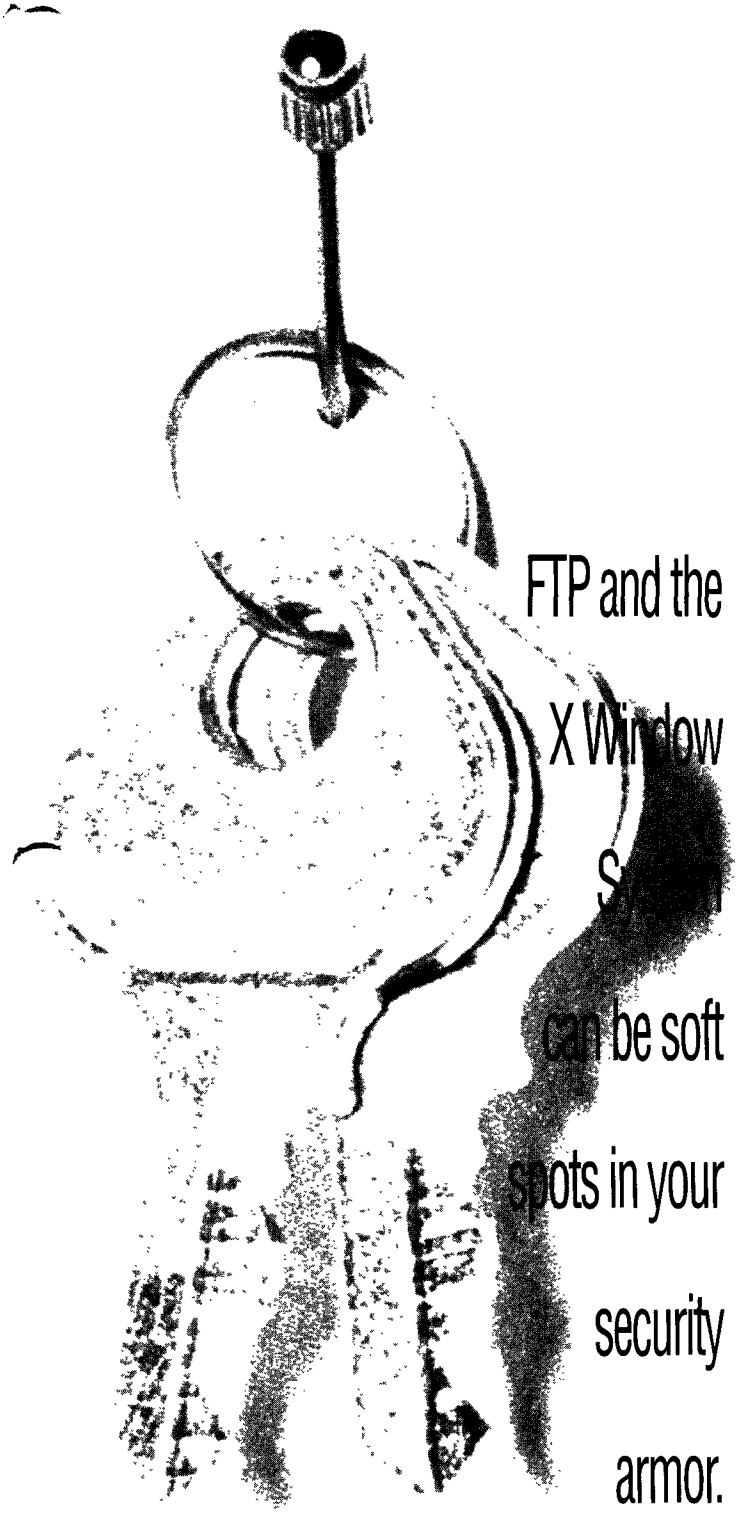

Practical UNIX Security

by SIMSON GARFINKEL
and
GENE SPAFFORD

Editor's note: Managing anonymous `ftp` *and* `xhost` *requires a little ingenuity. This month's installment gives networkers advice on how to maintain their guards while retaining functionality. There's also a summary of the security implications of the network services discussed in July, August and September.*

The File Transfer Protocol (FTP) allows you to transfer complete files between systems. `ftp` is the client program; `/etc/ftpd` (sometimes called `/usr/etc/in.ftpd`) is the server.

