portmap	TCP/IP port to RPC program number mapper
	rwalld	Network rwall server
	rusersd	Network user name server
	rexd	RPC-based remote execution server
/etc/rc2.d/S85nis	ypserv	NIS server and binder processes
	ypbind	NIS server and binder processes
/etc/rc2.d/S85tcp	maskreply	Sends gratuitous ICMP mask reply
	syslogd	Logs system messages
	inetd	Internet services daemon
	snmpd	Daemon that responds to SNMP requests
	named	Internet domain name server
	routed	Network routing daemon
	irdd	Internet Router Discovery daemon
	gated	Network routing daemon
	rarpd	TCP/IP Reverse Address Resolution Protocol server
	timed	Time server daemon
	rwhod	System status server
	lpd	Printer daemon
/etc/rc2.d/S89nfs	nfsd	Client file system requests
	mountd	NFS mount request server
	pcnfsd	(PC)NFS authentication and print Request server
	biod	Asynchronous block I/O daemons
	automount	Automatically mounts NFS file systems
	statd	Network status monitor
	lockd	Network lock daemon

Like the HP-UX implementation, a number of SCO Unix startup scripts are used to start daemons. Each script essentially is dedicated to starting the daemons for a specific function group. This is not necessarily bad design, but it requires a detailed level of understanding of the underlying system structure.

The following sections examine what each of these daemons offers to the system and to the users of that system.

Examining the System Daemons

A number of system daemons can exist in a Unix system. Some are only found in a specific version of Unix, but many daemons are common to all versions of Unix. This section discusses many of the common daemons and describes their function on the system.

init

The *init daemon* is known as the parent process for all the processes on the system. It performs a broad range of functions that are vital to the operation of a Unix system.

The most commonly known purpose of the init process is to boot the system. The method init uses to boot the system differs among Unix versions. The BSD and XENIX init programs, for example, do not work the same way as the System V implementation. The System V init program relies on the file /etc/inittab to provide details of how init is to govern the startup and initialization of the various services on the system. The init process is commonly known as "init" because of its role in the initialization of various processes during system operation.

The init program considers the system to be in a run level at any given time. *Run levels* are the operating states of the system. For the purposes of this section, a run level can be viewed as a software configuration; each configuration allows only a selected group of processes to exist.

swapper

Some Unix system administrators refer to swapper as a daemon, and others do not. The *swapper* process is responsible for scheduling the use of memory by the various processes on the system. The swapper process is actually part of the kernel, so you could say that it is not a daemon after all.

update and bdflush

update and *bdflush* are similar commands that periodically execute the sync system call to flush disk buffers. These daemons execute every 30 seconds. Users and system administrators rely on these daemons to update the file system in case of a crash. Although two commands are listed, your system will see one or the other, but rarely both.

lpd

The *lpd daemon* is part of the BSD print services. It listens for and accepts connections via TCP/IP to submit a print request. The lpd daemon relies on the LPD protocol to accept the job, and submit it to the requested printer. This daemon was almost exclusively found on BSD-based systems until the more popular System V derivatives started adding similar services.

Note Some System V implementations have an lpd daemon but still require the use of the System V print spooler for the job to be printed.

lpsched

The *lpsched daemon* is the System V version of the print spooler. It performs the same tasks as the BSD lpd program, but in a much different format. Despite lpsched's inability to communicate directly via the LPD protocol, it is still considered stronger than lpd because of its flexibility with printer interface scripts.

Note Some lpsched implementations, such as found on Solaris 2.x, are capable of receiving LPD requests.

cpd and sco_cpd (SCO)

The *cpd* and *sco_cpd daemons* are the license managers for SCO products. They are similar to license managers on other implementations of Unix in that they ensure that all products on the local network have unique serial numbers. With the release of SCO OpenServer 5.0, the license managers support shrink-wrapped software and operating system software.

cron

The *cron daemon* is the automated task scheduler; it runs scheduled jobs at the requested time. A user may want to execute a number of jobs at regular intervals, for example. To do this, a crontab file is created resembling the following:

```
0,15,30,45 * * * * /usr/stats/bin/getstats border1.ottawa
0 3 * * 0 /usr/stats/bin/merge border1.ottawa
0 4 * * 0 /usr/stats/bin/ar border1.ottawa
```

This specification identifies when the job is to be executed and what the command to be executed is. The cron daemon builds an internal list of the jobs to be executed, and runs them at the requested time intervals.

syslog

The *syslog daemon* is a UDP/IP service that allows information and status messages for different network services to be logged through a central logging mechanism. The syslog daemon is controlled through the file /etc/syslog.conf and can write messages of different types into different log files. A sample syslog.conf file is shown here:

```
user.*          /usr/log/user_logs
kern.*          /usr/log/kernel_logs
daemon.*        /usr/log/messages
mail.debug      /usr/log/mail
lpr.debug       /usr/log/mail
cron.debug      /usr/log/cron
news.debug      /7usr/log/news
auth.*          /usr/log/authenticate
local3.debug    /usr/log/wrapper
local7.debug    /usr/log/backbone
*.critical      /usr/log/critical
*.emerg                 *
```

Note Although you may have more than 16 entries in your syslog configuration file, many implementations of syslog can only open a maximum of 16 log files.

The syslog.conf file lists the facility priority level of the message, and where that message is to be stored when received. Any message that is received with a priority level of critical, for example, is written to the file /usr/log/critical.

syslogd reads and forwards system messages to the appropriate log files, to users, or to both, depending on the priority of a message and the system facility from which it originates. The following output lists sample syslog entries that show different types of information captured by syslogd.

```
Aug 26 05:21:37 nms in.tftpd[14037]: connect from C7-1.vcr.home.org
Aug 26 09:47:03 nms sendmail[14344]: AA14344: message-id=<9508261345.AA14344@nms
➥.home.org>
Aug 26 09:47:03 nms sendmail[14344]: AA14344: from=stats, size=149, class=0
Aug 26 09:47:05 nms sendmail[14347]: AA14344: to=stats@home.org, delay=00:01
➥:32, stat=Sent
Aug 26 11:00:01 nms cron: >  CMD: 14426 c /usr/stats/bin/getstats border1.
➥.montreal
Aug 26 11:00:01 nms cron: >  stats 14426 c Sat Aug 26 11:00:01 1995
Aug 26 11:00:01 nms cron: <  noc 14421 c Sat Aug 26 11:00:01 1995 Exit status 1
Aug 26 11:00:53 nms cron: <  stats 14422 c Sat Aug 26 11:00:53 1995
Aug 26 11:01:39 nms cron: <  stats 14423 c Sat Aug 26 11:01:39 1995
Aug 26 11:02:02 nms cron: <  stats 14426 c Sat Aug 26 11:02:02 1995
Aug 26 11:04:33 nms cron: <  stats 14425 c Sat Aug 26 11:04:33 1995
```

The sample log entries also show you what information is saved by syslog: a time stamp, the name of the machine where the message originated, the command and PID, and the message.

Note Manual syslog entries can be made using the logger command, discussed later in this chapter.

sendmail

The *sendmail daemon* is the common Mail Transport Agent included with current versions of Unix. Because this program is a daemon, it listens for and accepts incoming e-mail connections from external systems. This daemon receives and subsequently delivers messages to local or remote users. sendmail is not intended to function as a user interface, but rather as the processing agent for user mail programs such as elm, pine, mailx, and mush.

The sendmail program functions in two modes: incoming and outgoing. It accepts mail from internal and external sources and processes it according to the rules found in the /etc/sendmail.cf configuration file. The format of and options for the /etc/sendmail.cf configuration file are far too complex to cover here.

The sendmail program is capable of accepting TCP/IP connections on port 25. The following output illustrates a connection to sendmail on this port.

```
nms% telnet nms 25
Trying 198.53.64.4 ...
Connected to nms.
Escape character is '^]'.
220 nms.home.org Sendmail 4.1/ch-950121.1 ready at Sat, 26 Aug
95 11:28:36 EDT
help
214-Commands:
214-    HELO    MAIL    RCPT    DATA    RSET
214-    NOOP    QUIT    HELP    VRFY    EXPN
214-For more info use "HELP <topic>".
214-smtp
214-To report bugs in the implementation contact Sun Microsystems
214-Technical Support.
214-For local information contact postmaster at this site.
214 End of HELP info
quit
221 nms.home.org closing connection
Connection closed by foreign host.
nms%
```

The system administrator can test his or her configuration from the sendmail command directly. Unfortunately, this capability can also be used by the wily hacker to create a false mail message that looks like it came from somewhere else.

getty

The *getty daemon* is responsible for providing a login prompt on terminals and on serial devices directly connected to the system; getty is also responsible for providing a login prompt on the console. The getty command is started by the init process, and is part of the login->shell->logout process. It is important to note that when you log in through telnet, getty is not involved in the process. The telnet server, telnetd, displays the login message and collects the user name from the user.

rlogind

The *rlogind daemon* is the server side to the client rlogin program. It provides a remote login facility with authentication based on privileged port numbers and hostname-username pairs. rlogind is executed by the Internet daemon, inetd, when it receives a service request at the port indicated in the services database for login using the TCP/IP protocol.

deliver

The *deliver daemon* manages all mail delivery in the MMDF mail system. deliver does not deliver mail directly, but instead calls on MMDF channel programs to handle actual delivery. deliver's actions are guided by the MMDF configuration file, /usr/mmdf/mmdftailor, and by command-line options. This daemon also maintains a cache of host information on a per-channel basis, so that mail for unavailable hosts can be skipped until the host is available.

inetd

The *inetd daemon* listens on multiple ports for incoming connection requests. When it receives a request, inetd spawns the appropriate server. The use of a "super-server" allows other servers to be spawned only when needed and to terminate when they have satisfied a particular request. The following servers are normally started by inetd: fingerd, ftpd, rexecd, rlogind, rshd, talkd, telnetd, and tftpd. inetd can also start several internal services: these are described in inetd.conf, which is typically found in the /etc directory. Do not arrange for inetd to start named, routed, rwhod, sendmail, pppd, or any NFS server.

routed

The *routed daemon* is invoked by root at boot time to manage the Internet Routing Tables (usually during init 2). The routed daemon uses a variant of the Xerox NS Routing Information Protocol to maintain up-to-date kernel Routing Table entries. If the host is an internetwork router, routed periodically supplies copies of its Routing Tables to hosts and networks that are directly connected.

nfsd

The *nfsd daemon* starts the NFS server daemons that handle client file system requests. The nfsd daemon is a user application entry point into the kernel-based NFS server. Depending on the option or options used, server daemons are started to handle:

- Only NFS requests sent over UDP.
- Only NFS requests sent over TCP.
- UDP requests. Depending on the option or options used, server daemons are started to handle only NFS requests sent over UDP or only NFS requests sent over TCP.

mountd

The *mountd daemon* is an RPC server that responds to file system mount requests. It reads the file /etc/exports to determine which file systems are available to which machines and users. This daemon also provides information regarding clients with mounted file systems. This information can be printed using the showmount command.

pcnfsd

The *pcnfsd daemon* is an RPC server that supports ONC clients on PC (DOS, OS/2, and Macintosh) systems. There are two implementations of the PC-NFS protocol: Version 1 and Version 2. Version 2 supports extended printing features. It reads the configuration file /etc/pcnfsd.conf if present, and then services RPC requests directed to program number 150001. Many releases of the pcnfsd daemon support both version 1 and version 2 of the pcnfsd protocol.

statd, rpc.statd

The *statd* and *rpc.statd daemon*s are RPC servers that function as the RPC status monitor. It interacts with the lockd server to provide crash and recovery functions for the locking services on NFS. It is common to see either statd or rpc.statd but not both on your system.

lockd, rpc.lockd

The *lockd daemon* processes lock requests that are either sent locally by the kernel or remotely by another lock daemon. lockd forwards lock requests for remote data to the server site's lock daemon. lockd then requests the status monitor daemon, statd or rpc.statd, for monitor service. The reply to the lock request will not be sent to the kernel until the status daemon and the server site's lock daemon have replied.

Creating Daemons with the Bourne Shell

If you are a system administrator, you probably have no problems thinking up countless programs you would like to have. You also probably know of a number of tasks you wish were simpler.

In many cases, a daemon can be written to do the job for you. To help you understand how to write a daemon using the Bourne or Korn shells, suppose you want to build a monitor program.

For a system administrator, one of the most precious resources is disk space. If a root file system is filled, it can result in a severe failure or yet another minor annoyance. Fortunately, you can build a monitor program that runs a daemon to monitor and report on available disk space.

Before you dig in working with code, plan on setting some parameters for the program:

- It must be able to read a configuration file.
- It must be able to handle System V and BSD versions of df.
- It must be able to function with or without the Unix syslog facility.

With these requirements in mind, consider how this daemon will be implemented in the Bourne shell. The purpose of this exercise is not to teach programming in the Bourne shell, but to identify issues you need to consider when writing a daemon, such as handling input and output, and signals.

Handling Input and Output

The daemon process often is not associated with any specific terminal, so it must be able to write messages to a screen somewhere if desired. Assume that this program will be operating as the root user, and therefore will be able to access the system console.

Remember that three file descriptors are opened for each shell: standard input, which is normally the keyboard; standard output, which is normally the monitor; and standard error, which is used for error messages and is also directed to the monitor.

```
0    standard input
1    standard output
2    standard error
```

To send a message to standard error, you indicate the file descriptor that the message is to be sent to by using the following notation:

```
    echo "error!" >&2
```

This example prints “error!” on the standard error device.

This daemon will not need access to standard input for any reason, so you can close standard input, and still keep standard output and standard error available if required. By closing standard input you save a file descriptor that would otherwise be taken in the open file table; you also prevent users from being able to thwart the daemon by attempting to interact with it.

To close file descriptors use the notation:

```
exec >&-
```

This closes the standard output file. To close standard input, use the notation:

```
exec <&-
```

Handling Messages

You now need to consider where messages will be sent when the program generates them. One setup is to specify a device in the program where messages will be sent, such as /dev/console. The following syntax redirects both standard error and standard output to /dev/console:

```
exec >/dev/console
```

or

```
exec 2>&1 >/dev/console
```

Handling Signals

Handling signals is also important because you want to prevent the program from terminating for any reason other than a system shutdown. To do this, you must use the kill command to trap signals that can be sent to the script.

Programmers often use signals as a mechanism of communicating with the program. The program’s response to the signal can be to ignore the request, perform the default action, or to perform some other action. The command to trap signals is trap:

```
trap "" signal numbers
```

This sample command executes the command between the quotes when the specified signal number is received. Because the command list is empty, the signal is ignored. Table 2.4 lists different signals.

Table 2.4
Standard Unix Signals

Name	Number	Description
SIGHUP	1	Hangup
SIGINT	2	Interrupt
SIGQUIT	3	Quit
SIGILL	4	Illegal instruction
SIGTRAP	5	Trace trap
SIGABRT	6	Abort
SIGEMT	7	Emulator trap
SIGFPE	8	Arithmetic exception
SIGKILL	9	Kill (cannot be caught, blocked, or ignored)
SIGBUS	10	Bus error
SIGSEGV	11	Segmentation violation
SIGSYS	12	Bad argument to system call
SIGPIPE	13	Write on a pipe or other socket with no one to read it
SIGALRM	14	Alarm clock
SIGTERM	15	Software termination signal
SIGURG	16	Urgent condition present on socket
SIGSTOP	17	Stop (cannot be caught, blocked, or ignored)
SIGTSTP	18	Stop signal generated from keyboard
SIGCONT	19	Continue after stop
SIGCHLD	20	Child status has changed
SIGTTIN	21	Background read attempted from control terminal
SIGTTOU	22	Background write attempted to control terminal
SIGIO	23	I/O is possible on a descriptor
SIGXCPU	24	Cpu time limit exceeded
SIGXFSZ	25	File size limit exceeded
SIGVTALRM	26	Virtual time alarm
SIGPROF	27	Profiling timer alarm
SIGWINCH	28	Window changed
SIGLOST	29	Resource lost
SIGUSR1	30	User-defined signal 1
SIGUSR2	31	User-defined signal 2

This list may not include all the available signals for your implementation of Unix. To see a definitive list of all the signals in use by your system, check the file /usr/include/signal.h or use the man command to read the online manual page for the signal library function.

The dfmon Program

The program that you have read about to this point is shown in Listing 2.1 and Listing 2.2 at the end of the chapter. Listing 2.1 contains the entire Bourne shell source code for the daemon. It has been tested on numerous platforms including SunOS 4.1.3 and 4.1.4, SCO OpenServer 5.0, HP-UX 9.0, SCO Unix 3.2 Version 4.0, and BSDI Version 2.0.

This Bourne shell daemon sends little or no output to the standard input/output channels. With the exception of text that is printed during the startup of the program, the only other text generated is saved through the syslog service, or into a file directly.

Your dfmon program is added to the system startup scripts for the machine on which it will run. The script could be /etc/rc.local or /etc/rc2.d/S89dfmon. The exact location of the startup files differs from one Unix implementation to another.

The dfmon program starts by reading the configuration file that is defined in the shell script. This configuration file defines the operating environment for the script. The variables defined in the configuration file are shown in Listing 2.2. dfmon puts most of the system specific components of the program inside a shell function for each specific operating system. The majority of the differences among Unix implementations are in the df output, which is a direct problem for this script.

When the df output for the file system has been determined, it is examined and compared with the low and critical free space values from the configuration file. The available space is checked first against the value for the critical level. If the available space is less than the desired amount, the logger command is used to send a warning message to the syslog daemon, or a warning is sent to the system user using the wall command.

If the available disk space is greater than the critical level, then the low level is checked. If the value is less than the low level for the file system, the same procedure is followed.

The logger command provides the capability for non-compiled programs to send messages to the syslog daemon for logging and potential action. The command line looks like this:

```
/usr/bin/logger -t "dfmon" -p $SYSLOG_FAC.crit "CRITICAL Alarm : $FILESYS space:
➥Free=$FREE"
```

This command line creates the following message in the system syslog file:

```
Aug 26 15:16:39 hp dfmon: CRITICAL Alarm : / space: Free=1093530
```

This program enables any system administrator to record important system events in the syslog. Remember though, that if the syslog daemon is not running when the logger command is executed, the information sent by the logger is ignored.

For systems that do not have syslog, or when the system administrator chooses not to use syslog, this daemon is capable of saving the information to a file and sending a warning message to users who are logged in using the wall command, as seen in this example:

```
Sun Aug 27 17:47:35 EDT 1995 Starting Disk Monitor Daemon ....

Broadcast Message from chrish (pty/ttys0) Sun Aug 27 17:47:35...
Sun Aug 27 17:47:35 EDT 1995 hp dfmon: CRITICAL Alarm : / space: Free=1093048
```

This section has shown you how to create a daemon using the Bourne/Korn shell language. Many system administrators, however, do not use the shells for this type of activity because it still relies on the services of other programs, which may not be available. For this reason, you also need to know how to create daemons using PERL.

Creating Daemons with PERL

The PERL programming language, a cross between the capabilities and syntax of the Bourne/Korn shell and the syntax and facilities of C, is a strong environment for building daemons. Like the earlier discussion, this section uses a practical example to help you follow the construction of a daemon in PERL. However, this is not meant as a tutorial for the PERL language. This section considers similar issues as the Bourne shell, but with the idiosyncrasies of the PERL language.

For administrators responsible for maintaining a Unix system, a large amount of concern is focused on the currently executing processes. This is especially true of processes that are started when the system is brought up that should never exit. No matter how hard you try to keep these processes running, sometimes it just isn't possible. Several reasons include maximum parameters being reached, programming errors, or a system resource that isn't available when needed.

One way of combating this situation, which may not work depending upon the process involved, is to use the /etc/inittab file on System V systems. The /etc/inittab file contains a list of processes that are to be executed when the system enters a run level. This file also contains information on what to do when the process exits. A sample /etc/inittab file is shown here:

```
ck:234:bootwait:/etc/asktimerc </dev/console >/dev/console 2>&1
ack:234:wait:/etc/authckrc </dev/console >/dev/console 2>&1
brc::bootwait:/etc/brc 1> /dev/console 2>&1
mt:23:bootwait:/etc/brc </dev/console >/dev/console 2>&1
is:S:initdefault:
r0:056:wait:/etc/rc0  1> /dev/console 2>&1 </dev/console
```

```
r1:1:wait:/etc/rc1  1> /dev/console 2>&1 </dev/console
r2:2:wait:/etc/rc2 1> /dev/console 2>&1 </dev/console
r3:3:wait:/etc/rc3  1> /dev/console 2>&1 </dev/console
sd:0:wait:/etc/uadmin 2 0 >/dev/console 2>&1 </dev/console
fw:5:wait:/etc/uadmin 2 2 >/dev/console 2>&1 </dev/console
rb:6:wait:/etc/uadmin 2 1 >/dev/console 2>&1 </dev/console
co:2345:respawn:/etc/getty tty01 sc_m
co1:1:respawn:/bin/sh -c "sleep 20; exec /etc/getty tty01 sc_m"
```

The inittab file consists of four colon separated fields:

```
 ck:234:bootwait:/etc/asktimerc </dev/console >/dev/console 2>&1
➥identifier:run levels:action:command
```

This is a powerful capability, but it is found only on System V variants of Unix. Another drawback is that there is no indication with /etc/inittab and the init command that the process has exited and restarted unless the process continuously dies and init prints a message on the console. The message typically is something like `Command is respawning too rapidly`. How can you provide a system independent method of monitoring and restarting critical system processes? The answer is procmon.

Note BSD-based operating systems, such as SunOS, BSD/OS from BSD Inc., and FreeBSD do not use /etc/inittab to control processes spawned by init.

Handling Input and Output

As with the shell, PERL uses the same three standard input and output files: standard input, known as STDIN; standard output, known as STDOUT; and standard error, known as STDERR. Another similarity with the shell is the method of writing information into these streams. The printing of information to an open file descriptor such as STDOUT is accomplished by using the PERL command print or printf, as shown in this example.

```
printf STDOUT "This is a test\n";
```

The PERL language has a C language syntax for many of its commands. The preceding example prints the text “This is a test” followed by a newline on the standard output device. To print the same message only on standard error, use the command:

```
printf STDERR "This is a test\n";
```

Any file descriptor or file handle name can be used in place of STDOUT and STDERR, provided it has been opened for write first. If the corresponding file has not been opened, the text cannot be seen anywhere.

It may also be necessary to close one or more of the standard I/O streams to prevent unwanted text from "leaking" into places where it is not desired. A good programmer would not allow this to happen anyway. To close a file descriptor such as STDOUT in PERL, use the following command:

```
close(STDOUT);
```

If these standard I/O file descriptors are not needed to communicate with the "outside" world, then they can be closed. This means that all output from the program must be directed specifically to the location where you want it to go.

Handling Signals

Handling the types of signals that can be generated by this daemon must be included in the PERL program. Failure to capture the signals may result in the program terminating early for an unknown reason. If, for example, your program uses pipe to communicate with other programs, it may terminate abnormally if the other program wasn't ready. Catching the SIGPIPE signal will prevent the program from terminating.

Trapping a signal in PERL requires that you write a signal handler to respond to the signal, or set it to be ignored, as in the following example:

```
$SIG{"PIPE"} = "IGNORE";    # signal value 13
```

In this case, the SIGPIPE signal will be ignored. This means that if the program receives a signal SIGPIPE, it is ignored and does not cause the program to terminate.

The procmon Program

The *procmon* program is a PERL script that is started during system startup and runs for the life of the system. It has been written to be a system daemon, and it behaves as such. The purpose of this program is to monitor the operation of a set of defined processes; if they are not present in the process list, procmon restarts them. This daemon is also designed to log its actions, such as when the process fails and the time it is restarted. A sample of procmon's monitoring log, created by using the Unix syslog facility, is shown here:

```
Feb 20 07:31:21 nic procmon[943]: Process Monitor started
Feb 20 07:31:21 nic procmon[943]: Loaded config file /etc/procmon.cfg
Feb 20 07:31:22 nic procmon[943]: Command File: /etc/procmon.cmd
Feb 20 07:31:22 nic procmon[943]: Loop Delay = 300
Feb 20 07:31:22 nic procmon[943]: Adding named  to stored process list
Feb 20 07:31:22 nic procmon[943]: Monitoring : 1 processes
Feb 20 07:31:22 nic procmon[943]: named  running as PID 226
Feb 20 07:36:22 nic procmon[943]: named  running as PID 226
Feb 20 07:41:23 nic procmon[943]: named  running as PID 226
```

This syslog output shows procmon as it starts and records what it is doing with named. This setup is helpful for troubleshooting. You need to have as much logging information as possible about the process you were monitoring.

The benefit of a program such as this one is most noticeable when the program is started at system boot time. How the program starts depends on the Unix variant you are using. On System V systems, the command line shown here is added to /etc/rc2 or to a file in /etc/rc2.d directory, which is the preferred method. BSD-based systems use the same command, but in the /etc/rc.local directory.

```
/usr/local/bin/procmon &
```

procmon is in fact a daemon process. It handles all the system signals, and disconnects itself from a controlling terminal. When procmon starts, it prints a line indicating what configuration parameters it is using, and then quietly moves to the background. All logging at this point is performed by the Unix syslog facility. The output that is printed when using the procmon.cmd file is as follows:

```
Found /etc/procmon.cfg ... loading ...
```

When using the program defaults, this is what you will see:

```
no config file... using defaults ...
```

Two configuration files are used by procmon: procmon.cfg and procmon.cmd. Only the procmon.cmd file is absolutely necessary. If procmon.cfg exists, it will be used to alter the base configuration of the program.

The default configuration file is /etc/procmon.cfg. If this file is not found when procmon starts, it uses the default parameters built into the program. This configuration file enables the system administrator to change the location of the procmon.cmd file and to change the delay between checking the commands in the list.

If no /etc/procmon.cfg file is found, procmon uses the /etc directory to look for the procmon.cmd file, and a default delay of five minutes between checks. Notice that the delay value is in seconds not minutes. The /etc/procmon.cfg file is processed by procmon, and the needed values are checked for validity before they are used. This means that comments using the "#" symbol are supported, so long as "#" is the first character on the line. The procmon.cfg file is shown as follows:

```
#
# Configuration file for procmon.
#
#
# 5 minute delay
#
delay_between = 300;
#
```

```
# where is the process list file?
#
ConfigDir = "/etc";
```

The reason for the use of this configuration file is so that the parameters of the program can be modified without the need to change the source code. The delay_between variable is used to define the amount of delay between processing the list of commands. For example, if the delay_between variable is 300, a pause of 300 seconds takes place between processing.

The ConfigDir variable tells procmon where the procmon.cmd file is located. procmon defaults to /etc.

While it is possible to use the PERL require command to instruct PERL to read the named file, procmon.cfg, into the current program, this can cause potential security problems. If the configuration file is attacked, for example, a PERL program could be put into place by the wily hacker, thereby granting them access to the system.

Consequently, it is better to read the configuration file, process the entries, and validate the data values before using them. In our procom script, if the delay_between value contains anything other than a number, the value is not used, and a default of 300 seconds replaces the requested value. The same is true for ConfigDir: if the value is not a directory, the default of /etc is used.

The procmon.cmd file contains the list of processes that are to be monitored. This file contains two exclamation mark (!) separated fields: the first is the pattern to search for in the process list; the second is the name of the command to execute if the pattern is not found.

```
named !/etc/named
cron!/etc/cron
```

This file indicates that procmon will be watching for named and cron. If named is not in the process list, the command /etc/named is started. The same holds true for the cron command. The purpose of using a configuration file for this information is to allow the system administrator to configure this file on the fly. If the contents of this file change, the procmon daemon must be restarted to read the changes.

Some startup messages are recorded by syslog when procmon starts. The appropriate information is substituted for the values in <value>; the <timestamp> is replaced by the current time through syslog.; <PID> is the process identification number of the procmon process, and <system_name> is the name of the system.

```
timestamp system_name procmon[PID]: Process Monitor started
timestamp system_name procmon[PID]: Loaded config file value
timestamp system_name procmon[PID]: Command File: value
timestamp system_name procmon[PID]: Loop Delay = value
timestamp system_name procmon[PID]: Adding value  to stored process list
timestamp system_name procmon[PID]: Monitoring : value processes
```

Monitoring messages are printed during the monitoring process. These messages represent the status of the monitored processes:

```
timestamp system_name procmon[PID]: process  running as PID PID
    This record is printed after every check, and indicates that the monitored
process is running.

timestamp system_name procmon[PID]: process is NOT running
    This record is printed when the monitored process cannot be found in the
process list.

timestamp system_name procmon[PID]: Last Failure of process @ time
     This record is printed to record when the last (previous) failure of the
process was.

timestamp system_name procmon[PID]: issuing start_command to system
    This record is printed before the identified command is executed.

timestamp system_name procmon[PID]: start_command returns return_code
```

This last message is printed after the command has been issued to the system. The syslog may be able to give you clues regarding the status of the system after the command was issued. Actual procmon syslog entries are included here:

```
Feb 20 07:31:21 nic procmon[943]: Process Monitor started
Feb 20 07:31:21 nic procmon[943]: Loaded config file /etc/procmon.cfg
Feb 20 07:31:22 nic procmon[943]: Command File: /etc/procmon.cmd
Feb 20 07:31:22 nic procmon[943]: Loop Delay = 300
Feb 20 07:31:22 nic procmon[943]: Adding named  to stored process list
Feb 20 07:31:22 nic procmon[943]: Monitoring : 1 processes
Feb 20 07:31:22 nic procmon[943]: named  running as PID 226
Feb 20 07:36:22 nic procmon[943]: named  is NOT running
Feb 20 07:36:24 nic procmon[943]: Last Failure of named , @ Sun Feb  12 13:29:02
➥EST 1995
Feb 20 07:36:26 nic procmon[943]: issuing /etc/named to system
Feb 20 07:36:42 nic procmon[943]: /etc/named returns 0
Feb 20 07:41:22 nic procmon[943]: named  running as PID 4814
```

The procmon code displayed earlier has been written to run on System V systems. It has been in operation successfully since December 18, 1994. However, some enhancements could be made to the program. For example, it makes sense to report a critical message in syslog if the command returns anything other than 0. This is because a non-zero return code generally indicates that the command did not start. Another improvement would be to include a BSD option to parse the Ps output, and add an option in the configuration file to choose System V or BSD.

Unix Run Levels

Run levels, which are equivalent to system operation levels, have not been around as long as Unix. In fact, they are a recent development with System V. Early versions of System V did not include the concept of run-levels. A *run level* is an operating state that determines which facilities will be available for use. There are three primary run levels: halt, single-user, and multiuser, although there can be more.

The run level is adjustable by sending a signal to init. Whether this can be done depends on the version of Unix in use, and the version of init. Many Unix versions only have single-user, or system maintenance, and multiuser modes.

On SunOS 4.1.x systems, for example, init terminates multiuser operations and resumes single-user mode if it is sent a terminate (SIGTERM) signal with 'kill -TERM 1'. If processes are outstanding because they're deadlocked (due to hardware or software failure), init does not wait for all of them to die (which might take forever), but times out after 30 seconds and prints a warning message.

When init is sent a terminal stop (SIGTSTP) signal using 'kill -TSTP 1', it ceases to create new processes, and allows the system to slowly die away. If this is followed by a hang-up signal with 'kill -HUP' 1 init will resume full multi-user operations; For a terminate signal, again with 'kill -TERM 1', init will initiate a single-user shell. This mechanism of switching between multiuser and single-user modes is used by the reboot and halt commands.

The process differs greatly on System V systems, where there can be many different run levels. The run levels available under SCO OpenServer 5.0 are shown in table 2.5.

Table 2.5
SCO OpenServer 5.0 Run Levels

Run Level	Explanation
0	Shuts down the machine so that the power can be removed safely. Has the machine remove power if it can. This state can be executed only from the console.
1	Puts the system in single-user mode. Unmounts all file systems except the root file system. All user processes are killed except those connected to the console. This state can be executed only from the console.
2	Puts the system in multiuser mode. All multiuser environment terminal processes and daemons are spawned. This state is commonly referred to as multiuser mode.

continues

Table 2.5, Continued
SCO OpenServer 5.0 Run Levels

Run Level	Explanation
3, 4	These run levels, while being multiuser also, can be customized by the administrator to offer additional services. They are not necessary for system operation and are not normally used.
5	Stops the Unix system and goes to the firmware monitor.
6	Stops the Unix system and reboots to the run-level defined by the initdefault entry in /etc/inittab.
a, b, c	Processes only those /etc/inittab entries having the a, b, or c run-level set. These are pseudo-states that may be defined to run certain commands, but do not cause the current run-level to change.
q, Q	Re-examines /etc/inittab.
s, S	Enters single-user mode. When this occurs, the terminal that executed this command becomes the system console. This is the only run-level that doesn't require the existence of a properly formatted /etc/inittab file. If this file does not exist, then by default the only legal run-level that init can enter is the single-user mode. When the system enters S or s, all mounted file systems remain mounted and only processes spawned by init are killed.

As is evident, the differences in the services available with the different run levels are extensive. For most Unix systems, however, networking and non-root file systems are generally only available when operating in multiuser mode.

To switch run levels in the System V universe, run the command init, or telinit with the new run level. Afterward, the system prints the new run level information on the console and starts the process of switching to the new run level. To switch, the system reads the /etc/inittab file and executes commands that have the same run level. The /etc/inittab file consists of colon delimited records:

```
ck:234:bootwait:/etc/asktimerc </dev/console >/dev/console 2>&1
identifier:run levels:action:command
```

These fields consist of a unique identifier for the record, the run levels that this item is to be processed for, the action (what is to be done), and the command to be executed. init processes this file when it enters the given run level. The action field can consist of the following:

- **boot.** The entry is to be processed only at init's boot-time read of the inittab file. init is to start the process, not wait for its termination; and when it dies, not restart the process. For this instruction to be meaningful, the run level should be the default or it must

match init's run-level at boot time. This action is useful for an initialization function following a hardware reboot of the system.

- **bootwait.** The entry is to be processed the first time init goes from single-user to multi-user state after the system is booted. (If the value of initdefault is set to 2, the process will run right after the boot.) init starts the process, waits for its termination and, when it dies, does not restart the process.

- **initdefault.** An entry with this action is only scanned when init is first started. init uses this entry, if it exists, to determine which run level to enter first. It does this by taking the highest run level specified in the run level field and using that as its initial level. If the run level field is empty, it is interpreted as 0123456, which causes init to enter run level 6. If init does not find an initdefault entry in /etc/inittab, it will request an initial run level from the user at reboot time.

- **off.** If the process associated with this entry is currently running, send the warning signal (SIGTERM) and wait 20 seconds before forcibly terminating the process via the kill signal (SIGKILL). If the process is nonexistent, ignore the entry.

- **once.** When init enters a run level that matches the entry's run level, init is to start the process and not wait for its termination. When it dies, do not restart the process. If upon entering a new run level, when the process is still running from a previous run level change, the program will not be restarted.

- **ondemand.** This instruction is really a synonym for the respawn action. It is functionally identical to respawn but is given a different keyword to separate it from run levels. This is used only with the a, b, or c values described in the run level field.

- **powerfail.** Execute the process associated with this entry only when init receives a power fail signal.

- **powerwait.** Execute the process associated with this entry only when init receives a power fail signal, but wait until it terminates before continuing any processing of inittab.

- **respawn.** If the process does not exist, then start the process; do not wait for its termination (continue scanning the inittab file). When the process dies, restart the process. If the process currently exists, then do nothing and continue scanning the inittab file.

- **sysinit.** init executes these entries when the system first goes to single-user mode after being rebooted. It does not execute these entries if the system is subsequently put in single-user mode from any run-level 1 through 6. Entries with sysinit in their action field do not specify a run level in their run level field.

- **wait.** When init enters the run level that matches the entry's run level, start the process and wait for its termination. All subsequent reads of the inittab file while init is in the same run level will cause init to ignore this entry.

The init command will not report the run level it is operating in after it has switched run levels. For example, when switiching from single-user to multiuser mode, init many report this bay printing something like

```
init: Run-Level 2
```

before continuing. Currently, the only way to see what run level init is in is to use the command who. In addition to showing you who is logged on the system, this command is capable of showing the run level the system is currently operating in. To view the run level, use the option, -r. The output of this command is shown as follows:

```
$ who -r
 . run-level 2 Aug 27 21:31 2 0 S
$
```

According to this output, the current run level is 2. The date refers to when the system entered that run level; the digits to the right show the current, oldest, and last run level.

Program Listings

Each of the following code lists are programs discussed earlier in this chapter. Fortunately, you do not have to type these lists by hand; they are included on the disc at the back of this book.

Listing 2.1—The dfmon Program

This program and its output are discussed in the section on writing daemons in the Bourne/Korn shell. To use dfmon, install it /usr/local/bin, and add it to one of the system startup scripts. On the next reboot, dfmon will start up and commence monitoring your available disk space.

When installing dfmon.sh, do not execute it as root. Because none of the commands in the script are restricted, it is not necessary for the script to run as root. In fact, allowing it to run as root may contribute to lowering your security level. This is because the dfmon.sh script uses the Bourne/Korn shell source command to load in the configuration file.

