CHAPTER 3 [:. SCANNING .:]

This chapter covers all facets of Scanning, the second phase of a hacking process where hackers will be using aggressive network probing techniques and tools to uncover further information about the target organization. Since scanning is such a big topic, the chapter will start with basic introduction of scanning so that you can have a feel and good grip what it is all about. Then, you will delve further into the topic to examine which scanning objectives and types of information that hackers would love to accomplish and lay their hands on within this phase. In order to obtain and uncover all potential information related to a particular network, the hackers are required to employ many different scanning types, such as war-dialing, port scanning, OS fingerprinting, vulnerability scanning, and so forth. Associating with each of those scanning types is a relevant set of scanning techniques and tools that help the hackers accomplish the task obtain the desired information more effectively. This chapter will therefore discuss these crucial aspects of scanning in details to help you have a better understanding about the techniques taken by the hackers prior to the Penetrating phase. The chapter finally ends by introducing you to the tools and concepts behind proxy server and HTTP tunnel. Upon completion of this chapter, you will have been equipped with profound knowledge about scanning as well as its underlying mechanisms, and from there, be able to build an effective security defense to thwart those advanced scanning attempts carried out by the hackers.

SCANNING

As you may recall, footprinting is rather an easy task for hackers in that information about the target is freely available for public access and can be easily obtained by using non-intrusive reconnaissance methodology. However, footprinting alone is still not good enough to help hackers subvert the security architecture and penetrate into the target network, since information discovered during footprinting is still at a very preliminary level. For this reason, prior to the real hacking phase, it is imperative for the hackers to possess a greater amount of information pertaining to the target by utilizing sophisticated and aggressive reconnaissance techniques involved in the second phase of a hacking process — Scanning.

Scanning, as opposed to passive reconnaissance, requires hackers to actually communicate with the target network for acquiring further information. No longer can they go around and look for information from the public or open source databases as in footprinting. It is now time for them to say goodbye to the beloved Google search engine and start getting their hands dirty with assorted active network reconnaissance techniques in order to obtain useful information that reveals the details and all aspects of a network environment. For example, ping sweep and port scanning are both active reconnaissance approaches popularly used by hackers to determine live hosts and open ports.

However, actively probing and communicating with the target network also comes with an associated risk in which identity of the hackers will be exposed to a certain degree. Knowing this, most of the experienced and provident hackers often put a lot of consideration into this phase, in an attempt to effectively collect as much information

related to the network as possible, while at the same time, ultimately covering their real identity and keeping the attack low profile. Superficial hackers with an impromptu scanning plan and intrusive probes not only do give themselves away to the target, but also offer the target the opportunities to detect and prevent the attack before it could even be started. Thus, it should be noted that scanning itself requires the hackers to be much more cautious and considerate of what they do since no longer they can enjoy the same level of anonymity or protection as provided in footprinting.

Even though scanning a network might expose your identity to the target organization in someway, it is still certainly not a phase to be skipped or missed out prior to the real hacking phase — Penetrating.

SCANNING OBJECTIVES & TYPES

Hacking, be it in a movie or in the real world, is all about how much the hackers know about their opponent. The more information or intelligence they have about their target organization, the more chances for them to reveal the weak points, and thereby, taking advantage of such to gain unauthorized access to the network. Given that, since scanning is the precursor of the real hacking phase, it is unsurprisingly that scanning entails hackers performing numerous different types of reconnaissance so that the hackers can observe and scrutinize the target network from many different viewpoints, before carrying out the actual intrusion.

Knowing how to perform and what to achieve during scanning may differ to each of the hackers' perspective, as there are many conceivable scanning techniques and objectives, but typically, there are five commonly encountered scanning objectives in a premeditated hacking phase. These are (in sequential order):

- Finding accessible or active hosts on the network
- Discovering open ports or entry points to the network
- Identifying the associated network service of each port
- Detecting the operating system
- Finding the vulnerabilities

Objective	Type
A. Finding accessible hosts	Sweeping {Ping Sweep, War-dialing, War-driving, …}
B. Discovering open ports	Port Scanning {TCP Scan, UDP Scan, ICMP Scan, …}
C. Identifying network services	Application Mapping {Banner Grabbing, Response Analysis}
D. Detecting operating system	{Passive, Active} OS Fingerprinting
E. Finding vulnerabilities	Vulnerability Scanning

Exhibit 3-1. Scanning objectives and their respective types

Exhibit 3-1 lists some of the most popular scanning types employed by hackers for accomplishing those essential scanning objectives mentioned previously. It is important

to note that this list in no way can be an intensive list covering everything there is about scanning, instead, only the most common ones are provided.

Now that you have scratched the surface and known what hackers normally seek to achieve during scanning, it is time to move on to examine each of the scanning objectives and its respective methodology in detailed.

A. FINDING ACCESSIBLE HOSTS — SWEEPING

The initial scanning approach that hackers often take to map an organization network is *sweeping*, which is only conducted after the hackers already had some information about the address range of the target network. The address range can be a compilation of phone numbers or IP addresses which particularly differentiates one organization network from another. The idea behind sweeping the entire address range is to find out the number of active hosts or remote devices belong to the network, or in other words, the number of hosts that can be reached from the Internet.

Knowing the number of active hosts on the network can have substantial effects on the success of a hacking case wherein it offers the hackers a lot more possibilities for breaking into the network, rather than just the initially targeted host or main entrance. In real life, knowing the main entrances to or crucial hosts of the network will be constantly hammered and attacked by the hackers; network administrators often deploy significant security measures to thwart the foreseeable problems. As a result, hackers normally find it extremely unpleasant to infiltrate into the network through those thoroughly secured main entrances or servers, and that, forces them to unearth all accessible systems of the network for finding some other easier alternatives.

Upcoming section will help you have a closer look at the former king of network sweeping by introducing you to the concepts, tools, and purposes behind a *war-dialing* attack. Once you feel comfortable about the subject, you will be yet taken to examine another more popular and current form of network sweeping, namely, *ping sweeping*, where hackers rely on the prevalent IP network to locate accessible systems.

:.::.: WAR-DIALING

Traditionally, before the Internet becoming as pervasive as it is today, sweeping and all of its related tasks were normally conducted over the telephone network. By using a modem to dial into a range of phone numbers belonged to the target network, the hackers strive to search for responses from any remote computer that is connected to a modem. Based on the results, the hackers will then try to circumvent the security or protection implemented on those systems, if there is any, and ultimately, gain access to the network which those systems belong. The term used to describe the process of relying on the telephone network to find active hosts is *war-dialing* or *phone sweeping*. Although war-dialing now is not as ubiquitous as it was back in the day, it is still a very useful and widely used by attackers all around the world.

Practically speaking, the significance behind war-dialing is not really about discovering active systems of the network, but rather discovering any system that is connected to a modem and listening for incoming connection. You might be asking, "What's exactly wrong with having a computer connected to a modem? My computer is connected to a modem and I'm using Internet dialup just fine" Well, there would be no real concerns were your computer just a standalone personal computer solely used for Internet access and word processing. The per se problem lies in that if your computer takes part in a network where hundreds or even millions of dollars have been spent in an effort to create a strong and efficient security architecture to detect, prevent, and responds to outside threats, then, your little personal computer with that cute little modem and remote access software installed, such as PC Anywhere, will involuntarily create a security loophole, or backdoor, which facilitates the hackers' task and makes all those fancy security measures meaningless.

A heavily defended network with hundreds of thousand dollars worth of security mechanisms might not be as effective as it seems, in fact, it can only be as strong as its weakest link. In the real world, confronting the firewall, then the intrusion detection system, and even the intrusion prevention system just to get into the network is something not truly desirable by any hackers, for the reason that it is too much of an ordeal and exposure. Why spending all that precious resources and time to challenge the tough foe, when you can just go around and bully the little and weaker one? War-dialing is the scanning methodology designed to help hackers do just that, searching for negligent users who deploy no authentication, or weak form of authentication, for those remote control software installed on their modem-based computer systems. A stringent defense-in-depth security architecture that has a modem-based system attached somewhere in the chain is of the same kind as a fully lock down security vault that has two doors, where the one door utilizes biometric authentication schema and the other door just uses a conventional key and lock mechanism. Smart thieves, of course, will try to avoid going through the biometric door as much as they can, since it is highly unlikely that they can bypass the verification stage. The door using the conventional lock will be more tempting due to the lock can be picked easily and the key can be stolen and duplicated.

If a modem connected to a dial-up telephone line sitting inside a network can cause that much of damage, you may wonder why anyone would want to have it there in the first place. An organization may have the need to have modems installed in their network for a numerous of good reasons, but typically, facilitating remote control and administration of devices, such as HVAC (Heat, Ventilation, Air Conditioning) systems, voicemail systems, routers, servers and so forth. Having modems attached to the devices is not a cardinal sin as it might seem, since it provides the network administrators another means to communicate with the device should the network become unavailable. It is, indeed, the security unconscious and negligence of the users that make the present of such devices unacceptable anywhere in a well-secured and sophisticated security architecture.

Figure 3-1 demonstrates the two approaches that hackers frequently employ to hack into a relatively secured network environment. According to the figure, you can see that smarter hackers will obviously always try to hack the network using the easier approach,

which is performing war-dialing against the network to search for any unsecured modem attached, and thereby, gaining access to the vulnerable remote modem-based systems and the network as a whole.

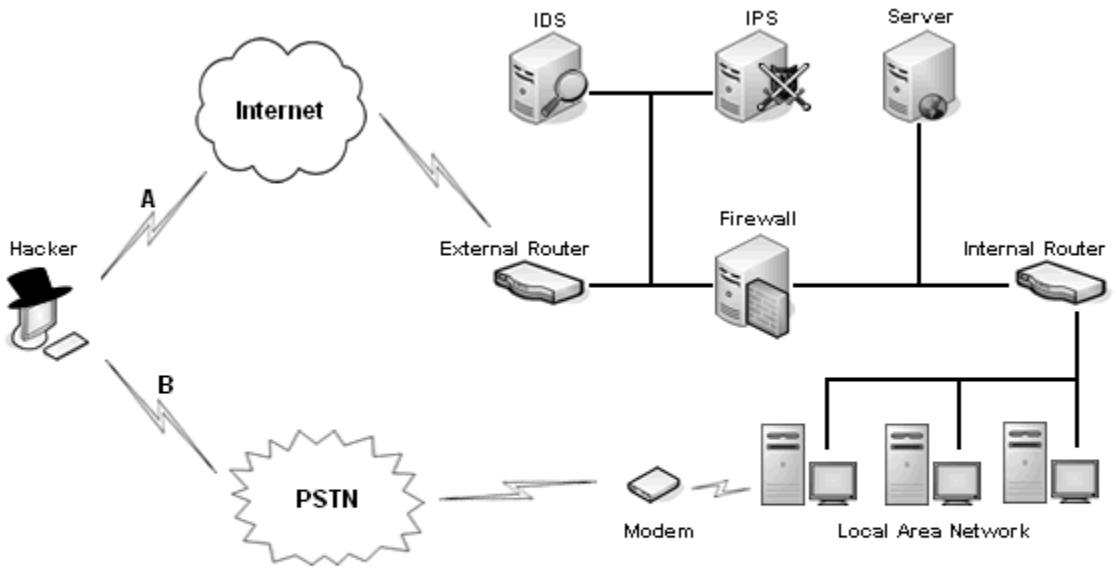

Figure 3-1. Hacking a well-defended network using war-dialing

When a connection to a particular system on a network is established through some other means different from the traditional network route, it is formally known as an *out of band* connection. In this example, the black hat hacker establishes an out of band connection to the network by relying on the PSTN (Publicly Switched Telephone Network) to communicate with the host over the unsecured modem, instead of the external router — which is where all network traffic normally should be sent.

War-dialing in practice can be much more useful than merely discovering active hosts on the network. Telecommunication fraud perpetrators also find war-dialing particularly useful in detecting repeat dial tones, PBX (Private Branch eXchange) devices that allow thru dialing. PBX, namely, is a device mainly used to control, switch, and manage phone calls within an organization's telephone network, allowing hundreds or even thousands of users to share a certain number for making outside calls, instead of dedicating each phone line to each user on the network. In order to place a phone call, PBX users are required to enter a predefined number to get to the second dial tone, and subsequently, authorize themselves by entering yet another sequence of special numbers, so-called authorization code. Nonetheless, in today's world, it is still very often to encounter poorly configured PBX systems with sloppy security mechanisms that require no verification or authentication of any sorts, letting those who request accessing the systems and using the service arbitrarily with no restriction. For this reason, it is very understandable why PBX devices are still the hottest target for the hackers and phone fraud perpetrators during war-dialing.

GET READY FOR WAR-DIALING

A typical war-dialing has set many different requirements which hackers must be able to fulfill before they can get the party started, but regardless of what the original intent of carrying out a war-dialing might be, hackers are all required to must have a defined range of phone numbers, a war-dialing software, and needless to say, a decent modem.

Determining the phone numbers or telephone exchange of a target organization and feeding it to the war-dialing tool is a simple task because such information can be easily collected from various public information sources, but some of the most popular ones include the phone book directories, the whois databases, the target organization's website and dumpsters, and so on. In the event those public information sources do not reveal the telephone exchange of the target organization, the hackers may also attempt to carry out social engineering to find out such information. It is imperative to note that the more phone numbers that the hackers can collect, the easier for them to infer the telephone exchange or potential range of the target organization's phone number. Such information is undeniably very helpful in narrowing down the scope and assisting the war-dialer to tackle the right target, which as a result, can save a tremendous amount of sweeping time.

Having a defined range of phone numbers of the target is a good thing but it is still not good enough. Instead of manually dialing an intensive list of phone numbers week after weeks, the hackers will try to make a smarter move by using a program that can automate such daunting process. The software used to dial a list of phone numbers, sequentially or randomly, provided by the hackers and detect the responses is normally referred to as a *war-dialer*. There are many war-dialers available — in both commercial and non-commercial form — providing the users or hackers many options when it comes down to choosing the best tool for the job. Each of the tools may differ in the number of functions provided but they all share and head to a common goal of war-dialing, which is detecting the modem carrier tone. One of the most powerful and freely available war-dialers up to date is the THC-Scan, although from time to time you will still hear people mentioning about ToneLoc — an obsolete tool which was once considered as the kingpin of war-dialing. A THC (The Hacker's Choice) member, Van Hauser, inspired by the underlying concepts and functions of ToneLoc, coded THC-Scan to address the shortcomings of ToneLoc and include a whole lot more functions into the tool. You will be introduced to THC-Scan shortly, but first, for those who don't know where to grab this "hacker's choice" war-dialing tool, feel free to pay the fine folks at http://www.thc.org a visit.

In all, when the range of phone numbers of the target organization is determined and the most appropriate war-dialer is selected, the hackers may now start their misadventure with no holdback. For instance, if the target organization's website indicating the IT Support department can be reached at 240-777-1230, then, the hackers can base on that provided number and instruct the war-dialer to dial each phone numbers between the range of 240-777-1000 and 240-7777-3000 one after another, and thereby, search for any responses coming from unsecured modems or PBX devices on the target network.

THC-SCAN

THC-Scan is often thought of as the successor of ToneLoc — a very powerful war-dialing tool back in the 1990's. Even there has been no major changes, or updates, since its last release in 1998, THC-Scan still proves to be the most robust war-dialer available today, for the reason that it is absolutely free and packed with many advanced features that allows you to tweak your war-dialing to whatever your liking to yield the most productive and reliable results.

Installing THC-Scan is relatively easy and straightforward since all you need to do is to extract all the files in the `thc-ts20.zip` archive to a specified folder, which will hold all the executables and necessary files for you to start THC-Scan. All of the binary files found in the `bin` subdirectory can be run in any DOS environment including MS-DOS, DR-DOS, PC-DOS, and DOSEMU under Linux.

The main executable file that does the real job is the `thc-scan.exe` but before you can execute the file and start war-dialing, you must first run the `mod-det.exe` to extract hardware information regarding your installed modem, which is one of the most important pieces of information that THC-Scan really needs to get going.

You will then need to run `ts-cfg.exe` to generate a configuration file for THC-Scan and change all the default modem settings to those that are corresponding to your modem hardware based on the information provided by `mod-det.exe`. Besides allowing you to change modem settings, `ts-cfg.exe` also offers a variety of other configuration options that let you adjust your scanning for the most optimal results, consider going through all the doc files come with THC-Scan should you have any problems with tweaking the program or configuring your modem.

Once you have set all the desired options, it is time to trigger `thc-scan.exe` and get the party started. You will be required to append a DATAFILE and DIALMASK upon triggering the executable thc-scan.exe. The following is the complete command that will actually get you to start your war-dialing (with the extra option specifying the range of phone numbers to scan):

```
C:\>THC-Scan\bin\thc-scan fun -m:240-777-xxxx –r:1000-3000
```

According to the above example, you can see the war-dialer was set to dial all phone numbers between the 240-777-1000 and 240-777-3000 range. `Fun` was specified as the data file and the `–m` and `–r` options were used to specify the prefix and range of phone numbers to dial. The dash "-" is not essential to the war-dialer, it was added for clarity and you can safely remove it without any problem. Figure 3-2 shows a screenshot of THC-Scan in execution.

It should be noted that war-dialing is such an intensive topic that it is impossible to wrap everything in just a few pages, let alone offer you a complete analysis of war-

dialing, or THC-Scan in particular. The program THC-Scan itself offers a whole lot more features and options for you to much around with, rather than just the `-m` and `-r` as seen above, please feel free to spend some times to read the manuals provided for all other available options.

