

## Telnet Based Attacks

by: Paul Gurgul, 10/18/2004

<http://www.securitydocs.com/library/2642>

This paper examines attacks developed over the years using the Telnet service. That examination begins with a bit of history. The Telnet protocol was first comprehensively defined by Postel in 1980. In RFC 764, Postel wrote: "The purpose of the Telnet protocol is to provide a fairly general, bi-directional, eight-bit byte oriented communications facility. Its primary goal is to allow a standard method of interfacing terminal devices and terminal-oriented processes to each other. It is envisioned that the protocol may also be used for terminal-terminal communication ("linking") and process-process communication (distributed computation). "

**Cross Reference:** RFC 764 can be found on the Web at <http://sunsite.auc.dk/RFC/rfc/rfc764.html>

### Telnet

Telnet is unique in its design with the notable exception of rlogin. Telnet is designed to allow a user to log in to a foreign machine and execute commands there. Telnet (like rlogin) works as though you are at the console of the remote machine, as if you physically approached the remote machine, turned it on, and began working.

**NOTE:** PC users can get a feel for this by thinking in terms of PCAnywhere or CloseUp. These programs allow you to remotely log in to another PC and execute commands at the remote machine's C: prompt (or even execute commands in Windows, providing you have a very high-speed connection to transmit those graphics over the wire).

**Virtual Terminal** The magic behind Telnet is that it imitates an ASCII terminal connection between two machines located great distances from each other. This is accomplished through the use of a virtual terminal, as described by Postel in this excerpt from RFC 854: When a Telnet connection is first established, each end is assumed to originate and terminate at a "Network Virtual Terminal," or NVT. An NVT is an imaginary device which provides a standard, network-wide, intermediate representation of a canonical terminal...The Network Virtual Terminal (NVT) is a bi-directional character device. The NVT has a printer and a keyboard. The printer responds to incoming data and the keyboard produces outgoing data which is sent over the Telnet connection and, if "echoes" are desired, to the NVT's printer as well. "Echoes" will not be expected to traverse the network (although options exist to enable a "remote" echoing mode of operation, no host is required to implement this option). The code set is seven-bit USASCII in an eight-bit field, except as modified herein. Any code conversion and timing considerations are local problems and do not affect the NVT.

**Cross Reference:** Read RFC 854 in its entirety at <http://sunsite.auc.dk/RFC/rfc/rfc854.html>

A virtual terminal is the equivalent (at least in appearance) of a hard-wired serial connection between the two machines. For example, you can simulate something very similar to a Telnet session by uncommenting the respawn instructions in the inittab file on a Linux box (and most other UNIX boxes) or by disconnecting both the monitor and keyboard on a SPARC and plugging a VT200 terminal into serial A or B. In the first instance, a login: prompt is issued. In the second, all boot process messages are echoed to the connected terminal and eventually, a boot prompt is issued (or perhaps, if the right SCSI disk drive is specified as the boot device in the PROM, the machine will boot and issue a login: prompt).

Therefore, Telnet-based connections are what are called bare bones connections. You will notice that if

you use a VT220 terminal as a head for your SPARC that, when the boot occurs, the cool Sun logo is not printed in color, nor do the cool graphics associated with it appear. Telnet and terminal sessions are completely text based. In addition, Telnet connections do not have facilities to interpret display-oriented languages such as HTML without the assistance of a text-based browser such as Lynx. Therefore, retrieving a Web page through Telnet will reveal no pictures or nicely formatted text; it will reveal only the source of the document (unless, of course, you have logged in via Telnet and are now using Lynx).

**NOTE:** Lynx is a completely terminal-based HTML browser for use with shell-account or even DOS-based TCP/IP connections. It is a no-frills way to access the World Wide Web.