When you use FTP to contact a remote machine, the remote computer requires that you log in by providing your username and password; `ftp` logins are recorded on the remote machine in the `/usr/adm/wtmp` file. Because the passwords typed to FTP are transmitted over the network, they can be intercepted (just like the `telnet` and `rexec` commands); for this reason, some sites may wish to disable the `ftp` and `ftpd` programs.

Older versions of `ftpd` had bugs in them that allowed crackers to break into a system. If your version of `ftpd` is older than December 1988, replace it with a newer version. (One way to tell the age of your `ftpd` program is to do an `ls -l` on the executable. This may not be reliable, however.) FTP can be set up for "anonymous" access, which allows people on the network who do not have an account on your machine to deposit or retrieve files from a special directory. Many institutions use anonymous FTP as a low-cost method to distribute software and databases to the public free of charge.

To use anonymous FTP, simply specify "anonymous" as your username, and your real identity as the password. (The real name that you provide is merely a courtesy to the person who manages the computer to which you are connecting. It is written into the log file used by `last`.)

```
% ftp athenadist.mit.edu
Connected to AENEAS.MIT.EDU.
220 aeneas FTP server (Version 4.136 Mon Oct 31
       23:18:38 EST 1988) ready
Name (athena-dist.mit.edu:fred): anonymous
331 Guest login ok, send ident as password.
password:
230 Guest login ok, access restrictions apply.
ftp>
```

If you want to restrict FTP access, it can be done. The `/etc/ftpusers` file contains a list of the users who are NOT allowed to use FTP to access any files. This file should contain all accounts that do not belong to bona fide human beings:

```
# cat /etc/ftpusers
root
uucp
news
bin
ingres
nobody
daemon
```

Setting Up Anonymous FTP

It is relatively easy to set up anonymous FTP on a server, but important to do it correctly, because you are potentially giving access to your system to everybody on the network.

To set up anonymous FTP, you must create a special account with the name `ftp`. Files that are available by anonymous FTP will be placed in the `ftp` home directory; you should therefore put the directory in a special place, such as `/usr/spool/ftp`. Remote users can transfer large files to your system. Therefore, it might be a good idea to put a file quota on user `ftp`, or else locate the home directory on an isolated partition.

When it is used for anonymous FTP, `ftpd` uses the `chroot(2)` function call to change the root of the file system to the home directory of the `ftp` account. For this reason, you must set up that account's home directory as a mini-filesystem. Three directories go into this mini-filesystem:

- `bin` – This directory holds a copy of the `/bin/ls` program, which `ftpd` uses to list files. If your system uses dynamic linking, not symbolic links, like `ls -s`, but dynamically linked, shared libraries, you must either install the dynamic libraries in the appropriate directory (viz; `/usr/spool/ftp/lib`), or else install programs that are statically linked.
- `etc` – This directory holds a copy of the `/etc/passwd` and `/etc/group` files, which are put there so the `/bin/ls` command will print user names and group names when it lists files. Replace the encrypted passwords in this file with asterisks. Some security-conscious sites may wish to delete some or all account names from the `passwd` file; the only one that needs to be present is `ftp`.
- `pub` – This directory, short for "public," holds the files that are actually made available for anonymous FTP transfer. You can have as many subdirectories as you wish in the `pub` directory.

Be sure to place the actual files in these directories, rather than using symbolic links pointing to other places on your system. Because the `ftpd` program uses the `chroot(2)` system call, symbolic links will not behave properly with anonymous FTP. (To set up FTP directories, execute the commands in Figure 1 as the superuser.)

Some sites set the mode of the `~ftp/pub` directory to 1777, which allows people on the network to leave files anonymously. Alternatively, you can create a subdirectory in the `pub` directory called open, and set the mode of that directory to be 1777. In either case, you may wish to establish a quota for the `ftp` account, so files left anonymously do not overrun the available space on your system. You should also monitor the contents of the directory on a regular basis, and delete anything that looks suspicious. In addition, you should set up a mail alias for the `ftp` user so that mail sent to `ftp` is delivered to a system managers.

TFPT

TFTP is the Trivial File Transfer Protocol. TFTP is a UDP-based file-transfer program that provides no security. There is a set of files that the TFTP program is allowed to transmit from your computer, and the program will transmit them to anybody on the Internet who asks for them. One of the main uses of TFTP is to allow workstations to boot over the network; the TFTP protocol is simple enough to be programmed into a small read-only memory.