This command simply "includes" the contents of the named file into the shell, and could be used to circumvent the system. If you must run dfmon.sh as root, be sure to put the configuration file in a protected directory, and use the appropriate file permissions to prevent non-root users from accessing it.

```
#!/bin/sh
#
# This is a shell program to monitor the available disk space on a system
# and report when it reaches a lower limit.
#
```

```
# The script is intended to run on both System V and BSD systems, and handle
# the different forms of df that exist"on both platforms.
#
# Configuration
# CONFIG=<path>/dfmon.cfg
CONFIG=./dfmon.cfg
#
# Load hn the configuration file using the source (.) command.
# NOTE
# THE DFMON.SH PROGRAM SHOULD NOT BE EXECUTED AS ROOT.
#
. $CONFIG
#
# With the configuration file loaded, we now start the daemon process.  To do
# this we run the balance of the command in a subshell, so the parent
# process can exit.
#
echo "`date` Starting Disk Monitor Daemon ...."
(
#
# Ignore TRAPS, so we can't be killed with anything but a kill -9 ...
#
# NOTE:
# on HP-UX, traps 11, 18 cannot be specified in the trap list
# on SCO, traps 11, 20-25 cannot be specified in the trap list
trap "" 1 2 3 4 5 6 7 8 10 12 13 14 15 16 17 18 19
#
# Assemble our environment
PATH=/usr/bin:/bin:/etc:/sbin:/usr/sbin
IFS="   "
# Comment this on systems that do not use dynamically loaded libraries like
# Sun-based systems.
unset LD_LIBRARY
#
# NOTE:
# Even though the PATH variable has been explicity set, the commands
# executed in this script are specified using their exact path.
#
# Even though the intent behind this program is to function as a daemon,
# the standard I/O files will not be closed as the standard I/O path is used
# to communicate between the main program and the loaded shell functions that
# are found in the configuration file.
#
# We need to get the df output first, and feed it into a while loop for
# processing.
#
# Here we run the correct version of the df_function, so that we get the
# information we want from the non-standard, non-compatible versions of
# df that are in existence. (And they say that there is a standard!).
#
```

```
while :
do
for filesystem in '/etc/mount ¦ /usr/bin/cut -d" " -f$MOUNT_POS'
do
    case $DF_TYPE in
        HPUX)
            LOGGER=/usr/bin/logger
            RESULTS='df_hpux $filesystem';;
        SCO_UNIX)
            LOGGER=/usr/bin/logger
            RESULTS='df_sco_unix $filesystem';;
        SunOS)
            LOGGER=/usr/ucb/logger
            RESULTS='df_sunos $filesystem';;
        BSDI)
            LOGGER=/usr/bin/logger
            RESULTS='df_bsdi $filesystem';;
        LINUX)
            LOGGER=/usr/bin/logger
            RESULTS='df_linux $filesystem';;
    esac
    set $RESULTS

    FILESYS=$1
    FREE=$2
    TOTAL=$3
    USED=$4

    #
    # we need to check the file system to determine what type of
    # control we want to place upon it.  For example, if the file system
    # is root, then the ROOT_LOW and ROOT_CRITICAL values are used for the
    # monitoring alarms.
    case "$FILESYS" in
        "/")
            LOW=$ROOT_LOW
            CRITICAL=$ROOT_CRITICAL;;
        "/usr")
            LOW=$USR_LOW
            CRITICAL=$USR_CRITICAL;;
        "/var")
            LOW=$VAR_LOW
            CRITICAL=$VAR_CRITICAL;;
        *)
            LOW=$OTHER_LOW
            CRITICAL=$OTHER_CRITICAL;;
    esac

    #
    # look at the bytes free versus the total bytes available
    # if the free space is lower than the lower water mark
    # from the config file, then sound the alarm, if and only
    # if the disk filesystem is alarmed.
    #
```

```
    # if syslog is in use, use the logger command to send a message
    # and save it in the syslog.  Otherwise, use the name of the
    # log file from the config file, and log the problem to the file,
    # to the console device, and to the user identified in the config
    # file.
    #
    # we will use a special facility so that these messages can be
    # appropriately handled.
    #
    # The CRITICAL level is checked first because it will be the lower of
    # the two test values.
    #
    if [ "$FREE" -le "$CRITICAL" ]
    then
        #
        # It is a critical level, so use syslog to record the alarm
        #
        if [ "$USE_SYSLOG" = "YES" ]
        then
            #
            # Use the logger command to send the information to
            # the syslog daemon.
            #
            /usr/bin/logger -t "dfmon" -p $SYSLOG_FAC.crit "CRITICAL Alarm :
➥$FILESYS space: Free=$FREE"
        else
            #
            # It is critical, but we do not have syslog, so we
            # use our fake_syslog function.
            #
            fake_syslog "CRITICAL Alarm : $FILESYS space: Free=$FREE"
        fi
    #
    # It isn't crtical, so lets check it against our low level alarm.
    #
    elif [ "$FREE" -le "$LOW" ]
    then
        #
        # Yes - it is a low level alarm, so use syslog to report
        # the alarm.
        #
        if [ "$USE_SYSLOG" = "YES" ]
        then
            #
            # Use the logger command to send the information to
            # the syslog daemon.
            #
            /usr/bin/logger -t "dfmon" -p $SYSLOG_FAC.emerg "WARNING Alarm :
➥$FILESYS space: Free=$FREE"
        else
            #
            # syslog is not available, so mimic it using the
            # fake syslog function.
            #
```

```
            fake_syslog "CRITICAL Alarm : $FILESYS space: Free=$FREE"
        fi
    fi
done
    #
    # Delay the number of seconds identified by PASS_DELAY in the config
    # file.
    #
    sleep $PASS_DELAY
    #
    # this constitutes the end of the daemon code.  This will execute until
    # the system is halted, or until a user kills it.
    #
done
) &
# end of the road
```

Listing 2.2—The dfmon Configuration File

This listing shows a sample configuration for dfmon, called dfmon.cfg. The dfmon.cfg file is used to add new shell functions to provide support for other versions of Unix. The configuration file should be installed in a protected directory, and use appropriate file permissions, such as a mode of 600 and an owner of root, thereby preventing any non-root user from being able to access the file. You might want to make some changes (as noted in the text) for different operating systems and desired modes of operation.

```
# ========================================================================
#                     User Configurable Parameters
# ========================================================================
#
# --------------------
# DF_TYPE
#        HPUX        HP-X 9.x
#        SCO_UNIX
#        SunOS        SunOS 4.1.[34]
#        BSDI        BSDI 2.0
#        LINUX        Linux Slackware 2.0
#
# Select the proper system from the above list.  If nothing is an exact match
# but there is a close approximation, then try it first.  If nothing works,
# then you will have to build your own df_func.  The df_func are found later
# in this file.
#
DF_TYPE=SunOS
#
# MOUNT_POS is the position where the mount point is found in the output from
# the mount command.  On most System V systems, a value of 1 is desired;
# on many BSD type systems, a value of 3 is desired.
MOUNT_POS=3
```

```
#
# --------------------
# Warning Alarms
#
# This section identifies the disk space, in blocks, that should be considered
# a problem.
#
# There are only FOUR specific possibilities without changing the original
# program.  These identify the LOW and CRITICAL disk space values for the
#
#        Root
#        usr
#        var
#        all other
#
# filesystems.  Support for other specific names will require modification
# within the actual dfmon program itself.
#
# root filesystem levels
#
ROOT_LOW=1100000
ROOT_CRITICAL=1100000
#
# usr filesystem levels
#
USR_LOW=1000
USR_CRITICAL=100
#
# var filesystem levels
#
VAR_LOW=1000
VAR_CRITICAL=100
#
# all other filesystem levels
#
OTHER_LOW=1000
OTHER_CRITICAL=100
#
# --------------------
#
# USE_SYSLOG
#
# This determines if the syslog daemon should be used to record the alarms.  If
# the value is YES, then syslog is used.  If the value is NO, then the dfmon
# program uses a function called fask_syslog to save the information in a log
# file and write the alarm information to the console device of the system
# and mail it to the identified user.
#
USE_SYSLOG=NO
#
# SYSLOG_FAC is applicable only if the value of USE_SYSLOG is YES.  It identifies
# the syslog facility that should be used to log these messages.
#
```

```
SYSLOG_FAC=user
#
# FAKE_SYSLOG_FILE is the path and name of a file that should be used to
# save the syslog like messages in.  This is used only if the value of
# USE_SYSLOG is NO.
# FAKE_SYSLOG=/var/log/dfmon.log
FAKE_SYSLOG=/usr/adm/dfmon.log
#
# PASS_DELAY is the amount of time in seconds that the dfmon program is to pause
# before it executes again.  The default is 15 minutes, or 900 seconds.
#
PASS_DELAY=900
#
# ======================================================================
#                          Shell Functions
# ======================================================================
#
# The following shell functions provide support for the df command on the
# followng systems.  They should not require modification.  If you do
# make changes that you think should be passed on to the author, please
# send them to chrish@unilabs.org.
#
# Because we don't want to chew up too many CPU cycles in doing this, please
# make your functions as tight as possible.  For example, the HP-UX version of
# the df (df_hpux) uses only three external commands: only one of which would
# be executed more than once for a multiple filesystem machine.
#
# ============================================================================
#
# Hewlett-Packard HP-UX Version 9.0
#
# Sample df output
# /                (/dev/dsk/c201d6s0):    1053202 blocks        127796 i-nodes
#                                        1768252 total blocks  142080 total i-nodes
#                                         538224 used  blocks    14284 used i-nodes
#                                          10 percent minfree
#
df_hpux()
{
#
# This function will take the output from the HP-UX 9.x df command and return
# the number of blocks free, used, and total available.
#
set 'df -t $1'
FILESYS=$1
FREE=$3
TOTAL=$7
shift 9
USED=$4
echo "$FILESYS $FREE $TOTAL $USED"
return
}
# ======================================================================
```

```
#
# SCO UNIX (OpenServer 5.0)
#
# /         (/dev/root        ):   282716 blocks    64603 i-nodes
#                          total:   751704 blocks    93968 i-nodes
#
df_sco_unix()
{
#
# This function will take the output from the SCO UNIX df command and return
# the number of blocks free, used, and total available.
#
set 'df -t $1'
FILESYS=$1
FREE=$4
TOTAL=$9
echo "$FILESYS $FREE $TOTAL 0"
return
}
# ========================================================================
#
# SunOS 4.1.[34]
#
# Filesystem            kbytes    used   avail capacity  Mounted on
# /dev/sd0a              30528   11086   16390    40%    /
# /dev/sd0g            1505102 1221718  132874    90%    /usr
# /dev/sd0d               9818      13    8824     0%    /export
# /dev/sd0e              31615       9   28445     0%    /export/swap
# /dev/sd0h             230158  196033   11110    95%    /home
#
df_sunos()
{
#
# This function will take the output from the SunOS 4.1.x df command and return
# the number of blocks free, used, and total available.
#
set 'df $1'
shift 8
FILESYS=$5
FREE=$3
TOTAL=$1
USED=$2
echo "$FILESYS $FREE $TOTAL $USED"
return
}
# ========================================================================
#
# BSDI 2.0
#
# Filesystem            kbytes    used   avail capacity  Mounted on
# /dev/sd0a              30528   11086   16390    40%    /
# /dev/sd0h             230158  196033   11110    95%    /home
```

```
df_bsdi()
{
#
# This function will take the output from the BSDI 2.x df command and return
# the number of blocks free, used, and total available.
#
set 'df $1'
shift 8
FILESYS=$5
FREE=$3
TOTAL=$1
USED=$2
echo "$FILESYS $FREE $TOTAL $USED"
return
}
# ====================================================================
#                                Support Functions
# ====================================================================
#
# The fake_syslog function will accept the provided arguments, and create a
# syslog like entry in the named support file.
fake_syslog()
{
#
# It is important to do these things:
#    1.  Write the message to the fake syslog file
#    2.  Use the wall command to send a message to everyone
#
# If the file exists, then we will append to it.  If not, then the first
# write to the file using echo will create the file.
#
# The text being written is passed as an argument to the function.
#
MESSAGE=$*
#
# Write the message to the file along with the date and name of the system.
#
echo "`date` `uname -n` dfmon: $MESSAGE" >> $FAKE_SYSLOG
#
# Send a message using the WALL command.
#
echo "`date` `uname -n` dfmon: $MESSAGE" ¦ wall
}
```

Listing 2.3—The procmon Command

The procmon command was presented in the section covering daemon programming with PERL. This command must be executed as root, so it and its configuration files should be protected appropriately.

```
#!/usr/local/bin/perl
#
# ------------------------------------------------------------------
#
# This program requires the PERL ctime(PERL) library for generating date
# strings in the standard ctime format. To prevent tampering with the
# Ctime library file from affecting the operation of this script, it has
# been included at the end of the actual procmon program.
#
# Any future changes to the ctime.pl script will need to be applied to the
# version in this file, although it appears to operate without difficulty.
#
# require "ctime.pl";
#
# This program requires the syslog(PERL) library for successful delivery
# of logging information to syslog on the host machine.  What syslog does
# with the information is up to syslog and the how the administrator
# has it configured.
# To prevent tampering with the syslog library file from affecting the
# operation of this script, it has been included at the end of the actual
# procmon program.
#
# Any future changes to the syslog.pl script will need to be applied to the
# version in this file, although it appears to operate without difficulty.
#
# require "syslog.pl";
#
# These changes are to reduce the likelihood of problems because of tampered
# scripts.
#
# Look to see if the confiuration file for the process monitor is
# present in /etc.  If so, then load in the configuration file.
#
# If not, then use the default values, included here.
#
if ( -e "./procmon.cfg" )
   {
   printf STDOUT "Found /etc/procmon.cfg ... loading ...\n";
   ($delay_between, $ConfigDir) = &readconfig ("./procmon.cfg");
   }
else
   {
   printf STDOUT "no config file ... using defaults ...\n";
   $delay_between = 300;
   $ConfigDir = "/etc";
   }

printf STDOUT "procmon: using delay of $delay_between seconds\n";
printf STDOUT "procmon: using config dir of $ConfigDir\n";

#
# This is the name of this program.  DO NOT CHANGE THIS.
```

```
#
$program = "procmon";
#
# This is the name of the process list and command file
#
$command_file = "$ConfigDir/procmon.cmd";

#
# Establish the signal handler
#
$SIG{'HUP'}    = "IGNORE";     # signal value 1
$SIG{'INT'}    = "IGNORE";     # signal value 2
$SIG{'QUIT'}   = "IGNORE";     # signal value 3
$SIG{'ILL'}    = "IGNORE";     # signal value 4
$SIG{'TRAP'}   = "IGNORE";     # signal value 5
$SIG{'IOT'}    = "IGNORE";     # signal value 6
$SIG{'ABRT'}   = "IGNORE";     # signal value 6, yes this is right!
$SIG{'EMT'}    = "IGNORE";     # signal value 7
$SIG{'FPE'}    = "IGNORE";     # signal value 8
$SIG{'KILL'}   = "DEFAULT";    # signal value 9, can't be caught anyway
$SIG{'BUS'}    = "IGNORE";     # signal value 10
$SIG{'SEGV'}   = "IGNORE";     # signal value 11
$SIG{'SYS'}    = "IGNORE";     # signal value 12
$SIG{"PIPE"}   = "IGNORE";     # signal value 13
$SIG{'ALRM'}   = "IGNORE";     # signal value 14
$SIG{'TERM'}   = "DEFAULT";    # signal value 15
$SIG{'USR1'}   = "IGNORE";     # signal value 16
$SIG{'USR2'}   = "IGNORE";     # signal value 17
$SIG{'CLD'}    = "IGNORE";     # signal value 18
$SIG{'CHLD'}   = "IGNORE";     # signal value 18, yes this is right too!
$SIG{'PWR'}    = "IGNORE";     # signal value 19
$SIG{'WINCH'}  = "IGNORE";     # signal value 20
$SIG{'PHONE'}  = "IGNORE";     # signal value 21, AT&T UNIX/PC only!
$SIG{'POLL'}   = "DEFAULT";    # signal value 22
$SIG{'STOP'}   = "IGNORE";     # signal value 23
$SIG{'TSTP'}   = "IGNORE";     # signal value 24
$SIG{'CONT'}   = "IGNORE";     # signal value 25
$SIG{'TTIN'}   = "IGNORE";     # signal value 26
$SIG{'TTOU'}   = "IGNORE";     # signal value 27
$SIG{'VTALRM'} = "IGNORE";     # signal value 28
$SIG{'PROF'}   = "IGNORE";     # signal value 29

#
# Close Standard Input and Standard output
#
# These lines of code have been commented out for testing purposes.
#
# close( STDIN );
# close( STDOUT );
# close( STDERR );
```

```
#
# Open syslog for recording the startup messages as debug messages
#
&openlog( $program, "ndelay,pid", "user" );
#
# Record the startup of the monitor
#
&syslog( info,  "Process Monitor started");
&syslog( info,  "Command File: $command_file");
&syslog( info,  "Loop Delay = $delay_between");
#
# Open the list of processes to be monitored.
#
if ( -e "$command_file" )
   {
   open( LIST, "$command_file" );
   }
else
   {
   &syslog( crit,  "CAN'T LOAD COMMAND FILE : $command_file: does not exist" );
   exit(2);
   }
#
exit(0);
while (<LIST>)
   {
   chop;
   #
   # We split because each entry has the name of the command that would be
   # present in a ps -e listing, and the name of the command that is used to
   # start it should it not be running.
   #
   # An exclamation point is used between the two fields in the file.
   #
   ( $process_name, $start_process ) = split(/!/,$_ );
   &syslog( info,  "Adding $process_name to stored process list");
   #
   # Save the name of the process being monitored into an array.
   #
   @process_list = ( @process_list, $process_name );
   #
   # Save the start command in an associative array using the process_name
   # as the key.
   #
   $start_commands{$process_name} = $start_process;
   #
   # The associative array last_failure is used to store the last failure time
   # of the indicated process.
   #
   $last_failure{$process_name} = "NEVER";
   #
   # The associative array last _start is used to store the time the process
   # was last started.
```

```
   #
   $last_start{$process_name} = "UNKNOWN";
   }
$num_processes = @process_list;
&syslog( info,  "Monitoring : $num_processes processes");

#
# Loop forever
#
while (1 == 1)
   {
   EACH_PROCESS:
   foreach $process_name (@process_list)
      {
      #
      # This program was originally written for AT&T System V UNIX
      # and derivatives.  (Someday I will port it to BSD versions!)
      #
      open( PS, "ps -e ¦ grep $process_name ¦" ) ¦¦ &syslog( warn, "can't create
➥PS pipe : $!");
      while (<PS>)
      {
      chop;
         $_name = "";
      #
      # There are a log of spaces in the PS output, so these have to
      # be squeezed to one space.
      #
      tr/a-zA-Z0-9?:/ /cs;
      #
      # Read the PS list and process the information
      #
      ( $junk, $_pid, $_tty, $_time, $_name ) = split(/ /,$_ );
      #
      # Check to see if we have any information
      #
      if ( $_name ne "" )
         {
         #
         # We likely have the process running
         #
         #
         # From here we go to the next process, as it is still
         # running, and we have made a syslog entry to that
         # effect.
         #
         &syslog( "info", "$process_name running as PID $_pid");
         close(PS);
            next EACH_PROCESS;
         }
      #
      # The process is not running, so record an entry in
```

```
      # syslog.
      #
      }
      close(PS);
         &syslog( "crit", "$process_name is NOT running");
      #
      # When did the process last fail?  Saving this allows the
      # system administrator to keep tabs on the failure rate of
      # the process.
      #
      &syslog( "crit", "Last Failure of $process_name, @
➥$last_failure{$process_name}" );
      chop( $current_time = &ctime(time) );
      #
      # Set the last failure to the current time.
      #
         $last_failure{$process_name} = $current_time;
      #
      # If we have a command to execute to restart the service,
      # execute the command.
      #
      if ( defined( $start_commands{$process_name} ) )
         {
         #
         # Record the sequence of event to restart the
         # service in syslog.
         #
            &syslog( "crit", "issuing $start_commands{$process_name} to system");
         #
         # Execute the system command, and save the return code to decide
         # if it was a clean start.
         #
         $retcode = system("$start_commands{$process_name}");
         #
         # Record the return code in syslog
         #
            &syslog( "info", "$start_commands{$process_name} returns $retcode");
         #
         # Calculate the time in ctime(3C) format
         chop( $current_time = &ctime(time) );
            $last_start{$process_name} = $current_time;
         #
         # Save the return code - it is in the standard format, so must be
         # divided by 256 to get the real return value.
         #
            $retcode = $retcode / 256;
         }
      }
   #
   # From here we have processed each of the commands in the monitoring list.
   # We will now pause for station identification .....
   #
```

```
   $secs = sleep($delay_between);
   }

sub sig_handler
{
local ($sig) = @_;
&closelog();
&openlog( $program, "ndelay,cons,pid", "user" );
&syslog( "crit", "PROCESS MONITOR: SIGNAL CAUGHT SIG$sig- TERMINATING");
&closelog();
exit(0);
}

;# ctime.pl is a simple PERL emulation for the well-known ctime(3C) function.
;#
;# Waldemar Kebsch, Federal Republic of Germany, November 1988
;# kebsch.pad@nixpbe.UUCP
;# Modified March 1990, Feb 1991 to properly handle timezones
;#  $RCSfile: ctime.pl,v $$Revision: 4.0.1.1 $$Date: 92/06/08 13:38:06 $
;#   Marion Hakanson (hakanson@cse.ogi.edu)
;#   Oregon Graduate Institute of Science and Technology
;#
;# usage:
;#
;#     #include <ctime.pl>          # see the -P and -I option in perl.man
;#     $Date = &ctime(time);

CONFIG: {
    package ctime;

    @DoW = ('Sun','Mon','Tue','Wed','Thu','Fri','Sat');
    @MoY = ('Jan','Feb','Mar','Apr','May','Jun',
         'Jul','Aug','Sep','Oct','Nov','Dec');
}

sub ctime {
    package ctime;

    local($time) = @_;
    local($[) = 0;
    local($sec, $min, $hour, $mday, $mon, $year, $wday, $yday, $isdst);

    # Determine what time zone is in effect.
    # Use GMT if TZ is defined as null, local time if TZ undefined.
    # There's no portable way to find the system default timezone.

    $TZ = defined($ENV{'TZ'}) ? ( $ENV{'TZ'} ? $ENV{'TZ'} : 'GMT' ) : '';
    ($sec, $min, $hour, $mday, $mon, $year, $wday, $yday, $isdst) =
        ($TZ eq 'GMT') ? gmtime($time) : localtime($time);

    # Hack to deal with 'PST8PDT' format of TZ
    # Note that this can't deal with all the esoteric forms, but it
    # does recognize the most common: [:]STDoff[DST[off][,rule]]
```

```
    if($TZ=~/^([^:\d+\-,]{3,})([+-]?\d{1,2}(:\d{1,2}){0,2})([^\d+\-,]{3,})?/){
        $TZ = $isdst ? $4 : $1;
    }
    $TZ .= ' ' unless $TZ eq '';

    $year += ($year < 70) ? 2000 : 1900;
    sprintf("%s %s %2d %2d:%02d:%02d %s%4d\n",
      $DoW[$wday], $MoY[$mon], $mday, $hour, $min, $sec, $TZ, $year);
}


#
# syslog.pl
#
# $Log:     syslog.pl,v $
# Revision 4.0.1.1  92/06/08  13:48:05  lwall
# patch20: new warning for ambiguous use of unary operators
#
# Revision 4.0  91/03/20  01:26:24  lwall
# 4.0 baseline.
#
# Revision 3.0.1.4  90/11/10  01:41:11  lwall
# patch38: syslog.pl was referencing an absolute path
#
# Revision 3.0.1.3  90/10/15  17:42:18  lwall
# patch29: various portability fixes
#
# Revision 3.0.1.1  90/08/09  03:57:17  lwall
# patch19: Initial revision
#
# Revision 1.2  90/06/11  18:45:30  18:45:30  root ()
# - Changed 'warn' to 'mail¦warning' in test call (to give example of
#   facility specification, and because 'warn' didn't work on HP-UX).
# - Fixed typo in &openlog ("ncons" should be "cons").
# - Added (package-global) $maskpri, and &setlogmask.
# - In &syslog:
#   - put argument test ahead of &connect (why waste cycles?),
#   - allowed facility to be specified in &syslog's first arg (temporarily
#     overrides any $facility set in &openlog), just as in syslog(3C),
#   - do a return 0 when bit for $numpri not set in log mask (see syslog(3C)),
#   - changed $whoami code to use getlogin, getpwuid($<) and 'syslog'
#     (in that order) when $ident is null,
#   - made PID logging consistent with syslog(3C) and subject to $lo_pid only,
#   - fixed typo in "print CONS" statement ($<facility should be <$facility).
#   - changed \n to \r in print CONS (\r is useful, $message already has a \n).
# - Changed &xlate to return -1 for an unknown name, instead of croaking.
#
#
# tom christiansen <tchrist@convex.com>
# modified to use sockets by Larry Wall <lwall@jpl-devvax.jpl.nasa.gov>
# NOTE: openlog now takes three arguments, just like openlog(3)
```

```
#
# call syslog() with a string priority and a list of printf() args
# like syslog(3)
#
#  usage: require 'syslog.pl';
#
#  then (put these all in a script to test function)
#
#
#     do openlog($program,'cons,pid','user');
#     do syslog('info','this is another test');
#     do syslog('mail¦warning','this is a better test: %d', time);
#     do closelog();
#
#     do syslog('debug','this is the last test');
#     do openlog("$program $$",'ndelay','user');
#     do syslog('notice','fooprogram: this is really done');
#
#     $! = 55;
#     do syslog('info','problem was %m'); # %m == $! in syslog(3)

package syslog;

$host = 'localhost' unless $host;     # set $syslog'host to change

require 'syslog.ph';

$maskpri = &LOG_UPTO(&LOG_DEBUG);

sub main'openlog {
    ($ident, $logopt, $facility) = @_;  # package vars
    $lo_pid = $logopt =~ /\bpid\b/;
    $lo_ndelay = $logopt =~ /\bndelay\b/;
    $lo_cons = $logopt =~ /\bcons\b/;
    $lo_nowait = $logopt =~ /\bnowait\b/;
    &connect if $lo_ndelay;
}

sub main'closelog {
    $facility = $ident = '';
    &disconnect;
}

sub main'setlogmask {
    local($oldmask) = $maskpri;
    $maskpri = shift;
    $oldmask;
}

sub main'syslog {
    local($priority) = shift;
    local($mask) = shift;
```

```
local($message, $whoami);
local(@words, $num, $numpri, $numfac, $sum);
local($facility) = $facility;     # may need to change temporarily.

die "syslog: expected both priority and mask" unless $mask && $priority;

@words = split(/\W+/, $priority, 2);# Allow "level" or "level¦facility".
undef $numpri;
undef $numfac;
foreach (@words) {
 $num = &xlate($_);          # Translate word to number.
 if (/^kern$/ ¦¦ $num < 0) {
     die "syslog: invalid level/facility: $_\n";
 }
 elsif ($num <= &LOG_PRIMASK) {
     die "syslog: too many levels given: $_\n" if defined($numpri);
     $numpri = $num;
     return 0 unless &LOG_MASK($numpri) & $maskpri;
 }
 else {
     die "syslog: too many facilities given: $_\n" if defined($numfac);
     $facility = $_;
     $numfac = $num;
 }
}

die "syslog: level must be given\n" unless defined($numpri);

if (!defined($numfac)) {     # Facility not specified in this call.
 $facility = 'user' unless $facility;
 $numfac = &xlate($facility);
}

&connect unless $connected;

$whoami = $ident;

if (!$ident && $mask =~ /^(\S.*):\s?(.*)/) {
 $whoami = $1;
 $mask = $2;
}

unless ($whoami) {
 ($whoami = getlogin) ¦¦
     ($whoami = getpwuid($<)) ¦¦
      ($whoami = 'syslog');
}

$whoami .= "[$$]" if $lo_pid;

$mask =~ s/%m/$!/g;
$mask .= "\n" unless $mask =~ /\n$/;
$message = sprintf ($mask, @_);
```

```
    $sum = $numpri + $numfac;
    unless (send(SYSLOG,"<$sum>$whoami: $message",0)) {
     if ($lo_cons) {
         if ($pid = fork) {
          unless ($lo_nowait) {
              do {$died = wait;} until $died == $pid ¦¦ $died < 0;
          }
         }
         else {
          open(CONS,">/dev/console");
          print CONS "<$facility.$priority>$whoami: $message\r";
          exit if defined $pid;          # if fork failed, we're parent
          close CONS;
         }
     }
    }
}

sub xlate {
    local($name) = @_;
    $name =~ y/a-z/A-Z/;
    $name = "LOG_$name" unless $name =~ /^LOG_/;
    $name = "syslog'$name";
    eval(&$name) ¦¦ -1;
}

sub connect {
    $pat = 'S n C4 x8';

    $af_unix = 1;
    $af_inet = 2;

    $stream = 1;
    $datagram = 2;

    ($name,$aliases,$proto) = getprotobyname('udp');
    $udp = $proto;

    ($name,$aliase,$port,$proto) = getservbyname('syslog','udp');
    $syslog = $port;

    if (chop($myname = 'hostname')) {
     ($name,$aliases,$addrtype,$length,@addrs) = gethostbyname($myname);
     die "Can't lookup $myname\n" unless $name;
     @bytes = unpack("C4",$addrs[0]);
    }
    else {
     @bytes = (0,0,0,0);
    }
    $this = pack($pat, $af_inet, 0, @bytes);
```

```
    if ($host =~ /^\d+\./) {
     @bytes = split(/\./,$host);
    }
    else {
     ($name,$aliases,$addrtype,$length,@addrs) = gethostbyname($host);
     die "Can't lookup $host\n" unless $name;
     @bytes = unpack("C4",$addrs[0]);
    }
    $that = pack($pat,$af_inet,$syslog,@bytes);

    socket(SYSLOG,$af_inet,$datagram,$udp) ¦¦ die "socket: $!\n";
    bind(SYSLOG,$this) ¦¦ die "bind: $!\n";
    connect(SYSLOG,$that) ¦¦ die "connect: $!\n";

    local($old) = select(SYSLOG); $¦ = 1; select($old);
    $connected = 1;
}

sub disconnect {
    close SYSLOG;
    $connected = 0;
}

sub main'readconfig {
    # This will read in the named configuration file and check it for
    # validity.
    local ( $configfile ) = @_;
    if ( ! -r $configfile )
       {
       $delay_between = 300;
       $configDir = "/etc";
       return( $delay_between, $ConfigDir);
       }
    open( CFG, $configfile );
    while (<CFG>)
       {
       next if ( $_ =~ /^#/ );
       chop;
       if ( $_ =~ /delay_between/ )
       {
       ( $var, $value ) = split(/=/);
       if ( $value !~ /[a-zA-Z]/ )
          {
          $delay_between = $value;
          }
       else
          {
          $delay_between = 300;
          }
       }
       if ( $_ =~ /ConfigDir/ )
       {
```

```
        }
        if ( $_ =~ /ConfigDir/ )
        {
        ( $var, $value ) = split(/=/);
           if ( -d $value )
           {
           $ConfigDir = $value;
           }
        else
           {
           $ConfigDir = "/etc";
           }
        }
        }
        return( $delay_between, $ConfigDir);
}
```

Listing 2.4—The procmon.cfg File

The sample configuration file shown here is used to provide the operating parameters for the procmon script. It can only accept two configuration parameters, as follows:

- **delay_between.** The number of seconds to wait between checks in the process list
- **ConfigDir.** The name of the directory where procmon should look to find the configuration file and the command file, procmon.cmd.

```
#
# Configuration file for procmon.
#
#
# 5 minute delay
#
delay_between = 300;
#
# where is the process list file?
#
$ConfigDir = "/etc";
```

Using UUCP

UUCP (Unix to Unix CoPy) is used for transferring files from one system to another. The copy may result in other work being done on one of the systems, such as invoking mail or news programs, which is discussed later in this chapter. Basically, UUCP is a collection of programs that are capable of the following:

- *Transferring files between Unix systems*
- *Executing commands on remote systems*

Although UUCP is not in as widespread use as TCP/IP, it is still available on your system. Without knowledge of this, it is possible for the experienced hacker to access your files using UUCP. It is shipped with standard logins and permissions established. This chapter presents what UUCP is, how it works, and how to configure it correctly.

The History of UUCP

Unlike many other networking environments, UUCP requires no special hardware. Its simplicity is due in part to its age: UUCP formed the basis for communication on USENET and early networks. The first network many Unix people worked on was a collection of XENIX and Unix machines joined together with serial cables and UUCP. Although various protocols can be run on LANs (local area networks), UUCP has traditionally been used in long haul dial-up networks. That is, call the remote machine when there is something for it; otherwise, UUCP does nothing.

UUCP has been used in many different situations. Large financial organizations, for example, used it to send daily processing information to a central computer; sales and distribution companies used UUCP to send inventory and pricing information to remote offices. Although UUCP made these transfers simple, this type of networking still involved cooperation between system administrators.

Two versions of UUCP are currently being shipped. The first UUCP was written by Mike Lesk of AT&T as a research project. During and after the project, UUCP became such a success that Mike joined with David Nowitz and Greg Chesson to build the version of UUCP that was shipped with Version 7 Unix, and subsequently became known as Version 2 UUCP. This version is still shipped by some vendors, and many older Unix systems have this version.

With the release of System V Release 3 from AT&T, a new UUCP version was released that was known by a number of names, most notably the Basic Networking Utilities, or HoneyDanBer (HDB) UUCP. HDB UUCP was written in 1983 by Peter Honeyman, David A. Nowitz, and Brian E. Redman. They named this version after the login names they used on the system (Honey, Dan, Ber). HDB provides additional capabilities, such as newer modem and network support, and corrects design deficiencies of Version 2.

For the most part, communication between different UUCP versions is transparent after they have been configured. Although many of the features have not changed, the names and layouts of many of the configuration files have changed. As a result, it is easy to tell which version you have. If you see a file named L.sys in the directory /usr/lib/uucp, you have Version 2. A file named Systems in this directory is only found on HoneyDanBer UUCP. No /usr/lib/uucp directory means that you don't have UUCP installed. Table 3.1 shows a list of the file names that make up each version, and their use.

Table 3.1
Comparison of Version 2 and BNU Files

Version 2 Files	BNU/HDB Files	Description
L-devices	Devices	Contains a description of the devices that are attached to your system.

Version 2 Files	BNU/HDB Files	Description
L-dialcodes	Dialcodes	Uses text to designate areas instead of the actual area codes. This file describes that text.
	Dialers	Contains the commands needed to get a modem or other device to make a call.
	Maxuuscheds	Contains a number that is the maximum number of uuscheds that can be running at one time.
	Maxuuxqts	The same as Maxuuscheds, but it is for uuxqt.
USERFILE	Permissions	Contains the needed information to control the security of your machine.
	Poll	You can elect to call sites at specific times, which is polling. This file describes who is called, and when.
	Sysfiles	Used to configure separate Systems files for cu and uucico.
L.sys	Systems	Contains the list of systems you can call, or that you know about, and how to contact them. It contains phone numbers, user names, and passwords.
	Uutry	Shell command to test chat scripts by calling the remote system.
	remote.unknown	Shell command to record calls from unknown systems.
	uucheck	Checks the UUCP directories and Permis sions file for proper setup and consistency.
uucico	uucico	The UUCICO program. This is the heart of the UUCP system.
uuclean	uucleanup	General clean-up tool.
	uudemon.admin	Shell command to send UUCP status information to the system administrator.
	uudemon.cleanu	Shell command to do general UUCP system cleanup tasks.
	uudemon.hour	Shell command to run the UUCP scheduler.
	uudemon.poll	Shell command to call (or poll) the specified systems at the designated hours (see Poll).
uudemon.day		UUCP maintenance tasks to be run once per day.

continues

Table 3.1, Continued
Comparison of Version 2 and BNU Files

Version 2 Files	BNU/HDB Files	Description
uudemon.kr		Runs uucico once per hour and processes all pending work.
uudemon.wk		UUCP maintenance tasks to be performed once per week.
	uugetty	Alternate getty program.
	uusched	The uucp scheduler.
	uutry	A debugging program.
uusub		Defines a UUCP sub-network and monitors the traffic between the hosts.
uuxqt	uuxqt	Executes jobs on your system that are required as part of the transfer.
SEQF		The sequence number for the next job.
L_stat		System status file.
L_sub		UUCP connection statistics.
R_stat		Request status file.
R_sub		UUCP traffic statistics.

Not all of the functionality or corresponding programs and files in table 3.1 exist between implementations. One implementation has one file or program, and the other does not.

The UUCP Network

The UUCP network consists of a group of machines connected by some communication mechanism that uses UUCP to transfer files and information. A sample network is depicted in figure 3.1.

This UUCP network consists of a gateway that provides UUCP services via dialup at 19.2 Kbaud to the machine unilabs. In turn, unilabs offers dedicated UUCP connections over direct serial cables to the machines bugs, thumper, and wabbit.

Machines are connected via a modem and phone line, a direct connection—serial cable—between machines in close proximity, or some form of a leased line in the case of distant systems. If desired, UUCP can even be deployed over TCP/IP.

Often the machines in a UUCP network are different models or use different operating system versions. The network in figure 3.1 has SunOS, SCO Unix, Motorola Unix, AT&T Unix, Spectrix XENIX, and SCO XENIX in it. Even though all the machines are running Unix, each version has a unique form of UUCP; the only was to ensure network communication is through sysop cooperation.

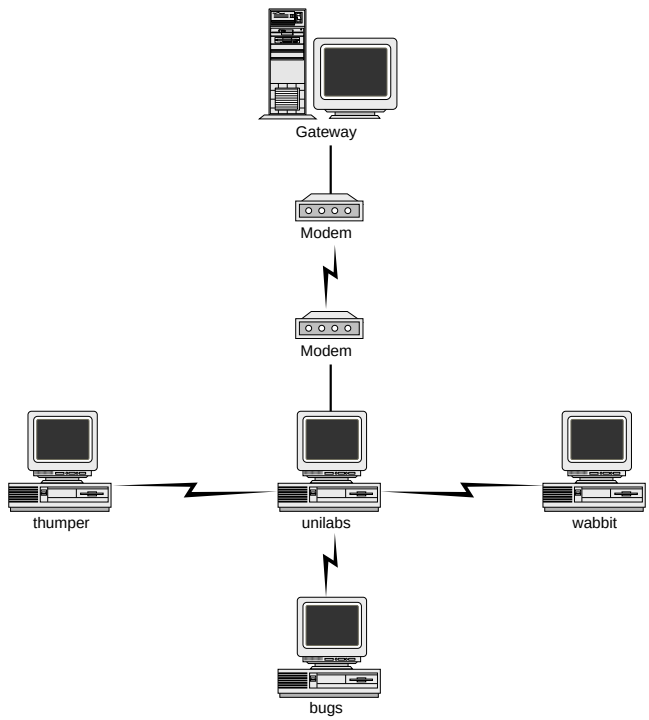


Figure 3.1
A sample UUCP network.

How UUCP Works

A UUCP job is submitted by a user through the commands uucp, uux, or uuto. These commands create a job file in the /usr/spool/uucp/sysname directory. The program uucico then runs at some point and calls the remote system. The uucico program then logs in and transfers the files; afterward, the program uuxqt is run to complete the processing on the remote system. UUCP is a batch-processing service: it may take a while for the job to be processed, depending on whether the system administrator restricted the times when the uucico daemon executes.

UUCP, like most networking tools, has a specific form of machine addressing. This form is commonly referred to *bang addressing*. Machines in the UUCP command are separated by "!" (pronounced "bang"), with the file name as the final component, as in this syntax:

```
thumper!unilabs!choreo!/tmp/transfer
```

A file travels from the machine thumper, to unilabs, to choreo, where it ends up as the file /tmp/transfer. UUCP commands do not understand Internet or Domain addressing, which is typically used in TCP/IP and many mail applications. Internet addressing uses the "@" symbol to separate machine and user:

```
chare@unilabs.org
```

Users who are working with the C shell know that the "!" has special meaning for that shell. When you use the UUCP commands, remember to 'escape' the "!" with a backslash to prevent the C shell from interpreting the "!", as shown in this example:

```
thumper\!unilabs\!choreo\!/tmp/transfer
```

Note You can execute a UUCP command, but whether the remote system will carry out your request depends on the level of security that has been established on the remote system.

Naming Your Host

Before you read about the process of naming your system, you first need to understand the various names your system already has. In addition to the name your machine uses for networking purposes (issue the uname command to see what this is), it has a system name that describes the operating system. You can find the system name by issuing uname with the -s option (although this can often be overridden). Sample output of the uname command follows:

```
pc:~# uname
Linux
pc:~# uname -s
Linux
pc:~# uname -a
Linux pc 1.0.9 #3 Tue Sep 13 04:45:23 CDT 1994 i486
pc:~#
```

This name is not part of UUCP but is inherent in the operating system itself. Yet another name for your system, called the nodename or hostname, appears when you issue the uname -n command, or the hostname command on those systems that support it:

```
pc:~# uname -n
pc
pc:~# hostname
pc
pc:~#
```

Many implementations of UUCP limit the number of characters allowable in the UUCP name to six or seven; because of this, the UUCP name may not even be the same as the node name.

Even though you can use longer names in the commands, the names of the different machines should be unique for the first seven characters. Consider two machines named "gateway1" and "gateway2." They are the same for the first seven characters; transactions for one machine could possibly be delivered to the other in error.

The Naming Process

If you are not connected to Usenet or the Internet, you basically have free license on what you call your machines. If you are part of Usenet, however, you should contact your Usenet hookup and ask them to check whether the name you want is already in use. Alternatively, you can contact a local UUCP registry, which will verify that the machine that you are connecting doesn't have a name conflict with someone else's machine. Always try to choose a name for your Usenet machine that describes the organization. For example, UniLabs Research Group initially started with the name "unilabs.uucp" for Internet work, and has since registered into the .org network domain. The name, unilabs, had to first be processed by the Canadian UUCP Registry to verify that the name didn't conflict.

Regardless of the existence of the Usenet connection, try to choose meaningful names. Resist the common urge to use names such as "host1," "server22," and "sco111," which do little to describe their purpose. Many organizations follow a common theme to describe their machines. For example, one company the author was involved with initially used cookie names (oreo, pirate, chipsahoy) for their machines, but has since started to incorporate cartoon characters (goofy, mickey, tazDevil).

A side effect of using obscure names is that it is harder for the system administrator to remember which machine serves what purpose. Commonly used names, such as ftp, will highlight these services provided by a particular machine, and invite attention.

The actual mechanism of changing or setting the system name depends on the Unix derivative you are using. On SCO Unix, for example, the name is generally compiled into the kernel; SCO XENIX uses the file /etc/systemid. Check your system documentation to determine the method in which the system name is defined.

The System V Basic Networking Utilities UUCP

With the release of System V Release 3 from AT&T, a new UUCP version appeared. This version is known by a number of names, most notably the Basic Networking Utilities, or HoneyDanBer (HDB) UUCP. HDB provides capabilities that version 2 UUCP lacks, and corrects some design deficiencies of version 2. Additional functions include the support for newer modems and networks.

The fastest way to know if you have Version 2 or BNU is to look in the directory, /usr/lib/uucp, or /etc/uucp. If you see a file named "Systems," then you have BNU. If you find "L.sys," then you have the Version 2 implementation of UUCP, which is described later in the chapter.

UUCP File Layout

UUCP is set up in three directories: /usr/bin, /usr/lib/uucp, and /usr/spool/uucp. A brief description of each file can be found in the following output:

```
/usr/bin
    This contains the user commands, such as
        cu          * interactive dialup access
        uucp        * job scheduler
        uudecode    * decode files
        uuencode    * encode files
        uulog       * show UUCP log files
        uuname      * get UUCP names
        uupick      * pick files
        uustat      * get UUCP status
        uuto        * send files
        uux         * command execution

/usr/lib/uucp or /etc/uucp
        Devices           * Device Configuration
        Dialcodes         * Dial code prefix expansion
        Dialers           * dialer programs
        Maxuuscheds       * configuration parameters
        Maxuuxqts         * configuration parameters
        Permissions       * Security Controls
        Poll              * Polling configuration
        Sysfiles          * file configuration
        Systems           * Calling System information
        Uutry             * debuging/text program
        remote.unknown    * handle unknown callers
        uucheck           * check configuration files
        uucico            * transfer program
        uucleanup         * maintenance program
        uudemon.admin     * maintenance program
        uudemon.cleanu    * maintenance program
        uudemon.hour      * maintenance program
        uudemon.poll      * maintenance program
        uugetty           * UUCP getty program
        uusched           * UUCP job scheduler
        uutry             * debuging/text program
        uuxqt             * remote execution program
```

The location of your system's UUCP files depends on the operating system that you are running. Most System V implementations, for example, have UUCP files in /usr/lib/uucp. Many BSD versions use the directory /etc/uucp. Your system may have other related files in

these directories. For example, SCO Unix has a number of files, such as binary dialer programs for specific modems, in these directories.

Configuring UUCP

Only three files must be modified to bring up a UUCP connection:

- Devices
- Systems
- Permissions

Even if you leave all of the other files the same as they were shipped from the vendor, changing only these three files results in a functioning UUCP system. To make these modifications easier, the files are typically shipped full of comments; you may never need to use any documentation.

The Devices File

Before a connection can be established, the modem and serial devices that can be used must be defined. These devices range from serial lines direct to another computer, to modems, and UUCP over TCP/IP or X.25 connections. The *Devices file* controls what physical devices are available for carrying UUCP connections, and their configuration parameters. Each entry consists of these fields:

```
Name    Device    dialer-port    speed    dialer-token pairs
HAYES    tty2A    -        Any    hayes    \T
```

Each field is required, and must contain either a value or a placeholder (a hyphen). The fields, allowable values, and their purposes are listed in table 3.2.

Table 3.2
The BNU Devices File

Field Name	Allowed Values	Description
Name		This field is the name or type of the device. In the preceding example, the name HAYES is used.
	ACU	This is an Automatic Call Unit, or modem. The modem may be directly attached to the computer, or accessed through a LAN switch, according to the UUCP documentation. ACUs or modems are usually attached directly to the system.

continues

Table 3.2, Continued
The BNU Devices File

Field Name	Allowed Values	Description
	Direct	This field indicates that this is a Direct link to a remote system, modem, or LAN Switch. The uucico program does not use lines of this type, only cu does. Note the capital D; this is an important note when looking at the available dialers. In the case of modems, a Direct line is also defined so that a command like "cu -l tty??" will work. This entry is also useful for configuring the modem.
	sysname	This field is an entry for connection to a specific system, perhaps using some other specialized dialer for routing through a LAN switch.
Device		This is the actual device that is used to establish the connection. For direct serial and modem links, this field contains the name of the actual device. For TCP connections, this field will say "TCP."
dialer-port		This is a carryover from the days when the dialer was separate from the modem. This is an optional field, and may be used if the Name keyword is "ACU" and the dialer is an 801 dialer. If you don't have an 801 dialer, put a hyphen (-) in this field.
speed	a single speed a range Any	This is the speed that connections will be accepted at for this device. It may a single speed, such as 2400, or the word "Any", which will match any requested speed, as defined in the Systems file. The speed may also be a range of values indicated by value–value, as in 300–2400.
dialer-token pairs		This field identifies the dialer program that will be used in conjunction with a specific modem dialer. The token parameter identifies what type of processing is to be used on the phone number. Often no token is used. The two tokens available are \T, which advises uucico that the phone number should be processed using the dialcodes file, and \D,

Field Name	Allowed Values	Description
		which indicates that the phone number in the Systems file is to be used. The dialer program named in this field can be a binary program in some implementations of UUCP, direct for use with direct serial connections, or the name of a dialer program (found in the Dialers file).

Sample Device file entries and explanations are shown here:

```
Name/Type    Device    Dialer    Speed    dialer-token
INTACU       tty01     -         1200     intacu
```

This entry defines a device named INTACU, which is accessed through /dev/tty01, using the intacu dialer program, at only 1200 baud.

```
TBIT         tty17     -         Any      TBIT
```

This entry defines a device named TBIT, which is accessed through /dev/tty17 using the TBIT dialer program. This entry allows any connect speed requested in the Systems file.

```
bugs         tty21     -         9600     direct
```

This entry defines a device named bugs, which is a name of one of the machines in the author's local UUCP network. The dialer program is direct, indicating that this is a direct connection, and connections are done at 9600 baud only. It is common for the device in a dedicated connection to be named the same as the remote system in order to establish system specific values, or enable certain permissions.

```
TCP          TCP,e     -         Any         TCP         540
```

The preceding entry defines a device used for TCP/IP connections using the "e" protocol. There is no dialer port, and any speed can be used. The TCP in the dialer field indicates that this is a TCP connection to be made on the port specified in the token field, TCP port 540.

