

Take advantage of the enemy's unreadiness,
make your way by unexpected routes,
and attack unguarded spots.

—Sun Tzu

Performing Host Reconnaissance

The Duke of Wellington, who fought Napoleon at Waterloo, once said, “The most difficult part of warfare was seeing what was on the other side of the hill.” Wellington realized that success at war meant more than combat; it also involved secrecy and reconnaissance.

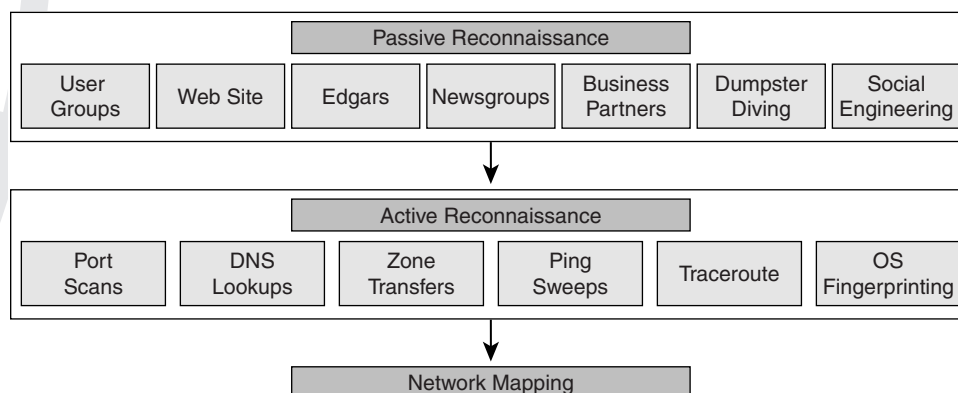
Malicious hackers also value reconnaissance as the first step in an effective attack. For them, seeing what is on the “other side of the hill” is crucial to knowing what type of attack to launch. Launching attacks pertaining to UNIX vulnerabilities if the target is running only Microsoft servers makes no sense. A little time spent investigating saves a lot of time during the penetration attack. A malicious hacker might scope out a target for months before attempting to breach its security.

Although penetration testers might not always have the luxury of time that a malicious hacker might have, they do recognize the value of reconnaissance. The goal of reconnaissance is to discover the following information:

- IP addresses of hosts on a target network
- Accessible User Datagram Protocol (UDP) and Transmission Control Protocol (TCP) ports on target systems
- Operating systems on target systems

Figure 5-1 illustrates the process of unearthing this information.

Figure 5-1 *Passive and Active Reconnaissance*



Passive reconnaissance, as the figure shows, involves obtaining information from user group meetings, websites, Edgars' database, UUNet newsgroups, business partners, dumpster diving, and social engineering. Passive reconnaissance takes patience, but it is the most difficult for the target company to detect. Active reconnaissance, in contrast, involves using technology in a manner that the target might detect. This could be by doing DNS zone transfers and lookups, ping sweeps, traceroutes, port scans, or operating system fingerprinting. After you gather the information, you create a network map that diagrams the live hosts, their open UDP and TCP ports (which offers hints to the type of applications running on the hosts), and their respective operating systems. This information forms the skeleton to knowing what type of attacks to launch.

In this chapter, you learn how to discover live hosts on your target network using these various information-gathering techniques. Using port-scanning tools, you also learn how to determine the operating systems and open TCP and UDP ports on your target hosts. Finally, you learn best practices for the detection and prevention of reconnaissance techniques.

Passive Host Reconnaissance

As previously mentioned, you can use two different reconnaissance methods to discover information on the hosts in your target network:

- Passive reconnaissance
- Active reconnaissance

Passive reconnaissance gathers data from open source information. Open source means that the information is freely available to the public. Looking at open source information is entirely legal. A company can do little to protect against the release of this information, but later sections of this chapter explore some of the options available. Following are examples of open source information:

- A company website
- Electronic Data Gathering, Analysis, and Retrieval (EDGAR) filings (for publicly traded companies)
- Network News Transfer Protocol (NNTP) USENET Newsgroups
- User group meetings
- Business partners
- Dumpster diving
- Social engineering

All of these, with the exception of dumpster diving and social engineering, are discussed in this chapter. Review Chapter 4, "Performing Social Engineering," for more information about dumpster diving and social engineering.

A Company Website

If you are hired to perform a penetration test against a company's Internet presence, the first place you should look, obviously, is the company website. Begin by downloading the website for offline viewing. This allows you to spend more time analyzing each page without being detected and provides benefits later when you attempt to penetrate the website. In the process of downloading the website, you can also collect orphan pages. Orphan pages are web pages that might have been parts of the company website at one time but now have no pages linking to them. While the pages should be removed from the server, they often are not. They can contain useful information for the penetration tester.

Two programs that you can use for downloading a website for offline viewing are GNU Wget (<ftp://ftp.gnu.org/pub/gnu/wget/>) and Teleport Pro (<http://www.tenmax.com>). GNU Wget is free under the GNU license and can be run under Linux or Windows. Teleport Pro is commercial software that runs only on Windows.

Wget is a noninteractive command-line-driven website retrieval application that creates local copies of remote websites. Figure 5-2 shows Wget retrieving the pages off of <http://www.hackmynetwork.com>. Notice the use of the `-r` switch, which enables recursive mirroring of all pages on the site. You can specify the recursion maximum depth level with the `-l` switch. If you select the recursive option, Wget follows the hyperlinks and downloads referenced pages. Wget continues following hyperlinks up to the depth specified in the `-l` option.

The goal of penetration testing is not only to see what access the auditor can gain, but also what the auditor is able to do without being detected. To minimize the possibility of detection when using Wget, you can use the following switches:

- **--random-wait**—Because some websites perform statistical analysis of website viewing to detect spidering and web retrieval software, you should use the **--random-wait** switch to vary your retrieval between 0 and $2 * \textit{wait}$ seconds. *Wait* refers to the time specified with the *wait* switch.
- **--wait=seconds**—This switch specifies the number of seconds between retrievals. You should use this along with the **--random-wait** switch.
- **--cookies=on/off**—Cookies enable web servers to keep track of visitors to their websites. Disabling this switch prevents the server from tracking your viewing of its website; however, you might want this enabled for cookie-based exploits discussed later in Chapter 7, “Performing Web-Server Attacks.”
- **--H**—This switch enables host spanning. Host spanning allows Wget not only to collect web pages on your target site but also enable recursive mirroring of any sites referenced by hyperlinks on the web pages. Be careful with this switch because it then mirrors the referenced site and any sites it references. This can consume a significant amount of hard drive space.
- **--D**—This is the domain switch that, when used with the **--H** switch, limits host spanning to only the domains listed.

Figure 5-2 Wget Web Retrieval

```

C:\wget\wget-1.9.1b>wget -r www.hackmynetwork.com
--09:48:57-- http://www.hackmynetwork.com/
=> `www.hackmynetwork.com/index.html'
Resolving www.hackmynetwork.com... 127.0.0.1
Connecting to www.hackmynetwork.com[127.0.0.1]:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 8,582 [text/html]

100%[=====] 8,582      --.-K/s

09:48:57 (8.18 MB/s) - `www.hackmynetwork.com/index.html' saved [8582/8582]
Loading robots.txt; please ignore errors.
--09:48:57-- http://www.hackmynetwork.com/robots.txt
=> `www.hackmynetwork.com/robots.txt'
Reusing connection to www.hackmynetwork.com:80.
HTTP request sent, awaiting response... 404 Not Found
09:48:57 ERROR 404: Not Found.

--09:48:57-- http://www.hackmynetwork.com/design_01.jpg
=> `www.hackmynetwork.com/design_01.jpg'
Connecting to www.hackmynetwork.com[127.0.0.1]:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 4,441 [image/jpeg]

100%[=====] 4,441      --.-K/s

09:48:57 (4.24 MB/s) - `www.hackmynetwork.com/design_01.jpg' saved [4441/4441]

--09:48:57-- http://www.hackmynetwork.com/design_02.jpg
=> `www.hackmynetwork.com/design_02.jpg'
Reusing connection to www.hackmynetwork.com:80.
HTTP request sent, awaiting response... 200 OK
Length: 1,774 [image/jpeg]

100%[=====] 1,774      --.-K/s

09:48:57 (1.69 MB/s) - `www.hackmynetwork.com/design_02.jpg' saved [1774/1774]

--09:48:57-- http://www.hackmynetwork.com/design_03.jpg
=> `www.hackmynetwork.com/design_03.jpg'
Reusing connection to www.hackmynetwork.com:80.
HTTP request sent, awaiting response... 200 OK
Length: 46,794 [image/jpeg]

```

Because you will probably use the same switches each time you use Wget, you can include the switches in the wget.rc file. By listing the switches in this file, you do not have to type them every time you launch the program. The syntax might vary slightly from the switches previously listed, so be sure to read the documentation before you create the file.

NOTE

Some CGI programs can cause problems with Wget. If you notice Wget attempting to download the same file repeatedly, use the **--ignore-length** switch. This switch circumvents issues caused by CGI scripts that send out bogus content-length headers.

If command-line switches are not your thing, you can use the Windows-based Teleport Pro program from Tennyson Maxwell. After you launch Teleport Pro, you are prompted with the New Project Wizard, as shown in Figure 5-3.

Figure 5-3 *Teleport Pro New Project Wizard*



For the purpose of offline browsing, select the **Create a browsable copy of a website on my hard drive** option. After you click **Next**, you are prompted with the screen shown in Figure 5-4.

On the Starting Address screen, enter the website you want to store offline. Note that the address is case-sensitive. You can choose how deep you want Teleport Pro to explore. The default is up to three links, which is sufficient for most retrievals. On the next screen (Project Properties), shown in Figure 5-5, you can specify what type of files you want to retrieve. Teleport Pro is limited to retrieving the files displayed in the Project Properties screen options. Typically, you would choose the **Everything** option, but if you are on limited bandwidth and do not care about graphics, you can choose the **Just text** option.

Figure 5-4 *Teleport Pro Starting Address Screen*



Figure 5-5 *Teleport Pro Project Properties Screen*

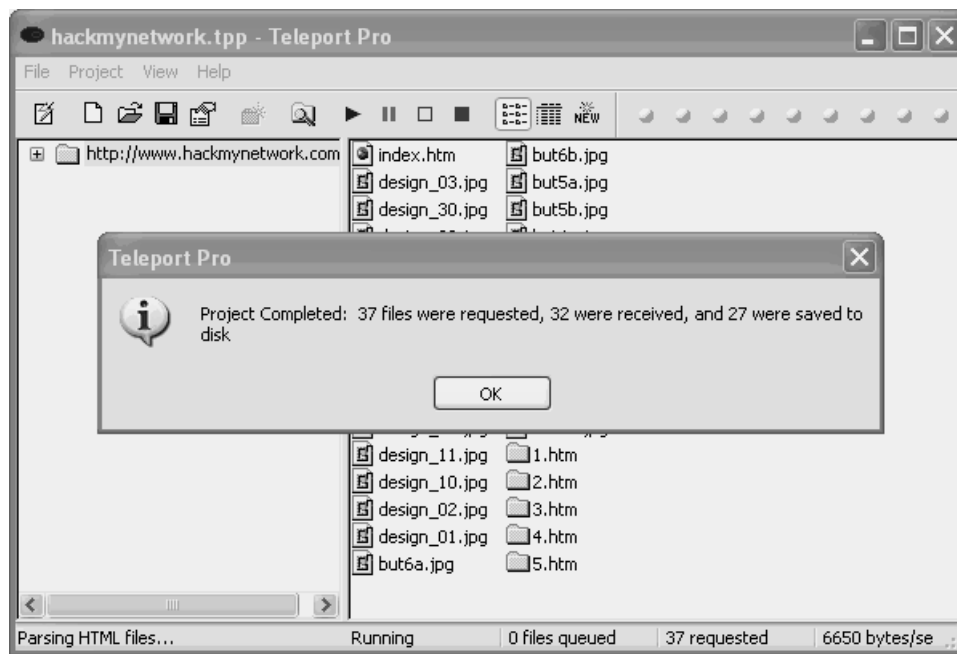


You can also enter an account name and password to access the site if it is needed, but because you probably do not know any usernames or passwords at this point (you will learn how to discover these in Chapter 7), you should leave this blank.

After you select **Next**, you are prompted to finish the wizard and select where to save the project file. Having a project file is useful if you want to return and copy the website again.

When you are ready to begin copying the target website, you can either go to the Project menu and choose **Start** or click the **Play** button on the toolbar. When the project is finished, you see a screen like that in Figure 5-6, which shows you how many files were requested and how many were received. If the number of failed requests is high, you can change retrieval parameters with the Project Properties screen under the Project menu.