Because TFTP has no security, `tftpd`, the TFTP daemon, is normally restricted so that it can transfer files only to or from a certain directory. Unfortunately, many early versions of `tftpd` had no such restriction.

You can test your version of `tftpd` with the `tftp` program for this restriction with the following sequence:

```
% tftp localhost
tftp> get /etc/passwd tmp
Error code 1: File not found
tftp> quit
%
```

If `tftp` does not respond `Error code 1: File not found`, or simply hang with no message, then get a current version of the program. Sun Microsystems Inc. operating systems prior to release 4.0 did not restrict file transfer from the TFTP program.

The X Window System

X is a popular network-based window system that allows many programs to share a single graphical display. X-based

programs display their output in windows, which can be either on the same computer on which the program is running or on any other computer on the network.

Each graphical device that runs X is controlled by a special program, called the X Window Server. Other programs, called X clients, connect to the X Window Server over the network and tell it what to display. Two popular X clients are `xterm` (the X terminal emulator) and `xclock` (which displays an analog or digital clock on the screen).

The X Window System can be a security hazard. Although there are a number of mechanisms inside X to give some security features, these can be circumvented in many circumstances.

The xhost Facility

X uses a system called `xhost` to provide a minimal amount of security for window system users. Each X Window Server has a built-in list of hosts from which it will accept connections; connections from all other hosts are refused. The `xhost` command lets users view and change the current list of "X hosted" hosts.

Typing `xhost` by itself displays a list of the current hosts that may connect to your X Window Server.

```
% xhost
prose.cambridge.ma.us
next.cambridge.ma.us
%
```

You can add a host to the `xhost` list by supplying a plus sign, followed by the host's name on the command line after the `xhost` command. You can remove a host from the `xhost` list by supplying its name preceded by a hyphen:

```
% xhost +idr.cambridge.ma.us
% xhost next.cambridge.ma.us
prose.cambridge.ma.us
idr.cambridge.ma.us
% xhost -next.cambridge.ma.us
prose.cambridge.ma.us
idr.cambridge.ma.us
```

Figure 1. Establishing FTP

```
# mkdir ~ftp/bin ~ftp/etc ~ftp/pub                    Create needed directories
```

Set up `~ftp/bin`:

```
# /bin/ls ~ftp/bin                                    Make a copy of the ls program.
# chmod 111 ~ftp/fin/ls                               Make sure ls can't be changed.
# chmod 111 ~ftp/bin                                  Make directory execute-only.
# chown root ~ftp/bin                                 Make sure root owns the directory
```

Set up `~ftp/etc`:

```
# sed -e 's/:[^:]*:/:*:/' /etc/passwd > ~ftp/etc/passwd  Make a copy of etc/passwd with all passwords changed to asterisks.
# sed -e 's/:[^:]*:/:*:/' /etc/passwd > ~ftp/etc/group   Make a copy of etc/group.
# chmod 444 ~ftp/etc/*                                Make sure files in etc are not writeable.
# chmod 111 ~ftp/etc                                  Make directory execute-only.
# chown root ~ftp/etc                                 Make sure root owns the directory.
```

Set up `~ftp/pub`:

```
# chmod 1777 ~ftp/pub                                 Make directory writeable by anyone.
# chown ftp ~ftp/pub                                  Make sure ftp owns the directory.
# chgrp ftp ~ftp/pub                                  Make sure directory is in group ftp.
```

Secure the `ftp` directory:

```
# chmod 555 ~ftp
# chown root ~ftp
```

You can disable xhost protection by typing:

```
prose% xhost +
```

If you xhost a computer, any user on that computer can connect to your X Server and issue commands. Because of the design of X, this effectively gives any user on that computer the ability to type any command on your keyboard. (For example, although it is difficult, it is possible to write an X application that takes over a user's cursor, moves the cursor to an X-terminal window, and then stuffs keypresses into the X-event queue.) If a client connects to your X Window Server, removing that host from your xhost list will not terminate the connection. It will simply prevent future access. The design of the X Window System allows any client that successfully connects to the X Window Server to exercise complete control over the display. Clients can take over the mouse or the keyboard, send keystrokes to other applications or even kill the windows associated with other clients.

For example, someone could overlay your entire screen with a transparent, invisible window, so that everything you type goes into that window and is copied. The program could then take those keystrokes and push them into the appropriate subwindows, so that you can't tell that you're being monitored. If you then remote login to another system or su, it is possible for someone to capture your password as you type it, without your knowing what has happened. If a person can log into your system, they can capture your keystrokes no matter how your xhosts is set.