Now that the entry in the Devices file has been created, file ownership must be discussed. UUCP requires that the Device files it uses are owned by UUCP, and have a group of UUCP. Failure to do this will result in errors about not being able to access the device in question.

Evaluating the owner and group is done using the ls command. If the group owner for the device, such as /dev/tty17, is not set to uucp, it is possible for connection problems with the device to occur.

Testing the Connection

Before you read about setting up the Systems file, you can test the existing connection using the cu command. Assume you are using the following device definition in the Devices file:

```
bugs          tty21     -         9600     direct
Direct        tty21     -         9600     direct
```

To test the connection to see if it is live, the use of cu is required, as illustrated in the following output (comments are marked in []):

```
chare@unilabs> cu -l tty21
Connected           [The Connected prompt indicates that we have
                    connected to the DEVICE, not necessarily to the
                    remote system.]
<RETURN>            [ Get the remote hosts attention]
login: anon         [ login ]
Welcome to Unilabs Research

CONNECTED TO : bugs.unilabs.org

anon@bugs>
anon@bugs> date
Tue Dec 29 21:00:11 EST 1992
anon@bugs> exit

bugs!login:
~[unilabs].               [ To disconnect from the remote system after logging
                    out, use the ~. command of cu to terminate the
                    connection.]
Disconnected               [ The connection to the device is closed.]
chare@unilabs>
```

If no communication is established with the remote machine, you should debug the actual connection by verifying the media used to make the connection. In the preceding example, debugging involves making sure the cable is plugged in at both ends, that the other machine is running, and the port is enabled for login. If that doesn't correct the problem, then it may be a communication issue regarding the cable. For example, the cable does not have enough RS-232 signals to establish communication between the modem and the computer system. Whatever the reason, when the connection is verified as working, you can continue with the setup of the UUCP files.

The Dialers File

The Dialers file contains a series of characters that are sent to a modem to initiate a phone call. In some UUCP implementations, such as SCO Unix, dialer programs may also be a binary program. SCO also provides a mechanism that uses a binary program and a custom dialer

script to communicate with the modem. It is more common, however, to see dialer scripts developed and implemented through the Dialers file. Each entry in the Dialers file has three parts:

```
Type     801     Chat Script
hayes    =,-,    "" \dAT\r\c OK\r ATL1M1\r\c OK\r \EATDT,,\T\r\c CONNECT
intacu   ==-,    "" \K\dEN\dAT\r\c : Q\c + \021\c "" \pEN\c + \0041 + AT\r\c :
D\T\r\c COMPLETE
TBIT     =W-,    "" ATZ\r\c OK\r A\pA\pA\pT OK ATQ4E0X0 OK\r
ATS53=1S48=1S50=255S51=255S101=0S111=30\r\c OK\r ATDT\T\r\c CONNECT
```

The Type field defines the type of device. For example, the hayes entry defines a Hayes 2400 baud modem; intacu defines a 1200 baud internal modem; and TBIT defines a Telebit Trailblazer Plus. The second field provides the translations for the 801 dialer codes. This field isn't used anymore, and is typically filled with =,-,. The third field is a chat script that works similarly to the chat script in the Systems file, discussed later in this chapter. The chat script is a series of expect-send pairs, meaning expect this, and when it is received, send this. The special characters, preceded with a backslash (\), are defined in table 3.3.

Table 3.3
Dialer File Special Characters

Sequence	Description
\p	Pause (approximately 1/4–1/2 second delay)
\d	Delay (2 seconds)
\D	Phone number/token
\T	Phone number with Dialcodes and character translation
\N	Insert a Null byte
\K	Insert a break
\E	Turn on echo checking (for slow devices)
\e	Turn off echo checking
\r	Insert a Carriage return
\c	Do not insert a new-line
\n	Send new-line
\nnn	Send octal number

To see how this dialer script functions, look at the operation of the Hayes dialer entry.

```
Expect          Send
"" (nothing)    \dAT\r\c (delay, send AT)
OK\r            ATL1M1\r\c
OK\r            \EATDT,,\T\r\c
CONNECT
```

This chat script is being used to communicate and configure the modem prior to making the call. In the 4 send sequence, when the modem is sent the string "\EATDT,,\T\r\c", it is being told to turn on echo, and to dial the phone number. The dialer token \T causes the phone number to be put in place when the script is actually used.

The Systems File

Now that the device has been defined in Devices, and the connection to the device, either modem, or remote system has been verified using the cu command, the Systems file can be set up to allow calls to be made by uucico and cu.

The *Systems file* defines the names of the UUCP systems that your system can connect to. In other words, you don't have to make an entry for every machine that might call you (you can use anonymous UUCP), but only for those machines that you want to call your machine. Each entry is in the format:

```
System_Name Schedule Type Speed Phone_Number Login_script
```

For example, consider this entry in unilabs' system file:

```
gateway Any INTACU 1200 9999999 "" \d\r\d\r ogin: nuucp22 word: testing
```

The *System_Name* (gateway) is the name of the remote machine you are trying to contact. Many UUCP implementations truncate the system name to seven characters. For this reason, this name should be unique to seven characters. More modern systems, however, allow the use of longer names.

Time_to_Call is the calling schedule for this machine. That is, you can restrict when this machine can be called. This can be handy if the link between the machines is expensive. This field can contain a number of different values and components, including the day of the week, and the start and end hour. The time or schedule field uses the following values to indicate the day of the week:

Mo Monday

Tu Tuesday

We Wednesday

Th Thursday

Fr	Friday
Sa	Saturday
Su	Sunday
Any	Any day of the week, at any time
Never	For machines that call you, but you don't call
Wk	Any weekday excluding Saturday and Sunday

The time specification for the start and end time is given using the 24-hour clock to distinguish between 5:00 a.m. and 5:00 p.m. The specification can span midnight, as in 2310–0700, allowing calls from 11:10 p.m. to 7:00 a.m. You should, however, consider how you select times with uucico. uucico itself interprets the times on a same day basis. In the previous example, uucico interprets the times to allow calls from 0000–0700, and 2310–2359 on the same day. This is really only a problem if you combine the time with a specification such as Wk, or specific days. For example, Wk2310–0700 works for Mon–Fri, but does not allow calls after 2359 Friday night, until 0000 Monday morning. Consequently, careful thought must be given to the call specification.

The schedule fields can have only a day specification, with no time value, which means that the system can call any time of the day. The time subfields, however, *must* have a day specification. If you want to allow calls between 2310–0700 every day of the week, for example, you would use the specification Any2310–0700 in the schedule field.

It also is possible to have multiple calling periods for a single system by separating each of the periods with a comma. Suppose, for example, that you want to allow calling only between 2300–0700 weekdays, and any time on the weekends. This would be written as Wk2300-0700,SaSu. This multiple time specification is not supported in all implementations. Consequently, some experimentation may be required.

The schedule field also may specify an optional retry number, which is used to change the default number of minutes before another attempt will be made to call this system. BNU systems default to a ten-minute delay (600 seconds), which increases with each unsuccessful connection. To specify this retry, follow the schedule with a semicolon, and the retry period in minutes, as in Any;2—a retry period of 2 minutes. This doesn't invoke uucico any quicker, but instructs uucico the next time it is invoked to wait for this period of time to elapse before attempting to connect again.

The *Type* field specifies the name of the device that will be used; this device must also appear in the Devices file. Any name defined in the Devices file is valid here. To use a TCP connection, the device named here is TCP, and includes the protocol to be used. After the device is named, an optional field enables you to specify the protocol for the call. The default is to use protocol "g," which is not terribly efficient when using a high-speed reliable transport such as

TCP/IP. For this purpose, the "e" protocol was created. Available protocols are discussed in the final part of this chapter. In addition, the TLI and TLIS transports can also be used with correct configuration.

The *Speed* field is used to define the speed of the connection to this system. This is defined in baud for the connection. No keywords are permitted.

Phone is the actual number to dial and is used with modems. Direct connections mark this field with a hyphen (-). The phone number can include any number of things, including dialcodes and support for calling card numbers. If, for example, you must dial 9 to get an outside line, you need to pause the dialer to wait for the secondary dial tone. The equal sign (=) is placed in the number to provide a delay. To access an outside line, 9= can be placed in the phone number. In some implementations of modem software, the comma generates a pause. It might be necessary to review your modem's documentation to determine the character for a delay.

The phone number may also contain an English word dialcode listed in the Dialcodes file. You can use English to describe each phone number and who it is for, and then let the system expand the listing into the actual numerical area code. Unfortunately, this helpful feature is not automatic. You need to list the area codes you want in the Systems file. This new file, Dialcodes, is described later in this chapter. *Dialcodes* are English words that can be used in place of area codes and dialing prefixes. The following sample file has two fields: the first is the English keyword, and the second is the dialstring that will be substituted:

```
toronto    9=1-416
```

When you specify the following phone number in the Systems file:

```
toronto5551212
```

it will be expanded to the following prior to dialing:

```
9=1-4165551212
```

In the original paper of Version 2 UUCP (written by David Nowitz), the Dialcodes file was originally intended to allow for the use of the same Systems files at different sites, simply by using different Dialcode files. The advantage to using this format is that only the Dialcodes file has to be different for the given site.

Some sample phone numbers are listed here:

```
4581422
14084291786
```

These two examples are simple phone numbers. The following number uses the comma to create a pause for a secondary dialtone.

```
9,,5551212
9,,06135551212,,,,,,,,123456789012
```

The preceding number uses several pauses to wait for a calling card tone before a calling card code is entered. The following example shows a number with a dialcode that will be expanded to provide the area code. See the section on Dialcodes for more information.

```
chicago8101242
```

The UUCP Chat Script

The final and possibly most difficult part of a UUCP entry is the chat script. The *chat script* is a combination of expect-send pairs that define the login sequence to gain access to the remote computer. Each pair is separated by a space, with optional subexpect-subsend pairs separated by hyphens. Consider the following example:

```
login:-BREAK-login: nuucp word: loginAok
```

uucico expects the remote system to print login:. If it doesn't see one within a predefined period of time, send a BREAK signal, and expect login:. The BREAK signal is a modem break, which may wake up a getty running on the remote system, or cause that getty to switch speeds to something more reasonable. When your system sees the login:, it sends nuucp, and waits for the word: string. When your system receives see it, you send loginAok, which is your password. When the script completes all these steps successfully, your workstation has logged in to the system.

If you don't use the subexpect-subsend pairs, such as in the following command, you might not be able to log in if the system answers at a different speed from the speed at which you are calling. This type of mixup would prevent you from seeing the login prompt.

```
login: nuucp word: loginAok
```

Suppose, for example, that you are calling at 1200 baud, and the remote system answers at 2400 baud. You would need to send a BREAK twice, assuming that the related gettydefs entry goes from 2400->300->1200. Your chat script would look like this:

```
login:-BREAK-login:-BREAK-login: nuucp word: loginAok
```

The difference to note between the primary expect-send and the subexpect-subsend is that the subexepect-subsend will only be used if the expect string is *not* received. It should be pointed out that uucico stops looking at the incoming characters after a match is found for the expect text. However, it is commonplace to use the last text expected to ensure that the send sequence isn't sent to the remote system too soon.

Before you can define the chat script, you need at least the uucp login name and password you use to access the remote system. You should then use the cu command to contact the remote system to find out what you need to define for the script. A sample session's output look likes this:

```
chare@unilabs> cu -l tty21
Connected

<send NEWLINE>
login: nuucp
Welcome to Unilabs Research

Shere=bugs
~.
```

To create a chat script for the system bugs, which is directly connected to the sample system, the setup looks like this:

```
expect nothing
send newline
expect login:
send nuucp
```

This setup translates into a chat script that looks like this:

```
"" \r\c login: nuucp
```

The pair of double quotes means "expect nothing"; \r\c is "send newline." These special characters are used in chat scripts along with the extra characters listed in table 3.4.

Table 3.4
Chat Script Special Characters

Sequence	Meaning
""	Expect null string
EOT	Send an End Of Transmission character
BREAK	Send a BREAK signal (may not be implemented on all systems)
@	Same as BREAK
\b	Send a backspace
\c	Suppress newline at the end of the string
\d	Delay for one second
\E	Start echo checking. (From now on, whenever a character is transmitted, it will wait for the character to be received before doing anything else.) Not implemented on all systems
\e	Turn echo check off
\K	Same as BREAK (BNU only)
\n	Send newline or linefeed character

Sequence	Meaning
\N	Send a NULL character (BNU only)
\0	Send a NULL character
\p	Pause for a fraction of a second (BNU only)
\r	Send a carriage return
\s	Send a space character
\t	Send a tab character
\\	Send a \ character
\nnn	Translate the octal digits nnn into a single ASCII character and sends that character

The chat script used in the preceding example is very simple. Chat scripts get much more complicated depending on the intended connection. The following sample chat scripts range from the simple to the complex:

```
in:--in: nuucp word: panzer
     Expect in: if not received, send a NULL string (newline)
     send nuucp
     expect word
     send panzer

"" @ ogin:-@-ogin:-BREAK-ogin:-BREAK-ogin: uucp word: frogin
     expect nothing
     send @ (equivalent to BREAK)
     expent ogin: if not received, send BREAK, maximum of two times
     send uucp
     expect word:
     send frogin

"" \r\d\r\d ogin:-BREAK-ogin:-BREAK-ogin: anon
     expect nothing
     send newline
     delay
     send newline
     delay
     expect ogin: if not received, send BREAK, maximum of two times send anon

"" \r\d service: uucp ogin: mimi word: guesswho
     expect nothing
     send newline
     delay
     expect service
     send uucp
```

```
expect ogin:
 send mimi
 expect word
 send guesswho
```

Warning When you use UUCP over TCP/IP, do not include BREAKs or other subexpect-subsend combinations. They are not needed and can often lead to problems in establishing a connection.

Now that you know how the System file is formatted, here are some additional examples. (Notice that these examples also include UUCP over TCP/IP. This configuration is described later in this chapter.)

```
sosco Any,15 TBIT 9600 14084291786 "" \r\d\r\d ogin:-BREAK-ogin:-BREAK-ogin: uusls
unilabs.org Any  TBIT 1200 5551211 "" \r\d\r\d ogin: nuucp word: nothing
uunet.ca Any TBIT 19200 5551210U "" \r\d service: uucp ogin: mimi word: none
uunet.ca Any TCP,e Any - ogin: mimi word: none
sffoo Any STARLAN - sffoo in:—in: nuucp word: nuucp
```

Testing the Connection—Using uucico

Now that the Systems file is set up, it is important to verify that your configuration to this point is correct. The uucico command is used with the -x option to specify a debug level, or with the uutry (Uutry on some systems) command.

The most difficult part of working with uucico is interpreting its error messages. Sometimes their messages are Greek to even the most experienced system administrator. To test the connection and the uucico process, you can use the -x# option with uucico. This option prints debug information on-screen. The # is a debug level, ranging from 1 to 9, with 9 being the maximum amount of debugging information available. From implementation to implementation, the amount of information provided at various levels depends on how the programmers coded it when they modified UUCP for their vendor. The following debug output illustrates a terminal session:

```
chare@unilabs> /usr/lib/uucp/uucico -r1 -x9 -sbugs
conn(bugs)
Device Type bugs wanted
getto ret -1
Call Failed: DEVICE LOCKED
Conversation Complete: Status FAILED
```

The debug output displayed is created by using the option -x followed by a level number. The higher the number, the higher the level of debug information that is printed. As mentioned earlier, the highest value is 9. In the following example, the connection to the remote system named bugs failed because the device is locked, meaning someone else is using the device. The only recourse you have is to wait until the device is free. Another attempt at establishing communications with the remote system is shown in the following:

```
chare@unilabs> /usr/lib/uucp/uucico -r1 -x9 -sthumper
conn(thumper)
Device Type thumper wanted
getto ret -1
Call Failed: CAN'T ACCESS DEVICE
Conversation Complete: Status FAILED
```

Note that the preceding example displayed a different error message: `CAN'T ACCESS DEVICE`. To correct this problem, suspect that the owner and group owner for the associated device file is incorrect. For example, if the connection attempts to use /dev/tty16 when connecting, check the ownership using the ls -l command. If the owner is not uucp, the UUCP-related programs will not be able to open the device. Reviewing the permissions for the associated device using the ls -l command shows that the owner of the device file is root, not uucp. As a result, uucp couldn't open the device. The following setup shows the next attempt at contacting the remote system:

```
chare@unilabs> /usr/lib/uucp/uucico -r1 -x9 -sbugs
conn(bugs)
Device Type bugs wanted
getto ret 6
expect: ("")
got it
sendthem (DELAY
^MDELAY
^M^M)
expect: (ogin:)
^M^Jroot@bugs> ^M^J^M^Jroot@bugs> root@bugs> timed out
Call Failed: LOGIN FAILED
Conversation Complete: Status FAILED
```

Security Concerns during Log Off

Whenever UUCP is used, security must be considered for the host system. To illustrate how the use of UUCP can create security problems, suppose that when the last user to use the port exited, he or she didn't log out, and the remote system is not set up to drop the line when the connection closes. This type of scenario should be immediately reported to the system administrator of the other machine. Note that in this example you could not access the machine because it couldn't successfully complete the chat script.

```
chare@unilabs> /usr/lib/uucp/uucico -r1 -x9 -sbugs
conn(bugs)
Device Type bugs wanted
getto ret 6
expect: ("")
got it
sendthem (DELAY
^MDELAY
^M^M)
expect: (ogin:)
^M^Jroot@bugs> ^M^J^M^Jroot@bugs> root@bugs> timed out
```

```
Call Failed: LOGIN FAILED
Conversation Complete: Status FAILED
```

When this occurs, anyone who placed a call to this machine gets a terminal session with a shell and the access privileges of the last user, which is root in this case.

Correcting the problem involves making sure that your cable has the appropriate RS-232 signals to receive Carrier Detect, and the hardware flow control signals. It is also important to ensure that the correct device is being used. On SCO Systems, for example, the /dev/tty1a device does not use modem control signals; therefore, a modem on this device will not hang up properly. This is solved by using the /dev/tty1A device.

This is only an example because the actual device name used on your system may be different for operating system implementation.

A successful UUCP transaction is shown in the following sequence of commands (the author's comments are enclosed in brackets):

```
chare@unilabs> /usr/lib/uucp/uucico -r1 -x9 -sbugs
conn(bugs)
[Connection wanted for bugs]
Device Type bugs wanted
[I want a device type known as bugs to make the connection]
getto ret 6
[Set the speed ]
expect: ("")
[Start the chat script - expect nothing ]
got it
[ OK ]
sendthem (DELAY
^MDELAY
^M^M)
[Send \d\r\d\r ]
expect: (ogin:)
^M^Jlogin:got it
[ OK ]

sendthem (unilabs^M)
imsg > ^M^J^M^Jlogin: login: unilabs^M^JWelcome to Unilabs Research^M^J^M^J
     ^M^J¦              SPECTRIX Microsystems Inc.
¦^M^J¦      Xenix-68000      Vers. 3.0      Release 1.0
¦^M^J¦_____________________________________________________¦^M^J^
M^JCONNECTED TO : thumper.unilabs.org    (613) 834-1439^M^JSYS
MANAGER  : Chris Hare      ^M^JMAIL TO      :
chare@unilabs^M^JWARNING      : Full System Security is
operational.^M^M^JPRIMARY ON-LINE STORAGE^M^J^PShere^@Login
Successful: System=o bugs
[ Login established ]
imsg >^PROK^@imsg >^PPg^@wmesg 'U'g
Proto started g
[ Protocol negotiation ]
```

```
wmesg 'H'
[ I have no work - request hangup ]
wmesg 'H'Y
[ Remote has no work - hang up ]
send OO 0,imsg >^P^I^H*"^I^P^B^Q^^^R^_HY^@imsg
>^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^A^R^X^@^_vJ
^@^AAn^@^@^@2^@^@^@^@ }^@^@^@^@^@^@^@^@^@^P^I"*^H^I^P^I"*^H^I^POOOOOO
^@Conversation Complete: Status SUCCEEDED
 [ Call terminated ]
```

Through this seemingly simple transaction, you now know that the UUCP connection is working. An important transaction on the first imsg line takes place concerning the validation of the calling systems. This transaction ensures that the two machines know each other.

The slave machine, usually the one being called, prints the message `Shere=`*`nodename`*, which means "system here"—*nodename* is its machine name. The master usually replies with *`Snodename-Qseq-xnum.nodename`* is the master's nodename; *seq* is the sequence number or zero if not used; and *num* is the level of debugging in effect. This is not consistent across implementations; as in the earlier example, the slave only prints `Shere`.

If the two machines agree to talk to each other, the slave prints the message `ROK`, and the call continues. Another place to look for debugging information is in the Log files that are kept by the various UUCP commands. These are presented later in this chapter under the heading "UUCP Log Files."

During the debugging process, it may be necessary to circumvent the normal retry mechanism so that you can fix a problem and immediately call back. UUCP status files are kept on a machine-by-machine basis in the directory /usr/spool/uucp/.Status. Currently this directory on this system has the following files that record status information on these three systems:

```
$ ls -l
total 3
-rw-rw-rw-   1 uucp     other         32 Dec 30 00:04 bugs
-rw-rw-rw-   1 uucp     other         45 Dec 30 00:11 thumper
-rw-rw-rw-   1 uucp     other         38 Dec 30 00:11 wabbit
```

The status file looks like this:

```
Type  Count   Time         Retry    Status         System
0     0       725691861    0        SUCCESSFUL     bugs
6     1       725692313    300      LOGIN FAILED   wabbit
```

These two examples were retrieved from separate files. The fields are defined as follows:

- **Type.** A code that indicates the status of the job. This is explained in the status field.
- **count.** The number of call attempts made by the local system. This is incremented each time uucico tries to call the remote system.

- **time.** The time in seconds since January 1, 1970, when this system was last called. Listing 1 at the end of the chapter has a program called gtimes that will print the correct text for this time.
- **retry.** When should the system retry the connection again. This value is stored in seconds.
- **status.** A text description of the status of the last call.
- **system.** The name of the system.

Status files are updated and removed periodically during the normal course of operation. In the event that UUCP fails abnormally, however, you may have to remove status files. In the case of the line for the system wabbitt, you won't be able to call this system for 300 seconds. To circumvent this retry limit, remove the status file for the desired system from /usr/spool/uucp/.Status/system. This will make uucp think that there wasn't a problem, and the call will be made immediately.

Permissions File

Now that the Device and Systems entries are configured and verified, the next step is to examine the security of the UUCP system. The normal user account protections are still in place with most UUCP accounts. One account that deviates from standard security is anonymous UUCP access, if you allow it. The login/password combination may differ from normal protection mechanisms. For this and other reasons, you should always configure your UUCP accounts to include passwords: This makes it harder for the bad guys to break into your system through UUCP. The standard issues surrounding passwords should still be considered, however.

The structure of the Permissions file enables you to control which authorizations are given to the remote system when that system calls you, and also which authorizations you have when you call a remote system. As an administrator, you may choose to use only the default values for each of the fields in the file. A sample Permissions file entry appears here:

```
LOGNAME=choreo    \
REQUEST=yes        \
SENDFILES=yes    \
READ=/             \
NOREAD=/etc         \
WRITE=/tmp:/usr/tmp:/usr/spool/uucppublic    \
CALLBACK=no        \
COMMANDS=ALL
```

The backslashes on each line indicate that the line continues as if it were all on one line. This example does not include all of the options, and this in a non-default entry. (Each option is explained shortly.) The default permissions associated with a login are shown as follows:

```
Default Permissions
READ=/usr/spool/uucppublic
WRITE=/usr/spool/uucppublic
REQUEST=no
SENDFILES=call
COMMANDS=rmail
```

Each entry must have either a LOGNAME or MACHINE option, or both. The LOGNAME option indicates that this entry is used when the system calls in and logs in using the logname specified. MACHINE entries are for when users on your system CALL the remote machine, and not for when the remote machine calls you: that is covered by LOGNAME. LOGNAME and MACHINE entries may be combined, but they don't have to be. However, if you want complete control over systems that access your machine using UUCP, separate login ids and combined LOGNAME/MACHINE entries are necessary in the Permissions file. All of the keyword options that can be used are defined in table 3.5. The L in the CLASS column indicates that the option applies to LOGNAME entries. M is for MACHINE entries, and both means that the option is applicable to both.

Table 3.5
Permissions File Keyword Definitions

Option	Class	Definition
LOGNAME	L	Specifies the login ids for remote sites that access the local site. LOGNAME=thumper LOGNAME=thumper:wabbit
MACHINE	M	When the named machine calls your computer, this option specifies the conditions that are in effect. MACHINE=wabbit MACHINE=wabbit:bugs
REQUEST	M, L	This option determines whether the remote system can set up UUCP transfers from your computer system. Allowable values are "yes" or "no". REQUEST=no REQUEST=yes
SENDFILES	L	The value of this option determines whether the called site can execute locally queued jobs during a session. A value of "yes" means that your system may send jobs queued for the remote site as long as it is logged in using one of the names specified in the LOGNAME field. The default value is "call," which means that the queued files are sent only when the local system calls the remote system.

continues

Table 3.5, Continued
Permissions File Keyword Definitions

Option	Class	Definition
		SENDFILES=yes SENDFILES=call
READ	M, L	Names the directories that uucico can read from when requesting files. The default is to use /usr/spool/uucppublic. Multiple directories can be named by putting colons between directory names. READ=/tmp:/usr/tmp:/usr/spool/uucppublic READ=/
NOREAD	M, L	Names the directories that are to be excluded from the READ list. NOREAD=/etc
WRITE	M, L	Names the directories that uucico can write to for depositing files. As with READ, the default directory is /usr/spool/uucppublic. WRITE=/tmp:/usr/tmp:/usr/spool/uucppublic WRITE=/
NWRITE	M, L	Identifies the directories that are excluded in the WRITE list. NOWRITE=/etc:/bin:/usr/bin
CALLBACK	L	Setting this value to "yes" instructs the local system to call the calling system back before allowing any work to be done. This is a good feature for enhanced security. Just make sure you don't set both systems to CALLBACK=yes, or nothing will get done. CALLBACK=no CALLBACK=yes
COMMANDS	M	Defines the commands that the remote system can execute locally. The defaults are different from system to system, and are defined in the source code for uuxqt. Multiple commands can be listed by separating each with a colon. The keyword ALL is allowable and permits the use of all commands. Keep in mind, however, that this is a security problem. Do not include uucp as a command, unless you will allow other users to route UUCP jobs through your machine.

Option	Class	Definition
		COMMANDS=rmail COMMANDS=rmail:rnews:uucp COMMANDS=ALL
VALIDATE	L	Helps validate the calling system when potentially dangerous commands are executed. The value is a list of system names that are permitted to use this logname. Multiple names may be used. VALIDATE=unilabs VALIDATE=unilabs:wabbit
MYNAME	M	Used to circumvent the name length restrictions that were discussed earlier. Note, however, that MYNAME is used only when the local machine calls out, and not when a remote machine calls in. MYNAME=testdrive
PUBDIR	M, L	Specifies the name of the directory used for the public UUCP directory. Typically this is /usr/spool/uucppublic. PUBIDR=/usr/ftp

Keep in mind the following rules when defining Permissions entries for systems:

- Blanks are not allowed before or after the equal sign in the assignment.
- Multiple option=value pairs can be combined on a single line, but they must be separated by a space.
- Each line is one entry, although the line can be continued onto the next line by using the backslash (\) as a continuation character.
- Comment lines start with a pound (#) symbol, and end with a newline.
- Remote system names may appear in one and only one LOGNAME entry.

Some sample Permissions entries are shown as follows:

```
#
# For ANONYMOUS LOGINS
#
    LOGNAME=nuucp

# With this entry, any machine that calls in and logs in using the
# login name nuucp will use this LOGNAME entry, which sets the
# default values for the remote machine.
.
```

```
#
# For dealing with wabbit - AT&T 7300
#
    LOGNAME=wabbit    \
    MACHINE=wabbit    \
    REQUEST=yes        \
    SENDFILES=yes    \
    READ=/            \
    NOREAD=/etc        \
    WRITE=/tmp:/usr/tmp:/usr/spool/uucppublic    \
    COMMANDS=ALL

# This entry is used when "wabbit" logs in to your machine, or when
# you call wabbit. In both cases, the permissions are extensive and
# very liberal. If you manage both machines, this shouldn't be a
# problem. Notice that even though wabbit can read files from any
# directory, access to /etc/ has been explicitly blocked off to
# prevent the retrieval of the password file.
```

Some advanced security measures can be put into place to further protect your system from "marauders." The SENDFILES, REQUEST, VALIDATE, and CALLBACK options are, after proper login and password controls, the next step in UUCP security measures. VALIDATE validates the hostname of the calling system, and terminates the call if the caller does not have one of the listed hostnames. The following line shows how this option is used:

```
LOGNAME=choreo    VALIDATE=choreo:gateway
```

If a system logs in to the machine using the logname of choreo, but it isn't the machine choreo or a gateway, the call will be terminated. To further inform you of who is calling, the CALLBACK feature can be used. With this option, the calling system is told during the negotiation that your system must call them back. Your system terminates the call, and then uses the information in the Systems file to call the remote machine. In this case, no work will take place until your system calls back the remote machine. The CALLBACK feature can also be used to determine who will pay for the phone call.

The SENDFILES option determines whether your system will send locally queued work after the remote has finished with its requests. Suppose, for example, that a machine called wabbit and another called thumper are communicating. A user on thumper initiates a request to send a file to wabbit. A user on wabbit starts a UUCP job to send a file to thumper. The Permissions entry for thumper on the machine wabbit has SENDFILES=call. The system thumper calls wabbit, logs in, and sends the file to wabbit. The call is terminated at this point, even though wabbit has a job to send to thumper. The call is terminated because the SENDFILES=call option indicates that jobs will NOT be sent to the remote machine if it calls; only if the local machine calls the remote machine.

In addition to SENDFILES, you can set the value REQUEST=no so that remote systems cannot request files from you. With both SENDFILES and REQUEST set this way, your system is pretty much in control of what is going on, regardless of the CALLBACK and VALIDATE values. The reason: all work destined for outside of your system must originate with your system.

Note Two things to watch for when using SENDFILES and CALLBACK: if both systems are set CALLBACK=yes, or one system is set CALLBACK=yes and the other is SENDFILES=call, no work will be performed in either case.

Allowing Anonymous UUCP Access

As in any situation where unathenticated, or anonymous access to a system is granted, there are certain risks to be borne, and certain steps to take in protecting your assets. This is also true for UUCP. You might need to set up anonymous UUCP access at some point, or disable it. Being able to deal with either case results from a through knowledge of the security problems for such a setup, and how they can be minimized with a proper Permissions entry. During anonymous UUCP setup, you need to include two steps. The first step is to define a generic entry in Permissions:

```
#
# For ANONYMOUS LOGINS
#
LOGNAME=nuucp
```

Make sure a user login account exists on your system called nuucp, and executes the uucico program as its login shell. If this account doesn't exist, make one. By defining only the LOGNAME option, any system that calls and uses the login name "nuucp" is granted only the DEFAULT permissions:

```
READ=/usr/spool/uucppublic
WRITE=/usr/spool/uucppublic
REQUEST=no
SENDFILES=call
COMMANDS=rmail
```

With this configuration, the amount of information users can get is limited. You probably will be setting up anonymous UUCP so that employees or outside users can download information. If this is the case, you may want to change REQUEST=yes.

The second step concerns checking for or creating a file in the /usr/lib/uucp /directory called "remote.unknown." The normal mode of operation in BNU requires that machines must exist in each other's Systems files for the conversation to be successful. If a machine name does not exist, the program searches for remote.unknown. If it is found and is executable, the call is connected, the contents of the file are executed, and the call is then terminated. A sample remote.unknown file is listed here:

```
#!/bin/sh
#ident    "@(#)uucp:remote.unknown    2.3"
FOREIGN=/usr/spool/uucp/.Admin/Foreign
echo "`date`: call from system $1" >> $FOREIGN
```

For each unknown system that is it is missing from the Systems file, the remote.unknown script is executed. To prevent this from occurring, remove the execute bits using the chmod command. This allows calls from all unknown systems.

UUCP Log Files

To track what happens on a system, UUCP keeps records of its transactions in a number of different log files; the command that is currently executing determines which log file is used.

All UUCP log files are stored in the /usr/spool/uucp/.Log subdirectory. This structure has a directory for each of the uucp commands you want to track: uucp, uucico, uux, uuxqt, and a file in each subdirectory for each system you interact with. With this information, you can see a record of the transactions for a given system at any time.

Usually the log file that is examined the most is the uucico command's log file, with the following syntax:

```
user machine  date   time     PID seq comment
```

A log file entry with information in each of these fields would indicate that a conversation was in progress, as illustrated in the following excerpt:

```
uucp gateway  (12/30-22:04:06,1429,0) SUCCEEDED (call to gateway )
uucp gateway  (12/30-22:04:12,1429,0) OK (startup)
chare gateway gatewayN7bd0 (12/30-22:04:12,1429,0) REQUEST (unilabs!D.unila9309b03
----> gateway!D.unila9309b03 (chare))
chare gateway gatewayN7bd0 (12/30-22:04:30,1429,1) REQUEST (unilabs!D.gatew7bd0b03
----> gateway!X.gatewayN7bd0 (chare))
uucp gateway  (12/30-22:04:37,1429,2) OK (conversation complete tty01 75)
```

The log file indicates that a call to a machine gateway started at 22:04 on 12/30, and a connection was established (OK (startup)). A REQUEST was made to transfer a file (unilabs!unila9309b03) to gateway for user chare. A second file was transferred (notice that the SEQ number following the PID has incremented by one), and finally the call was terminated after the files successfully transferred.

Log files are the best way to spot trouble, as shown in this example:

```
uucp thumper  (12/30-20:41:03,1213,0) CONN FAILED (CAN'T ACCESS DEVICE)
uucp thumper  (12/30-21:11:03,1280,0) FAILED (generic open)
uucp thumper  (12/30-21:11:03,1280,0) CONN FAILED (CAN'T ACCESS DEVICE)
uucp thumper  (12/30-21:41:03,1351,0) FAILED (generic open)
uucp thumper  (12/30-21:41:03,1351,0) CONN FAILED (CAN'T ACCESS DEVICE)
uucp thumper  (12/30-22:11:04,1464,0) FAILED (generic open)
uucp thumper  (12/30-22:11:04,1464,0) CONN FAILED (CAN'T ACCESS DEVICE)
```

The log file indicates that a number of unsuccessful attempts have been made to contact the remote machine thumper. This probably should be investigated because the error message

indicates that there is a problem accessing the device. This could be the result of a problem with permissions, ownership, or hardware.

The uux logs show what requests were queued on the local system, as illustrated here. There would be few error messages written to this file:

```
user machine jobid         date  time     pid SEQ comment
news gateway gatewayd7bcf (12/30-18:30:18,926,0) QUEUED (rnews )
chare gateway gatewayN7bd0 (12/30-22:03:22,1425,0) QUEUED (rmail rdpub.com!martha)
```

The log for uuxqt is similar, in that it simply records which commands were executed at the request of a remote system. The error messages described in table 3.6 would most commonly show up in the .Log/uucico/system file.

Table 3.6
UUCP Error Messages

Error Message	Description
`ASSERT ERROR`	An ASSERT error occurred. The message will be stored in the Admins/errors file. See the documentation for the ASSERT messages for your version of UUCP.
`BAD LOGIN/MACHINE COMBINATION`	The node name and/or login name used by the calling machine aren't permitted in the Permissions file.
`CALLBACK REQUIRED`	The remote system requires a callback.
`CAN'T ACCESS FILE`	Either the device doesn't exist, or the permissions are wrong.
`DEVICE FAILED`	The attempt to open the device using the open(2) system call failed.
`DEVICE LOCKED`	The requested device is currently in use.
`DIALER SCRIPT FAILED`	The script in the Dialers file was not negotiated successfully.
`LOGIN FAILED`	The login for the given machine failed. It could be a wrong login/password, wrong number, a very slow machine, or failure in getting through the chat script.
`NO CALL (RETRY TIME NOT REACHED)`	Default time for the System status file has not been reached. Remove this file if you want uucico to try again sooner.

continues

Table 3.6, Continued
UUCP Error Messages

Error Message	Description
`NO DEVICES AVAILABLE`	There may be no valid device for calling the system. Check that the device named in Systems has a corresponding entry in Devices.
`OK`	Things are working perfectly.
`REMOTE DOES NOT KNOW ME`	The remote system doesn't have the name of your system in its Systems file.
`REMOTE HAS A LCK FILE FOR ME`	The remote may be trying to call you, or has a lock file leftover from a previous attempt.
`REMOTE REJECT AFTER LOGIN`	Your system was logged in, but had insufficient permissions on the remote system.
`REMOTE REJECT, UNKNOWN MESSAGE`	The remote system rejected your call with a non-standard message. The remote may be running a hacked UUCP implementation.
`SUCCEEDED`	Self-explanatory.
`SYSTEM NOT IN Systems`	One of your users made a request for a system that was not defined in the Systems file.
`TALKING`	A conversation is currently in progress.
`WRONG MACHINE NAME`	The machine you called is using a different name from the name your system has for it.
`WRONG TIME TO CALL`	The Systems file doesn't allow a call at this time.

A wide variety of programs and even troubleshooting can benefit from the use of the log files. They are not only a source of debugging information, but also a valuable resource in tracking connect times to other systems, and lists of files that have been sent and received.

Maintenance

Now that you have UUCP running, you need to maintain it. Log files constantly consume disk space. Unless they are cleaned out on some type of schedule, they will expand to fill your hard disk. To prevent this, usually a crontab file is supplied for the uucp owner. A sample uucp crontab file is shown in the following:

```
Here are some sample UUCP crontab entries that are used to start the
various UUCP maintenance commands, that were described earlier.

45 23 * * * ulimit 5000; /bin/su uucp -c "/usr/lib/uucp/uudemon.cleanup" > /dev/
➥null 2>&1
41,11 * * * * /usr/lib/uucp/uudemon.hour > /dev/null
0,15,30,45 * * * * /usr/lib/uucp/uudemon.poll > /dev/null
```

The shell scripts uudemon.poll, uucleanup, uudemon.cleanu, and uudemon.hour are provided by Unix vendors to help the system administrator maintain his or her UUCP system. Each of these programs is called at various intervals to perform specific maintenance tasks.

The uudemon.poll program looks at the Poll file, as shown in the following, to determine what, if any, systems should be called:

```
# Lines starting with # are ignored.
# NOTE a tab must follow the machine name

gateway    0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
thumper    0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
wabbit     0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
```

If a system should be called, this program schedules a dummy job to "fake" uucico into thinking that there is work to be done. This is primarily used to "poll" other UUCP systems that do not regularly call your system.

The uucleanup program, typically started by uudemon.cleanup, cleans up the UUCP spool directory (/usr/spool/uucp typically) somewhat intelligently. For systems that cannot be reached, a mail message is sent back to the originator. uucleanup works by deleting locally created rnews files, executing remotely created rnews files, and removing everything that shouldn't be there. A number of options are used to alter the processing done by uucleanup. Unfortunately, each version of uucleanup is different, and you will be forced to check your vendor's documentation for your version's methods.

The uustat command enables the user or the system administrator to obtain various statistics on how the UUCP system is operating. This command can also be used to cancel individual UUCP jobs by using the -k option. To kill UUCP jobs with the uustat command, you must know the UUCP job number, and be the owner of the job, or root.

```
choreoN90cf   02/08-09:09  S  choreo  chare 27332 D.unila96b2919
              02/08-09:09  S  choreo  chare  rmail chare

    # uustat -kchoreoN90cf
    Job: choreoN90cf successfully killed
```

What uustat will not do is cancel all requests or cancel multiple UUCP jobs based on a user or a system.

Configuring Version 2 UUCP

Version 2 UUCP is not as common since BNU (Basic Networking Utilities) and HoneyDanBer UUCP were introduced, although it still can be found on some older versions of Unix.

What Is Version 2 UUCP?

Version 2 UUCP was the first commercial release of UUCP that was included in Unix. It was used widely within AT&T and other organizations until the Basic Networking Utilities were developed. Version 2 was the predominant version of UUCP shipped with the BSD Unix implementation for many years.

To determine if you have Version 2 UUCP, look in /usr/lib/uucp or /etc/uucp. If you find a file called L.sys, then you have Version 2 UUCP. If you find a file called Systems, then you have the Basic Networking Utilities.

File Layout

Version 2 UUCP has a directory structure similar to that of HDB UUCP, with the majority of differences at the file level. Table 3.7 depicts the file layout seen in a typical version 2 implementation.

Table 3.7
UUCP File Descriptions

Directory and File	Description
/bin	Contains the cu command. Note that this command is in a different place than HDB.
/usr/bin	Contains the user commands. Keep in mind that many Version 2 UUCP sites do not have the uuencode and uudecode commands. Commands in this directory include the following: uucp * job scheduler uulog * show UUCP log files uuname * get UUCP names uupick * pick files uustat * get UUCP status uuto * end files uux * command execution uusub * manage a UUCP sub-network

Directory and File	Description
/usr/lib/uucp or /etc/uucp	The UUCP library and configuration file directory. L-devices * device configuration L-dialcodes * cialcode expansion file USERFILE * security controls L.Cmds * security controls L.sys * calling system information L_stat, R_stat \ statistics files used by the uustat command L_sub, R_sub \ traffic statistics used by the uusub command uucheck * check configuration files uucico * transfer program uusched * UUCP job scheduler uuxqt * remote execution program uuclean * administration support command

Now that you know the differences between Version 2 UUCP and BSD, configuring Version 2 UUCP should be easier.

Configuring UUCP

To use Version 2 UUCP, the following files must be configured for a minimal UUCP network. These files must be configured before you can initiate a call to a remote system using either cu or uucico.

```
L.sys
L-devices
USERFILE
L.cmds
```

The L-devices File

The L-devices file contains descriptions for devices that Version 2 UUCP can use to make connections to remote systems. An L-devices entry looks like this:

```
Type    Device    Call-unit  Speed
ACU     tty000    -          1200
```

The *Type* field is usually the type of link that is being used for this device. ACU (automatic call unit or modem) and DIR (direct) are usually the only types supported. Multiple entries of the same type may be listed in the file. In fact, there may be other types supported, depending on the operating system version. For example, BSD Unix supports TCP (TCP/IP) and PAD

(X.25). Very few implementations of Version 2 UUCP actually support any device types other than ACU and DIR.