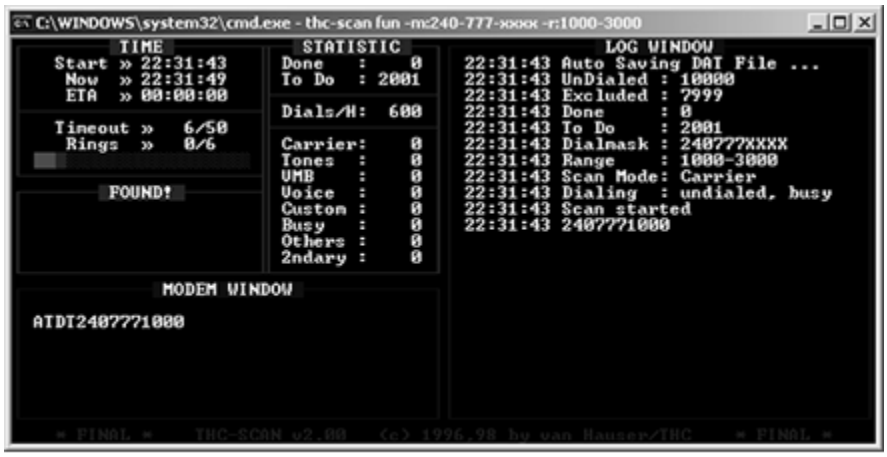

Figure 3-2. THC-Scan in execution

It can't be emphasized enough that you, the ethical hackers, must first obtain permission, in written form, before performing war-dialing against your target organization. Without such, it is probable that war-dialing will lead you to several legal implications, no matter how legitimate it might sound.

COUNTERMEASURE: NO ROGUE MODEM PLEASE

It is obvious that the first step that an organization can take to mitigate the risks induced by war-dialing is to prevent rogue modems sitting inside the network at all. A stringent modem policy must be existed and enforced to facilitate inventory of all legitimate modems and dial-in lines throughout the whole network, as well as to punish those who attempt to install and use modems without authorization.

Even when there is a stringent modem policy in place, external or rogue modems will still be brought and installed in the network by negligent users for a variety of reasons and excuses. Hence, the only way to find out such covert uses of modem is to do what the hackers would do, conducting war-dialing against your network on a regular basis. War-dialing is better to be conducted after working hours to avoid interruption of normal business operation such as engagement of telephone lines. Moreover, users are more inclined to have their sinful remote control software and modems disabled temporarily while they are at work and only have all those enabled again before going home; therefore, war-dialing during working hours may only provide you an incomplete picture and false sense of security about your network's security posture.

Upon discovering systems that are actively open and offer weak or no authentication, it is your responsibility to review the security policy and either remove such systems

from the network completely, or strengthen the security safeguard implemented on those systems to add difficulties to the hackers' task and prevent easy access. Welcome banners should be reviewed and edited in a way to limit the amount of information that the hackers can learn about the remote systems during reconnaissance. Two factors authentication should also be considered applying for critical modem-based systems that have a vital role in the network, for example, requiring users to supply their private key and password before granting access to a database server.

Another method that you can often take to secure your dial-in architecture is to isolate all the systems that must be accessed remotely into a group or zone. By isolating such systems from the internal network into a specific zone, you just add another layer of security to your network infrastructure by preventing hackers from taking advantage of the vulnerable remote access systems and gaining access to the internal network. Besides, it is always easier to manage and concentrate on the security of a specific zone that has specific systems.

:.::.: PING SWEEPING

In case war-dialing didn't reveal anything useful, the hackers normally can just safely forget about that unfortunate event and may try to look for accessible hosts on the target network by using another popular sweeping method known as *ping sweeping*.

`Ping`, similarly to `traceroute`, is a popular network troubleshooting utility that relies heavily on the ICMP (Internet Control Message Protocol) to function. By sending an ICMP "echo request" message to an IP address of the respective host, `ping` requires the receiving host to send an acknowledgement, or more specifically, an ICMP "echo reply" to the originator. If there is no ICMP echo reply packet returned, then `ping` assumes the target host is either dead or non-existent, and by using this technique repetitively against several IP addresses of the target organization, the hackers can elicit live hosts on the network without a hitch. However, just like war-dialing, manually pinging a range of IP addresses belong to the target network can be very a overwhelming task, and that, leads to the advent of ping sweeping which is an automate pinging process that attempts to ping a huge number of hosts, or IP addresses, in a short amount of time.

Here is what it looks like to use `ping` to find out whether a host named n0p in the domain gotrice.com is actually reachable from the Internet. This is the ugly and slow way to carry out the task and you certainly don't want to do this same thing all over again for 253 other IP addresses yourself, without the help of a ping sweeping utility.

```
C:\>ping n0p.gotrice.com
Pinging n0p.gotrice.com [203.83.34.24] with 32 bytes of data:

Reply from 203.83.34.24: bytes=32 time<2285ms TTL=47
Reply from 203.83.34.24: bytes=32 time<1318ms TTL=47
Reply from 203.83.34.24: bytes=32 time<2296ms TTL=47
Reply from 203.83.34.24: bytes=32 time<3284ms TTL=47
```

The output is self-explanatory, indicating the host n0p.gotrice.com whose associated

IP address is 203.83.34.24 is alive and kicking. You know that because the remote host acknowledges the four ICMP Echo Request messages sent by `ping` with its own four ICMP Echo Reply packets. Notice how `ping` also performs a forward name resolution for the host n0p.gotrice.com using the current default DNS server.

To resolve a host name into its corresponding IP address quickly, it is best for you to have the `ping` utility handle the job given the fact that it is so prevalent and relatively easy to use. Don't touch those advanced DNS tools like `nslookup` or `dig` for simple task as such, try `ping` first.

Although ping sweeping does seem to be the most effective reconnaissance technique that can help hackers inventory active hosts on the network, it might not always provide reliable or expected results if ICMP messages were purposely blocked by the border or external router (a laudable deed of the security paranoid) of the network. In real life, many network administrators often choose to disable ICMP traffic altogether so that the threats caused by malicious uses of ICMP can be reduced to a minimal, and therefore, assuming the target hosts inaccessible just because there is no ICMP echo reply packets returned is indeed a fatal oversight. Yet, determined hackers will not give up so easily as they will strive to discover active systems by sending TCP or UDP packets to the target network in lieu of those notorious ICMP messages. Note, simply sending TCP or UDP network packets arbitrarily will not be of any use, the hackers will need to send them packets to well-known Internet ports, such as port 80/TCP (HTTP) or 53/UDP (DNS), in order to avoid being dropped by the routers.

ICMP Query Types	Description
Type 8 — Echo Request	This is what the infamous ping utility uses to determine the accessibility of a remote host, sending an ICMP Echo Request message and waiting for the ICMP Echo Reply from the target.
Type 13 — Timestamp Request	Querying the current time of a remote node. Using this allows one to calculate the latency of a network as well as to alternatively discover active host should ICMP Echo Request message is explicitly blocked by the target organization.
Type 15 — Information Request	Rather obsolete, this type of query is useful for a diskless workstation to discover its own IP address at boot-time.
Type 17 — Address Mask Request	Querying a particular host for its subnet mask. This type of query is useful in helping the hackers determine the size and address space of the target organization.

Exhibit 3-2. Different types of ICMP queries

As can be seen from Exhibit 3-2, ICMP allows one to conduct many different types of query against the target host, rather than just the ICMP Echo Request (Type 8). In order to send custom ICMP queries, or any other queries beside the Echo Request, you can use

the ICMPush, ICMPQuery, and SING (Send ICMP Nasty Garbage) tools for this kind of activity. They can be easily downloaded from http://www.packetstormsecurity.org.

If you want to learn more about ICMP probing techniques, *ICMP Scanning V3.0*, written by Ofir Akin from the Sys-Security Group, is a great security paper that provides in-depth examination on many different techniques and aspects of ICMP scanning which you may apply in real life to facilitate your ethical hacking assignment. It is available for download at http://www.sys-security.com.

As usual, upcoming sections will introduce you to a handful amount of scanning tools that can make ping sweeping an enjoyable and exciting moment. It is explainable why some of you are still very reluctant to hacking tools and such, but as long as you folks can understand the mechanisms behind each of those tools, you can be sure that the term *script kiddies* is simply not for you. Note, some of the tools you are about to see are extremely inappropriate to today's world in term of quality and usability, but to conform to the official C|EH course outline, they will still have to be introduced, but briefly.

FRIENDLY PINGER

Similar to those network inventory tools that you may have seen in previous chapters, such as Visual Route, Friendly Pinger incorporates many neat network utilities into a single software package, allowing users to perform `ping` and all other network monitoring and inventory related tasks through a friendly and an intuitive GUI interface.

You can grab the latest trial version of Friendly Pinger, 5.0 at this time of writing, from its official web site at http://www.kilievich.com/fpinger. If you plan to use Friendly Pinger longer than a 30-day trial period, you are required to pay $68 USD for the registration key which will allow you to unlock the software for unlimited uses.

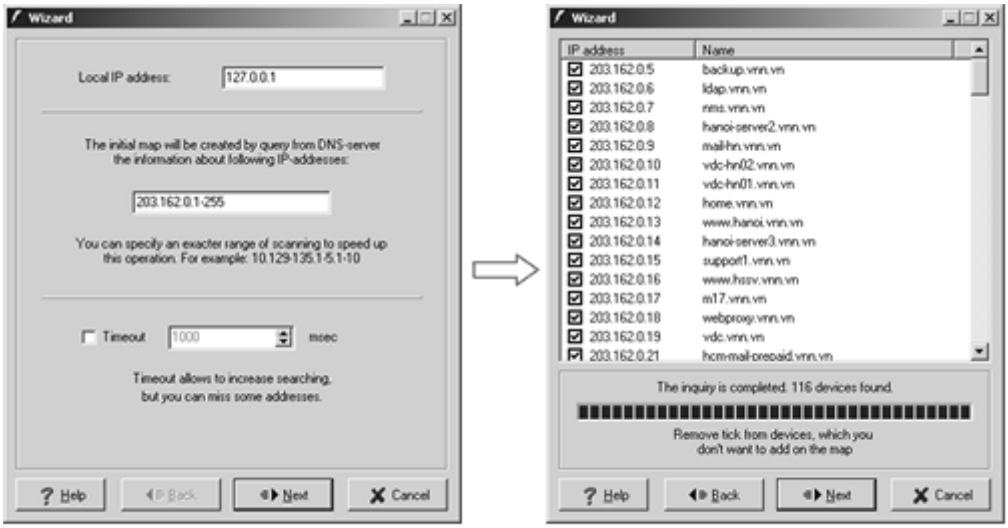

Figure 3-3. Friendly Pinger's Wizard

Initially, you will need to create a map if you want to work with Friendly Pinger at all. You can do so by either manually adding all possible remote devices of the target network to the map or using the Wizard found under the File menu option. It is recommended that you use the Wizard to accomplish this task since it is the whole point for you using this friendly and pretty pinging tool, isn't it? Figure 3-3 shows how you can use the Wizard of Friendly Pinger to create an initial map based on a provided range of IP addresses from 203.162.0.1-255. This map is where all your pinging and scanning works will mainly be based on. Note, the Wizard does not ping all those provided IP addresses just yet, it merely queries the DNS server to resolve the IP addresses to their respective host names and only those that can be resolved will be added to the map.

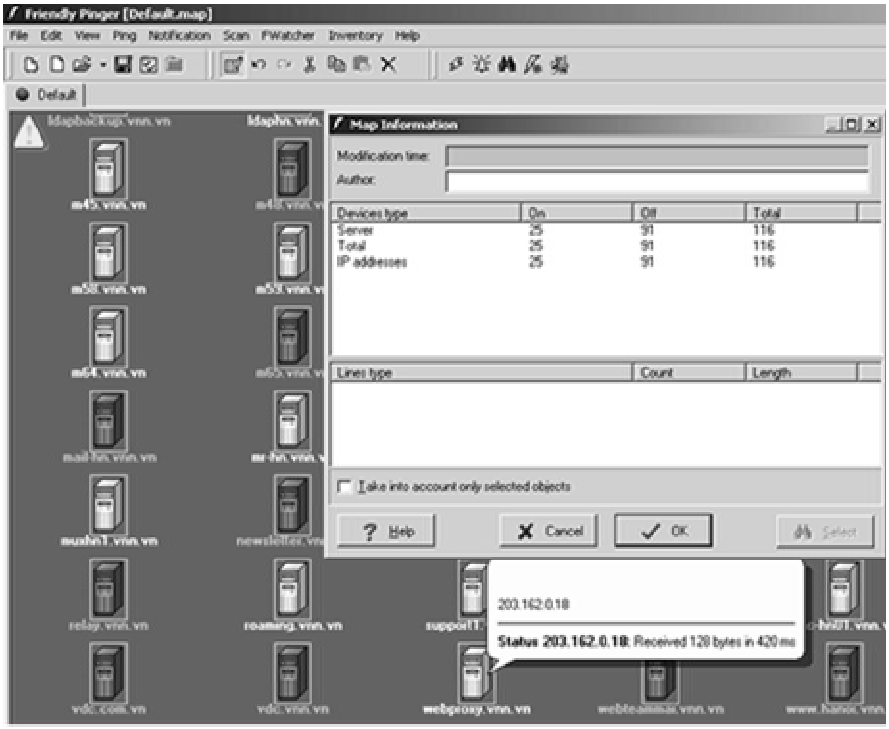

Figure 3-4. Friendly Pinger's Map and Map Information

Once you are done with the Wizard, you will be taken to the main interface of Friendly Pinger, which really is just a graphical map listing out all the devices that have resolvable IP addresses. Therefore, it is imperative to note that all the devices shown in the map do not necessarily mean they are all active. In order to ascertain which devices are active, you will have to wait until Friendly Pinger finished pinging all the devices. Figure 3-4 shows screenshots of the main map view of Friendly Pinger and the Map Information window. Active IP addresses or remote devices are represented in white icons whereas those that are not active and non-responsive are in black and darker icons. Friendly Pinger will continuously ping and monitor all those remote devices or systems until it is explicitly configured to pause or stop the action. The Map Information window provides detailed information of the map such as the number of IP addresses that are active (on) and non-active (off), the total number of IP addresses and the time when the map was last modified.

Overall, Friendly Pinger supports many other unique features besides pinging, such as scanning, graphical traceroute, notification when a device wakes up or goes down, port scanning, user access tracking, and so on. Discussing each of these features in-depth is certainly beyond the scope of this book but should you have any further needs to study the tool thoroughly, please spare a few minutes to consult with the official manual.

CHEOPS

As Friendly Pinger is only available for Windows users who can afford to pay $68 USD for the software, one folk from the open source software community, Mark Spencer, doesn't seem to like that idea very much and decides to offer Linux worshippers a similar toy for the Linux platform, namely Cheops. You can download this fine piece of program at no cost from the author's website at http://www.marko.net/cheops.

Cheops is a network-mapping and monitoring tool that houses a comparable set of features like Friendly Pinger, such as, displaying a visual layout of the target network, detecting and monitoring active hosts from a given range of IP addresses, port scanning, and event notification, but perhaps the most interesting feature that typically outclasses Friendly Pinger is the OS detection. Cheops will identify the OS of active hosts on the network and conveniently depict them in the most appropriate icons via the graphical user interface (as shown in Figure 3-5, can you folks spot the cute little penguin?).

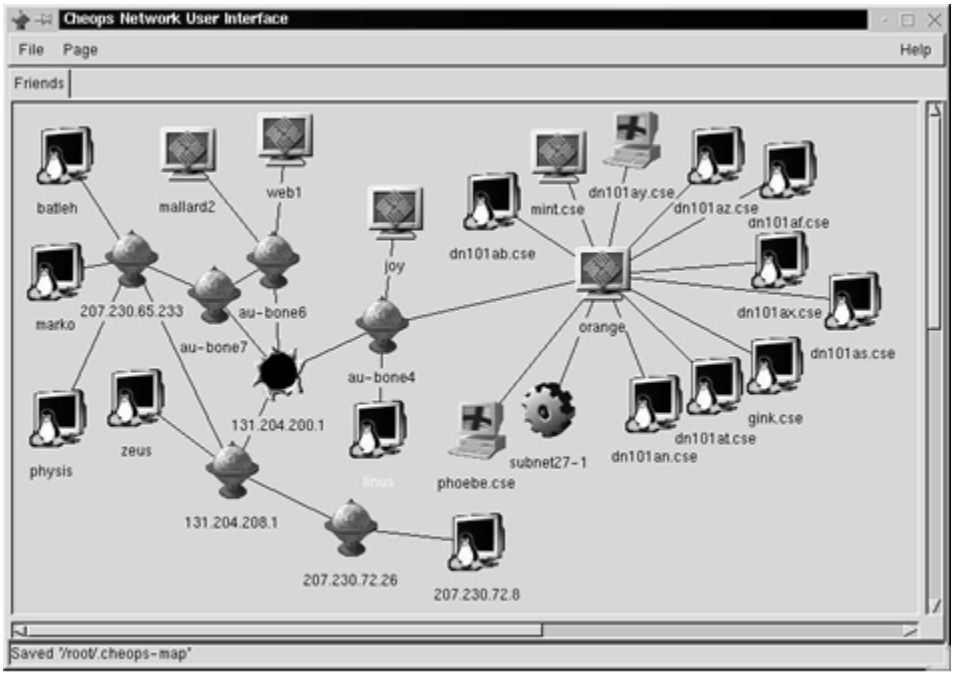

Figure 3-5. Cheops' main interface

For brevity, this section won't go too deep in discussing the mechanisms that make up Cheops since `ping` or `traceroute` or `nslookup` should be no stranger to you by now. The techniques used by Cheops for OS detection and port scanning will be discussed

more thoroughly later in this chapter. It can't be stressed enough that you must use Cheops with great consideration because the tool was originally developed for the sole purpose of assisting people monitoring their networks, and for that very reason, Cheops makes no real efforts at helping one stay undetected by the remote network.

