Chapter 1

Introduction to Netcat

Solutions in this chapter:

- **Introduction**
- **Installation**
- **Options**
- **Basic Operations**

- ☑ **Summary**
- ☑ **Solutions Fast Track**
- ☑ **Frequently Asked Questions**

Introduction

Originally released in 1996, Netcat is a networking program designed to read and write data across both Transmission Control Protocol TCP and User Datagram Protocol (UDP) connections using the TCP/Internet Protocol (IP) protocol suite. Netcat is often referred to as a "Swiss Army knife" utility, and for good reason. Just like the multi-function usefulness of the venerable Swiss Army pocket knife, Netcat's functionality is helpful as both a standalone program and a back-end tool in a wide range of applications. Some of the many uses of Netcat include port scanning, transferring files, grabbing banners, port listening and redirection, and more nefariously, a backdoor.

There is some debate on the origin of the name Netcat, but one of the more common (and believable) explanations is that Netcat is simply a network version of the vulnerable *cat* program. Just as cat reads and writes information to files, Netcat reads and writes information across network connections. Furthermore, Netcat is specifically designed to behave as cat does.

Originally coded for UNIX, and despite not originally being maintained on a regular basis, Netcat has been rewritten into a number of versions and implementations. It has been ported to a number of operating systems, but is most often seen on various Linux distributions as well as Microsoft Windows.

NOTE

For the sake of this chapter, we will work with Netcat in two different operating systems: Windows XP and UNIX/Linux. Windows is in a category by itself. The UNIX and Linux variants are essentially the same thing. Furthermore, the differences within the various Linux distributions are minimal. Also be aware that there are at least two slightly different implementations: the original UNIX release of Netcat as well as a more recent implementation called GNU Netcat.

In the 2006 survey of users of the nmap-hackers mailing list, Netcat was the 4th rated tool overall. In fact, in three consecutive surveys (2000, 2003, and 2006) Netcat was rated no. 2, no. 4, and no. 4 despite the considerable proliferation of more advanced and more powerful tools. In the day and age when users seek the latest and greatest of the edge tools, Netcat's long reign continues.

The goal of this chapter is to provide you with a basic understanding of Netcat. To that end, we'll start with installation and configuration (Windows and UNIX/Linux), and follow up with an explanation of the various options and an understanding of Netcat's basic operations. As we explore some of Netcat's operations, we'll introduce various chapters in the book that cover those operations in greater detail. To that end, consider this introductory chapter as the starting point for your journey.

Installation

Netcat being a rather simple and small program, it is no wonder that installation is straightforward, regardless of the operating system you choose. The Windows port of Netcat comes already compiled in binary form, so there is no true installation required. As previously noted, there are two common UNIX/Linux implementations: the original UNIX version as well as GNU Netcat. Virtually all flavors of UNIX/Linux will come with one of these implementations of Netcat already compiled; however, it is useful to know how to install it if necessary. Furthermore, depending upon your particular implementation, you may need to re-compile Netcat to obtain full functionality.

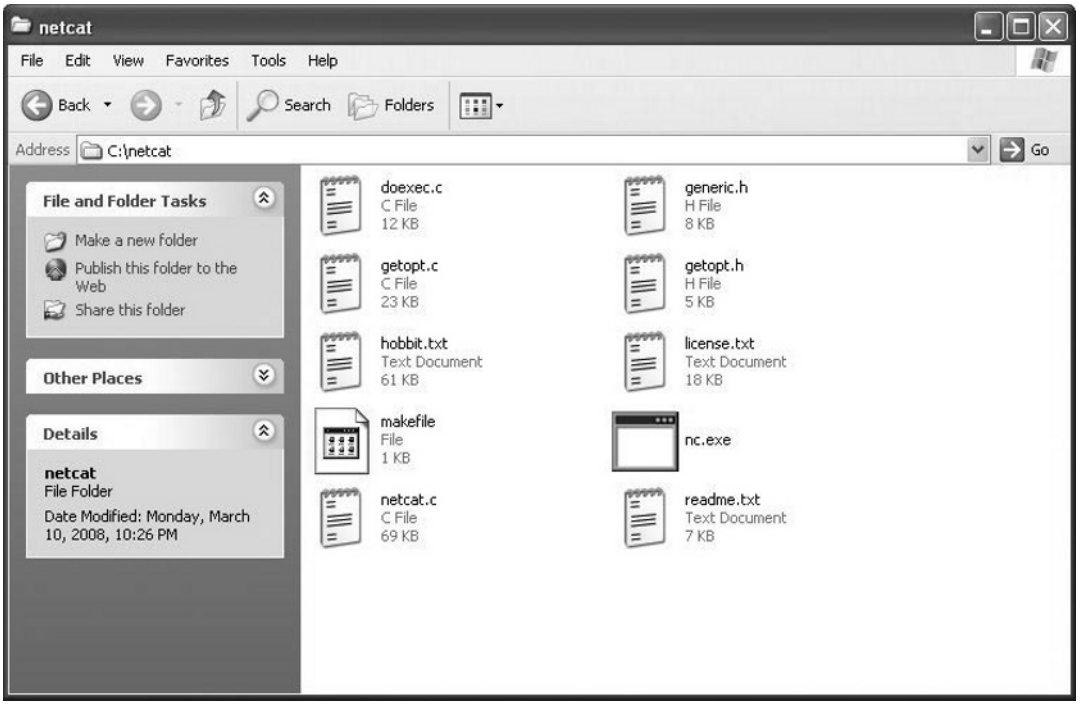
Windows Installation

Windows installation couldn't be any easier. Simply download the zip file from www.vulnwatch.org/netcat/nc111nt.zip. Unzip to the location of your choice, and you're finished (see Figure 1.1). There are a couple of important files to check out: *hobbit.txt* is the original documentation, *readme.txt* is an explanation of a security fix from version 1.10 to 1.11, and *license.txt* is the standard GNU general public license.

NOTE

Remember that Netcat is a command-line tool. Double-clicking on the **nc.exe** icon from Windows Explorer will simply run Netcat without any switches or arguments and will present you with a *cmd line:* prompt. You can run Netcat this way, but once the instance is complete the window will close immediately. This is not very helpful, especially if you want feedback. It is much easier to use from the command line directly. **Start | Run | cmd.exe. nc –h** will show you the help screen for further guidance.

Figure 1.1 Netcat Installation Under Windows



Are You Owned?

My Anti-virus said Netcat was a Trojan!

Netcat's potent communications ability is not limited to network administrators. Penetration testers use Netcat for testing the security of target systems (for example, Netcat is included in the Metasploit Framework). Malicious users use Netcat (or one of the many variations of it) as a means of gaining remote access to a system. In this sense, it is understandable why many anti-virus programs have labeled Netcat as a "trojan" or a "hacktool."