### Telnet Security History

Telnet has cropped up in security advisories many times. Telnet security problems vary considerably, with a large number of vulnerabilities surfacing due to programming errors. However, programming errors are not the only reasons Telnet has appeared on advisories. In August of 1989, for example, the problem was a trojan, as the CERT advisory "Telnet Break-in Warning" explains: Many computers connected to the Internet have recently experienced unauthorized system activity. Investigation shows that the activity has occurred for several months and is spreading. Several UNIX computers have had their "Telnet" programs illicitly replaced with versions of "Telnet" which log outgoing login sessions (including user names and passwords to remote systems). It appears that access has been gained to many of the machines which have appeared in some of these session logs.

**Cross Reference:** To view this CERT advisory in its entirety, visit [ftp://ftp.uwsg.indiana.edu/pub/security/cert/cert\\_advisories/CA-89:03.telnet.breakin.warning](ftp://ftp.uwsg.indiana.edu/pub/security/cert/cert_advisories/CA-89:03.telnet.breakin.warning)

That attack occurred just prior to the establishment of the DDN Security Coordination Center (September 1989), so there is little documentation about whether it affected government computers. Also, although the efforts of CERT are appreciated and vital to Internet security, DDN advisories sometimes contain a more technical analysis of the problem at hand.

In March, 1991, the telnetd daemon on certain Sun distributions was found to be flawed. As the CERT advisory "SunOS in.telnetd Vulnerability" notes:  
*The Computer Emergency Response Team/Coordination Center (CERT/CC) has obtained information from Sun Microsystems, Inc. regarding a vulnerability affecting SunOS 4.1 and 4.1.1 versions of in.telnetd on all Sun 3 and Sun 4 architectures. This vulnerability also affects SunOS 4.0.3 versions of both in.telnetd and in.rlogind on all Sun3 and Sun 4 architectures. To our knowledge, a vulnerability does not exist in the SunOS 4.1 and 4.1.1 versions of in.rlogind. The vulnerability has been fixed by Sun Microsystems, Inc.*

**Cross Reference:** To view this CERT advisory in its entirety, visit [http://info.cert.org/pub/cert\\_advisories/CA-91%3A02a.SunOS.telnetd.vulnerability](http://info.cert.org/pub/cert_advisories/CA-91%3A02a.SunOS.telnetd.vulnerability)

**TIP:** If you buy an old Sun 3/60 over the Net, you will want to get the patches, which are included in the previous advisory.

Months later, it was determined that a specialized LAT/Telnet application developed by Digital Corporation was flawed. As the CERT advisory "ULTRIX LAT/Telnet Gateway Vulnerability" explains:

*A vulnerability exists such that ULTRIX 4.1 and 4.2 systems running the LAT/Telnet gateway software can allow unauthorized privileged access...Anyone who can access a terminal or modem connected to the LAT server running the LAT/Telnet service can gain unauthorized root privileges.*

**Cross Reference:** To view this CERT advisory in its entirety, visit [ftp://info.cert.org/pub/cert\\_advisories/CA-91%3A11.Ultrix.LAT-Telnet\\_gateway.vulnerability](ftp://info.cert.org/pub/cert_advisories/CA-91%3A11.Ultrix.LAT-Telnet_gateway.vulnerability)

The first Telnet problem that rocked the average man on the street was related to a distribution of the NCSA Telnet client for PC and Macintosh machines. So that there is no misunderstanding here, this was a client Telnet application that included an FTP server within it. The hole was fostered primarily from users' poor understanding of how the application worked. As articulated by the folks at DDN: The default configuration of NCSA Telnet for both the Macintosh and the PC has a serious vulnerability in its implementation of an FTP server...Any Internet user can connect via FTP to a PC or Macintosh running the default configuration of NCSA Telnet and gain unauthorized read and write access to any of its files, including system files.

The problem was related to a configuration option file in which one could enable or disable the FTP server. Most users assumed that if the statement enabling the server was not present, the server would not work. This was erroneous. By omitting the line (or adding the line option ftp=yes), one allowed unauthorized individuals read and write access to the files on your hard drive.

I hope this will settle the argument regarding whether a PC user could be attacked from the outside. So many discussions on Usenet become heated over this issue. The NCSA Telnet mishap was only one of many situations in which a PC or Mac user could be attacked from the void. So depending on the circumstances, the average user at home on his or her PC can be the victim of an attack from the outside. People may be able to read your files, delete them, and so forth.

