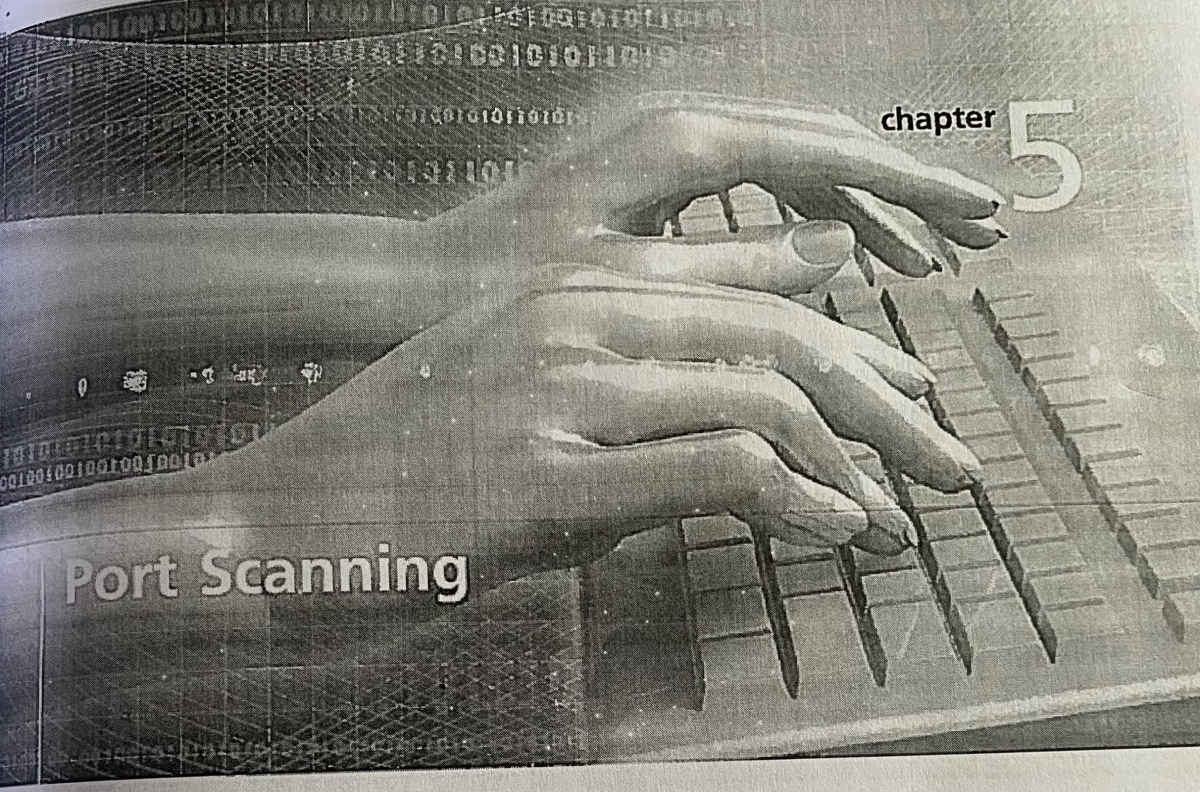


chapter 5

Port Scanning

After reading this chapter and completing the exercises, you will be able to:

- Describe port scanning and types of port scans
- Describe port-scanning tools
- Explain what ping sweeps are used for
- Explain how shell scripting is used to automate security tasks

Unit-II (2).

Port scanning, also referred to as service scanning, is the process of examining a range of IP addresses to determine what services are running on a network. As you learned in Chapter 2, open ports on a computer identify the services running on it. For example, HTTP uses port 80 to connect to a Web service. Instead of pinging each IP address in a range of addresses and waiting for an ICMP Echo Reply (type 0) to see whether a computer can be reached, you can use scanning tools to simplify this procedure. After all, pinging several thousand IP addresses manually is time consuming.

Port-scanning tools can be complex, so you need to devote time to learning their strengths and weaknesses and understanding how and when you should use these tools. In this chapter, you look at port-scanning tools that enable you to identify services running on a network and use this knowledge to conduct a security test. In addition, you see how to use shell scripting to automate ping sweeps and other security-testing tasks.

Introduction to Port Scanning

In Chapter 4, you performed a zone transfer with the Dig command to determine a network's IP addresses. Suppose the zone transfer indicates that a company is using a subnetted Class C address with 126 available host IP addresses. How do you verify whether all these addresses are being used by computers that are up and running? You use a port scanner to ping the range of IP addresses you discovered.

A more important question a security tester should ask is "What services are running on the computers that were identified?" Port scanning is a method of finding out which services a host computer offers. For example, if a server is hosting a Web site, is it likely that the server has port 80 open? Are any of the services vulnerable to attacks or exploits? Are any services not being filtered by a firewall, thus making it possible to load a Trojan program that can send information from the attacked computer? Which computer is most vulnerable to an attack? You already know how to search for known vulnerabilities by using the Common Vulnerabilities and Exposures (*www.cve.mitre.org*) and US-CERT (*www.us-cert.gov*) Web sites. There are also port-scanning tools that identify vulnerabilities. For example, AW Security Port Scanner (*www.atelierweb.com*), a reasonably priced commercial scanner with a GUI interface (see Figure 5-1), shows the type of Trojan program known to operate on a particular port. Using this tool, an attacker can quickly identify a vulnerable port and then launch an exploit to attack the system.

As a security tester, you need to know which ports attackers are going after so those ports can be closed or protected. Security professionals must scan all ports when doing a test, not just the well-known ports. (Ports 1 to 1023, the most common, are covered in Chapter 2.) Many programs use port numbers outside the range of well-known ports. For example, pcAnywhere operates on ports 65301, 22, 5631, and 5632. A hacker who discovers that port 65301 is open can check the information at the CVE Web site for a possible vulnerability in pcAnywhere. After a hacker discovers an open service, finding a vulnerability or exploit isn't difficult.