Figure 5-6 *Teleport Pro Retrieval Completion Screen*



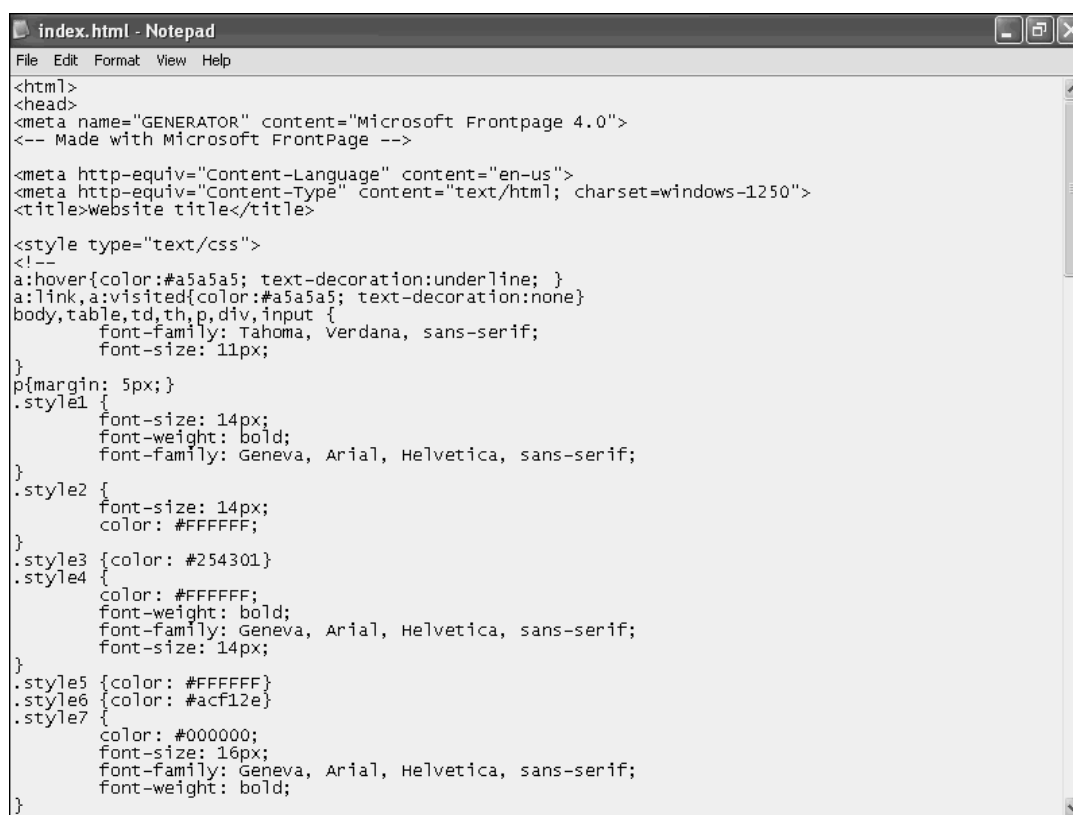
After you have copied down the website, either through Teleport Pro or Wget, you can browse it offline in your preferred web browser. With respect to host reconnaissance, you should be looking for two things:

- Comments in the source code
- Contact information

Comments in the source code might reveal what platform the website is running on, which is useful later when you attempt to infiltrate the target web server. You can view the source

code by opening the web pages in an HTML editor, text editor, or within the browser. In Internet Explorer, you can view source code by choosing **Source** under the View menu. Figure 5-7 shows sample source code of a web page.

Figure 5-7 Sample Web Page Source Code: Comments Reveal HTML Authoring Tool Used



```
index.html - Notepad
File Edit Format View Help
<html>
<head>
<meta name="GENERATOR" content="Microsoft Frontpage 4.0">
<!-- Made with Microsoft FrontPage -->

<meta http-equiv="Content-Language" content="en-us">
<meta http-equiv="Content-Type" content="text/html; charset=windows-1250">
<title>website title</title>

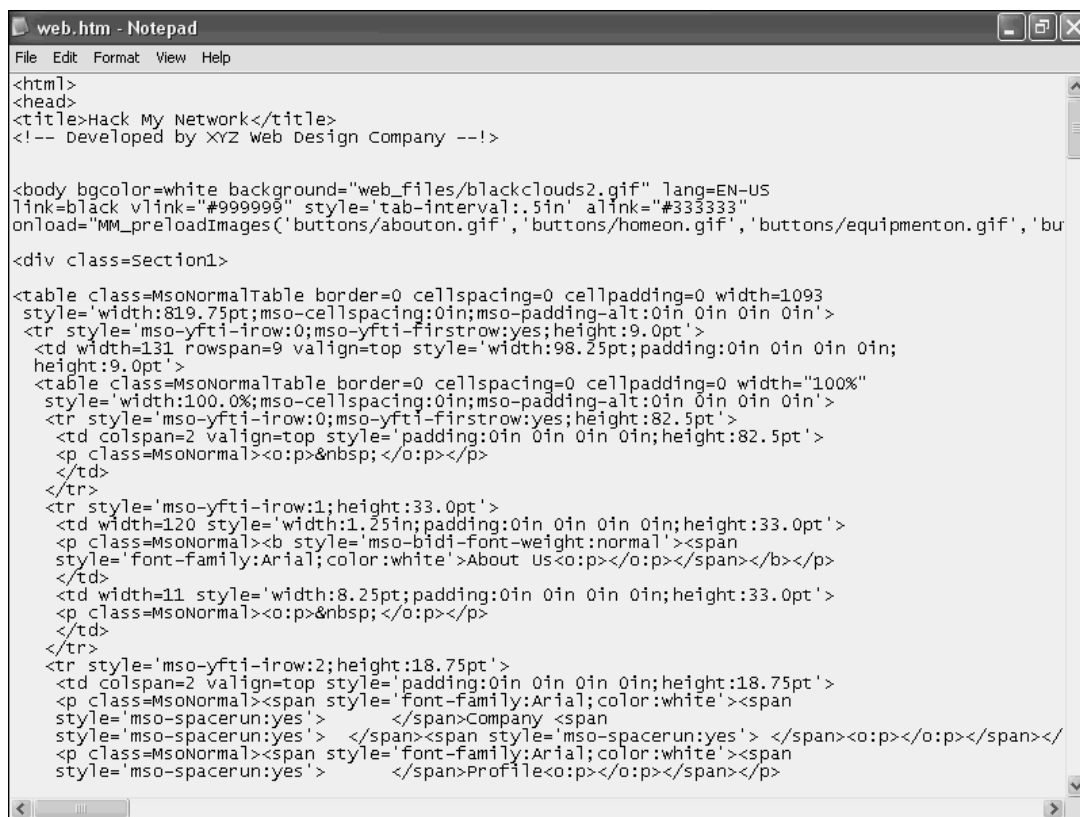
<style type="text/css">
<!--
a:hover{color:#a5a5a5; text-decoration:underline; }
a:link,a:visited{color:#a5a5a5; text-decoration:none}
body,table,td,th,p,div,input {
    font-family: Tahoma, Verdana, sans-serif;
    font-size: 11px;
}
p{margin: 5px;}
.style1 {
    font-size: 14px;
    font-weight: bold;
    font-family: Geneva, Arial, Helvetica, sans-serif;
}
.style2 {
    font-size: 14px;
    color: #FFFFFF;
}
.style3 {color: #254301}
.style4 {
    color: #FFFFFF;
    font-weight: bold;
    font-family: Geneva, Arial, Helvetica, sans-serif;
    font-size: 14px;
}
.style5 {color: #FFFFFF}
.style6 {color: #acf12e}
.style7 {
    color: #000000;
    font-size: 16px;
    font-family: Geneva, Arial, Helvetica, sans-serif;
    font-weight: bold;
}
}
```

Comments start with the `<!--` HTML tag and end with `-->`. Figure 5-7 shows that the web page was written with Microsoft FrontPage. Exploits are related to Microsoft FrontPage, so document this information for later.

Figure 5-8 shows another example of useful comment information. Here, you can see that this site was developed by XYZ Web Design Company. Although at first glance this might not reveal much, it is actually useful information. Many web design companies advertise what type of platform they develop sites for, such as Microsoft or UNIX. By going to the XYZ Web Design website, you might learn that they specialize in ASP, .NET, and

FrontPage. You can conclude with fair certainty that because these specializations are all used on Microsoft platforms, the target website is running on Microsoft Internet Information Server (IIS). If XYZ Web Design advertises that its specialty is Perl, CGI, PHP, and Python, the target website is most likely running on a UNIX-based platform. Although all of these technologies can also run on Windows, they are more common on the UNIX platform.

Figure 5-8 Sample Web Page Source Code: A Third-Party Web Developer Is Revealed



```

web.htm - Notepad
File Edit Format View Help
<html>
<head>
<title>Hack My Network</title>
<!-- Developed by XYZ web Design Company --!>

<body bgcolor=white background="web_files/blackclouds2.gif" lang=EN-US
link=black vlink="#999999" style='tab-interval:.5in' alink="#333333"
onload="MM_preloadImages('buttons/abouton.gif','buttons/homeon.gif','buttons/equipmenton.gif','bu

<div class=section1>

<table class=MsoNormalTable border=0 cellspacing=0 cellpadding=0 width=1093
style='width:819.75pt;mso-cellspacing:0in;mso-padding-alt:0in 0in 0in 0in'>
<tr style='mso-yfti-irow:0;mso-yfti-firstrow:yes;height:9.0pt' >
<td width=131 rowspan=9 valign=top style='width:98.25pt;padding:0in 0in 0in 0in;
height:9.0pt'>
<table class=MsoNormalTable border=0 cellspacing=0 cellpadding=0 width="100%"
style='width:100.0%;mso-cellspacing:0in;mso-padding-alt:0in 0in 0in 0in'>
<tr style='mso-yfti-irow:0;mso-yfti-firstrow:yes;height:82.5pt'>
<td colspan=2 valign=top style='padding:0in 0in 0in 0in;height:82.5pt'>
<p class=MsoNormal><o:p>&nbsp;</o:p></p>
</td>
</tr>
<tr style='mso-yfti-irow:1;height:33.0pt'>
<td width=120 style='width:1.25in;padding:0in 0in 0in 0in;height:33.0pt'>
<p class=MsoNormal><b style='mso-bidi-font-weight:normal'><span
style='font-family:Arial;color:white'>About Us<o:p></o:p></span></b></p>
</td>
<td width=11 style='width:8.25pt;padding:0in 0in 0in 0in;height:33.0pt'>
<p class=MsoNormal><o:p>&nbsp;</o:p></p>
</td>
</tr>
<tr style='mso-yfti-irow:2;height:18.75pt'>
<td colspan=2 valign=top style='padding:0in 0in 0in 0in;height:18.75pt'>
<p class=MsoNormal><span style='font-family:Arial;color:white'><span
style='mso-spacerun:yes'> </span><span>Company <span
style='mso-spacerun:yes'> </span><span style='mso-spacerun:yes'> </span><o:p></o:p></span></p>
<p class=MsoNormal><span style='font-family:Arial;color:white'><span
style='mso-spacerun:yes'> </span><span>Profile<o:p></o:p></span></p>

```

After you look at the source code, examine any contact information published on the target site. Typically, you can find this by clicking on links labeled About Us or Contact Information. Figure 5-9 shows an example of a page with information about the company executives.

Figure 5-9 Sample Contact Information Web Page



On this web page, you see executive names along with their phone numbers and e-mail information. This can be useful for performing social engineering, as discussed in Chapter 4. The phone numbers in Figure 5-9 are also useful for war dialing techniques, in which you dial a range of phone numbers with software such as Tone Loc or THC war dialer and attempt to establish remote access connectivity. In the figure, all phone numbers start with the prefix 503 555-1 followed by the extension number. Armed with this knowledge, you can configure your war dialing software to dial all numbers within the range 503 555-1000 through 503 555-1999 and detect modems used for remote access.

If possible, companies should list only 800 numbers on their website connecting the caller to a receptionist to minimize the risk of war dialing attacks. If employee information is to be displayed, make sure policies are in place and enforced that protect against social engineering attacks.

NOTE Companies that provide technology solutions are particularly at risk because they often advertise their platform of choice on their website. For example, some online banking companies advertise that they run solely on Microsoft IIS and SQL servers. Although this information might be helpful for marketing purposes, it should not be public knowledge. Instead, sales personnel can give out the information to potential clients who request it.

EDGAR Filings

Publicly traded companies in the United States are required to file with the Security and Exchange Commission (SEC). You can access this information through the EDGAR database, which you can view at <http://www.sec.gov/edgar.shtml>. Searches can reveal financial information and press releases. Some companies advertise the technology used in their organization in a press release posted to EDGAR filings. This saves time when trying to determine the operating system through other means.

NNTP USENET Newsgroups

If you have ever had to troubleshoot a difficult problem, you know the value of networking with others to find a solution. One of the methods that engineers use to seek help is by posting questions on USENET newsgroups. Unfortunately, some post too much information when they are seeking help.

Example 5-1 shows an engineer named Bill asking a question about a problem he is experiencing. In his message, he describes that he is running Red Hat Linux 6.2. No company should give up this information so freely to the public.