NMAP — NETWORK MAPPER

To date, nmap is the most respected and well-known port scanner in the hacker community for its efficiency, flexibility, and lightening scanning speed. Coded by Fyodor and packed with many different unique features, nmap gives you all the flexibilities you need to adjust your scanning for optimal results. You will be able to examine nmap much more thoroughly later in the chapter, but for now, let me swiftly introduce you to a quick and dirty way to sweep an entire range of IP addresses (class C) using nmap version 3.50 for Linux (Windows version is also available). By the by, you can get this indispensable hacker's tool from its official website http://www.insecure.org/nmap/

```
[bash]$ nmap -sP 203.162.0.0/24

Starting nmap 3.50 (http://www.insecure.org/nmap/) at 2004-08-11 16:53
Host vdc-hn01.vnn.vn (203.162.0.11) appears to be up.
Host home.vnn.vn (203.162.0.12) appears to be up.
Host www.hanoi.vnn.vn (203.162.0.13) appears to be up.
Host support1.vnn.vn (203.162.0.15) appears to be up.
Host webproxy.vnn.vn (203.162.0.18) appears to be up.

---results truncated for brevity---

Host fone.vnn.vn (203.162.0.195) appears to be up.
Host 203.162.0.196 appears to be up.
Host hanoi-cs7513.vnn.vn (203.162.0.250) appears to be up.
Nmap run completed -- 254 IP addresses (59 hosts up) scanned in 17.055 seconds
```

Do you see how fast nmap performs ping sweeps for an entire class C subnet (254 hosts)? It only takes a little over 17 seconds. I must tell you that I am using an ancient 56kpbs Internet connection (you can laugh all you want) and had I got a decent Internet speed like 128kpbs or 512kpbs, the results could have been much faster. Nmap reports 59 out of 254 hosts belong to the 203.162.0.x network are up and compare this result to Figure 3-4 (Friendly Pinger's Map and Map Information) you will probably notice how each of the tools reports the numbers of found accessible hosts differently, only 25 hosts belong to the 203.162.0.x network are up according to Friendly Pinger. That happens because Friendly Pinger only tries to ping hosts that have resolvable IP addresses — which can be fairly inaccurate if a given host doesn't have a fully qualified domain name — whereas nmap will attempt to ping all IP addresses, regardless of whether they can be resolved to some pronounceable host names or not. In addition, nmap will also strive to look up host names for the IP addresses in question.

COUNTERMEASURE: ECHO REQUEST <> ECHO NO REPLY

You can effectively thwart ping sweeps and reduce the amount of information that an

attacker can learn about your network by filtering ICMP Echo Request message (Type 8) at the border router. In doing so, not only will you be able to bring more challenges to the game but also minimize the occurrence of Denial-of-Service (DoS) attacks caused by misused ICMP Echo Request packets. Some people even go as far as stopping all ICMP traffic, irrespective of its type, to eradicate the root cause of all information leakages and security problems relating with the protocols, but that also has its own side effects since many types of ICMP can be very useful for administrative and troubleshooting purposes. Moreover, it is kind of a misconception to think ping sweeping utilities merely rely on ICMP to function, as you will very soon see TCP or UDP can be used as an alternative approach. Hence, it should be noted that there is no single definitive method to prevent ping sweeps because it is a matter of balancing security and administration capability of a network and you must always consult with your current security policy to configure your border router accordingly.

In the events where you can't stop ICMP traffic and must let it pass your router for a variety of good reasons, such as verifying network connectivity, you can still at least look out for suspicious ICMP packets indicating that a full-scale attack is on its way. By using an Intrusion Detection System (IDS) to proactively search for those ICMP packets that do nothing but scouting the network, you can have your chance to prepare and well stop the real attack before it could ever have a chance to occur. For network-based IDS, it is recommended that you use Snort (http://www.snort.org) because it's powerful, versatile, free (the most important factor), and well supported by the open source community. It is highly recommended that host-based IDS must also be installed on critical servers to minimize the amount of information that the hackers can collect from such machines. ISS BlackIce (http://www.iss.net/products_services/products.php) is a commercial host-based IDS, as well as a firewall in a sense, for Windows that can be used to detect and prevent many other different attack vectors besides ICMP ping sweeping, including port scanning and DoS. There is a myriad of free ping sweeping detection utilities that can be used for Linux and UNIX systems, but in particular, you should check out Protolog, Ippl (my personal choice), and Scanlogd. These utilities can be searched for and downloaded from everyone's favorite security portal, http://www.packetstormsecurity.org.

The last step that may be very worth to consider is separating the machines that you or the public need to ping into a different zone, such as the DMZ (Demilitarized Zone), so that other systems on the local network do not have to deal with the nefarious hackers and their mischievous ICMP packets. For instance, if your only need is to constantly verify and assure the connectivity of a web server sitting in your DMZ using `ping`, it is advisable to place the rest of the computers that do not require the same attention into a zone, where you can unambiguously configure your router to drop and deny any ICMP packets heading for that zone, including Echo Request, Information Request and Time-stamp Request packets. Only the DMZ zone where your web server resides should be able to receive ICMP traffic that is explicitly permitted by your organization's security policy. In addition, you should only allow incoming ping or ICMP traffic from a certain range of IP addresses or authorized machines to reduce the likelihood of abuse cause by unauthorized Internet users.

B. DISCOVERING OPEN PORTS — PORT SCANNING

Once all possible active hosts on the network are discovered, the hackers will move on to the next phase, finding out open ports on a remote host. A port is equivalent to a point of entry to a system, providing a means for authorized users communicating and exchanging data with the system in question through the listening network service. Yet for the hackers, a port listened by a network service means so much more. It represents a potential point of vulnerability where one may try to exploit and gain unauthorized access to the system. The ultimate goal of the hackers here is to find out as many open ports on the remote systems as possible. The more open ports can they discover, the more possibilities will they have for hacking into the system, or the network as a whole.

So how does a hacker determine whether a port is open or closed? Well, in a typical port scanning fashion, the hackers will attempt to send out a network packet, or more specifically, a request for connection, to a particular TCP or UDP port on the target system and then wait for a response. If there is no response, they will have to cross their fingers and scan for another port. However, if there is a network service listening for incoming connection on that particular port, it will acknowledge the hackers with a response. Upon receiving the response, the hackers may choose to conduct further probes against the system to identify the software used and in the end they will try to exploit and gain unauthorized access to the system through that specific vector — assuming the network software is misconfigured or vulnerable to remote exploitation.

Since this section is talking about various port scanning techniques and types, you will probably want to spend a few minutes here for discussions regarding TCP and UDP ports before proceeding any further.

A system with a typical TCP/IP suite has 65535 TCP ports and another 65535 UDP ports. However, in practice, it is extremely rare to come across a system that uses up all available TCP and UDP ports as such so it is not necessary for you to find out and memorize what each of the ports offers individually. Generally, TCP and UDP ports can be divided into two categories, Well Known Ports and Ephemeral Ports.

The Well Known Ports are those common ports that you encounter everyday, ranging from port 0 to 1023 (1024 ports in total), some examples of which include port 80/TCP (Web service), or port 53/UDP (DNS service). The well-known ports are pre-assigned by the IANA and normally can be used only by privileged processes or users on a system, such as root, administrator, or super user. It should be noted that you can indeed assign any port number to a network service, but in the norm, well-known port numbers are assigned to popular network services, such as web or mail, to facilitate communication with other systems. You can download a current list of well-known ports assigned by the IANA at http://www.iana.org/assignments/port-numbers.

The Ephemeral Ports are much less common to the server as they have a much higher port numbers, from 1024 to 65535. These port numbers are set aside for unprivileged or ordinary users who have the needs to run normal networking processes or programs, for

example, ftp or email clients. Of course, privileged users can also occupy these ports if needed. The Ephemeral Ports can be further divided into two specific categories, The Registered Ports (ranging from port 1024 to 49151) and Dynamic/Private Ports (from port 49152 through 65535).

Instead of spending so much time to verify all 65535 TCP and UDP ports (131070 in total) manually, the hackers can use automated tools, so-called port scanners, to handle and speedup this tedious but yet important task. When it comes to picking which port scanners should be used for the job, users normally have to rack their brains for the best one, and in the end, buried alive with zillions and zillions of options. Fortunately, to save you readers from such catastrophe, Exhibit 3-3 provides you with a list of efficient and recommended port scanners in the field. Note, the keyword Linux as mentioned in the exhibit implicitly refers to all Linux and UNIX-like platforms unless stated otherwise.

Port Scanner	Platform	Download Web Site
Nmap	Linux, Windows	http://www.insecure.org/nmap
Strobe	Linux	http://www.netsw.org/net/ip/firewall/attack/strobe
SuperScan	Windows	http://www.foundstone.com/resources/scanning.htm
Hping2	Linux	http://www.hping.org/download.html
Blaster Scan 3.1	Linux	http://www.l0t3k.net/tools/Portscanner/blasterv3.1.tgz
NetScan Tools Pro	Windows	http://www.netscantools.com

Exhibit 3-3. Popular Port Scanners

Given the complexity and flexibility of the TCP/IP stack, port scanning techniques adopted by port scanners and hackers for scanning a network can be very diverse in nature. Generically, they can be divided into three distinct categories, denoting for three different levels of anonymity:

- Open Scanning
- Half-open Scanning
- Stealth Scanning

Besides those three popular scanning methods, you will often encounter these two in the real world as well:

- Sweeping
- Miscellaneous

Before jumping to the forthcoming section for detailed discussion on each of these port scanning methods, it is essential that you have a fair understanding of networking or the TCP, UDP, and IP protocols in particular. For an official document that describes and explains each of those network protocols in details, you perhaps should be looking for what is known as Request For Comments (RFCs). Specifications about the UDP protocol can be found in RFCs 768. RFCs 793 and RFCs 791 contain specifications about the TCP and IP protocols respectively. They can be downloaded from http://www.ietf.org/RFCs/.

:.::.: OPEN SCANNING

Open scanning, also known as *vanilla* or *TCP Connect()*, uses a TCP/IP three-way handshake to make a full connection to a remote computer to look for a listening port. RFCs 793 describes this communication procedure more thoroughly, but if you don't have the time to spare to go through that lengthy RFCs, you might want to refer to Figure 3-6 which clearly depicts a TCP/IP three-way handshake.

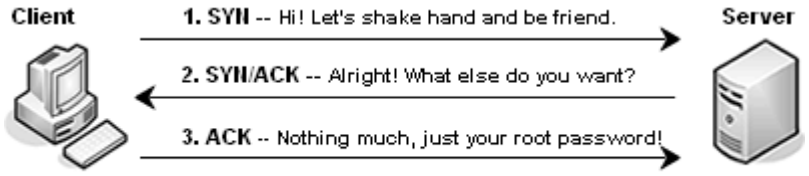

Figure 3-6. TCP/IP three-way handshake

The client that wants to establish a connection must first send a SYN (Synchronized Sequence Number) packet to the desired port on the server. The server will acknowledge the client with a SYN/ACK (Acknowledgement) packet if the port in question is listening and accepts incoming connection. Finally, the client sends an ACK packet back to the server to establish the connection, and real transaction, or data exchange, between the two computers may be started from there. However, if no network service is listening on that particular port, the server will alternatively send a RST(Reset)/ACK packet to notify the client. TCP/IP three-way handshake vaguely resembles to what you normally do before starting a conversation over the phone. You don't just say "I love you" and all that without knowing if it is really your loved one on the other line (it could be anyone, like the grouchy parents). Instead, you will be better off saying hello and waiting for a response from the other line to see if the person you're really expecting is available.

Because a full connection must be made to each port on the remote system, it is apparent that open scanning can be easily detected and logged by the administrators or intrusion detection systems (IDS) for all the noises that it makes during scanning. However, from a brighter side, this is certainly one of the most accurate, reliable, and fastest scanning methods there is because it actually establishes a full connection to the remote system to get a firm confirmation whether or not a port is listening. Moreover, hackers are not required to have a special super-user or root privilege to perform open scanning because this method doesn't require them or their port scanners to compose any special or custom TCP packet in particular.

:.::.: HALF-OPEN SCANNING

Do you still remember the time when you used to go to someone house, ring the doorbell, and run to a place where you can see the irritated face of your victim and then burst into laughter? Well, congratulations, now you can enjoy that little funny game from your own computer as half-open scanning, or also formally referred to as *SYN scanning*, is mostly done based on that "naughty" concept.

SYN SCANNING

Contrary to the open scanning method, SYN scanning doesn't have the goodwill to make a full connection to a remote computer and it tears down the TCP/IP three-way handshake with no remorse, before giving it a chance to complete. Initially, the client sends a SYN packet to the server as if it is going to make a real connection. The server then responds with a SYN/ACK packet if the desired port is open and listened by a network service, or a RST/ACK packet to indicate the requested port is closed. Up until this stage, the client already knows what it wants — whether a port is open or closed — so it decides not to send an ACK packet to complete the TCP/IP three-way handshake, instead, a RST packet is sent to destroy the connection (as shown in Figure 3-7). "Hey buddy, thanks for the fish, but I don't want to talk to you anymore, goodbye!" This is perhaps to be what the client would say to the server via that RST packet ultimately.

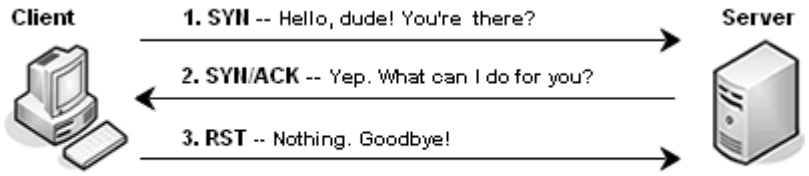

Figure 3-7. Half-Open Scanning (Open Port)

Since the required ACK packet can never be received by the server in order to establish an actual connection, the server pretends as if nothing happens and doesn't record the event into its log file. Thus, SYN scanning can be very stealthy when compared to open scanning; it lets the hackers to conduct port scanning without the fear of being detected by the remote system. This stealthy technique also proves to be just as reliable as open scanning in that it waits for a firm response from the remote server in order to determine whether a port is listening.

It is critical to note that as SYN scanning gradually becomes more popular, many advanced IDS or firewalls are now capable of logging and thwarting this particular type of scanning. In addition, anyone who wants to perform this type of scanning is required to have a special privilege, such as root, for the construction of customized TCP packets.

IP ID/IDLE/DUMB SCANNING – THE VARIANT OF SYN SCANNING

Idle scanning, also known as *IP ID header* or *dumb* scanning, is a similar method like SYN scanning, requiring an attacker to send an initial SYN flagged packet to the remote host as if a TCP/IP three-way handshake is on the way. Nonetheless, what sets the two methods apart is that for idle scanning the remote host may never know the true source of the attack, or in other words, may never find out who actually sent that devilish SYN packet. But how is all this possible? Well, in order to attain such an extraordinary level of anonymity, the attacker must be able to find and use a dumb host as a scapegoat. A dumb host is an inactive host that sends and receives a little or no network traffic at all, and you will see why this is a crucial factor very soon.

Typically, the attacker will have to repetitively ping the dumb host and analyze the ID (Identification) field of IP header of the packets returned by the dumb host. As a rule for many operating systems, for every network packet sent, they must increase the value in the IP ID field by one. Therefore, since the dumb host doesn't communicate with anyone else than the attacker at this point, the IP ID field should be sequentially increased by one for each ICMP Echo Reply packet it sends to the attacker (as in the following example)

```
len=28 ip=192.168.0.16 ttl=255 id=3501 seq=0 rtt=0.7 ms
len=28 ip=192.168.0.16 ttl=255 id=3502 seq=1 rtt=0.8 ms
len=28 ip=192.168.0.16 ttl=255 id=3503 seq=2 rtt=0.3 ms
len=28 ip=192.168.0.16 ttl=255 id=3504 seq=3 rtt=0.4 ms
```

Keeping in mind 3504 is the IP ID number of the last packet returned by the dumb host, the attacker then sends a spoofed SYN packet to the target system using the IP address of the dumb host as the source address. Suppose the port in question is listening, the target system will acknowledge the dumb host (owing to the spoofed IP) with a SYN/ACK packet to carry on the three-way handshake. Upon receiving a SYN/ACK packet which it never expects and asks for in the first place, the dumb host will send a RST/ACK packet to reset the connection, and as a result, its current IP ID number is now changed and increased by one. Now, the attacker will attempt to ping the dumb host and analyze its IP ID field again (`hping2` is a tool of the trade, you will be shown how to use this in a moment). If the IP ID number of the dumb host has been increased by two since the last time the attacker saw it, the attacker then can make a safe guess that the scanned port is open. Here is the log which captures the ping session conducted after the spoofed SYN packet has been sent to the target system:

```
len=28 ip=192.168.0.16 ttl=255 id=3506 seq=0 rtt=0.4 ms
len=28 ip=192.168.0.16 ttl=255 id=3507 seq=1 rtt=0.5 ms
len=28 ip=192.168.0.16 ttl=255 id=3508 seq=2 rtt=0.7 ms
```

According to the log, the attacker can observe that the value in IP ID field has been indeed increased by two, from 3504 the last time and straight to 3506 this time, implying the dumb host has sent one network packet to another host, or to the system being probed, before sending replies to the attacker's ping packets. If the port being probed on the target system is closed, a RST/ACK packet will be sent to inform the dumb host and reset the connection. Since the RST/ACK packet is invalid and not expected by the dumb host, it will be disregarded and the dumb host will not send any response back to the target system, and thus, its IP ID field should remain *unchanged*.