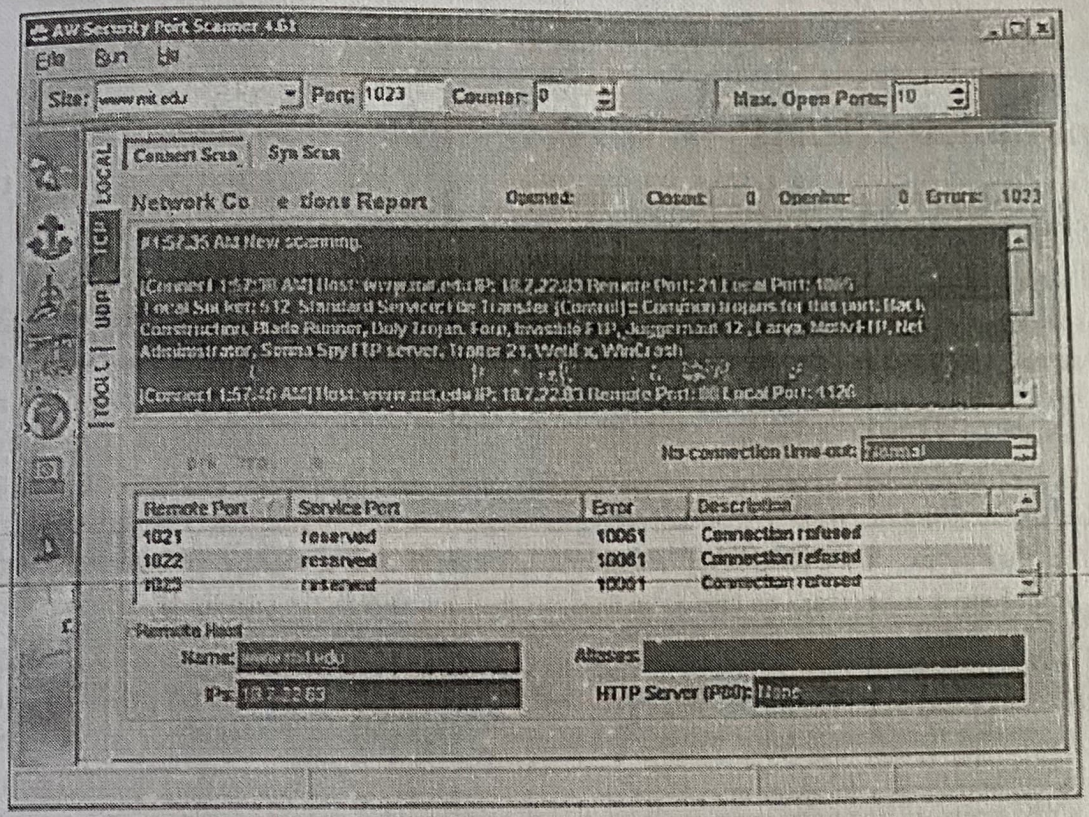


Figure 5-1 The AW Security Port Scanner interface

Courtesy Course Technology/Cengage Learning

Security Bytes

Most security testers and hackers argue that port scanning is legal simply because it doesn't invade others' privacy; it merely discovers whether the party being scanned is available. The typical analogy is a person walking down the street and turning the doorknob of every house along the way. If the door opens, the person notes that the door is open and proceeds to the next house. Of course, entering the house is a crime in most parts of the world, just as entering a computer or network system without the owner's permission is a crime. To date, no one has been convicted just for port scanning, although laws exist for prosecuting scanning if it causes damage or loss of more than $5000 (U.S. Code 18 1030).

Port scanning helps you answer questions about open ports and services by enabling you to scan thousands or even tens of thousands of IP addresses quickly. Many port-scanning tools produce reports of their findings, and some give you best-guess assessments of which OS is running on a system. Most, if not all, scanning programs report open ports, closed ports, and filtered ports in a matter of seconds. An open port allows access to applications and can be vulnerable to an attack. When a Web server needs to communicate with applications or other computers, for example, port 80 is opened. A closed port doesn't allow entry or access to a service. For instance, if port 80 is closed on a Web server, users can't access Web sites. A port reported as filtered might indicate that a firewall is being used to allow specified traffic into or out of the network.

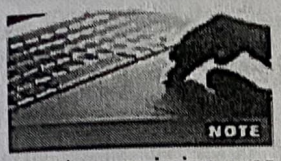
Types of Port Scans

Before delving into using port-scanning tools, take a look at the types of scans that can be used for port scanning:

- *SYN scan*—In a normal TCP session, a packet is sent to another computer with the SYN flag set. The receiving computer sends back a packet with the SYN/ACK flag set, indicating an acknowledgment. The sending computer then sends a packet with the ACK flag set. If the port the SYN packet is sent to is closed, the computer responds with an RST/ACK (reset/acknowledgment) packet. If an attacker's computer receives a SYN/ACK packet, it responds quickly with an RST/ACK packet, closing the session. This is done so that a full TCP connection is never made and logged as a transaction. In this sense, it's "stealthy." After all, attackers don't want a transaction logged showing their connection to the attacked computer and listing their IP addresses.
- *Connect scan*—This type of scan relies on the attacked computer's OS, so it's a little more risky to use. A connect scan is similar to a SYN scan, except that it does complete the three-way handshake. This means the attacked computer most likely logs the transaction or connection, indicating that a session took place. Therefore, unlike a SYN scan, a connect scan isn't stealthy and can be detected easily.
- *NULL scan*—In a NULL scan, all packet flags are turned off. A closed port responds to a NULL scan with an RST packet, so if no packet is received, the best guess is that the port is open.
- *XMAS scan*—In this type of scan, the FIN, PSH, and URG flags are set. (Refer to Chapter 2 for a review of the different flags.) Closed ports respond to this type of packet with an RST packet. This scan can be used to determine which ports are open. For example, an attacker could send this packet to port 53 on a system and see whether an RST packet is returned. If not, the DNS port might be open.
- *ACK scan*—Attackers typically use ACK scans to get past a firewall or other filtering device. A filtering device looks for the SYN packet, the first packet in the three-way handshake, that the ACK packet was part of. Remember this packet order: SYN, SYN/ACK, and ACK. If the attacked port returns an RST packet, the packet filter was fooled, or there's no packet-filtering device. In either case, the attacked port is considered to be "unfiltered."
- *FIN scan*—In this type of scan, a FIN packet is sent to the target computer. If the port is closed, it sends back an RST packet. When a three-way handshake ends, both parties send a FIN packet to end the connection.
- *UDP scan*—In this type of scan, a UDP packet is sent to the target computer. If the port sends back an ICMP "Port Unreachable" message, the port is closed. Again, not getting that message might imply the port is open, but this isn't always true. A firewall or packet-filtering device could undermine your assumptions.

As you learned in Chapter 2, a computer that receives a SYN packet from a remote computer responds with a SYN/ACK packet if its port is open. In a three-way handshake, a SYN packet is sent from one computer, a SYN/ACK is sent from the receiving computer to the sender, and finally, the sender sends an ACK packet to the receiving computer. If a port is closed and receives a SYN packet, it sends back an RST/ACK packet. Determining whether a port is filtered is more complex. Many scanning tools, such as Nmap, use a best-guess

approach. That is, if a UDP packet doesn't receive a response from the receiving port, many scanning tools report that the port is open.