Revision 4 of the X Window Protocol has a "secure" feature on the xterm command that makes the window change its color if it is not receiving its input directly from the keyboard. This is a partial fix, but it is not complete. Future versions of X are expected to address this problem in a better way, although it is not immediately obvious how this is going to be accomplished. Even if you use the xhost facility, your X Window System may be vulnerable to attack from computers that are not in your xhost list. The X11R3 Window Server reads a small packet from the client before it determines whether or not the client is in the xhost list. If a client connects to the X Server but does not transmit this initial packet, the X Server halts all operation until it times out in 30 seconds. You can determine whether your X server has this problem by executing the following command:

```
prose% telnet localhost 6001
```

Here "6001" is the TCP/IP port address of the first X server on the system. (The second X display on the system has a TCP/IP address of "6002.")

If your X server has this problem, your workstation's display will freeze. The cursor will not move, and you will be unable to type anything. In some X implementations, the X server will time out after 30 seconds and resume normal operations. Under other X implementations, the server will remain blocked until the connection is aborted.

Although this attack cannot be used to destroy information, it can be used to incapacitate any workstation that runs X11R3 and is connected to the network. If you have this problem with your software, ask your vendor for a corrected update.

Security and Network Services

Network servers are the portals through which the outside world accesses the information stored on your computer. Every server must:

- Determine what information or action the client requests.
- Decide whether or not the client is entitled to the information (optionally authenticating the person [or program] on the other side of the network that is requesting service).
- Transfer the requested information or perform the desired service.

By design, many servers must run with root privileges. A bug or an intentional back door built into a server can therefore compromise the security of an entire computer, opening the system to any user of the network who is aware of the flaw. Even a relatively innocuous program can be the downfall of an entire computer. Flaws may remain in programs distributed by vendors for many years, only to be uncovered sometime in the future.

Perhaps the best-known example of such a flaw was a single line of code in the program /etc/fingerd, the finger server, exploited in1988 by Robert T. Morris' Internet Worm. fingerd provides finger(1) service over the network. One of the very first lines of the program reads a single line of text from stdin containing the name of the user to be fingered.

The original fingerd program contained the lines of code:

```
char line[512];

line[0] = '\0';
gets(line);
```

Because the gets(3) function does not check the length of the line read, it was possible for a rogue program to supply more than 512 bytes of valid data, causing the stack frame of the fingerd server to be overrun. Morris wrote code that caused fingerd to execute a shell, giving the rogue program virtually unrestricted access to the server computer.

The fix for the finger program is simple: Replace the gets(3) function with the fgets(3) function, which does not allow its input buffer to be overridden:

```
fgets(line,sizeof(line),stdin);
```

Fortunately, Morris' program did not actually damage programs or data on computers that it broke into. Nevertheless, it illustrates the fact that any portal program can potentially compromise the system. Remember that just because a hole has never been discovered in a program does not mean that no hole exists. You can use the netstat(1) command to list all of the active and pending TCP/IP connections between your machine and every other machine on the Internet. This is very important if you suspect that somebody is breaking

into your computer or using your computer to break into another one. `netstat` lets you see which machines your machine is talking to. The command's output includes the host and port number of each end of the connection, as well as the number of bytes in the receive and transmit queues.

Monitoring the Net with netstat

If a port has a name assigned in the `/etc/services` file, `netstat` will print it instead of the port number. (See Figure 2 for sample output from the `netstat` command.) The `netstat` command is a powerful way to monitor which computers are "talking" to your computer over the network.

The first two lines indicate `telnet` connections between the machines GHOTI.LCS.MIT.EDU and AMWAY.CH.APOLLO.COM and the machine CHARON.MIT.EDU. Both of these connections originated at the remote machine and represent interactive sessions currently being run on CHARON. You can tell this because unnamed port numbers