Example 5-1 *Sample Newsgroup Posting*

```
From: bsmith@hackmynetwork.com
Subject: Apache Problem
Newsgroups: comp.infosystems.www.servers.unix, comp.os.linux,
alt.apache.configuration, comp.lang.java.programmer
Date: 2004-07-07 09:19:28 PST

I am having a problem with Apache reverse proxy not communicating with web
applications using HTTP 1.1 keepalive. I am using Apache 1.3.23 on Red Hat Linux
6.2. It is compiled with mod_proxy and mod_ssl.

Any ideas would be greatly appreciated.
Thank you.

-----
bsmith@hackmynetwork.com
Sr. Systems Administrator
Hackmynetwork.com
```

Example 5-1 also shows the e-mail address of Bill: bsmith@hackmynetwork.com. This not only reveals the name of the company Bill works for (Hackmynetwork), but also might reflect his user account on the network. Unfortunately, many companies still use the same network logon name as their e-mail name. Although you can not know for certain, you should document his e-mail address when doing host reconnaissance. Because he works on production servers for the target company, you might be able to gain full access to his network if you crack his password. (You will learn more about password cracking in Chapter 9, “Cracking Passwords.”)

You can browse newsgroups using software such as Microsoft Outlook Express, Netscape, Xnews, and many others. Alternatively, and perhaps more effectively, you can search newsgroups using Google. Just enter the name of the target company, and you will obtain all newsgroup messages posted from or related to your target company.

User Group Meetings

If searching through thousands of Newsgroup messages is not your forte (or your idea of a fun afternoon), you might try attending user group meetings. Most cities hold user group meetings related to various technologies, such as Microsoft development, Cisco technologies, Linux, and even penetration testing. User group meetings provide an opportunity for people in a community to receive information and meet others who work with the same technology.

Attending user groups is a great way to practice your social engineering skills learned in Chapter 4. By arriving early or staying late after the meeting, you can network with others and discover what companies people work for and what technologies their companies use.

Of course, knowing which user groups your target employees are attending is difficult. Penetration testers should frequent user group meetings and talk to as many people as possible at each meeting. When a client requests your service, you might already know that the client runs Microsoft servers, for example, because you met an employee of the client at a Microsoft user group.

Business Partners

If perusing a target website, searching EDGAR filings and newsgroups, or attending user groups does not provide you with the information you need, you might have to check the business partners of the target for more information. Although the target might protect against giving away technical information, the partners might not.

A company website often reveals who the business partners are, but a more effective means of obtaining partner information is using Google. For instance, if you enter **link:www.hackmynetwork.com** in the Google search box, your search pulls up all sites that link back to your target site.

By going to all the sites listed in your search results, you might uncover technologies in use by your target. Network integrators are notorious for listing their client names and the technologies they specialize in. If you see a network integrator that specializes in Sun Solaris solutions and links back to your target website, you can safely assume that your target is running on Sun Solaris servers.

Active Host Reconnaissance

Although the passive reconnaissance means are effective, they are often time intensive and do not always produce the most accurate results. In active reconnaissance, you use technical tools to discover information on the hosts that are active on your target network. The drawback to active reconnaissance, however, is that it is easier to detect. For example, consider a criminal who walks past a house she wants to burglarize (passive reconnaissance) versus looking into each window of the house to see what goods are inside (active reconnaissance). Obviously, a burglar peeking into the windows of a house is much more conspicuous than simply walking past it. The same is true for active reconnaissance. It reveals more information but is detected easily.

Some of the tools that are useful in active host reconnaissance include the following:

- NSLookup/Whois/Dig lookups
- SamSpade
- Visual Route/Cheops
- Pinger/WS_Ping_Pro

NSLookup/Whois Lookups

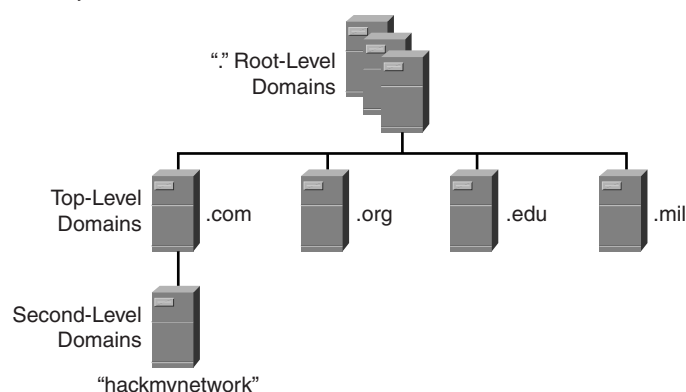
When you are doing black-box testing and you are not given detailed information on the target network, the client might give you only a network range of IP addresses to test. Often, you might be given only the website address, leaving you to discover the network range on your own. In this case, you have to perform some DNS lookups to ascertain the IP addresses associated with the website.

Before you can venture into performing DNS lookups, you need to understand how DNS works. The Domain Name System (DNS) allows you to use friendly names, such as <http://www.cisco.com>, instead of IP addresses when referencing hosts on an IP network.

NOTE RFCs 1034 and 1035 define DNS operation. You can read about them at <http://www.ietf.org>.

DNS is a hierarchical, distributed database shared among servers and queried by hosts and other servers. The highest level of the hierarchy is the last label in a domain name. Top-level names can be either two- or three-letter organizational designators, such as .com for commercial or .edu for educational organizations, .biz for businesses, or two-letter country designators, such as .uk for the United Kingdom or .au for Australia. Figure 5-10 shows the DNS hierarchy for the website <http://www.hackmynetwork.com>. Companies register their DNS with a naming authority, such as ARIN in the United States or RIPE in Europe.

Figure 5-10 *DNS Hierarchy*



A contiguous portion of the DNS namespace is called a zone. A zone can contain one or more domain names. When an update needs to be made to a DNS zone, it is done to a primary zone on a master server. Secondary zones are copies of the primary zone that have been replicated from the master server. A server can house multiple zones with both primary and secondary copies. When a secondary DNS server needs to replicate from the master server, it performs a zone transfer. The section “SamSpade,” later in this chapter, discusses zone transfers in more detail.

Included in the zone information are resource records (RRs). Several types of resource records define information about the hosts in a domain. Table 5-1 defines the different types of record types.

Table 5-1 *DNS Resource Records*

Record	Type	Used for
A	Host record	Single hosts
MX	Mail record	Mail servers
PTR	Pointer record	IP to name reverse lookups
CNAME	Alias record	Creating aliases
NS	Name Service record	DNS servers
SOA	Start-of-Authority record	A master record for the entire zone

When you are performing a penetration test, do DNS lookups to get IP address information of hosts on your target network. DNS lookups can also give you information on the purpose of the host. For example, if an MX record exists for a host named smtp.hackmyntework.com, you know that the host is being used for e-mail because MX is the record for mail exchange.

If DNS servers are the doors to discovering what public hosts belong to your target site, Whois, NSLookup, and Dig are the keys to unlocking those doors.

Whois (RFC 812) is found installed by default on most UNIX and Linux platforms, but on Windows, you need third-party software such as SamSpade to perform Whois queries.

Whois, which in its early days was called NICNAME, is a TCP transaction-based query/response utility to look up registration information for a specific domain. You can obtain Whois at <http://www.linux.it/~md/software>. By default, Whois queries servers set by the NICNAMSERVER and WHOISSERVER environment variables, and, if neither is set, it queries whois.crsnic.net. Typing **whois** without any options reveals the default server being used in the query. Example 5-2 shows the output of a query on hackmynetwork.com.

Example 5-2 *Sample Whois Query*

```
#whois hackmynetwork.com
Registrant:
HackMyNetwork (hackmynetwork-DOM)
 123 Main Street
Portland, OR 97415
Domain Name: hackmynetwork.com

Administrative Contact:
John Nobody (RJXX2-ORG) hackmynetwork@HD1.VSNL.NET.IN
HackMyNetwork
123 Main Street
Portland, OR 97415

Technical Contact:
John Nobody (VSXX) jnobody@hackmynetwork.com
123 Main Street
Portland, OR 97415
Record expires on 14-Nov-2006
Record created on 13-Nov-2003
Dataabase last updated on 17-May-2004

Billing contact:
John Nobody
123 Main Street
Portland, OR 97415

Domain servers in listed order:
NS1.hackmynetwork.com 172.29.140.12
NS2.hackmynetwork.com 172.22.145.12
```


Whois queries are useful for two purposes:

- You learn administrative contact information that is helpful in social engineering. (For more on social engineering, see Chapter 4.)
- You learn the authoritative DNS servers for the domain. As you will see shortly, this is helpful when you want to attempt a DNS zone transfer with a tool such as SamSpade.

NSLookup, Dig, and Host are three other command-line tools that you can use to unearth information about your target network. NSLookup is available on both UNIX and Windows platforms, although NSLookup is being deprecated on most Linux systems, with Dig and Host being its replacement. NSLookup can reveal additional IP addresses and records when the authoritative DNS server is known. Example 5-3 shows an NSLookup query.

Example 5-3 *NSLookup Query*

```
#nslookup
>set type=mx
>hackmynetwork.com
Server: smtp.hackmynetwork.com
Address: 172.28.135.16

Non-authoritative answer:
hackmynetwork.com
  origin = hackmynetwork.com
  mail addr: webmaster.hackmynetwork.com
  serial = 20108130
  refresh = 720 (2H)
  retry = 3600 (1H)
  expire = 1728000 (2w6d)
  minimum ttl = 7200 (2H)
hackmynetwork.com  nameserver = ns1.hackmynetwork.com
```

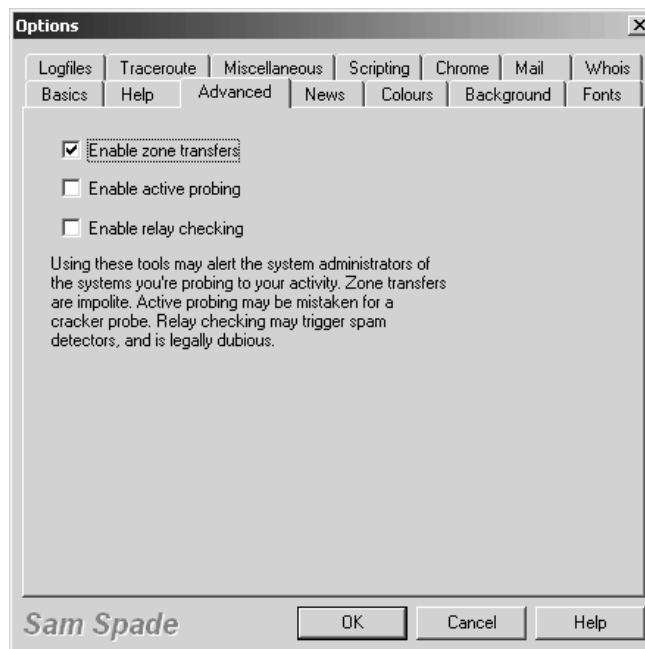
Although NSLookup and Dig are effective tools, they are limited compared to SamSpade.

SamSpade

If the tools previously discussed in this chapter are like taking files out of a filing cabinet, DNS Zone transfers are like taking the entire drawer of files out. DNS servers perform zone transfers to keep themselves up to date with the latest information. In a secured environment, these zone transfers should be restricted to DNS servers that need to exchange information; however, in most environments, this is not the case. A zone transfer of a target domain gives you a list of all public hosts, their respective IP addresses, and the record type.

Although you can use command-line tools like Dig to perform zone transfers, you might prefer a graphical tool like SamSpade (<http://www.samspade.org>). SamSpade is a free Windows tool created by Steve Atkins. It can perform a plethora of functions, including DNS lookups, mail relay checking, and website parsing. SamSpade can also attempt to do zone transfers. In the words of SamSpade's creator, however, "zone transfers are impolite." As such, they are disabled by default. To enable zone transfer functionality, you need to go to the Edit menu and select **Options**. From there, select the **Advanced** tab, as shown in Figure 5-11. Check the **Enable zone transfers** check box to enable this option.

Figure 5-11 *SamSpade Advanced Options: Enable Zone Transfers*



Before you can perform a zone transfer, you need to know what the authoritative name server is, which you can find out by querying your own name server. Enter the IP address of your DNS server by going to the Basics screen, as shown in Figure 5-12. Under TCP/IP settings, you can choose either to learn your DNS information via DHCP or statically enter in your DNS server IP address. After that, click **OK** to exit out of the Options screen.