The *Device* field is the name of the physical device that is used to establish the link. The *Call-unit* field is used only if you are using a Bell 801 dialer and separate modem. You would put the device for the data line in the device field, and the name of the device for the dialer. These dialers are not commonly used any longer and have been replaced with the "smart" modem.

The *Speed* field is the connect speed for this device. Like the entries in the BNU Devices file, the Speed field may contain a range of values, such as 1200–2400, so that connections from 1200 to 2400 baud will work.

Testing the Connection

Before you continue with the Version 2 UUCP configuration, you first need to evaluate the continuity of the connection between these systems. To do this, use the cu command. This command is in the /bin directory in Version 2 UUCP, and the /usr/bin directory in HDB. To run cu on a Version 2 UUCP system, the syntax is as follows:

```
cu -l tty01
```

Substitute the appropriate device name in the command line for tty01. If you get a connected message, you can attempt to log in to the remote system. If you receive an error message such as `NO DEVICE AVAILABLE` or `REQUESTED DEVICE NAME UNKNOWN`, you do not have the L-devices file configured properly. The following output shows a sample cu connection:

```
Comments are marked in []

chare@unilabs> cu -l tty21
Connected
[The Connedt prompt indicates that you have connected to the DEVICE, not
necessarily to the remote system.]
<RETURN>
login: anon
Welcome to Unilabs Research

CONNECTED TO : bugs.unilabs.org

anon@bugs>
anon@bugs> date
Tue Dec 29 21:00:11 EST 1992
anon@bugs> exit

bugs!login:
~[unilabs].
[ To disconnect from the remote system after logging out, use the ~.
command of cu to terminate the connection.]

Disconnected
```

```
chare@unilabs>
```

In this example, the connected prompt printed by the cu command indicates that a connection has been established to the device. After the Enter key is pressed several times, a login message prints from the remote system. After providing a login name, you can establish a session successfully on the remote end. With the knowledge that the connection works, the cu connection is then terminated using the tilde period (~.) sequence.

The next step in setting up Version 2 UUCP is to configure the L.sys file to allow for automated connections.

The L.sys File

The L.sys file is essentially the same as the Systems file in HDB UUCP. The format of each entry is as follows:

```
System   Schedule      Device   Speed   Phone     Chat-script
xray     Wk0800-1700   ACU      1200    5551234   ogin: anon
```

The *System* field identifies what system this entry is used to contact. It must be unique to seven characters for the entries in this file. *Schedule* identifies when you can call the remote system, and uses a series of times and related keywords. The Schedule field can be used to control the cost of a calling system if telephone links are expensive. The contents of this field may contain the following:

Field Contents	Description
start-end	The starting hour and ending hour based on the 24-hour clock; 0800 is 8:00 a.m., and 2000 is 8:00 p.m.
Any	No limit on calling.
Never	No calling permitted.
Wk	Restrict calling to weekdays: Monday to Friday.
Mo, Tu, We, Th, Fr, Sa, Su	Days of the week.

These components can be combined to build a restrictive definition. For example, if you want to allow calling only between 8:00 a.m. and 11:00 a.m. on Saturday and Sunday, the appropriate entry would be as follows:

```
SaSu0800-1100
```

An optional retry subfield can be included by following the schedule with a comma and a time period in seconds. This is the delay period used after an unsuccessful call, not the amount of time after which uucico restarts automatically.

The *Device* type in V2 is restricted to one of the device types allowed in L-devices; either ACU or DIR. When a call is made to the remote system, the device entry is checked against device entries in the L-devices file. If multiple entries exist for a device, the first available device will be used.

The *Speed* field identifies the baud rate to be used when calling the remote system. A corresponding entry in the L-devices file for this speed must exist. The *phone* number is the actual number to call to connect with the remote system. This field contains a hyphen if it is for a direct link to a remote system.

The *Chat-script* field is used to negotiate the login to the remote system. This entry consists of a series of expect-send sequences. The chat script is a combination of expect-send pairs that define the login sequence to gain access to the remote computer. Each pair is separated by a space, with optional subexpect-subsend pairs separated by hyphens. Consider the following example:

```
login:-BREAK-login: nuucp word: loginAok
```

uucico expects the remote system to print login:. If you don't see one within a predefined period of time, send a BREAK signal and expect login:. The BREAK signal is a modem break, which may wake up a getty running on the remote system, or cause the getty to switch speeds to something more reasonable for your system. When your system sees the login: prompt, your system sends nuucp, and then waits for the word: string. When you or your system sees it, send loginAok, which is your password. When the script successfully completes, you have logged in to the system.

You may not be able to log in if the system answers at a different speed from the speed at which your system is calling. This little glitch would prevent you from seeing the login prompt. For example, if you are calling at 1200 baud, and the remote system answers at 2400 baud, you need to send a BREAK twice, assuming that the related gettydefs entry says go from 2400->300->1200. Therefore, the chat script would look like this:

```
login:-BREAK-login:-BREAK-login: nuucp word: loginAok
```

The difference between the primary expect-send and the subexpect-subsend is that the subexepect-subsend will only be used if the expect strings are *not* received.

The uucico program stops looking at the incoming characters when a match is found for the expect text. It is common, however, to use the last text expected to ensure that the send sequence isn't sent to the remote system too soon.

Before you can define the chat script, you need at the very least the uucp login name and password, which you will use to access the remote system. You should use the cu command to contact the remote system to find out what you need to define for the script. A sample session appears here:

```
chare@unilabs> cu -l tty21
```

```
Connected

<send NEWLINE>
login: nuucp
Welcome to Unilabs Research

Shere=bugs
~.
```

To create a chat script for the system bugs, which is directly connected to your system, describe it this way:

expect nothing

send newline

expect login:

send nuucp

This would translate into a chat script that looks like this:

```
"" \r\c login: nuucp
```

The pair of double quotes means "expect nothing;" \r\c is "send newline." These special characters are two of a number of characters used in chat scripts—table 3.3 lists the others. It is wise to avoid the use of subexpect-subsend pairs unless needed, because they can often lead to problems in establishing a connection.

Testing the Connection with uucico

When the L-devices and L.sys files have been configured, communication can then be evaluated with the uucico program's debug option. The process uucico uses is essentially the same for Version 2 and HDB. When the call is initiated, uucico keeps track of the process by creating a status file for the machine in the /usr/spool/uucp directory. This status file is called STST.*machine name*. The status file contains the details of the last connection. If the status file is present when the remote system calls in, the connection is dropped after telling the remote that there is a LOCK file. If the status file is there when you want to make an outgoing call, the call is prevented, and a message is logged saying there is a status file to prevent the call. The contents of the status file look like this:

```
pointer to error code
¦
¦ number of calls
¦ ¦
¦ ¦ time of last call
¦ ¦ ¦
4 1 729570390 LOGIN FAILED unilabs
                ¦            ¦
                status       ¦
                             system
```

The error code is explained in the status part of the entry, which eliminates the need for interpreting the error code manually.

When the status file exists, any other jobs queued for the affected system do not result in a call until the retry time period has been reached. To circumvent this, you can remove the status file.

Version 2 Permissions

In HDB UUCP, one file essentially controls access to commands and files on your system. Version 2 UUCP uses as many as five files; three files are used in the typical configuration:

USERFILE

L.cmds

SQFILE

USERFILE

USERFILE controls access to the files on your system for both remote and local users. It is highly recommended that for each entry in your /etc/passwd file, you create an entry in USERFILE. The code listing section at the end of the chapter includes genUSER, which generates a default USERFILE. The following sample USERFILE is created by genUSER:

```
root,    /usr/spool/uucppublic /
daemon,    /usr/spool/uucppublic /
bin,    /usr/spool/uucppublic /bin
sys,    /usr/spool/uucppublic /
adm,    /usr/spool/uucppublic /usr/adm
uucp,    /usr/spool/uucppublic /usr/lib/uucp/uucico
nuucp,    /usr/spool/uucppublic /usr/lib/uucp/uucico
uucpadm,    /usr/spool/uucppublic /usr/lib/uucp
lp,    /usr/spool/uucppublic /bin
tutor,    /usr/spool/uucppublic
install,    /usr/spool/uucppublic
chare,    /usr/spool/uucppublic
```

Like the entries in the HDB Permissions files, each USERFILE entry defines a number of constraints for file transfer, including:

- Which files on the system can be accessed for UUCP by a local user. Both USERFILE and Unix file permissions must be satisfied for the request to succeed.
- Which files can be accessed by a remote system.

- The login name the remote system must use when calling in.
- The callback configuration the local machine uses to call back and confirm the remote machine's identity.

Not all constraints must be configured, but keep in mind that the fewer the constraints you implement, the greater the risk of a security violation.

USERFILE entries consist of:

```
username,system [c] pathname(s)

uu101,thumper /usr/spool/uucppublic /usr/tmp
uu102,bugs c /u/tmp
```

The *username* (uu101, uu102) defines the name that must be used to log in to the local system. This username must be configured in the ./etc.passwd file using a shell of /usr/lib/uucp/uucico.

The *system* portion defines the name of the remote system. The callback flag, shown as a single "c" separated by a space, is similar to the CALLBACK option in HDB UUCP. If the letter 'c' exists after the system name, then when your system answers a call from the remote system, your system will hang up the phone, and then call the remote system back. In this way you can validate the identity of the remote system.

The *pathname* component(s) is a space-delimited absolute pathname list of directories that are accessible by the remote machine.

Unfortunately, the USERFILE is unnecessarily complicated. The system administrator usually has to spend many hours debugging relatively simple problems. In many cases, the only clue that there is a problem is a loss of security, which usually isn't visible until data has already been compromised on your system!

To maintain consistent security and avoid the headaches associated with debugging USERFILE, keep these suggestions in mind when using uucico:

- Whenever uucp and uux are run by users, and when uusico runs in Master mode, only the username portion of the USERFILE entry is used.
- When uucico runs in Slave mode, only the system name part of the entry is used. Remember that in the course of any conversation, uucico can switch between slave and master any number of times.
- In the USERFILE file on systems that use any version other than BSD 4.2 and BSD 4.3, there must be an entry that has no system name, and an entry with no user name. In BSD 4.2 and 4.3, these entries can be combined into one entry. The non-system name

entity is used when uucico is in Slave mode and has already searched the USERFILE and cannot find a matching entry. The non-username entry is used by uucp, uux, uuxqt, and uucico when in Master mode, only when it cannot find a matching username (in the directory /etc/passwd).

The exact operation and use of USERFILE can differ greatly depending on the implementation of Version 2 UUCP you receive. For this reason, make sure you check the documentation shipped with your operating system.

The following descriptions are for some special USERFILE entries. If no username is specified in the entry, as in the following, any user on the system can request outbound transfers of any file on your system.

```
,xray    /
```

If you don't want to use an entry like this, you will need an entry for EVERY user on your system.

To allow uuxqt file access while uucico is in Slave mode, an entry with no system name must exist in the USERFILE:

```
nuucp,              /usr/spool/uucppublic
```

This entry is used even when uuxqt is started on your local system! Based on what has been presented thus far, you would think that this entry would mean that any system logging in with a username of nuucp will have access to ./usr/spool/uucppublic. Although this may seem intuitive, this isn't exactly true. When the local uucico is in Slave mode, *only* the system name is used to validate file transfers that are requests.

You can also grant individual users special access permissions for certain systems, and then combine the system name and user name entry in the USERFILE file, but you should also have that system call in with its own login name and password. Here is one example:

```
uu101,thumper /usr/spool/uucppublic/ /usr/tmp /u/src
```

It is not uncommon to see people set up entries that look like this:

```
nuucp,              /usr/spool/uucppublic
nuucp,thumper         /usr/spool/uucppublic
nuucp,bugs          /usr/spool/uucppublic
```

There is a problem with this arrangement however. There is nothing to prevent someone from changing the name of his or her system and then calling your system. The reason why this is a problem is that uucico doesn't use the login name when in Slave mode. The best way to limit this danger is to set up individual UUCP login names for each system that will be calling you.

L.cmds

The next component in the issue of security is that of remote command execution, which is defined in the L.cmds file. Typically, the administrator will restrict commands that can be run by a remote system. The L.cmds file is used to limit commands from the remote system. If the command in question is not listed in this file, execution of it via uux is denied. Usually, L.cmds contains one command : rmail.

The L.cmds on most systems contain the following entries:

```
rmail
/usr/lib/uucp/uucico
```

This setup indicates that both the rmail and uucico commands can be executed by uux. Be careful when adding commands to this file. Even innocuous commands such as cat can be dangerous to your system.

SQFILE

Finally, SQFILE is used to track conversations that have taken place between machines. This is an optional file, and if you want to use conversation counts, you must create it in /usr/lib/uucp. SQFILE must be owned by uucp, and have a file mode of 400. For this to work, SQFILE has an entry in it for each file that your system wants to have conversation checks with. The remote system must also be configured to use SQFILE.

When the file is created, edit it to include the names of the files you want to monitor, one system per line. After the first call, uucico adds the number of conversations, and the date and time of the last contact.

When one system calls another, uucico compares the SQFILE information on the two systems. If they don't agree, the login fails. The log files on the calling system will then add a message indicating an SEQ number problem. To correct this, the two system administrators must get together and correct the files manually.

Log Files

The log files on V2 are quite different from HDB log files. Unlike HDB, all of the log entries are placed into a single file in /usr/spool/uucp, appropriately named LOGFILE. A second file exists called SYSLOG, which records the actual amount of data transferred and the time it took to do it. The LOGFILE will grow continually. If you are running short of disk space, this is the first place to check.

An entry from LOGFILE looks like this:

```
user system  date/time   comment

root unilabs (2/12-5:42) NO (AVAILABLE DEVICE)
root unilabs (2/12-5:42) FAILED (call to unilabs )
root unilabs (2/12-5:59) QUEUED (C.unilabsn0297)
root unilabs (2/12-5:59) QUEUED (C.unilabsn0298)
root unilabs (2/12-5:59) SUCCEEDED (call to unilabs )
root unilabs (2/12-5:59) HANDSHAKE FAILED (LOGIN)
unilabs unilabs (2/12-18:35) OK (startup)
```

In the next few entries, you can see that the files /tmp/spool and /tmp/sys were sent to unilabs. Files that are sent show in a REQUEST entry with an S followed by the name of the file.

```
root unilabs (2/12-18:35) REQUEST (S /tmp/spool ~ root)
root unilabs (2/12-18:35) REQUEST (SUCCEEDED)
root unilabs (2/12-18:35) REQUEST (S /tmp/sys ~ root)
root unilabs (2/12-18:35) REQUEST (SUCCEEDED)
root unilabs (2/12-18:35) OK (conversation complete)
```

These log entries don't contain as much information as the log files in HDB, but fortunately you have a second log file, SYSLOG, that can be examined for other important information.

The SYSLOG file contains information of the actual transfer. The first few examples shown here indicate that this machine received data from the remote machine, unilabs.

```
user  system  date/time     secs        comments

chare unilabs (11/21-22:56) (722404580) received data 148 bytes 2 secs
chare unilabs (11/21-22:56) (722404593) received data 1197 bytes 6 secs
chare unilabs (11/21-22:56) (722404601) received data 148 bytes 1 secs
```

These entries relate to the two files you saw transferred in the LOGFILE, namely /tmp/sys and /tmp/spool. These two files were sent from bugs to unilabs.

```
root unilabs (2/12-18:35) (729560123) sent data 97 bytes 0 secs
root unilabs (2/12-18:35) (729560125) sent data 115 bytes 0 secs
```

It takes time to process and review logfile information. Consequently, an understanding of this information is essential to the system troubleshooter.

Maintenance

Version 2 maintenance is simplified somewhat through the use of the uuclean command, which operates a lot like HDB. The uuclean command cleans up the UUCP spool directory (/usr/spool/uucp, typically) somewhat intelligently. For systems that cannot be reached, a mail

message is sent back to the originator. uuclean works by deleting locally created rnews files, executing remotely created rnews files, and removing everything that shouldn't be there.

The periodic removal of logfiles should also be performed to eliminate redundant UUCP log information and free up disk space. However, the original Version 2 UUCP implementation cannot perform this task automatically.

Configuring UUCP over TCP/IP

Although some feel that using the UUCP protocol over the TCP/IP transport is redundant, it can be useful at times. If, for example, you have been using a Usenet news feed over UUCP, you can switch to a TCP/IP transport until you are ready to implement INN or some other TCP/IP-based news server software using your existing implementation. UUCP use over TCP/IP is restricted to the Basic Networking Utilities, which are also known as HoneyDanBer UUCP.

Two files need to be changed before TCP/IP can be used as the transport: the Systems and Devices files. Although the UUCP g protocol can be used, the UUCP e protocol is optimized for the transport characteristics of TCP/IP. The desired protocol type is included with the device identification. Because the UUCP g protocol has extensive error-checking built into it, it is considered to be a waste of resources when used with a high-speed connection such as TCP/IP. In this case, protocol e is often used. To define the protocol, follow the dataport with a comma and the protocol to use, as shown in the following example:

```
TCPnet        TCP,e    ...
```

Aside from direct serial and modem connections, UUCP also supports connections over other transports, such as TCP/IP and the Streams-based TLIS connections. Connections using the TLIS are not directly supported by all vendors of Unix.

TLIS connections are configured in the Devices file. It is possible to configure TLIS to make a connection with or without the Network Listener Service. TCP/IP connections do not use this service. To establish a direct connection to another machine using the TLIS, but not the Network Listener, the device entry would be as follows:

```
STARLAN,eg starlan - - TLIS \D
```

This would define an entry to access the STARLAN network, and allow both the 'e' and 'g' protocols to be used, depending on what was determined at connect time. The device being used is called starlan, with no dialer-port or speed associated with the device. The dialer-token is set to TLIS, and the \D instructs uucico to use the phone number, which is defined in the Systems file.

The TCP/IP entry for the device file looks quite similar to the TLIS network device:

```
TCP    TCP,e    -    Any    TCP    540
```

This entry defines a device called TCP that uses the UUCP e protocol. This TCP keyword is known to UUCP systems that support TCP/IP connections. Notice that the protocol in use here is protocol e, which is the best choice when using end-to-end error free connections. There is no dialer port, and the speed of the connection in this example is Any. The dialer used is TCP, and the TCP/IP port number is defined as 540 for connecting to the remote machine.

To use either the TLIS or the TCP transports, the administrator of the other system must have previously configured his or her system to allow a connection in this manner.

Although it works, the configuration and use of the TCP/IP transport is not common for UUCP. It is more common for UUCP to be completely removed when TCP/IP is placed into operation.

Code Listings

The following two programs can assist you in analyzing log files and building a secure USERFILE for use with Version 2 UUCP.

Listing 3.1—gtimes.c

The gtimes program takes a Unix clock value and converts it to a human readable date. The UUCP log files use this Unix clock value to save disk space. To make sense of the log, however, you must have the real date.

To use gtimes, compile it using your system's C compiler. This can be easily done by using the following command:

```
make gtimes
```

To use the command, execute gtimes with the clock value as seen in the following examples.

```
nms% gtimes 100394857
Clock : 100394857
Date  : Wed Mar  7 18:27:37 1973

nms% gtimes 809283745
Clock : 809283745
Date  : Thu Aug 24 13:02:25 1995
/* ---------------------------------------------------------------

NAME
```

```
   gtimes.c - Calculate clock times.

SYNOPSIS

   gtimes [ clock value ]
    - where clock value is a long integer that was previously returned
      by the time system call.

DESCRIPTION

   This program without an argument will report the current system
   time both as an ASCII string, and as a long integer that is
   directly reported from time(S).
   Invocation with an argument results in the ASCII time string that
   matches the clock value on the command line.

RETURN VALUE

   Always returns 0.

WARNINGS

   There are no provisions for bad data, or overflow.

 ------------------------------------------------------------------ */
/* Copyright 1988 Chris Hare */

#include <stdio.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <time.h>
#include <errno.h>

main( argc, argv )
int argc;
char *argv[];

   {

   char *ctime(),        /* declare ctime(S) */
    *timestr;        /* storage area for time string */
   long int t_secs,        /* return value from time(S) */
        o_secs,        /* long integer value of command argument */
        atol(),        /* declare atol(S) */
        time();        /* declare time(S) */
    struct tm *mytime;
    struct tm *localtime();
    char *atime_str;
```

```
if ( argc == 1 )
   t_secs = time(0L);
else
   t_secs = atol(argv[1]);

timestr = ctime(&t_secs);

printf( "Clock : %ld\nDate  : %s\n", t_secs, timestr );

exit(0);

}
```

Listing 3.2—genUSER

The genUSER program creates a standard USERFILE for the users on your system. It reads the /etc/passwd file and creates a USERFILE entry for each password file entry.

To use genUSER, execute it and the results will be written to a file called USERFILE in your current directory.

A sample USERFILE generated is shown here:

```
noc,    /usr/spool/uucppublic /home/noc
ansnoc, /usr/spool/uucppublic /home/noc
danc,   /usr/spool/uucppublic /home/danc
briand, /usr/spool/uucppublic /home/briand
```

```
#
# @(#) genUSER - generate a USERFILE from /etc/passwd
# CHris Hare, 1993
#
# This script will process /etc/passwd, and create a default USERFILE for
# use with Version 2 UUCP.
#
PASSWD=/etc/passwd            # Location of /wtc/passwd
USERFILE=./USERFILE            # Location of USERFILE
OLD_IFS="$IFS"                # Save current Field Separators
IFS=":"                     # Set new field separator
#
# Process the entire passwd file
#
exec < /etc/passwd
#
# Read each entry
#
while read USERNAME PWORD UID GID COMMENT HOME SHELL
do
```

```
   #
   # write each entyry consisting of
   #    USERNAME,
   #    /usr/spool/uucppublic
   #    HOME directory
   echo "${USERNAME},     /usr/spool/uucppublic $HOME" >> $USERFILE
done
#
# exit ... you are finished
#
exit 0
```

Audit Trails

The National Computer Security Center in Fort Meade, Maryland, defines an audit trail in its Rainbow series of security publications as follows:

"A chronological record of system activities that is sufficient to enable the reconstruction, reviewing, and examination of the sequence of environments and activities surrounding or leading to an operation, a procedure, or an event in a transaction from its inception to final results."

In layman's terms, audit trails *are any files that record the time users log in, from where they log in, what they try to do, and any other action an administrator might want to save for later analysis.*

When used intelligently, audit trails can provide system administrators valuable information in tracking security violations and break-in attempts.

Audit Trails under Unix

Unix is by far the most prevalent operating system in use on the Internet. Luckily for administrators, Unix provides a large number of auditing and logging tools and utilities. Many of these logs are generated automatically by utilities that are part of the default configuration of every Unix machine. Other logging utilities must be turned on and configured by the administrator.

Common Unix Logs

The Unix operating system stores most of its logging in ASCII text files, through which you can sort easily with normal text-editing utilities. Some logs, however, are stored in various binary formats and require specialized utilities for their contents to be viewed.

lastlog

The lastlog file keeps track of each user's most recent login time and each user's originating destination. When a user logs in to a Unix system, the login program looks for the user's UID in the lastlog file. If the program finds it, Unix displays the time and TTY of the user's last login. Some versions of Unix display successful logins as well as unsuccessful login attempts.

```
BSDI BSD/386 1.1 unixbox (ttyp5)
login: phrack
Password:
Last login: Sun Apr 2 16:35:49 from phrack.com
```

The login program then updates the lastlog file with the new login time and TTY information. Further, the program updates the UTMP and WTMP files.

UTMP

The Unix operating system keeps track of users currently logged in to the system with a file called the UTMP. This file is constantly changing as users enter and leave the system. It does not keep a long historical tally of users who have been on the system; it only keeps track of those online at the exact moment.

UTMP might not contain entirely accurate information. Sporadic errors can cause a user's shell to terminate without UTMP having been updated. UTMP is also not particularly reliable because it comes world-writable by default on many Unix platforms.

The normal user's ability to modify this file makes it very easy for an intruder to hide from view.

The UTMP log is usually stored in the file /etc/utmp, although you might find it in other locations on some Unix versions. UTMP is usually viewed with commands such as who or w, but you can also access the file through other commands, such as finger, rwho, and users.

The following is sample output from the who command:

```
freeside % who
phrack   ttyp3   Apr  2 16:35   (phrack)
user     ttyp4   Apr  2 17:12   (fakehost.com)
slip1    ttya0   Apr  2 17:13
ppp1     ttya1   Apr  2 16:29
ccr      ttya6   Apr  2 16:35
ppp2     ttyb2   Apr  2 13:48
freeside %
```

WTMP

The WTMP file keeps track of logins and logouts. It is similar to the UTMP file but continually grows in length with each login or logout. In some Unix versions, programs such as ftp record access information in WTMP as well. WTMP also records the times of normal system shutdowns, such as those caused by the reboot or shutdown commands. Unix generally stores WTMP in the file /var/adm/wtmp.

The WTMP file is normally accessed by the last command. Unix displays output generated by the last command in reverse order—the most recent information appears first, followed by older entries. The last command also can generate reports based on name, TTY or event (such as shutdown); or print only a specified number of entries.

```
freeside % last -10
slip1     ttya0                    Sun Apr  2 17:13   still logged in
user      ttyp4    fakehost.com    Sun Apr  2 17:12   still logged in
Uaust     ttya0                    Sun Apr  2 17:10 - 17:11  (00:00)
user2     ftp      college.edu     Sun Apr  2 17:07 - 17:11  (00:03)
slip1     ttya3                    Sun Apr  2 16:50 - 16:53  (00:03)
slip2     ttyb5                    Sun Apr  2 16:46 - 16:48  (00:01)
aco       ttya5                    Sun Apr  2 16:45 - 17:09  (00:24)
dzz       ttyp4    slip00          Sun Apr  2 16:45 - 16:47  (00:02)
ppp2      ttya3                    Sun Apr  2 16:44 - 16:45  (00:00)
dzz       ftp      slip00          Sun Apr  2 16:43 - 16:48  (00:04)
freeside %
```

Another command, ac, formats the data stored in the WTMP file in a different way. It can generate its reports either by person (ac -p) or by day (ac -d). These reports might quickly alert the administrator to improper usage. An inactive account that suddenly starts logging numerous hours of connect time, for example, is easily spotted in an ac report.

```
freeside % ac -p
ftp     573.56
foo     898.05
spot    125.62
rickm    39.24
faust    27.21
test      4.02
jj      178.77
```

```
cma      10.97
gre      10.73
erikb    12.39
sp        0.18
total  1880.73
```

The ac report can also be sorted by user and date combined. If the administrator feels, for example, that the utilization of 898.05 connect hours for the foo account looks out of place, that administrator can run a more specific ac report:

```
freeside % ac -dp foo
Apr 1 total    10.30
Apr 2 total    12.50
Apr 3 total     8.20
Apr 4 total   815.04
Apr 5 total    12.01
```

The April 4 system usage is obviously out of character for the foo account. Logs, unfortunately, aren't usually this easy to read. With the growing use of multiple login instances through applications such as X-windows, a single user can easily record several hundred hours worth of connect time in just a few days.

syslog

syslog is an extremely useful message-logging facility. Originally developed for BSD-based Unix as a companion to sendmail, it is now included with almost every Unix variant. syslog makes it easier for administrators to keep track of logs generated by a variety of programs by providing a central reference point for the destination of log entries.

To utilize syslog, a daemon called syslogd is executed at startup and runs in the background. This daemon listens for log messages from three sources:

- **/dev/log.** A Unix domain socket that receives messages generated by processes running on the local machine.
- **/dev/klog.** A device that receives messages from the Unix kernel.
- **port 514.** An Internet domain socket that receives syslog messages generated by other machines through UDP.

When syslogd receives a message from any of these sources, it checks its configuration file—syslog.conf—for the appropriate destination of the message. A message can go to multiple destinations, or it might be ignored, depending on the corresponding entries in the configuration file.

Entries in the syslog.conf file are comprised of two basic parts:

- **Selector field.** Tells syslog what kind of messages to log (see tables 4.1 and 4.2).
- **Action field.** Tells syslog what to do with the message it receives.

The selector field is comprised of a program that is sending the log message, often called the facility, and the severity level of the message. Table 4.1 shows syslog facilities.

Table 4.1
syslog Facilities

Facility Name	Originating Program
kern	The kernel
user	User processes
mail	The mail system
auth	Programs using security (login, su, and so forth)
lpr	The line printer system
daemon	System daemons
news	The news system
uucp	UUCP
cron	The cron daemon
mark	Regularly generated timestamps
local0-7	Locally generated messages
syslog	Syslogd messages
authpriv	Other authorization messages
ftp	ftpd messages
*	All facilities (except mark)

Table 4.2 shows syslog severity levels.

Table 4.2
syslog Severity Levels

Severity Level	Meaning
emerg	Emergency situations, such as system crashes potential
alert	Urgent conditions that need correction immediately
crit	Critical conditions

continues

Table 4.2, Continued
syslog Severity Levels

Severity Level	Meaning
err	Ordinary errors
warning	Warning messages
notice	Non-error related that might need special attention
info	Informational messages
debug	Messages used when debugging

The following text is an example syslog.conf file:

```
#
# syslog configuration file.
#
*.err;kern.debug;auth.notice;                    /dev/console
*.err;kern.debug;daemon.info;auth.notice;        /var/adm/messages
mail.crit;daemon.info;                           /var/adm/messages
lpr.debug                                        /var/adm/lpd-errs
*.alert;kern.err;daemon.err;                     operator
*.alert;                                         root
*.emerg;                                         *
auth.notice                                      @logginghost.com
```

In the above example, all emergency messages (*.emerg) go to all users on the system (*). All regular errors (*.err), kernel debugging messages (kern.debug), and authorization failures—such as illegal logins (auth.notice)—are reported to the system console as well as to the file /var/adm/messages. Authorization failures are also sent to a separate host (@logginghost.com) over the network, where they are picked up by that machine's syslog program listening to port 514.

syslog passes most messages to the /var/adm/messages file. In most default configurations, nearly all messages are passed to this file.

sulog

The switch user command, su, also records its usage through syslog. This information is often also stored in a file called sulog, in the /var/adm directory. Some intruders might use the su command to switch to usernames that have rlogin access to other hosts. This activity is reported in the sulog.

Many sites are now using the sudo command rather than su. By using sudo, properly authorized users can execute commands as another user without having to actually log in with the

password of that user (as they would using su). sudo also logs its usage through the syslog facility. sudo logs the command executed, the username of the requester, the time the command was executed, and the directory from which the command was invoked.

A log entry of an authorized user, fred, using sudo to edit the /etc/group file would look something like the following:

```
Apr 2 06:45:22 hostname sudo: fred: PWD=/usr/fred; COMMAND=/bin/vi /etc/group
```

aculog

When a user employs dial-out facilities, such as tip or cu, a log entry is made to a file called aculog. This file is most often stored as /var/adm/aculog. This log contains a record of the user name, time and date, phone number dialed, and completion status of the call. UUCP-related commands also record their information in the aculog file.

A typical aculog entry looks like the following:

```
uucp:daemon (Mon Apr 3 12:31:03 1995) <host, 5551212, usr> call completed
```

Checking aculog entries would be valuable if an intruder were using the Unix host as a conduit, when dialing out to other systems, as a means of avoiding long-distance charges, or avoiding a direct telephone trace.

cron

The cron utility maintains records of the utilities that are executed by cron. Usually this utility is in the file /var/log/cron, but because cron versions now make use of syslog, the messages could be stored in a variety of files.

If intruders can manipulate the crontab file or programs or scripts named in that file, they might be able to use the cron utility to gain higher privileges. Logs generated by cron offer clues to such improper usage.

sendmail Logs

The sendmail program performs its logging with syslog. Messages generated by sendmail are labeled with the facility “mail” and the severity levels “debug” through “crit,” depending on the severity of the message generated. All messages generated by the program include the sendmail program name within the message text.

sendmail has a command-line option (-L), which specifies the lowest severity level that will cause it to log. Higher values of the -L option cause more information to be logged. An -L value of 0 means no logging will occur.

sendmail logs provide important clues to the administrator when intruders are attempting to exploit bugs from the SMTP port.

UUCP Logs

The UUCP utilities store information in various log files, depending on the version of UUCP being used. On BSD-based Unix platforms, a file called LOGFILE contains information regarding UUCP usage. This file is updated both by local UUCP activity and by actions initiated by remote sites. Information in this file consists of calls attempted or received, requests attempted, by whom, at what time, and from what host.

The UUCP log file syslog (not to be confused with the message-handling utility) contains information regarding file transfer statistics. The file shows the byte count of each UUCP transaction, the username and site requesting the file, the time and date of the transaction, and the time needed to complete the transfer.

The ERRLOG file contains any errors that occur during UUCP operation.

Today, most intruders don't utilize UUCP in their activities, because many hosts either don't use it, or don't have it installed. If UUCP is in use, however, logs should be audited for suspicious activity, because files can be compromised from a remote site using UUCP.

LPD Logs

The lpd-errs file represents one of the most common logs dealing with printers. This file is usually designated as /var/adm/lpd-errs in the syslog.conf file. In most instances, the information this file has to offer is not of any use in tracking security incidents. Given the recent discovery of lpd-related bugs, however, any number of odd errors might turn up as a result of an intruder attempting to exploit a bug. Further, any entries that occur on systems that don't even use the line printer daemon are certainly worth investigating.

The following is sample data from an lpd-errs file:

```
Feb 19 17:14:31 host1 lpd[208]: lp0: output filter died (26)
Feb 19 17:14:31 host1 lpd[208]: restarting lp0
Feb 19 17:17:08 host1 lpd[311]: lp0: output filter died (0)
Feb 19 17:17:08 host1 lpd[311]: restarting lp0
Feb 19 17:31:48 host1 lpd[524]: lp0: unknown mode -cs
Feb 19 17:33:12 host1 lpd[523]: exiting
Feb 19 17:33:24 host1 lpd[541]: lp0: unknown mode -cs8
Feb 19 17:34:02 host1 lpd[540]: exiting
```

ftp Logs

Most current versions of the ftp daemon, ftpd, can be set to log incoming connections. ftpd uses syslog to handle the messages it generates.

Logging is activated by executing ftpd with the -l option. The line that invokes ftpd in the inetd.conf file should read as follows:

```
ftp   stream  tcp  nowait  root   /etc/ftpd   ftpd -l
```

The syslog.conf should also be edited to add the following:

```
daemon.info          ftplogfile
```

HTTPD Logs

With the emergence of the World Wide Web as one of the dominating Internet services, almost every domain has set up a WWW server to advertise, inform, and entertain Internet users. HTTPD servers can log every Web access and also report errors generated during normal operation. Many administrators keep these logs to generate demographic usage reports—what hosts access the server most often, what pages are the most popular, and so on.

Two separate files are typically generated—one containing errors and the other containing the accesses. The filenames for these log files are set in the httpd.conf file.

History Logs

One of the most overlooked logs kept under Unix is the shell history log. This file keeps a record of recent commands entered by the user. Both the C shell and the Korn shell support the command history feature.

An environment variable determines the number of command lines retained. Under the C shell, the variable is $history; under the Korn shell, the variable is $HISTSIZE. The commands are stored in a file under the user's home directory. Under the C shell, the file is called .history. Under the Korn shell, the file is called .sh_history by default but can be changed with the $HISTFILE environment variable.

The history command displays the contents of the history logs in chronological order, with preceding numbers. Using the history command with an -h option causes the contents to be displayed without the preceding numbers.

Many intruders forget to erase their shell histories upon initial access to a new system. Even after they edit other logs and replace utilities, every command they have entered remains clearly visible in the history file of the account with which they initially gained access.

Process Accounting

In the past, process accounting was an important part of computing resources. When users were billed solely on the actual amount of CPU time they used, computer centers could not have functioned properly without mechanisms in place that kept track of each command entered.

Today, many systems do not use process accounting; most Unix platforms disable it by default. When it is enabled, however, the process accounting logs often can help administrators locate any intruders that might have gained access to the system.

Enabling Process Accounting

Process accounting is turned on at startup by using the accton command in the following format: accton logfilename. The log file is usually

```
/var/adm/acct or /var/adm/pacct
```

Note Executing accton without a file-name parameter turns off process accounting.

Because process accounting immediately begins recording a great deal of information when enabled, the administrator must make sure that plenty of free disk space is available on the file system storing the process accounting log.

Many administrators review and purge their process accounting logs regularly because of the rapid speed with which the accounting file grows. Some even have cron jobs configured to handle rotating the process accounting files.

Generating Reports

The lastcomm command supplies information on all commands executed on the system. It formats its output to show the command executed, the user who executed the command, what tty that user was using, the time to complete execution, and the time and date the command was executed.

The following output is a small portion of lastcomm data. Because process accounting stores every command executed by every user, normal output could continue scrolling for several minutes.

```
freeside % lastcomm
whoami     F     root     ttyp5      0.01 secs Sun Apr  2 17:17
sh         F     user     ttyp4      0.00 secs Sun Apr  2 17:16
rm         F     user     ttyp4      0.02 secs Sun Apr  2 17:16
sendmail   F     user     ttyp4      0.00 secs Sun Apr  2 17:16
sendmail   F     phrack   ttyp4      0.34 secs Sun Apr  2 17:16
sh         F     user     ttyp4      0.03 secs Sun Apr  2 17:16
sh         F     user     ttyp4      0.00 secs Sun Apr  2 17:16
sh         F     phrack   ttyp5      0.02 secs Sun Apr  2 17:16
more       F     phrack   ttyp5      0.05 secs Sun Apr  2 17:16
lastcomm   FX    phrack   ttyp5      0.23 secs Sun Apr  2 17:16
sendmail   F     user     ttyp4      0.20 secs Sun Apr  2 17:16
sh         F     user     ttyp4      0.02 secs Sun Apr  2 17:16
rm         F     user     ttyp4      0.02 secs Sun Apr  2 17:16
sendmail   F     user     ttyp4      0.31 secs Sun Apr  2 17:16
sendmail   F     user     ttyp4      0.00 secs Sun Apr  2 17:16
sh         F     user     ttyp4      0.02 secs Sun Apr  2 17:16
sh         F     user     ttyp4      0.02 secs Sun Apr  2 17:16
httpd      SF    www      __         0.05 secs Sun Apr  2 17:16
pico       F     ccr      ttya6      0.05 secs Sun Apr  2 17:15
```

Careful examination of the preceding sample output reveals possible intruder activity. During the two-minute span shown in the sample, several users—root, www, ccr, user, and phrack—are running commands. Look closely at the output; the root command entry occurred at the same time and on the same tty as the phrack account. Because the phrack account did not execute an su or sudo command, more than likely the user of that account did something improper to become root. The fact that sendmail was the last command executed by the phrack account before this discrepancy indicates that the user might have exploited some kind of sendmail-based bug.

The sa command offers another useful command for generating reports from the process accounting logs. This command generates output based on CPU time consumed either by users (sa -m) or by commands (sa -s). The sa command helps administrators locate the source of users or commands that are allocating too many system resources.

```
freeside % sa —m
root      73271    500.85cpu   22747961tio    112295694k*sec
daemon     1668      5.45cpu     817411tio       353179k*sec
sys        4239     20.79cpu    4840469tio       411555k*sec
gopherd      66      0.77cpu      17194tio        94396k*sec
www       30935    119.68cpu    2674466tio      4345219k*sec
bobs          8      0.23cpu      52076tio        60909k*sec
erikb       447      2.43cpu     386568tio       389052k*sec
rickm      5325    111.08cpu    7892131tio     -4722301k*sec
cma         121      0.78cpu     149312tio       111471k*sec
faust      1349     11.47cpu    1355051tio      2629341k*sec
jj          489      6.37cpu    1069151tio      1231814k*sec
gre           4      0.11cpu      98032tio        13844k*sec
foo       14574     87.25cpu     432077tio      4170422k*sec
sqr       46641    877.97cpu   63720573tio    243279830k*sec
nobody      209      4.69cpu     321321tio      1601114k*sec
```

Useful Utilities in Auditing

Several other utilities can greatly help the system administrator conduct audits. Although these utilities might not make use of specific log files to generate their information, you can collect output from these utilities and use them in conjunction with other logs to create a much clearer picture of the true state of the system.

ps

Individual users often have better luck with the ps command than they do with commands such as who or users when tracking system utilization. The ps command displays the following information: current process IDs, the associated TTY, owner of the process, execution time, and the actual command being executed. Because ps draws from the kernel's process table in generating its report, ps output cannot be altered by simply editing a log file.

The ps command is useful for locating background processes left running by intruders, for locating user processes running on active TTYs for which no UTMP entries exist, and for tracking all activity of given users.

The following sample is a portion of ps output from a BSDI Unix machine:

```
freeside % ps -uax
USER     PID  %CPU %MEM    VSZ    RSS   TT  STAT   STARTED      TIME   COMMAND
root      73   2.1  0.0   1372   1004   ??     S   24Mar95  84:38.60   gated
root       0   0.0  0.0      0      0   ??   DLs   24Mar95   0:00.38   (swapper)
root       1   0.0  0.0    244    116   ??    Is   24Mar95   3:21.42   init —
root       2   0.0  0.0      0     12   ??    DL   24Mar95   0:03.42   (pagedaemon)
root      35   0.0  0.0    208    144   ??    Ss   24Mar95   4:50.63   syslogd
root      68   0.0  0.0     72     28   ??    Ss   24Mar95  29:03.42   update
root      70   0.0  0.0    280    160   ??    Is   24Mar95   0:17.43   cron
root      76   0.0  0.0  10660  10612   ??    Ss   24Mar95  46:10.08   named
root      80   0.0  0.0    236     52   ??   IWs   24Mar95   0:00.10   lpd
root      83   0.0  0.0    172     96   ??    Is   24Mar95   0:00.08   portmap
root      88   0.0  0.0    244    180   ??    Is   24Mar95   0:00.13   mountd
root      90   0.0  0.0    140     16   ??   IWs   24Mar95   0:00.02   (nfsd)
root      99   0.0  0.0    100     16   ??     I   24Mar95   0:00.22   nfsiod 4
root     104   0.0  0.0    216    112   ??    Ss   24Mar95   1:46.96   inetd
root    2106   0.0  0.0    240    172   p0-    I   25Mar95   1:18.26   freeside
root    5747   0.0  0.0    520    220   ??    Is   Wed12PM   2:07.20   (sendmail)
phrack 14289   0.0  0.0    240    176   b1-    I   Wed06PM   0:00.15   archie
phrack 22626   0.0  0.0    752    712   p4    Ss+  12:30PM   0:42.35   irc
phrack 26785   0.6  0.0    584    464   p4     Ss  11:57PM   0:00.40   -tcsh
phrack 26793   0.0  0.0    320    224   p4     R+  11:57PM   0:00.06   ps -uax
freeside %
```

The preceding example shows several root processes running as background processes. The sample also shows several current processes running that the phrack account owns. One process in particular—14289—might warrant a closer look. It appears to be an archie request that has been running for longer than normal, and is on a different TTY than the phrack account is currently logged in on. This discrepancy could be the result of a process that did not exit properly, but it also could be a malicious utility running in the background, a utility compiled with an inconspicuous name to avoid suspicion.

netstat

The netstat command displays useful information regarding the state of the TCP/IP network traffic running to and from the host computer. In some instances, netstat is the only monitoring tool the administrator has to locate intruders.