Figure 3-8 illustrates this creative, yet complicated, scanning method in a more understandable manner, which may help clear any confusions regarding idle scanning.

There are two most important elements that the attacker must know before conducting idle scanning against a target system. Firstly, the dumb host must be truly idle. No traffic may be received or sent by the dumb host while the scan is being performed. If the host is rather active and does talk to other hosts, its IP ID number will be constantly changed as a result and that may cause a lot of confusion, since the attacker won't be able to know whether the dumb host communicated with the target system or not. Secondly, the dumb

host must have a predictable IP ID number's generation. Some operating systems will simply increase the value in the IP ID field by one for each of the network packet sent, including any version of Microsoft Windows and older version of UNIX and Linux distributions. Newer versions of many UNIX and Linux operating systems are reported to handle the IP ID field differently; hence, immune to this type of scan.

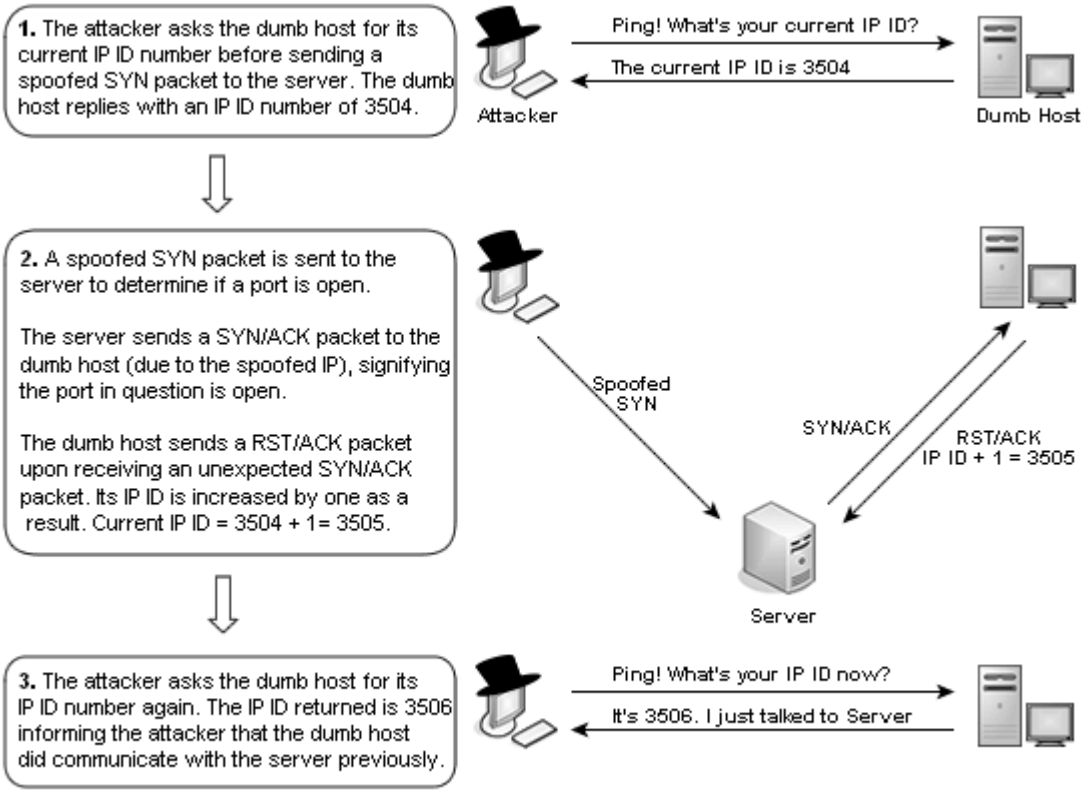

Figure 3-8. Idle Scanning (Open Port)

:.::.: STEALTH SCANNING

Although SYN scanning can be stealthy to some extent, it is still not stealthy enough given by the fact that most IDS and firewall systems nowadays are able to detect and stop SYN scanning without a hitch. You can be sure that not everyone is happy about this miserable issue (including me), and therefore, many different stealth scanning techniques have been proposed and developed over years, in an attempt to evade the not-so-hacker-friendly firewalls and system administrators.

From time to time, you may hear people refer to SYN scanning as stealth scanning for the level of stealth, anonymity, and evasion capability that it offers.

In order to stay stealthy, all the scanning techniques available under this category use a common mechanism to detect the closing or opening state of a port, popularly known as *inverse mapping*.

As oppose to the way open scanning and half-open scanning used to infer open port, which is based on the response from the remote system, inverse mapping — initially reported by CERT in 1998 — typically needs no response from the remote system at all. In fact, it is the absence of response from the remote system that helps inverse mapping knows if the port in question is open or not.

According to RFCs 793, a host must send a RST flagged packet to reset the connection if its requested port is closed or unavailable for incoming connection. Therefore, by sending crafted network packets with custom flag bits set to a range of desired ports on a remote host, an attacker can determine if the ports are closed based on the reception of the RST packet. Those packets that generate the RST flagged responses indicate that the scanned ports are closed. By knowing what ports are closed, the attacker can then elicit what ports are open. This also explains why this technique is so named inverse mapping.

In order to make inverse mapping work and trigger the host to send the RST packet, the attacker must build and send custom network packets that are not expected by the remote host at the start of a connection. For example, as you already know that a connection between two computers must always start with a typical three-way handshake in which one computer has to send a SYN packet to initiate the connection. However, if the one computer decides to send a packet with the SYN/ACK or FIN (Finish) flag bit set instead, the computer on the other end will think that the connection is bogus since no initial SYN packet was received, and as a result, immediately generate a packet with the RST flag bit set to reset the connection.

It is important to note that inverse mapping can also be used to elicit active and non-active hosts on the network. For this case, the attacker will have to infer and map the network by sending crafted RST packets to a range of IP addresses. The return of ICMP Host Unreachable message by the router signifies the host associating with the scanned IP address is not in use and vice versa.

Now that you know what inverse mapping is for and how it works, let's move on and examine some primary stealth scanning techniques that rely on inverse mapping to determine open ports.

SYN/ACK SCANNING

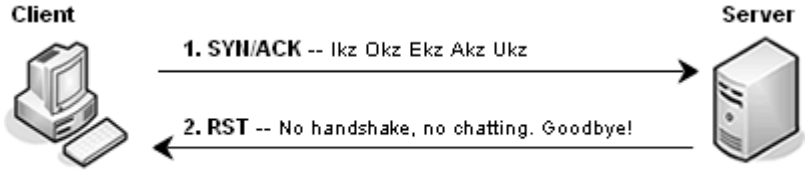

Figure 3-9. SYN/ACK Scanning (Closed Port)

By skipping the first step of half-open scanning, which is sending a SYN packet to a target system to setup a connection, this stealth scanning technique deliberately jumps straight to the second step and send a packet with the SYN/ACK flag bits set in an effort

to elicit a response from the target system. Figure 3-9 shows the result of SYN scanning against a closed port (note how a RST flagged packet is returned).

Figure 3-10 shows the result of SYN scanning against an open port. As you can see, the server ignores the packet sent to an open port and returns no response at all.

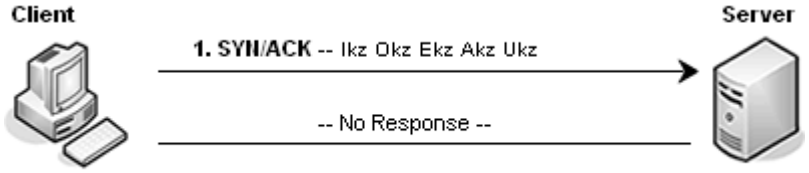

Figure 3-10. SYN/ACK Scanning (Open Port)

Even though SYN/ACK scanning can be very stealthy and fast since an attacker only needs to send one custom built TCP packet from his or her machine to the server to infer open port, this technique also has its own downsides. Firstly, SYN/ACK scanning, or any other scanning technique that resort to inverse mapping, is not as reliable as half-open scanning or TCP Connect() due to the volume of false positive results. No response from the server does not necessarily mean the port being scanned is open. It could also mean the SYN/ACK packet sent from the attacker's machine was never able to reach the server due to being filtered by sophisticated stateful firewalls or lost during the transmission. Secondly, custom TCP packets must be crafted for inverse mapping, and in order to do so, the attacker is required to have a root or super-user privilege, which may be, or may be not, an issue for the attacker.

FIN SCANNING

Because FIN (Finish) scanning is also another stealth scanning type that uses inverse mapping to infer a state of a port, it is very similar to the SYN/ACK scanning in concept. This technique sends a FIN flagged TCP packet to the target port. According to RFCs 793 (TCP specification), the target system will send back a RST packet if the port is closed and send nothing if the port is open. Some more of the stealth scans you are about to see below will all work and discover open ports stealthily in this manner. The only difference lies in how each of the stealth scanning techniques chooses to flag the TCP packets. Hence, to keep thing simple, I will only brief you quickly on each of these techniques. Figure 3-10 depicts how a server with closed port responds to FIN scanning.

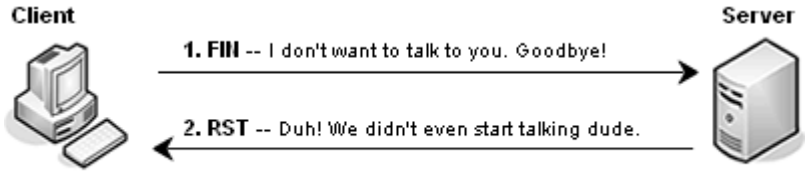

Figure 3-11. FIN Scanning (Closed Port)

XMAS SCANNING

Like FIN scanning, XMAS scanning attempts to send a packet with invalid flags set to a remote host at the beginning of a three-way handshake in the hope of eliciting some responses. However, XMAS scanning takes even a step further by sending a packet with the FIN, URG (Urgent), and PSH (Push) flag bits set simultaneously. The method used to infer open ports and closed ports is similar to what you have been introduced earlier, closed or non-listening ports return RST packets to reset the connection, whereas open ports just simply ignore the XMAS packets. Note, some available port scanners perform XMAS scanning by sending a probe packet with all available header flags turned on.

NULL SCANNING

Null scanning is often thought of as an inversion of XMAS scanning in that it sends to the target system crafted TCP packets with no flag bits set at all. Upon receiving the packets, closed ports on the target system will send RST packets to reset the connection while open ports just drop those invalid null packets and send nothing.

In sum, SYN/ACK, FIN, XMAS, and Null scanning are all very stealthy in nature and can pass through many IDS and firewall implementations undetected. However, using these scanning techniques in a busy network environment or against a security-conscious network may generate a high false positive rate since those crafted packets may be lost, filtered, ignored, or dropped during the transmission. All these scanning techniques can only be used against RFCs 793 compliant systems, such as UNIX and the like. Microsoft, however, pays a little or no attention to the RFCs 793 and implements the TCP/IP stack to their own standards, and for this very reason, Windows-based systems do not react to these stealth scans the same way UNIX-based systems do.

ACK SCANNING

In a commonly seen setup of a network environment, only the internal network is allowed to communicate to the outside — normally the Internet — and not vice versa, which means no Internet traffic is allowed to enter the internal network unless a connection is already established and originated from the internal network. Consequently, the packet filtering devices will drop any packet coming from the Internet that has the initial SYN flag bit set, since it signifies for a new connection setup request. Only those ACK flagged packets are allowed to get through as they are seen as responses to an established connection originated from the internal network. Therefore, crafted ACK packets sent by an attacker are more likely to be able to bypass some packet filtering devices as they indeed look like legitimate responses to a connection.

Originally addressed by Uriel Maimon in Phrack 49 article 15, ACK scanning is a stealthy technique that exploits a bug (or popularly referred to as a *feature* by some organizations) in the TCP/IP stack of BSD-derived operating systems. In essence, ACK scanning is of the same kind as SYN/ACK, FIN, XMAS, and Null scanning in that it relies on those RST packets returned from the remote system to determine whether the scanned ports are open are closed. However, determining a state of a port using ACK scanning is not as simple as merely waiting for the RST packet returned by the target system. In fact,

you have to analyze and look deeper into the RST packet header for inference. There are typically two ways for you to analyze the response RST flagged packet: one is to analyze the TCP TTL (Time-to-Live) field and the other is to analyze the TCP Window field.

To be technically correct, ACK scanning cannot explicitly reveal whether a port is open or closed, but it merely helps the attacker to deduce if a port is filtered or unfiltered by packet filtering devices. RST packets returned by the target system indicates unfiltered ports. ICMP Port Unreachable or no response returned by the target system indicates something is preventing the crafted ACK packets from reaching their destination, and hence, filtered. In order to ascertain if a port is indeed open, the attacker must make the decision based on several results of other scan types.

Analyzing the TCP TTL field

Ideally, if the value in the TTL field of one RST packet is smaller than the rest of the returned RST packets, it indicates the port in question is open. Likewise, those packets with the highest TTL value indicate the scanned ports are closed.

For example, if you send ten crafted ACK packets to the remote system and receive ten RST packets in return, it is only sufficient to say that the ports are unfiltered and whether they are open or listening is another story. Below is a log showing the details of the first four RST packets returned from the remote system (using `hping2` and you will be shown how to use this nifty utility very soon):

```
1: host 192.168.0.69 port 79: F:RST -> ttl: 66 win: o
2: host 192.168.0.69 port 80: F:RST -> ttl: 40 win: o
3: host 192.168.0.69 port 81: F:RST -> ttl: 66 win: o
4: host 192.168.0.69 port 82: F:RST -> ttl: 66 win: o
```

When analyzing all these four packets, you will probably notice that the RST packet returned from port 80 (the second row) has the smallest TTL value — which is 40 — compared to the other three RST packets. This gives you the indication that port 80 on the remote system 192.168.0.69 is unfiltered and listening.

Analyzing the TCP Window field

This technique is also generally known as *Window scanning*. It is to say that certain operating systems, such as BSD and HP-UX, report the TCP Window size differently, which may help an attacker infer open ports in the event where analyzing the TTL field wasn't of any help. The following shows yet another example:

```
1: host 192.168.0.16 port 23: F:RST -> ttl: 64 win: o
2: host 192.168.0.16 port 24: F:RST -> ttl: 64 win: o
3: host 192.168.0.16 port 25: F:RST -> ttl: 64 win: 512
4: host 192.168.0.16 port 26: F:RST -> ttl: 64 win: o
```

As you can see, depending on the TTL value to verify the state of a port as in this

case is not a good idea at all, since all the four RST packets received share the same TTL value of 64. However, unlike the other RST packets that have a zero TCP Window size, the third RST packet with the TCP Window size of 64 (non-zero) indicates the port in question is listening.

ACK scanning is, unquestionably, a very stealthy technique that can help the attacker discover open ports on the target system effectively, without the need of confronting the host-based IDS and firewall system. However, the limitation of ACK scanning is that this technique resorts to a bug in the TCP/IP stack of some UNIX and BSD-derived operating systems to work, and hence, many other different operating systems are immune to this type of probing, including Microsoft Windows family. Moreover, most of the modern BSD and UNIX operating systems nowadays are patched to fix this issue and don't be upset if ACK scanning just won't work out for you.

Nonetheless, ACK scanning is not all useless as it might seem. In fact, it is still very widely used by hackers all around the world for mapping out firewall rulesets and identifying the type of firewall used by the target network. As you may recall, if a crafted ACK packet is able to reach its target system, an attacker will receive a RST packet in return, indicating the scanned port is unfiltered. Unfiltered usually indicates the packet-filtering device used by the target network is a stateless firewall, which does not keep track of the connection and is only able to stop unsolicited SYN packets from setting up a new connection with the internal systems. Stateless firewall is not capable of differentiating between a legitimate ACK packet from an valid connection and a crafted ACK packet from an attacker.

No response or ICMP Port Unreachable returned by the target system indicates a filtered port. Filtered port implies the crafted ACK packet can't make it way through a stateful firewall, which is a device that does keep track of the connection and capable of identifying and blocking unsolicited ACK packet.

:.::.: SWEEPING

Hackers may also use port scanning as an alternative technique to discover active hosts should ICMP traffic is completely blocked by the border routers of the protected network. The hackers can send TCP or UDP packets to probe a network for active hosts because most routers, if not all, do have to allow TCP and UDP traffic flowing in and out of a network. One of the leading examples of TCP network sweeping is inverse mapping. The hackers send specially crafted RST packets or DNS response packets to a range of IP addresses and determine the accessibility of a host based on the ICMP Host Unreachable message. Non-active hosts will trigger the router to send ICMP Host Unreachable messages to *inform* the hackers. By knowing which hosts are not active, the hackers can infer those that are active. It is recommended that you utilize both ICMP and TCP or UDP sweeping techniques for discovering active hosts on the network, since relying solely on either one of them can generate false positive results.

:.::.: MISCELLANEOUS

Miscellaneous section covers techniques that are not essentially related to discovering open ports or accessible systems on a network; instead, it addresses some popular techniques that can be used to facilitate port scanning and related activities to occur with a greater impact.