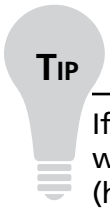
Some anti-virus programs may try to prevent you from installing Netcat, or even try to prevent you from downloading Netcat or another application that includes Netcat. As with virtually any tool, there is no internal moral compass that

limits its use for only legitimate purposes. Your decision in this case is simply to determine if Netcat was purposely downloaded and installed by you (and thus not a threat), or surreptitiously installed by a malicious user for nefarious purposes.

You may consider configuring your anti-virus program to exclude a particular directory where you install Netcat when it scans or auto-protects your file system. Of course, you need to be aware of the dangers associated with this.

Linux Installation

Many mainstream Linux distributions come with Netcat already compiled and installed. Others have at least one or more versions of Netcat available as a pre-compiled package. To determine the version of Netcat, simply type **nc –h** or **netcat –h**. The original UNIX version will return a version line of *[v1.10],* while the GNU version will return *GNU Netcat 0.7.1*, a rewrite of the famous networking tool. Even if Netcat is already installed on your system, you may not want to skip this section. Many pre-installed, pre-compiled, or packaged versions of Netcat that come with a Linux distribution are not compiled with what is called the GAPING_SECURITY_HOLE option (this allows Netcat to execute programs with the *–e* option). These are typically "safe" compilations of the original Netcat source code. The GNU version of Netcat automatically compiles with the *–e* option enabled, so by installing this version no additional configuration is necessary. Despite this, all other functionality of the original Netcat remains intact. Of course, executing programs is what makes Netcat such a powerful tool. Furthermore, many of the demonstrations in this book take advantage of the *–e* option, so you may want to consider re-compiling if you wish to follow along.



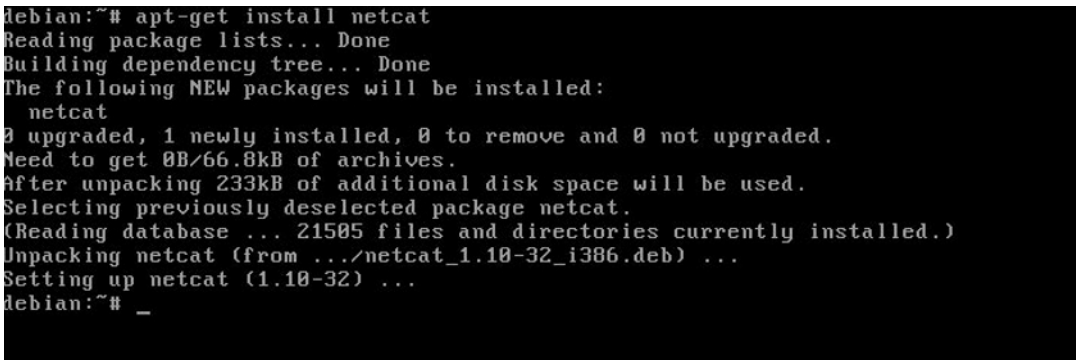
TIP

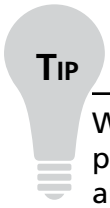
If you have Netcat already installed and are unsure about whether or not it was already compiled with the *–e* option, simply run Netcat with the *–h* (help) switch to display the help screen. If *–e* is among your options, then Netcat was installed with this option. If *–e* is not among the options, you'll have to re-compile Netcat, or use the GNU version.

Installing Netcat as a Package

Most distributions have Netcat pre-compiled as a package. Some may even have more than one version, or different implementations with different functionality. Note, as we did above, that these packages are not likely to have the execute option enabled (and generally for good reason). For example, to install Netcat from a pre-compiled package on a Debian system, type **apt-get install netcat** (see Figure 1.2).

Figure 1.2 Installing Netcat as a Package





TIP

While beyond the scope of this book, it is important to make sure that your package sources are up to date. For example, with Debian and APT, sources are listed in */etc/apt/sources.list*. Furthermore, be sure to keep your list of packages updated with the *apt-get update* command. For other distributions, check your documentation for sources and updating package lists.

Figure 1.2 shows the simple Netcat package installation process. Notice that in this case, Netcat has no dependencies, even on this minimalist install of Debian. Also notice the package name *netcat_1.10-32_i386.deb*. The key here is 1.10, which is the version information. This confirms that this package is in fact compiled from the original UNIX Netcat as opposed to GNU Netcat. Furthermore, *nc –h* reveals that this package has been pre-compiled with the all-powerful *–e* option.

NOTE

To install Netcat via package for other flavors of Linux, consult your documentation for the specific method of install pre-compiled packages.

Installing Netcat from Source

If you want to compile it from source code, you have two options, which are more or less the same thing, with one important exception. First is the original UNIX Netcat, which can be found at www.vulnwatch.org/netcat. Your second option is GNU Netcat, which is located at netcat.sourceforge.net. The key difference between these two versions of Netcat is that the original Netcat requires manual configuration to compile with the *–e* option, while GNU Netcat does it automatically. This manual configuration is not complicated, but can be tricky if you're not used to looking at source code.

If you're relatively new to Linux and compiling a program from the source code seems daunting, rest easy. The entire installation process is simple and easy, and takes all of a few minutes. For the sake of this installation, and so we can install Netcat

without having to manually configure the –*e* option, we'll download, configure, and compile the GNU version of Netcat:

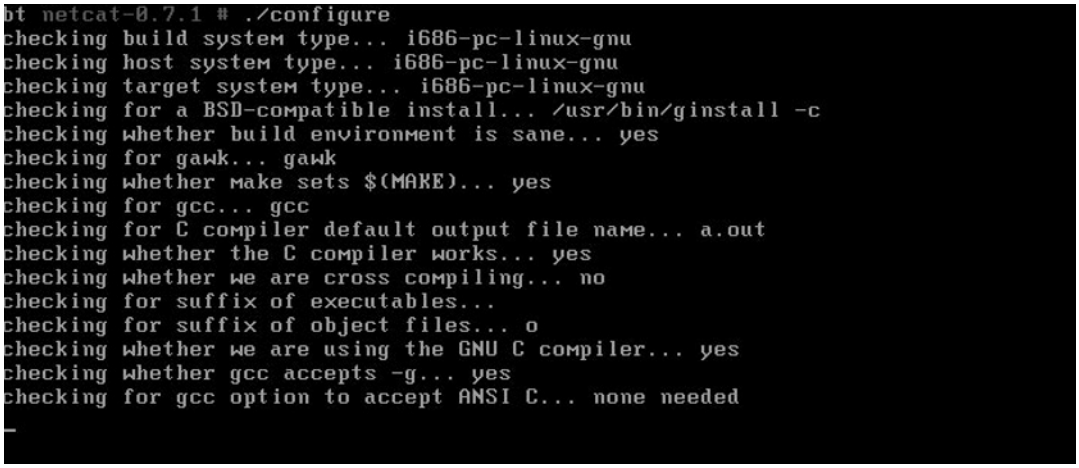
```
wget http://osdn.dl.sourceforge.net/sourceforge/netcat/netcat-0.7.1.tar.gz
tar -xzf netcat-0.7.1.tar.gz
cd netcat-0.7.1
./configure
make
make install
```

Your first step toward installation is to download the source. You can choose to use the simple *wget* command-line utility, as shown in Figure 1.3, or download via a Web browser or other means.