In the active connections portion of netstat output a list of addresses corresponding to open incoming connections is given. Even if an intruder has removed himself from UTMP or other logs, his incoming connection might still be visible through netstat.

The following sample is a portion of netstat output on a BSDI Unix machine:

```
freeside% netstat
Active Internet connections
Proto   Recv-Q   Send-Q    Local Address    Foreign Address         (state)
tcp          0        0    freeside.1581    bbs.sdd8.nanaimo.smtp   ESTABLISHED
tcp          0        0    freeside.1580    avarice.mrrr.lut.smtp   ESTABLISHED
tcp          0        0    freeside.http    slip09.1125             TIME_WAIT
tcp          0        6    freeside.1579    tibal.supernet..smtp    ESTABLISHED
tcp          0        0    freeside.http    slip0.1124              TIME_WAIT
tcp          0        0    freeside.http    slip0.1123              TIME_WAIT
tcp          0        0    freeside.http    slip0.1122              TIME_WAIT
tcp          0        0    freeside.1576    vangogh.rtppc.ep.smtp   TIME_WAIT
tcp          0        0    freeside.http    slip0.1121              TIME_WAIT
tcp          0        0    freeside.http    slip0.1120              TIME_WAIT
tcp          0      468    freeside.telnet  phrack.1032             ESTABLISHED
tcp          0        0    freeside.1572    vulcan.cblink.co.smtp   TIME_WAIT
tcp          0        0    freeside.1568    dewey.cs.texas.e.6667   ESTABLISHED
tcp          0        0    freeside.1493    zilla.nntp              ESTABLISHED
tcp          0        0    freeside.4897    yod.texas.net.6667      ESTABLISHED
tcp          0     4096    freeside.http    cicaa2-5.dial.1246      LAST_ACK
tcp          0     3584    freeside.http    cicaa2-5.dial.1245      LAST_ACK
tcp          0     1627    freeside.http    cicaa2-5.dial.1241      LAST_ACK
tcp          0     3584    freeside.http    cicaa2-5.dial.1237      LAST_ACK
tcp          0     3584    freeside.http    p.cincinnati.1327       LAST_ACK
tcp          0        1    freeside.telnet  pcnet.utsa.ed.16014     CLOSING
udp          0        0    loopback.domain  *.*
udp          0        0    freeside.domain  *.*
udp          0        0    freeside.1042    raw.2049
udp          0        0    freeside.1039    bull.2049
udp          0        0    freeside.1036    zilla.2049
```

As seen in the preceding output, the full hostname information might not be displayed in the foreign address field due to length restrictions, but it is often more than enough to determine the true address. The local domain, for example, has incoming telnet sessions from pcnet.utsa.edu and from the phrack host. If no users are known from pcnet.utsa.edu, then connections from the host might be a good indicator of possible intruder activity.

Ethernet Sniffers

An *ethernet sniffer* is a program that logs all activity over the local ethernet segment. Some Unix versions might include sniffing utilities, like tcpdump or snoop, but utilities such as these are available on the Internet as well.

Ethernet sniffer programs are priceless for debugging network problems such as broadcast storms, or for locating the source of problem output; but in the wrong hands they can be deadly. Because the purpose of the program is to intercept and view (or log) all packets on the network, many intruders run these utilities to intercept username and password information as it passes across the network.

Administrators who use these tools should take precautions to ensure that normal users cannot access them. Administrators also might want to check periodically for any indication that an intruder has started his own ethernet sniffer; an administrator can do so by looking to see if any of the machines' ethernet interfaces are running in promiscuous mode.

Other Reporting Tools Available Online

A plethora of monitoring and logging utilities have been written in recent years to help system administrators keep track of potential break-in attempts and other problems.

Many such utilities are available for free in various ftp archive sites on the Internet, and new ones are released continuously.

asax

The advanced security audit trail analyzer on Unix (asax) utility helps system administrators process and analyze data maintained in log files. Sorting through numerous large files of logged data can be extremely tiresome and difficult. asax is designed to remove some of that burden.

asax can be found at the following ftp site:

```
ftp.fc.net
/pub/security/asax-1.0.tar.gz
```

chklastlog and chkwtmp

chklastlog and chkwtmp analyze the lastlog and WTMP files to ensure that no entries have been deleted.

These two utilities can be found at the following ftp site:

```
ftp.fc.net
/pub/security/chklastlog-1.0.tar.gz
/pub/security/chkwtmp-1.0.tar.gz
```

lsof

lsof lists all open files being used by running processes. Based on the files that the process accesses, this utility can clearly illustrate whether a particular process is actually benign or a disguised piece of malicious software.

The lsof utility can be found at the following ftp site:

```
ftp.cert.org
/pub/tools/lsof/lsof_3.02.tar.gz
```

netlog

netlog is an advanced sniffer package containing three utilities:

- **TCPLOGGER.** Logs all TCP connections on a subnet.
- **UDPLOGGER.** Logs all UDP connections on a subnet.
- **EXTRACT.** Processes the logs generated by tcplogger and udplogger.

Administrators at Texas A&M University developed and implemented these programs.

The netlog package can be found at the following ftp site:

```
ftp.fc.net/pub/security/netlog-1.2.tar.gz
```

NFS watch

The NFS watch utility monitors NFS requests to specific machines or to all machines on the local network. Its main function is to monitor NFS client traffic, but it also logs reply traffic from NFS servers to measure traffic statistics, such as response times.

NFS watch can be found at the following ftp site:

```
ftp.fc.net
/pub/security/nfswatch4.1.tar.gz
```

TCP wrapper

Wietse Venema's TCP wrapper utility enables the administrator to easily monitor and filter incoming TCP traffic to network services such as systat, finger, ftp, telnet, rlogin, rsh, talk, and others.

This program can be found at the following ftp site:

```
ftp.cert.org
/pub/tools/tcp_wrappers/tcp_wrappers_7.2.tar
```

tripwire

tripwire is a useful tool that measures all changes to a Unix file system. It keeps a database of inode information and logs of file and directory information based on a user-defined configuration file. Each time it is run, tripwire compares the stored values against flags set in the configuration file. If any deviations from the original value show up, the program alerts the administrator.

tripwire can be found at the following ftp site:

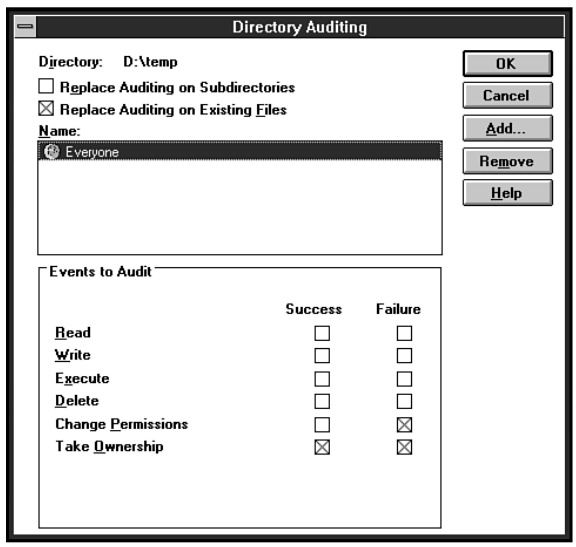
```
ftp.cert.org
/pub/tools/tripwire/tripwire-1.2.tar.Z
```

Audit Trails under Windows NT

Almost every transaction under Windows NT can be audited to some degree. Administrators, therefore, should choose carefully the actions they want to audit so as not to tie up system resources and needlessly fill up disk space.

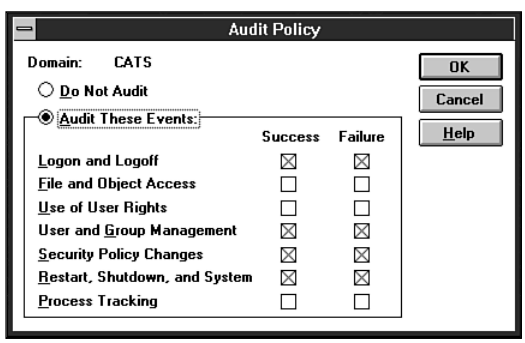
Auditing can be turned on in two places under Windows NT—the File Manager, and the User Manager. Under the File Manager, choose Security and then Auditing to activate the Directory Auditing dialog box (see fig. 4.1). From this window, the administrator can select to track both valid and invalid file accesses.

Figure 4.1
Configuring file-access auditing under Windows NT.



Under the User Manager, the administrator has the option to select audit policy based on the success and failure of several user events, such as login and logout, file access, rights violations, and shutdowns (see fig. 4.2).

Figure 4.2
Setting user audit policy under Windows NT.



Using the Event Viewer

Windows NT stores its log files in a special format that can be read using the Event Viewer application. The Event Viewer is found in the Administrative Tools program group. The Event Viewer's Filter option enables the administrator to select the log entries he wants to view based on criteria such as category, user, and message type (see fig. 4.3).

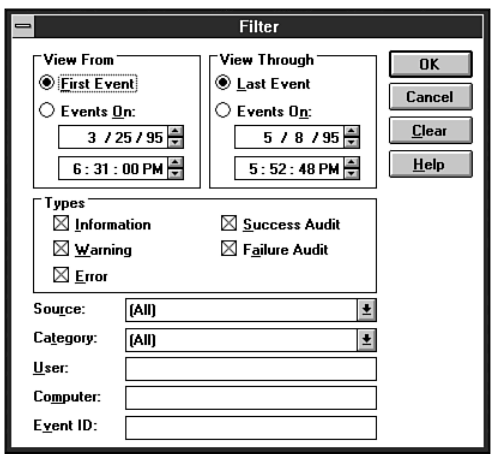


Figure 4.3
Selecting filter criteria under the Event Viewer.

The Event Viewer (see fig. 4.4) differentiates various types of messages by using small icons, each representing one of five distinct types of entries:

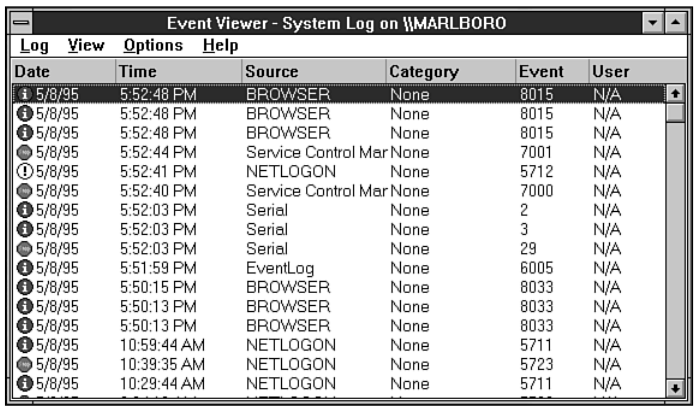
- A red stop sign indicates an error.
- An exclamation point within a yellow circle indicates a warning message.
- The letter I within a blue circle indicates an informational message.
- A gray padlock indicates an invalid authorization message.
- A gold key indicates a successful authorization message.

Windows NT stores auditing information in three separate log files:

- Application Log
- Security Log
- System Log

The Application Log contains information generated by applications registered with the NT Security Authority.

Figure 4.4
One of the three Windows NT auditing logs under NT Event Viewer.



The Security Log contains information about system accesses through NT-recognized security providers and clients. Other events—such as illegal file accesses, invalid password entries, access to certain privileged objects, and account name or password changes—can be tracked as well if the administrator chooses to do so. Individual applications also can assign their own security events, which appear in both the Security Log and the Application Log.

The System Log contains information on all system-related events, some of which might also be in the Security Log, the Applications Log, or both. The System Log acts as a default storage file for much of the regularly generated Windows NT auditing information.

Logging the ftp Server Service

Incoming ftp connections can be logged under Windows NT, but only after changes have been made in the Registry. You can specify whether NT should log connections made by anonymous ftp users, by normal ftp users, or by both. These log entries can be viewed in the System Log by using the Event Viewer.

Warning You can seriously disable Windows NT if you make incorrect changes to the Registry when using the Registry Editor. Unix provides no error warnings when you improperly change values with the Registry Editor. Exercise caution when using this utility.

To enable logging with ftp, perform the following tasks:

1. Run the REGEDIT32.EXE utility.
2. When the Registry Editor window appears, select HKEY_LOCAL_MACHINE on Local Machine, then click on the icons for the SYSTEM subtree until you reach the following subkey:

```
\SYSTEM\CurrentControlSet\Services\ftpsvc\Parameters
```

3. The relevant parameters for enabling logging are LogAnonymous and LogNonAnonymous. The possible values are 0 and 1. The default value is 0, which means do not log. Changing the values to 1 turns on the logging option.

4. Restart the ftp Server service for the changes to take effect.

Logging httpd Transactions

The NT httpd service enables administrators to log access attempts to a specified file. The logging feature can be activated by selecting a check box on the httpd configuration utility, found in the Control Panel (see fig. 4.5). The httpd server adds entries to the Application Log and maintains its own logs in a filename specified during configuration.

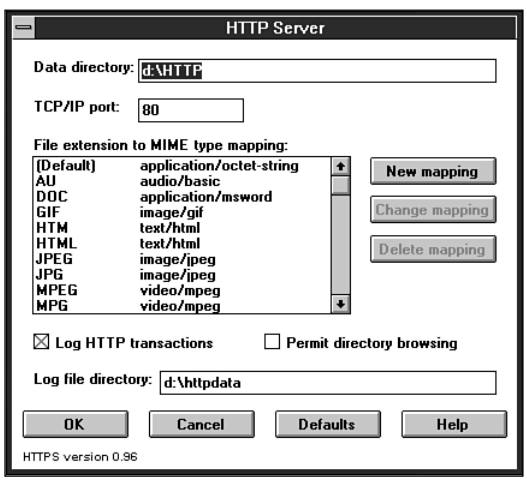


Figure 4.5
The NT HTTPD configuration dialog box.

Logging by Other TCP/IP Applications under NT

Other NT-based applications that utilize the TCP/IP suite of protocols can provide the administrator with valuable auditing information. This section offers an overview of these applications.

SNMP

The Windows SNMP service can provide the administrator with useful traffic statistics from the local network, from the server itself, and from applications that use TCP/IP.

The application also can be configured to accept SNMP information from only certain IP addresses and to send traps for failed SNMP authentications. Only the administrator can configure SNMP options.

SQL Server

The SQL Server for NT automatically logs its transaction requests in the Application Log.

Systems Management Server

The Systems Management Server (SMS) product contains an application called Network Monitor that allows the administrator to monitor all TCP/IP traffic. Network Monitor is an ethernet sniffer program similar to Novell's Lanalyzer product. You can configure it to record data based on protocol type, source address, and destination address. This utility can be a valuable tool in monitoring suspicious traffic both to and from the LAN.

Audit Trails under DOS

Because access to the network in many installations comes from DOS-based workstations, administrators might want to begin tracking all usage from the moment end users turn on their PCs. In many cases, however, this tracking might be more work than is desired; maintaining logs on multiple machines requires a great deal of logistical planning on the part of the administrator.

A large number of PC-auditing packages are available on the market. Some are even available as shareware. These programs generally allow for multiple-user logins or multiple-security levels; device control, such as keyboard locking, serial port locking, and screen blanking; boot control; encryption; file access control; and audit trail features.

PC/DACS

Mergent International's PC/DACS product maintains logs pertaining to three types of events:

- Session events
- Operation events
- Violations

The session events logged include logins, logouts, user time-outs, and logins generated after system time-outs.

Operation events tracked include program executions (normal or TSRs); subdirectory creation and deletion; user additions and deletions; changes to access rules; COM and LPT port accesses; and file attributes.

Violations tracked include invalid user ids; invalid passwords; unauthorized attempts to write to COM and LPT ports; and unauthorized file and directory accesses or modification attempts.

PC/DACS enables the administrator to generate standard reports based on system events mentioned previously. It also has the option to extract various audit log information to a text file.

Watchdog

Fisher's Watchdog product audits user command activity; directory accesses; program execution; date and time changes; and illegal and legal login attempts.

Audit trail reports can be displayed to the screen, printed, or saved to a file. The package has a report generator that enables the administrator to predefine multiple-report formats for later use.

LOCK

Secure Computing's LOCK, a shareware product, offers many of the same features as the commercial products. It enables user authentication; access control over files, directories, and ports; encryption; and audit trail features.

LOCK's auditing features enable administrators to track command execution; invalid login or password entries; unauthorized file or directory access; and changes to any settings.

LOCK is available on the Internet at the following ftp site:

```
ftp.fc.net
/pub/security/lock.zip
```

Using System Logs to Discover Intruders

Because daily system upkeep and user support is so overwhelming at times, many administrators cannot undertake all the security-related duties they would like. When a system is properly configured to monitor and log user and network activity, however, discovering intrusion attempts is much easier.

By implementing a combination of logging utilities, such as process accounting and TCP wrappers (along with the regular verification of standard system logs), an administrator can almost certainly detect any suspicious activity.

Common Break-In Indications

The most common indicator of a computer break-in involves improper account usage. An account that has been inactive for months that suddenly starts utilizing large amounts of system time is definitely suspect. An account designated for the secretarial staff that suddenly starts trying to view files owned by an engineering group is another good indication of possible intruder activity.

Some common security bugs used by intruders often leave traces in various system logs. Recent sendmail bugs have caused errors to be generated during the invocation of the bug. These showed up in the sendmail log files and in the postmaster mail spool. Another recent lpd bug has caused errors to be reported to printer error log. Staying informed of security holes exploited by intruders and knowing how their aftereffects can be revealed in system logs is critical for the security-minded administrator.

Regular reviews of TCP wrapper logs often reveal activities indicative of break-in attempts. Incoming TCP-based connections—such as telnet, ftp, and finger—from strange sites might be warning signs of intruder activity.

Potential Problems

Even though system logs provide the administrator with a wealth of information, they by no means offer a complete solution for tracking security violations. System logs are subject to data corruption, modification, and deletion. In many cases, they only generate entries after a break-in has occurred. The practice of reactive security measures, rather than proactive, is not a good idea.

Note On more than one occasion, individuals have generated fake syslog messages to have it appear as if numerous invalid root login attempts had occurred by specific users from foreign sites. In some cases, these acts were done to frame other users; in other cases, they were done to draw attention away from other, more serious breaches of security. The logs were determined to be false by comparing login records at the remote site and determining that the user indicated in the logs was not online at the time the syslog messages indicated.

Today, most computer break-ins involve several steps, as follows:

- Probing available services or in-roads to the system
- Utilizing known bugs or bad password entries to gain access
- Gaining super-user access
- Erasing any indications of the break-in
- Modifying utilities to ensure undetected future access

Compromised System Logs

When intruders gain access to a system, they almost immediately try to remove themselves from view. Most intruders have a wide array of tools to edit lastlog, WTMP, UTMP, and other logs. Such logs are usually modifiable only by root, but a surprisingly large number of systems still have UTMP world-writable.

Depending on how careless the intruder was and the tools used to edit the logs, some indications of the modification might be left visible to the administrator. One common lastlog editor used by the underground community writes null characters over the entry it wants to remove, rather than actually completely removing it. Although it appears as if the entry has been removed when viewed with last, an examination of the log file clearly shows that the entry has been tampered with.

Modified System Utilities

To ensure that they can always get back into a system after they have broken into it the first time, most intruders replace utilities with modified versions that contain backdoor passwords. Along with these back doors, the modified utilities also remove any routines that generate log entries. An intruder might install a modified login program, for example, that allows him super-user access when a certain backdoor password is entered, and grants him shell access without updating UTMP, WTMP, or lastlog.

Because source code for almost all Unix platforms has fallen into the hands of the underground community, it stands to reason that members of that community have the capability to modify every utility that contributes logging information. It doesn't take too much time or skill to search through source code and look for syslog calls or other logging functions.

In some recent cases, the intruders had recompiled the Unix kernel itself with a special set of instructions to use when dealing with specific utilities such as ifconfig, netstat, ls, ps, and login. Not only had these utilities been modified to hide the intruders, but whenever the kernel received instructions to open or execute these files, it was set to report back information that made it look as if they had not been modified. Because the kernel itself was modified to report back false information about itself, administrators would have never found the intruders had they not booted from the distribution CD and mounted their old root file system under a different kernel to do a full investigation.

In most cases, when an administrator feels that a utility has been tampered with, he merely replaces it with an original from a distribution tape or CD. In this case, however, the administrators reinstalled the entire operating system and rebuilt the kernel.

RFC 1244—The Site Security Handbook

The following document is RFC 1244—The Site Security Handbook from the Internet Task Force. This is the original document first written in July 1991 that is commonly looked to as the working administrators' bible. Currently, progress is underway to rewrite the handbook with a proposed RFC publication date of March 1996.

This handbook is the product of the Site Security Policy Handbook Working Group (SSPHWG), a combined effort of the Security Area and User Services Area of the Internet Engineering Task Force (IETF). This FYI RFC provides information for the Internet community. It does not specify an Internet standard. Distribution of this memo is unlimited.

Contributing Authors

The following are the authors of the Site Security Handbook. Without their dedication, this handbook would not have been possible.

Dave Curry (Purdue University), Sean Kirkpatrick (Unisys), Tom Longstaff (LLNL), Greg Hollingsworth (Johns Hopkins University), Jeffrey Carpenter (University of Pittsburgh), Barbara Fraser (CERT), Fred Ostapik (SRI NISC), Allen Sturtevant (LLNL), Dan Long (BBN), Jim Duncan (Pennsylvania State University), and Frank Byrum (DEC).

Editors' Note

This FYI RFC is a first attempt at providing Internet users guidance on how to deal with security issues in the Internet. As such, this document is necessarily incomplete. There are some clear shortfalls; for example, this document focuses mostly on resources available in the United States. In the spirit of the Internet's "Request for Comments" series of notes, we encourage feedback from users of this handbook. In particular, those who utilize this document to craft their own policies and procedures.

This handbook is meant to be a starting place for further research and should be viewed as a useful resource, but not the final authority. Different organizations and jurisdictions will have different resources and rules. Talk to your local organizations, consult an informed lawyer, or consult with local and national law enforcement. These groups can help fill in the gaps that this document cannot hope to cover.

Finally, we intend for this FYI RFC to grow and evolve. Please send comments and suggestions to:

`ssphwg@cert.sei.cmu.edu.`

1. Introduction

1.1 Purpose of this Work

This handbook is a guide to setting computer security policies and procedures for sites that have systems on the Internet. This guide lists issues and factors that a site must consider when setting their own policies. It makes some recommendations and gives discussions of relevant areas.

This guide is only a framework for setting security policies and procedures. In order to have an effective set of policies and procedures, a site will have to make many decisions, gain agreement, and then communicate and implement the policies.

1.2 Audience

The audience for this work are system administrators and decision makers (who are more traditionally called "administrators" or "middle management") at sites. This document is not directed at programmers or those trying to create secure programs or systems. The focus of this document is on the policies and procedures that need to be in place to support any technical security features that a site may be implementing.

The primary audience for this work are sites that are members of the Internet community. However, this document should be useful to any site that allows communication with other sites. As a general guide to security policies, this document may also be useful to sites with isolated systems.

1.3 Definitions

For the purposes of this guide, a site" is any organization that owns computers or network-related resources. These resources may include host computers that users use, routers, terminal servers, PC's or other devices that have access to the Internet. A site may be a end user of Internet services or a service provider such as a regional network. However, most of the focus of this guide is on those end users of Internet services.

We assume that the site has the ability to set policies and procedures for itself with the concurrence and support from those who actually own the resources.

The "Internet" is those set of networks and machines that use the TCP/IP protocol suite, connected through gateways, and sharing a common name and address spaces [1].

The term "system administrator" is used to cover all those who are responsible for the day-to-day operation of resources. This may be a number of individuals or an organization.

The term "decision maker" refers to those people at a site who set or approve policy. These are often (but not always) the people who own the resources.

1.4 Related Work

The IETF Security Policy Working Group (SPWG) is working on a set of recommended security policy guidelines for the Internet [23]. These guidelines may be adopted as policy by regional networks or owners of other resources. This handbook should be a useful tool to help sites implement those policies as desired or required. However, even implementing the proposed policies isn't enough to secure a site. The proposed Internet policies deal only with network access security. It says nothing about how sites should deal with local security issues.

1.5 Scope

This document covers issues about what a computer security policy should contain, what kinds of procedures are need to enforce security, and some recommendations about how to deal with the problem. When developing a security policy, close attention should be made not only on the security needs and requirements of the local network, but also the security needs and requirements of the other interconnected networks.

This is not a cookbook for computer security. Each site has different needs; the security needs of a corporation might well be different than the security needs of an academic institution. Any security plan has to conform to the needs and culture of the site.

This handbook does not cover details of how to do risk assessment, contingency planning, or physical security. These things are essential in setting and implementing effective security policy, but this document leaves treatment of those issues to other documents.

We will try to provide some pointers in that direction.

This document also doesn't talk about how to design or implement secure systems or programs.

1.6 Why Do We Need Security Policies and Procedures?

For most sites, the interest in computer security is proportional to the perception of risk and threats. The world of computers has changed dramatically over the past twenty-five years. Twenty-five years ago, most computers were centralized and managed by data centers. Computers were kept in locked rooms and staffs of people made sure they were carefully managed and physically secured. Links outside a site were unusual.

Computer security threats were rare, and were basically concerned with insiders: authorized users misusing accounts, theft and vandalism, and so forth. These threats were well understood and dealt with using standard techniques: computers behind locked doors, and accounting for all resources.

Computing in the 1990's is radically different. Many systems are in private offices and labs, often managed by individuals or persons employed outside a computer center. Many systems are connected into the Internet, and from there around the world: the United States, Europe, Asia, and Australia are all connected together. Security threats are different today. The time honored advice says "don't write your password down and put it in your desk" lest someone find it. With world-wide Internet connections, someone could get into your system from the other side of the world and steal your password in the middle of the night when your building is locked up.

Viruses and worms can be passed from machine to machine. The Internet allows the electronic equivalent of the thief who looks for open windows and doors; now a person can check hundreds of machines for vulnerabilities in a few hours.

System administrators and decision makers have to understand the security threats that exist, what the risk and cost of a problem would be, and what kind of action they want to take (if any) to prevent and respond to security threats.

As an illustration of some of the issues that need to be dealt with in security problems, consider the following scenarios (thanks to Russell Brand [2, BRAND] for these):

- A system programmer gets a call reporting that a major underground cracker newsletter is being distributed from the administrative machine at his center to five thousand sites in the US and Western Europe.

  Eight weeks later, the authorities call to inform you the information in one of these newsletters was used to disable "911" in a major city for five hours.

- A user calls in to report that he can't login to his account at 3 o'clock in the morning on a Saturday. The system staffer can't login either. After rebooting to single user mode, he finds that password file is empty.

  By Monday morning, your staff determines that a number of privileged file transfers took place between this machine and a local university.

  Tuesday morning a copy of the deleted password file is found on the university machine along with password files for a dozen other machines.

  A week later you find that your system initialization files had been altered in a hostile fashion.

- You receive a call saying that a breakin to a government lab occurred from one of your center's machines. You are requested to provide accounting files to help trackdown the attacker.

  A week later you are given a list of machines at your site that have been broken into.

- A reporter calls up asking about the breakin at your center. You haven't heard of any such breakin. Three days later, you learn that there was a breakin. The center director had his wife's name as a password.

- A change in system binaries is detected. The day that it is corrected, they again are changed. This repeats itself for some weeks.

- If an intruder is found on your system, should you leave the system open to monitor the situation or should you close down the holes and open them up again later?

- If an intruder is using your site, should you call law enforcement? Who makes that decision? If law enforcement asks you to leave your site open, who makes that decision?
- What steps should be taken if another site calls you and says they see activity coming from an account on your system? What if the account is owned by a local manager?

1.7 Basic Approach

Setting security policies and procedures really means developing a plan for how to deal with computer security. One way to approach this task is suggested by Fites, et. al. [3, FITES]:

- Look at what you are trying to protect.
- Look at what you need to protect it from.
- Determine how likely the threats are.
- Implement measures which will protect your assets in a cost-effective manner.
- Review the process continuously, and improve things every time a weakness is found.

This handbook will concentrate mostly on the last two steps, but the first three are critically important to making effective decisions about security. One old truism in security is that the cost of protecting yourself against a threat should be less than the cost recovering if the threat were to strike you. Without reasonable knowledge of what you are protecting and what the likely threats are, following this rule could be difficult.

1.8 Organization of this Document

This document is organized into seven parts in addition to this introduction.

The basic form of each section is to discuss issues that a site might want to consider in creating a computer security policy and setting procedures to implement that policy. In some cases, possible options are discussed along with some of the ramifications of those choices. As far as possible, this document tries not to dictate the choices a site should make, since these depend on local circumstances. Some of the issues brought up may not apply to all sites. Nonetheless, all sites should at least consider the issues brought up here to ensure that they do not miss some important area.

The overall flow of the document is to discuss policy issues followed by the issues that come up in creating procedures to implement the policies.

Section 2 discusses setting official site policies for access to computing resources. It also goes into the issue of what happens when the policy is violated. The policies will drive the procedures that need to be created, so decision makers will need to make choices about policies before many of the procedural issues in following sections can be dealt with. A key part of

creating policies is doing some kind of risk assessment to decide what really needs to be protected and the level of resources that should be applied to protect them.

Once policies are in place, procedures to prevent future security problems should be established. Section 3 defines and suggests actions to take when unauthorized activity is suspected. Resources to prevent secruity breaches are also discussed. Section 4 discusses types of procedures to prevent security problems.

Prevention is a key to security; as an example, the Computer Emergency Response Team/Coordination Center (CERT/CC) at Carnegie-Mellon University (CMU) estimates that 80% or more of the problems they see have to do with poorly chosen passwords.

Section 5 discusses incident handling: what kinds of issues does a site face when someone violates the security policy. Many decisions will have to made on the spot as the incident occurs, but many of the options and issues can be discussed in advance. At very least, responsibilities and methods of communication can be established before an incident. Again, the choices here are influenced by the policies discussed in section 2.

Section 6 deals with what happens after a security violation has been dealt with. Security planning is an on-going cycle; just after an incident has occurred is an excellent opportunity to improve policies and procedures.

The rest of the document provides references and an annotated bibliography.

2. Establishing Official Site Policy on Computer Security

2.1 Brief Overview

2.1.1 Organization Issues

The goal in developing an official site policy on computer security is to define the organization's expectations of proper computer and network use and to define procedures to prevent and respond to security incidents. In order to do this, aspects of the particular organization must be considered.

First, the goals and direction of the organization should be considered. For example, a military base may have very different security concerns from a those of a university.

Second, the site security policy developed must conform to existing policies, rules, regulations and laws that the organization is subject to. Therefore it will be necessary to identify these and take them into consideration while developing the policy.

Third, unless the local network is completely isolated and standalone, it is necessary to consider security implications in a more global context. The policy should address the issues when local security problems develop as a result of a remote site as well as when problems occur on remote systems as a result of a local host or user.

2.1.2 Who Makes the Policy?

Policy creation must be a joint effort by technical personnel, who understand the full ramifications of the proposed policy and the implementation of the policy, and by decision makers who have the power to enforce the policy. A policy which is neither implementable nor enforceable is useless.

Since a computer security policy can affect everyone in an organization, it is worth taking some care to make sure you have the right level of authority on the policy decisions. Though a particular group (such as a campus information services group) may have responsibility for enforcing a policy, an even higher group may have to support and approve the policy.

2.1.3 Who is Involved?

Establishing a site policy has the potential for involving every computer user at the site in a variety of ways. Computer users may be responsible for personal password administration. Systems managers are obligated to fix security holes and to oversee the system.

It is critical to get the right set of people involved at the start of the process. There may already be groups concerned with security who would consider a computer security policy to be their area. Some of the types of groups that might be involved include auditing/control, organizations that deal with physical security, campus information systems groups, and so forth. Asking these types of groups to "buy in" from the start can help facilitate the acceptance of the policy.

2.1.4 Responsibilities

A key element of a computer security policy is making sure everyone knows their own responsibility for maintaining security.

A computer security policy cannot anticipate all possibilities; however, it can ensure that each kind of problem does have someone assigned to deal with it. There may be levels of responsibility associated with a policy on computer security. At one level, each user of a computing resource may have a responsibility to protect his account. A user who allows his account to be compromised increases the chances of compromising other accounts or resources.

System managers may form another responsibility level: they must help to ensure the security of the computer system. Network managers may reside at yet another level.

2.2 Risk Assessment

2.2.1 General Discussion

One of the most important reasons for creating a computer security policy is to ensure that efforts spent on security yield cost effective benefits. Although this may seem obvious, it is possible to be mislead about where the effort is needed. As an example, there is a great deal of publicity about intruders on computers systems; yet most surveys of computer security show that for most organizations, the actual loss from "insiders" is much greater.

Risk analysis involves determining what you need to protect, what you need to protect it from, and how to protect it. Is is the process of examining all of your risks, and ranking those risks by level of severity. This process involves making cost-effective decisions on what you want to protect. The old security adage says that you should not spend more to protect something than it is actually worth.

A full treatment of risk analysis is outside the scope of this document. [3, FITES] and [16, PFLEEGER] provide introductions to this topic. However, there are two elements of a risk analysis that will be briefly covered in the next two sections:

1. Identifying the assets
2. Identifying the threats

For each asset, the basic goals of security are availability, confidentiality, and integrity. Each threat should be examined with an eye to how the threat could affect these areas.

2.2.2 Identifying the Assets

One step in a risk analysis is to identify all the things that need to be protected. Some things are obvious, like all the various pieces of hardware, but some are overlooked, such as the people who actually use the systems. The essential point is to list all things that could be affected by a security problem.

One list of categories is suggested by Pfleeger [16, PFLEEGER, page 459]; this list is adapted from that source:

1. Hardware: cpus, 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.

2.2.3 Identifying the Threats

Once the assets requiring protection are identified, it is necessary to identify threats to those assests. The threats can then be examined to determine what potential for loss exists. It helps to consider from what threats you are trying to protect your assets.

The following sections describe a few of the possible threats.

2.2.3.1 Unauthorized Access

A common threat that concerns many sites is unauthorized access to computing facilities. Unauthorized access takes many forms.

One means of unauthorized access is the use of another user's account to gain access to a system. The use of any computer resource without prior permission may be considered unauthorized access to computing facilities.

The seriousness of an unauthorized access will vary from site to site. For some sites, the mere act of granting access to an unauthorized user may cause irreparable harm by negative media coverage. For other sites, an unauthorized access opens the door to other security threats. In addition, some sites may be more frequent targets than others; hence the risk from unauthorized access will vary from site to site. The Computer Emergency Response Team (CERT - see section 3.9.7.3.1) has observed that well-known universities, government sites, and military sites seem to attract more intruders.

2.2.3.2 Disclosure of Information

Another common threat is disclosure of information. Determine the value or sensitivity of the information stored on your computers. Disclosure of a password file might allow for future unauthorized accesses. A glimpse of a proposal may give a competitor an unfair advantage. A technical paper may contain years of valuable research.

2.2.3.3 Denial of Service

Computers and networks provide valuable services to their users. Many people rely on these services in order to perform their jobs efficiently. When these services are not available when called upon, a loss in productivity results. Denial of service comes in many forms and might affect users in a number of ways. A network may be rendered unusable by a rogue packet, jamming, or by a disabled network component. A virus might slow down or cripple a computer system. Each site should determine which services are essential, and for each of these services determine the affect to the site if that service were to become disabled.

2.3 Policy Issues

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

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 will be discussed below. In addition you may wish to include a section in your policy concerning ethical use of computing resources. Parker, Swope and Baker [17, PARKER90] and Forester and Morrison [18, FORESTER] are two useful references that address ethical issues.

2.3.1 Who is Allowed to use the Resources?

One step you must take in developing your security policy is defining who is allowed to use your system and services. The policy should explicitly state who is authorized to use what resources.

2.3.2 What is the Proper Use of the Resources?

After determining who is allowed access to system resources it is necessary to provide guidelines for the acceptable use of the resources. You may have different guidelines for different types of users (i.e., students, faculty, external users). The policy should state what is acceptable use as well as unacceptable use.

It should also include types of use that may be restricted. Define limits to access and authority. You will need to consider the level of access various users will have and what resources will be available or restricted to various groups of people.

Your acceptable use policy should clearly state that individual users are responsible for their actions. Their responsibility exists regardless of the security mechanisms that are in place. It should be clearly stated that breaking into accounts or bypassing security is not permitted. The following points should be covered when developing an acceptable use policy:

- 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?

The answer to most of these questions will be "no."

You may wish to incorporate a statement in your policies concerning copyrighted and licensed software. Licensing agreements with vendors may require some sort of effort on your part to ensure that the license is not violated. In addition, you may wish to inform users that the copying of copyrighted software may be a violation of the copyright laws, and is not permitted.

Specifically concerning copyrighted and/or licensed software, you may wish to include the following information:

- Copyrighted and licensed software may not be duplicated unless it is explicitly stated that you may do so.
- Methods of conveying information on the copyright/licensed status of software.
- When in doubt, DON'T COPY.

Your acceptable use policy is very important. A policy which does not clearly state what is not permitted may leave you unable to prove that a user violated policy.

There are exception cases like tiger teams and users or administrators wishing for "licenses to hack"—you may face the situation where users will want to "hack" on your services for security research purposes. You should develop a policy that will determine whether you will permit this type of research on your services and if so, what your guidelines for such research will be.

Points you may wish to cover in this area:

- Whether it is permitted at all.
- What type of activity is permitted: breaking in, releasing worms, releasing viruses, etc..
- What type of controls must be in place to ensure that it does not get out of control (e.g., separate a segment of your network for these tests).
- How you will protect other users from being victims of these activities, including external users and networks.

- The process for obtaining permission to conduct these tests.

In cases where you do permit these activities, you should isolate the portions of the network that are being tested from your main network. Worms and viruses should never be released on a live network.

You may also wish to employ, contract, or otherwise solicit one or more people or organizations to evaluate the security of your services, one of which may include "hacking." You may wish to provide for this in your policy.

2.3.3 Who Is Authorized to Grant Access and Approve Usage?

Your policy should state who is authorized to grant access to your services. Further, it must be determined what type of access they are permitted to give. If you do not have control over who is granted access to your system, you will not have control over who is using your system. Controlling who has the authorization to grant access will also enable you to know who was or was not granting access if problems develop later.

There are many schemes that can be developed to control the distribution of access to your services. The following are the factors that you must consider when determining who will distribute access to your services:

- Will you be distributing access from a centralized point or at various points?

  You can have a centralized distribution point to a distributed system where various sites or departments independently authorize access. The trade off is between security and convenience. The more centralized, the easier to secure.

- What methods will you use for creating accounts and terminating access?

From a security standpoint, you need to examine the mechanism that you will be using to create accounts. 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. Generally, 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. If this is the choice you make, you need to select someone who is trustworthy to perform this task. The opposite solution is to have an integrated system that the people authorized to create accounts run, or the users themselves may actually run. Be aware that even in the restrictive case of having a mechanized facility to create accounts does not remove the potential for abuse. You should have specific procedures developed for the creation of accounts. These procedures should be well documented to prevent confusion and reduce mistakes. A security vulnerability in the account authorization process is not only possible through abuse, but is also possible if a mistake is made. Having clear and well documented procedure will help ensure that these mistakes won't happen. You should also be sure that the people who will be following these procedures understand them.

The granting of access to users is one of the most vulnerable of times. You should ensure that the selection of an initial password cannot be easily guessed. You should avoid using an initial password that is a function of the username, is part of the user's name, or some algorithmically generated password that can easily be guessed. In addition, you should not permit users to continue to use the initial password indefinitely. If possible, you should force users to change the initial password the first time they login. Consider that some users may never even login, leaving their password vulnerable indefinitely. Some sites choose to disable accounts that have never been accessed, and force the owner to reauthorize opening the account.

2.3.4 Who May Have System Administration Privileges?

One security decision that needs to be made very carefully is who will have access to system administrator privileges and passwords for your services. Obviously, the system administrators will need access, but inevitably other users will request special privileges. The policy should address this issue. Restricting privileges is one way to deal with threats from local users. The challenge is to balance restricting access to these to protect security with giving people who need these privileges access so that they can perform their tasks. One approach that can be taken is to grant only enough privilege to accomplish the necessary tasks.

Additionally, people holding special privileges should be accountable to some authority and this should also be identified within the site's security policy. If the people you grant privileges to are not accountable, you run the risk of losing control of your system and will have difficulty managing a compromise in security.

2.3.5 What Are The Users' Rights and Responsibilities?

The policy should incorporate a statement on the users' rights and responsibilities concerning the use of the site's computer systems and services. It should be clearly stated that users are responsible for understanding and respecting the security rules of the systems they are using. The following is a list of topics that you may wish to cover in this area of the policy:

- What guidelines you have regarding resource consumption (whether users are restricted, and if so, what the restrictions are).
- What might constitute abuse in terms of system performance.
- Whether users are permitted to share accounts or let others use their accounts.
- How "secret" users should keep their passwords.
- How often users should change their passwords and any other password restrictions or requirements.
- Whether you provide backups or expect the users to create their own.

- Disclosure of information that may be proprietary.
- Statement on Electronic Mail Privacy (Electronic Communications Privacy Act).
- Your policy concerning controversial mail or postings to mailing lists or discussion groups (obscenity, harassment, etc.).
- Policy on electronic communications: mail forging, etc.

The Electronic Mail Association sponsored a white paper on the privacy of electronic mail in companies [4]. Their basic recommendation is that every site should have a policy on the protection of employee privacy. They also recommend that organizations establish privacy policies that deal with all media, rather than singling out electronic mail.

They suggest 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?

2.3.6 What Are The Rights and Responsibilities of System Administrators Versus Rights of Users

There is a tradeoff between a user's right to absolute privacy and the need of system administrators to gather sufficient information to diagnose problems. There is also a distinction between a system administrator's need to gather information to diagnose problems and investigating security violations. The policy should specify to what degree system administrators can examine user files to diagnose problems or for other purposes, and what rights you grant to the users. You may also wish to make a statement concerning system administrators' obligation to maintaining the privacy of information viewed under these circumstances. A few questions that should be answered are:

- Can an administrator monitor or read a user's files for any reason?
- What are the liabilities?
- Do network administrators have the right to examine network or host traffic?

2.3.7 What To Do With Sensitive Information

Before granting users access to your services, you need to determine at what level you will provide for the security of data on your systems. By determining this, you are determining the level of sensitivity of data that users should store on your systems. You do not want users to store very sensitive information on a system that you are not going to secure very well. You need to tell users who might store sensitive information what services, if any, are appropriate for the storage of sensitive information. This part should include storing of data in different ways (disk, magnetic tape, file servers, etc.). Your policy in this area needs to be coordinated with the policy concerning the rights of system administrators versus users (see section 2.3.6).