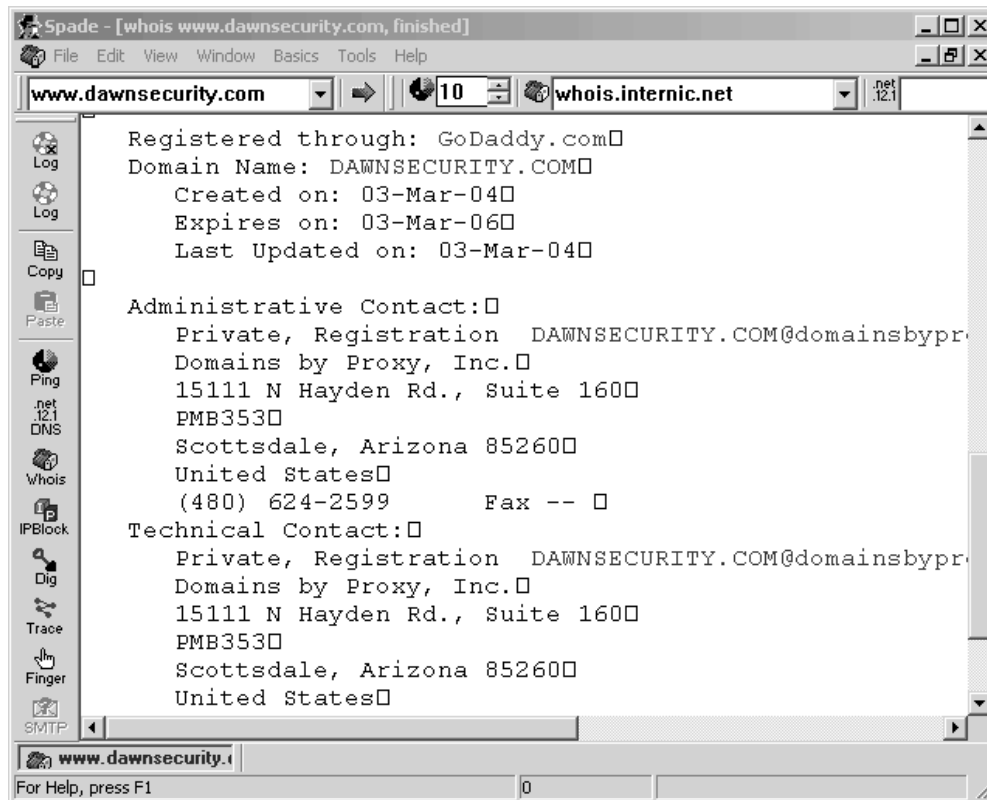
Figure 5-12 SamSpade Basic Options



Now you can perform a DNS lookup by entering the website domain name in the Address box. In Figure 5-13, the domain name `www.dawnsecurity.com` is entered. The output reveals the name of the company that registered this domain name in addition to administrative and technical contact information. Not shown in the graphic is the authoritative DNS server address of `PARK15.SECURESERVER.NET`, which is also included in DNS lookups. Equipped with this address, you can attempt a DNS zone transfer.

Begin your attempt by going to the Tools menu and choosing **Zone Transfer**. You are shown a screen like that in Figure 5-13. Enter the domain name of your target and the IP address of the authoritative DNS server that you discovered in the previous step. You have the option of displaying the output within SamSpade or saving the output to a file. First view the information within SamSpade to determine if you can perform a zone transfer. Then, if you are successful, you can save the output to a file for later viewing.

Figure 5-13 SamSpade DNS Lookup

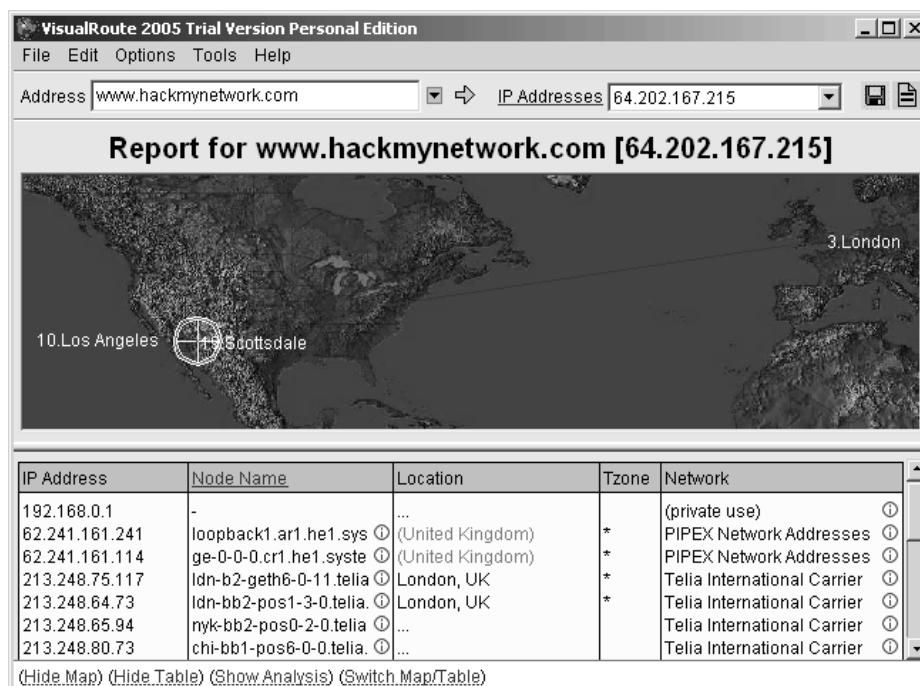


Visual Route

Although SamSpade provides excellent output and should be part of any penetration tester toolkit, it does not provide graphical maps or detailed information of hops along the way to the destination. To see a representation of a packet traveling across the Internet to a target destination, you need a tool like Visual Route. Visual Route (<http://www.visualware.com>) runs on Linux, Windows, Solaris, and Mac OS X.

Figure 5-14 shows the Visual Route screen. A trace is run from a computer in London to the website <http://www.hackmynetwork.com>. Visual Route lists each hop along the way to the site, along with the IP addresses and millisecond delay.

Figure 5-14 Visual Route



What makes Visual Route interesting is that you can double-click on any of the hops along the way and perform a Whois query. The information is the same as you get in a Whois lookup, but Visual Route is more graphically appealing and makes it easy to quickly look up information. You can save both the Whois lookups and the visual map in .jpg or .png format, making it perfect for penetration testers who are preparing reports for clients.

Port Scanning

Now that you know what hosts are publicly accessible on your target network, you need to determine what ports are open on these hosts. You can do this through port scanning, which is the process of scanning a host to determine which TCP and UDP ports are accessible.

Most network applications today run on top of TCP or UDP. These protocols are the transport mechanism used by such applications as FTP, Simple Mail Transfer Protocol (SMTP), Dynamic Host Configuration Protocol (DHCP), and HTTP. TCP is a connection-oriented protocol, which means it provides reliability by establishing a connection between hosts. In contrast, UDP is a connectionless protocol; it does not provide reliability.

TCP is analogous to delivering a package via priority mail where the recipient has to sign for the package, making the delivery reliable. In comparison, UDP is analogous to regular postal mail, which provides no guarantee that the package will be delivered. UDP applications, such as DHCP, rely on the application to provide reliability if necessary. Applications that use TCP (such as FTP) have mechanisms built into the TCP protocol to provide reliability.

TCP and UDP identify the applications they are transporting through port numbers. Table 5-2 lists common TCP and UDP port numbers. It makes sense, then, to determine what applications are running on your target host. You should look to see what TCP and UDP ports are open on the host by performing a port scan.

Table 5-2 *Port Numbers*

TCP		UDP	
Application	Port Number(s)	Application	Port Number(s)
FTP	20–21	DNS	53
Telnet	23	DHCP	67–68
SMTP	25	TFTP	69
DNS	53	NTP ¹	123
HTTP	80	SNMP ²	161
POP ³	110		
NNTP ⁴	119		
HTTPS ⁵	443		

1 NTP = Network Time Protocol

2 SNMP = Simple Network Management Protocol

3 POP = Post Office Protocol

4 NNTP = Network News Transfer Protocol

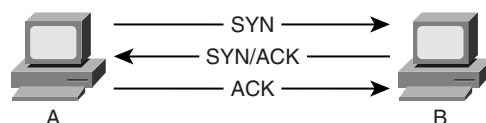
5 HTTPS = Hypertext Transfer Protocol Secure

Port scans are available in numerous types, including these:

- TCP Connect() scan
- SYN
- NULL
- FIN
- ACK
- Xmas-Tree
- Dumb scan
- Reverse Ident

The TCP connect() port scan attempts to create an established connection with the target host. An established connection is one that has completed the three-way handshake that occurs when two hosts initiate communication with each other, as illustrated in Figure 5-15.

Figure 5-15 *Three-Way Handshake*

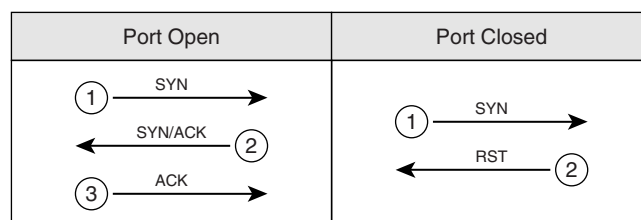


As the figure shows, when Computer A seeks to create a TCP connection to Computer B, it sends out a synchronize (SYN) packet with its initial sequence number (ISN). The initial sequence number is a pseudorandom number between 0 and $2^{32}-1$ (4,294,967,295). Computer B sends an acknowledgement (ACK) back with the ISN+1 of Computer A, indicating the next sequence number it predicts. Computer B also sets the SYN flag and includes its own ISN. Computer A then responds to Computer B with an ACK to acknowledge the SYN packet of Computer B. The ACK sequence number is the ISN+1 of Computer B, indicating the next sequence number it expects from Computer B. Going through this initial handshake provides reliability because any deviation from the handshaking process or any discrepancy of sequence number causes the computers to send reset (RST) packets, thus dropping the connection.

TCP Connect() Scan

A TCP Connect() scan attempts the three-way handshake with every TCP port. Going through the entire three-way handshake as shown in Figure 5-16 provides the best accuracy when performing a port scan. However, this type of scan is also the most easily detected by firewalls and intruder detection systems. Therefore, you should look at using other types of scans that have a better chance of avoiding detection.

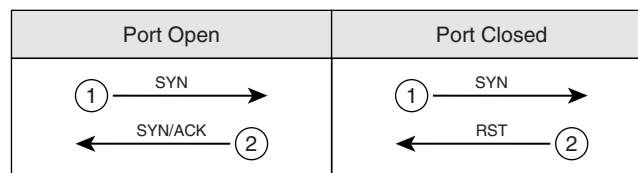
Figure 5-16 *TCP Connect() Scan*



SYN Scan

A slightly stealthier approach to port scans is to perform a SYN scan. As mentioned earlier, the TCP three-way handshake involves SYN, SYN-ACK, and ACK packets (in that order). A SYN scan only sends out the initial SYN to the target. As shown in Figure 5-17, if the port is open, the target responds with a SYN-ACK. If it is closed, it responds with an RST.

Figure 5-17 SYN Scan

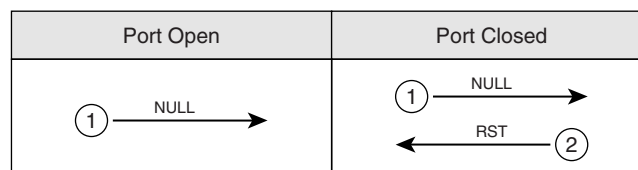


At this point, the behavior of a SYN scan is exactly like a TCP Connect() scan. What makes it different, however, is what the SYN scan does next. Computer A does not respond with an ACK packet, which is the expected response in the three-way handshake. Instead, Computer A responds with an RST packet, dropping the connection. By dropping the connection before the session can become established, the SYN scan can go unnoticed by some firewalls. However, many intrusion detection systems (IDSs) detect SYN scans, so you should avoid this approach also.

NULL Scan

In a NULL scan, a packet is sent to a TCP port with no flags set. In normal TCP communication, at least one bit—or flag—is set. In a NULL scan, however, no bits are set. RFC 793 states that if a TCP segment arrives with no flags set, the receiving host should drop the segment and send an RST. As Figure 5-18 illustrates, when you send packets to each TCP port with no flags set, the target responds with an RST packet if the port is closed. If the port is open, the host ignores the packet, and no response arrives.

Figure 5-18 NULL Scan



This is, of course, assuming that all hosts comply with RFC 793. In reality, Windows hosts do not comply with this RFC. Subsequently, you cannot use a NULL scan against a

Windows machine to determine which ports are active. When a Microsoft operating system receives a packet that has no flags set, it sends an RST packet in response, regardless of whether the port is open. With all NULL packets receiving an RST packet in response, you cannot differentiate open and closed ports.

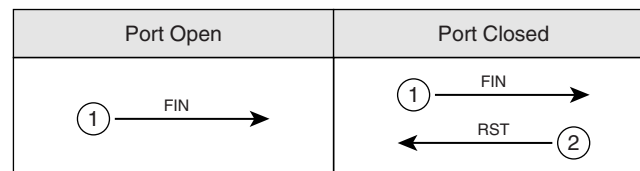
UNIX-based systems do comply with RFC 793; therefore, they send RST packets back when the port is closed and no packet when the port is open.

Note that this is the opposite effect of the SYN and TCP Connect() scans mentioned previously. In those scans, a response indicated an open port, but in a NULL scan, a response indicates a closed port. This is why a NULL scan is called an inverse scan. Inverse scans are stealthier than the TCP Connect() and SYN scans, but they are not as accurate.