Security Bytes

In Canada, a man was found guilty of scanning a company's computers. The company actually prosecuted him for using microwatts of its electrical power to perform the scan. Doing so without the company's permission was considered a crime—petty, yes, but effective. To play it safe, always get permission from a company if you're going to perform an intensive scan on its network infrastructure. If your scan slows down a network's traffic, the company might argue that a low-level DoS attack, which is illegal, was performed.

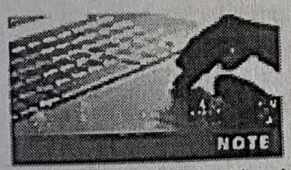
Using Port-Scanning Tools

Hundreds of port-scanning tools are available for both hackers and security testers. Some are commercial, and some are freeware or open source. How do you decide which tool to use? Not all are accurate, so using more than one port-scanning tool is recommended. In addition, becoming familiar with a variety of tools is wise. Although you should practice often with a tool to gain proficiency in using it, don't fall into the trap of using one tool exclusively.

Nmap

Originally written for *Phrack* magazine in 1997 by Fyodor, Nmap has become one of the most popular port scanners and adds new features constantly, such as OS detection and fast multiple-probe ping scanning. Nmap also has a GUI front end called Zenmap that makes working with complex options easier. Nmap has been enhanced over the years because, like many other security tools, it's open source; if bugs are found, users can offer suggestions for correcting them.

Nmap is referred to often in this book because it's currently the standard port-scanning tool for security professionals. Regardless of the other port-scanning tools available, any security tester with a modicum of experience has worked with Nmap. As a beginning student, you can use it for every part of a security or penetration test, but remember to build proficiency in all the tools discussed in this book.

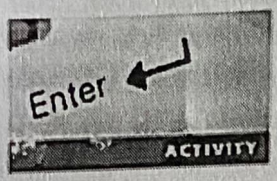


Security Bytes

As most security professionals will tell you, Hollywood seldom depicts attackers actually hacking into a system. Typically, they're using a GUI program, frantically clicking or typing a decryption algorithm. One exception is *The Matrix Reloaded*. The female protagonist, Trinity, sits in front of a computer terminal and runs Nmap. She discovers that port 22 (SSH) is open, runs an SSHv1 CRC32 exploit (an actual bug in SSH) that allows her to change the root password to Z1ON0101, and then proceeds to shut down the grid. Moral of the story? Know your tools and exploits, and you might save the world.

You don't have to memorize how each flag is set when running a port scan with Nmap. In fact, just typing the command `nmap 193.145.85.201` scans every port on the computer with this IP address. However, port scanning can be an involved process. Some attackers

want to be hidden from network devices or IDSs that recognize an inordinate amount of pings or packets being sent to their networks, so they use stealth attacks that are more difficult to detect. In the following activities, you become familiar with the basic Nmap commands and then learn some of the more complex options.



Activity 5-1: Getting to Know Nmap

Time Required: 30 minutes

Objective: Learn the basic commands and syntax of Nmap.

Description: In this activity, you're introduced to using Nmap for quick scans of a network. You send a SYN packet to a host on the attack network your instructor has supplied. In this example, the attack network IP addresses are 193.145.85.201 to 193.145.85.211, but your attack range might be different. Make sure to follow the rules of engagement, and don't perform port scanning on any systems not included in the IP range your instructor gives you.

1. Boot your computer into Linux with the BackTrack DVD. By default, the system starts with the BackTrack command shell (no graphical environment). At the shell prompt, type startx and press Enter to open the KDE desktop manager for BackTrack. Then open a command shell by clicking the Konsole terminal icon on the panel taskbar. Type nmap -h |less and press Enter to see all available Nmap commands. Your screen should look like Figure 5-2. You can scroll to review the command parameters.

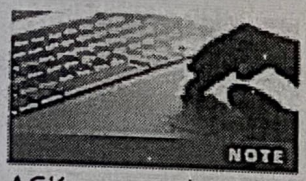
```
Nmap 4.85BETA10 ( http://nmap.org )
Usage: nmap [Scan Type(s)] [Options] {target specification}
TARGET SPECIFICATION:
  Can pass hostnames, IP addresses, networks, etc.
  Ex: scanme.nmap.org, microsoft.com/24, 192.168.0.1; 10.0.0-255.1-254
  -iL <inputfilename>: Input from list of hosts/networks
  -iR <num hosts>: Choose random targets
  --exclude <host1[,host2][,host3],...>: Exclude hosts/networks
  --excludefile <exclude file>: Exclude list from file
HOST DISCOVERY:
  -sL: List Scan - simply list targets to scan
  -sP: Ping Scan - go no further than determining if host is online
  -PN: Treat all hosts as online -- skip host discovery
  -PS/PA/PU/PY[portlist]: TCP SYN/ACK, UDP or SCTP discovery to given ports
  -PE/PP/PM: ICMP echo, timestamp, and netmask request discovery probes
  -PO[protocol list]: IP Protocol Ping
  -n/-R: Never do DNS resolution/Always resolve [default: sometimes]
  --dns-servers <serv1[,serv2],...>: Specify custom DNS servers
  --system-dns: Use OS's DNS resolver
  --traceroute: Trace hop path to each host
SCAN TECHNIQUES:
  -sS/sT/sA/sW/sM: TCP SYN/Connect()/ACK/Window/Maimon scans
  -sU: UDP Scan
:
```

Figure 5-2 The Nmap help screen

Courtesy Course Technology/Cengage Learning

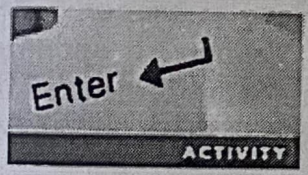
2. After reviewing the parameters, write down three options that can be used with the Nmap command, and then press q to exit the help screen.
3. To send a SYN packet to an IP address in your attack range, type nmap -sS -v 193.145.85.201 and press Enter. What are the results of your SYN scan?