Figure 1.3 Downloading Netcat

Next, un-tar the archive and change into the newly created Netcat directory. Then, configure Netcat (see Figure 1.4). The configure script creates a configuration file called Makefile.

Figure 1.4 Configuring Netcat



The *make* command builds the binary (Netcat executable file) from the Makefile created in the previous step.

The *make install* command installs Netcat to your system. Note that running *make install* does require root privileges. That's it! You'll find that, more often than not, this is a fairly common set of procedures for installing programs to Linux from source code.

NOTE

If you encounter any errors during the installation process, they are most likely to occur during the last two steps. If this is the case, you may not have the correct packages installed to properly compile Netcat. This is most likely to happen if you have a minimalist installation. Be sure to check out the references to your particular installation to ensure the proper packages are installed.

Depending upon the version of Netcat that you install, the executable binary may be *nc* or *netcat*. For the sake of conformity throughout this chapter, we'll use *nc*.

Confirming Your Installation

Regardless of whether or not you choose to install the Windows or Linux version of Netcat, to confirm that Netcat installed correctly, type **nc –h** or **netcat –h** to display the help screen (see Figures 1.5 and 1.6). Notice there are a few differences in options. In the Windows version, *–L* represents a persistent listening mode (to be described later), while it represents a tunneling mode in the Linux version. Also, the Linux version includes *–V* (note the capital letter), which displays version information. The Windows version lacks this option. Finally, the Linux version includes *–x* (hexdump incoming and outgoing traffic), which is not included in the Windows version, but is implied by the *–o* option.

Figure 1.5 Netcat Installed in Windows

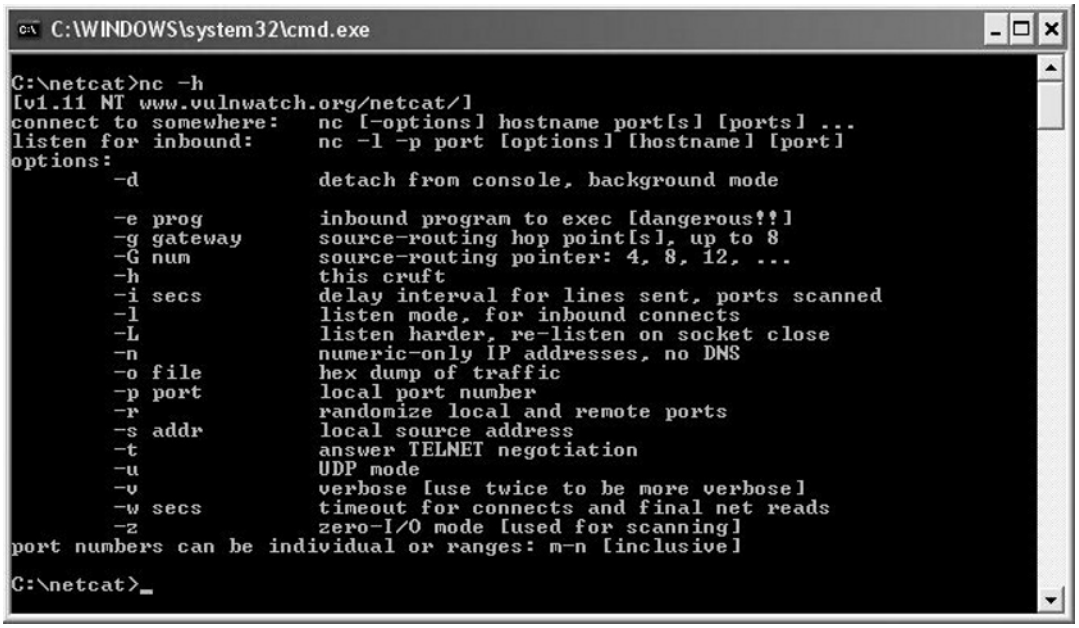
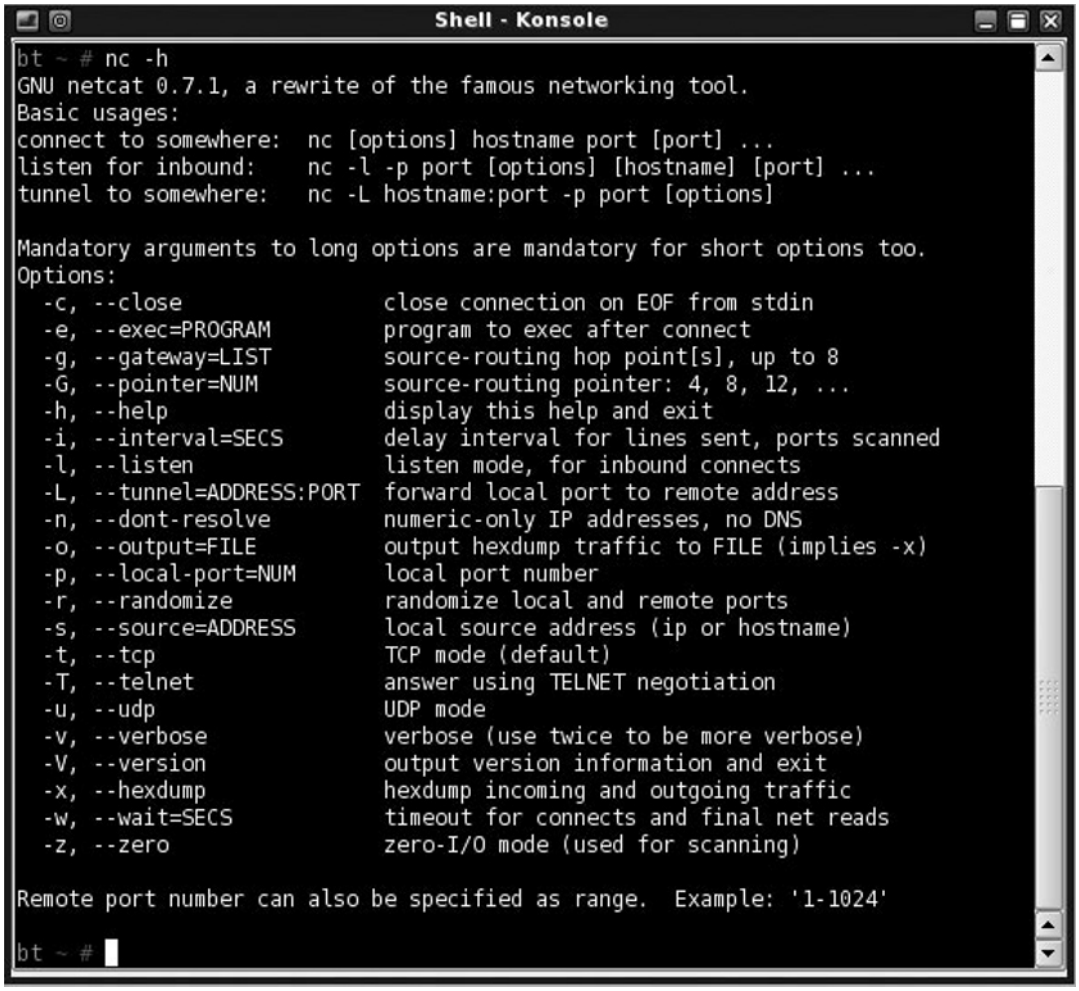