FIN Scan

Another type of inverse scan is the FIN scan. Just like the NULL scan, this is stealthier than the SYN and TCP Connect() scans. In a FIN scan, a packet is sent to each TCP port with the `-FIN` bit set to on. The FIN bit indicates the ending of a TCP session. Like all inverse scans, an RST response indicates the port being closed, and no response indicates that the port is listening. Keep in mind, however, that Windows PCs do not comply with RFC 793; therefore, they do not provide accurate results with this type of scan. Figure 5-19 displays the response to a FIN scan.

Figure 5-19 *FIN Scan*



ACK Scan

In normal TCP operation, acknowledgements (ACKs) are sent after the number of packets specified in the advertised window size of the receiving host. In an ACK scan, you use the acknowledgements to discover the configuration of a firewall. If a port is filtered on a firewall, nothing comes back. If a port is unfiltered (traffic destined for that port is allowed through the firewall), however, an RST is sent back. By listening to the RST messages, you can learn which ports are filtered and unfiltered on a firewall.

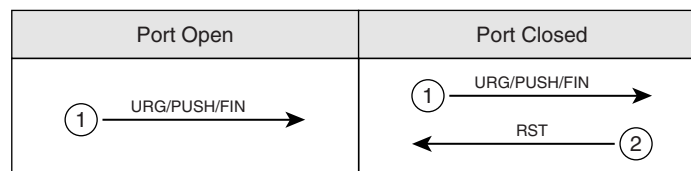
Xmas-Tree Scan

Figure 5-20 shows the formation of a packet in a Xmas-Tree scan. The Xmas-Tree scan sends a TCP packet with the following flags:

- **URG**—Indicates that the data is urgent and should be processed immediately
- **PSH**—Forces data to a buffer
- **FIN**—Used when finishing a TCP session

The trick in this scan is not the purpose of these flags, but the fact that they are used together. A TCP connection should not be made with all three of these flags set. Xmas-Tree returns the same results as other inverse scans and subsequently has the same limitations when used against Windows platforms.

Figure 5-20 *Xmas-Tree Scan*



Dumb Scan

The dumb scan (also called idle or reverse scan) was discovered by Salvatore Sanfilippo, who goes by the handle ‘antirez.’ (See the paper at <http://www.kyuzz.org/antirez/papers/dumbscan.html>.) Dumb scans are an alternative method of scanning that uses a third zombie computer to act as a “dumb” host in the process of scanning your target. A zombie host is a compromised idle host. Typically, this host does not store sensitive data, and access to it is often unnoticed. Many companies have idle hosts that are used for the transferal of data over dial-up modems. You can discover these easily by using war dialer software like ToneLoc. For example, small branch offices for credit unions might use a host for either dial-in access or for dialing in to a credit reporting company to gather financial reports on a client. If you can gain access to these hosts, you can usually gain access to the rest of their data network.

Malicious hackers often use idle systems on the Internet that they have compromised. This is why no network is safe from malicious hackers.

Just like a normal SYN scan, with a dumb scan, a SYN is sent to the target. This time, however, the zombie host sends it. If a port is listening, the target responds with the expected SYN/ACK response. If the port is closed, the target responds with an RST message. At this stage, nothing distinguishes a normal SYN scan from a dumb scan.

What makes a dumb scan different is that the scan is not sent from your computer, but from a zombie host. While the scan is launched from the zombie host, you are performing a continuous ping from Computer X against the zombie host. Looking at the ID field in the echo response from the zombie host, you can determine which ports are open and which are closed on the target system. For example, using the HPING Linux utility with the `-r` switch to see ID increments, you can see the following output when pinging a zombie host:

```
HPING B (eth0 172.16.15.12): no flags are set, 40 data bytes
60 bytes from 172.16.15.12: flags=RA seq=0 ttl=64 id=41660 win=0 time=1.2 ms
60 bytes from 172.16.15.12: flags=RA seq=1 ttl=64 id=+1 win=0 time=88 ms
60 bytes from 172.16.15.12: flags=RA seq=2 ttl=64 id=+1 win=0 time=93 ms
60 bytes from 172.16.15.12: flags=RA seq=3 ttl=64 id=+1 win=0 time=75 ms
60 bytes from 172.16.15.12: flags=RA seq=4 ttl=64 id=+1 win=0 time=93 ms
60 bytes from 172.16.15.12: flags=RA seq=5 ttl=64 id=+1 win=0 time=80 ms
```

Here, no ports are open. You start with the initial ID of 41660 and then increase by one each ping. Computer X continues its ping of the zombie host, but this time when the zombie host sends a SYN to an open port of the target, the response changes:

```
60 bytes from 172.16.15.12: flags=RA seq=1 ttl=64 id=+1 win=0 time=87 ms
60 bytes from 172.16.15.12: flags=RA seq=2 ttl=64 id=+2 win=0 time=90 ms
60 bytes from 172.16.15.12: flags=RA seq=3 ttl=64 id=+1 win=0 time=91 ms
60 bytes from 172.16.15.12: flags=RA seq=4 ttl=64 id=+1 win=0 time=92 ms
60 bytes from 172.16.15.12: flags=RA seq=5 ttl=64 id=+1 win=0 time=92 ms
```

On the second line of this output, the ID incremented by two. This indicates that whatever port is being scanned at the time of that ping is a listening port on the target.

NMap

Now that you have learned of the different scanning options, you will learn how to implement these scans using a tool called NMap.

All penetration testers have a toolbox of software applications frequently used in testing. Included in every penetration tester toolbox should be NMap. NMap, written by Fyodor and available at <http://www.insecure.org>, is available on both Windows and Linux platforms. Although the Windows version of NMap might be easier to use because of its graphical user interface, this chapter uses the Linux version for explanatory purposes. At press time, the Windows version did not yield as accurate results as its Linux counterpart.

NOTE

NMap, although the most popular, is not the only port scanner available. Other port scanners include Superscan, Scanline, VScan, and Angry IP. See Appendix B, “Tools,” for information on these and other port scanners.

In the man (manual) page, NMap is described as a tool to “allow system administrators and curious individuals to scan large networks <determine> which hosts are up and what services they are offering.” (To view more of the man page, type **man NMap** at the Linux command line.) NMap allows you to perform many of the scans previously covered.

NMap Switches and Techniques

The predominant switches available in NMap as they correspond to the scans covered earlier are as follows::

- **-sT**—TCP Connect() scan
- **-sS**—SYN scan
- **-sF**—FIN scan
- **-sX**—Xmas-Tree scan
- **-sN**—NULL scan
- **-sI**—Dumb scan (also called an idle scan)
- **-sA**—ACK scan

In addition, other parameters are helpful:

- **-P0**—Do not try to ping hosts before scanning them.
- **-PP**—Uses the ICMP timestamp request (ICMP type 13) packet to find listening hosts. Normally, NMap attempts to ping the hosts using ICMP echo request (ICMP type 1) packets to see if the host is there. Some firewalls and routers block echo requests yet still allow other traffic to penetrate. This switch also uses ICMP to determine if the host is live, but it uses a different ICMP packet for this purpose.
- **-6**—Enables IPv6 support. You can perform a port scan against a host name through DNS (assuming the DNS server has the IPv6 AAAA records) or through the IP address.
- **-oN *logfile***—Sends the output in human-readable format to the file of your choosing.
- **-oX *logfile***—Same as **-oN**, but this time send it to the logfile in XML format.
- **-oG *logfile***—Same as **-oN**, but stores all the results on a single line for querying through the Grep program.
- **--append_output**—Appends the output to your existing log files instead of overwriting them.

- **-p**— Specifies the port number(s) to scan. TCP and UDP ports total 65,536. This switch lets you specify single ports, ranges, or lists of ports to scan. You can also specify whether you want to ping UDP or TCP ports only. For example, to scan TCP ports 23 (Telnet), 25 (SMTP), and 80 (HTTP), you can type this:

```
NMap -p T:23,25,80
```

- **-v**— Verbose mode.
- **-vv**— Very verbose mode. Enable this to see the most detailed output.
- **-M *max sockets***— Sets the maximum number of sockets used by NMap. Limiting this value decreases the scan rate, which is helpful when scanning some hosts that have been known to crash when being scanned. Of course, discovering that these hosts crash is a vulnerability that you should document in your penetration report.
- **-T {*paranoid* | *sneaky* | *polite* | *normal* | *aggressive* | *insane*}**— Changes the timing policies for scanning. The default is *normal*, which attempts to scan as quickly as possible. ***paranoid*** is helpful to avoid IDS systems and waits five minutes between sending packets. ***sneaky*** sends packets every 15 seconds. ***polite*** waits every 0.4 seconds and is designed to prevent host crashing. ***aggressive*** and ***insane*** attempt to speed up the scans, but because accuracy and stealth are important, you should avoid these unless you have a justifiable reason to use them.
- **--host_timeout *milliseconds***— Specifies how long to wait for a response before scanning stops for a single host. If NMap appears to hang, you might want to adjust this timer.
- **--scan_delay *milliseconds***— Similar to **-T**, this specifies how long to wait between probes. Increasing this value might let you go undetected past IDS systems.
- **-O**— Attempts to detect the operating system. It also attempts TCP Sequence Predictability Classification to report how difficult it would be to forge a TCP connection against your target. Beware that NMap is not always accurate in detecting the operating system.

In addition to the switches just listed, NMap is capable of performing more advanced techniques, such as changing the source port number, fragmenting packets, performing Identd scanning, and doing FTP bounce scanning:

- **--source_port *port number***— Specify the port number. Firewalls and routers might block your attempts to scan a host if your port number is above 1023. However, many firewalls and routers allow DNS (port 53) or FTP-Data (port 21) packets through. If you are having difficulties getting past a firewall, try changing your port number to 53 or 21.

- **-f**—Fragment your packets. By breaking up your scans into smaller TCP fragments, you can often go undetected by low-end security devices that do not want to process fragments to see if a scan is taking place.
- **-I**—Perform an Identd scan. The Identd protocol (RFC 1413) allows for the disclosure of the username associated with a TCP process. This allows you to connect to web servers and find out if it is running with root privileges (full administrator access). If so, cracking the web server enables you full rights to the server that is hosting the site. This scan rarely works, however, because most hosts disable the Identd service for this very reason.
- **-b**—Perform an FTP bounce scan. This is an older scan that, like the Identd scan, rarely works. It relies on your having access to a proxy FTP server and performing a scan from that FTP server. Again, most administrators have taken necessary precautions to prevent against such scans.

Compiling and Testing NMap

Compiling NMap is similar to compiling other programs in Linux. Follow these steps:

Step 1 Download the latest version from <http://www.insecure.org>.

Step 2 Unzip NMap using the `gzip` program.

Step 3 Untar NMap using the `tar` program.

Step 4 Navigate to the directory containing the NMap files and type `./configure`.

Step 5 Type `make install`.

Step 6 Type `./install`.

Next, perform a TCP Connect() scan against the IP address 64.202.167.192. At the command line, type the following:

```
NMap -sT -vv -p T:1-1023 -P0 -O 64.202.167.192
```

This performs a TCP Connect() scan with very verbose output. You are scanning TCP ports 1 through 1023 and not pinging the host first to see if it is active. Finally, you have enabled the `-O` switch to attempt to determine the operating system.

Based on the results, you now know that TCP ports 80 and 443 are available. This tells you that this particular server is running as a web server. NMap is unable to determine the type of operating system, however. Still, if it found ports 137, 138, and 139 open, it would know that the target was most likely running a Windows operating system, because these ports are used with NetBIOS (a service commonly seen on Windows systems). NMap knows more than 500 different operating systems and can detect the operating systems not just of servers, but network devices like routers, firewalls, and others.

Fingerprinting

Determining the operating system of your target is important because many of the exploits are specific to the platform. The process of discovering the underlying operating system is called *fingerprinting*. Besides using the built-in fingerprinting features of NMap, you can try other techniques such as Telnet or HTTP to get requests.

For example, you would know that your target was running HP-UX if you Telneted to a device and got this response:

```
Trying 10.0.0.1...
Connected to server.hackmynetwork.com
Escape character is '^]'.
```

```
HP-UX B.10.01 A 9000/715 (ttyp3)
```

```
login:
```

Because most networks do not allow Telnet access, you might have to try to Telnet to another port, such as TCP port 21 (FTP). You would know your target was running the Sun operating system if you received the following response:

```
#telnet 10.0.0.1 21
220 ftp FTP server (UNIX(r)System V Release 4.0) ready.
SYST
215 UNIX Type: L8 Version: SUNOS
```

You can also try to perform an HTTP get request. Here is the output you might receive if your target is running Microsoft IIS:

```
#echo 'GET / HTTP/1.0\n' | grep '^Server'
Server: Microsoft-IIS/5.0
```

Another means of detecting the operating system of the target system is through stack fingerprinting. Stack fingerprinting actively sends packets to the target TCP/IP stack and analyzes the response. TCP/IP stacks differ from vendor to vendor, making this a prime means of detecting an operating system. You can do stack fingerprinting through the following methods:

- **BOGUS probe**—This technique detects older Linux systems. It sets bits 7 and 8 of the TCP header in a SYN packet. Linux systems prior to the 2.0.35 kernel respond with the same bits set. These bits were originally undefined, but now they are used to declare a device as being explicit congestion notification (ECN) capable. Routers utilizing random early detection (RED) can set the congestion experienced (CE) bit on packets to notify end stations that congestion occurred.
- **TCP ISN sampling**—This technique finds patterns in the initial sequence numbers used in connection requests. Some UNIX systems use 64000 as the sequence number. Newer versions of Solaris and FreeBSD, however, employ random increments. In comparison, Windows computers are incremented by a small fixed amount each time. Finally, some devices always start with the same ISN. 3Com hubs, for example, start with 0x803, and Apple printers start with 0xC7001.