4. Next, try sending a new SYN packet to a different IP address in your attack range. What are the results of this new scan? Do you see any differences? If so, list them.
5. Nmap can scan through a range of IP addresses, so entering one IP address at a time isn't necessary. To send a SYN packet to every IP address in your attack range, type nmap -sS -v 193.145.85.201-211 and press Enter. To see the output in a format you can scroll, press the up arrow key, add the |less option to the end of the Nmap command, and press Enter. The command should look like this: nmap -sS -v 193.145.85.201-211 |less.
6. Next, add one more parameter to the Nmap command to determine which computers in your attack range have the SMTP service or HTTP service running. Using what you've learned so far in this activity, enter the command and note the output. (*Hint*: What ports do SMTP and HTTP use?) The command's output might vary, but what's important is learning how to build on the Nmap command. You can select specific ports in the Nmap command, so not all 65,000 ports have to be scanned.
7. Leave the Konsole shell open for the next activity.



Security Bytes

A security professional came to work one evening and noticed that the company's firewall had crashed because someone ran a port-scanning program on the network by using ACK packets. Many attackers use ACK scans to bypass packet-filtering devices (such as firewalls, discussed in Chapter 13). In this case, the company's firewall was disabled because it was flooded with tens of thousands of ACK packets bombarding its routing tables. This ACK scan constituted a DoS attack on the network, so don't get complacent when running port scans on networks. Always get the network owner's written permission before doing a port scan.



Activity 5-2: Using Additional Nmap Commands

Time Required: 30 minutes

Objective: Perform more complex port-scanning attacks with Nmap.

Description: In this activity, you continue to use Nmap for port scanning on your attack network. You add to the parameters used in Activity 5-1 and send FIN, XMAS, and ACK packets to selected ports. You should practice these commands until they are second nature, but Fyodor developed a well-written help page (called a "man page" in UNIX/Linux circles) that you can use as a resource. You begin this activity by looking at this help page.

1. If a Konsole shell isn't open, boot your computer into Linux with the BackTrack DVD. Open a Konsole shell, and at the command prompt, type man nmap and press Enter. You can see that this command produces more information than the nmap -h command. Don't be concerned about memorizing the manual; just know it's there when you need it. When you're finished, press q to exit the help screen.
2. Open another Konsole shell so that you can run Tcpdump, which displays traffic generated from the packets you're creating. Like Wireshark, Tcpdump is a packet analyzer and is included on the book's DVD. You might want to get into the habit of having Tcpdump running in the background in a different shell so that you can view the packets generated when you scan a network. Type tcpdump -h in the new shell and press Enter to view the parameters you can use with this command.

3. You can type "man tcpdump" to examine this tool's help manual, but that isn't necessary now. Just type tcpdump and press Enter. Your network adapter card is now listening for traffic over the network.
4. Referring to the Nmap help pages for guidance, enter the command for sending a FIN packet to five computers in your IP attack range. Look at the traffic generated from your FIN scan. What responses did your computer receive, if any?
5. Next, enter the command to send an XMAS packet to the same five computers used in the FIN scan. What are the results?
6. Finally, enter the Nmap command for sending an ACK packet to the same five computers. What responses did your computer receive, if any? Leave one shell open for the next activity.

Unicornscan

Unicornscan was developed to assist security testers in conducting tests on large networks and to consolidate many of the tools needed for large-scale endeavors. The developers thought that many current products were too slow at scanning thousands of IP addresses. Also, maintaining several security tools can be daunting, so the Unicornscan developers created a product to meet all the needs of security testers.

Unicornscan running on a typical Pentium computer can scan one port on each IP address of a Class B network. This equates to scanning 65,535 computers in 3 to 7 seconds, which brings UDP scanning to a new level. Most scanners using UDP scans can just make best guesses when trying to determine whether a port is closed, open, or filtered. Many security testers consider UDP scanning an unreliable method of discovering live systems on a network. Although Unicornscan can handle TCP, ICMP, and IP port scanning, it optimizes UDP scanning beyond the capabilities of any other port scanner. Unicornscan is included on the BackTrack DVD along with a Web-based Unicornscan analysis tool. You can learn more about this tool at *www.unicornscan.org*.

Nessus and OpenVAS

Security testers should also investigate Nessus, a tool first released in 1998. Although Nessus is no longer under the GPL license, as most open-source software is, you can still download it free from Tenable Network Security Corporation (*www.nessus.org*) for noncommercial personal use. An open-source fork of Nessus called OpenVAS was developed in 2005, and it's one of the tools included on the BackTrack DVD. OpenVAS functions much like a database server, performing complex queries while the client interfaces with the server to simplify reporting and configuration.

What makes this tool unique is the capability to update security check plug-ins when they become available. An OpenVAS plug-in is a security test program (script) that can be selected from the client interface. The person who writes the plug-in decides whether to designate it as dangerous, and the author's judgment on what's considered dangerous might differ from yours. Therefore, leaving the Safe checks check box selected, as shown in Figure 5-3, is wise before you start a scan.