Figure 1.6 Netcat Installed in Linux



Netcat's Command Options

In this section, we'll talk about Netcat's two distinct modes of operation, as well as some of the most common options.

Modes of Operation

Netcat has two primary modes of operation, as a *client*, and as a *server*. The first two lines of the help screen in Figure 1.5 (below the version information) explain the proper syntax for each of these modes:

```
connect to somewhere: nc [-options] hostname port[s] [ports] …
listen for inbound: nc -l -p port [options] [hostname] [port]
```

Connect to somewhere indicates the syntax for Netcat's client mode. Typically, you're using Netcat as a client on your machine to obtain some sort of information from another machine. *Listen for inbound* indicates the syntax for Netcat's server mode. Notice the *–l* switch, which puts Netcat into listen mode. In this case, you're setting up Netcat to listen for an incoming connection. Netcat doesn't really care what mode it's using, and will do most anything you ask of it in either mode.

Common Command Options

In this section we'll talk about the most common options that you'll likely see used in the basic operations of Netcat. With a few exceptions (previously described and specifically noted in the text), these options are the same for both the Windows and Linux versions. Please refer to the individual chapters in this book for more advanced uses of Netcat's options depending upon what you're trying to accomplish. Remember that the *–l* option will determine Netcat's mode of operation. The command *nc –l* will put Netcat into server or listening mode, and *nc* by itself will run Netcat in client mode.

The first available option, *–c*, commands Netcat to close at end of file (EOF) from standard input (stdin). This option is only available in the Linux variant.

Netcat's next option is *–d*. This switch enables Netcat to be detached from the console and run in background mode. This is particularly useful if you don't want Netcat to open up a console window (especially if someone might be watching). Note that this option is only available in the Windows version.

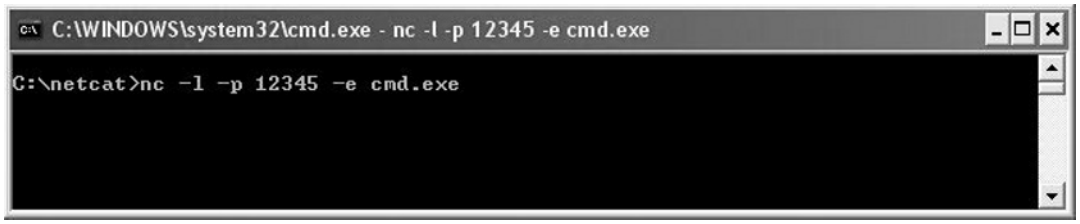
Netcat's most powerful option is undoubtedly *–e prog*. This option, available only in server mode, allows Netcat to execute a specified program when a client connects to it. Consider the following commands:

`nc -l -p 12345 -e cmd.exe` (Windows)

`nc -l -p 12345 -e /bin/bash` (Linux)

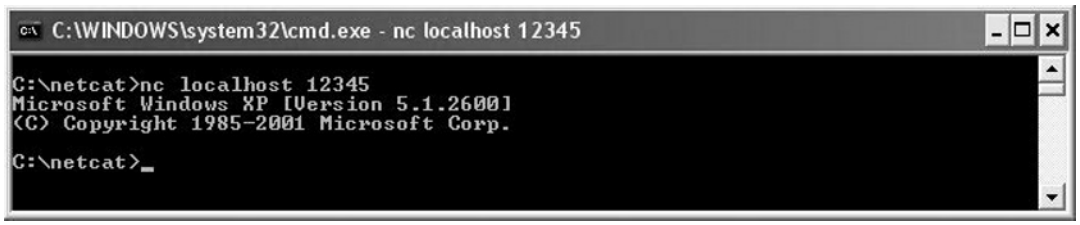
Both of these commands do essentially the same thing, but on different systems. The first command executes Netcat in server mode on local port 12345, and will execute *cmd.exe* (the Windows command shell) when a client connects to it. The second command does precisely the same thing, except that it executes a bash shell in Linux. To test this option, start Netcat in server mode (Figure 1.7):

Figure 1.7 Starting Netcat in server mode (Windows)



Open a second window, and start Netcat in client mode (Figure 1.8):

Figure 1.8 Starting Netcat in Client Mode (Windows to Windows)



After you hit **enter**, you are greeted with the Microsoft banner information and a new command prompt. This might seem underwhelming, but make no mistake about it: you're running this command prompt through Netcat. If you were running Netcat over a network instead of on the same computer, you would have direct shell access on the server. Type **exit** at the prompt, and you'll see that the Netcat server closes in the first window.

To start Netcat in server mode on a Linux box type **nc –l –p 12345 –e /bin/bash**.

Now open a command prompt in Windows and start Netcat in client mode (see Figure 1.9).

Figure 1.9 Starting Netcat in Client Mode (Windows to Linux)

Unlike when we connected to Windows, the Linux bash shell does not echo any characters to your screen. Try using *uname –a* to display the system information. In this case, it confirms we are connected to a Linux box because it accepted a common Linux command. Furthermore, it returned the relevant system information: kernel name and version, processor information, and so forth.