2.4 What Happens When the Policy is Violated

It is obvious that when any type of official policy is defined, be it related to computer security or not, it will eventually be broken. The violation may occur due to an individual's negligence, accidental mistake, having not been properly informed of the current policy, or not understanding the current policy. It is equally possible that an individual (or group of individuals) may knowingly perform an act that is in direct violation of the defined policy.

When a policy violation has been detected, the immediate course of action should be predefined to ensure prompt and proper enforcement. An investigation should be performed to determine how and why the violation occurred. Then the appropriate corrective action should be executed. The type and severity of action taken varies depending on the type of violation that occurred.

2.4.1 Determining the Response to Policy Violations

Violations to policy may be committed by a wide variety of users. Some may be local users and others may be from outside the local environment. Sites may find it helpful to define what it considers "insiders" and "outsiders" based upon administrative, legal or political boundaries. These boundaries imply what type of action must be taken to correct the offending party; from a written reprimand to pressing legal charges. So, not only do you need to define actions based on the type of violation, you also need to have a clearly defined series of actions based on the kind of user violating your computer security policy. This all seems rather complicated, but should be addressed long before it becomes necessary as the result of a violation.

One point to remember about your policy is that proper education is your best defense. For the outsiders who are using your computer legally, it is your responsibility to verify that these individuals are aware of the policies that you have set forth.

Having this proof may assist you in the future if legal action becomes necessary.

As for users who are using your computer illegally, the problem is basically the same. What type of user violated the policy and how and why did they do it? Depending on the results of

your investigation, you may just prefer to "plug" the hole in your computer security and chalk it up to experience. Or if a significant amount of loss was incurred, you may wish to take more drastic action.

2.4.2 What to do When Local Users Violate the Policy of a Remote Site

In the event that a local user violates the security policy of a remote site, the local site should have a clearly defined set of administrative actions to take concerning that local user. The site should also be prepared to protect itself against possible actions by the remote site. These situations involve legal issues which should be addressed when forming the security policy.

2.4.3 Defining Contacts and Responsibilities to Outside Organizations

The local security policy should include procedures for interaction with outside organizations. These include law enforcement agencies, other sites, external response team organizations (e.g., the CERT, CIAC) and various press agencies.

The procedure should state who is authorized to make such contact and how it should be handled. Some questions to be answered include:

- Who may talk to the press?
- When do you contact law enforcement and investigative agencies?
- If a connection is made from a remote site, is the system manager authorized to contact that site?
- Can data be released? What kind?

Detailed contact information should be readily available along with clearly defined procedures to follow.

2.4.4 What are the Responsibilities to our Neighbors and Other Internet Sites?

The Security Policy Working Group within the IETF is working on a document entitled, "Policy Guidelines for the Secure Operation of the Internet" [23]. It addresses the issue that the Internet is a cooperative venture and that sites are expected to provide mutual security assistance. This should be addressed when developing a site's policy. The major issue to be determined is how much information should be released. This will vary from site to site according to the type of site (e.g., military, education, commercial) as well as the type of security violation that occurred.

2.4.5 Issues for Incident Handling Procedures

Along with statements of policy, the document being prepared should include procedures for incident handling. This is covered in detail in the next chapter. There should be procedures available that cover all facets of policy violation.

2.5 Locking In or Out

Whenever a site suffers an incident which may compromise computer security, the strategies for reacting may be influenced by two opposing pressures.

If management fears that the site is sufficiently vulnerable, it may choose a "Protect and Proceed" strategy. This approach will have as its primary goal the protection and preservation of the site facilities and to provide for normalcy for its users as quickly as possible. Attempts will be made to actively interfere with the intruder's processes, prevent further access and begin immediate damage assessment and recovery. This process may involve shutting down the facilities, closing off access to the network, or other drastic measures. The drawback is that unless the intruder is identified directly, they may come back into the site via a different path, or may attack another site.

The alternate approach, "Pursue and Prosecute," adopts the opposite philosophy and goals. The primary goal is to allow intruders to continue their activities at the site until the site can identify the responsible persons. This approach is endorsed by law enforcement agencies and prosecutors. The drawback is that the agencies cannot exempt a site from possible user lawsuits if damage is done to their systems and data.

Prosecution is not the only outcome possible if the intruder is identified. If the culprit is an employee or a student, the organization may choose to take disciplinary actions. The computer security policy needs to spell out the choices and how they will be selected if an intruder is caught. Careful consideration must be made by site management regarding their approach to this issue before the problem occurs. The strategy adopted might depend upon each circumstance. Or there may be a global policy which mandates one approach in all circumstances. The pros and cons must be examined thoroughly and the users of the facilities must be made aware of the policy so that they understand their vulnerabilities no matter which approach is taken.

The following are checklists to help a site determine which strategy to adopt: "Protect and Proceed" or "Pursue and Prosecute."

Protect and Proceed

1. If assets are not well protected.
2. If continued penetration could result in great financial risk.
3. If the possibility or willingness to prosecute is not present.

4. If user base is unknown.
5. If users are unsophisticated and their work is vulnerable.
6. If the site is vulnerable to lawsuits from users, e.g., if their resources are undermined.

Pursue and Prosecute

1. If assets and systems are well protected.
2. If good backups are available.
3. If the risk to the assets is outweighed by the disruption caused by the present and possibly future penetrations.
4. If this is a concentrated attack occurring with great frequency and intensity.
5. If the site has a natural attraction to intruders, and consequently regularly attracts intruders.
6. If the site is willing to incur the financial (or other) risk to assets by allowing the penetrator continue.
7. If intruder access can be controlled.
8. If the monitoring tools are sufficiently well-developed to make the pursuit worthwhile.
9. If the support staff is sufficiently clever and knowledgable about the operating system, related utilities, and systems to make the pursuit worthwhile.
10. If there is willingness on the part of management to prosecute.
11. If the system adminitrators know in general what kind of evidence would lead to prosecution.
12. If there is established contact with knowledgeable law enforcement.
13. If there is a site representative versed in the relevant legal issues.
14. If the site is prepared for possible legal action from its own users if their data or systems become compromised during the pursuit.

2.6 Interpreting the Policy

It is important to define who will interpret the policy. This could be an individual or a committee. No matter how well written, the policy will require interpretation from time to time and this body would serve to review, interpret, and revise the policy as needed.

2.7 Publicizing the Policy

Once the site security policy has been written and established, a vigorous process should be engaged to ensure that the policy statement is widely and thoroughly disseminated and discussed. A mailing of the policy should not be considered sufficient. A period for comments should be allowed before the policy becomes effective to ensure that all affected users have a chance to state their reactions and discuss any unforeseen ramifications. Ideally, the policy should strike a balance between protection and productivity. Meetings should be held to elicit these comments, and also to ensure that the policy is correctly understood. (Policy promulgators are not necessarily noted for their skill with the language.) These meetings should involve higher management as well as line employees.

Security is a collective effort.

In addition to the initial efforts to publicize the policy, it is essential for the site to maintain a continual awareness of its computer security policy. Current users may need periodic reminders. New users should have the policy included as part of their site introduction packet. As a condition for using the site facilities, it may be advisable to have them sign a statement that they have read and understood the policy. Should any of these users require legal action for serious policy violations, this signed statement might prove to be a valuable aid.

3. Establishing Procedures to Prevent Security Problems

The security policy defines what needs to be protected. This section discusses security procedures which specify what steps will be used to carry out the security policy.

3.1 Security Policy Defines What Needs to be Protected

The security policy defines the WHAT's: what needs to be protected, what is most important, what the priorities are, and what the general approach to dealing with security problems should be.

The security policy by itself doesn't say HOW things are protected.

That is the role of security procedures, which this section discusses. The security policy should be a high level document, giving general strategy. The security procedures need to set out, in detail, the precise steps your site will take to protect itself. The security policy should include a general risk assessment of the types of threats a site is mostly likely to face and the consequences of those threats (see section 2.2). Part of doing a risk assessment will include creating a

general list of assets that should be protected (section 2.2.2). This information is critical in devising cost-effective procedures.

It is often tempting to start creating security procedures by deciding on different mechanisms first: "our site should have logging on all hosts, call-back modems, and smart cards for all users." This approach could lead to some areas that have too much protection for the risk they face, and other areas that aren't protected enough. Starting with the security policy and the risks it outlines should ensure that the procedures provide the right level of protect for all assets.

3.2 Identifying Possible Problems

To determine risk vulnerabilities must be identified. Part of the purpose of the policy is to aid in shoring up the vulnerabilities and thus to decrease the risk in as many areas as possible. Several of the more popular problem areas are presented in sections below. This list is by no means complete. In addition, each site is likely to have a few unique vulnerabilities.

3.2.1 Access Points

Access points are typically used for entry by unauthorized users. Having many access points increases the risk of access to an organization's computer and network facilities.

Network links to networks outside the organization allow access into the organization for all others connected to that external network. A network link typically provides access to a large number of network services, and each service has a potential to be compromised.

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.

Terminal servers themselves can be a source of problem. Many terminal servers do not require any kind of authentication. Intruders often use terminal servers to disguise their actions, dialing in on a local phone and then using the terminal server to go out to the local network. Some terminal servers are configured so that intruders can TELNET [19] in from outside the network, and then TELNET back out again, again serving to make it difficult to trace them.

3.2.2 Misconfigured Systems

Misconfigured systems form a large percentage of security holes. Today's operating systems and their associated software have become so complex that understanding how the system works has become a full-time job. Often, systems managers will be non-specialists chosen from the current organization's staff. Vendors are also partly responsible for misconfigured systems. To make the system installation process easier, vendors occasionally choose initial configurations that are not secure in all environments.

3.2.3 Software Bugs

Software will never be bug free. Publicly known security bugs are common methods of unauthorized entry. Part of the solution to this problem is to be aware of the security problems and to update the software when problems are detected. When bugs are found, they should be reported to the vendor so that a solution to the problem can be implemented and distributed.

3.2.4 "Insider" Threats

An insider to the organization may be a considerable threat to the security of the computer systems. Insiders often have direct access to the computer and network hardware components. The ability to access the components of a system makes most systems easier to compromise. Most desktop workstations can be easily manipulated so that they grant privileged access. Access to a local area network provides the ability to view possibly sensitive data traversing the network.

3.3 Choose Controls to Protect Assets in a Cost-Effective Way

After establishing what is to be protected, and assessing the risks these assets face, it is necessary to decide how to implement the controls which protect these assets. The controls and protection mechanisms should be selected in a way so as to adequately counter the threats found during risk assessment, and to implement those controls in a cost effective manner. It makes little sense to spend an exorbitant sum of money and overly constrict the user base if the risk of exposure is very small.

3.3.1 Choose the Right Set of Controls

The controls that are selected represent the physical embodiment of your security policy. They are the first and primary line of defense in the protection of your assets. It is therefore most important to ensure that the controls that you select are the right set of controls. If the major threat to your system is outside penetrators, it probably doesn't make much sense to use biometric devices to authenticate your regular system users. On the other hand, if the major threat is unauthorized use of computing resources by regular system users, you'll probably want to establish very rigorous automated accounting procedures.

3.3.2 Use Common Sense

Common sense is the most appropriate tool that can be used to establish your security policy. Elaborate security schemes and mechanisms are impressive, and they do have their place, yet there is little point in investing money and time on an elaborate implementation scheme if the simple controls are forgotten. For example, no matter how elaborate a system you put into place on top of existing security controls, a single user with a poor password can still leave your system open to attack.

3.4 Use Multiple Strategies to Protect Assets

Another method of protecting assets is to use multiple strategies. In this way, if one strategy fails or is circumvented, another strategy comes into play to continue protecting the asset. By using several simpler strategies, a system can often be made more secure than if one very sophisticated method were used in its place. For example, dial-back modems can be used in conjunction with traditional logon mechanisms. Many similar approaches could be devised that provide several levels of protection for assets. However, it's very easy to go overboard with extra mechanisms. One must keep in mind exactly what it is that needs to be protected.

3.5 Physical Security

It is a given in computer security if the system itself is not physically secure, nothing else about the system can be considered secure. With physical access to a machine, an intruder can halt the machine, bring it back up in privileged mode, replace or alter the disk, plant Trojan horse programs (see section 2.13.9.2), or take any number of other undesirable (and hard to prevent) actions. Critical communications links, important servers, and other key machines should be located in physically secure areas. Some security systems (such as Kerberos) require that the machine be physically secure.

If you cannot physically secure machines, care should be taken about trusting those machines. Sites should consider limiting access from non-secure machines to more secure machines. In particular, allowing trusted access (e.g., the BSD Unix remote commands such as rsh) from these kinds of hosts is particularly risky. For machines that seem or are intended to be physically secure, care should be taken about who has access to the machines. Remember that custodial and maintenance staff often have keys to rooms.

3.6 Procedures to Recognize Unauthorized Activity

Several simple procedures can be used to detect most unauthorized uses of a computer system. These procedures use tools provided with the operating system by the vendor, or tools publicly available from other sources.

3.6.1 Monitoring System Use

System monitoring can be done either by a system administrator, or by software written for the purpose. Monitoring a system involves looking at several parts of the system and searching for anything unusual. Some of the easier ways to do this are described in this section.

The most important thing about monitoring system use is that it be done on a regular basis. Picking one day out of the month to monitor the system is pointless, since a security breach can be isolated to a matter of hours. Only by maintaining a constant vigil can you expect to detect security violations in time to react to them.

3.6.2 Tools for Monitoring the System

This section describes tools and methods for monitoring a system against unauthorized access and use.

3.6.2.1 Logging

Most operating systems store numerous bits of information in log files. Examination of these log files on a regular basis is often the first line of defense in detecting unauthorized use of the system.

- Compare lists of currently logged in users and past login histories. Most users typically log in and out at roughly the same time each day. An account logged in outside the "normal" time for the account may be in use by an intruder.
- Many systems maintain accounting records for billing purposes. These records can also be used to determine usage patterns for the system; unusual accounting records may indicate unauthorized use of the system.
- System logging facilities, such as the UNIX "syslog" utility, should be checked for unusual error messages from system software. For example, a large number of failed login attempts in a short period of time may indicate someone trying to guess passwords.
- Operating system commands which list currently executing processes 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.

3.6.2.2 Monitoring Software

Other monitoring tools can easily be constructed using standard operating system software, by using several, often unrelated, programs together. For example, checklists of file ownerships and permission settings can be constructed (for example, with "ls" and "find" on UNIX) and stored off-line. These lists can then be reconstructed periodically and compared against the master checklist (on UNIX, by using the "diff" utility).

Differences may indicate that unauthorized modifications have been made to the system.

Still other tools are available from third-party vendors and public software distribution sites. Section 3.9.9 lists several sources from which you can learn what tools are available and how to get them.

3.6.2.3 Other Tools

Other tools can also be used to monitor systems for security violations, although this is not their primary purpose. For example, network monitors can be used to detect and log connections from unknown sites.

3.6.3 Vary the Monitoring Schedule

The task of system monitoring is not as daunting as it may seem.

System administrators can execute many of the commands used for monitoring periodically throughout the day during idle moments (e.g., while talking on the telephone), rather than spending fixed periods of each day monitoring the system. By executing the commands frequently, you will rapidly become used to seeing "normal" output, and will easily spot things which are out of the ordinary. In addition, by running various monitoring commands at different times throughout the day, you make it hard for an intruder to predict your actions. For example, if an intruder knows that each day at 5:00 p.m. the system is checked to see that everyone has logged off, he will simply wait until after the check has completed before logging in. But the intruder cannot guess when a system administrator might type a command to display all logged-in users, and thus he runs a much greater risk of detection.

Despite the advantages that regular system monitoring provides, some intruders will be aware of the standard logging mechanisms in use on systems they are attacking. They will actively pursue and attempt to disable monitoring mechanisms. Regular monitoring therefore is useful in detecting intruders, but does not provide any guarantee that your system is secure, nor should monitoring be considered an infallible method of detecting unauthorized use.

3.7 Define Actions to Take When Unauthorized Activity is Suspected

Sections 2.4 and 2.5 discussed the course of action a site should take when it suspects its systems are being abused. The computer security policy should state the general approach towards dealing with these problems.

The procedures for dealing with these types of problems should be written down. Who has authority to decide what actions will be taken? Should law enforcement be involved? Should your organization cooperate with other sites in trying to track down an intruder? Answers to all the questions in section 2.4 should be part of the incident handling procedures.

Whether you decide to lock out or pursue intruders, you should have tools and procedures ready to apply. It is best to work up these tools and procedures before you need them. Don't wait until an intruder is on your system to figure out how to track the intruder's actions; you will be busy enough if an intruder strikes.

3.8 Communicating Security Policy

Security policies, in order to be effective, must be communicated to both the users of the system and the system maintainers. This section describes what these people should be told, and how to tell them.

3.8.1 Educating the Users

Users should be made aware of how the computer systems are expected to be used, and how to protect themselves from unauthorized users.

3.8.1.1 Proper Account/Workstation Use

All users should be informed about what is considered the "proper" use of their account or workstation ("proper" use is discussed in section 2.3.2). This can most easily be done at the time a user receives their account, by giving them a policy statement. Proper use policies typically dictate things such as whether or not the account or workstation may be used for personal activities (such as checkbook balancing or letter writing), whether profit-making activities are allowed, whether game playing is permitted, and so on. These policy statements may also be used to summarize how the computer facility is licensed and what software licenses are held by the institution; for example, many universities have educational licenses which explicitly prohibit commercial uses of the system. A more complete list of items to consider when writing a policy statement is given in section 2.3.

3.8.1.2 Account/Workstation Management Procedures

Each user should be told how to properly manage their account and workstation. This includes explaining how to protect files stored on the system, how to log out or lock the terminal or workstation, and so on. Much of this information is typically covered in the "beginning user" documentation provided by the operating system vendor, although many sites elect to supplement this material with local information.

If your site offers dial-up modem access to the computer systems, special care must be taken to inform users of the security problems inherent in providing this access. Issues such as making sure to log out before hanging up the modem should be covered when the user is initially given dial-up access.

Likewise, access to the systems via local and wide-area networks presents its own set of security problems which users should be made aware of. Files which grant "trusted host" or "trusted user" status to remote systems and users should be carefully explained.

3.8.1.3 Determining Account Misuse

Users should be told how to detect unauthorized access to their account. If the system prints the last login time when a user logs in, he or she should be told to check that time and note whether or not it agrees with the last time he or she actually logged in.

Command interpreters on some systems (e.g., the UNIX C shell) maintain histories of the last several commands executed. Users should check these histories to be sure someone has not executed other commands with their account.

3.8.1.4 Problem Reporting Procedures

A procedure should be developed to enable users to report suspected misuse of their accounts or other misuse they may have noticed. This can be done either by providing the name and telephone number of a system administrator who manages security of the computer system, or by creating an electronic mail address (e.g., "security") to which users can address their problems.

3.8.2 Educating the Host Administrators

In many organizations, computer systems are administered by a wide variety of people. These administrators must know how to protect their own systems from attack and unauthorized use, as well as how to communicate successful penetration of their systems to other administrators as a warning.

3.8.2.1 Account Management Procedures

Care must be taken when installing accounts on the system in order to make them secure. When installing a system from distribution media, the password file should be examined for "standard" accounts provided by the vendor. Many vendors provide accounts for use by system services or field service personnel. These accounts typically have either no password or one which is common knowledge. These accounts should be given new passwords if they are needed, or disabled or deleted from the system if they are not.

Accounts without passwords are generally very dangerous since they allow anyone to access the system. Even accounts which do not execute a command interpreter (e.g., accounts which exist only to see who is logged in to the system) can be compromised if set up incorrectly. A related concept, that of "anonymous" file transfer (FTP) [20], allows users from all over the network to access your system to retrieve files from (usually) a protected disk area. You should carefully weigh the benefits that an account without a password provides against the security risks of providing such access to your system. If the operating system provides a "shadow" password facility which stores passwords in a separate file accessible only to privileged users, this facility should be used. System V UNIX, SunOS 4.0 and above, and versions of Berkeley UNIX after 4.3BSD Tahoe, as well as others, provide this feature. It protects passwords by hiding their encrypted values from unprivileged users. This prevents an attacker from copying your password file to his or her machine and then attempting to break the passwords at his or her leisure.

Keep track of who has access to privileged user accounts (e.g., "root" on UNIX or "MAINT" on VMS). Whenever a privileged user leaves the organization or no longer has need of the privileged account, the passwords on all privileged accounts should be changed.

3.8.2.2 Configuration Management Procedures

When installing a system from the distribution media or when installing third-party software, it is important to check the installation carefully. Many installation procedures assume a

"trusted" site, and hence will install files with world write permission enabled, or otherwise compromise the security of files.

Network services should also be examined carefully when first installed. Many vendors provide default network permission files which imply that all outside hosts are to be "trusted," which is rarely the case when connected to wide-area networks such as the Internet.

Many intruders collect information on the vulnerabilities of particular system versions. The older a system, the more likely it is that there are security problems in that version which have since been fixed by the vendor in a later release.

For this reason, it is important to weigh the risks of not upgrading to a new operating system release (thus leaving security holes unplugged) against the cost of upgrading to the new software (possibly breaking third-party software, etc.).

Bug fixes from the vendor should be weighed in a similar fashion, with the added note that "security" fixes from a vendor usually address fairly serious security problems.

Other bug fixes, received via network mailing lists and the like, should usually be installed, but not without careful examination. Never install a bug fix unless you're sure you know what the consequences of the fix are - there's always the possibility that an intruder has suggested a "fix" which actually gives him or her access to your system.

3.8.2.3 Recovery Procedures—Backups

It is impossible to overemphasize the need for a good backup strategy. File system backups not only protect you in the event of hardware failure or accidental deletions, but they also protect you against unauthorized changes made by an intruder. Without a copy of your data the way it's "supposed" to be, it can be difficult to undo something an attacker has done.

Backups, especially if run daily, can also be useful in providing a history of an intruder's activities. Looking through old backups can establish when your system was first penetrated. Intruders may leave files around which, although deleted later, are captured on the backup tapes. Backups can also be used to document an intruder's activities to law enforcement agencies if necessary.

A good backup strategy will dump the entire system to tape at least once a month. Partial (or incremental") dumps should be done at least twice a week, and ideally they should be done daily. Commands specifically designed for performing file system backups (e.g., UNIX "dump" or VMS "BACKUP") should be used in preference to other file copying commands, since these tools are designed with the express intent of restoring a system to a known state.

3.8.2.4 Problem Reporting Procedures

As with users, system administrators should have a defined procedure for reporting security problems. In large installations, this is often done by creating an electronic mail alias which

contains the names of all system administrators in the organization. Other methods include setting up some sort of response team similar to the CERT, or establishing a "hotline" serviced by an existing support group.

3.9 Resources to Prevent Security Breaches

This section discusses software, hardware, and procedural resources that can be used to support your site security policy.

3.9.1 Network Connections and Firewalls

A "firewall" is put in place in a building to provide a point of resistance to the entry of flames into another area. Similarly, a secretary's desk and reception area provides a point of controlling access to other office spaces. This same technique can be applied to a computer site, particularly as it pertains to network connections.

Some sites will be connected only to other sites within the same organization and will not have the ability to connect to other networks. Sites such as these are less susceptible to threats from outside their own organization, although intrusions may still occur via paths such as dial-up modems. On the other hand, many other organizations will be connected to other sites via much larger networks, such as the Internet. These sites are susceptible to the entire range of threats associated with a networked environment.

The risks of connecting to outside networks must be weighed against the benefits. It may be desirable to limit connection to outside networks to those hosts which do not store sensitive material, keeping "vital" machines (such as those which maintain company payroll or inventory systems) isolated. If there is a need to participate in a Wide Area Network (WAN), consider restricting all access to your local network through a single system. That is, all access to or from your own local network must be made through a single host computer that acts as a firewall between you and the outside world. This firewall system should be rigorously controlled and password protected, and external users accessing it should also be constrained by restricting the functionality available to remote users. By using this approach, your site could relax some of the internal security controls on your local net, but still be afforded the protection of a rigorously controlled host front end.

Note that even with a firewall system, compromise of the firewall could result in compromise of the network behind the firewall. Work has been done in some areas to construct a firewall which even when compromised, still protects the local network [6, CHESWICK].

3.9.2 Confidentiality

Confidentiality, the act of keeping things hidden or secret, is one of the primary goals of computer security practitioners. Several mechanisms are provided by most modern operating systems to enable users to control the dissemination of information.

Depending upon where you work, you may have a site where everything is protected, or a site where all information is usually regarded as public, or something in-between. Most sites lean toward the in-between, at least until some penetration has occurred.

Generally, there are three instances in which information is vulnerable to disclosure: when the information is stored on a computer system, when the information is in transit to another system (on the network), and when the information is stored on backup tapes.

The first of these cases is controlled by file permissions, access control lists, and other similar mechanisms. The last can be controlled by restricting access to the backup tapes (by locking them in a safe, for example). All three cases can be helped by using encryption mechanisms.

3.9.2.1 Encryption (Hardware and Software)

Encryption is the process of taking information that exists in some readable form and converting it into a non-readable form. There are several types of commercially available encryption packages in both hardware and software forms. Hardware encryption engines have the advantage that they are much faster than the software equivalent, yet because they are faster, they are of greater potential benefit to an attacker who wants to execute a brute-force attack on your encrypted information. The advantage of using encryption is that, even if other access control mechanisms (passwords, file permissions, etc.) are compromised by an intruder, the data is still unusable. Naturally, encryption keys and the like should be protected at least as well as account passwords.

Information in transit (over a network) may be vulnerable to interception as well. Several solutions to this exist, ranging from simply encrypting files before transferring them (end-to-end encryption) to special network hardware which encrypts everything it sends without user intervention (secure links). The Internet as a whole does not use secure links, thus end- to-end encryption must be used if encryption is desired across the Internet.

3.9.2.1.1 Data Encryption Standard (DES)

DES is perhaps the most widely used data encryption mechanism today. Many hardware and software implementations exist, and some commercial computers are provided with a software version. DES transforms plain text information into encrypted data (or ciphertext) by means of a special algorithm and "seed" value called a key. So long as the key is retained (or remembered) by the original user, the ciphertext can be restored to the original plain text.

One of the pitfalls of all encryption systems is the need to remember the key under which a thing was encrypted (this is not unlike the password problem discussed elsewhere in this document). If the key is written down, it becomes less secure. If forgotten, there is little (if any) hope of recovering the original data.

Most UNIX systems provide a DES command that enables a user to encrypt data using the DES algorithm.

3.9.2.1.2 Crypt

Similar to the DES command, the UNIX crypt" command allows a user to encrypt data. Unfortunately, the algorithm used by "crypt" is very insecure (based on the World War II "Enigma" device), and files encrypted with this command can be decrypted easily in a matter of a few hours. Generally, use of the "crypt" command should be avoided for any but the most trivial encryption tasks.

3.9.2.2 Privacy Enhanced Mail

Electronic mail normally transits the network in the clear (i.e., anyone can read it). This is obviously not the optimal solution. Privacy enhanced mail provides a means to automatically encrypt electronic mail messages so that a person eavesdropping at a mail distribution node is not (easily) capable of reading them. Several privacy enhanced mail packages are currently being developed and deployed on the Internet.

The Internet Activities Board Privacy Task Force has defined a draft standard, elective protocol for use in implementing privacy enhanced mail. This protocol is defined in RFCs 1113, 1114, and 1115 [7,8,9]. Please refer to the current edition of the "IAB Official Protocol Standards" (currently, RFC 1200 [21]) for the standardization state and status of these protocols.

3.9.3 Origin Authentication

We mostly take it on faith that the header of an electronic mail message truly indicates the originator of a message. However, it is easy to "spoof," or forge the source of a mail message. Origin authentication provides a means to be certain of the originator of a message or other object in the same way that a Notary Public assures a signature on a legal document. This is done by means of a "Public Key" cryptosystem.

A public key cryptosystem differs from a private key cryptosystem in several ways. First, a public key system uses two keys, a Public Key that anyone can use (hence the name) and a Private Key that only the originator of a message uses. The originator uses the private key to encrypt the message (as in DES). The receiver, who has obtained the public key for the originator, may then decrypt the message.

In this scheme, the public key is used to authenticate the originator's use of his or her private key, and hence the identity of the originator is more rigorously proven. The most widely known implementation of a public key cryptosystem is the RSA system [26]. The Internet standard for privacy enhanced mail makes use of the RSA system.

3.9.4 Information Integrity

Information integrity refers to the state of information such that it is complete, correct, and unchanged from the last time in which it was verified to be in an "integral" state. The value of information integrity to a site will vary. For example, it is more important for military and

government installations to prevent the "disclosure" of classified information, whether it is right or wrong. A bank, on the other hand, is far more concerned with whether the account information maintained for its customers is complete and accurate.

Numerous computer system mechanisms, as well as procedural controls, have an influence on the integrity of system information. Traditional access control mechanisms maintain controls over who can access system information. These mechanisms alone are not sufficient in some cases to provide the degree of integrity required. Some other mechanisms are briefly discussed below.

It should be noted that there are other aspects to maintaining system integrity besides these mechanisms, such as two-person controls, and integrity validation procedures. These are beyond the scope of this document.

3.9.4.1 Checksums

Easily the simplest mechanism, a simple checksum routine can compute a value for a system file and compare it with the last known value. If the two are equal, the file is probably unchanged. If not, the file has been changed by some unknown means.

Though it is the easiest to implement, the checksum scheme suffers from a serious failing in that it is not very sophisticated and a determined attacker could easily add enough characters to the file to eventually obtain the correct value.

A specific type of checksum, called a CRC checksum, is considerably more robust than a simple checksum. It is only slightly more difficult to implement and provides a better degree of catching errors. It too, however, suffers from the possibility of compromise by an attacker.

Checksums may be used to detect the altering of information.

However, they do not actively guard against changes being made.

For this, other mechanisms such as access controls and encryption should be used.

3.9.4.2 Cryptographic Checksums

Cryptographic checksums (also called cryptosealing) involve breaking a file up into smaller chunks, calculating a (CRC) checksum for each chunk, and adding the CRCs together. Depending upon the exact algorithm used, this can result in a nearly unbreakable method of determining whether a file has been changed. This mechanism suffers from the fact that it is sometimes computationally intensive and may be prohibitive except in cases where the utmost integrity protection is desired.

Another related mechanism, called a one-way hash function (or a Manipulation Detection Code (MDC)) can also be used to uniquely identify a file. The idea behind these functions is that no two inputs can produce the same output, thus a modified file will not have the same

hash value. One-way hash functions can be implemented efficiently on a wide variety of systems, making unbreakable integrity checks possible. (Snefru, a one-way hash function available via USENET as well as the Internet is just one example of an efficient one-way hash function.) [10]

3.9.5 Limiting Network Access

The dominant network protocols in use on the Internet, IP (RFC 791) [11], TCP (RFC 793) [12], and UDP (RFC 768) [13], carry certain control information which can be used to restrict access to certain hosts or networks within an organization. The IP packet header contains the network addresses of both the sender and recipient of the packet. Further, the TCP and UDP protocols provide the notion of a "port," which identifies the endpoint (usually a network server) of a communications path. In some instances, it may be desirable to deny access to a specific TCP or UDP port, or even to certain hosts and networks altogether.

3.9.5.1 Gateway Routing Tables

One of the simplest approaches to preventing unwanted network connections is to simply remove certain networks from a gateway's routing tables. This makes it "impossible" for a host to send packets to these networks. (Most protocols require bidirectional packet flow even for unidirectional data flow, thus breaking one side of the route is usually sufficient.)

This approach is commonly taken in "firewall" systems by preventing the firewall from advertising local routes to the outside world. The approach is deficient in that it often prevents "too much" (e.g., in order to prevent access to one system on the network, access to all systems on the network is disabled).

3.9.5.2 Router Packet Filtering

Many commercially available gateway systems (more correctly called routers) provide the ability to filter packets based not only on sources or destinations, but also on source-destination combinations. This mechanism can be used to deny access to a specific host, network, or subnet from any other host, network, or subnet.

Gateway systems from some vendors (e.g., cisco Systems) support an even more complex scheme, allowing finer control over source and destination addresses. Via the use of address masks, one can deny access to all but one host on a particular network.

The cisco Systems also allow packet screening based on IP protocol type and TCP or UDP port numbers [14]. This can also be circumvented by "source routing" packets destined for the "secret" network. Source routed packets may be filtered out by gateways, but this may restrict other legitimate activities, such as diagnosing routing problems.

3.9.6 Authentication Systems

Authentication refers to the process of proving a claimed identity to the satisfaction of some permission-granting authority. Authentication systems are hardware, software, or procedural mechanisms that enable a user to obtain access to computing resources. At the simplest level, the system administrator who adds new user accounts to the system is part of the system authentication mechanism. At the other end of the spectrum, fingerprint readers or retinal scanners provide a very high-tech solution to establishing a potential user's identity. Without establishing and proving a user's identity prior to establishing a session, your site's computers are vulnerable to any sort of attack.

Typically, a user authenticates himself or herself to the system by entering a password in response to a prompt.

Challenge/Response mechanisms improve upon passwords by prompting the user for some piece of information shared by both the computer and the user (such as mother's maiden name, etc.).

3.9.6.1 Kerberos

Kerberos, named after the dog who in 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 login and start a session from any computer located on the campus. This has clear advantages in certain environments where there are a large number of potential users who may establish a connection from any one of a large number of workstations. Some vendors are now incorporating Kerberos into their systems. It should be noted that while Kerberos makes several advances in the area of authentication, some security weaknesses in the protocol still remain [15].

3.9.6.2 Smart Cards

Several systems use smart cards" (a small calculator-like device) to help authenticate users. These systems depend on the user having an object in their possession. One such system involves a new password procedure that require a user to enter a value obtained from a "smart card" when asked for a password by the computer. Typically, the host machine will give the user some piece of information that is entered into the keyboard of the smart card. The smart card will display a response which must then be entered into the computer before the session will be established. Another such system involves a smart card which displays a number which changes over time, but which is synchronized with the authentication software on the computer.

This is a better way of dealing with authentication than with the traditional password approach. On the other hand, some say it's inconvenient to carry the smart card. Start-up costs are likely to be high as well.

3.9.7 Books, Lists, and Informational Sources

There are many good sources for information regarding computer security. The annotated bibliography at the end of this document can provide you with a good start. In addition, information can be obtained from a variety of other sources, some of which are described in this section.

3.9.7.1 Security Mailing Lists

The UNIX Security mailing list exists to notify system administrators of security problems before they become common knowledge, and to provide security enhancement information. It is a restricted-access list, open only to people who can be verified as being principal systems people at a site. Requests to join the list must be sent by either 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 address you want on the list, an indication of whether you want to be on the mail reflector list or receive weekly digests, the electronic mail address and voice telephone number of the site contact if it isn't you, and the name, address, and telephone number of your organization. This information should be sent to SECURITY-REQUEST@CPD.COM. The RISKS digest is a component of the ACM Committee on Computers and Public Policy, moderated by Peter G. Neumann. It is a discussion forum on risks to the public in computers and related systems, and along with discussing computer security and privacy issues, has discussed such subjects as the Stark incident, the shooting down of the Iranian airliner in the Persian Gulf (as it relates to the computerized weapons systems), problems in air and railroad traffic control systems, software engineering, and so on. To join the mailing list, send a message to RISKS-REQUEST@CSL.SRI.COM. This list is also available in the USENET newsgroup "comp.risks."

The VIRUS-L list is a forum for the discussion of 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 the line:

```
SUB VIRUS-L your full name
```

to the address LISTSERV%LEHIIBM1.BITNET@MITVMA.MIT.EDU. This list is also available via the USENET newsgroup "comp.virus."

The Computer Underground Digest "is an open forum dedicated to sharing information among computerists and to the presentation and debate of diverse views." While not directly a security list, it does contain discussions about privacy and other security related topics. The list can be read on USENET as alt.society.cu-digest, or to join the mailing list, send mail to Gordon Myer (TK0JUT2%NIU.bitnet@mitvma.mit.edu). Submissions may be mailed to: cud@chinacat.unicom.com.

3.9.7.2 Networking Mailing Lists

The TCP-IP mailing list is intended to act as a discussion forum for developers and maintainers of implementations of the TCP/IP protocol suite. It also discusses network-related security problems when they involve programs providing network services, such as "Sendmail." To join the TCP-IP list, send a message to TCP-IP-REQUEST@NISC.SRI.COM. This list is also available in the USENET newsgroup "comp.protocols.tcp-ip." SUN-NETS is a discussion list for items pertaining to networking on Sun systems. Much of the discussion is related to NFS, NIS (formally Yellow Pages), and name servers. To subscribe, send a message to SUN-NETS-REQUEST@UMIACS.UMD.EDU.

The USENET groups misc.security and alt.security also discuss security issues. misc.security is a moderated group and also includes discussions of physical security and locks. alt.security is unmoderated.

3.9.7.3 Response Teams

Several organizations have formed special groups of people to deal with computer security problems. These teams collect information about possible security holes and disseminate it to the proper people, track intruders, and assist in recovery from security violations. The teams typically have both electronic mail distribution lists as well as a special telephone number which can be called for information or to report a problem.

Many of these teams are members of the CERT System, which is coordinated by the National Institute of Standards and Technology (NIST), and exists to facilitate the exchange of information between the various teams.

3.9.7.3.1 DARPA 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) to address computer security concerns of research users of the Internet. It is operated by the Software Engineering Institute (SEI) at Carnegie-Mellon University (CMU). The CERT can immediately confer with experts to diagnose and solve security problems, and also establish and maintain communications with the affected computer users and government authorities as appropriate.

The CERT/CC serves as a clearing house for the identification and repair of security vulnerabilities, informal assessments of existing systems, improvement of emergency response capability, and both vendor and user security awareness. In addition, the team works with vendors of various systems in order to coordinate the fixes for security problems.

The CERT/CC sends out security advisories to the CERT- ADVISORY mailing list whenever appropriate. They also operate a 24-hour hotline that can be called to report security problems (e.g., someone breaking into your system), as well as to obtain current (and accurate) information about rumored security problems.

To join the CERT-ADVISORY mailing list, send a message to CERT@CERT.SEI.CMU.EDU and ask to be added to the mailing list. The material sent to this list also appears in the USENET newsgroup "comp.security.announce." Past advisories are available for anonymous FTP from the host CERT.SEI.CMU.EDU. The 24-hour hotline number is (412) 268- 7090.

The CERT/CC also maintains a CERT-TOOLS list to encourage the exchange 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 a message to CERT-TOOLS- REQUEST@CERT.SEI.CMU.EDU and ask to be added to the mailing list.

The CERT/CC maintains other generally useful security information for anonymous FTP from CERT.SEI.CMU.EDU. Get the README file for a list of 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.

3.9.7.3.2 DDN Security Coordination Center

For DDN users, the Security Coordination Center (SCC) serves a function similar to CERT. The SCC is the DDN's clearing-house for host/user security problems and fixes, and works with the DDN Network Security Officer. The SCC also distributes the DDN Security Bulletin, which communicates information on network and host security exposures, fixes, and concerns to security and management personnel at DDN facilities. It is available online, via kermit or anonymous FTP, from the host NIC.DDN.MIL, in SCC:DDN-SECURITY-yy-nn.TXT (where "yy" is the year and "nn" is the bulletin number). The SCC provides immediate assistance with DDN- related host security problems; call (800) 235-3155 (6:00 a.m. to 5:00 p.m. Pacific Time) or send email to SCC@NIC.DDN.MIL. For 24 hour coverage, call the MILNET Trouble Desk (800) 451-7413 or AUTOVON 231-1713.

3.9.7.3.3 NIST Computer Security Resource and Response Center

The National Institute of Standards and Technology (NIST) has responsibility within the U.S. Federal Government for computer science and technology activities. NIST has played a strong role in organizing the CERT System and is now serving as the CERT System Secretariat. NIST also operates a Computer Security Resource and Response Center (CSRC) to provide help and information regarding computer security events and incidents, as well as to raise awareness about computer security vulnerabilities.

The CSRC team operates a 24-hour hotline, at (301) 975-5200.

For individuals with access to the Internet, on-line publications and computer security information can be obtained via anonymous FTP from the host CSRC.NCSL.NIST.GOV (129.6.48.87). NIST also operates a personal computer bulletin board that contains information regarding computer viruses as well as other aspects of computer security. To access this board, set your modem to 300/1200/2400 BPS, 1 stop bit, no parity, and 8-bit characters, and call (301) 948-5717. All users are given full access to the board immediately upon registering.

NIST has produced several special publications related to computer security and computer viruses in particular; some of these publications are downloadable. For further information, contact NIST at the following address:

Computer Security Resource and Response Center
A-216 Technology
Gaithersburg, MD 20899
Telephone: (301) 975-3359
Electronic Mail: CSRC@nist.gov

3.9.7.3.4 DOE Computer Incident Advisory Capability (CIAC)

CIAC is the Department of Energy's (DOE's) Computer Incident Advisory Capability. CIAC is a four-person team of computer scientists from Lawrence Livermore National Laboratory (LLNL) charged with the primary responsibility of assisting DOE sites faced with computer security incidents (e.g., intruder attacks, virus infections, worm attacks, etc.). This capability is available to DOE sites on a 24-hour-a-day basis.

CIAC was formed to provide a centralized response capability (including technical assistance), to keep sites informed of current events, to deal proactively with computer security issues, and to maintain liaisons with other response teams and agencies. CIAC's charter is to assist sites (through direct technical assistance, providing information, or referring inquiries to other technical experts), serve as a clearinghouse for information about threats/known incidents/vulnerabilities, develop guidelines for incident handling, develop software for responding to events/incidents, analyze events and trends, conduct training and awareness activities, and alert and advise sites about vulnerabilities and potential attacks.

CIAC's business hours phone number is (415) 422-8193 or FTS 532-8193. CIAC's e-mail address is CIAC@TIGER.LLNL.GOV. 3.9.7.3.5 NASA Ames Computer Network Security Response Team The Computer Network Security Response Team (CNSRT) is NASA Ames Research Center's local version of the DARPA CERT. Formed in August of 1989, the team has a constituency that is primarily Ames users, but it is also involved in assisting other NASA Centers and federal agencies. CNSRT maintains liaisons with the DOE's CIAC team and the DARPA CERT. It is also a charter member of the CERT System. The team may be reached by 24 hour pager at (415) 694-0571, or by electronic mail to CNSRT@AMES.ARC.NASA.GOV.

3.9.7.4 DDN Management Bulletins

The DDN Management Bulletin is distributed electronically by the DDN NIC under contract to the Defense Communications Agency (DCA). It is a means of communicating official policy, procedures, and other information of concern to management personnel at DDN facilities.