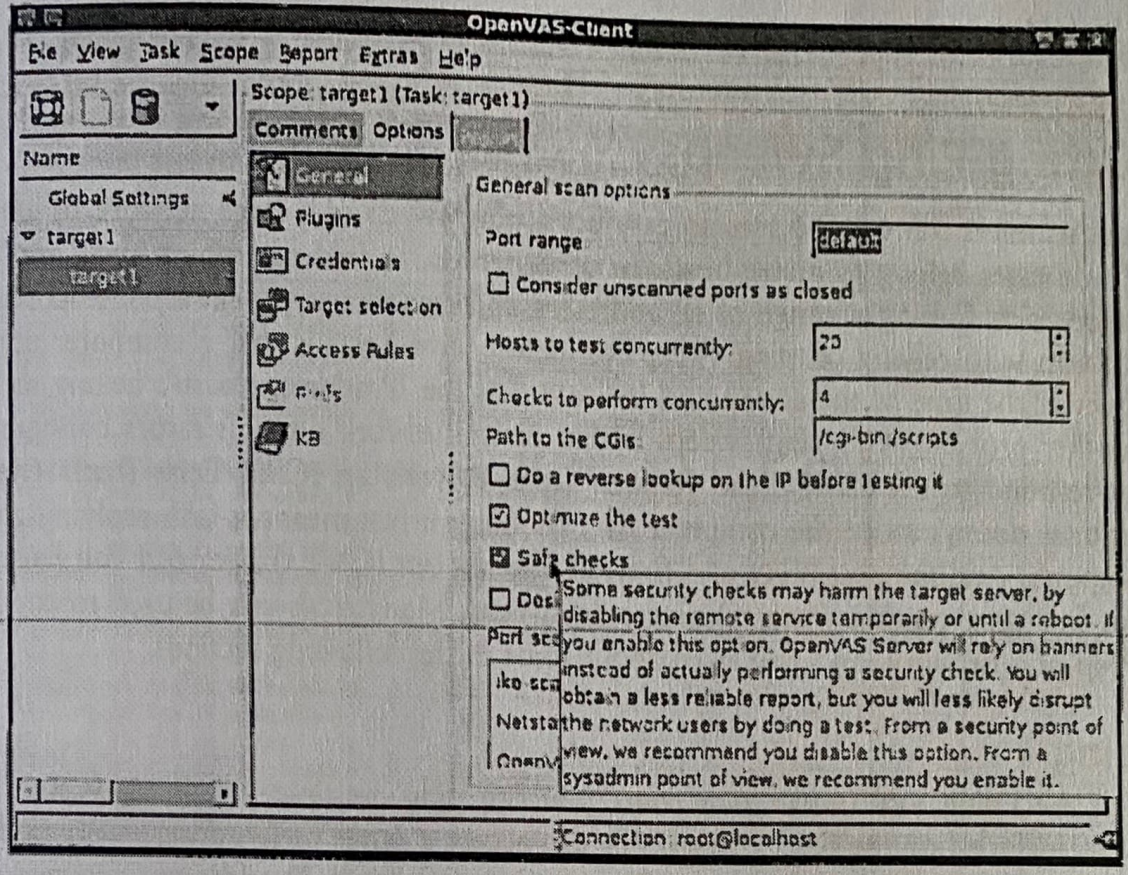


Figure 5-3 OpenVAS with a safe checks warning

Courtesy Course Technology/Cengage Learning

An OpenVAS scan isn't limited to determining which services are running on a port. OpenVAS plug-ins can also determine what vulnerabilities are associated with these services, as shown in Figure 5-4. (You use OpenVAS in later chapters.)

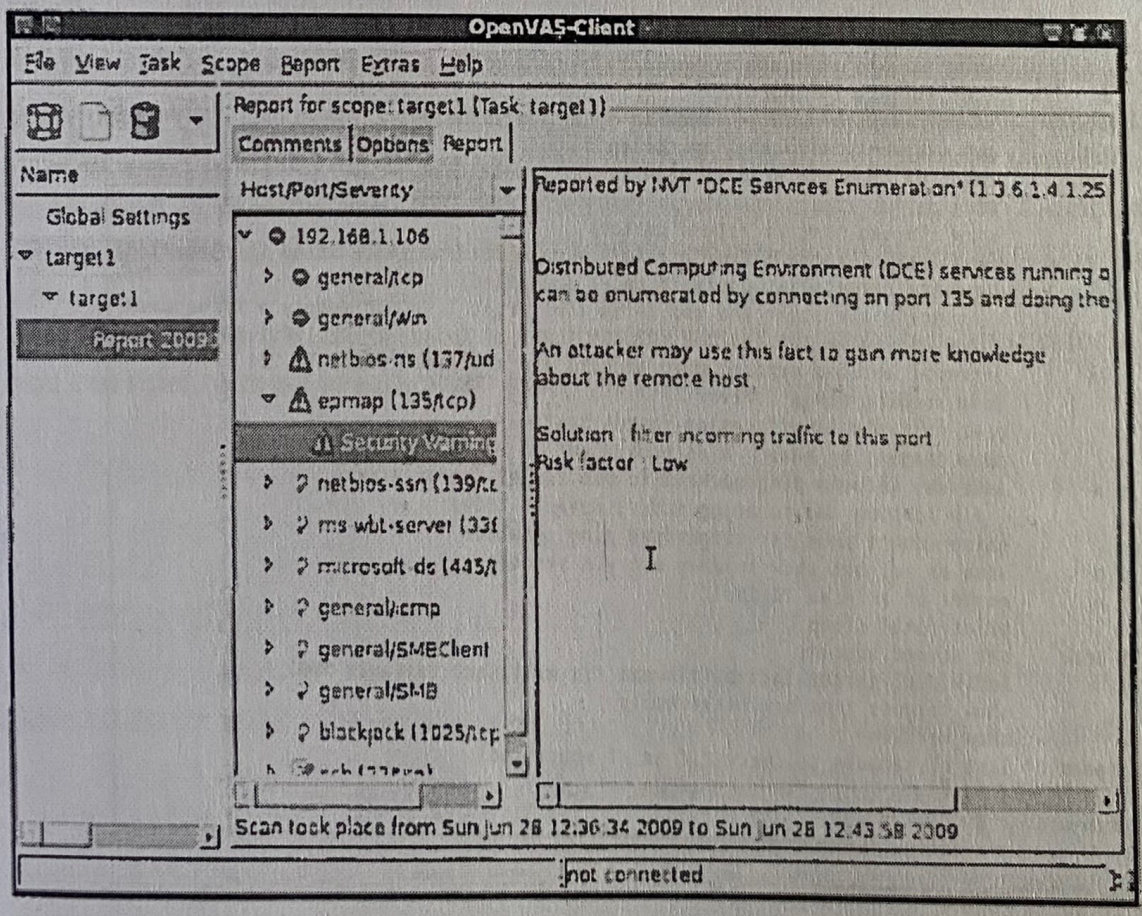


Figure 5-4 OpenVAS discovers a vulnerability

Courtesy Course Technology/Cengage Learning

Conducting Ping Sweeps

Port scanners can also be used to conduct a ping sweep of a large network to identify which IP addresses belong to active hosts. In other words, to find out which hosts are "live," ping sweeps simply ping a range of IP addresses and see what type of response is returned. The problem with relying on ping sweeps to identify live hosts is that a computer might be shut down at the time of the sweep and indicate that the IP address doesn't belong to a live host. Another problem with ping sweeps is that many network administrators configure nodes to not respond to an ICMP Echo Request (type 8) with an ICMP Echo Reply (type 0). This response doesn't mean the computer isn't running; it just means it isn't replying to the attack computer. Add the possibility of a firewall filtering out ICMP traffic, and you have many reasons for using caution when running ping sweeps. Many tools can be used to conduct a ping sweep of a network, and you learn about some in the following sections.