WARNING

It cannot be stressed enough how powerful the *–e* option is in Netcat. By allowing an incoming client to connect to Netcat, you are giving that client direct shell access. Furthermore, there is no user identification or authentication process associated with this access. It is important to understand that while you might have legitimate reasons to do this, there are undoubtedly many nefarious uses for such an option. Chapter 5, *The Dark Side of Netcat*, will explore this option in much further detail.

The *–g* and *–G* options allow you to configure Netcat to use source routing. In source routing, the sender specifies the route that a packet takes through a network. Since most routers block source-routed packets, this option is more or less obsolete.

As we have already seen, the help screen is displayed with the *–h* switch.

To set a delay interval (between lines sent or ports scanned), use the *–i* option. This may be useful for scanning ports if rate limiting is encountered.

To place Netcat in listening mode, or as we have called it in this chapter, server mode, use the *–l* option. Normally, Netcat is a single-use program. In other words, once the connection is closed, Netcat closes and is no longer available. However the *–L* option reopens Netcat with the same command line after the original connection is closed:

```
nc -l -p 12345 -e cmd.exe -L
```

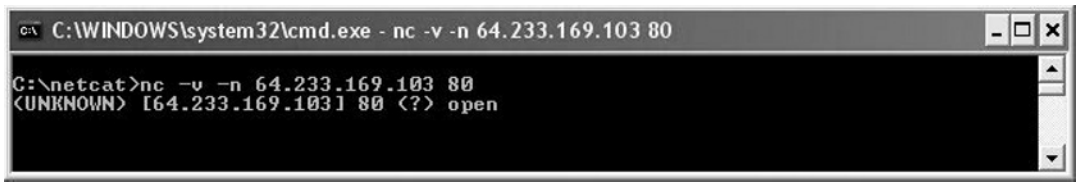
Connecting to this instance of Netcat will open a command shell to the client. Exiting that command shell will close the connection, but the *–L* option will open it up again.

NOTE

The *–L* "persistent" option is only available in the Windows version of Netcat. However, you can overcome this limitation in Linux with a bit of scripting. To complicate matters, the GNU version of Netcat uses *–L* for tunneling. This option allows you to forward a local port to a remote address.

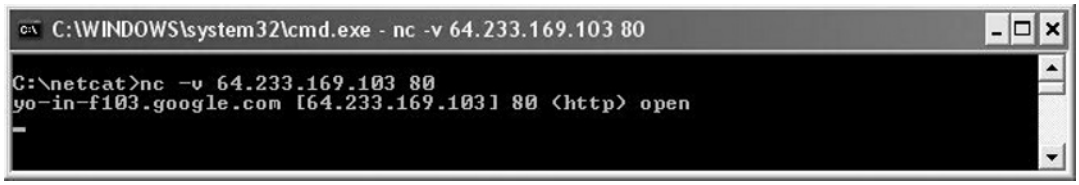
To allow numeric-only IP addresses and no reverse lookup, use the *–n* option. It is also useful to know what Netcat will do if you *don't* include the *–n* option. Without *–n* (and assuming you have included the *–v* switch), Netcat will display forward and reverse name and address lookup for the specified host. Let's take a look at an example. In Figure 1.10, we've included the *–n* option:

Figure 1.10 Netcat with the *–n* Option



With the *–n* option enabled, Netcat accepts only a numeric IP address and does no reverse lookup. Compare to the same command line, without enabling *–n* (Figure 1.11):

Figure 1.11 Netcat without the –n Option



Without the *–n* option, Netcat does a reverse lookup and tells us that the specified IP address belongs to Google. It is not uncommon for Netcat to display warnings when doing forward or reverse Domain Name System (DNS) searches. These warnings usually relate to the possibility of mismatched DNS records.

To do a hex dump of Netcat traffic to a file, use the *–o filename* option.

To specify on which port on the local (server) machine Netcat should listen, use the *-p port* switch:

```
nc -l -p 12345
```

In this example, Netcat is run in server mode and listening for inbound connections on port 12345.

Netcat can also scan ports in client mode. You can specify more than one port (separated by commas), ranges (all-inclusive), or even common port names. When specifying the port number of a host in client mode, the *–p* option is not necessary. Simply list the hostname followed by the port number(s) or range. If you specify a range of ports, Netcat starts at the top and works toward the bottom. Therefore, if you ask Netcat to scan ports 20–30, it will start at 30 and work backwards to 20.

To randomize ports, use the *–r* option. If you're using Netcat to scan ports, *–r* will allow Netcat to scan in a random manner as opposed to the standard top to bottom approach. Furthermore, *–r* will also randomize your local source ports in server mode.

We can use the *–s* option to change the source address of a packet, which is useful for spoofing the location of origin. This is another command whose usefulness has degraded over time due to smarter routers that drop such packets. The other obvious limitation is that replies are sent to the spoofed address instead of the true location.

To configure Netcat to answer Telnet negotiations, use the server-specific *–t* command. In other words, Netcat can be setup as a simple Telnet server. Consider the following command:

```
nc -l -p 12345 -e cmd.exe -t
```

Note that the previous command is specific to a Netcat server running on Windows. If your server instance of Netcat is running in Linux, you'd want to execute */bin/bash* instead of *cmd.exe*.

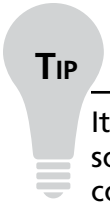
Use Netcat, Telnet, or any client such as PuTTY to connect to this server, and you'll have shell access via Telnet.

WARNING

Recall that Netcat is not encrypted. Furthermore, Telnet is a clear-text protocol. Likewise, any communications over such a link are subject to sniffing.

The UDP rather than the default TCP is configured with the *–u* switch. Since UDP is a connectionless protocol, it is recommended that you use timeouts with this option.

The *–v* option, common to many command-line programs, controls verbosity, or the amount of information that is displayed to the user. While you can run Netcat perfectly without this option, Netcat will run silently and only provide you information if an error occurs. Again, as with many other programs, you can increase the verbosity level with more than one *v* (both *–v –v* or *–vv* will work).



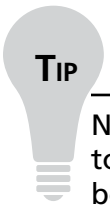
TIP

It is highly recommended to use the *–v* switch every time you use Netcat, so you can see information about what it's trying to do. Many users also combine *–v* with *–w* (see below).

Take note that in the GNU Linux version, *-V* displays the version information and then exits.

Use *–w secs* to set the network inactivity timeout. This option is useful for closing connections when servers don't do it automatically, and for speeding up your requests. A common time is 3 seconds.

Zero input/output mode is designated by the *–z* switch. This option is primarily used for port scanning. When *–z* is selected, Netcat will not send any data to a TCP connection, and will send only limited data to a UDP connection.