What is more interesting is that even today, those using the NCSA Telnet application are at some risk, even if they only allow access to the FTP server by so-called authorized individuals. If a cracker manages to obtain from the target a valid username and password (and the cracker is therefore an authorized user), the cracker may then obtain the file FTTPASS. This is an authentication file where the usernames and passwords of users are stored. The encrypted passwords in this file are easily cracked.

The username in this file is not stored in encrypted form (in reality, few programs encrypt usernames). The password is encrypted, but the encryption scheme is very poorly implemented. For example, if the password is fewer than six characters, it will take only seconds to crack. In fact, it is so trivial to crack such passwords that one can do so with a 14-line BASIC program.

**Cross Reference:** The BASIC program that cracks passwords can be found at <http://www.musa.it/gorgo/txt/NCSATelnetHack.txt>

If you are a Mac or PC user currently using NCSA Telnet (with the FTP server), disallow all FTP access to anyone you do not trust. If you fail to heed this warning, you may get cracked. Imagine a scenario where a single individual on a network was using NCSA Telnet. Even if the rest of the network was reasonably secure, this would blow its security to pieces. Moreover, the application does not perform logging (in the normal sense) and therefore, no trail is left behind. Any network running this application can be attacked, disabled, or destroyed, and no one will be able to identify the intruder.

The most interesting Telnet hole ever discovered, though, was related to the environment variable passing option. The DDN bulletin on it was posted on November 20, 1995:  
*A vulnerability exists in some versions of the Telnet daemon that support RFC 1408 or 1572, both titled the "Telnet Environment Option," running on systems that also support shared object libraries...Local and remote users with and without local accounts can obtain root access on the targeted system.*

Many sites suffer from this vulnerability. To understand the problem, you must understand the term

environment. In UNIX vernacular, this generally refers to the environment of the shell (that is, what shell you might use as a default, what terminal emulation you are using, and so forth).

**NOTE:** DOS/Windows users can most easily understand this by thinking about some of the statements in their AUTOEXEC.BAT and CONFIG.SYS files. For example, variables are established using the SET command, as in SET PATH=C::C:WINDOWS; (the PATH environment variable is one of several that can be specified in the DOS environment). These statements define what your programming environment will be like when you boot into command mode. Some common environment variables that can be set this way are the shell you are using, the path, the time zone, and so forth.

## Changing the Environment

In UNIX, you can view or change the environment by using either the setenv or printenv command. Here is an example of what one might see on such an instruction:

```
> setenv

ignoreeof=10
HOSTNAME=samshacker.samshack.net
LOGNAME=tr
MINICOM=-c on
MAIL=/spool/mail/samshack
TERM=ansi
HOSTTYPE=i386-linux
PATH=/usr/local/bin:/bin:/usr/bin:./sbin:/usr/sbin:./
HOME=/usr/local/etc/web-clients/samshacker/./
SHELL=/bin/bash
LS_OPTIONS=---8bit --color=tty -F -T 0
PS1=h:w$
PS2=>
TAPE=/dev/nftape
MANPATH=/usr/local/man:/usr/man/preformat:/usr/man:/usr/X11/man:/usr/openwin/man
LESS--MM
OSTYPE=Linux
OPENWINHOME=/usr/openwin
SHLVL=2
BASH=/bin/bash
LS_COLORS=
_=/bin/csh
PWD=/usr/local/etc/web-clients/samshacker/./
USER=tr
HOST=samshack
```

This listing is a very extensive output of the command on a machine on which a virtual domain has been established. A more manageable (and more easily explained)

version can be taken from a bare shell machine. Here is the output:

```
samshacker% /usr/ucb/printenv
HOME=/home/hacker
HZ=100
LOGNAME=hacker
MAIL=/var/mail/hacker
PATH=/usr/bin:
SHELL=/sbin/sh
TERM=ansi
TZ=US/Pacific
PWD=/home/hacker
USER=hacker
```

This output is from a SPARCstation 10 on which I set up a mock shell account (the first output was from a Linux box). This is a very stripped-down environment. The PATH statement (line 6) points only to /usr/bin. In practice, this is impractical because there are many more binaries on a UNIX system than those located in /usr/bin. For example, there are binaries located in /usr/sbin, /usr/bin/X11, and so forth. You can see, for example, that even the command given (setenv) was done by issuing the absolute path statement (/usr/ucb/setenv). In practice, I would have (within a day or so) set a much longer path, pointing to man pages, binaries, and perhaps even include directories.