The DDN Security Bulletin is distributed electronically by the DDN SCC, also under contract to DCA, as a means of communicating information on network and host security exposures, fixes, and concerns to security and management personnel at DDN facilities.

Anyone may join the mailing lists for these two bulletins by sending a message to NIC@NIC.DDN.MIL and asking to be placed on the mailing lists. These messages are also posted to the USENET newsgroup "ddn.mgt-bulletin." For additional information, see section 8.7.

3.9.7.5 System Administration List

The SYSADM-LIST is a list pertaining exclusively to UNIX system administration. Mail requests to be added to the list to SYSADM-LIST-REQUEST@SYSADMIN.COM.

3.9.7.6 Vendor Specific System Lists

The SUN-SPOTS and SUN-MANAGERS lists are discussion groups for users and administrators of systems supplied by Sun Microsystems. SUN-SPOTS is a fairly general list, discussing everything from hardware configurations to simple UNIX questions. To subscribe, send a message to SUN-SPOTS- REQUEST@RICE.EDU. This list is also available in the USENET newsgroup "comp.sys.sun." SUN-MANAGERS is a discussion list for Sun system administrators and covers all aspects of Sun system administration. To subscribe, send a message to SUN-MANAGERS-REQUEST@EECS.NWU.EDU.

The APOLLO list discusses the HP/Apollo system and its software. To subscribe, send a message to APOLLO- REQUEST@UMIX.CC.UMICH.EDU. APOLLO-L is a similar list which can be subscribed to by sending SUB APOLLO-L your full name to LISTSERV%UMRVMB.BITNET@VM1.NODAK.EDU. HPMINI-L pertains to the Hewlett-Packard 9000 series and HP/UX operating system. To subscribe, send SUB HPMINI-L your full name to LISTSERV%UAFSYSB.BITNET@VM1.NODAK.EDU. INFO-IBMPC discusses IBM PCs and compatibles, as well as MS- DOS. To subscribe, send a note to INFO-IBMPC-REQUEST@WSMR- SIMTEL20.ARMY.MIL.

There are numerous other mailing lists for nearly every popular computer or workstation in use today. For a complete list, obtain the file "netinfo/interest-groups" via anonymous FTP from the host FTP.NISC.SRI.COM.

3.9.7.7 Professional Societies and Journals

The IEEE Technical Committee on Security & Privacy publishes a quarterly magazine, "CIPHER."

IEEE Computer Society
1730 Massachusetts Ave. N.W.
Washington, DC 2036-1903

The ACM SigSAC (Special Interest Group on Security, Audit, and Controls) publishes a quarterly magazine, "SIGSAC Review."

Association for Computing Machinery
11 West 42nd St.
New York, NY 10036

The Information Systems Security Association publishes a quarterly magazine called ISSA Access."

Information Systems Security Association
P.O. Box 9457
Newport Beach, CA 92658

Computers and Security" is an "international journal for the professional involved with computer security, audit and control, and data integrity."

$266/year, 8 issues (1990)

Elsevier Advanced Technology
Journal Information Center
655 Avenue of the Americas
New York, NY 10010

The Data Security Letter" is published "to help data security professionals by providing inside information and knowledgable analysis of developments in computer and communications security."

$690/year, 9 issues (1990)

Data Security Letter
P.O. Box 1593
Palo Alto, CA 94302

3.9.8 Problem Reporting Tools

3.9.8.1 Auditing

Auditing is an important tool that can be used to enhance the security of your installation. Not only does it give you a means of identifying who has accessed your system (and may have done something to it) but it also gives you an indication of how your system is being used (or abused) by authorized users and attackers alike. In addition, the audit trail traditionally kept by computer systems can become an invaluable piece of evidence should your system be penetrated.

3.9.8.1.1 Verify Security

An audit trail shows how the system is being used from day to day. Depending upon how your site audit log is configured, your log files should show a range of access attempts that can show what normal system usage should look like. Deviation from that normal usage could be the result of penetration from an outside source using an old or stale user account. Observing a deviation in logins, for example, could be your first indication that something unusual is happening.

3.9.8.1.2 Verify Software Configurations

One of the ruses used by attackers to gain access to a system is by the insertion of a so-called Trojan Horse program. A Trojan Horse program can be a program that does something useful, or merely something interesting. It always does something unexpected, like steal passwords or copy files without your knowledge [25]. Imagine a Trojan login program that prompts for username and password in the usual way, but also writes that information to a special file that the attacker can come back and read at will. Imagine a Trojan Editor program that, despite the file permissions you have given your files, makes copies of everything in your directory space without you knowing about it.

This points out the need for configuration management of the software that runs on a system, not as it is being developed, but as it is in actual operation. Techniques for doing this range from checking each command every time it is executed against some criterion (such as a cryptoseal, described above) or merely checking the date and time stamp of the executable. Another technique might be to check each command in batch mode at midnight.

3.9.8.2 Tools

COPS is a security tool for system administrators that checks for numerous common security problems on UNIX systems [27]. COPS is a collection of shell scripts and C programs that can easily be run on almost any UNIX variant. Among other things, it checks the following items and sends the results to the system administrator:

- Checks “/dev/kmem” and other devices for world read/writability.

- Checks special or important files and directories for "bad" modes (world writable, etc.).
- Checks for easily-guessed passwords.
- Checks for duplicate user ids, invalid fields in the password file, etc..
- Checks for duplicate group ids, invalid fields in the group file, etc..
- Checks all users' home directories and their ".cshrc," ".login," ".profile," and ".rhosts" files for security problems.
- Checks all commands in the "/etc/rc" files and "cron" files for world writability.
- Checks for bad "root" paths, NFS file systems exported to the world, etc..
- Includes an expert system that checks to see if a given user (usually "root") can be compromised, given that certain rules are true.
- Checks for changes in the setuid status of programs on the system.

The COPS package is available from the "comp.sources.unix" archive on "ftp.uu.net," and also from the UNIX-SW repository on the MILNET host "wsmr-simtel20.army.mil."

3.9.9 Communication Among Administrators

3.9.9.1 Secure Operating Systems

The following list of products and vendors is adapted from the National Computer Security Center's (NCSC) Evaluated Products List. They represent those companies who have either received an evaluation from the NCSC or are in the process of a product evaluation. This list is not complete, but it is representative of those operating systems and add on components available in the commercial marketplace.

For a more detailed listing of the current products appearing in the NCSC EPL, contact the NCSC at:

National Computer Security Center
9800 Savage Road
Fort George G. Meade, MD 20755-6000
(301) 859-4458

```
Version Evaluation
Evaluated Product Vendor Evaluated Class
------------------------------------------------------------------------
Secure Communications Honeywell Information 2.1 A1
Processor (SCOMP) Systems, Inc.
Multics Honeywell Information MR11.0 B2
Systems, Inc.
```

```
System V/MLS 1.1.2 on UNIX AT&T 1.1.2 B1
System V 3.1.1 on AT&T 3B2/500and 3B2/600
OS 1100 Unisys Corp. Security B1
Release 1
MPE V/E Hewlett-Packard Computer G.03.04 C2
Systems Division
AOS/VS on MV/ECLIPSE series Data General Corp. 7.60 C2
VM/SP or VM/SP HPO with CMS, IBM Corp. 5 C2
RACF, DIRMAINT, VMTAPE-MS,
ISPF
MVS/XA with RACF IBM Corp. 2.2,2.3 C2
AX/VMS Digital Equipment Corp. 4.3 C2
NOS Control Data Corp. NOS
Security C2
Eval Product
TOP SECRET CGA Software Products 3.0/163 C2
Group, Inc.
Access Control Facility 2 SKK, Inc. 3.1.3 C2
UTX/32S Gould, Inc. Computer 1.0 C2
Systems Division
A Series MCP/AS with Unisys Corp. 3.7 C2
InfoGuard Security
Enhancements
Primos Prime Computer, Inc. 21.0.1DODC2A C2
Resource Access Control IBM Corp. 1.5 C1
Facility (RACF)


Version Candidate
Candidate Product Vendor Evaluated Class
------------------------------------------------------------------------
Boeing MLS LAN Boeing Aerospace A1 M1
Trusted XENIX Trusted Information
Systems, Inc. B2
VSLAN VERDIX Corp. B2
System V/MLS AT&T B1
VM/SP with RACF IBM Corp. 5/1.8.2 C2
Wang SVS/OS with CAP Wang Laboratories, Inc. 1.0 C2
```

3.9.9.2 Obtaining Fixes for Known Problems

It goes without saying that computer systems have bugs. Even operating systems, upon which we depend for protection of our data, have bugs. And since there are bugs, things can be broken, both maliciously and accidentally. It is important that whenever bugs are discovered, a should fix be identified and implemented as soon as possible. This should minimize any exposure caused by the bug in the first place.

A corollary to the bug problem is: from whom do I obtain the fixes? Most systems have some support from the manufacturer or supplier. Fixes coming from that source tend to be implemented quickly after receipt. Fixes for some problems are often posted on the network and are left to the system administrators to incorporate as they can. The problem is that one wants to

have faith that the fix will close the hole and not introduce any others. We will tend to trust that the manufacturer's fixes are better than those that are posted on the net.

3.9.9.3 Sun Customer Warning System

Sun Microsystems has established a Customer Warning System (CWS) for handling security incidents. This is a formal process which includes:

- Having a well advertised point of contact in Sun for reporting security problems.
- Pro-actively alerting customers of worms, viruses, or other security holes that could affect their systems.
- Distributing the patch (or work-around) as quickly as possible.

They have created an electronic mail address, SECURITY- ALERT@SUN.COM, which will enable customers to report security problems. A voice-mail backup is available at (415) 688-9081.

A "Security Contact" can be designated by each customer site; this person will be contacted by Sun in case of any new security problems. For more information, contact your Sun representative.

3.9.9.4 Trusted Archive Servers

Several sites on the Internet maintain large repositories of public-domain and freely distributable software, and make this material available for anonymous FTP. This section describes some of the larger repositories. Note that none of these servers implements secure checksums or anything else guaranteeing the integrity of their data. Thus, the notion of "trust" should be taken as a somewhat limited definition.

3.9.9.4.1 Sun Fixes on UUNET

Sun Microsystems has contracted with UUNET Communications Services, Inc., to make fixes for bugs in Sun software available via anonymous FTP. You can access these fixes by using the "ftp" command to connect to the host FTP.UU.NET. Then change into the directory "sundist/security," and obtain a directory listing. The file "README" contains a brief description of what each file in this directory contains, and what is required to install the fix.

3.9.9.4.2 Berkeley Fixes

The University of California at Berkeley also makes fixes available via anonymous FTP; these fixes pertain primarily to the current release of BSD UNIX (currently, release 4.3).

However, even if you are not running their software, these fixes are still important, since many vendors (Sun, DEC, Sequent, etc.) base their software on the Berkeley releases.

The Berkeley fixes are available for anonymous FTP from the host UCBARPA.BERKELEY.EDU in the directory "4.3/ucb-fixes." The file "INDEX" in this directory describes what each file contains. They are also available from UUNET (see section 3.9.9.4.3).

Berkeley also distributes new versions of "sendmail" and "named" from this machine. New versions of these commands are stored in the "4.3" directory, usually in the files "sendmail.tar.Z" and "bind.tar.Z," respectively.

3.9.9.4.3 Simtel-20 and UUNET

The two largest general-purpose software repositories on the Internet are the hosts WSMR-SIMTEL20.ARMY.MIL and FTP.UU.NET.

WSMR-SIMTEL20.ARMY.MIL is a TOPS-20 machine operated by the U.S. Army at White Sands Missile Range (WSMR), New Mexico. The directory "pd2:<unix-c>" contains a large amount of UNIX software, primarily taken from the "comp.sources" newsgroups. The directories "pd1:<msdos>" and "pd2:<msdos2>" contains software for IBM PC systems, and "pd3:<macintosh>" contains software for the Apple Macintosh.

FTP.UU.NET is operated by UUNET Communications Services, Inc. in Falls Church, Virginia. This company sells Internet and USENET access to sites all over the country (and internationally). The software posted to the following USENET source newsgroups is stored here, in directories of the same name:

comp.sources.games

comp.sources.misc

comp.sources.sun

comp.sources.unix

comp.sources.x

Numerous other distributions, such as all the freely distributable Berkeley UNIX source code, Internet Request for Comments (RFCs), and so on are also stored on this system.

3.9.9.4.4 Vendors

Many vendors make fixes for bugs in their software available electronically, either via mailing lists or via anonymous FTP. You should contact your vendor to find out if they offer this service, and if so, how to access it. Some vendors that offer these services include Sun Microsystems (see above), Digital Equipment Corporation (DEC), the University of California at Berkeley (see above), and Apple Computer [5, CURRY].

4. Types of Security Procedures

4.1 System Security Audits

Most businesses undergo some sort of annual financial auditing as a regular part of their business life. Security audits are an important part of running any computing environment. Part of the security audit should be a review of any policies that concern system security, as well as the mechanisms that are put in place to enforce them.

4.1.1 Organize Scheduled Drills

Although not something that would be done each day or week, scheduled drills may be conducted to determine if the procedures defined are adequate for the threat to be countered. If your major threat is one of natural disaster, then a drill would be conducted to verify your backup and recovery mechanisms. On the other hand, if your greatest threat is from external intruders attempting to penetrate your system, a drill might be conducted to actually try a penetration to observe the effect of the policies.

Drills are a valuable way to test that your policies and procedures are effective. On the other hand, drills can be time-consuming and disruptive to normal operations. It is important to weigh the benefits of the drills against the possible time loss which may be associated with them.

4.1.2 Test Procedures

If the choice is made to not to use scheduled drills to examine your entire security procedure at one time, it is important to test individual procedures frequently. Examine your backup procedure to make sure you can recover data from the tapes. Check log files to be sure that information which is supposed to be logged to them is being logged to them, etc.. When a security audit is mandated, great care should be used in devising tests of the security policy. It is important to clearly identify what is being tested, how the test will be conducted, and results expected from the test. This should all be documented and included in or as an adjunct to the security policy document itself.

It is important to test all aspects of the security policy, both procedural and automated, with a particular emphasis on the automated mechanisms used to enforce the policy. Tests should be defined to ensure a comprehensive examination of policy features, that is, if a test is defined to examine the user logon process, it should be explicitly stated that both valid and invalid user names and passwords will be used to demonstrate proper operation of the logon program.

Keep in mind that there is a limit to the reasonableness of tests. The purpose of testing is to ensure confidence that the security policy is being correctly enforced, and not to “prove” the absoluteness of the system or policy. The goal should be to obtain some assurance that the reasonable and credible controls imposed by your security policy are adequate.

4.2 Account Management Procedures

Procedures to manage accounts are important in preventing unauthorized access to your system. It is necessary to decide several things: Who may have an account on the system? How long may someone have an account without renewing his or her request? How do old accounts get removed from the system? The answers to all these questions should be explicitly set out in the policy.

In addition to deciding who may use a system, it may be important to determine what each user may use the system for (is personal use allowed, for example). If you are connected to an outside network, your site or the network management may have rules about what the network may be used for. Therefore, it is important for any security policy to define an adequate account management procedure for both administrators and users. Typically, the system administrator would be responsible for creating and deleting user accounts and generally maintaining overall control of system use. To some degree, account management is also the responsibility of each system user in the sense that the user should observe any system messages and events that may be indicative of a policy violation. For example, a message at logon that indicates the date and time of the last logon should be reported by the user if it indicates an unreasonable time of last logon.

4.3 Password Management Procedures

A policy on password management may be important if your site wishes to enforce secure passwords. These procedures may range from asking or forcing users to change their passwords occasionally to actively attempting to break users' passwords and then informing the user of how easy it was to do. Another part of password management policy covers who may distribute passwords - can users give their passwords to other users?

Section 2.3 discusses some of the policy issues that need to be decided for proper password management. Regardless of the policies, password management procedures need to be carefully setup to avoid disclosing passwords. The choice of initial passwords for accounts is critical. In some cases, users may never login to activate an account; thus, the choice of the initial password should not be easily guessed. Default passwords should never be assigned to accounts: always create new passwords for each user. If there are any printed lists of passwords, these should be kept off-line in secure locations; better yet, don't list passwords.

4.3.1 Password Selection

Perhaps the most vulnerable part of any computer system is the account password. Any computer system, no matter how secure it is from network or dial-up attack, Trojan horse programs, and so on, can be fully exploited by an intruder if he or she can gain access via a poorly chosen password. It is important to define a good set of rules for password selection, and distribute these rules to all users. If possible, the software which sets user passwords should be modified to enforce as many of the rules as possible.

A sample set of guidelines for password selection is shown below:

- DON'T use your login name in any form (as-is, reversed, capitalized, doubled, etc.).
- DON'T use your first, middle, or last name in any form.
- DON'T use your spouse's or child's name.
- DON'T use other information easily obtained about you. This includes license plate numbers, telephone numbers, social security numbers, the make of your automobile, the name of the street you live on, etc..
- DON'T use a password of all digits, or all the same letter.
- DON'T use a word contained in English or foreign language dictionaries, spelling lists, or other lists of words.
- DON'T use a password shorter than six characters.
- DO use a password with mixed-case alphabetics.
- DO use a password with non-alphabetic characters (digits or punctuation).
- DO use a password that is easy to remember, so you don't have to write it down.
- DO use a password that you can type quickly, without having to look at the keyboard.

Methods of selecting a password which adheres to these guidelines include:

- Choose a line or two from a song or poem, and use the first letter of each word.
- Alternate between one consonant and one or two vowels, up to seven or eight characters. This provides nonsense words which are usually pronounceable, and thus easily remembered.
- Choose two short words and concatenate them together with a punctuation character between them.

Users should also be told to change their password periodically, usually every three to six months. This makes sure that an intruder who has guessed a password will eventually lose access, as well as invalidating any list of passwords he/she may have obtained. Many systems enable the system administrator to force users to change their passwords after an expiration period; this software should be enabled if your system supports it [5, CURRY].

Some systems provide software which forces users to change their passwords on a regular basis. Many of these systems also include password generators which provide the user with a set of passwords to choose from. The user is not permitted to make up his or her own password.

There are arguments both for and against systems such as these. On the one hand, by using generated passwords, users are prevented from selecting insecure passwords. On the other hand, unless the generator is good at making up easy to remember passwords, users will begin writing them down in order to remember them.

4.3.2 Procedures for Changing Passwords

How password changes are handled is important to keeping passwords secure. Ideally, users should be able to change their own passwords on-line. (Note that password changing programs are a favorite target of intruders. See section 4.4 on configuration management for further information.)

However, there are exception cases which must be handled carefully. Users may forget passwords and not be able to get onto the system. The standard procedure is to assign the user a new password. Care should be taken to make sure that the real person is requesting the change and gets the new password. One common trick used by intruders is to call or message to a system administrator and request a new password. Some external form of verification should be used before the password is assigned. At some sites, users are required to show up in person with ID.

There may also be times when many passwords need to be changed. If a system is compromised by an intruder, the intruder may be able to steal a password file and take it off the system. Under these circumstances, one course of action is to change all passwords on the system. Your site should have procedures for how this can be done quickly and efficiently. What course you choose may depend on the urgency of the problem. In the case of a known attack with damage, you may choose to forcibly disable all accounts and assign users new passwords before they come back onto the system. In some places, users are sent a message telling them that they should change their passwords, perhaps within a certain time period. If the password isn't changed before the time period expires, the account is locked.

Users should be aware of what the standard procedure is for passwords when a security event has occurred. One well-known spoof reported by the Computer Emergency Response Team (CERT) involved messages sent to users, supposedly from local system administrators, requesting them to immediately change their password to a new value provided in the message [24]. These messages were not from the administrators, but from intruders trying to steal accounts. Users should be warned to immediately report any suspicious requests such as this to site administrators.

4.4 Configuration Management Procedures

Configuration management is generally applied to the software development process. However, it is certainly applicable in a operational sense as well. Consider that the since many of the system level programs are intended to enforce the security policy, it is important that these be "known" as correct. That is, one should not allow system level programs (such as the

operating system, etc.) to be changed arbitrarily. At very least, the procedures should state who is authorized to make changes to systems, under what circumstances, and how the changes should be documented.

In some environments, configuration management is also desirable as applied to physical configuration of equipment. Maintaining valid and authorized hardware configuration should be given due consideration in your security policy.

4.4.1 Non-Standard Configurations

Occasionally, it may be beneficial to have a slightly non-standard configuration in order to thwart the "standard" attacks used by some intruders. The non-standard parts of the configuration might include different password encryption algorithms, different configuration file locations, and rewritten or functionally limited system commands.

Non-standard configurations, however, also have their drawbacks. By changing the "standard" system, these modifications make software maintenance more difficult by requiring extra documentation to be written, software modification after operating system upgrades, and, usually, someone with special knowledge of the changes.

Because of the drawbacks of non-standard configurations, they are often only used in environments with a "firewall" machine (see section 3.9.1). The firewall machine is modified in non-standard ways since it is susceptible to attack, while internal systems behind the firewall are left in their standard configurations.

5. Incident Handling

5.1 Overview

This section of the document will supply some guidance to be applied when a computer security event is in progress on a machine, network, site, or multi-site environment. The operative philosophy in the event of a breach of computer security, whether it be an external intruder attack or a disgruntled employee, is to plan for adverse events in advance. There is no substitute for creating contingency plans for the types of events described above.

Traditional computer security, while quite important in the overall site security plan, usually falls heavily on protecting systems from attack, and perhaps monitoring systems to detect attacks. Little attention is usually paid for how to actually handle the attack when it occurs. The result is that when an attack is in progress, many decisions are made in haste and can be damaging to tracking down the source of the incident, collecting evidence to be used in prosecution efforts, preparing for the recovery of the system, and protecting the valuable data contained on the system.

5.1.1 Have a Plan to Follow in Case of an Incident

Part of handling an incident is being prepared to respond before the incident occurs. This includes establishing a suitable level of protections, so that if the incident becomes severe, the damage which can occur is limited. Protection includes preparing incident handling guidelines or a contingency response plan for your organization or site. Having written plans eliminates much of the ambiguity which occurs during an incident, and will lead to a more appropriate and thorough set of responses. Second, part of protection is preparing a method of notification, so you will know who to call and the relevant phone numbers. It is important, for example, to conduct "dry runs," in which your computer security personnel, system administrators, and managers simulate handling an incident.

Learning to respond efficiently to an incident is important for numerous reasons. The most important benefit is directly to human beings—preventing loss of human life. Some computing systems are life critical systems, systems on which human life depends (e.g., by controlling some aspect of life-support in a hospital or assisting air traffic controllers).

An important but often overlooked benefit is an economic one. Having both technical and managerial personnel respond to an incident requires considerable resources, resources which could be utilized more profitably if an incident did not require their services. If these personnel are trained to handle an incident efficiently, less of their time is required to deal with that incident.

A third benefit is protecting classified, sensitive, or proprietary information. One of the major dangers of a computer security incident is that information may be irrecoverable. Efficient incident handling minimizes this danger. When classified information is involved, other government regulations may apply and must be integrated into any plan for incident handling.

A fourth benefit is related to public relations. News about computer security incidents tends to be damaging to an organization's stature among current or potential clients. Efficient incident handling minimizes the potential for negative exposure.

A final benefit of efficient incident handling is related to legal issues. It is possible that in the near future organizations may be sued because one of their nodes was used to launch a network attack. In a similar vein, people who develop patches or workarounds may be sued if the patches or workarounds are ineffective, resulting in damage to systems, or if the patches or workarounds themselves damage systems. Knowing about operating system vulnerabilities and patterns of attacks and then taking appropriate measures is critical to circumventing possible legal problems.

5.1.2 Order of Discussion in this Session Suggests an Order for a Plan

This chapter is arranged such that a list may be generated from the Table of Contents to provide a starting point for creating a policy for handling ongoing incidents. The main points to be included in a policy for handling incidents are:

- Overview (what are the goals and objectives in handling the incident).
- Evaluation (how serious is the incident).
- Notification (who should be notified about the incident).
- Response (what should the response to the incident be).
- Legal/Investigative (what are the legal and prosecutorial implications of the incident).
- Documentation Logs (what records should be kept from before, during, and after the incident).

Each of these points is important in an overall plan for handling incidents. The remainder of this chapter will detail the issues involved in each of these topics, and provide some guidance as to what should be included in a site policy for handling incidents.

5.1.3 Possible Goals and Incentives for Efficient Incident Handling

As in any set of pre-planned procedures, attention must be placed on a set of goals to be obtained in handling an incident. These goals will be placed in order of importance depending on the site, but one such set of goals might be:

- Assure integrity of (life) critical systems.
- Maintain and restore data.
- Maintain and restore service.
- Figure out how it happened.
- Avoid escalation and further incidents.
- Avoid negative publicity.
- Find out who did it.
- Punish the attackers.

It is important to prioritize actions to be taken during an incident well in advance of the time an incident occurs. Sometimes an incident may be so complex that it is impossible to do everything at once to respond to it; priorities are essential. Although priorities will vary from institution-to-institution, the following suggested priorities serve as a starting point for defining an organization's response:

- Priority one—protect human life and people's safety; human life always has precedence over all other considerations.
- Priority two—protect classified and/or sensitive data (as regulated by your site or by government regulations).
- Priority three—protect other data, including proprietary, scientific, managerial and other data, because loss of data is costly in terms of resources.
- Priority four—prevent damage to systems (e.g., loss or alteration of system files, damage to disk drives, etc.); damage to systems can result in costly down time and recovery.
- Priority five—minimize disruption of computing resources; it is better in many cases to shut a system down or disconnect from a network than to risk damage to data or systems.

An important implication for defining priorities is that once human life and national security considerations have been addressed, it is generally more important to save data than system software and hardware. Although it is undesirable to have any damage or loss during an incident, systems can be replaced; the loss or compromise of data (especially classified data), however, is usually not an acceptable outcome under any circumstances. Part of handling an incident is being prepared to respond before the incident occurs. This includes establishing a suitable level of protections so that if the incident becomes severe, the damage which can occur is limited. Protection includes preparing incident handling guidelines or a contingency response plan for your organization or site. Written plans eliminate much of the ambiguity which occurs during an incident, and will lead to a more appropriate and thorough set of responses. Second, part of protection is preparing a method of notification so you will know who to call and how to contact them. For example, every member of the Department of Energy's CIAC Team carries a card with every other team member's work and home phone numbers, as well as pager numbers. Third, your organization or site should establish backup procedures for every machine and system. Having backups eliminates much of the threat of even a severe incident, since backups preclude serious data loss. Fourth, you should set up secure systems. This involves eliminating vulnerabilities, establishing an effective password policy, and other procedures, all of which will be explained later in this document. Finally, conducting training activities is part of protection. It is important, for example, to conduct "dry runs," in which your computer security personnel, system administrators, and managers simulate handling an incident.

5.1.4 Local Policies and Regulations Providing Guidance

Any plan for responding to security incidents should be guided by local policies and regulations. Government and private sites that deal with classified material have specific rules that they must follow.

The policies your site makes about how it responds to incidents (as discussed in sections 2.4 and 2.5) will shape your response. For example, it may make little sense to create mechanisms

to monitor and trace intruders if your site does not plan to take action against the intruders if they are caught. Other organizations may have policies that affect your plans. Telephone companies often release information about telephone traces only to law enforcement agencies.

Section 5.5 also notes that if any legal action is planned, there are specific guidelines that must be followed to make sure that any information collected can be used as evidence.

5.2 Evaluation

5.2.1 Is It Real?

This stage involves determining the exact problem. Of course many, if not most, signs often associated with virus infections, system intrusions, etc., are simply anomalies such as hardware failures. To assist in identifying whether there really is an incident, it is usually helpful to obtain and use any detection software which may be available. For example, widely available software packages can greatly assist someone who thinks there may be a virus in a Macintosh computer. Audit information is also extremely useful, especially in determining whether there is a network attack. It is extremely important to obtain a system snapshot as soon as one suspects that something is wrong. Many incidents cause a dynamic chain of events to occur, and an initial system snapshot may do more good in identifying the problem and any source of attack than most other actions which can be taken at this stage. Finally, it is important to start a log book.

Recording system events, telephone conversations, time stamps, etc., can lead to a more rapid and systematic identification of the problem, and is the basis for subsequent stages of incident handling. There are certain indications or "symptoms" of an incident which deserve special attention:

- System crashes.
- New user accounts (e.g., the account RUMPLESTILTSKIN has unexplainedly been created), or high activity on an account that has had virtually no activity for months.
- New files (usually with novel or strange file names, such as data.xx or k).
- Accounting discrepancies (e.g., in a UNIX system you might notice that the accounting file called /usr/admin/lastlog has shrunk, something that should make you very suspicious that there may be an intruder).
- Changes in file lengths or dates (e.g., a user should be suspicious if he/she observes that the .EXE files in an MS DOS computer have unexplainedly grown by over 1800 bytes).
- Attempts to write to system (e.g., a system manager notices that a privileged user in a VMS system is attempting to alter RIGHTSLIST.DAT).
- Data modification or deletion (e.g., files start to disappear).

- Denial of service (e.g., a system manager and all other users become locked out of a UNIX system, which has been changed to single user mode).
- Unexplained, poor system performance (e.g., system response time becomes unusually slow).
- Anomalies (e.g., "GOTCHA" is displayed on a display terminal or there are frequent unexplained "beeps").
- Suspicious probes (e.g., there are numerous unsuccessful login attempts from another node).
- Suspicious browsing (e.g., someone becomes a root user on a UNIX system and accesses file after file in one user's account, then another's).

None of these indications is absolute "proof" that an incident is occurring, nor are all of these indications normally observed when an incident occurs. If you observe any of these indications, however, it is important to suspect that an incident might be occurring, and act accordingly. There is no formula for determining with 100 percent accuracy that an incident is occurring (possible exception: when a virus detection package indicates that your machine has the nVIR virus and you confirm this by examining contents of the nVIR resource in your Macintosh computer, you can be very certain that your machine is infected).

It is best at this point to collaborate with other technical and computer security personnel to make a decision as a group about whether an incident is occurring.

5.2.2 Scope

Along with the identification of the incident is the evaluation of the scope and impact of the problem. It is important to correctly identify the boundaries of the incident in order to effectively deal with it. In addition, the impact of an incident will determine its priority in allocating resources to deal with the event. Without an indication of the scope and impact of the event, it is difficult to determine a correct response.

In order to identify the scope and impact, a set of criteria should be defined which is appropriate to the site and to the type of connections available. Some of the issues are:

- Is this a multi-site incident?
- Are many computers at your site effected by this incident?
- Is sensitive information involved?
- What is the entry point of the incident (network, phone line, local terminal, etc.)?
- Is the press involved?
- What is the potential damage of the incident?

- What is the estimated time to close out the incident?
- What resources could be required to handle the incident?

5.3 Possible Types of Notification

When you have confirmed that an incident is occurring, the appropriate personnel must be notified. Who and how this notification is achieved is very important in keeping the event under control both from a technical and emotional standpoint.

5.3.1 Explicit

First of all, any notification to either local or off-site personnel must be explicit. This requires that any statement (be it an electronic mail message, phone call, or fax) provides information about the incident that is clear, concise, and fully qualified. When you are notifying others that will help you to handle an event, a "smoke screen" will only divide the effort and create confusion. If a division of labor is suggested, it is helpful to provide information to each section about what is being accomplished in other efforts. This will not only reduce duplication of effort, but allow people working on parts of the problem to know where to obtain other information that would help them resolve a part of the incident.

5.3.2 Factual

Another important consideration when communicating about the incident is to be factual. Attempting to hide aspects of the incident by providing false or incomplete information may not only prevent a successful resolution to the incident, but may even worsen the situation. This is especially true when the press is involved. When an incident severe enough to gain press attention is ongoing, it is likely that any false information you provide will not be substantiated by other sources. This will reflect badly on the site and may create enough ill-will between the site and the press to damage the site's public relations.

5.3.3 Choice of Language

The choice of language used when notifying people about the incident can have a profound effect on the way that information is received. When you use emotional or inflammatory terms, you raise the expectations of damage and negative outcomes of the incident. It is important to remain calm both in written and spoken notifications.

Another issue associated with the choice of language is the notification to non-technical or off-site personnel. It is important to accurately describe the incident without undue alarm or confusing messages. While it is more difficult to describe the incident to a non-technical audience, it is often more important.

A non-technical description may be required for upper-level management, the press, or law enforcement liaisons. The importance of these notifications cannot be underestimated and may

make the difference between handling the incident properly and escalating to some higher level of damage.

5.3.4 Notification of Individuals

- Point of Contact (POC) people (Technical, Administrative, Response Teams, Investigative, Legal, Vendors, Service providers), and which POCs are visible to whom.
- Wider community (users).
- Other sites that might be affected.

Finally, there is the question of who should be notified during and after the incident. There are several classes of individuals that need to be considered for notification. These are the technical personnel, administration, appropriate response teams (such as CERT or CIAC), law enforcement, vendors, and other service providers. These issues are important for the central point of contact, since that is the person responsible for the actual notification of others (see section 5.3.6 for further information). A list of people in each of these categories is an important time saver for the POC during an incident. It is much more difficult to find an appropriate person during an incident when many urgent events are ongoing.

In addition to the people responsible for handling part of the incident, there may be other sites affected by the incident (or perhaps simply at risk from the incident). A wider community of users may also benefit from knowledge of the incident. Often, a report of the incident once it is closed out is appropriate for publication to the wider user community.

5.3.5 Public Relations—Press Releases

One of the most important issues to consider is when, who, and how much to release to the general public through the press. There are many issues to consider when deciding this particular issue.

First and foremost, if a public relations office exists for the site, it is important to use this office as liaison to the press.

The public relations office is trained in the type and wording of information released, and will help to assure that the image of the site is protected during and after the incident (if possible).

A public relations office has the advantage that you can communicate candidly with them, and provide a buffer between the constant press attention and the need of the POC to maintain control over the incident.

If a public relations office is not available, the information released to the press must be carefully considered. If the information is sensitive, it may be advantageous to provide only minimal or overview information to the press. It is quite possible that any information provided to the press will be quickly reviewed by the perpetrator of the incident. As a contrast

to this consideration, it was discussed above that misleading the press can often backfire and cause more damage than releasing sensitive information.

While it is difficult to determine in advance what level of detail to provide to the press, some guidelines to keep in mind are:

- Keep the technical level of detail low. Detailed information about the incident may provide enough information for copy-cat events or even damage the site's ability to prosecute once the event is over.
- Keep the speculation out of press statements. Speculation of who is causing the incident or the motives are very likely to be in error and may cause an inflamed view of the incident.
- Work with law enforcement professionals to assure that evidence is protected. If prosecution is involved, assure that the evidence collected is not divulged to the press.
- Try not to be forced into a press interview before you are prepared. The popular press is famous for the "2am" interview, where the hope is to catch the interviewee off guard and obtain information otherwise not available.
- Do not allow the press attention to detract from the handling of the event. Always remember that the successful closure of an incident is of primary importance.

5.3.6 Who Needs to Get Involved?

There now exists a number of incident response teams (IRTs) such as the CERT and the CIAC. (See sections 3.9.7.3.1 and 3.9.7.3.4.) Teams exists for many major government agencies and large corporations. If such a team is available for your site, the notification of this team should be of primary importance during the early stages of an incident. These teams are responsible for coordinating computer security incidents over a range of sites and larger entities. Even if the incident is believed to be contained to a single site, it is possible that the information available through a response team could help in closing out the incident.

In setting up a site policy for incident handling, it may be desirable to create an incident handling team (IHT), much like those teams that already exist, that will be responsible for handling computer security incidents for the site (or organization). If such a team is created, it is essential that communication lines be opened between this team and other IHTs.

Once an incident is under way, it is difficult to open a trusted dialogue between other IHTs if none has existed before.

5.4 Response

A major topic still untouched here is how to actually respond to an event. The response to an event will fall into the general categories of containment, eradication, recovery, and follow-up.

Containment

The purpose of containment is to limit the extent of an attack. For example, it is important to limit the spread of a worm attack on a network as quickly as possible. An essential part of containment is decision making (i.e., determining whether to shut a system down, to disconnect from a network, to monitor system or network activity, to set traps, to disable functions such as remote file transfer on a UNIX system, etc.). Sometimes this decision is trivial; shut the system down if the system is classified or sensitive, or if proprietary information is at risk!

In other cases, it is worthwhile to risk having some damage to the system if keeping the system up might enable you to identify an intruder.

The third stage, containment, should involve carrying out predetermined procedures. Your organization or site should, for example, define acceptable risks in dealing with an incident, and should prescribe specific actions and strategies accordingly.

Finally, notification of cognizant authorities should occur during this stage.

Eradication

Once an incident has been detected, it is important to first think about containing the incident. Once the incident has been contained, it is now time to eradicate the cause. Software may be available to help you in this effort. For example, eradication software is available to eliminate most viruses which infect small systems. If any bogus files have been created, it is time to delete them at this point. In the case of virus infections, it is important to clean and reformat any disks containing infected files. Finally, ensure that all backups are clean. Many systems infected with viruses become periodically reinfected simply because people do not systematically eradicate the virus from backups.

Recovery

Once the cause of an incident has been eradicated, the recovery phase defines the next stage of action. The goal of recovery is to return the system to normal. In the case of a network-based attack, it is important to install patches for any operating system vulnerability which was exploited.

Follow-up

One of the most important stages of responding to incidents is also the most often omitted—the follow-up stage. This stage is important because it helps those involved in handling the incident develop a set of "lessons learned" (see section 6.3) to improve future performance in such situations. This stage also provides information which justifies an organization's computer security effort to management, and yields information which may be essential in legal proceedings.

The most important element of the follow-up stage is performing a postmortem analysis. Exactly what happened, and at what times?

How well did the staff involved with the incident perform? What kind of information did the staff need quickly, and how could they have gotten that information as soon as possible? What would the staff do differently next time? A follow-up report is valuable because it provides a reference to be used in case of other similar incidents. Creating a formal chronology of events (including time stamps) is also important for legal reasons. Similarly, it is also important to as quickly obtain a monetary estimate of the amount of damage the incident caused in terms of any loss of software and files, hardware damage, and manpower costs to restore altered files, reconfigure affected systems, and so forth. This estimate may become the basis for subsequent prosecution activity by the FBI, the U.S. Attorney General's Office, etc..

5.4.1 What Will You Do?

- Restore control.
- Relation to policy.
- Which level of service is needed?
- Monitor activity.
- Constrain or shut down system.

5.4.2 Consider Designating a "Single Point of Contact"

When an incident is under way, a major issue is deciding who is in charge of coordinating the activity of the multitude of players.

A major mistake that can be made is to have a number of "points of contact" (POC) that are not pulling their efforts together. This will only add to the confusion of the event, and will probably lead to additional confusion and wasted or ineffective effort.

The single point of contact may or may not be the person "in charge" of the incident. There are two distinct rolls to fill when deciding who shall be the point of contact and the person in charge of the incident. The person in charge will make decisions as to the interpretation of policy applied to the event. The responsibility for the handling of the event falls onto this person. In contrast, the point of contact must coordinate the effort of all the parties involved with handling the event. The point of contact must be a person with the technical expertise to successfully coordinate the effort of the system managers and users involved in monitoring and reacting to the attack. Often the management structure of a site is such that the administrator of a set of resources is not a technically competent person with regard to handling the details of the operations of the computers, but is ultimately responsible for the use of these resources.

Another important function of the POC is to maintain contact with law enforcement and other external agencies (such as the CIA, DoD, U.S. Army, or others) to assure that multi-agency involvement occurs.

Finally, if legal action in the form of prosecution is involved, the POC may be able to speak for the site in court. The alternative is to have multiple witnesses that will be hard to coordinate in a legal sense, and will weaken any case against the attackers. A single POC may also be the single person in charge of evidence collected, which will keep the number of people accounting for evidence to a minimum. As a rule of thumb, the more people that touch a potential piece of evidence, the greater the possibility that it will be inadmissible in court. The section below (Legal/Investigative) will provide more details for consideration on this topic.

5.5 Legal/Investigative

5.5.1 Establishing Contacts with Investigative Agencies

It is important to establish contacts with personnel from investigative agencies such as the FBI and Secret Service as soon as possible, for several reasons. Local law enforcement and local security offices or campus police organizations should also be informed when appropriate. A primary reason is that once a major attack is in progress, there is little time to call various personnel in these agencies to determine exactly who the correct point of contact is. Another reason is that it is important to cooperate with these agencies in a manner that will foster a good working relationship, and that will be in accordance with the working procedures of these agencies. Knowing the working procedures in advance and the expectations of your point of contact is a big step in this direction. For example, it is important to gather evidence that will be admissible in a court of law. If you don't know in advance how to gather admissible evidence, your efforts to collect evidence during an incident are likely to be of no value to the investigative agency with which you deal. A final reason for establishing contacts as soon as possible is that it is impossible to know the particular agency that will assume jurisdiction in any given incident. Making contacts and finding the proper channels early will make responding to an incident go considerably more smoothly. If your organization or site has a legal counsel, you need to notify this office soon after you learn that an incident is in progress. At a minimum, your legal counsel needs to be involved to protect the legal and financial interests of your site or organization. There are many legal and practical issues, a few of which are:

1. Whether your site or organization is willing to risk negative publicity or exposure to cooperate with legal prosecution efforts.

2. Downstream liability—if you leave a compromised system as is so it can be monitored and another computer is damaged because the attack originated from your system, your site or organization may be liable for damages incurred.

3. Distribution of information—if your site or organization distributes information about an attack in which another site or organization may be involved or the vulnerability in a product that may affect ability to market that product, your site or organization may again be liable for any damages (including damage of reputation).

4. Liabilities due to monitoring—your site or organization may be sued if users at your site or elsewhere discover that your site is monitoring account activity without informing users.

Unfortunately, there are no clear precedents yet on the liabilities or responsibilities of organizations involved in a security incident or who might be involved in supporting an investigative effort. Investigators will often encourage organizations to help trace and monitor intruders—indeed, most investigators cannot pursue computer intrusions without extensive support from the organizations involved. However, investigators cannot provide protection from liability claims, and these kinds of efforts may drag out for months and may take lots of effort.

On the other side, an organization's legal council may advise extreme caution and suggest that tracing activities be halted and an intruder shut out of the system. This in itself may not provide protection from liability, and may prevent investigators from identifying anyone.

The balance between supporting investigative activity and limiting liability is tricky; you'll need to consider the advice of your council and the damage the intruder is causing (if any) in making your decision about what to do during any particular incident.

Your legal counsel should also be involved in any decision to contact investigative agencies when an incident occurs at your site. The decision to coordinate efforts with investigative agencies is most properly that of your site or organization.

Involving your legal counsel will also foster the multi-level coordination between your site and the particular investigative agency involved which in turn results in an efficient division of labor. Another result is that you are likely to obtain guidance that will help you avoid future legal mistakes.

Finally, your legal counsel should evaluate your site's written procedures for responding to incidents. It is essential to obtain a "clean bill of health" from a legal perspective before you actually carry out these procedures.