DECOY SCANNING

The purpose of decoy scanning is to misguide the target and mask the identity of the attacker. This is made possible when the attacker sends a probe packet in a jumble with other packets that have been spoofed with many different IP addresses. For example, when ten packets are sent to a port on a remote system, only one packet actually contains the real IP address of the attacker, while the others were all spoofed with different IP addresses in an effort to cause misleading. It is important that the attacker must provide his or her real IP address so that the remote system knows where to send the response to. The remote system will of course respond to the other nine decoy packets the same way as it would with the attacker's probe packet. The more decoys sent to the remote systems, the better, because the network administrators or law enforcement officials would have to manually verify and trace back to the source of each of the probe packets — including the decoys — in order to identify where the attack really originated.

TCP FRAGMENTING

As IDS and firewall systems are being more sophisticated with advanced rulesets that can detect and block various types of attacks, including scanning, sending crafted TCP packets in the hope of stealthily mapping out the target system, or the target network as a whole, is no longer an easy task. As usual, given the level of creativity and desires to gain access to a network, hackers of course do not give up so easily as they yet introduce a very effective solution that can overcome such obstacle, *TCP fragmenting*.

In its truest sense, TCP fragmenting is not a new scanning technique that can help the hackers find open ports on a remote system. In fact, it is used to facilitate stealth scanning by helping the probe TCP packets avoid vigorous IDS or packet filtering devices and reach their destination. The main idea behind this is to split a TCP packet header into several smaller fragments, hoping the IDS or firewall systems cannot examine and detect anything anomaly in each of those packet fragments. Once sneak past the firewall, those small packet fragments will start to reassemble into its complete form, an evil crafted TCP packet, and head to the destination.

This technique can be very effective in delivering notorious probe TCP packets past a firewall because the firewall won't be able to anticipate the bigger picture and ultimate purpose of those fragments by examining each of them individually. Do note that some firewalls do queue all the packet fragments before processing them; however, doing so will cause delays and add overheads to the network, which might be something that is not truly desirable by an organization. Another point worth noting is that many systems can't handle tiny packet fragments, and therefore, might be malfunctioned, crashed, or rebooted as a result.

SLOW SCANNING

Slow scanning is not a scanning technique so to speak. It's just a simple trick where an attacker tries to send scan packets to a network sparsely, over a long period of time, to avoid being logged and detected by the target network. This tactic arises to deal with the fact that sending so many probe packets to the network in such a short amount of time can generate a lot of noises, which in turn, may cause an alarm to the network administrators. By sending only one or two probe packet(s) to the network a day, the attacker can still map the entire network eventually, while at the same time, remains undetected for a long time. Yet, the biggest disadvantage with slow scanning is that it is not for everyone, especially not for those who are impatience and don't have enough time to spare.

RPC SCANNING

RPC (Remote Procedure Calls) services were once consider as the sweetest and finest unauthorized entrance points to a network due to the countless serious security vulnerabilities that lie underneath. Even though many network administrators are now aware of the problems and proactively patch the system against such attack, RPC services are still the number one scanning target. There are typically two ways to find running RPC services. First is to consult with the `portmapper` or `rpcbind`, which is listening on port 111, about translating RPC program numbers to port numbers. This method is getting far less common since many firewalls nowadays are set to prevent Internet traffic from flowing into this notorious port. The second method is to scan for RPC programs directly by sending an empty RPC command to each of those discovered TCP and UDP ports.

IDENT SCANNING

Ident scanning is not a port scanning technique in and of itself, but it takes advantage of the `identd` (an Identification Protocol as addressed in RFCs 1413) network daemon to help hackers find out the username of the owner of a listening TCP port. To obtain such information, the hackers need to establish an actual full TCP connection with the target listening TCP port and submit the relevant TCP port number pair to `identd`, which is normally listening on port 113. Once received the query and making sure the provided TCP port number pair is valid, `identd` will reply to the hackers with a character string that identifies the owner of the respective TCP connection.

The hackers normally use this method to find which TCP service is owned by root or high-privilege users so that they can focus and direct their attack toward that specific service, since obtaining root or super-user privilege is the most desired objective in any hack attack. However, some of the hackers usually decide to gain unauthorized remote access to the server first, irrespective of under which privilege, and will worry about how they can escalade to the root privilege later, which is not a really hard task given the huge amount of local security vulnerabilities existed in many operating systems today.

It is worth noting that there is no such thing called stealth or anonymity in ident scanning because a full connection must always be established between the hackers and the

server in order for ident scanning to work at all. Hence, the shortcoming of this technique is that it can be very easily detected and filtered by any decent firewall. Besides, it can only be used against Linux, UNIX and related system and therefore the scope of this type of scanning is somewhat limited.

FTP BOUNCE SCANNING

In the same way as decoy scanning and idle scanning, FTP bounce scanning is yet another creative technique that can help the attacker conceal the true source of an attack. By exploiting an inherent feature, right now it is a serious security vulnerability, in the FTP protocol, the attacker can use an innocent FTP server as a proxy for performing all the scanning against the desired system and delivering the results. This technique can be somewhat related to idle scanning in which an attacker maps a particular network with the help of a third-party machine and uses such machine as a scapegoat should the scans be revealed afterward by the target network.

After the attacker logged in to the vulnerable FTP server, the attacker will then issue the PORT command to specify the IP address and port which the FTP server should be connecting to next. If the specified port is closed, the FTP server will generate an error message and inform the attacker that it couldn't establish the connection to the specified system. If the specified port on the target system is open, the FTP server will also generate an error message, notifying the attacker that the connection is open, but of no use, as the protocols between the FTP server and the specified system are mismatched. The error messages provided by the FTP server are clearly very informational and the attacker can easily infer a state of a port without having to do much work.

The attacker also relies on FTP bounce scanning to bypass a certain type of packet filtering devices, and thereby, gain insightful information about a network. This happens when the vulnerable FTP server is trusted by the target network, or belongs to the same subnet as the target system. By initiating a new connection from the FTP server to the target system, the attacker effectively tricks the target system into thinking that the connection is originated from the FTP server, a trusted machine in the same subnet, and not from somewhere else. The attacker from there can choose to load and send arbitrary files to the system and port in question from the trusted FTP server without the fear of being detected and blocked by the horrific and ugly firewalls.

In all, FTP bounce scanning is great way to sneak past firewalls as well as to scan a network stealthily. However, it should be noted that FTP bounce scanning is quite well addressed by many FTP servers that you might come across today, due to all the security issues it induces. Still, that doesn't mean you won't be able to find one FTP server that supports bouncing feature at all as people of the "warez" scene are actually trading this kind of stuff everyday.

UDP SCANNING

Thus far, you have only seen how one goes about finding open TCP network ports on

a remote system, but you haven't really seen what it takes to scan for open UDP ports, which is relatively easy anyway.

If the hackers and their port scanners can determine the state of a port based on the flag bits set in the header of a TCP packet, then with UDP scanning, they can only make their inference based on the return of an error message. The hackers first send an empty or a zero byte UDP datagram to each UDP port on the target system. If the port is open, the probe UDP datagram will simply be discarded and return no response. However, for each closed port, the target system will generate an ICMP Port Unreachable message and send it back to notify the hackers. By going through the process of exclusion and ruling out closed UDP ports, the hackers can infer which UDP ports are open. This inverted technique is very similar to inverse mapping if some of you still can recall.

UDP scanning is good for uncovering network services that are listening on undocumented UDP ports on a remote system, yet it tends to generate more false positive results than any other type of scanning because the protocol is connectionless and does not have anything to assure or guarantee the delivery of each UDP datagram. In a busy network environment, it is very common to loose a few UDP datagram during the transmission.

WOO HOO — IT'S A WRAP

Congratulations! You made it. You have just finished examining many different yet important aspects of port scanning theoretically, which is no doubt a very daunting and intensive study. Next section will take you through some popular port scanning tools in the field so that you can apply what you have just learnt into the real world. Remember, do not attempt to scan any network without obtaining prior permission in written form.

BLASTER SCAN & A BUNCH OF OTHER OUTDATED PORT SCANNERS

Written by Polos of the Ezkracho Team, Blaster Scan is not only a TCP port scanner itself but it also incorporates many other scanning and reconnaissance techniques into one package, including ping sweeping, CGI scanning, OS fingerprinting, users enumeration, and even FTP and POP3 password brute forcing.

After installing Blaster Scan by typing `./install.sh` at the Linux command prompt, you can view all the available scanning options by executing `./blaster` or `man blaster` for more information. It's sad but I don't really remember when the last time I see people playing with `blaster` since it is relatively old and now replaced by tons of other sophisticated port scanners in the field. Hence, as you probably already noticed, I include Blaster Scan and some other outdated port scanners that you are about to see in this section only to follow the official C|EH course outlines. Using these port scanners is extremely irrelevant and highly deprecated for today's pace. I even have a hard time finding these tools myself, let alone using them.

PortScan Plus is rather a primitive and basic TCP port scanner for Windows. It pro-

vides a clean interface with a few options that allow you to enter a range of IP addresses which you need to scan and the number of connections to be made at the same time. This tool is nonetheless too old and basic; you should consider using Super Scan — a fine tool from Foundstone — as a replacement.

IdentTCPScan, initially released by Dave Goldsmith in 1996, does exactly what it promises, performs ident scanning to find out the username of the owner of a particular running TCP process. This ident focused port scanner is very compact, fast, and powerful because it requires only a small amount of computer resources to scan. Moreover, it is known to be compatible with many platforms, including Linux, UNIX, and SunOS. Many versions of `nmap` support ident scanning capability so probably you will not be able to see many people using this tool anymore. But if you still want to take a good look at it, you can still grab it here http://www.packetstormsecurity.org/UNIX/scanners/ident-scan.c

FLOPPY SCAN

FloppyScan might give you the impression that it will scan a floppy disk to look for virus or malware, but in fact, it only does something in similar effects. FloppyScan gives you the luxury to initiate port scanning on any computer from a floppy disk and have the results emailed to your specified email address automatically. Hence, you can actually go and enjoy your cappuccino without having to wait for the scan to finish. The concept is pretty original but the underlying tool to help perform port scanning is no stranger to you, it is the renowned `nmap`. FloppyScan relies solely on `nmap` to perform SYN scanning and OS fingerprinting. In addition, FloppyScan is also capable of performing NetBios scanning against Windows systems with the help of `nbtscan`. FloppyScan is freely available for download at http://www.tgs-security.com/downloads/floppyscan-dist-0.zip

You will need to create a bootable floppy disk from the image file that comes with FloppyScan, dosdisc.img. If you're using Windows, you can use rawwrite to perform the task. It is downloadable at http://uranus.it.swin.edu.au/~jn/linux/rawwritewin-0.7.zip. For Linux platform, you can use the built-in `dd` command like the following:

```
[bash]# dd if=/dosdisc.img of=/dev/fd0 bs=1440k
```

After writing the image file to the floppy disk, you will need to change the config.cfg on the disk first before attempting to boot the computer and start FloppyScan. The two most important changes that you need to make are the email address, which you want to have the scan results sent to, and the mail server which FloppyScan will use to send the results. Without supplying the correct information, you can rest assure that you will enjoy your fine cappuccino all day and not receive any results at all.

Once everything is all done, it's time to start the show, booting the computer from the floppy disk. FloppyScan, optionally and by default, displays a Blue Screen of Death after the boot is done (just for fun I guess). From here, you have two options. You can either choose to walk away and pretend like nothing has ever happened while FloppyScan is doing its job or you can also choose to sit there and watch how it processes. To catch all

the live actions and see what FloppyScan is doing, you can switch the view by pressing `Alt+F4`. Pressing `Alt+F3` or `Alt+F2` will take you to a basic Linux shell prompt.

"Hey, wait a minute, I didn't even tell FloppyScan what to scan yet so what the heck is it scanning after boot up anyway?" Well, FloppyScan doesn't want you to tell it what to do. It will use the IP address of the computer in question as a starting point and then scan the entire class C subnet of which the computer resides. In more simple terms, it will scan all 255 hosts that have the same network address as the computer performing the scan. The IP address of the computer that starts FloppyScan can be manually defined in the config.cfg file or queried from the DHCP server.

In sum, FloppyScan is useful when you don't have the time and chances to conduct port scanning and wait for the results. In an unethical scenario, imagine an opportunist is given only one opportunity to visit the target organization network, wouldn't it be nice to insert the FloppyScan disk into a local computer there and have it do all the works while he or she can talk to the director and play the "I wasn't there and I'm innocent" game?

STROBE

If you don't care much about being anonymous and just want to scan a network as fast as possible, then `strobe` is just the right tool for the job. Written by Julian Assange, `strobe` aims to be one of the fastest and most compact TCP port scanners for Linux by using the TCP connect() scanning method to probe for open ports on the target system. Because a full connection is made to each port, `strobe` yields reliable results. `Strobe` version 1.04 to 1.06 (latest) offers a feature called banner grabbing along with port scanning. By grabbing greetings or banners from those open ports, `strobe` gives you more chances to determine the remote network services and OS more accurately.

Running `strobe` can be very simple. All you need to do is appending an IP address or a domain name of a host to `strobe` on the command line. If you want to scan multiple hosts all at once, you can append multiple IP addresses and domain names separated by space. Below is an example of using `strobe` to scan a single host:

```
[bash]# strobe www.gotrice.com
strobe 1.05 (c) 1995-1999 Julian Assange <proff@iq.org>.
www.gotrice.com    25 smtp  Simple Mail Transfer [102,JBP]
                        -> 220 I don't like SPAM for breakfast ESMTP\r\n
www.gotrice.com   110 pop3  Post Office Protocol - Version 3 [122,MTR]
                        -> +OK <29624.1109775440@domain.com>\r\n
www.gotrice.com    22 ssh   Secure Shell - RSA encrypted rsh
                        -> SSH-2.0-SSH\r\n
www.gotrice.com    80 http  www www-http World Wide Web HTTP
                        www   World Wide Web HTTP [TXL]
```

Note, `strobe` is only designed to scan for listening TCP ports and nothing else. If you want to have a complete view of network, you can't just rely on this compact TCP port scanner alone. Another point worth noting is that `strobe` tends to make a lot of noises during scanning as it has to establish a full connection to each open port. Thus, you may

want to look for another tool if you want to stay stealthy and avoid the remote firewall. Refer to Exhibit 3-3 on page 18 the download location of the tool.

HPING2

Hping2, by Salvatore Sanfilippo, is a sophisticated TCP/IP ping utility that allows the users to construct and send custom TCP/UDP/ICMP packets for many different purposes. Typically, hping2 is used for stress testing firewall rules and effectiveness, fingerprinting remote OS, discovering open ports, auditing the TCP/IP stack, creating covert channels, effecting various types of DoS attacks, and so forth. Since hping2 is a powerful and low-level network packets generator, it is recommended that you are familiar with specifications of the TCP/IP and other important protocols.

The first notable feature of hping2 is port scanning. Although the author originally didn't have the intention to build hping2 to be a sophisticated port scanner like nmap, hping2 can act as a port scanner itself and supports many different stealth scanning techniques. In order to take full advantage of hping2, such as generating and sending custom TCP packets, it is important to note that you must login as a root user.

```
[bash]# hping2 –-scan 79-83 www.scan-me.com
using eth0, addr: 192.168.0.12, MTU: 1500
   79 finger     : ..R.A...  42     0     0
   82 xfer       : ..R.A...  42     0     0
   81 hosts2-ns  : ..R.A...  42     0     0
   83 mit-ml-dev : ..R.A...  42     0     0
   using eth0, addr: 192.168.0.12, MTU: 1500
Not responding ports: (80 http)
```

In this example, hping2 scans the specified host www.scan-me.com to see if any port number ranging from 79 to 83 is open. The --scan option takes hping2 into port scanning mode. By default, hping2 perform null scanning, which is sending packets with no flags turned on at all, to avoid detection by the target system. Hence, based on the results above, you can easily deduce only port 80 is open on the target system because it returns no response to the bogus and invalid TCP packet sent by hping2. Those ports that return RST/ACK packets are closed ports, according to RFCs 793.

Even though null flag packets are sent by hping2 by default, that still doesn't mean you have to stay with it. You can have hping2 send packets with other flag too. Here is an alternative way of scanning a target, without the --scan option and null flag packets:

```
[bash]# hping2 -X www.scan-me.com –p ++20
HPING www.scan-me.com (eth0 192.168.0.12): X set, 40 headers + 0
len=46 ip=192.168.0.12 ttl=42 DF id=0 sport=20 flags=RA seq=0 win=0
len=46 ip=192.168.0.12 ttl=42 DF id=0 sport=23 flags=RA seq=3 win=0
len=46 ip=192.168.0.12 ttl=42 DF id=0 sport=24 flags=RA seq=4 win=0

---results truncated for brevity---

len=46 ip=192.168.0.12 ttl=42 DF id=0 sport=79 flags=RA seq=0 win=0
len=46 ip=192.168.0.12 ttl=42 DF id=0 sport=81 flags=RA seq=2 win=0
```

```
len=46 ip=192.168.0.12 ttl=42 DF id=0 sport=82 flags=RA seq=3 win=0
```

The two new options introduced in the above scan are the `-X` and `–p`. The `-X` option tells `hping2` to perform Xmas scanning instead of null scanning, whereas `–p` specifies the port which `hping2` should be sending the probe packet too. Notice the `++` standing before port 20, it tells `hping2` to increase the port number by one by each packet sent. For Xmas scanning, you use the same inverted mapping technique as null scanning to infer a state of a port. According to the results, only port 21, 22, and 80 do not reply to the Xmas scans, which may indicate they are open and listening for incoming connection.

As promised, I will show you how you can use `hping2` to perform idle scanning. Just in case some of you might don't know, idle scanning was originally discovered by the same author of `hping2`; hence, needless to say, the tool is just the perfect one for the job.