**NOTE:** The PATH statement in UNIX works almost exactly as it does in DOS. Directories that you intend to be in the path must be articulated on the PATH statement line and separated by colons (instead of semicolons). By articulating these on the PATH line, you give the user access to commands within these directories (no matter which directory the user is currently located in).

### Terminal Emulation

Other variables set in the preceding statements include HOME, MAIL, SHELL, and TERM. TERM, one of the most important variables, expresses the type of terminal emulation that you will be using. Because not all readers know what terminal emulation is, I want to quickly explain it.

Years ago, the majority of servers were mainframes. In those days, users did not have powerful PCs attached to the mainframe; they had terminals, which were (usually) boxes without hard drives. These were screens attached to keyboards. Behind terminals were a series of connectors, which might offer different methods of connection. One popular method was a bare-bones serial connection (we're talking primitive here: a straight serial-to-serial interface). Other terminals might sport hardware options such as Ethernet connections.

In any event, these terminals had very little functionality (at least in comparison to the average PC). Contained on the main board of such a terminal was a small portion of memory and firmware (software hardwired into the board itself). This firmware would grant the user several options. For example, one could set the speed and type of connection, the need for local echo, and so forth. Sometimes, there were options to set the type of printer that might be used or even what port the data was to be sent from.

**TIP:** Such terminals are still sold on certain Usenet newsgroups. If you are a student with limited funds and you have been granted some form of Ethernet or even serial connection to your college's server, and if that server account is a shell account, get a terminal. For a mere \$25-40, you can get high-speed access to the Internet. True, you cannot generally save materials to a disk, but you can print what is currently on the screen. You will not believe how quickly the screen will update. It is the absolutely ideal situation for Internet Relay Chat (IRC). These boxes are small, cheap, and fast.

The two best-known terminals were the Tektronix 4010 and the VT100 (also the IBM 3270, which is a bit different). Each had a set number of characters per line and lines per screen that could be displayed. In fact, most terminals usually had two settings. As terminals became more fancy, one could even set columns and, eventually, graphics (the Tektronix was graphics oriented).

Because these terminals became the standard method of connecting to mainframes, they also bled into the UNIX world. As such, all UNIX operating systems have keyboard and screen mappings for terminals. Mappings are descriptions of the screen and the keyboard settings (for example, how many lines and columns per screen or, more importantly, what Ctrl key sequences represent special characters). These are required because certain terminals use more keys than are offered on the standard PC or Mac keyboard. In addition to the regular typewriter keyboard and F function keys, there may be P

keys that perform special actions, including the activation of menus and the navigation of the screen cursor in databases. To make up for this on PC, Mac, or even some UNIX keyboards, Esc or Ctrl sequences are defined. These are combinations of keystrokes that equal a P key. These key assignments are called key bindings, which are statements made within the program code that define what happens if this or that key combination is executed. Key bindings are a big part of programming, especially in C where you offer a semi-graphical interface (for example, where you use Borland's famous TurboVision libraries to create drop-down menus in a DOS application).

One can generally define key bindings in a program (at least, in a well written one). This gives the user application-level control over which keys do what. For example, perhaps the user can set the binding of the Ctrl key plus the letter F to perform a variety of functions. Some specialized applications actually ask the user to do so before launching the program for the first time. There is one such program--a freeware editor for UNIX, written in Germany--that allows you to completely remap the keyboard.