Fping

With the Fping tool (*www.fping.com*), you can ping multiple IP addresses simultaneously. Fping, included on the BackTrack DVD, can accept a range of IP addresses entered at a command prompt, or you can create a file containing multiple IP addresses and use it as input for the Fping command. For example, the `fping -f ip address.txt` command uses ip address.txt, which contains a list of IP addresses, as its input file. The input file is usually created with a shell-scripting language so that you don't need to type the thousands of IP addresses needed for a ping sweep on a Class B network, for example. Figure 5-5 shows some parameters you can use with the Fping command.

```
   -a          show targets that are alive
   -A          show targets by address
   -b n        amount of ping data to send, in bytes (default 56)
   -B f        set exponential backoff factor to f
   -c n        count of pings to send to each target (default 1)
   -C n        same as -c, report results in verbose format
   -e          show elapsed time on return packets
   -f file     read list of targets from a file ( - means stdin) (only if no -g specified)
   -g          generate target list (only if no -f specified)
                 (specify the start and end IP in the target list, or supply a IP netmask)
                 (ex. fping -g 192.168.1.0 192.168.1.255 or fping -g 192.168.1.0/24)
   -i n        interval between sending ping packets (in millisec) (default 25)
   -l          loop sending pings forever
   -m          ping multiple interfaces on target host
   -n          show targets by name (-d is equivalent)
   -p n        interval between ping packets to one target (in millisec)
                 (in looping and counting modes, default 1000)
   -q          quiet (don't show per-target/per-ping results)
   -Q n        same as -q, but show summary every n seconds
   -r n        number of retries (default 3)
   -s          print final stats
   -S addr     set source address
   -t n        individual target initial timeout (in millisec) (default 500)
   -u          show targets that are unreachable
   -v          show version
   targets     list of targets to check (if no -f specified)
root@hoh-bt4: #
```

Figure 5-5 Fping parameters

Courtesy Course Technology/Cengage Learning

To ping sweep a range of IP addresses without using an input file, you use the command `fping -g BeginningIPaddress EndingIPaddress`. The `-g` parameter is used when no input file is available. For example, the `fping -g 193.145.85.201 193.145.85.220` command returns the results shown in Figure 5-6.

```
root@hoh-bt4: # fping -g 193.145.85.201 193.145.85.220
193.145.85.201 is unreachable
193.145.85.202 is unreachable
193.145.85.203 is unreachable
193.145.85.204 is unreachable
193.145.85.205 is unreachable
193.145.85.206 is unreachable
193.145.85.207 is unreachable
193.145.85.208 is unreachable
193.145.85.209 is unreachable
193.145.85.210 is unreachable
193.145.85.211 is unreachable
193.145.85.212 is unreachable
193.145.85.213 is unreachable
193.145.85.214 is unreachable
193.145.85.215 is unreachable
193.145.85.216 is unreachable
193.145.85.217 is unreachable
193.145.85.218 is unreachable
193.145.85.219 is unreachable
193.145.85.220 is unreachable
root@hoh-bt4: #
```

Figure 5-6 Results of an Fping command

Courtesy Course Technology/Cengage Learning

Hping

You can also use the Hping tool (*www.hping.org/download*) to perform ping sweeps. However, many security testers use it to bypass filtering devices by injecting crafted or otherwise modified IP packets. This tool offers a wealth of features, and security testers should spend as much time as possible learning this advanced port-scanning tool. For a quick overview, use the `hping -help | less` command, and browse through the parameters you can use (see Figures 5-7, 5-8, and 5-9). As you can see, many parameters can be added to the Hping command, enabling you to craft an IP packet for your purposes. In Activity 5-3, you craft an IP packet, and you can refer to these figures when using the Hping tool.

```
File  Edit  View  Terminal  Go  Help
usage: hping host [options]
  -h  --help        show this help
  -v  --version     show version
  -c  --count       packet count
  -i  --interval    wait (uX for X microseconds, for example -i u1000)
      --fast        alias for -i u10000 (10 packets for second)
      --faster      alias for -i u1000 (100 packets for second)
      --flood       sent packets as fast as possible. Don't show replies.
  -n  --numeric     numeric output
  -q  --quiet       quiet
  -I  --interface   interface name (otherwise default routing interface)
  -V  --verbose     verbose mode
  -D  --debug       debugging info
  -z  --bind        bind ctrl+z to ttl           (default to dst port)
  -Z  --unbind      unbind ctrl+z

Mode
      default mode  TCP
  -0  --rawip       RAW IP mode
  -1  --icmp        ICMP mode
  -2  --udp         UDP mode
  -8  --scan        SCAN mode.
                    Example: hping --scan 1-30,70-90 -S www.target.host
  -9  --listen      listen mode

IP
  -a  --spoof       spoof source address
      --rand-dest   random destination address mode. see the man.
      --rand-source random source address mode. see the man.
  -t  --ttl         ttl (default 64)
  -N  --id          id (default random)
  -W  --winid       use win* id byte ordering
  -r  --rel         relativize id field          (to estimate host traffic)
  -f  --frag        split packets in more frag.  (may pass weak acl)
  -x  --morefrag    set more fragments flag
  -y  --dontfrag    set dont fragment flag
  -g  --fragoff     set the fragment offset
  -m  --mtu         set virtual mtu, implies --frag if packet size > mtu
  -o  --tos         type of service (default 0x00), try --tos help
  -G  --rroute      includes RECORD_ROUTE option and display the route buffer
      --lsrr        loose source routing and record route
      --ssrr        strict source routing and record route
  -H  --ipproto     set the IP protocol field, only in RAW IP mode
```