First, you will need to obtain the current IP ID number of the dumb host (I assume you already found one):

```
[bash]# hping2 -X www.i-am-dumb.com –c 1
HPING www.i-am-dumb.com (eth0 192.168.0.20): X set, 40 headers + 0
len=46 ip=192.168.0.20 ttl=102 id=43150 sport=0 flags=RA seq=0 win=0
```

The `–c 1` tells `hping2` to send only one probe packet, since that is all it takes to obtain the IP ID number. The value in the IP ID field of the returned packet is 43150, you need to keep this in mind before advancing to the next step — sending a spoofed packet to the target system.

```
[bash]# hping2 –a www.i-am-dumb.com –S www.scan-me.com –p 80 –c 1
```

You will not be able to receive any reply from www.scan-me.com since you spoofed the packet with the source address of the dumb host using the `–a` command option. Now, to see whether port 80 on www.scan-me.com is open, you need to check the dumb host for its current IP ID number again.

```
[bash]# hping2 -X www.i-am-dumb.com –c 1
HPING www.i-am-dumb.com (eth0 192.168.0.20): X set, 40 headers + 0
len=46 ip=192.168.0.20 ttl=102 id=43152 sport=0 flags=RA seq=0 win=0
```

If port 80 is open, the dumb host will receive an unexpected SYN/ACK packet originated from target system, and because of which, it will have to increase the IP ID number by one to send a RST packet to reset the erroneous connection. If port 80 is closed, the target system will send a RST/ACK packet, which the dumb host will do nothing upon receiving such packet and its IP ID number stays the same. From there, you can see the IP ID number returned by the dumb host this time, 43152, is increased by two compared to the last one, 43150, which may give an indication that the dumb host did send a RST packet to the target system during the interval. Voila! Port 80 on the www.scan-me.com system is open folks.

`Hping2` can also be used for testing firewall rules or simply identifying the type of the

remote packet-filtering device. The following examples show you how to differentiate between a stateless and stateful packet-filtering device using hping2:

```
[bash]# hping2 -A www.microsoft.com –p 80
HPING www.microsoft.com (eth0 207.46.156.156): A set, 40 headers + 0
len=46 ip=207.46.156.156 ttl=109 id=62564 sport=80 flags=R seq=0 win=0
len=46 ip=207.46.156.156 ttl=109 id=44237 sport=80 flags=R seq=1 win=0
```

You can see the server www.microsoft.com replies to the crafted ACK packets very well, regardless of whether they are part of a valid connection or not. This indicates a stateless firewall is in place and port 80 is unfiltered. A stateless firewall cannot tell the difference between an unsolicited ACK packet and a legitimate ACK packet.

```
[bash]# hping2 -A www.cisco.com –p 80
HPING www.cisco.com (eth0 198.133.219.25): A set, 40 headers + 0
```

On the other hand, the server www.cisco.com reacts differently to the same scanning method used against Microsoft server previously. The ACK packets crafted and sent by hping2 elicit no response from the server at all. This may give the indication that a stateful firewall may have checked and dropped the ACK packets due to unmatched entries in its connection table. Oh well, at least now you know Cisco PIX Firewall isn't worth more than $10.000USD for nothing.

Hping2 can do much more than what you have seen earlier but it would well occupy 20 pages if I shown you all possible options and usage of hping2. Should you have the need to learn more about this neat packet generator, please consult with the official man page accordingly, or you can also visit http://wiki.hping.org for many other related tutorials. Note, hping2 is only available for Linux, UNIX platform and the like.

NMAP

As you have seen previously, nmap can be used to ping sweep an entire IP address range of a network to discover active hosts at a lightening speed. However, that certainly is not the only reason that makes nmap become what it is today. Packed with many scanning options that you would ever want to see in a port scanner, nmap allows you to manipulate and modify your scan to whatever you like best in order to gain optimum results. Furthermore, nmap is also one of a very few port scanners that incorporates almost all port scanning techniques available, including all those that you examined previously. It is worth bringing to your attention that Fyodor has rewritten and made some dramatic changes to the core scan engine of nmap, particularly from version 3.70 to current, to improve scanning speed and provide the users with many more scanning options and possibilities. It is recommended that you download and use the newer versions instead of those that come by default with your UNIX and Linux distributions. Nmap is also available for Windows platform. Refer to Exhibit 3-3 page 18 for download location.

You can conduct your nmap scans via the powerful command line or the intuitive GUI interface. Accurate remote OS and software version detection is also a unique feature of

nmap which you can hardly find in any other non-commercial security tool. Since remote OS detection is a big topic, it deserves its own space and will be discussed more thorough in the OS Fingerprinting section. For now, let's examine some of the popular features of nmap that may facilitate and be useful for your profession.

```
[bash]# nmap -sS www.scan-me.com

Starting nmap 3.81 ( http://www.insecure.org/nmap/ ) at 2005-02-11
Interesting ports on www.scan-me.com (192.168.0.12):
(The 1647 ports scanned but not shown below are in state: closed)
PORT     STATE    SERVICE
21/tcp   open     ftp
22/tcp   open     ssh
25/tcp   open     smtp
53/tcp   open     domain
80/tcp   open     http
110/tcp  open     pop3
3306/tcp open     mysql
```

The example just shows a quick and simple way to get nmap running. You can see nmap isn't asked to do anything fancy here, except for performing a stealth SYN scan -sS against the host www.scan-me.com. You can easily change from SYN scanning to any other stealth scanning type by replacing the -sS accordingly, such as -sF or -sN for FIN scanning or Null scanning respectively. However, do note that in order to take full advantage of the program, such as performing stealth scanning, you must be able to run nmap as root, though to some people it's a security risk to run any program using root privilege.

Of course nmap isn't just limited to executing one scanning option or type at a time, you can indeed have nmap to handle various tasks concurrently by combining and including all your desired scanning options on one single command line. In the following example, you will see how nmap copes up with a rather intensive and complex port scanning:

```
[bash]# nmap -P0 -sSUV 203.162.0.0/24 -O -oN my-fish
```

The -P0 option is used to specify nmap not to ping sweep the target network before port scanning. The -sS as you already know, instructs nmap perform TCP SYN scanning type. However, scanning for TCP ports is still not good enough since UDP ports on the target network may be listening too, and hence, the -sU is necessary. The –sV instructs nmap to detect version of the network software listening on each open port on the target network. According the example, the target network which nmap should point the gun at is 203.162.0.0/24. Instead of defining an IP address range using the mask notation as in this case, you can optionally use 203.162.0.* or 203.162.0.0-255 and it would still mean the same. The -0 option specifies nmap to detect the remote OS of each scanned system on the network. Finally, the -oN option used to ask nmap to save all the scanned results into a log file named my-fish.

Instead of having to repeatedly type -s to combine different scanning types, you can save time by typing -s only once. Nonetheless, it is important to note that the combination of your scanning types must make sense. In other words, you

can't just ask nmap to perform two different TCP scanning types at the same time, for instance, you can ask nmap perform either Xmas scanning or null scanning at a time but not both.

Nmap also supports idle scanning just like hping2, but with much less hassle. All you need to do is to provide nmap with the IP address of the dumb host and the host you wish to scan and nmap will do all the scanning and IP ID number monitoring itself:

```
[bash]# nmap -P0 –sI 192.168.0.20 www.scan-me.com

Starting nmap 3.81 ( http://www.insecure.org/nmap/ ) at 2005-02-11
Idlescan using zombie 192.168.0.20; Class: Incremental
Interesting ports on 192.168.0.12:
(The 1640 ports scanned but not shown below are in state: closed)
PORT     STATE    SERVICE
21/tcp   open     ftp
22/tcp   open     ssh
25/tcp   open     smtp
53/tcp   open     domain
---results truncated---
```

The –sI option tells nmap to start an idle scanning. Following the –sI should be the address of the dumb host and the host you wish to scan, as in this case, 192.168.0.20 and 192.168.0.12 respectively. It can't be emphasized enough that if you want to stay truly stealthy, you must specify nmap not to ping the remote host before scanning. Sending a ping packet to the remote host before scanning will definitely give you and your IP address away to the remote IDS or security safeguards and make idle scanning less effective or meaningless even.

Decoy scanning with nmap can't be any easier. No longer must you craft spoofed TCP packets and send them one by one to the remote system manually. Instead, you would be better off giving nmap the IP addresses of the decoy hosts and the remote system, and needless to say, the rest is history.

```
[bash]# nmap -P0 -sS -p 21-110 -D 192.168.0.20 192.168.7.22

Starting nmap 3.81 ( http://www.insecure.org/nmap/ ) at 2005-02-11
Interesting ports on 192.168.7.22:
(The 88 ports scanned but not shown below are in state: filtered)
PORT     STATE    SERVICE
21/tcp   open     ftp
22/tcp   open     ssh
25/tcp   open     smtp
80/tcp   open     http
```

Yet again, you have to use the –P0 to prevent nmap from pinging the remote system before scanning. Having your real IP address appeared in the logs of the remote system is bad enough but it's even worse to have it appeared more than those decoys' IP addresses, many thanks to the extra ping packet. So if you are serious about concealing your identity and being anonymous, under no circumstances should you let that ping packet slipping by and drawing all the attention to itself. Continue with the preceding example, the -p option tells nmap to scan the defined port range, from 21 to 110. The -D option enables decoy

scanning mode, followed by the IP address of the decoy host 192.168.0.20. Practically, you are allowed to use as many decoy hosts as you want and not limited to just one host as above. Having said that, 10 decoy hosts are normally good enough since having too many decoy hosts for just one scan can also be a disadvantage in which the time taken to complete the scan may be substantially increased and the results may be far less accurate.

Since the core of nmap has been rewritten, nmap version 3.70 and newer no longer support ident scanning; hence, if you want to perform this particular type of scanning you should either use nmap 3.50 and older, or stick with IdentTCPscan as introduced earlier.

```
[bash]# nmap -I 192.168.7.22

Starting nmap 3.50 ( http://www.insecure.org/nmap/ ) at 2004-08-11
Interesting ports on 192.168.7.22:
(The 1642 ports scanned but not shown below are in state: closed)
PORT     STATE    SERVICE  OWNER
21/tcp   open     ftp      root
22/tcp   open     ssh      root
25/tcp   open     smtp     root
80/tcp   open     http     nobody
```

If you don't want to sacrifice the speed and power offered by the newer core of nmap just for the one idle scanning option, you can always do this:

```
[bash]# ./ident-scan 192.168.0.2
Port:  21 Service:       ftp  Userid: root
Port:  80 Service:       http Userid: nobody
Port: 113 Service:       auth Userid: nobody
```

Of course, there are many other finer aspects of nmap but I just can't simply squeeze them all into these tiny space. When in doubts, you should look up the nmap manual or visit its official website at http://www.insecure.org/nmap/nmap_documentation.html for links to many nmap related tutorials. What I showed you earlier by no means could reflect the true power of this port scanner. You can only feel it by actually playing around with nmap more often, and you should, because it is very vital to your everyday security tasks.

SUPERSCAN

SuperScan is a fine multithreaded Windows port scanner from Foundstone, Inc. From version 3 and backward, SuperScan is a rather simple port scanner that will try to connect to each scanned port using the three-way handshake in an effort to gather the most accurate and reliable results. The users can optionally set the timeout for ping and connection attempts and adjust the scanning speed to conform to their Internet connection. The interface is very intuitive and easy to manage given by the fact that the program doesn't offer the users a lot of scanning options like nmap, which is a good thing in a way since the users then do not have to spend a lot of time to learn using the tool.

As advanced users require more from SuperScan, Foundstone crews rewrote the tool completely and added loads of features to the updated version, version 4 (as shown in

Figure 3-12). Some of the notable features that have been improved or added to the tool include but not necessarily limited to the following:

- Support different scanning types: TCP connect(), SYN scanning, UDP scanning
- Multiple ICMP ping sweeping methods for the discovery of active host
- Various techniques to enumerate Windows-based systems
- Facilitate network discovery with utilities like `traceroute`, `ping`, and `whois`
- Simple IDS and firewall evasion capability with the source port scanning option
- Display scan results in a HTML report
- More accurate service detection with improved banner grabbing technique
- Fast scanning and hostname resolving speed

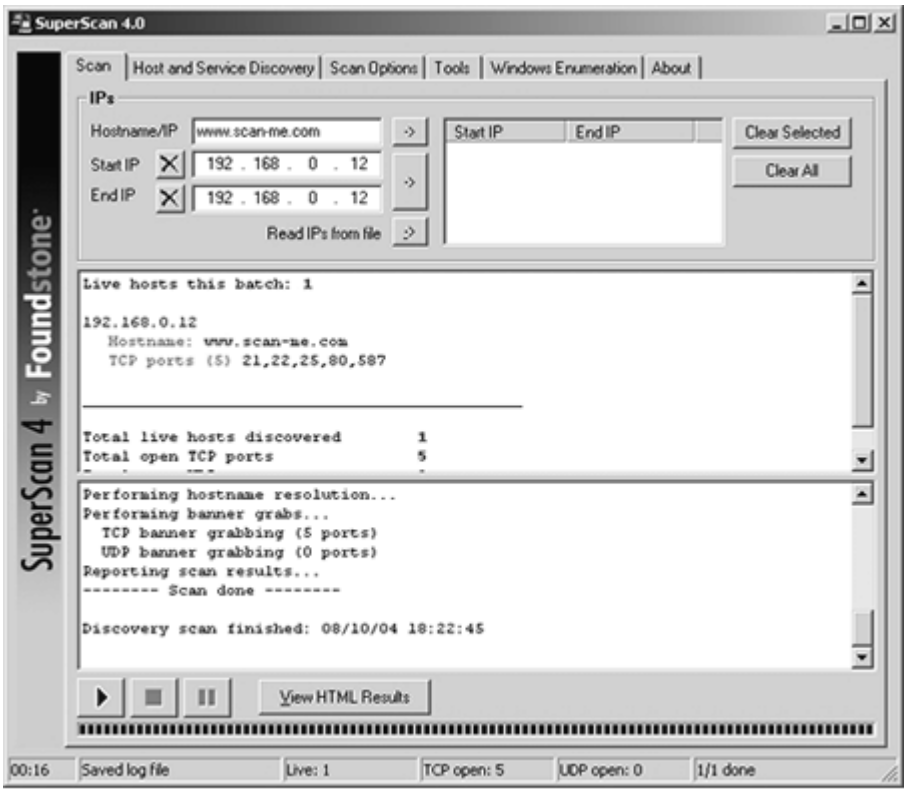

Figure 3-12. SuperScan 4.0

Scanning with SuperScan 4 is a really stroll in the park — nice and easy. If you're an impatient type of user, you don't even need to tweak or change anything to get the tool work. All you need to do is to enter the domain name or IP address of host you wish to scan into the provided box and press the play sign located on the bottom of the `Scan` tab window, as can be seen on Figure 3-12. The tool will then use all of its default options and start scanning the target with no question asked. Once the scan is done, you can either view the results displayed directly from the tool, which is rather primitive, or you can alternatively click on the View HTML Results button for a pretty and detailed report. If you are an advanced user and want to change a few things before the fun begins, you can click on other tab windows, particularly the `Scan Options` tab, for more options.

While SuperScan version 3 is a perfect tool for novice users or hackers who do not want to worry about anything else than finding open ports and hacking the heck out of the target systems, SuperScan 4 is usually more favorable for the more serious and advanced users. SuperScan 4 can only be run on Windows 2000 or XP by users belonging to the Administrator group.

NETSCAN TOOLS PRO

NetScanTools Pro, by Northwest Performance Software, Inc., is a sophisticated information gathering and network discovery tool for Windows platform. Packed with more than 30 different Internet and information gathering utilities into a single visual package, the tool aims to provide the users an all-in-one network discovery solution. You can just buy this single product and will not have to worry about missing or finding any other essential discovery utilities ever again. However, the sad thing is, you won't be able to download this tool for free as it comes with a price of $249. Trial or demo version of this tool is also not available to the public, especially home users. Having said that, if you're a business or government person, you are entitled to request for a demo version, but each request will be manually verified. I guess this is perhaps one of the steps that the people at Northwest Performance Software try to take to prevent software piracy and the horrific crackers — those who attempt to subvert software protection.

Listing and explaining all the essential features or tools that come with NetScanTools Pro is beyond the scope of this book. The list below only provides you with short descriptions of some popular and standout reconnaissance related features of the tool so that you can have a general idea of how NetScanTools Pro can benefit your hacking tasks:

- Enhanced `ping` – Support sending standard or custom ICMP request packets or UDP packets to discover active hosts
- Enhanced `traceroute` – Use conventional ICMP Echo Request packets or UDP packets to map a network topology. TCP traceroute can be used as an alternative
- Visual traceroute – Graphical network mapping tool based on `traceroute`
- Packet generator – Generate custom TCP/UDP/ICMP packets to stress test a network or firewall
- Port scanning – Many different port scanning types are supported, including TCP connect() scanning, SYN scanning, stealth scanning, and UDP scanning
- OS fingerprinting – Simple OS fingerprinting based on ICMP probe packets
- DNS query – Easy DNS lookup and zone transfer with `nslookup` and `dig`
- Whois query – Support `whois`, `rwhois`, and over 70 domain extensions
- NetBIOS enumeration – Locate NetBIOS network shares, such as shared drives and printers. Enumerate user lists, account information, password policy, system time, and so forth
- IP packet viewer – Capture IP packets for future examination

Should you have the need to find out about this tool before making a purchase, you should visit http://www.netscantools.com/nstprodetails.html for a comprehensive list of

supported features. Well, if you are a GUI lover and money is not a decisive factor, you might want to consider purchasing this tool for its convenience and completeness. Nonetheless, you don't really miss anything if you decide not to use this tool either because most of the functions offered by NetScanTools Pro can be easily found in many other free information gathering utilities.