In UNIX, terminal mappings are generally stored in a file called termcap. The termcap library, reportedly introduced with Berkeley UNIX, is a very important addition to the system. Without it, many machines would not communicate well with each other. For example, if you perform a fresh install of a Linux operating system and do nothing to alter the TERM variable, it will be set to Linux. If you then Telnet to a SPARCstation (or other machine that also has its default TERM configuration), you will be unable to clear the screen with the well-known command clear. This is because the two terminal emulation settings are incompatible. Furthermore, if you try to execute a program such as PINE--which relies on compatible terminal types--the program will exit on error, reporting that the terminal is not supported. (SysV systems traditionally use terminfo as opposed to termcap.)

**CAUTION:** Many distributions of UNIX have complete termcap listings, which sometimes contain hundreds of terminal emulations. If you are new to UNIX and are toying with the idea of altering your termcap entries, be extremely careful. You may end up with bizarre results. In some cases, what once looked like nicely formatted text may appear as strange, disjointed, scattered blocks of words that are largely illegible. Study the man page before fiddling with your termcap file.

Many different environmental variables can be set. These variables can strongly influence how a remote machine will receive, process, and support your remote Telnet connection. Thus, the Telnet protocol was designed to allow the passing of certain environment variables at the time of the connection. As explained in RFC 1408:

*Many operating systems have startup information and environment variables that contain information that should be propagated to remote machines when Telnet connections are established. Rather than create a new Telnet option each time someone comes up with some new information that they need propagated through a Telnet session, but that the Telnet session itself doesn't really need to know about, this generic information option can be used.*

**Cross Reference:** To view RFC 1408 in its entirety, visit [http://sunsite.auc.dk/RFC/rfc/rfc\\_1408.html](http://sunsite.auc.dk/RFC/rfc/rfc_1408.html)

This Telnet security hole was based on the capability of a Telnet server to receive, respond to, and authorize the passing of these environment variables. Because this option was so prominent in the UNIX system, an incredible number of platforms were vulnerable to this attack.

This vulnerability is more common than one would expect. In a rather engrossing report, one firm, Novatech, posted the results of an actual security audit of a network with 13 hosts. In it, the Telnet vulnerability appears, as do 138 other holes. The most extraordinary thing is that the site had already been assessed as having a clean bill of health, complete with a firewall. As Novatech's sample audit report notes:

*This is a copy of a actual attack report with definitions and possible rectifications of actual problems found. The network had a state of the art firewall installed and had been checked by CERT. As you can see there were many small problems and a number of larger ones as well. This was not the fault of the systems administration but of a mix that systems change and need constant attention and the lack of knowledge of how intruders gain access (a specialist field). We are able to check your system for nearly 390 different forms of access vulnerability all of which are Internet only type access.*

**Cross Reference:** For those who have a "let's wait and see" attitude about security, I suggest that you go immediately to this site and view the results. They are astonishing. See the results of the audit at <http://www.novatech.net.au/sample.htm>

The line that reveals the Telnet environment option vulnerability reads as follows:

*Dynamic Linker Telnet Vulnerability [High Risk]2*

This line reports that a Telnet vulnerability in the high risk category was found (in the audit cited previously, this vulnerability was found on two hosts within the same subnet). [High Risk]2 refers to the level of risk the hole represents. This is an extremely high risk vulnerability. Remember, this was found on a host with a state-of-the-art firewall!

To understand the method, you must understand precisely what options can be passed from the client to the server. One of these involves the passing of a custom libc.

**NOTE:** libc is the standard C library. A full distribution of libc commonly contains header and include files for use in C programming. All UNIX flavors have (or should have) this library installed. It is a requisite for compiling programs written in the C programming language. As Sam Hartman of MIT notes in his article "Telnet Vulnerability: Shared Libraries":

*The problem is that telnetd will allow the client to pass LD\_LIBRARY\_PATH, LD\_PRELOAD, and other run-time linker options into the process environment of the process that runs login.*

**Cross Reference:** Find Hartman's article on the Web at [http://geek-girl.com/bugtraq/1\\_995\\_4/0032.html](http://geek-girl.com/bugtraq/1_995_4/0032.html)

By passing the LD\_LIBRARY\_PATH environment option to the server, the cracker can add to this search path a custom directory (and therefore a custom library). This can alter the dynamic linking process, greatly increasing the chances of a root compromise.