5.5.2 Formal and Informal Legal Procedures

One of the most important considerations in dealing with investigative agencies is verifying that the person who calls asking for information is a legitimate representative from the agency in question. Unfortunately, many well intentioned people have unknowingly leaked sensitive information about incidents, allowed unauthorized people into their systems, etc., because a caller has masqueraded as an FBI or Secret Service agent. A similar consideration is using a secure means of communication.

Because many network attackers can easily reroute electronic mail, avoid using electronic mail to communicate with other agencies (as well as others dealing with the incident at hand). Non-secured phone lines (e.g., the phones normally used in the business world) are also frequent targets for tapping by network intruders, so be careful!

There is no established set of rules for responding to an incident when the U.S. Federal Government becomes involved. Except by court order, no agency can force you to monitor, to disconnect from the network, to avoid telephone contact with the suspected attackers, etc.. As discussed in section 5.5.1, you should consult the matter with your legal counsel, especially before taking an action that your organization has never taken. The particular agency involved may ask you to leave an attacked machine on and to monitor activity on this machine, for example.

Your complying with this request will ensure continued cooperation of the agency—usually the best route towards finding the source of the network attacks and, ultimately, terminating these attacks.

Additionally, you may need some information or a favor from the agency involved in the incident. You are likely to get what you need only if you have been cooperative. Of particular importance is avoiding unnecessary or unauthorized disclosure of information about the incident, including any information furnished by the agency involved. The trust between your site and the agency hinges upon your ability to avoid compromising the case the agency will build; keeping "tight lipped" is imperative.

Sometimes your needs and the needs of an investigative agency will differ. Your site may want to get back to normal business by closing an attack route, but the investigative agency may want you to keep this route open. Similarly, your site may want to close a compromised system down to avoid the possibility of negative publicity, but again the investigative agency may want you to continue monitoring. When there is such a conflict, there may be a complex set of tradeoffs (e.g., interests of your site's management, amount of resources you can devote to the problem, jurisdictional boundaries, etc.). An important guiding principle is related to what might be called "Internet citizenship" [22, IAB89, 23] and its responsibilities. Your site can shut a system down, and this will relieve you of the stress, resource demands, and danger of negative exposure. The attacker, however, is likely to simply move on to another system, temporarily leaving others blind to the attacker's intention and actions until another path of attack can be detected. Providing that there is no damage to your systems and others, the most responsible course of action is to cooperate with the participating agency by leaving your compromised system on. This will allow monitoring (and, ultimately, the possibility of terminating the source of the threat to systems just like yours). On the other hand, if there is damage to computers illegally accessed through your system, the choice is more complicated: shutting down the intruder may prevent further damage to systems, but might make it impossible to track down the intruder. If there has been damage, the decision about whether it is important to leave systems up to catch the intruder should involve all the organizations effected. Further complicating the issue of network responsibility is the consideration that if you do not cooperate with the agency involved, you will be less likely to receive help from that agency in the future.

5.6 Documentation Logs

When you respond to an incident, document all details related to the incident. This will provide valuable information to yourself and others as you try to unravel the course of events. Documenting all details will ultimately save you time. If you don't document every relevant phone call, for example, you are likely to forget a good portion of information you obtain, requiring you to contact the source of information once again. This wastes yours and others' time, something you can ill afford. At the same time, recording details will provide evidence for prosecution efforts, providing the case moves in this direction. Documenting an incident also will help you perform a final assessment of damage (something your management as well as law enforcement officers will want to know), and will provide the basis for a follow-up analysis in which you can engage in a valuable "lessons learned" exercise.

During the initial stages of an incident, it is often infeasible to determine whether prosecution is viable, so you should document as if you are gathering evidence for a court case. At a minimum, you should record:

- All system events (audit records).
- All actions you take (time tagged).
- All phone conversations (including the person with whom you talked, the date and time, and the content of the conversation).

The most straightforward way to maintain documentation is keeping a log book. This allows you to go to a centralized, chronological source of information when you need it, instead of requiring you to page through individual sheets of paper. Much of this information is potential evidence in a court of law. Thus, when you initially suspect that an incident will result in prosecution or when an investigative agency becomes involved, you need to regularly (e.g., every day) turn in photocopied, signed copies of your logbook (as well as media you use to record system events) to a document custodian who can store these copied pages in a secure place (e.g., a safe). When you submit information for storage, you should in return receive a signed, dated receipt from the document custodian. Failure to observe these procedures can result in invalidation of any evidence you obtain in a court of law.

6. Establishing Post-Incident Procedures

6.1 Overview

In the wake of an incident, several actions should take place. These actions can be summarized as follows:

1. An inventory should be taken of the systems' assets, i.e., a careful examination should determine how the system was affected by the incident,

2. The lessons learned as a result of the incident should be included in revised security plan to prevent the incident from re-occurring,

3. A new risk analysis should be developed in light of the incident,

4. An investigation and prosecution of the individuals who caused the incident should commence, if it is deemed desirable.

All four steps should provide feedback to the site security policy committee, leading to prompt re-evaluation and amendment of the current policy.

6.2 Removing Vulnerabilities

Removing all vulnerabilities once an incident has occurred is difficult. The key to removing vulnerabilities is knowledge and understanding of the breach. In some cases, it is prudent to remove all access or functionality as soon as possible, and then restore normal operation in limited stages. Bear in mind that removing all access while an incident is in progress will obviously notify all users, including the alleged problem users, that the administrators are aware of a problem; this may have a deleterious effect on an investigation. However, allowing an incident to continue may also open the likelihood of greater damage, loss, aggravation, or liability (civil or criminal).

If it is determined that the breach occurred due to a flaw in the systems' hardware or software, the vendor (or supplier) and the CERT should be notified as soon as possible. Including relevant telephone numbers (also electronic mail addresses and fax numbers) in the site security policy is strongly recommended. To aid prompt acknowledgment and understanding of the problem, the flaw should be described in as much detail as possible, including details about how to exploit the flaw.

As soon as the breach has occurred, the entire system and all its components should be considered suspect. System software is the most probable target. Preparation is key to recovering from a possibly tainted system. This includes checksumming all tapes from the vendor using a checksum algorithm which (hopefully) is resistant to tampering [10]. (See sections 3.9.4.1, 3.9.4.2.) Assuming original vendor distribution tapes are available, an analysis of all system files should commence, and any irregularities should be noted and referred to all parties involved in handling the incident. It can be very difficult, in some cases, to decide which backup tapes to recover from; consider that the incident may have continued for months or years before discovery, and that the suspect may be an employee of the site, or otherwise have intimate knowledge or access to the systems. In all cases, the pre-incident preparation will determine what recovery is possible. At worst-case, restoration from the original manufacturers' media and a re-installation of the systems will be the most prudent solution.

Review the lessons learned from the incident and always update the policy and procedures to reflect changes necessitated by the incident.

6.2.1 Assessing Damage

Before cleanup can begin, the actual system damage must be discerned. This can be quite time consuming, but should lead into some of the insight as to the nature of the incident, and aid investigation and prosecution. It is best to compare previous backups or original tapes when possible; advance preparation is the key. If the system supports centralized logging (most do), go back over the logs and look for abnormalities. If process accounting and connect time accounting is enabled, look for patterns of system usage. To a lesser extent, disk usage may shed light on the incident. Accounting can provide much helpful information in an analysis of an incident and subsequent prosecution.

6.2.2 Cleanup

Once the damage has been assessed, it is necessary to develop a plan for system cleanup. In general, bringing up services in the order of demand to allow a minimum of user inconvenience is the best practice. Understand that the proper recovery procedures for the system are extremely important and should be specific to the site. It may be necessary to go back to the original distributed tapes and recustomize the system. To facilitate this worst case scenario, a record of the original systems setup and each customization change should be kept current with each change to the system.

6.2.3 Follow up

Once you believe that a system has been restored to a "safe" state, it is still possible that holes and even traps could be lurking in the system. In the follow-up stage, the system should be monitored for items that may have been missed during the cleanup stage. It would be prudent to utilize some of the tools mentioned in section 3.9.8.2 (e.g., COPS) as a start. Remember, these tools don't replace continual system monitoring and good systems administration procedures.

6.2.4 Keep a Security Log

As discussed in section 5.6, a security log can be most valuable during this phase of removing vulnerabilities. There are two considerations here; the first is to keep logs of the procedures that have been used to make the system secure again. This should include command procedures (e.g., shell scripts) that can be run on a periodic basis to recheck the security. Second, keep logs of important system events. These can be referenced when trying to determine the extent of the damage of a given incident.

6.3 Capturing Lessons Learned

6.3.1 Understand the Lesson

After an incident, it is prudent to write a report describing the incident, method of discovery, correction procedure, monitoring procedure, and a summary of lesson learned. This will aid in

the clear understanding of the problem. Remember, it is difficult to learn from an incident if you don't understand the source.

6.3.2 Resources

6.3.2.1 Other Security Devices, Methods

Security is a dynamic, not static process. Sites are dependent on the nature of security available at each site, and the array of devices and methods that will help promote security.
Keeping up with the security area of the computer industry and their methods will assure a security manager of taking advantage of the latest technology.

6.3.2.2 Repository of Books, Lists, Information Sources

Keep an on site collection of books, lists, information sources, etc., as guides and references for securing the system. Keep this collection up to date. Remember, as systems change, so do security methods and problems.

6.3.2.3 Form a Subgroup

Form a subgroup of system administration personnel that will be the core security staff. This will allow discussions of security problems and multiple views of the site's security issues. This subgroup can also act to develop the site security policy and make suggested changes as necessary to ensure site security.

6.4 Upgrading Policies and Procedures

6.4.1 Establish Mechanisms for Updating Policies, Procedures, and Tools

If an incident is based on poor policy, and unless the policy is changed, then one is doomed to repeat the past. Once a site has recovered from an incident, site policy and procedures should be reviewed to encompass changes to prevent similar incidents. Even without an incident, it would be prudent to review policies and procedures on a regular basis. Reviews are imperative due to today's changing computing environments.

6.4.2 Problem Reporting Procedures

A problem reporting procedure should be implemented to describe, in detail, the incident and the solutions to the incident. Each incident should be reviewed by the site security subgroup to allow understanding of the incident with possible suggestions to the site policy and procedures.

7. References

[1] Quarterman, J., "The Matrix: Computer Networks and Conferencing Systems Worldwide," Pg. 278, Digital Press, Bedford, MA, 1990.

[2] Brand, R., "Coping with the Threat of Computer Security Incidents: A Primer from Prevention through Recovery," R. Brand, available on-line from: cert.sei.cmu.edu:/pub/info/primer, 8 June 1990.

[3] Fites, M., Kratz, P. and A. Brebner, "Control and Security of Computer Information Systems," Computer Science Press, 1989.

[4] Johnson, D., and J. Podesta, "Formulating a Company Policy on Access to and Use and Disclosure of Electronic Mail on Company Computer Systems," Available from: The Electronic Mail Association (EMA) 1555 Wilson Blvd, Suite 555, Arlington VA 22209, (703) 522-7111, 22 October 1990.

[5] Curry, D., "Improving the Security of Your UNIX System," SRI International Report ITSTD-721-FR-90-21, April 1990.

[6] Cheswick, B., "The Design of a Secure Internet Gateway," Proceedings of the Summer Usenix Conference, Anaheim, CA, June 1990.

[7] Linn, J., "Privacy Enhancement for Internet Electronic Mail: Part I—Message Encipherment and Authentication Procedures," RFC 1113, IAB Privacy Task Force, August 1989.

[8] Kent, S., and J. Linn, "Privacy Enhancement for Internet Electronic Mail: Part II—Certificate-Based Key Management," RFC 1114, IAB Privacy Task Force, August 1989.

[9] Linn, J., "Privacy Enhancement for Internet Electronic Mail: Part III—Algorithms, Modes, and Identifiers," RFC 1115, IAB Privacy Task Force, August 1989.

[10] Merkle, R., "A Fast Software One Way Hash Function," Journal of Cryptology, Vol. 3, No. 1.

[11] Postel, J., "Internet Protocol - DARPA Internet Program Protocol Specification," RFC 791, DARPA, September 1981.

[12] Postel, J., "Transmission Control Protocol - DARPA Internet Program Protocol Specification," RFC 793, DARPA, September 1981.

[13] Postel, J., "User Datagram Protocol," RFC 768, USC/Information Sciences Institute, 28 August 1980.

[14] Mogul, J., "Simple and Flexible Datagram Access Controls for UNIX-based Gateways," Digital Western Research Laboratory Research Report 89/4, March 1989.

[15] Bellovin, S., and M. Merritt, "Limitations of the Kerberos Authentication System," Computer Communications Review, October 1990.

[16] Pfleeger, C., "Security in Computing," Prentice-Hall, Englewood Cliffs, N.J., 1989.

[17] Parker, D., Swope, S., and B. Baker, "Ethical Conflicts: Information and Computer Science, Technology and Business," QED Information Sciences, Inc., Wellesley, MA.

[18] Forester, T., and P. Morrison, "Computer Ethics: Tales and Ethical Dilemmas in Computing," MIT Press, Cambridge, MA, 1990.

[19] Postel, J., and J. Reynolds, "Telnet Protocol Specification," RFC 854, USC/Information Sciences Institute, May 1983.

[20] Postel, J., and J. Reynolds, "File Transfer Protocol," RFC 959, USC/Information Sciences Institute, October 1985.

[21] Postel, J., Editor, "IAB Official Protocol Standards," RFC 1200, IAB, April 1991.

[22] Internet Activities Board, "Ethics and the Internet," RFC 1087, Internet Activities Board, January 1989.

[23] Pethia, R., Crocker, S., and B. Fraser, "Policy Guidelines for the Secure Operation of the Internet," CERT, TIS, CERT, RFC in preparation.

[24] Computer Emergency Response Team (CERT/CC), "Unauthorized Password Change Requests," CERT Advisory CA-91:03, April 1991.

[25] Computer Emergency Response Team (CERT/CC), "TELNET Breakin Warning," CERT Advisory CA-89:03, August 1989.

[26] CCITT, Recommendation X.509, "The Directory: Authentication Framework," Annex C.

[27] Farmer, D., and E. Spafford, "The COPS Security Checker System," Proceedings of the Summer 1990 USENIX Conference, Anaheim, CA, Pgs. 165-170, June 1990.

8. Annotated Bibliography

The intent of this annotated bibliography is to offer a representative collection of resources of information that will help the user of this handbook. It is meant provide a starting point for further research in the security area. Included are references to other sources of information for those who wish to pursue issues of the computer security environment.

8.1 Computer Law

[ABA89]

American Bar Association, Section of Science and Technology, "Guide to the Prosecution of Telecommunication Fraud by the Use of Computer Crime Statutes," American Bar Association, 1989.

[BENDER]

Bender, D., "Computer Law: Evidence and Procedure," M. Bender, New York, NY, 1978-present. Kept up to date with supplements. Years covering 1978-1984 focuses on: Computer law, evidence and procedures. The years 1984 to the current focus on general computer law. Bibliographical references and index included.

[BLOOMBECKER]

Bloombecker, B., "Spectacular Computer Crimes," Dow Jones- Irwin, Homewood, IL, 1990.

[CCH]

Commerce Clearing House, "Guide to Computer Law," (Topical Law Reports), Chicago, IL., 1989. Court cases and decisions rendered by federal and state courts throughout the United States on federal and state computer law. Includes Case Table and Topical Index.

[CONLY]

Conly, C., "Organizing for Computer Crime Investigation and Prosecution," U.S. Dept. of Justice, Office of Justice Programs, Under Contract Number OJP-86-C-002, National Institute of Justice, Washington, DC, July 1989.

[FENWICK]

Fenwick, W., Chair, "Computer Litigation, 1985: Trial Tactics and Techniques," Litigation Course Handbook Series No. 280, Prepared for distribution at the Computer Litigation, 1985: Trial Tactics and Techniques Program, February-March 1985.

[GEMIGNANI]

Gemignani, M., "Viruses and Criminal Law," Communications of the ACM, Vol. 32, No. 6, Pgs. 669-671, June 1989.

[HUBAND]

Huband, F., and R. Shelton, Editors, "Protection of Computer Systems and Software: New Approaches for Combating Theft of Software and Unauthorized Intrusion," Papers presented at a workshop sponsored by the National Science Foundation, 1986.

[MCEWEN]

McEwen, J., "Dedicated Computer Crime Units," Report Contributors: D. Fester and H. Nugent, Prepared for the National Institute of Justice, U.S. Department of Justice, by Institute for Law and Justice, Inc., under contract number OJP-85-C-006, Washington, DC, 1989.

[PARKER]

Parker, D., "Computer Crime: Criminal Justice Resource Manual," U.S. Dept. of Justice, National Institute of Justice, Office of Justice Programs, Under Contract Number OJP-86-C-002, Washington, D.C., August 1989.

[SHAW]

Shaw, E., Jr., "Computer Fraud and Abuse Act of 1986, Congressional Record (3 June 1986), Washington, D.C., 3 June 1986.

[TRIBLE]

Trible, P., "The Computer Fraud and Abuse Act of 1986," U.S. Senate Committee on the Judiciary, 1986.

8.2 Computer Security

[BRAND]

Brand, R., "Coping with the Threat of Computer Security Incidents: A Primer from Prevention through Recovery," R. Brand, 8 June 1990.

[CAELLI]

Caelli, W., Editor, "Computer Security in the Age of Information," Proceedings of the Fifth IFIP International Conference on Computer Security, IFIP/Sec '88.

[CARROLL]

Carroll, J., "Computer Security," 2nd Edition, Butterworth Publishers, Stoneham, MA, 1987.

[CHESWICK]

Cheswick, B., "The Design of a Secure Internet Gateway," Proceedings of the Summer Usenix Conference, Anaheim, CA, June 1990.

[COOPER]

Cooper, J., "Computer and Communications Security: Strategies for the 1990s," McGraw-Hill, 1989.

As computer security becomes a more important issue in modern society, it begins to warrant a systematic approach. The vast majority of the computer security problems and the costs associated with them can be prevented with simple inexpensive measures. The most important and cost effective of these measures are available in the prevention and planning phases. These methods are presented in this paper, followed by a simplified guide to incident handling and recovery. Available on-line from:

`cert.sei.cmu.edu:/pub/info/primer.`

Brief abstract (slight paraphrase from the original abstract): AT&T maintains a large internal Internet that needs to be protected from outside attacks, while providing useful services between the two. This paper describes AT&T's Internet gateway. This gateway passes mail and many of the common Internet services between AT&T internal machines and the Internet. This is accomplished without IP connectivity using a pair of machines: a trusted internal machine and an untrusted external gateway. These are connected by a private link. The internal machine provides a few carefully-guarded services to the external gateway. This configuration helps protect the internal internet even if the external machine is fully compromised.

This is a very useful and interesting design. Most firewall gateway systems rely on a system that, if compromised, could allow access to the machines behind the firewall. Also, most firewall systems require users who want access to Internet services to have accounts on the firewall machine. AT&T's design allows AT&T internal internet users access to the standard services of TELNET and FTP from their own workstations without accounts on the firewall machine. A very useful paper that shows how to maintain some of the benefits of Internet connectivity while still maintaining strong security.

[CURRY]

Curry, D., "Improving the Security of Your UNIX System," SRI International Report ITSTD-721-FR-90-21, April 1990.

This paper describes measures that you, as a system administrator can take to make your UNIX system(s) more secure. Oriented primarily at SunOS 4.x, most of the information covered applies equally well to any Berkeley UNIX system with or without NFS and/or Yellow Pages (NIS). Some of the information can also be applied to System V, although this is not a primary focus of the paper. A very useful reference, this is also available on the Internet in various locations, including the directory cert.sei.cmu.edu:/pub/info.

[FITES]

Fites, M., Kratz, P. and A. Brebner, "Control and Security of Computer Information Systems," Computer Science Press, 1989.

This book serves as a good guide to the issues encountered in forming computer security policies and procedures. The book is designed as a textbook for an introductory course in information systems security.

The book is divided into five sections: Risk Management (I), Safeguards: security and control measures, organizational and administrative (II), Safeguards: Security and Control Measures, Technical (III), Legal Environment and Professionalism (IV), and CICA Computer Control Guidelines (V).

The book is particularly notable for its straight-forward approach to security, emphasizing that common sense is the first consideration in designing a security program. The authors note that there is a tendency to look to more technical solutions to security problems while overlooking organizational controls which are often cheaper and much more effective. 298 pages, including references and index.

[GARFINKEL]

Garfinkel, S, and E. Spafford, "Practical Unix Security," O'Reilly & Associates, ISBN 0-937175-72-2, May 1991.

Approx 450 pages, $29.95. Orders: 1-800-338-6887

(US & Canada), 1-707-829-0515 (Europe), email: nuts@ora.com

This is one of the most useful books available on Unix security. The first part of the book covers standard Unix and Unix security basics, with particular emphasis on passwords. The second section covers enforcing security on the system. Of particular interest to the Internet user are the sections on network security, which address many of the common security problems that afflict Internet Unix users. Four chapters deal with handling security incidents, and the book concludes with discussions of encryption, physical security, and useful checklists and lists of resources. The book lives up to its name; it is filled with specific references to possible security holes, files to check, and things to do to improve security. This book is an excellent complement to this handbook.

[GREENIA90]

Greenia, M., "Computer Security Information Sourcebook," Lexikon Services, Sacramento, CA, 1989.

A manager's guide to computer security. Contains a sourcebook of key reference materials including access control and computer crimes bibliographies.

[HOFFMAN]

Hoffman, L., "Rogue Programs: Viruses, Worms, and Trojan Horses," Van Nostrand Reinhold, NY, 1990.

(384 pages, includes bibliographical references and index.)

[JOHNSON]

Johnson, D., and J. Podesta, "Formulating A Company Policy on Access to and Use and Disclosure of Electronic Mail on Company Computer Systems."

A white paper prepared for the EMA, written by two experts in privacy law. Gives background on the issues, and presents some policy options.

Available from:

The Electronic Mail Association (EMA)
1555 Wilson Blvd, Suite 555
Arlington, VA, 22209
(703) 522-7111

[KENT]

Kent, Stephen, "E-Mail Privacy for the Internet: New Software and Strict Registration Procedures will be Implemented this Year," Business Communications Review, Vol. 20, No. 1, Pg. 55, 1 January 1990.

[LU]

Lu, W., and M. Sundareshan, "Secure Communication in Internet Environments: A Hierachical Key Management Scheme for End-to-End Encryption," IEEE Transactions on Communications, Vol. 37, No. 10, Pg. 1014, 1 October 1989.

[LU1]

Lu, W., and M. Sundareshan, "A Model for Multilevel Security in Computer Networks," IEEE Transactions on Software Engineering, Vol. 16, No. 6, Page 647, 1 June 1990.

[NSA]

National Security Agency, "Information Systems Security Products and Services Catalog," NSA, Quarterly Publication. NSA's catalogue contains chapter on: Endorsed Cryptographic Products List; NSA Endorsed Data Encryption Standard (DES) Products List; Protected Services List; Evaluated Products List; Preferred Products List; and Endorsed Tools List. The catalogue is available from the Superintendent of Documents, U.S. Government Printing Office, Washington, D.C. One may place telephone orders by calling: (202) 783-3238.

[OTA]

United States Congress, Office of Technology Assessment, "Defending Secrets, Sharing Data: New Locks and Keys for Electronic Information," OTA-CIT-310, October 1987.

This report, prepared for congressional committee considering Federal policy on the protection of electronic information, is interesting because of the issues it raises regarding the impact of

technology used to protect information. It also serves as a reasonable introduction to the various encryption and information protection mechanisms. 185 pages. Available from the U.S. Government Printing Office.

[PALMER]

Palmer, I., and G. Potter, "Computer Security Risk Management," Van Nostrand Reinhold, NY, 1989.

[PFLEEGER]

Pfleeger, C., "Security in Computing," Prentice-Hall, Englewood Cliffs, NJ, 1989.

A general textbook in computer security, this book provides an excellent and very readable introduction to classic computer security problems and solutions, with a particular emphasis on encryption. The encryption coverage serves as a good introduction to the subject. Other topics covered include building secure programs and systems, security of database, personal computer security, network and communications security, physical security, risk analysis and security planning, and legal and ethical issues. 538 pages including index and bibliography.

[SHIREY]

Shirey, R., "Defense Data Network Security Architecture," Computer Communication Review, Vol. 20, No. 2, Page 66, 1 April 1990.

[SPAFFORD]

Spafford, E., Heaphy, K., and D. Ferbrache, "Computer Viruses: Dealing with Electronic Vandalism and Programmed Threats," ADAPSO, 1989. (109 pages.)

This is a good general reference on computer viruses and related concerns. In addition to describing viruses in some detail, it also covers more general security issues, legal recourse in case of security problems, and includes lists of laws, journals focused on computers security, and other security-related resources.

Available from: ADAPSO, 1300 N. 17th St, Suite 300, Arlington VA 22209. (703) 522-5055.

[STOLL88]

Stoll, C., "Stalking the Wily Hacker," Communications of the ACM, Vol. 31, No. 5, Pgs. 484-497, ACM, New York, NY, May 1988.

This article describes some of the technical means used to trace the intruder that was later chronicled in "Cuckoo's Egg" (see below).

[STOLL89]

Stoll, C., "The Cuckoo's Egg," ISBN 00385-24946-2, Doubleday, 1989.

Clifford Stoll, an astronomer turned UNIX System Administrator, recounts an exciting, true story of how he tracked a computer intruder through the maze of American military and research networks. This book is easy to understand and can serve as an interesting introduction to the world of networking. Jon Postel says in a book review, "[this book] ... is absolutely essential reading for anyone that uses or operates any computer connected to the Internet or any other computer network."

[VALLA]

allabhaneni, S., "Auditing Computer Security: A Manual with Case Studies," Wiley, New York, NY, 1989.

8.3 Ethics

[CPSR89]

Computer Professionals for Social Responsibility, "CPSR Statement on the Computer Virus," CPSR, Communications of the ACM, Vol. 32, No. 6, Pg. 699, June 1989.

This memo is a statement on the Internet Computer Virus by the Computer Professionals for Social Responsibility (CPSR).

[DENNING]

Denning, Peter J., Editor, "Computers Under Attack: Intruders, Worms, and Viruses," ACM Press, 1990.

A collection of 40 pieces divided into six sections: the emergence of worldwide computer networks, electronic breakins, worms, viruses, counterculture (articles examining the world of the "hacker"), and finally a section discussing social, legal, and ethical considerations. A thoughtful collection that addresses the phenomenon of attacks on computers. This includes a number of previously published articles and some new ones. The previously published ones are well chosen, and include some references that might be otherwise hard to obtain. This book is a key reference to computer security threats that have generated much of the concern over computer security in recent years.

[ERMANN]

Ermann, D., Williams, M., and C. Gutierrez, Editors, "Computers, Ethics, and Society," Oxford University Press, NY, 1990. (376 pages, includes bibliographical references).

[FORESTER]

Forester, T., and P. Morrison, "Computer Ethics: Tales and Ethical Dilemmas in Computing," MIT Press, Cambridge, MA, 1990. (192 pages including index.)

From the preface: "The aim of this book is two-fold: (1) to describe some of the problems created by society by computers, and (2) to show how these problems present ethical dilemmas for computers professionals and computer users. The problems created by computers arise, in turn, from two main sources: from hardware and software malfunctions and from misuse by human beings. We argue that computer systems by their very nature are insecure, unreliable, and unpredictable—and that society has yet to come to terms with the consequences. We also seek to show how society has become newly vulnerable to human misuse of computers in the form of computer crime, software theft, hacking, the creation of viruses, invasions of privacy, and so on." The eight chapters include "Computer Crime," "Software Theft," "Hacking and Viruses," "Unreliable Computers," "The Invasion of Privacy," "AI and Expert Systems," and "Computerizing the Workplace." Includes extensive notes on sources and an index.

[GOULD]

Gould, C., Editor, "The Information Web: Ethical and Social Implications of Computer Networking," Westview Press, Boulder, CO, 1989.

[IAB89]

Internet Activities Board, "Ethics and the Internet," RFC 1087, IAB, January 1989. Also appears in the Communications of the ACM, Vol. 32, No. 6, Pg. 710, June 1989.

This memo is a statement of policy by the Internet Activities Board (IAB) concerning the proper use of the resources of the Internet. Available on-line on host ftp.nisc.sri.com, directory rfc, filename rfc1087.txt. Also available on host nis.nsf.net, directory RFC, filename RFC1087.TXT-1.

[MARTIN]

Martin, M., and R. Schinzinger, "Ethics in Engineering," McGraw Hill, 2nd Edition, 1989.

[MIT89]

Massachusetts Institute of Technology, "Teaching Students About Responsible Use of Computers," MIT, 1985-1986. Also reprinted in the Communications of the ACM, Vol. 32, No. 6, Pg. 704, Athena Project, MIT, June 1989.

This memo is a statement of policy by the Massachusetts Institute of Technology (MIT) on the responsible use of computers.

[NIST]

National Institute of Standards and Technology, "Computer Viruses and Related Threats: A Management Guide," NIST Special Publication 500-166, August 1989.

[NSF88]

National Science Foundation, "NSF Poses Code of Networking Ethics," Communications of the ACM, Vol. 32, No. 6, Pg. 688, June 1989.

Also appears in the minutes of the regular meeting of the Division Advisory Panel for Networking and Communications Research and Infrastructure, Dave Farber, Chair, November 29-30, 1988.

This memo is a statement of policy by the National Science Foundation (NSF) concerning the ethical use of the Internet.

[PARKER90]

Parker, D., Swope, S., and B. Baker, "Ethical Conflicts: Information and Computer Science, Technology and Business," QED Information Sciences, Inc., Wellesley, MA. (245 pages). Additional publications on Ethics:

The University of New Mexico (UNM)

The UNM has a collection of ethics documents. Included are legislation from several states and policies from many institutions.

Access is via FTP, IP address ariel.umn.edu. Look in the directory /ethics.

8.4 The Internet Worm

[BROCK]

Brock, J., "November 1988 Internet Computer Virus and the Vulnerability of National Telecommunications Networks to Computer Viruses," GAO/T-IMTEC-89-10, Washington, DC, 20 July 1989.

Testimonial statement of Jack L. Brock, Director, U. S. Government Information before the Subcommittee on Telecommunications and Finance, Committee on Energy and Commerce, House of Representatives.

[EICHIN89]

Eichin, M., and J. Rochlis, "With Microscope and Tweezers: An Analysis of the Internet Virus of November 1988," Massachusetts Institute of Technology, February 1989.

Provides a detailed dissection of the worm program. The paper discusses the major points of the worm program then reviews strategies, chronology, lessons and open issues, Acknowledgments; also included are a detailed appendix on the worm program subroutine by subroutine, an appendix on the cast of characters, and a reference section.

[EISENBERG89]

Eisenberg, T., D. Gries, J. Hartmanis, D. Holcomb, M. Lynn, and T. Santoro, "The Computer Worm," Cornell University, 6 February 1989.

A Cornell University Report presented to the Provost of the University on 6 February 1989 on the Internet Worm.

[GAO]

U.S. General Accounting Office, "Computer Security - Virus Highlights Need for Improved Internet Management," United States General Accounting Office, Washington, DC, 1989.

This 36 page report (GAO/IMTEC-89-57), by the U.S. Government Accounting Office, describes the Internet worm and its effects. It gives a good overview of the various U.S. agencies involved in the Internet today and their concerns vis-a-vis computer security and networking. Available on-line on host nnsc.nsf.net, directory pub, filename GAO_RPT; and on nis.nsf.net, directory nsfnet, filename GAO_RPT.TXT.

[REYNOLDS89]

The Helminthiasis of the Internet, RFC 1135, USC/Information Sciences Institute, Marina del Rey, CA, December 1989.

This report looks back at the helminthiasis (infestation with, or disease caused by parasitic worms) of the Internet that was unleashed the evening of 2 November 1988. This document provides a glimpse at the infection, its festering, and cure. The impact of the worm on the Internet community, ethics statements, the role of the news media, crime in the computer world, and future prevention is discussed. A documentation review presents four publications that describe in detail this particular parasitic computer program. Reference and bibliography sections are also included. Available on-line on host ftp.nisc.sri.com directory rfc, filename rfc1135.txt. Also available on host nis.nsf.net, directory RFC, filename RFC1135.TXT-1.

[SEELEY89]

Seeley, D., "A Tour of the Worm," Proceedings of 1989 Winter USENIX Conference, Usenix Association, San Diego, CA, February 1989.

Details are presented as a "walk thru" of this particular worm program. The paper opened with an abstract, introduction, detailed chronology of events upon the discovery of the worm, an overview, the internals of the worm, personal opinions, and conclusion.

[SPAFFORD88]

Spafford, E., "The Internet Worm Program: An Analysis," Computer Communication Review, Vol. 19, No. 1, ACM SIGCOM, January 1989. Also issued as Purdue CS Technical Report CSD-TR-823, 28 November 1988.

Describes the infection of the Internet as a worm program that exploited flaws in utility programs in UNIX based systems. The report gives a detailed description of the components of the worm program: data and functions. Spafford focuses his study on two completely independent reverse-compilations of the worm and a version disassembled to VAX assembly language.

[SPAFFORD89]

Spafford, G., "An Analysis of the Internet Worm," Proceedings of the European Software Engineering Conference 1989, Warwick England, September 1989.

Proceedings published by Springer-Verlag as: Lecture Notes in Computer Science #387. Also issued as Purdue Technical Report #CSD-TR-933.

8.5 National Computer Security Center (NCSC)

All NCSC publications, approved for public release, are available from the NCSC Superintendent of Documents.

NCSC = National Computer Security Center
9800 Savage Road
Ft Meade, MD 20755-6000

CSC = Computer Security Center: an older name for the NCSC

NTISS = National Telecommunications and Information Systems Security

NTISS Committee, National Security Agency
Ft Meade, MD 20755-6000

[CSC]

Department of Defense, "Password Management Guideline," CSC-STD-002-85, 12 April 1985, 31 pages.

The security provided by a password system depends on the passwords being kept secret at all times. Thus, a password is vulnerable to compromise whenever it is used, stored, or even known. In a password-based authentication mechanism implemented on an ADP system, passwords are vulnerable to compromise due to five essential aspects of the password system: 1) a password must be initially assigned to a user when enrolled on the ADP system; 2) a user's password must be changed periodically; 3) the ADP system must maintain a 'password database'; 4) users must remember their passwords; and 5) users must enter their passwords into the ADP system at authentication time. This guideline prescribes steps to be taken to minimize the vulnerability of passwords in each of these circumstances.

[NCSC1]

CSC, "A Guide to Understanding AUDIT in Trusted Systems," NCSC-TG-001, Version-2, 1 June 1988, 25 pages.

Audit trails are used to detect and deter penetration of a computer system and to reveal usage that identifies misuse. At the discretion of the auditor, audit trails may be limited to specific events or may encompass all of the activities on a system. Although not required by the criteria, it should be possible for the target of the audit mechanism to be either a subject or an object. That is to say, the audit mechanism should be capable of monitoring every time John accessed the system as well as every time the nuclear reactor file was accessed; and likewise every time John accessed the nuclear reactor file.

[NCSC2]

NCSC, "A Guide to Understanding DISCRETIONARY ACCESS CONTROL in Trusted Systems," NCSC-TG-003, Version-1, 30 September 1987, 29 pages.

Discretionary control is the most common type of access control mechanism implemented in computer systems today. The basis of this kind of security is that an individual user, or program operating on the user's behalf, is allowed to specify explicitly the types of access other users (or programs executing on their behalf) may have to information under the user's control. [...] Discretionary controls are not a replacement for mandatory controls. In any environment in which information is protected, discretionary security provides for a finer granularity of control within the overall constraints of the mandatory policy.

[NCSC3]

NCSC, "A Guide to Understanding CONFIGURATION MANAGEMENT in Trusted Systems," NCSC-TG-006, Version-1, 28 March 1988, 31 pages.

Configuration management consists of four separate tasks: identification, control, status accounting, and auditing. For every change that is made to an automated data processing (ADP) system, the design and requirements of the changed version of the system should be identified. The control task of configuration management is performed by subjecting every change to documentation, hardware, and software/firmware to review and approval by an authorized authority. Configuration status accounting is responsible for recording and reporting on the configuration of the product throughout the change. Finally, through the process of a configuration audit, the completed change can be verified to be functionally correct, and for trusted systems, consistent with the security policy of the system.

[NTISS]

NTISS, "Advisory Memorandum on Office Automation Security Guideline," NTISSAM CONPUSEC/1-87, 16 January 1987, 58 pages.

This document provides guidance to users, managers, security officers, and procurement officers of Office Automation Systems. Areas addressed include: physical security, personnel security, procedural security, hardware/software security, emanations security (TEMPEST), and communications security for stand-alone OA Systems, OA Systems used as terminals connected to mainframe computer systems, and OA Systems used as hosts in a Local Area Network (LAN). Differentiation is made between those Office Automation Systems equipped with removable storage media only (e.g., floppy disks, cassette tapes, removable hard disks) and those Office Automation Systems equipped with fixed media (e.g., Winchester disks).

Additional NCSC Publications:

[NCSC4]

National Computer Security Center, "Glossary of Computer Security Terms," NCSC-TG-004, NCSC, 21 October 1988.

[NCSC5]

National Computer Security Center, "Trusted Computer System Evaluation Criteria," DoD 5200.28-STD, CSC-STD-001-83, NCSC, December 1985.

[NCSC7]

National Computer Security Center, "Guidance for Applying the Department of Defense Trusted Computer System Evaluation Criteria in Specific Environments," CSC-STD-003-85, NCSC, 25 June 1985.

[NCSC8]

National Computer Security Center, "Technical Rationale Behind CSC-STD-003-85: Computer Security Requirements," CSC-STD-004-85, NCSC, 25 June 85.

[NCSC9]

National Computer Security Center, "Magnetic Remanence Security Guideline," CSC-STD-005-85, NCSC, 15 November 1985.

This guideline is tagged as a "For Official Use Only" exemption under Section 6, Public Law 86-36 (50 U.S. Code 402). Distribution authorized of U.S. Government agencies and their contractors to protect unclassified technical, operational, or administrative data relating to operations of the National Security Agency.

[NCSC10]

National Computer Security Center, "Guidelines for Formal Verification Systems," Shipping list no.: 89-660-P, The Center, Fort George G. Meade, MD, 1 April 1990.

[NCSC11]

National Computer Security Center, "Glossary of Computer Security Terms," Shipping list no.: 89-254-P, The Center, Fort George G. Meade, MD, 21 October 1988.

[NCSC12]

National Computer Security Center, "Trusted UNIX Working Group (TRUSIX) rationale for selecting access control list features for the UNIX system," Shipping list no.: 90-076-P, The Center, Fort George G. Meade, MD, 1990.

[NCSC13]

National Computer Security Center, "Trusted Network Interpretation," NCSC-TG-005, NCSC, 31 July 1987.

[NCSC14]

Tinto, M., "Computer Viruses: Prevention, Detection, and Treatment," National Computer Security Center C1 Technical Report C1-001-89, June 1989.

[NCSC15]

National Computer Security Conference, "12th National Computer Security Conference: Baltimore Convention Center, Baltimore, MD, 10-13 October, 1989: Information Systems Security, Solutions for Today - Concepts for Tomorrow," National Institute of Standards and National Computer Security Center, 1989.

8.6 Security Checklists

[AUCOIN]

Aucoin, R., "Computer Viruses: Checklist for Recovery," Computers in Libraries, Vol. 9, No. 2, Pg. 4, 1 February 1989.

[WOOD]

Wood, C., Banks, W., Guarro, S., Garcia, A., Hampel, V., and H. Sartorio, "Computer Security: A Comprehensive Controls Checklist," John Wiley and Sons, Interscience Publication, 1987.

8.7 Additional Publications

Defense Data Network's Network Information Center (DDN NIC) The DDN NIC maintains DDN Security bulletins and DDN Management bulletins online on the machine:

NIC.DDN.MIL. They are available via anonymous FTP. The DDN Security bulletins are in the directory: SCC, and the DDN Management bulletins are in the directory: DDN-NEWS.

For additional information, you may send a message to:

NIC@NIC.DDN.MIL, or call the DDN NIC at: 1-800-235-3155.

[DDN88]

Defense Data Network, "BSD 4.2 and 4.3 Software Problem Resolution," DDN MGT Bulletin #43, DDN Network Information Center, 3 November 1988.

A Defense Data Network Management Bulletin announcement on the 4.2bsd and 4.3bsd software fixes to the Internet worm.

[DDN89]

DCA DDN Defense Communications System, "DDN Security Bulletin 03," DDN Security Coordination Center, 17 October 1989.

IEEE Proceedings

[IEEE]

"Proceedings of the IEEE Symposium on Security and Privacy," published annually.

IEEE Proceedings are available from:

Computer Society of the IEEE
P.O. Box 80452
Worldway Postal Center
Los Angeles, CA 90080

Other Publications:

Computer Law and Tax Report

Computers and Security

Security Management Magazine

Journal of Information Systems Management

Data Processing & Communications Security

SIG Security, Audit & Control Review

Site Security Policy Handbook Working Group

9. Acknowledgments

Thanks to the SSPHWG's illustrious "Outline Squad," who assembled at USC/Information Sciences Institute on 12-June-90: Ray Bates (ISI), Frank Byrum (DEC), Michael A. Contino (PSU), Dave Dalva (Trusted Information Systems, Inc.), Jim Duncan (Penn State Math Department), Bruce Hamilton (Xerox), Sean Kirkpatrick (Unisys), Tom Longstaff (CIAC/LLNL), Fred Ostapik (SRI/NIC), Keith Pilotti (SAIC), and Bjorn Satdeva (/sys/admin, inc.).

Many thanks to Rich Pethia and the Computer Emergency Response Team (CERT); much of the work by Paul Holbrook was done while he was working for CERT. Rich also provided a very thorough review of this document. Thanks also to Jon Postel and USC/Information Sciences Institute for contributing facilities and moral support to this effort.

Last, but NOT least, we would like to thank members of the SSPHWG and Friends for their additional contributions: Vint Cerf (CNRI), Dave Grisham (UNM), Nancy Lee Kirkpatrick (Typist Extraordinaire), Chris McDonald (WSMR), H. Craig McKee (Mitre), Gene Spafford (Purdue), and Aileen Yuan (Mitre).

10. Security Considerations

If security considerations had not been so widely ignored in the Internet, this memo would not have been possible.

11. Authors' Addresses

J. Paul Holbrook
CICNet, Inc.
2901 Hubbard
Ann Arbor, MI 48105
Phone: (313) 998-7680
EMail: holbrook@cic.net

Joyce K. Reynolds
University of Southern California
Information Sciences Institute
4676 Admiralty Way
Marina del Rey, CA 90292
Phone: (213) 822-1511
EMail: JKREY@ISI.EDU