Figure 2. Monitoring the Network

```
charon% netstat
Active Internet connections
Proto  Recv-Q Send-Q  Local Address          Foreign Address        (state)
tcp    0      0       CHARON.MIT.EDU.telnet  GHOTI.LCS.MIT.ED.1300  ESTABLISHED
tcp    0      0       CHARON.MIT.EDU.telnet  amway.ch.apollo..4196  ESTABLISHED
tcp    4096   0       CHARON.MIT.EDU.1313    E40-0087.MIT.ED.telne  ESTABLISHED
tcp    0      0       CHARON.MIT.EDU.1312    MINT.LCS.MIT.EDU.6001  ESTABLISHED
tcp    0      0       CHARON.MIT.EDU.1309    MINT.LCS.MIT.EDU.6001  ESTABLISHED
tcp    0      0       CHARON.MIT.EDU.telnet  MINT.LCS.MIT.EDU.1218  ESTABLISHED
tcp    0      0       CHARON.MIT.EDU.1308    E40-0087.MIT.ED.telne  ESTABLISHED
tcp    0      0       CHARON.MIT.EDU.login   RINGO.MIT.EDU.1023     ESTABLISHED
tcp    0      0       CHARON.MIT.EDU.1030    *.*                    LISTEN

charon% netstat -a
...
(Previous netstat printout)
...
tcp    0      0       *.telnet               *.*                    LISTEN
tcp    0      0       *.smtp                 *.*                    LISTEN
tcp    0      0       *.finger               *.*                    LISTEN
tcp    0      0       *.printer              *.*                    LISTEN
tcp    0      0       *.time                 *.*                    LISTEN
tcp    0      0       *.daytime              *.*                    LISTEN
tcp    0      0       *.chargen              *.*                    LISTEN
tcp    0      0       *.discard              *.*                    LISTEN
tcp    0      0       *.echo                 *.*                    LISTEN
tcp    0      0       *.exec                 *.*                    LISTEN
tcp    0      0       *.login                *.*                    LISTEN
tcp    0      0       *.shell                *.*                    LISTEN
tcp    0      0       *.ftp                  *.*                    LISTEN
udp    0      0       *.time                 *.*
udp    0      0       *.daytime              *.*
udp    0      0       *.chargen              *.*
udp    0      0       *.discard              *.*
udp    0      0       *.echo                 *.*
udp    0      0       *.ntalk                *.*
udp    0      0       *.talk                 *.*
udp    0      0       *.biff                 *.*
udp    0      0       *.tftp                 *.*
udp    0      0       *.syslog               *.*
%
```

on the foreign machines are connecting to CHARON's `telnet` port (used for remote virtual terminal service). Likewise, the third `telnet` connection, between CHARON and E40-008-7.MIT.EDU originated at CHARON to the machine E40-008-7. The next two lines are connections to port 6001 (the X Window Server) on MINT.LCS.MIT. EDU. There is a `telnet` from MINT to CHARON, one from CHARON to E40-008-7.MIT.EDU and `rlogin` from RINGO.MIT. EDU to CHARON. The last line indicates that a user program running on CHARON is listening for connections on port 1030. If you run `netstat` on your computer, you will likely see many connections. If you use the X Window System, you may also see "UNIX domain sockets," which are the local network connections from your X clients to the X Window Server.

With the `-a` option, `netstat` will also print a list of all of the TCP and UDP sockets to which programs are listening. Using the `-a` option will provide you with a list of all the ports that programs and users outside your computer can use to enter the system via the network. (Unfortunately, `netstat` will not give you the name of the program that is listening on the socket).

There are weaknesses in the implementation of network services that can be exploited to masquerade temporarily as another machine. There is nothing that you can do to prevent this, assuming the attacker gets the code correct and has access to the network.

This kind of "spoof" is not easy to carry out, may require physical access to your local network and needs exact timing of events to occur. It is also the case that such spoofs are possible to spot afterwards.

Nonetheless, if you are storing something extremely critical on your system, you should consider keeping that system isolated from local and external networks. At the very least, disable all trusted hosts/trusted users. A network connection lets your computer communicate with the outside world, but it also makes it possible for attackers in the outside world to reach into your computer and do damage.

Finally, you should know all of the services that your computer makes available on the network and remove or disable those that you think are too dangerous. You should also decide if the convenience of `.rhosts` files is outweighed by their danger. If so, delete them, or modify your system software to disable the feature. ⊸

***This article** is excerpted from material in* Practical UNIX Security, *by Simson Garfinkel and Gene Spafford, ISBN 0-937175-72-2, published by O'Reilly & Associates, Sebastopol CA, (800) 338-6887 or (707) 829-0515. For further information, contact Linda Lamb, Director of Marketing, O'Reilly & Associates, (617) 354-5800.*