**NOTE:** Hartman noted that if the target was using a Kerberos-aware telnetd, only users with a valid account on the remote box could actually implement the attack. My guess, however, is that the larger majority of machines out there are not using such a means of secure Telnet.

One interesting note about this hole: It was determined that one could identify Telnet sessions in which the environment variables had been passed by executing a ps instruction. However, one individual (Larry Doolittle) determined that on certain flavors of UNIX (Linux, specifically), one has to be root to ID those processes. In response to the Hartman report, Doolittle advised:

*Recent Linux kernels do not allow access to environment strings via ps, except for the user him/herself. That is, /proc/\*/environ is protected 400. This could confuse people reading your instructions, since they would see environments for their own process but not root's. To verify environment strings of login, you need to run ps as root.*

**Cross Reference:** Find Larry Doolittle's article on the Web at [http://geek-girl.com/bugtraq/1\\_995\\_4/0042.html](http://geek-girl.com/bugtraq/1_995_4/0042.html)

Here are patches for various distributions of telnetd:

- DEC. (OSF/1):  
[ftp://ftp.servi ce.digital.com/public/osf/v3.2c/ssrt0367\\_c032](ftp://ftp.servi ce.digital.com/public/osf/v3.2c/ssrt0367_c032)
- A compressed version is available at  
<ftp://ftp.ox.ac.uk/pub/comp/security/software/patches/telnetd/>
- Linux:  
<ftp://ftp.ox.ac.uk/pub/comp/security/software/patches/telnetd/linux/telnetd>
- Red Hat:  
<http://www.io.com/~ftp/mirror/linux/redhat/redhat/updates/i386/NetKit -B-0.09-1.1.i386.rpm>
- SGI (IRIX):  
<ftp://sgigate.sgi.com/security/>

**NOTE:** Although patches have been issued for this problem, some other Telnet-related modules and programs may still be affected. In Late February, 1997, in.telnetd was reported as vulnerable to the LD\_PRELOAD passing on some platforms, including Linux. There is reportedly a patch for this problem, and it has been uploaded to <ftp://sunsite.unc.edu>.

Garden-variety Telnet is not a particularly secure protocol. One can easily eavesdrop on Telnet sessions. In fact, there is a utility, called ttynsnoop, designed for this purpose. As describe by its author, Carl Declerck: [ttynsnoop] allows you to snoop on login tty's through another tty-device or pseudo-tty. The snoop-tty becomes a "clone" of the original tty, redirecting both input and output from/to it.

**Cross Reference:** Declerck's README for ttynsnoop 0.12 (alpha) can be found on the Web at <http://ion.apana.org.au/pub/linux/sources/admin/ttynsnoop-0.12.README>

**NOTE:** ttynsnoop is not simply a Telnet-specific snooper; it snoops on the tty, not the Telnet protocol. A network sniffer like sniffit can also be used (and is probably more suitable) to sniff the Telnet protocol.

Telnet sessions are also especially sensitive. One reason for this is that these sessions are often conducted in an island-hopping pattern. That is, the user may Telnet to one network to tidy his or her Web page; from there, the user may Telnet to another machine, and another machine, and so on. If a cracker can snoop on such a session, he or she can obtain login IDs and passwords to other systems.

### **Aren't These Attacks No Longer Effective?**

No; this is due primarily to a lack of education. The environment option attack described previously is quite effective on many systems in the void. This is so even though advisories about the attack are readily available on the Internet.