Figure 5-7 Hping help, page 1

Courtesy Course Technology/Cengage Learning

```
File  Edit  View  Terminal  Go  Help
ICMP
  -C  --icmptype    icmp type (default echo request)
  -K  --icmpcode    icmp code (default 0)
      --force-icmp  send all icmp types (default send only supported types)
      --icmp-gw     set gateway address for ICMP redirect (default 0.0.0.0)
      --icmp-ts     Alias for --icmp --icmptype 13 (ICMP timestamp)
      --icmp-addr   Alias for --icmp --icmptype 17 (ICMP address subnet mask)
      --icmp-help   display help for others icmp options
UDP/TCP
  -s  --baseport    base source port             (default random)
  -p  --destport    [+][+]<port> destination port(default 0) ctrl+z inc/dec
  -k  --keep        keep still source port
  -w  --win         winsize (default 64)
  -O  --tcpoff      set fake tcp data offset     (instead of tcphdrlen / 4)
  -Q  --seqnum      shows only tcp sequence number
  -b  --badcksum    (try to) send packets with a bad IP checksum
                    many systems will fix the IP checksum sending the packet
                    so you'll get bad UDP/TCP checksum instead.
  -M  --setseq      set TCP sequence number
  -L  --setack      set TCP ack
  -F  --fin         set FIN flag
  -S  --syn         set SYN flag
  -R  --rst         set RST flag
  -P  --push        set PUSH flag
  -A  --ack         set ACK flag
  -U  --urg         set URG flag
  -X  --xmas        set X unused flag (0x40)
  -Y  --ymas        set Y unused flag (0x80)
      --tcpexitcode use last tcp->th_flags as exit code
      --tcp-timestamp enable the TCP timestamp option to guess the HZ/uptime
Common
  -d  --data        data size                    (default is 0)
  -E  --file        data from file
  -e  --sign        add 'signature'
  -j  --dump        dump packets in hex
  -J  --print       dump printable characters
  -B  --safe        enable 'safe' protocol
  -u  --end         tell you when --file reached EOF and prevent rewind
  -T  --traceroute  traceroute mode              (implies --bind and --ttl 1)
```

```
File  Edit  View  Terminal  Go  Help
      --icmp-ts       Alias for --icmp --icmptype 13 (ICMP timestamp)
      --icmp-addr     Alias for --icmp --icmptype 17 (ICMP address subnet mask)
      --icmp-help     display help for others icmp options
TCP/TCP
  -s  --baseport      base source port                    (default random)
  -p  --destport      [+][+]<port> destination port(default 0) ctrl+z inc/dec
  -k  --keep          keep still source port
  -w  --win           winsize (default 64)
  -O  --tcpoff        set fake tcp data offset            (instead of tcphdrlen / 4)
  -Q  --seqnum        shows only tcp sequence number
  -b  --badsum        (try to) send packets with a bad IP checksum
                      many systems will fix the IP checksum sending the packet
                      so you'll get bad UDP/TCP checksum instead.
  -M  --setseq        set TCP sequence number
  -L  --setack        set TCP ack
  -F  --fin           set FIN flag
  -S  --syn           set SYN flag
  -R  --rst           set RST flag
  -P  --push          set PUSH flag
  -A  --ack           set ACK flag
  -U  --urg           set URG flag
  -X  --xmas          set X unused flag (0x40)
  -Y  --ymas          set Y unused flag (0x80)
      --tcpexitcode   use last tcp->th_flags as exit code
      --tcp-timestamp enable the TCP timestamp option to guess the HZ/uptime
Common
  -d  --data          data size                           (default is 0)
  -E  --file          data from file
  -e  --sign          add 'signature'
  -j  --dump          dump packets in hex
  -J  --print         dump printable characters
  -B  --safe          enable 'safe' protocol
  -u  --end           tell you when --file reached EOF and prevent rewind
  -T  --traceroute    traceroute mode                     (implies --bind and --ttl 1)
      --tr-stop       Exit when receive the first not ICMP in traceroute mode
      --tr-keep-ttl   Keep the source TTL fixed, useful to monitor just one hop
      --tr-no-rtt     Don't calculate/show RTT information in traceroute mode
ADS packet description (new, unstable)
      --apd-send      Send the packet described with APD (see docs/APD.txt)
[root@...]
```

Figure 5-9 Hping help, page 3

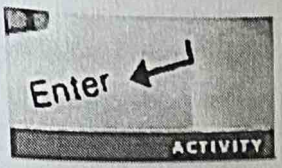
Courtesy Course Technology/Cengage Learning

Security Bytes

If you decide to use ping sweeps, be careful not to include the broadcast address in the range of IP addresses. Including it by mistake might happen if subnetting is used in an organization. For example, if the IP network 193.145.85.0 is subnetted with the 255.255.255.192 subnet mask, four subnets are created: 193.145.85.0, 193.145.85.64, 193.145.85.128, and 193.145.85.192. The broadcast addresses for each subnet are 193.145.85.63, 193.145.85.127, 193.145.85.191, and 193.145.85.255. If a ping sweep is activated inadvertently on the range of hosts 193.145.85.65 to 193.145.85.127, an inordinate amount of traffic could flood the network because the broadcast address 193.145.85.127 is included. This error is more of a problem on a Class B address, but if you perform ping sweeps, make sure your client signs a written agreement authorizing the testing.

Crafting IP Packets

Packets contain source and destination IP addresses as well as information about the flags you learned earlier: SYN, ACK, FIN, and so on. You can create a packet with a specific flag set. For example, if you aren't satisfied with the response you get from the host computer after sending a SYN packet, you can create another packet with the FIN flag set. The SYN flag might have returned a "closed port" message, but a FIN packet sent to the same computer might return a "filtered port" message. You can craft any type of packet you like. Hping and Fping are helpful tools for crafting IP packets, and you work with both tools in Activity 5-3.



Activity 5-3: Crafting IP Packets with Fping and Hping

Time Required: 30 minutes

Objective: Learn to craft IP packets with Fping and Hping.

Description: In this activity, you see how security testers can craft IP packets to find out what services are running on a network. The more ways you know how to send a packet to an unsuspecting port on a computer and get a response, the better. If a computer doesn't respond to an ICMP packet sent to a particular port, it doesn't mean any packet sent to the same port will get the same response. You might need to send different packets to get the results you need for a thorough security test.

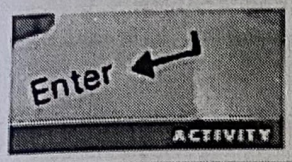
1. If necessary, boot your computer into Linux with the BackTrack DVD. Open a Konsole shell, and then type **fping -h** and press Enter.
2. To see the live computers in the attack range your instructor gave you, type **fping -g** *BeginningIPaddress EndingIPaddress* and press Enter. Note the results. (Be sure to use the beginning and ending IP addresses in your attack range.)
3. Next, type **hping -S** *IPAddressAttackedComputer* (substituting an IP address from your attack range) and press Enter. By using the `-S` parameter, you have just crafted a TCP SYN packet.
4. Open another Konsole shell, and then type **tcpdump** and press Enter.
5. Arrange both shell windows next to each other so that you can observe what happens after entering the Hping command. In the shell that's not running Tcpdump, press Ctrl+C to return to the command prompt, type **hping -S** *IPaddressAttackedComputer*, and press Enter. Watch the Tcpdump window fill with the traffic that's generated. To stop Tcpdump from capturing packets, press Ctrl+C in that shell window.
6. If time permits, consult the Hping help pages (refer to Figures 5-7, 5-8, and 5-9, if needed) and experiment with creating different types of packets. Note the differences in network traffic generated with the Tcpdump command. Security testers need to understand how slight variations in packets sent to an attacked computer can produce different results. For example, if a computer doesn't respond to a SYN packet, try sending an ACK packet. What happens when a FIN packet is sent? If you aren't having any success, try sending the same packets to different ports. Does this method change the response from the attacked computer?
7. When you're done, close both shells.

Understanding Scripting

Some tools might need to be modified to better suit your needs as a security tester. Creating a customized script—a program that automates a task that takes too much time to perform manually—can be a time-saving solution. As mentioned, Fping can use an input file to perform ping sweeps. Creating an input file manually with thousands of IP addresses isn't worth the time, however. Instead, most security testers rely on basic programming skills to write a script for creating an input file.