IPSECSCAN

Before discussing the purpose and usage of IPSecScan, it is essential that you understand what IPSec is and how it can be related to IPSecScan. Upcoming section just gives you a brief overview of IPSec. For more information, please consult with RFCs 2401.

The IP protocol is not secured by design; if it were then you can rest assure that 50% percent of hundred thousands of security professionals all around the world would be out of a job right now. IP security, or IPSec for short, was later introduced as a framework to help secure communications at the relatively weak IP layer. Developed by the IETF and addressed in RFCs 2401, IPSec aim to offer what the IP protocol is lacking, which are the confidentiality, integrity, and authenticity. Confidentiality preserves the secrecy of the data contained in the packet. Integrity prevents the data from unauthorized modification. Authenticity ensures the true origin and trustworthiness of the received data.

IPSec is able to provide all those security services with much help coming from the three very important protocols: Authentication Header (AH), Encryption Security Payload (ESP), and Internet Key Exchange (IKE).

The AH protocol does not encrypt the packet data; it merely uses a key-hashed algorithm to sign the IP header and payload data to ensure the authenticity and integrity of the IP packet.

The ESP protocol adds encryption capability to IPSec, which helps render the packet data unreadable to unauthorized parties during the transmission. Optionally, ESP also offers similar services like AH, message integrity and authenticity. You can choose to use either AH or ESP or even both, depending on you security needs. Nonetheless, using both of the aforementioned protocols is perhaps the only way that can truly ensure the security of your data.

IKE, on the other hand, doesn't necessarily provide any security services to the IP packet yet its role is just as important as the other two protocols. IKE is a key management protocol that provides a means for IPSec peers to securely exchange secret keys for setting up an IPSec enabled connection.

IPSec typically operates in two modes: transport mode or tunnel mode. In transport mode, the two communicating peers are required to have IPSec enabled and only the data payload is protected and not the packet header. Tunnel mode is the more secure mode, which involves protecting the entire IP packet, including the data payload and the packet header. In tunnel mode, only the two end gateways are require to support IPSec.

Trading with all the essential security services offered by IPSec is the complexity and non-universal standard. Thousand vendors have thousand different ways to implement and integrate IPSec into their products, and even within a single vendor, their product lines may have several different IPSec implementations as well. Obviously, the vendors can only do so much and not all of their IPSec implementations are secured and up to the expectation. Some typical examples of insecure IPSec implementations can be seen as follow (ranging from remote code execution to service interruption):

- Cisco VPN Client IKE Security Parameter Index Payload Buffer Overflow
- OpenBSD ISAKMPD Malformed IPSec SA Payload Denial Of Service
- HP Tru64 UNIX Unspecified IPSec/IKE Remote Privilege Escalation
- Microsoft Windows 2000 IKE Denial of Service Vulnerability

According to the list above, you can see IPSec products are not security goddess and it is with certainty that you will encounter buggy and vulnerable IPSec-based products from time to time. Thus, it is essential that you include IPSec scanning in your to-do-list. The tool of the trade is of course IPSecScan. IPSecScan does exactly what it says, scanning and checking whether a system is IPSec enabled. You can easily download this Windows 2000/XP compatible scanner at http://ntsecurity.nu/downloads/ipsecscan.exe. The good thing about this tool is it is very straightforward to use, even for the non-geek computer user like your grandma as you will see below:

```
C:\>ipsecscan 192.168.0.3 192.168.0.5

IPSecScan 1.1  - (c) 2001, Arne Vidstrom, arne.vidstrom@ntsecurity.nu

               - http://ntsecurity.nu/toolbox/ipsecscan/

192.168.0.3 IPSec status: Indeterminable
192.168.0.4 IPSec status: Disabled
192.168.0.5 IPSec status: Enabled
```

Indeterminable status means either the IP address is not in used or the scanner isn't really sure whether IPSec is enabled or not. Disabled and Enabled statuses are rather self-explanatory. According to the author, the result Enabled presented by the scanner is very reliable and you can trust it completely; however, the Disabled result is just a qualified guess because the scanner isn't capable of telling you exactly if the system doesn't have IPSec enabled.

FIREWALK

Strictly speaking, `firewalk` is not a port scanner per se. It is a network mapping utility that was written by David Goldsmith and Michael Schiffman to help determine filtering rules of a firewall, or a packet-filtering device, and thereby, map an entire protected network. Unlike any other scanning technique that was introduced previously — which mostly communicates with a remote host to determine a state of a port — `firewalk` is only interested in talking and exchanging packets with a firewall to determine which port the firewall allows network packets to pass through. `Firewalk` is

exclusively available for UNIX and Linux platform and its latest version (current latest version is 5) can be downloaded at http://packetfactory.net/firewalk/dist/.

There are two main phases involved in a typical `firewalk` scan. For the first phase — network discovery — `firewalk` does something in a similar effect like `traceroute`, consecutively sending packets with incremented Time-to-Live (TTL) values (starting of one) to count the number of hops between the sending host and the target network's firewall. The hop count number is then used as a metric for `firewalk` to proceed to its subsequent phase — scanning. It is important to note that you must have the IP address of the target firewall in order for `firewalk` and the hop count to work at all. The IP address of the firewall can be gathered by using many different methods, including `traceroute` and social engineering.

If you can't remember what `traceroute` is or how to identify the target network's firewall from its outputs, it is highly recommended that you refer to the previous chapter — Footprinting — for a complete review. `Traceroute` is the easiest and safest way there is that can help you obtain the IP address of the firewall and `firewalk` does need this kind of information to perform its magic.

During the scanning phase, `firewalk` attempts to find out whether a port is filtered by sending a TCP or UDP packet with a TTL value set to expire one hop past the firewall in question. In other words, the TTL value is set to be one greater than the hop count number which `firewalk` discovered from the previous phase. If the firewall allows the packet to pass through, the packet will be able to make another hop and use up all of its TTL value before an ICMP Time Exceeded message (Type 11 Code 0) is generated and returned to the host performing firewalking. If the port is filtered and no traffic is allowed in, the firewall will simply drop the packet and `firewalk` will receive either no error message or an ICMP Port Unreachable message. By sending subsequent TCP or UDP packets using this manner, `firewalk` can well determine all filtering rules of the firewall.

For demonstration purposes, let's look at how you can perform `firewalk` against the host www.gotrice.com without knowing its firewall IP address in advance. Note, the host www.gotrice.com is fictional; any similarity to real life event or situation is unintentional.

```
[bash]# traceroute www.gotrice.com
traceroute to www.gotrice.com (203.162.168.130), 30 hops max, 38 byte
1  202.155.7.1 (202.155.7.1)  80.147 ms  74.863 ms  59.949 ms
2  202.155.7.162 (202.155.7.162)  140.144 ms  139.960 ms  139.863 ms
3  202.84.154.57 (202.84.154.57)  144.885 ms  114.808 ms  109.942 ms
4  134.159.129.174 (134.159.129.174)  380.082 ms  334.85 ms  345.20 ms
5  203.162.231.233 (203.162.231.233)  349.61 ms  344.908 ms  354.983 ms
6  203.162.95.46 (203.162.95.46)  354.922 ms  339.922 ms  349.736 ms
7  203.162.168.130 (203.162.168.130)  365.106 ms 384.931 ms 354.884 ms
```

`Traceroute` has completed successfully according to the example. No packets were dropped during the way and all devices along the path were revealed. The 7th hop is the last hop of `traceroute`, which should be the host www.gotrice.com. The second to the last hop, the 6th hop, reveals the IP address of the firewall, which is 203.162.95.46.

Armed with this knowledge, you can move on to the next stage, performing firewalking.

```
[bash]# firewalk -n –p TCP –S 21,22,23,25,53,80,110,143 203.162.95.46 \
203.162.168.130

Firewalk 5.0 [gateway ACL scanner]
Firewalk state initialization completed successfully.
TCP-based scan.
Ramping phase source port: 53, destination port: 33434
Hotfoot through 203.162.95.46 using 203.162.168.130 as a metric.
Ramping Phase:
1 (TTL 1): expired [202.155.7.1]
2 (TTL 2): expired [202.155.7.162]
3 (TTL 3): expired [202.84.154.57]
4 (TTL 5): expired [134.159.129.174]
5 (TTL 5): expired [203.162.231.233]
6 (TTL 6): expired [203.162.95.46]
Binding host reached.
Scan bound at 7 hops.
Scanning Phase:
port  21: A! open (port not listen) [203.162.168.130]
port  22: A! open (port listen) [203.162.168.130]
port  23: A! open (port not listen) [203.162.168.130]
port  25: A! open (port not listen) [203.162.168.130]
port  53: A! open (port not listen) [203.162.168.130]
port  80: A! open (port listen) [203.162.168.130]
port 110: A! open (port not listen) [203.162.168.130]
port 143: A! open (port not listen) [203.162.168.130]
```

The initial `–n` option used to specify `firewalk` not to perform DNS name resolution — doing this can reduce the amount of probing time. The `–p TCP` option then specifies `firewalk` to perform a TCP scan type. The `-S` specifies the ports which to be scanned. Ports can also be defined by range, delimited by dashes instead of commas as in this case. Finally, the IP address the firewall 203.162.95.46 and the IP address of the destination host 203.162.168.130 are provided in order for `firewalk` to start the party. It is worth noting that the destination host does not have to be reachable; `firewalk` in fact only needs any IP address that is belong to the other side of the firewall.

COUNTERMEASURE: MINIMIZE THE EXPOSURE WITH LESS OPEN PORTS

This probably had already been said a million times but I think it's still worth it to remind you that the first and foremost important countermeasure against port scanning is to disable unnecessary network services that do nothing else than providing potential hackers with more chances and opportunities to penetrate the network. Ironically, many operating systems nowadays are designed to be insecure by default, and that is having all network services turned on by default instead of the other way around. Hence, as a rule of thumb, you should disable all network services and only enable what are absolutely needed, as explicitly stated in the security policy.

For Windows OS, you can go to Settings → Control Panel → Administrative Tools → Services. Through the Services console, you can start, stop, disable, enable, and configure the startup behavior of various Windows applications, including those network

applications. Look out for those network services that have the startup type as Automatic and disable them if necessary. For Linux and UNIX OS, you can easily go to the startup directory `init.d`, which is normally located inside `/etc`, and remove the execution bit set on the startup scripts of networks services that you don't want to startup at boot time. You must also carefully review the `inetd.conf` or `xinetd.conf` file located in the `/etc` directory and comment out the lines that would enable unwanted network services otherwise. As the first line of defense, the border router should be set properly to only allow traffic in through a certain ports while explicitly blocking the rest.

Scanning usually implies an attack is on the way, and for this reason, knowing when the scanning is taking place is clearly a great advantage that would help prevent the real attack from occurring, or at least, to reduce the threats to a minimal. You can use both Host-based and Network-based IDS for tasks like these and as introduced earlier in the ping sweeping section, Snort, BlackIce, and Scanlogd are all good tools for the job.

Security through obscurity is not out of a question either. Many have come to prove that security can't be achieved through obscurity, but in this case, security can certainly be improved through obscurity. For network services that hold critical roles, it is a good call to make them listen on other different ports instead of the regular well-known ports. It is imperative to note that in doing so doesn't guarantee a perfect security solution; it only helps make the system and network look less attractive to the hackers when all the potential well-known ports are found to be not listened by any network services. Port Knocking (http://www.portknocking.org) is a free and an innovative tool that offers you an extra layer of protection through obscurity. Port Knocking allows you to have a port closed and only open when you send a predefined sequence of network packets to "knock" on it. The hackers and their port scanners won't be able to determine if a port is really closed or protected by Port Knocking. You should visit the website for more information on how Port Knocking can help strengthen the security of the network.

Defending against firewalking can be a tough task in which the hackers do not specifically probe any target host for open port, but rather survey the firewall directly to determine filtering rules. Normally, many people just choose to accept the faith and allow the hackers to learn the rulesets of the firewall with this clever technique. Having said that, all is not lost, there is still a few ways which you can effectively use to prevent firewalking. You can enforce a stricter rule at the border router so that ICMP Time Exceeded message cannot leave the network but doing this would also reduce administration's capability since `traceroute` and various other network troubleshooting tools rely on ICMP Time Exceeded error message to function. Alternatively, you can use an application level firewall, also known as a proxy server, in lieu of the packet-filtering device. The proxy-based firewall replaces the packet TTL value with its own TTL value before transmitting the packet to the destination host, and thus, rendering `firewalk` ineffective. Still, this method must be used with consideration because the proxy server doesn't have the filtering capability as found in the real firewall or packet-filtering device.

Similar to war-dialing, you must conduct port scanning against your network on a regular basis to uncover ports that are inadvertently open by careless users or network

application. Port scanning must be handled professionally and conducted with great care as malformed scanning packets may cause the whole network go down as a result. Last but not least, do not scan without prior written permission from the upper management.

C. IDENTIFYING NETWORK SERVICES — APPLICATION MAPPING

Identifying the network service associating with each opened port of a system helps the hackers gain more knowledge of the target network environment. Even though the hackers may already know which service might be listening on a particular open port, it is still very crucial to find out the exact software application used for providing the service and its version. Knowing the network application and its corresponding version not only give the hackers the luxury to proceed to the vulnerability mapping stage — which is where they will attempt to identify the associated vulnerabilities of the network service — but also greatly increase their odds of success.

Banner grabbing is the oldest and yet the most effective method to help identify the software application that sits behind the open port. It works by sending a packet to a port and then analyzing the response returned by the port to look for a string, also known as a *banner message*, which advertises or gives away information about the software. Several programs such as `nmap` and `amap` from The Hacker's Choice (http://www.thc.org/amap) are capable of identifying the software and its corresponding version with high probability. The uses of these two tools are illustrated in more details as follow:

```
[bash]# nmap -sSV -p 22,25,80,34567 www.shady.com -P0
Starting nmap 3.81 ( http://www.insecure.org/nmap/ ) at 2004-04-18
Interesting ports on www.shady.com (64.21.248.29):
PORT      STATE    SERVICE VERSION
22/tcp    filtered ssh
25/tcp    open     smtp    netqmail smtpd 1.04
80/tcp    open     http    Microsoft IIS webserver 6.0
34567/tcp open     unknown
```

You should be familiar with most of the `nmap` options used in the above example. The only new option introduced here is the `-V`, requesting `nmap` to perform version detection besides looking for open ports on the remote host www.shady.com. Three ports 22, 80, and 34567 are reported to be open. In addition, `nmap` also reports the software and the version associated with each of those open port, for instance, port 25 is reported to be listened by the mail daemon called netqmail smtpd, version 1.04.

`Amap` works in a similar concept but not quite. Sometimes network administrators can be tricky and they can switch things around like having FTP service listen on port 80 instead of its regular port 21. Knowing this, Van Hauser releases `amap` to address that very issue. `Amap` works by sending a trigger packet to elicit a response from an open port, compare the response to the application database, and finally print the one that is most closely match. `Amap` can also be a port scanner but using it for such task is generally not recommended since it is only capable of performing a full TCP connect scan, which is not clandestine in anyway.

```
[bash]# amap -Ab www.shady.com 25 34567
```

```
amap v5.0 (www.thc.org/thc-amap) started at 2005-04-18 - MAPPING mode

Protocol on 64.21.248.29:25/tcp matches smtp - banner: 220 Ready ESMTP\r\n
Protocol on 64.21.248.29:34567/tcp matches ssh - banner: SSH-2.0-4.3.2.1\r\n
```

Do you still remember how in the previous example `nmap` wasn't able to reveal software information regarding the irregular port 34567? In this example, `amap` is asked to try doing something analogous to the `nmap –V`, and as you can see, `amap` does deliver what it promises. Response returned by port 34567 matches the signature of the SSH protocol listed in the database and `amap` reveals that correctly. Moreover, `amap` also grabs the banner of that port and reveal the version of the SSH daemon. The `-A` option enables application mapping mode, requiring `amap` to send trigger packets and analyze the response. This is the default mode. `Amap` also performs banner grabbing with the use of the `-b` option. This mapping tool has many more options, which you can use to fine-tune your scan. For a complete list of all possible options, you can execute `amap` alone on the command line or type `man amap`.

So far, you have been introduced to the easy way of banner grabbing and application mapping but I know some of you might not always prefer to take that route and prefer to do things the hard way. Yes, of course you can. You can always let `nmap` and `amap` go dusted and use `telnet` or `nc` (netcat) to find out which service is listening on which port manually. The following example shows how you can perform banner grabbing using `telnet`, since `telnet` is ubiquitous and available on most OS platform today. If you prefer to use or want to find out more about netcat however, you can still download it here http://packetstormsecurity.org/UNIX/utilities/nc110.tgz.

```
[bash]# telnet www.whatever.com 80
Trying 66.21.24.5...
Connected to www.whatever.com.
Escape character is '^]'.
GET / HTTP/1.0

HTTP/1.1 200 OK
Date: Sun, 17 Apr 2004 20:50:20 GMT
Server: Microsoft-IIS/6.0
Last-Modified: Fri, 01 Apr 2004 19:47:51 GMT
Accept-Ranges: bytes
Content-Length: 377
Connection: close
Content-Type: text/html
```

There is nothing fancy about this manual method of banner grabbing really. It's just a simple example showing how easy it is for one to find out information about an open port without the need of using powerful tools like `amap` or `nmap`.

COUNTERMEASURE: WEAR A DIFFERENT SHIRT AND GO UNNOTICED!