TIP

Netcat switches can be used individually, or together. For example, you want to start Netcat in server mode to listen on port 12345, and include the verbose option. Your command line would be *nc –v –l –p 12345*. However, you can also use multiple letter switches, which would result in a command *nc –vlp 12345*.

Redirector Tools

Finally, there are some standard UNIX redirectors that can be used with Netcat. The most useful are >, >>, <, and the pipe (|).

The single "greater than" redirector will redirect output:

```
nc -l -p 12345 > dumpfile
```

This command will redirect all received information into *dumpfile*. This could simply be any text input from the other end of the connection, or even a file being transmitted. In other words, whatever is being pushed into the listener will be redirected to *dumpfile*.

The double "greater than" redirector will redirect output, but append rather than replace:

```
nc -l -p 12345 >> dumpfile
```

WARNING

The single "greater than" redirector is designed to redirect output into a specified location or file. It is important to keep in mind that if you use the same filename, the single redirector will overwrite your original file. If you want to keep your original file, your safer option is to use the double "greater than" redirector to append the file instead of replacing it. The double redirector will also create a new file if one doesn't already exist to append.

The "less than" redirector will redirect input:

```
nc -l -p 12345 < dumpfile
```

When a client connects to this server, Netcat will send the *dumpfile* to the client. In other words, the connecting Netcat client is pulling the file from the server.

Another useful redirector tool is the pipe (|), which allows output from one command to serve as input to a second command (and so on). These processes together constitute a "pipeline." Some common commands that are often used in concert with Netcat are cat (sending a file), echo, and tar (compressing and sending a directory). You could even run Netcat twice to set up a relay. There are really no limits to the possibilities.

Basic Operations

In the remainder of this chapter, we'll explore some of the basic operations of Netcat.

Simple Chat Interface

We stated at the outset that Netcat is a networking program designed to read and write data across connections. Perhaps the easiest way to understand how this works is to simply set up a server and client. You can set up both of these on the same computer, or use two different computers. For the sake of this demonstration, we'll start both server and client on the same interface. In one terminal window, start the server:

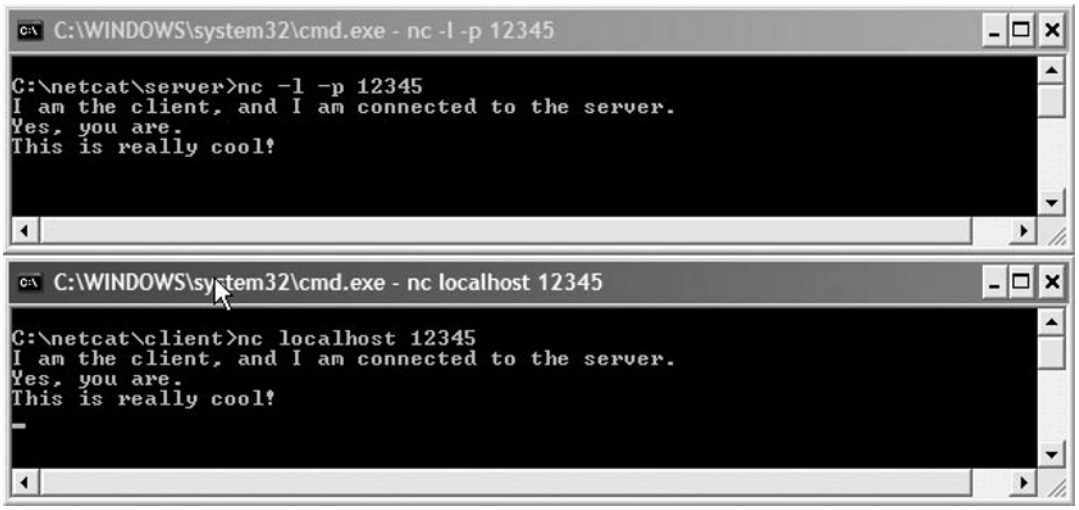
```
nc -l -p 12345
```

In a second window, connect to the server with the client:

```
nc localhost 12345
```

The result is a very elementary chat interface (see Figure 1.12). Text entered on one side of the connection is simply sent to the other side of the connection when you hit **enter**. Notice there is nothing to indicate the source of the text, only the output is printed.

Figure 1.12 Sending Data Across a Connection



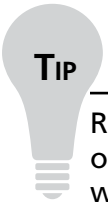
Port Scanning

Although it is not necessarily the best option for port scanning (Nmap is widely considered to be the cream of the crop), Netcat does have some rudimentary port scanning capabilities. As BackTrack developer Mati Aharoni has said, "It's not always the best tool for the job, but if I was stranded on an island, I'd take Netcat with me." I would guess that many people, given the choice of only one tool, would also choose Netcat.

Port scanning with Netcat occurs in the client mode. The syntax is as follows:

```
nc -[options] hostname [ports]
```

The most common options associated with port scanning are *–w* (network inactivity timeout) and *–z*, both of which may help to speed up your scan. Other possibilities are *–i* (sets a delay interval between ports scanned), *–n* (prevents DNS lookup), and *–r* (scans ports randomly). See Figure 1.13 for an example.



TIP

Remember to use the *–v* (verbose) option while port scanning (another option would be to redirect the output to a file). If you don't do this, Netcat will still scan the ports, but won't send you any output. In general, *–v* is almost always a good option to use.

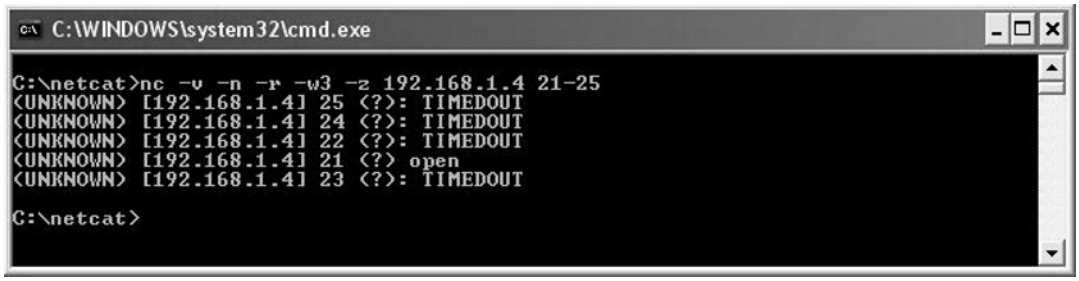
When listing ports, you have a number of options. You can list an individual port number, a series of ports separated by commas, or a range of ports (inclusive). You can even list a port by its service name. The following are all valid examples:

```
nc –v 192.168.1.4 21, 80, 443
nc –v 192.168.1.4 1-200
nc –v 192.168.1.4 http
```

Among common ports, Netcat will tell you the service associated with a specific port. Within Windows, the recognized services are located in */WINDOWS/system32/drivers/etc/services*. In Linux, the */etc/services* file serves the same purpose. These files are also the reference for using service names instead of port numbers.