### **Telnet as a Weapon**

Telnet is an interesting protocol. As explained earlier, one can learn many things using Telnet. For example, you can cull what version of the operating system is being run. Most distributions of UNIX will report this information on connection. It is reported by at least one authoritative source that various scanners use the issue information at connect to identify the type of system (SATAN being one such scanner). The operating system can generally be determined by attacking any of these ports:

- Port 21: FTP
- Port 23: Telnet (Default)
- Port 25: Mail
- Port 70: Gopher
- Port 80: HTTP

**NOTE:** Although there are only five listed ports, one can connect to the majority of TCP/IP ports by

initiating a Telnet session. Some of these ports will remain in an entirely passive state while the connection is active, and the user will see nothing happen in particular. This is so with port 80 (HTTP), for example. However, you can issue perfectly valid requests to port 80 using Telnet and if those requests are valid, port 80 will respond. (The request needn't necessarily be valid. Issuing an erroneous GET instruction will elicit a lively response from the Web server if the request is sufficiently malformed.)

In their now-famous paper, "Improving the Security of Your Site by Breaking Into It," Dan Farmer and Wietse Venema point out ports that can be attacked. Specifically, they address the issue of port 6000:

*X windows is usually on port 6000...If not protected properly (via the magic cookie or xhost mechanisms), window displays can be captured or watched, user keystrokes may be stolen, programs executed remotely, etc. Also, if the target is running X and accepts a Telnet to port 6000, that can be used for a denial of service attack, as the target's windowing system will often "freeze up" for a short period of time.*

**Cross Reference:** "Improving the Security of Your Site by Breaking Into It" can be found on the Web at [http://stos-www.cit.cornell.edu/Mark\\_html/Satan\\_html/docs/admin\\_guide\\_to\\_cracking.html](http://stos-www.cit.cornell.edu/Mark_html/Satan_html/docs/admin_guide_to_cracking.html)

In the paper by Farmer and Venema are many attacks implemented with Telnet alone or in conjunction with other programs. One such attack involves an X terminal:

*X Terminals are generally diskless clients. These are machines that have the bare minimum of hardware and software to connect to an X server. These are most commonly used in universities and consist of a 17" or 19" screen, a base, a keyboard and a mouse. The terminal usually supports a minimum of 4 megabyte of RAM but some will hold as much as 128 megabytes. X terminals also have client software that allows them to connect to the server. Typically, the connection is via fast Ethernet, hardwired to the back of the terminal. X Terminals provide high-speed connectivity to X servers, coupled with high-powered graphics. These machines are sold on the Internet and make great "additional" terminals for use at home. (They are especially good for training.)*

The Farmer-Venema X terminal technique uses a combination of rsh and Telnet to produce a coordinated attack. The technique involves stacking several commands. The cracker uses rsh to connect to the X terminal and calls the X terminal's Telnet client program. Finally, the output is redirected to the cracker's local terminal via the specification of the DISPLAY option or variable. Another interesting thing that Telnet can be used for is to instantly determine whether the target is a real or virtual domain (this can be done through other methods, but none perform this function quite as quickly). This can assist a cracker in determining exactly which machine he or she must crack to reach your resources or, more precisely, exactly which machine he or she is engaged in cracking.

Under normal circumstances, a real domain is a domain that has been registered with InterNIC and also has its own dedicated server. Somewhere in the void is a box with a permanent IP address, and that box is attached permanently to the Internet via 28.8Kbps modem, ISDN, 56Kbps modem, frame relay, T1, T3, ATM, or perhaps, if the owner spares no expense, SONET. As such, when you Telnet to such a real site, you are reaching that machine and no other.

Virtual domains, however, are simply directories on a real server, aliased to a particular domain name. That is, you pay some ISP to register your domain name and create a directory on its disk where your virtual domain exists. This technique allows your\_company.com to masquerade as a real server. Thus, when users point their browsers to www.your\_company.com, they are reaching the ISP's server. The ISP's server redirects the connection request to your directory on the server. This virtual domain scheme is popular for several reasons, including cost. It saves your company the trouble of establishing a real

server and therefore eliminates some of these expenses:

- Hardware
- Software
- 24-hour maintenance
- Tech support

Basically, you pay a one-time fee (and monthly fees thereafter) and the ISP handles everything. To crackers, this might be important. For example, if crackers are about to crack your domain--without determining whether your machine is truly a server--they may get into trouble. They think they are cracking some little machine within your internal offices when in fact, they are about to attack a large, well-known network provider.