Scripting Basics

If you have worked with DOS batch programming, scripting will be familiar. If you're from a networking background and new to programming, however, this topic might seem a little overwhelming, but Chapter 7 focuses on getting nonprogrammers up to speed. A script or batch file is a text file containing multiple commands that would usually be entered manually at the command prompt. If you see that you're using a set of commands repeatedly to perform the same task, this task might be a good candidate for a script. You can run the script by using just one command. The best way to learn how to create a script is by doing it, so you get an opportunity to practice writing one in Activity 5-4.



Activity 5-4: Creating an Executable Script

Time Required: 45 minutes

Objective: Learn to create, save, and run an executable script.

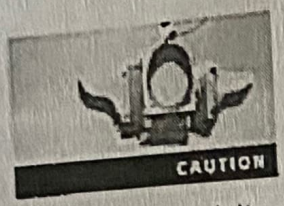
Description: Many hacking tools are written in scripting languages, such as VBScript or JavaScript. In this activity, you create a script that populates a file with a range of IP addresses. This type of file can be used as an input file for Nmap or Fping.

1. If necessary, boot your computer into Linux with the BackTrack DVD, and then open a Konsole shell. Type vi Myshell and press Enter.
2. To activate the screen, press Esc and then press i. Make sure Caps Lock isn't activated because the vi program is case sensitive, and you can get strange results if you don't pay careful attention to letter case. If this is your first time using the vi editor, you might need to use Table 5-1 as a reference. (For a more detailed description of this versatile editor, type man vi in a different Konsole shell and press Enter.)

Table 5-1 Summary of vi commands

vi commands	Description
j	Moves the insertion point down one line
k	Moves the insertion point up one line
h	Moves the insertion point back one character
l (lowercase L)	Moves the insertion point forward one character
Enter key	Moves the insertion point to the beginning of the next line
a	After pressing Esc, appends text after the insertion point
i	After pressing Esc, inserts text before the insertion point
Delete key	Overwrites the last character when in Insert mode
x	Deletes the current character
dd	Deletes the current line
dw	Deletes the current word
p	Replaces the previously deleted text
ZZ	Exits vi and saves all changes
wq	Writes changes and quits the edit session

3. First, type #!/bin/sh and press Enter. This line is important because it identifies the file you're writing as a script. You should enter a few lines of documentation in any scripts or programs you write because they help with program modifications and maintenance done later. When a line is used for documentation purposes, it's preceded with a # character. Figure 5-10 shows examples of documentation comments added, but don't enter them for this activity.



Make sure the slashes point in the correct direction (/). Microsoft users often make this mistake because they're used to typing backslashes (\).

4. The second line is the name of the script you're creating. Type # Myshell and press Enter. If this script were used in a production setting, you would also enter the date and your name.
5. Read the documentation comments added in Figure 5-10 about the purpose of the script, but don't type them in your script. Your script should have only `#!/bin/sh` and `# Myshell` statements so far.
6. Next, type network id="193.145.85." and press Enter. Be sure to include the quotation marks and the period after 85. (Because you aren't actually using this script, the address entered in this line doesn't matter.)
7. Type count=0 and press Enter. You're initializing the `count` variable to zero, which is always wise because a variable shouldn't be used in a program without having a value set. (You learn more about setting values for variables in Chapter 7.)
8. Figure 5-10 shows more documentation comments added as an example, but skip entering them and move on to entering the program code. You need your script to add the number 1 to the 193.145.85. network ID and continue incrementing and adding numbers to the network ID until the IP address range 193.145.85.1 to 193.145.85.254 is written to a file named ip address.txt. In programming lingo, this repeated process is called looping. To avoid creating an endless loop, you need to add a condition to the `while` statement: Type while ["$count" -le 253] and press Enter. Note the spaces inside the square brackets and pay close attention to the use of quotation marks and dollar signs.
9. Next, type do and press Enter. This statement is where the script performs its main task. The action takes place between the do statement and the done statement (added in Step 11). First, to increment the count variable by 1, type count=$(($count+1)), paying careful attention to the parentheses, and press Enter.
10. The next line is covered in more detail in Chapter 7. For now, just understand that you can use the `printf` function to write data to a file. Type printf "%s%s\n" $network id $count >> ip address.txt and press Enter. The >> characters are used to add each IP address to the end of the ip address.txt file.
11. Type done and press Enter, and then type exit 0 and press Enter. Figure 5-10 shows the entire script. Save your hard work by pressing Esc and typing : (a colon). At the : prompt, type wq and press Enter.

```
File  Edit  View  Terminal  Go  Help
#!/bin/sh
# Myshell
# This program creates a text file named ip_address.txt that contains 254
# IP addresses using 193.145.85.0 as the network ID. The file created can
# be used as an input file for the fping utility. For example:
#    fping -f ip_address.txt

# Initialize variables

network_id="193.145.85."
count=0

# Stop the loop when count is equal to 254. The 'le' signifies less than
# or equal to 253, so the count variable will be incremented one more
# time after count is equal to 253. We do not want to create an IP
# address of 193.145.85.255  because this would be the broadcast address
# of the 193.145.85.0/24 network. Ping sweeping a broadcast address can
# be problematic.

while [ "$count" -le 253 ]
do

        count=$((count+1))
        printf "%s%s\n" $network_id $count >> ip_address.txt
done

exit 0

"Myshell" 27L, 818C written                                  2,2           All
```

Figure 5-10 A shell script

Courtesy Course Technology/Cengage Learning

12. Now that you've saved your script, you need to make it executable so that you can run it. At the command prompt, type chmod +x Myshell and press Enter.
13. To run your script, type ./Myshell and press Enter. Because your script doesn't create any output onscreen, you need to examine the contents of the ip address.txt file to see whether the script worked.
14. Type cat ip address.txt. How many IP addresses were created in the ip address.txt file?
15. Close the shell. You can leave your system running for the end-of-chapter projects.

Chapter Summary

- Port scanning, also referred to as service scanning, is the process of examining a range of IP addresses to determine what services are running on a system or network.
- Different port scans might elicit different information, so security testers need to be aware of the port scan types, such as SYN, ACK, FIN, and so on.
- A multitude of port-scanning tools are available. The most popular are Nmap, Nessus, OpenVAS, and Unicornscan.
- Ping sweeps are used to determine which computers on a network are "live" (computers the attack computer can reach).
- Using scripts can help security professionals by automating time-consuming tasks.