In Figure 1.13, Netcat is run in client mode with the following options: verbose, no DNS lookup, randomize the order of scanned ports, network inactivity timeout of 3 seconds, and zero input/output mode. The host is 192.168.1.4, and the ports to scan are 21–25. Netcat returned port 21 open, which is most likely used for FTP. For more information on port scanning with Netcat, see Chapter 10, *Auditing with Netcat*.

Figure 1.13 Port Scanning with Netcat



NOTE

You can also scan UDP ports by using the *–u* option, but be aware that "no reply" is recognized as an open port. This, of course, is probably not the case under most circumstances.

Transferring Files

One common use for Netcat is for transferring files. Netcat has the ability to both pull and push files. Consider the following example:

```
nc -l -p 12345 < textfile
```

In this case, Netcat is started in server mode on local port 12345, and is offering *textfile*. A client who connects to this server is pulling the file from the server, and will receive *textfile*:

```
nc 192.168.1.4 12345 > textfile
```

Notes from the Underground…

Pulling Files with Netcat

You might wonder, with good reason, why you would use Netcat to transfer files instead of using the much more common File Transfer Protocol (FTP). In truth, FTP might be the better option in many cases. However, consider the potentially nefarious situation in which you have shell access on a target computer inside a firewall. You need to transfer some files to the destination, but the firewall is blocking inbound traffic.

In this case, you can run Netcat locally in server mode, offering the file(s) you want to send. Next, run Netcat in client mode from the target. In most cases, firewalls allow common outbound traffic, so you can probably hide your file transfers on a common port such as 80 (HTTP). See Chapter 5, *The Dark Side of Netcat*, and Chapter 6, *File Transfers with Netcat*, for more information.

Netcat can also be used to push files. If you're running Netcat from the destination (the place you want the file to end up), start Netcat in server mode:

```
nc -l -p 12345 > textfile
```

On the source machine, push the file by starting Netcat in client mode:

```
nc 192.168.1.4 12345 < textfile
```

As with all connections using Netcat, file transfers are unencrypted. If you are concerned about the privacy of the data you are transferring over Netcat, consider using Cryptcat, a version of Netcat that incorporates encrypted tunnels. Cryptcat uses the same command-line syntax as Netcat, but uses twofish encryption. Also consider using Netcat inside an Secure Shell (SSH) tunnel as a means of encrypting Netcat's traffic. This section was meant to be a very basic introduction to transferring files with Netcat. For more detailed information, especially in reference to encrypting and decrypting file transfers, see Chapter 6, *File Transfers with Netcat*.

Banner Grabbing

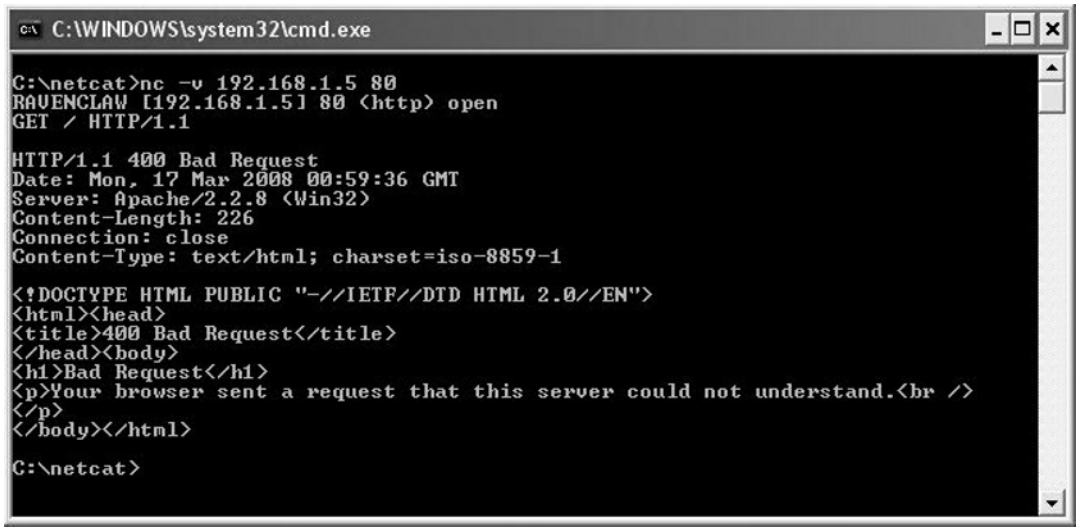
Banner grabbing is an enumeration technique, which is designed to determine the brand, version, operating system, or other relevant information about a particular service or application. This is especially important if you are looking for a vulnerability associated with a particular version of some service.

The syntax of a banner grab is not unlike the standard Netcat command line. Run Netcat in client mode, list the appropriate hostname, and finally list the port number of the appropriate service. In some cases, you may not have to enter any information (see Figure 1.14). In other cases, you will have to enter a valid command based on the particular protocol (see Figure 1.15).

Figure 1.14 SSH Banner Grabbing with Netcat

In Figure 1.14, opening Netcat to our target gave us two pieces of information: the hostname associated with the IP, and the version information for the SSH service running on that computer.

Figure 1.15 HTTP Banner Grabbing With Netcat



In Figure 1.15, we started Netcat in client mode. Our target is a Web server running on the target IP. By issuing the GET command (regardless of the fact that it is a bad request), the returned information gives us the Web server software and version number. It also tells us that this particular version of Apache is running on a Windows box.

For more detailed information, see Chapter 4, *Banner Grabbing with Netcat*.

Redirecting Ports and Traffic

Moving to a slightly darker shade of operation, Netcat can be used to redirect both ports and traffic. This is particularly useful if you want to obscure the source of an attack. The idea is to run Netcat through a middle man so that the attack appears to be coming from the middle man and not the original source. The following example is very simple, but multiple redirections could be used. This example also requires that you "own" the middle man and have already transferred Netcat to that box. This redirection of traffic is called a *relay*. From the source computer:

```
nc <hostname of relay> 12345
```

On the relay computer:

```
nc -l -p 12345 | nc <hostname of target> 54321
```

In this basic scenario, input from the source computer (in client mode) is sent to the relay computer (in server mode). The output is piped into a second instance of Netcat (in client mode), which ultimately connects to the target computer. Second, Netcat originates on port 12345, yet the attacker would see the attack coming from port 54321. This is a simple case of *port redirection*. This technique can also be used to hide Netcat traffic on more common ports, or change ports of applications whose normal ports might be blocked by a firewall.