- **TCP initial window size**—This technique examines the window size on return packets. AIX sends a window size of 16,165; Microsoft, OpenBSD, and FreeBSD use 16,430; Linux uses 32,120.
- **RTO delay**—Sometimes called temporal response analysis, this is a more complicated technique because it requires the addition of a firewall device. A firewall is configured to deny incoming TCP packets with the SYN and ACK flags set. You send a SYN, but when the target responds with SYN/ACK, it is blocked. You then listen to the delay between transmissions (retransmission time-out) and compare the results with a signature database. A patch to NMAP called NMap-ringv2 uses this technique. Ringv2 has a similar technique that measures the RTO of FIN packets.
- **IP ID sampling**—Every system uses an ID field in the IP header when data needs to be fragmented across multiple packets. Most increment a value by one, but some do not, giving you the opportunity to detect those operating systems that are an exception to the rule. OpenBSD, for example, uses a random IP. Microsoft has its own style; it increments by 256 each time.
- **MSS response**—You can examine the maximum segment size (MSS) response to determine whether your target is running the Linux operating system. If you send a packet with a small MSS value to a Linux box, it echoes that MSS value back to you in its response. Other operating systems give you different values.

You can use several different tools for OS fingerprinting. You have already learned about NMap and the patch to NMap, Ringv2. Other tools include the following:

- Xprobe2
- Ettercap
- p0f v2
- Queso
- SS
- CheckOS

NOTE Xprobe2 is a unique tool in that it uses fuzzy matching. It still maintains a fingerprint database of well-known signatures, but it also includes a probabilistic score to guess the operating system.

Footprinting

The methods described in this chapter are called footprinting a target network. Be careful not to get this confused with fingerprinting. *Fingerprinting* is the process of determining

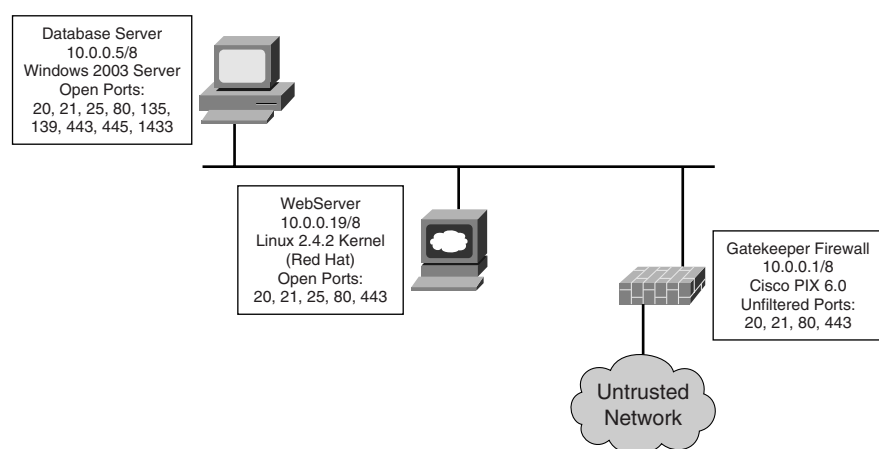
the operating system on a device, whereas *footprinting* is the combination of active and passive reconnaissance techniques for the purposes of establishing a strategy of attack.

After you finish footprinting (gathering all the information that is relevant to your target), you can draw out a network map. The network map should contain the following:

- Host names
- IP addresses
- Listening port numbers
- Operating systems

Figure 5-21 shows an example of a network map.

Figure 5-21 *Sample Network Map*



Assume that you have detected three servers and a firewall. The servers are running Microsoft Windows, either 2000 or 2003 edition. You have discovered that the servers are running IIS and have a SQL database. Although you do not know for certain what type of database application is running, the probability of it running Microsoft SQL server is high because that is the most preferred database system on Windows.

Armed with this valuable information, you can begin to strategize as to what type of attacks to launch against the target network. The attacks will center on the vulnerabilities found in the Windows operating systems and applications. You can also try generic firewall attacks. These types of attacks are covered in subsequent chapters.

All the techniques mentioned so far, although not necessarily intrusive to a company network, can lead to dangerous consequences. Therefore, a company should make every effort to mitigate the risks associated with reconnaissance attacks.

Detecting a Scan

Host and network scanning cannot go unnoticed because they are usually just a symptom of other possible exploits and attacks to come. This section covers the use of a Cisco Intrusion Detection System IDS-4215 sensor to monitor and detect a network that has been scanned with NMap. As explained earlier, NMap is a tool of choice for penetration testers when it comes to port scanning. This is primarily because of its extreme flexibility and versatility of the types of scans it can perform.

Building up a defense barricade to protect against NMap scans involves several components. Before this chapter delves into scan detection, you need to examine these necessary security components, as discussed in the sections that follow.

Intrusion Detection

IDSs are similar to home security systems (burglar alarms) that monitor entry or breach into your home or office. Like the home security systems, IDSs log an alarm entry into the network. Unlike most home systems, however, you can configure an IDS to actually fight back with TCP RSTs and SHUN commands in the efforts to stop further entry or damage to the network. Location is critical with these systems, just like a standard security camera is to a security guard. That is why most IDSs are located where they can see as much traffic as possible.

Anomaly Detection Systems

Anomaly detection systems (also called profile-based detection systems) are designed to watch user or network profiles. For example, an anomaly detection system alarms if it notices a network that normally is at 30 percent utilization peak up to 90 percent for a long period.

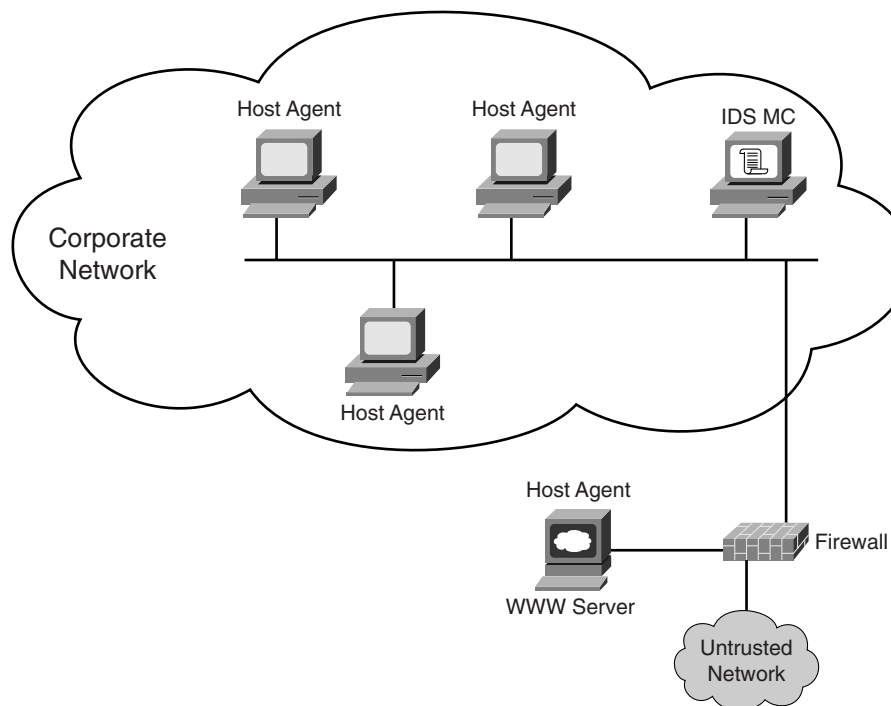
Misuse Detection System

Misuse detection uses pattern matching. These systems contain a database of hundreds of patterns and signatures that are used to match with traffic on a network cable. You can compare misuse detection to standard disk antivirus software, where the antivirus software scans your hard drive looking for patterns in programs and files that represent malicious alterations. Misuse detection reads frames and packets off a cable instead of a hard drive. These are the most commonly used detection systems today. However, they can quickly become out of date as new attacks emerge that are not within the signature database.

Host-Based IDSs

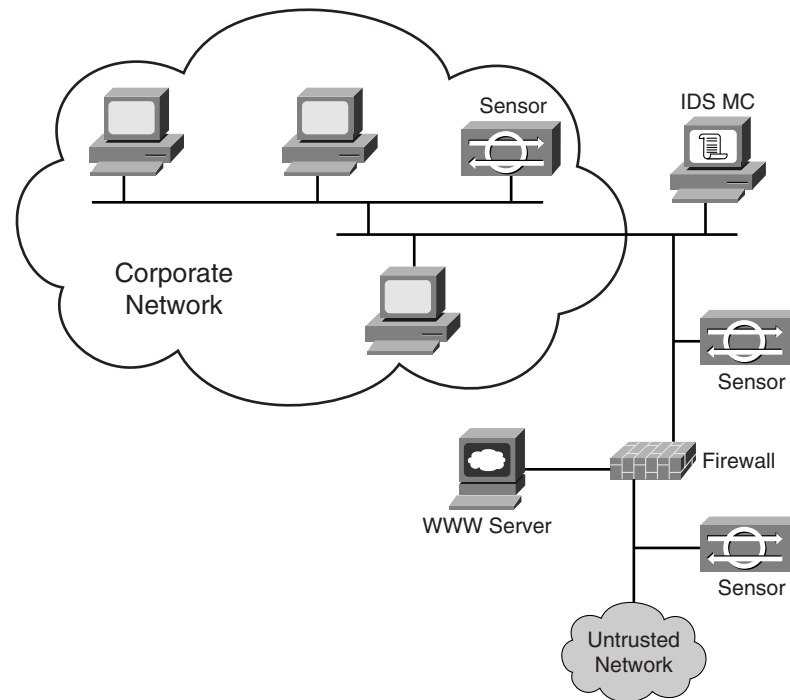
Host-based IDSs are installed locally on a host computer and are used to check that local system only for items such as system calls, audit logs, error messages, and network traffic. The benefit of host-based IDS systems is the protection and warning they can provide to a specific system. However, they are not designed to protect the entire campus network; only the specific host is protected. Figure 5-22 illustrates how to deploy a host-based IDS.

Figure 5-22 *Host-Based IDS Deployment*



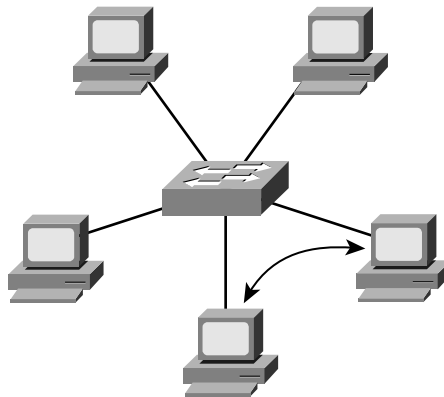
Network-Based IDSs

Network-based IDSs such as the Cisco 4200 series appliances are dedicated to one task—monitoring the entire network. They are located at check points or special ports where they can monitor network traffic that is directed to any host. Figure 5-23 illustrates how to deploy a network-based IDS.

Figure 5-23 *Network-Based IDS Deployment*

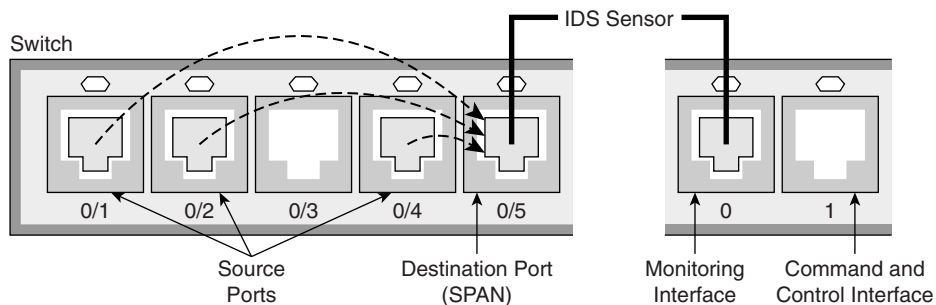
Network Switches

Switches appeared shortly after the network hubs came into the scene. They provide the same star topology as hubs; however, they do not interconnect all computers to the same bus. When computers communicate, switches are designed to monitor the Layer 2 frames and develop a MAC address table. This increases switch performance by creating an internal map of computers to specific interface ports. Now when computers need to communicate across the switch, their frames are forwarded only to the specific interface that contains the destination host, as Figure 5-24 illustrates.