Telnet instantly reveals the state of your server. When a cracker initiates a Telnet connection to your\_company.com (and on connect, sees the name of the machine as a node on some other, large network), he or she immediately knows that your address is a virtual domain.

Moreover, Telnet can be used for other nefarious purposes. One is the ever-popular brute-force attack. I am not sure why brute-force attacks are so popular among young crackers; almost all servers do some form of logging these days. Nevertheless, the technique has survived into the 1990s. These attacks are most commonly initiated using Telnet clients that have their own scripting language built in. Tera Term is one such application.

Tera Term sports a language that allows you to automate Telnet sessions. This language can be used to construct scripts that can determine valid usernames on a system that refuses to cough up information on finger or sendmail-expn queries. Versions of Telnet reveal this information in a variety of ways. For example, if a bogus username is given, the connection will be cut. However, if a valid username is given, a new login: prompt is reissued.

**Cross Reference:** Tera Term can be found on the Web at <http://tu cows.phx.cox.com/files /ttermv13.zip>

Moreover, Telnet is a great tool for quickly determining whether a particular port is open or whether a server is running a particular service. Telnet can also be used as a weapon in denial-of-service attacks. For example, sending garbage to certain ports on an NT Web server under IIS can cause the targeted processor to jump to 100 percent utilization. Initiating a Telnet session to other ports on an NT Web server can cause the machine to hang or crash. This is particularly so when issuing a Telnet connection request to port 135.

**Cross Reference:** A fix for this problem, issued by Microsoft, can be found at <ftp://ftp.microsoft.com/bussys/winnt/winnt-public/fixes/usa/nt40/hotfixes-postS>

One can also crash Microsoft's Internet Information Server by Telnetting to port 80 and issuing a GET.../... request. Reportedly, however, that problem was remedied with the Microsoft Windows NT Service Pack 2 for Windows NT 4.0. If you do not have that patch/service pack, get it. A good treatment of this and other problems can be found in the Denial of Service Info post, posted by Chris Klaus of Internet Security Systems. In it, Klaus writes:

*The file sharing service if available and accessible by anyone can crash the NT machine and require it to be rebooted. This technique using the dot...dot bug on a Windows 95 machine potentially allows anyone to gain access to the whole hard drive...Solution: This vulnerability is documented in Microsoft Knowledge Base article number Q140818 last revision dated March 15, 1996. Resolution is to install the latest service pack for Windows NT version 3.51. The latest service pack to have the patch is in*

*service pack 4.*

**Cross Reference:** Visit the Denial of Service Info post at [http://geek-girl.com/bugtraq/1996\\_2/0052.html](http://geek-girl.com/bugtraq/1996_2/0052.html)

**NOTE:** This was only a vulnerability in the Internet Information Server 2.0 World Wide Web server (HTTP). Later versions of IIS are reportedly clean.

Finally, Telnet is often used to generate fakemail and fakenews. Spammers often use this option instead of using regular means of posting Usenet messages. There are certain options that can be set this way that permit spammers to avoid at least some of the screens created by spam-killing robots on the Usenet network.

### **Summary**

Telnet is a very versatile protocol and, with some effort, it can be made secure. (I personally favor SSH as a substitute, for it prevents against snooped Telnet sessions.) Nevertheless, Telnet is not always secure out of the box. If you are using older software (pre 1997), check whether the appropriate patches have been installed.

Telnet can also be used in a variety of ways to attack or otherwise cull information from a remote host (some of those are discussed in this chapter). By the time this book is released, many more Telnet attack techniques will have surfaced. If you run a network and intend to supply your users with Telnet access, beware. This is especially so on new Telnet servers. These new servers may have bugs that have not yet been revealed. And, because Telnet is so interactive and offers the user so much power to execute commands on remote machines, any hole in a Telnet distribution is a critical one. It stands in the same category as FTP or HTTP in this respect (or is perhaps even worse).