There is an obvious limitation to this relay. The piped data is a one-way connection. Therefore, the source computer has no way of receiving any response from the target computer. The solution here would be to establish a second relay from the target computer back to the source computer (preferably through another middle man!).

For more detailed information on traffic redirection, see Chapter 5, *The Dark Side of Netcat*, and Chapter 7, *Controlling Traffic with Netcat*.

Other Uses

This section covered basic operations of Netcat, but the only limit to Netcat's operations is your imagination. Other potential, more advanced operations for Netcat include:

- Vulnerability scanning (see Chapter 2, *Netcat and Network Penetration Testing*, and Chapter 3, *Netcat and Application Penetration Testing*)
- General network troubleshooting (see Chapter 8, *Troubleshooting with Netcat*)
- Network and device auditing (see Chapter 9, *Auditing with Netcat*)
- Backing up files, directories, and even drives

The remainder of this book is dedicated to these and many other uses of Netcat.

Summary

Netcat is a networking program designed to read and write data across both TCP and UDP connections using the IP protocol suite. More simply, Netcat is the network version of the UNIX program *cat*. In the same way that *cat* reads and writes information to files, *Netcat* reads and writes information across network connections. Despite the introduction of more advanced tools over the last decade, Netcat remains popular among users for its simple, yet powerful capabilities.

Simple yet powerful is a theme that ties this chapter together. As we have seen, installation of Netcat, whether by Windows or by Linux (via package or source), is straightforward. There are only a handful of commonly used switches, which makes learning the command line practically effortless. Yet the trouble-free installation and the easy command line belie the fact that Netcat is indeed a potent and powerful program.

Netcat's simplicity may cause some people to overlook it. People have said they "underestimated" Netcat's usefulness. Others talk of "rediscovering" Netcat after several years. Regardless of the source, the answer always seems to be … go with Netcat! Many users even recommend replacing Telnet with Netcat.

Netcat is useful enough to have a place in most users' toolkit. Whether you are a network administrator troubleshooting your network, a penetration tester assessing a client's security, or just a user trying to learn something new, Netcat has something for you.

A few years back, Mati Aharoni, one of the core developers of the BackTrack penetration testing CD and founder of www.offensive-security.com, wrote a short security paper that demonstrated an entire hack from start to finish. It began with a port scan, and then continued with a banner grab, application vulnerability scan, setting up a back door, and finally transferring a file to the owned system. The file was a short text message that simply said, "You have been hacked!" If you've come this far, you know that this hack was completed from start to finish with only one tool, Netcat.

Solutions Fast Track

Introduction

- ☑ Netcat is a simple program that reads and writes data across networks, much the same way that cat reads and writes data to files.
- ☑ Netcat is available on most systems: UNIX/Linux, Windows, BSD, Mac, and others. Linux and Windows are the most common implementations.
- ☑ Despite newer and more powerful tools, Netcat remains a popular choice among users.

Installation

- ☑ Windows installation is a cinch. Simply download and unzip!
- ☑ Linux installation is not too difficult. Install a pre-compiled package or download the source and compile it yourself.
- ☑ The Netcat help screen is useful not only to display the various options, but also to confirm an installation, determine the version of a previously installed package, or confirm it was compiled with the GAPING_SECURITY_HOLE option.

Options

- ☑ Netcat has two modes of operation: client and server (or listening mode).
- ☑ The *–e* option, which allows Netcat to execute programs, is what makes Netcat so powerful.
- ☑ Standard UNIX redirector tools allow Netcat to push and pull data from various sources and destinations, and pipe data to and from other processes.

Basic Operations

- ☑ Netcat's basic operations include a rudimentary chat interface and transferring files.
- ☑ For penetration testers, Netcat allows enumeration through port scanning and banner grabbing.
- ☑ Netcat can be used for port and traffic redirection, which can obscure the source of an attack.

Frequently Asked Questions

Q: I haven't even downloaded Netcat yet, but my anti-virus found Netcat as a trojan! What should I do?

A: If you have never downloaded or installed Netcat, you may well have an issue. In addition to the vanilla version of Netcat, there are many other versions already compiled that auto-configure themselves to specific ports (*ncx.exe* ran on port 80, while *ncx99.exe* was configured for port 99).

Q: My anti-virus program won't let me download */install/* using Netcat. Why not?

A: At least two major anti-virus vendors (and probably more) flag Netcat as a problem. In a few test cases, one of them actually prevented a download from completing, because Netcat was inside the larger installable package. The second quarantined it as part of a live "auto-protect" feature. There are a few ways around this, and they typically involve modifying "default" parameters. First, you can disable live protection, at least for the short period that you download Netcat. Second, you can create a special directory for Netcat (and other such tools that might be setting off your anti-virus) and configure your live or auto-protect feature to ignore this directory. Finally, you can exclude this directory from your normal, scheduled anti-virus scans.

Q: Netcat is already installed on my system. Why would I want to install it again?

A: Many packages of Netcat that come pre-installed with Linux distributions are "safe" compiled without the GAPING_SECURITY_HOLE option. Without this capability, Netcat cannot execute programs. Since most of Netcat's power comes from this option, you should recompile or reinstall Netcat if you want this capability.

Q: How do I know if Netcat was compiled with the *–e* option?

A: If you're running Netcat on Windows, this version has already been compiled with this option and no further action is necessary. If you're running Netcat on Linux, simply bring up the help screen by typing **nc –h**. GNU Netcat (version 0.7.1) is already compiled with this option, so again, no further action is necessary. The original UNIX version of Netcat (typically version 1.10) is compiled with this option if the help screen displays this option. On Macs, Netcat is compiled without this option by default.

Q: How do I know if Netcat is running in client or server mode?

A: The *–l* switch denotes listening, or server mode. The absence of it indicates client mode.

Q: Netcat shuts down server mode when I disconnect, but I want the connection to be persistent. Is this possible?

A: Yes. In Windows, use the *–L* option, which reopens Netcat with the same options every time it is closed. This particular option is not available in Linux, but you can write a simple work-around script, which will accomplish the same thing.

Q: Netcat would be even cooler if it could just do [insert über-leet feature here]! How can I do it?

A: Netcat is open source. That means you can download the source code, modify it to your delight, and then recompile it with your über-leet options.

Q: Where can I find more information about Netcat?

A: First, refer to the remaining chapters in this book. The contributing authors are extremely knowledgeable, and experts in their fields. Second, Google it. There is a wide range of Netcat documents and tutorials on the Internet. Third, find a forum somewhere and post a question. There are a lot of people out there willing to help, if you know how to ask!

This page intentionally left blank