Figure 5-24 *Switched Network*

Because of this basic design where traffic is forwarded only where it is needed, lower-cost switches are difficult to effectively locate an IDS on. Higher-cost programmable switches typically support what is known as switched port analyzer (SPAN) ports or port monitoring. (These terms are used interchangeably.) SPAN functionality allows the network administrator to select the specific ports to which they want to forward copies of all traffic. These ports in turn are where the IDS is connected

As Figure 5-25 illustrates, the switch on the right is configured for SPAN. Traffic going into ports 0/1 through 0/4 is copied to the destination port of 0/5. Port 0/5 is subsequently connected to the monitoring interface (port 0) on the IDS sensor.

Figure 5-25 *SPAN Port in Use*

Examples of Scan Detection

The sections that follow take you through some specific examples of detecting port scans that are executed with NMap. The examples use a basic install of the Cisco 4215 IDS

Sensor attached to the network with IDS Event Viewer (IEV) software to monitor sensor alarms in real-time.

Detecting a TCP Connect() Scan

NMap TCP Connect() scan, as mentioned earlier, is a reliable port scanning technique that determines the status of open or closed ports. IDS sensors, however, are keen on detecting normal TCP connections that do not actually send data and sound off an alarm. Example 5-4 shows the syntax used and the output returned in scanning a Windows 2003 Server.

Example 5-4 *Using NMap TCP Connect Scan on a Windows 2003 Server*

```
C:\>NMap -sT -vv -P0 192.168.200.100

Starting NMap 3.81 ( http://www.insecure.org/NMap ) at 2005-03-21 19:19 GMT
Standard Time
Initiating Connect() Scan against WEB1 (192.168.200.100) [1663 ports] at 19:19
Discovered open port 53/tcp on 192.168.200.100
Discovered open port 23/tcp on 192.168.200.100
Discovered open port 1433/tcp on 192.168.200.100
Discovered open port 1026/tcp on 192.168.200.100
Discovered open port 1031/tcp on 192.168.200.100
Discovered open port 1025/tcp on 192.168.200.100
Discovered open port 139/tcp on 192.168.200.100
Discovered open port 1434/tcp on 192.168.200.100
Discovered open port 445/tcp on 192.168.200.100
Discovered open port 135/tcp on 192.168.200.100
Discovered open port 1029/tcp on 192.168.200.100
The Connect() Scan took 52.38s to scan 1663 total ports.
Host WEB1 (192.168.200.100) appears to be up ... good.
Interesting ports on WEB1 (192.168.200.100):
(The 1652 ports scanned but not shown below are in state: filtered)
PORT      STATE SERVICE
23/tcp    open  telnet
53/tcp    open  domain
135/tcp   open  msrpc
139/tcp   open  netbios-ssn
445/tcp   open  microsoft-ds
1025/tcp  open  NFS-or-IIS
1026/tcp  open  LSA-or-nterm
1029/tcp  open  ms-lsa
1031/tcp  open  iad2
1433/tcp  open  ms-sql-s
1434/tcp  open  ms-sql-m
```

Now that you have scanned successfully, look at the Cisco IEV real-time output in Figure 5-26. As you can see, the sensor accurately detected the scan.

Figure 5-26 *TCP Connect() Scan Detected*

Signature Name	Sig ID	Severity Level	Device Name	Dst Address	Src Address
TCP SYN Port Sweep	3002	Low	DAWN-IDS	192.168.200.100	192.168.200.13
TCP SYN Port Sweep	3002	Low	DAWN-IDS	192.168.200.100	192.168.200.13
TCP SYN Port Sweep	3002	Low	DAWN-IDS	192.168.200.100	192.168.200.13
TCP SYN Port Sweep	3002	Low	DAWN-IDS	192.168.200.100	192.168.200.13
TCP SYN Port Sweep	3002	Low	DAWN-IDS	192.168.200.100	192.168.200.13

Below the table are three buttons: Pause, Resume, and Reconnect.

Detecting a SYN Scan

SYN scans are a little more difficult to detect because they are just trying to leave a connection open and relying on the timeout to clear the connections. Example 5-5 displays the syntax used and output generated when scanning the same Windows 2003 Server.

Example 5-5 *SYN Scan on a Windows 2003 Server*

```
C:\>NMap -sS -vv -P0 192.168.200.100

Starting NMap 3.81 ( http://www.insecure.org/NMap ) at 2005-03-21 19:22 GMT
Standard Time
Initiating SYN Stealth Scan against WEB1 (192.168.200.100) [1663 ports] at 19:22

Discovered open port 23/tcp on 192.168.200.100
Discovered open port 53/tcp on 192.168.200.100
Discovered open port 445/tcp on 192.168.200.100
Discovered open port 1031/tcp on 192.168.200.100
Discovered open port 1025/tcp on 192.168.200.100
Discovered open port 1433/tcp on 192.168.200.100
Discovered open port 139/tcp on 192.168.200.100
Discovered open port 1026/tcp on 192.168.200.100
Discovered open port 135/tcp on 192.168.200.100
Discovered open port 1434/tcp on 192.168.200.100
Discovered open port 1029/tcp on 192.168.200.100
The SYN Stealth Scan took 0.11s to scan 1663 total ports.
Host WEB1 (192.168.200.100) appears to be up ... good.
Interesting ports on WEB1 (192.168.200.100):
(The 1652 ports scanned but not shown below are in state: closed)
PORT      STATE SERVICE
23/tcp    open  telnet
53/tcp    open  domain
135/tcp   open  msrpc
139/tcp   open  netbios-ssn
445/tcp   open  microsoft-ds
1025/tcp  open  NFS-or-IIS
1026/tcp  open  LSA-or-nterm
1029/tcp  open  ms-lsa
```

Example 5-5 SYN Scan on a Windows 2003 Server (Continued)

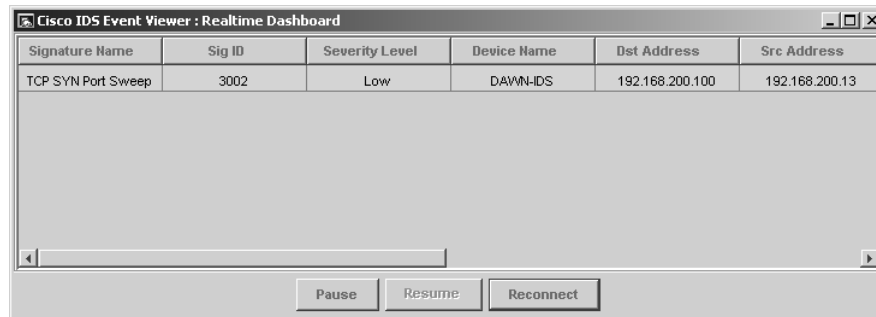
```

1031/tcp open  iad2
1433/tcp open  ms-sql-s
1434/tcp open  ms-sql-m
MAC Address: 00:50:56:EE:EE:EE

NMap finished: 1 IP address (1 host up) scanned in 0.344 seconds
Raw packets sent: 1663 (66.5KB) | Rcvd: 1663 (76.5KB)
    
```

As stated earlier, SYN scans leave the connection open. This is an expected anomaly that takes place between two computers if one goes down or just never returns the last ACK. SYN scans are harder for sensors to typically detect because of their natural occurrence “in the wild”; however, should you flood the network with them, it will trigger an alarm, as seen in Figure 5-27. Notice that the alarm signature is the same as an **-sT** connection scan. However, only 1 alarm was detected as opposed to 6 to 8 alarms triggered in a normal **-sT** scan. This proves that **-sS** scans are less detected.

Figure 5-27 SYN Scan Detected



Detecting FIN, NULL, and Xmas-Tree Scans

Now that you have seen two basic scans in action—TCP Connect() and SYN scans—this section investigates the three inverse scans. These scans usually result in poor port scanning reliability against Windows computers because they always return an RST state in response. This shows all ports closed, even if they are really open. Example 5-6 shows the scan syntax and output against the Windows 2003 Server. As you can see, all ports scanned are returning closed.

Example 5-6 Conducting FIN, NULL, and Xmas-Tree Scans

```

C:\>NMap -sF -vv -P0 192.168.200.100

Starting NMap 3.81 ( http://www.insecure.org/NMap ) at 2005-03-21 19:26 GMT
Standard Time
Initiating FIN Scan against WEB1 (192.168.200.100) [1663 ports] at 19:26
    
```

continues

Example 5-6 *Conducting FIN, NULL, and Xmas-Tree Scans (Continued)*

```

The FIN Scan took 0.09s to scan 1663 total ports.
Host WEB1 (192.168.200.100) appears to be up ... good.
All 1663 scanned ports on WEB1 (192.168.200.100) are: closed
MAC Address: 00:50:56:EE:EE:EE

NMap finished: 1 IP address (1 host up) scanned in 0.312 seconds
                Raw packets sent: 1663 (66.5KB) | Rcvd: 1663 (76.5KB)

C:\>NMap -sN -vv -P0 192.168.200.100

Starting NMap 3.81 ( http://www.insecure.org/NMap ) at 2005-03-21 19:24 GMT Stan
dard Time
Initiating NULL Scan against WEB1 (192.168.200.100) [1663 ports] at 19:24
The NULL Scan took 0.08s to scan 1663 total ports.
Host WEB1 (192.168.200.100) appears to be up ... good.
All 1663 scanned ports on WEB1 (192.168.200.100) are: closed
MAC Address: 00:50:56:EE:EE:EE

NMap finished: 1 IP address (1 host up) scanned in 0.312 seconds
                Raw packets sent: 1663 (66.5KB) | Rcvd: 1663 (76.5KB)

C:\>NMap -sX -vv -P0 192.168.200.100

Starting NMap 3.81 ( http://www.insecure.org/NMap ) at 2005-03-21 19:27 GMT Stan
dard Time
Initiating XMAS Scan against WEB1 (192.168.200.100) [1663 ports] at 19:27
The XMAS Scan took 0.08s to scan 1663 total ports.
Host WEB1 (192.168.200.100) appears to be up ... good.
All 1663 scanned ports on WEB1 (192.168.200.100) are: closed
MAC Address: 00:50:56:EE:EE:EE

NMap finished: 1 IP address (1 host up) scanned in 0.312 seconds
                Raw packets sent: 1663 (66.5KB) | Rcvd: 1663 (76.5KB)

```

However, the sensor detects inverse scans quite well and even displays the actual scan being executed. Figure 5-28 shows the real-time alarms detecting FIN scans, NULL packets, and an OOB error that is generated as a side effect of the Xmas-Tree scan.

Figure 5-28 *Inverse Scans Detected*

Signature Name	Sig ID	Severity Level	Device Name	Dst Address	Src Address
TCP FIN Packet	3042	High	DAWN-IDS	192.168.200.100	192.168.200.13
Netbios OOB Data	3300	High	DAWN-IDS	192.168.200.100	192.168.200.13
TCP NULL Packet	3040	High	DAWN-IDS	192.168.200.100	192.168.200.13
TCP FIN Packet	3042	High	DAWN-IDS	192.168.200.100	192.168.200.13
TCP Null Port Sweep	3015	High	DAWN-IDS	192.168.200.100	192.168.200.13
TCP FIN High Port Sweep	3011	High	DAWN-IDS	192.168.200.100	192.168.200.13
TCP FIN Port Sweep	3005	High	DAWN-IDS	192.168.200.100	192.168.200.13

Buttons: Pause, Resume, Reconnect

Detecting OS Guessing

The last detection to perform is operating system detection. NMap uses the `-O` switch to signal operating system detection against a target. Example 5-7 shows the scan syntax and output used against the Windows 2003 Server.

Example 5-7 Scanning to Determine the Target Operating System

```
C:\>NMap -O -vv -P0 192.168.200.100

Starting NMap 3.81 ( http://www.insecure.org/NMap ) at 2005-03-21 19:28 GMT Standard Time
Initiating SYN Stealth Scan against WEB1 (192.168.200.100) [1663 ports] at 19:28

Discovered open port 23/tcp on 192.168.200.100
Discovered open port 53/tcp on 192.168.200.100
Discovered open port 1434/tcp on 192.168.200.100
Discovered open port 139/tcp on 192.168.200.100
Discovered open port 1031/tcp on 192.168.200.100
Discovered open port 445/tcp on 192.168.200.100
Discovered open port 1029/tcp on 192.168.200.100
Discovered open port 1025/tcp on 192.168.200.100
Discovered open port 1026/tcp on 192.168.200.100
Discovered open port 1433/tcp on 192.168.200.100
Discovered open port 135/tcp on 192.168.200.100
The SYN Stealth Scan took 0.09s to scan 1663 total ports.
For OSScan assuming port 23 is open, 1 is closed, and neither are firewalled
Host WEB1 (192.168.200.100) appears to be up ... good.
Interesting ports on WEB1 (192.168.200.100):
(The 1652 ports scanned but not shown below are in state: closed)
PORT      STATE SERVICE
23/tcp    open  telnet
53/tcp    open  domain
135/tcp   open  msrpc
139/tcp   open  netbios-ssn
445/tcp   open  microsoft-ds
1025/tcp  open  NFS-or-IIS
1026/tcp  open  LSA-or-nterm
1029/tcp  open  ms-lsa
1031/tcp  open  iad2
1433/tcp  open  ms-sql-s
1434/tcp  open  ms-sql-m
MAC Address: 00:50:56:EE:EE:EE
Device type: general purpose
Running: Microsoft Windows 2003/.NET|NT|2K/XP
OS details: Microsoft Windows 2003 Server or XP SP2P
OS Fingerprint:
TSeq(Class=TR%IPID=I%TS=0)
T1(Resp=Y%DF=Y%W=402E%ACK=S++%Flags=AS%Ops=MNWNNT)
T2(Resp=Y%DF=N%W=0%ACK=S%Flags=AR%Ops=)
T3(Resp=Y%DF=Y%W=402E%ACK=S++%Flags=AS%Ops=MNWNNT)
T4(Resp=Y%DF=N%W=0%ACK=0%Flags=R%Ops=)
T5(Resp=Y%DF=N%W=0%ACK=S++%Flags=AR%Ops=)
T6(Resp=Y%DF=N%W=0%ACK=0%Flags=R%Ops=)
```

continues

Example 5-7 *Scanning to Determine the Target Operating System (Continued)*

```

T7(Resp=Y%DF=N%W=0%ACK=S+++Flags=AR%Ops=)
PU(Resp=Y%DF=N%TOS=0%IPLEN=B0%RIPTL=148%RIPCK=E%UCK=E%ULEN=134%DAT=E)

TCP Sequence Prediction: Class=truly random
                        Difficulty=9999999 (Good luck!)
TCP ISN Seq. Numbers: 69D80142 413B414C 4E54B424 74F4775C 1DE05ABB AC9A1054
IPID Sequence Generation: Incremental

NMap finished: 1 IP address (1 host up) scanned in 1.062 seconds
                Raw packets sent: 1676 (67.3KB) | Rcvd: 1677 (77.4KB)

```

The interesting thing about this scan is that it did not succeed in guessing the exact operating system. However, it does narrow it down to just Windows 2003 Server or XP with SP2. By using a little deductive reasoning and looking at the ports that are open, such as TCP 23, which is used for a Telnet server, you would lean more toward the Windows 2003 server rather than XP. Looking at the error generated in Figure 5-29, you can see Cisco IDS detect the OS guessing with an error called NMap fingerprinting. Yes, this scan is easily detectable.

Figure 5-29 *OS Guessing Detected*

Signature Name	Sig ID	Severity Level	Device Name	Dst Address	Src Address
NmapFingerprint	3046	Medium	DAWN-IDS	192.168.200.100	192.168.200.13
TCP SYN Port Sweep	3002	Low	DAWN-IDS	192.168.200.100	192.168.200.13

Case Study

This case study chains together several of the items learned within the chapter to perform a successful scan of a network. This case study trails Evil Jimmy the Hacker as he scans a small company called Little Company Network (LCN). He uses DNS to gather information before moving onto NMap for some scanning as he attempts to start his diagramming of the network.

The scene is set as LCN rejects Evil Jimmy for a position. He is skilled in penetration testing, and because LCN obviously did not even read to the end of his r sum , Jimmy plans to make use of his skills in an unauthorized manner. Jimmy knows the DNS names of his target LCN.com, so he plugs his laptop into the wall and begins his attack. Knowing that preparation is vital to a successful outcome, Jimmy starts by making a plan and gathering his tools. The following steps illustrate the execution.

- Step 1** Evil Jimmy heads straight for the company website and uses the Wget tool to download the entire website. He can later browse this information at his leisure to look for e-mail addresses, address information, and any other details about the company that might later prove useful.
- Step 2** Evil Jimmy uses SamSpade to discover the company address, contact, and registration information posted for the website at the time it was created. The following example displays these output details from SamSpade.

```
Registrant:
LITTLE COMPANY NETWORK
  100 NW JOHN OLSEN PLACE
  HILLSBORO, OR 97123
  US
Domain Name: LCN.COM
Administrative Contact, Technical Contact:
  Little Company Network  jbates@LCN.COM
  100 NW JOHN OLSEN PL
  HILLSBORO, OR 97123
  US
  503-123-5555 fax: - 503-123-5555

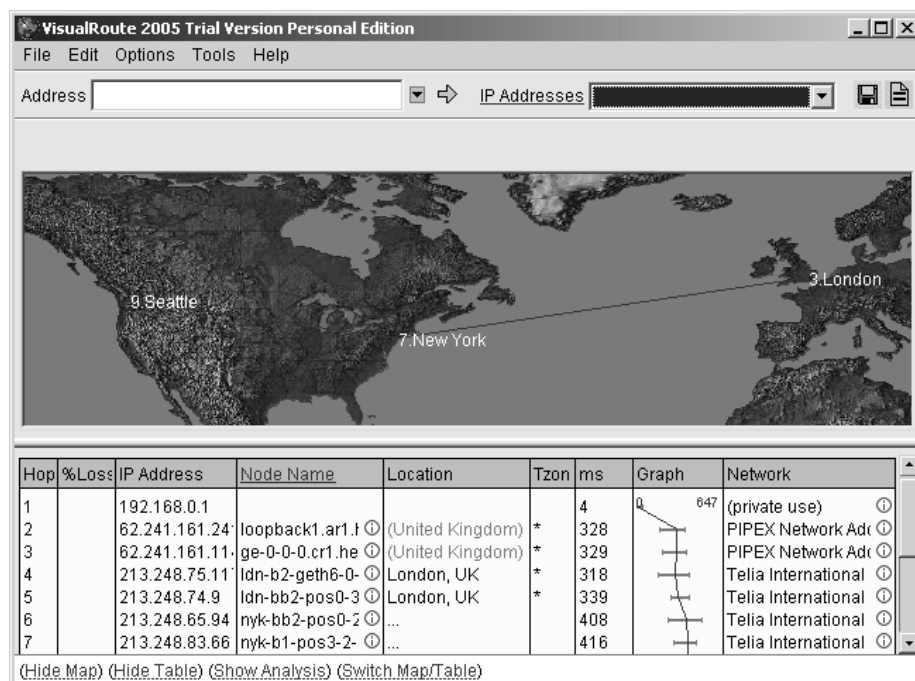
Record expires on 11-Apr-2005.
Record created on 10-Apr-1997.
Database last updated on 20-Mar-2005 17:16:56 EST.

Domain servers in listed order:

  NS1.SECURESERVERS.NET
  NS2.SECURESERVERS.NET
```

- Step 3** Using his Visual Route tool, Jimmy gets a general idea of where the web server is. As Figure 5-30 shows, the web server is in Seattle, Washington, so the address in Oregon is probably the office address with the web server being hosted elsewhere in Washington..

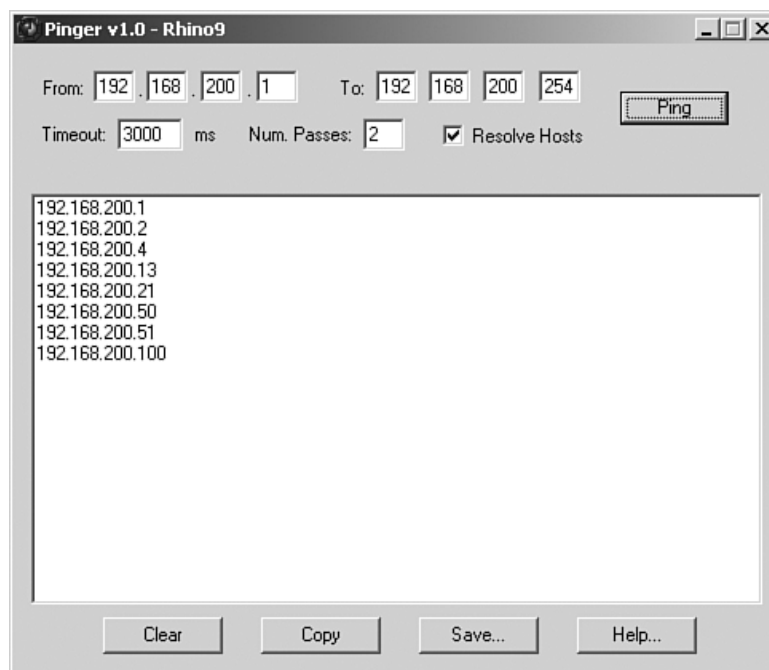
Figure 5-30 Visual Route Results



Step 4 Armed with company address information, Evil Jimmy drives right over to the company office and plugs into the network to do a little scanning. (In the real world, this might or might not take place, but for the example, it works great.)

NOTE Wireless access is becoming increasingly viable as a way into a company network without ever needing to physically “touch” their network.

Step 5 Now that Jimmy has local network access, he can ping sweep the network. Using Pinger, Jimmy discovers several computers across the network. Figure 5-31 displays the computers on the network that respond to standard ICMP requests.

Figure 5-31 *Pinger Results*

Step 6 Next, Jimmy begins port scanning computers to help enumerate details of which programs are running on each computer. Also, Jimmy uses the NMap **-O** switch to detect which operation system is running. The following example shows the output information:

```
C:\>NMap -sS -O 192.168.200.21,100
```

```
Interesting ports on Desk1 (192.168.200.21):
(The 1658 ports scanned but not shown below are in state: closed)
PORT      STATE SERVICE
21/tcp    open  ftp
25/tcp    open  smtp
135/tcp   open  msrpc
139/tcp   open  netbios-ssn
5713/tcp  open  proshareaudio
MAC Address: 08:00:46:F3:14:72
Device type: general purpose
Running: Microsoft Windows NT/2K/XP
OS details: Microsoft Windows XP SP2
```

```
NMap finished: 2 IP addresses (2 hosts up) scanned in 3.203 seconds

Starting NMap 3.81 ( http://www.insecure.org/NMap ) at 2005-03-21 21:07
GMT
  Standard Time
Interesting ports on WEB1 (192.168.200.100):
(The 1652 ports scanned but not shown below are in state: closed)
PORT      STATE SERVICE
23/tcp    open  telnet
53/tcp    open  domain
135/tcp   open  msrpc
139/tcp   open  netbios-ssn
445/tcp   open  microsoft-ds
1025/tcp  open  NFS-or-IIS
1026/tcp  open  LSA-or-nterm
1029/tcp  open  ms-lsa
1031/tcp  open  iad2
1433/tcp  open  ms-sql-s
1434/tcp  open  ms-sql-m
MAC Address: 00:50:56:EE:EE:EE
Device type: general purpose
Running: Microsoft Windows 2003/.NET!NT/2K/XP
OS details: Microsoft Windows 2003 Server or XP SP2
```

Step 7 Jimmy is finished scanning and leaves the building just as the networking team commences the search for the intruder. Fortunately for Jimmy, it took several minutes for the team to detect the scan before they could start searching for the guilty hacker.

Step 8 Back in the comfort of his home, Evil Jimmy starts to collate the information into an easy-to-read diagram that displays computer addresses, services open, and operating systems on each.

As you can see, collecting information about a company and its network is easy, fun, and relatively quick.

Summary

Reconnaissance can be split into two categories; passive, which can be likened to a burglar glancing at houses as he walks along the road; and active, where he walks right up and peers in your windows.

Passive reconnaissance can be time intensive and yield varying degrees of success. The most obvious starting point is the website of your target. Two popular tools are available to help grab the whole site for offline browsing:

- Wget (command-line tool)
- Teleport Pro (graphical tool)

Analyzing site content can reveal information such as the following:

- Hardware, operating system, and application information from commented code
- Contact information for use in social engineering attacks

You can also glean potentially useful information from public sources, including these:

- EDGAR filings
- USENET newsgroups
- User group meetings
- Business partners

Active reconnaissance can be far more revealing, but the downside is that it is a riskier process and is more easily detected.

The first step in active reconnaissance is to identify hosts within the target network. You can use the following tools to accomplish this:

- NSLookup
- Whois
- SamSpade
- Visual Route

Simply performing an NSLookup to search for an IP address is passive, but the moment you begin doing a zone transfer using some of these tools, you are beginning to do active reconnaissance.

After the hosts have been identified, you can use port scanning to identify potential vulnerabilities. A range of different port scan techniques is available:

- TCP Connect() scan
- SYN scan
- FIN scan
- Xmas-Tree scan
- NULL scan
- Dumb scan

In addition, this chapter examined NMap, a popular and powerful tool that carries out port scanning.

This chapter looked at fingerprinting—the process of examining the characteristics of the host to identify its underlying operating system. Although this chapter discussed NMap, other fingerprinting tools are available:

- Xprobe2
- Ettercap
- p0f v2
- Queso
- SS
- CheckOS

All these steps constitute the footprinting of a target network. After the footprint is complete, you should be able to create a network map containing information such as the following:

- Host names
- IP addresses
- Listening port numbers
- Operating systems

Reconnaissance against a target network, such as that described in this chapter, can be detected using an IDS, which can take various forms:

- Anomaly detection
- Misuse detection
- Host-based detection
- Network-based detection