As obvious as it may sound, the best way to counter banner grabbing is to manipulate the banner message of the software so that it advertises or reveals information that may be totally unattractive, confused, or misleading for the hackers. Changing the banner of a

network service is generally not recommended because the result can be disastrous if not done properly. You may cause compatibility and interoperability issues for changing the banner message and version info of a particular service, as many network services are known to rely on that information to work properly. You may wish refer to the official documentation of the network software to see if the vendor provide any instructions for such action, which may be safer than hex editing and doing it all by yourself. However, if the benefits brought by changing banner message are not worth the risk and effort, you should consider leave it as is and focus on other finer security measures that do improve the overall security posture of your network.

D. DETECTING OPERATING SYSTEMS — OS FINGERPRINTING

Merely knowing which port is open on a remote host is still not good enough. A good and determined attacker must also be able to detect the Operating System (OS) of the remote host. This kind of information is exactly what the attacker will need for the vulnerability mapping phase, much owing to the fact that vulnerabilities and exploits are very dependent on the OS version. In many cases, an exploit for a vulnerability may work well against a specific type of OS but will not work at all against the others — albeit it is to be used to exploit the exact same vulnerability. In addition, many OSs are known to be vulnerable to security holes themselves; therefore, accurately identifying the OS of the target host also gives the attacker the opportunities to compile a nice list of security bugs associating with that OS. It is imperative to note that a successful attack normally relies much on the accuracy of OS information discovered during this phase.

There are typically two ways one can go about identifying remote OS of a system: active stack fingerprinting and passive fingerprinting. The following sections will help you have a closer look at how each of these OS fingerprinting method works, the pros and cons, and the respective tools that you can use to accomplish the task more quickly.

:.::.: ACTIVE STACK FINGERPRINTING

While the TCP/IP stack is pretty much implemented as a standard and an indispensable part of an OS, it is still no guaranteed that the TCP/IP stack in every OS will behave the same way — especially upon receiving malformed network packets. Much of the explanation for this lies in how each OS vendor interprets the standard differently and implements the TCP/IP stack to their own liking as a result.

Active stack fingerprinting is a clever OS detection technique that takes advantages of all the differences among various TCP/IP stack implementations. It involves sending a crafted network packet to a remote system to elicit a unique response from the TCP/IP stack of the underlying OS. That unique response is referred to as an OS *fingerprint* or *signature*, which uniquely identifies one OS from another. The attacker then carefully analyzes and compares the fingerprint to a database — comprising a wide range of known OS fingerprints — and the OS that is most closely matched to the fingerprint in question would be the OS that is sending the response to the attacker.

There are many active stack fingerprint techniques available but here are some of the popular ones that are employed by `nmap`'s OS fingerprinting feature (`-O`):

- **FIN Probing –** As you have seen earlier in the Stealth Scanning section, RFCs 793 requires any system with an open port to not respond and ignore the FIN packet upon receiving it at the start of a connection. Nonetheless, Microsoft Windows and many other OSs in the field, such as BSDI and IRIX, completely disregard the requirement and reply to the FIN packet with a RST packet.
- **Bogus Flag Probing –** A SYN packet is sent to the target system with an undefined flag set in the TCP header. While many OSs would normally reset the flag in their response packet, other OSs, such as Linux prior to 2.0.35, respond with the flag bit set as is.
- **ACK Value –** Normally the value in the ACK field in the response packet should be the same as the Initial Sequence Number (ISN) of the sent packet. However, OSs such as Microsoft Windows (yet again) sets the ACK value to be ISN + 1.
- **IP ID Sampling –** You have seen previously how the ID field in the IP header can be used to perform an extraordinary stealthy port scanning. It is put in use here once again. The idea behind this technique is to find out how the OS uses and generates its IP ID sequence number. Microsoft Windows OS usually uses a predictable IP ID sequence number, such as increasing the number by 1 or 256 for each packet sent. For others OSs, such as OpenBSD and Linux, the IP ID number is randomized or set to 0 instead.
- **TCP Initial Sequence Number (ISN) Sampling –** When receiving a request to establish a connection, an OS must choose an Initial Sequence Number to respond and continue the three-way handshake. This fingerprinting technique involves finding out how each ISN is generated and chosen by the underlying OS when responding to a connection request. Some OSs are known to choose the ISN based on randomized values, while other OSs such as Windows generate the ISN based on system's internal clock. Predictable ISN is a also potential security issue. The infamous Mitnick's attack against the security guru Shimomura was formulated based on the predictable ISN generation.
- **TCP Initial Window Size –** This involves checking the TCP initial Window size reported in the returned packets. Using this technique sometimes can help you precisely detect the remote OS since some OSs are known to report a unique Window size.
- **TCP Timestamp –** Some stack implementations increase the value in the TCP Timestamp option at frequencies of 2HZ, 100HZ, or 1000HZ and this technique based on the Timestamp feature to detect the OS type. Do note that not all systems support this feature because TCP Timestamp is rather an option — not a requirement.
- **TCP Options –** TCP options are added overtime to enhance the TCP protocol. Typically, these options are of course optional and therefore not all systems implement them. However, if an OS supports TCP options, it will set the options in the returned packet. The number of supported options, the value, or the order of which the TCP options is listed can be use to uniquely identify one OS from another. For example, Linux 2.0.x kernels support only a few options while Linux 2.1.x kernels support all of them.
- **ICMP Error Message Quoting –** According to RFCs 793, one OS must quote some parts of the original ICMP message — particularly 8 bytes — when generating an ICMP error message. However, Linux and Solaris do not abide by the specifications and include much more information than the requirement. Since ICMP is used, this technique

can still detect the remote OS regardless whether a port is listening or not.

- **ICMP Error Message Echo Integrity –** Some OSs manipulate the IP header when generating an ICMP error message. This stack fingerprinting technique takes note of all the differences and changes to identify the OS that causes all these changes.
- **ICMP Error Message Type of Service (TOS) –** By examining the value in the TOS field in the ICMP Port Unreachable error message, you may be able to identify the OS that generates the message. Normally, the TOS value should be 0, but Linux OS gives itself away by returning a different a value.
- **Don't Fragment Bit –** Some OSs are reported to set the "Don't Fragment" bit in some of their packets for performance reasons while the others are reported to not have this bit set at all. Noting these differences may give you some hints about the target OS.

It is clear that active stack fingerprinting can be extremely powerful. One can use many of the techniques mentioned above to gain highly accurate information about the OS of the remote host. Furthermore, not only the information is extremely accurate, but it is also very detailed. An attacker can get information as detailed as the patch level of the remote host and that has significant contribution to the success of the attack. Having said that, active stack fingerprinting is not God and it does fall short in a few areas. First, it is not a toy for unprivileged users. To manually compose anomalous network packets, the attacker needs to have the highest privilege on the system. This limitation is necessary to prevent normal users from accidentally creating faulty network packets that may render the system unusable. Second, most of these active stack fingerprinting techniques are not stealthy in nature. One of the main attributes to this is that the attacker is required to send crafted network packets directly to the remote host to elicit responses, and that, may expose his or her identity to the remote IDS or firewall system in one way or another.

There is a myriad of security tools that are capable of detecting OS of a remote host using active stack fingerprinting, including `queso`, `nmap`, and `xprobe`. `Queso` is rather dated and rarely used, leaving `nmap` and `xprobe` to be the most renowned and powerful OS fingerprinting tools available to date. However, you should remember that `nmap` and `xprobe` each has a different approach to perform active stack fingerprinting. While `nmap` mainly relies on the TCP protocol for the OS fingerprinting tasks, `xprobe` takes advantages of the ICMP protocol thoroughly to obtain a similar kind of information.

NMAP

Most of the powerful features of `nmap` had all been laid out on the table and discussed thoroughly in the port scanning section. Therefore, to keep thing short and simple, this section will only introduce you to the other powerful scanning option that you haven't seen so far, the `-O`, which specifies `nmap` to perform stack fingerprinting against the system being scanned.

```
[bash]# nmap -O -p 80,1 www.scan-me.com

Starting nmap 3.81 ( http://www.insecure.org/nmap/ ) at 2004-10-02
Interesting ports on www.scan-me.com (192.168.0.12):
PORT   STATE  SERVICE
```

```
49/tcp closed tacacs
80/tcp open   http
Device type: general purpose
Running: FreeBSD 4.X
OS details: FreeBSD 4.6.2-RELEASE - 4.8-RELEASE, FreeBSD 4.7-RELEASE
Uptime 124.155 days (since Tue Dec 28 21:10:48 2004)
```

Ideally, in order for nmap to work its magic and give you the most reliable OS information about the remote host, you should provide nmap with at least one open and one closed TCP port on the remote host. In the precedent example, after the –O option, one open port, 80, and one closed port, 1, are provided to satisfy nmap's OS detection criteria. Of course, the port numbers are not restricted to port 80 or 1. You are free to pick other port numbers if you wish, as long as one of them is closed to make nmap happy. As you can clearly see, not only can nmap provide the OS information of www.scan-me.com, which is FreeBSD 4.X, but it also reveals the uptime information of the host. Uptime information is deduced and calculated from the values in the TCP Timestamp option in the response packets. The following example shows you whether nmap will work at all against a system that has no port open.

```
[bash]# nmap -O -p 1 www.scan-me.com

Starting nmap 3.81 ( http://www.insecure.org/nmap/ ) at 2004-10-02
Warning:  OS detection will be MUCH less reliable because we did not find at
least 1 open and 1 closed TCP port
Interesting ports on www.scan-me.com (192.168.0.12):
PORT  STATE  SERVICE
1/tcp closed tcpmux
Too many fingerprints match this host to give specific OS details
```

Needless to say, nmap becomes far less effective when it has no ports to toy with. It is unable to provide any result at all due to too many matches as in this example. For this reason, it is highly recommended that you find at least one open port on a target system before having nmap to perform OS detection.

XPROBE

Xprobe is a sublime OS stack fingerprinting tool written and maintained by the three famous pioneers in this arena, Fyodor Yarochkin (nmap's author), Ofir Akin, and Meder Kydyraliev. Before going any further into xprobe discussions and examples, you should first download the tool at http://www.sys-security.com/index.php?page=xprobe. While you are at it, make sure you download all the papers and tutorials relating to xprobe as well because it is impossible to explain and wrap all the advanced features of this nifty tool in just a few pages. Those additional papers will provide you with necessary information that you can use to customize your fingerprinting tests and even write your own testing modules should you have the needs for it.

As I said before, xprobe takes a total different approach to active OS stack fingerprinting. Instead of relying on the TCP protocol and sacrificing the speed as well as the accuracy, xprobe uses the ICMP protocol to address all the deficiencies of many active

stack fingerprinting tools that you may come across today.

The probing speed is dramatically improved because the need of building and sending many anomalous network packets to a remote system is no longer a necessary. OS information of the remote system in fact can be elicited by a mere single probe packet. Moreover, unlike `nmap` or `queso` using the strict signature matching method, `xprobe` is the first and probably the only active OS fingerprinting tool that uses the matrix based fingerprinting matching method, also known as "fuzzy" matching, to detect OS information. Fuzzy matching doesn't compare one OS signature to a signature database and look for a 100% match as in the strict signature matching method. Instead, it calculates the score of each fingerprinting result and adds them all up together to provide the best effort match — which is far more reliable and accurate.

Besides the accuracy and speed, `xprobe` is also extremely versatile and well documented. Users are given complete control over their fingerprinting tests. They can specifically choose which test modules to be loaded, which packet should be sent, and the time elapse between sending each packet. Not only that, they are also given the freedom to create their own module of fingerprinting techniques and update the database with new OS signatures without having to wait and rely on the authors' mercy. Ranting is over; it is now time to look at what exactly `xprobe` can do for you.

```
[bash]# xprobe2

Xprobe2 v.0.2.2 Copyright (c) 2002-2005 fyodor@o0o.nu, ofir@sys-security.com,
meder@o0o.nu

usage: xprobe2 [options] target
Options:
    -v                      Be verbose
    -r                      Show route to target(traceroute)
    -p <proto:portnum:state> Specify portnumber, protocol and state.
                            Example: tcp:23:open, UDP:53:CLOSED
    -c <configfile>         Specify config file to use.
    -h                      Print this help.
    -o <fname>              Use logfile to log everything.
    -t <time_sec>           Set initial receive timeout or roundtrip time.
    -s <send_delay>         Set packsending delay (milseconds).
    -d <debuglv>            Specify debugging level.
    -D <modnum>             Disable module number <modnum>.
    -M <modnum>             Enable module number <modnum>.
    -L                      Display modules.
    -m <numofmatches>       Specify number of matches to print.
    -T <portspec>           Enable TCP portscan for specified port(s).
                            Example: -T21-23,53,110
    -U <portspec>           Enable UDP portscan for specified port(s).

---options truncated for brevity---

[bash]# xprobe2 -v –m 5 -p tcp:80:open -p udp:53:closed www.scan-me.com

Xprobe2 v.0.2.2 Copyright (c) 2002-2005 fyodor@o0o.nu, ofir@sys-security.com,
meder@o0o.nu

[+] Target is www.scan-me.com
[+] Loading modules.
[+] Following modules are loaded:
```

```
[x] [1] ping:icmp_ping  -  ICMP echo discovery module
[x] [2] ping:tcp_ping  -  TCP-based ping discovery module
[x] [3] ping:udp_ping  -  UDP-based ping discovery module
[x] [4] infogather:ttl_calc  -  TCP and UDP based TTL distance calculation
[x] [5] infogather:portscan  -  TCP and UDP PortScanner
[x] [6] fingerprint:icmp_echo  -  ICMP Echo request fingerprinting module
[x] [7] fingerprint:icmp_tstamp  -  ICMP Timestamp request fingerprinting
[x] [8] fingerprint:icmp_amask  -  ICMP Address mask request fingerprinting
[x] [9] fingerprint:icmp_port_unreach  -  ICMP port unreachable fingerprinting
[x] [10] fingerprint:tcp_hshake  -  TCP Handshake fingerprinting module
[x] [11] fingerprint:tcp_rst  -  TCP RST fingerprinting module
[+] 11 modules registered
[+] Initializing scan engine
[+] Running scan engine
[+] Host: 192.168.0.12 is up (Guess probability: 100%)
[+] Target: 192.168.0.12 is alive. Round-Trip Time: 0.33328 sec
[+] Selected safe Round-Trip Time value is: 0.66655 sec
[+] Primary guess:
[+] Host 192.168.0.12 Running OS: "FreeBSD 4.8" (Guess probability: 84%)
[+] Other guesses:
[+] Host 192.168.0.12 Running OS: "FreeBSD 4.9" (Guess probability: 84%)
[+] Host 192.168.0.12 Running OS: "FreeBSD 4.10" (Guess probability: 84%)
[+] Host 192.168.0.12 Running OS: "FreeBSD 5.2.1" (Guess probability: 79%)
[+] Host 192.168.0.12 Running OS: "FreeBSD 5.2" (Guess probability: 79%)
[+] Cleaning up scan engine
[+] Modules deinitialized
[+] Execution completed.
```

When no modules are explicitly disabled or enabled on the command line, `xprobe` enables and registers all eleven modules — which are categorized into three distinctive probing phases. The first three modules are responsible for carrying out reachability tests to verify if the target host is dead or alive. The 4th and 5th modules allow `xprobe` to gather further information about the target host before the real fingerprinting tasks can begin. The rest of registered modules are fingerprinting modules that handle the real job — performing stack fingerprinting to identify OS of the target host.

In this example, the `-v` option is used to specify `xprobe` to be verbose. Then `xprobe` is set to display only five matches, as can be seen with the `-m 5` option. Do note that `xprobe` displays up to ten matches by default, but I have to limit them to five to save some space. Just like its cousin `nmap`, `xprobe` also requires you to feed it with at least one open TCP port and one close UDP port and you can do that using the `-p` option as can be seen in the example. Finally, www.scan-me.com is passed to `xprobe` as the address of the target host. After receiving and putting the host signatures through several calculations, `xprobe` offers FreeBSD 4.8 as the primary guess. With a score of 84%, you can be confident and trust the result right away. However, it is critical to note that there's always a chance for the result to be false positive. Besides the primary guess, `xprobe` also gives you other possibilities with various scores. As a rule of thumb, you should put your trust into the one with the highest score. For those who want to know, yes the target host www.scan-me.com is in fact running FreeBSD 4.8.

The following example shows how you can specifically enable only one module and have `xprobe` perform a single stack fingerprinting test against the same target host.

```
[bash]# xprobe2 -v –m 5 -M 8 -p tcp:80:open -p udp:53:closed www.scan-me.com

Xprobe2 v.0.2.2 Copyright (c) 2002-2005 fyodor@o0o.nu, ofir@sys-security.com,
meder@o0o.nu

[+] Target is www.scan-me.com
[+] Loading modules.
[+] Following modules are loaded:
[x] [1] fingerprint:icmp_amask  -  ICMP Address mask request fingerprinting
[+] 1 modules registered
[+] Initializing scan engine
[+] Running scan engine
[+] All alive tests disabled
[+] Target: 192.168.0.12 is alive. Round-Trip Time: 0.00000 sec
[+] Selected safe Round-Trip Time value is: 10.00000 sec
[+] Primary guess:
[+] Host 192.168.0.12 Running OS: "HP JetDirect ROM G.06.00 EEPROM G.06.00"
(Guess probability: 100%)
[+] Other guesses:
[+] Host 192.168.0.12 Running OS: "Cisco IOS 12.2" (Guess probability: 100%)
[+] Host 192.168.0.12 Running OS: "Cisco IOS 11.1" (Guess probability: 100%)
[+] Host 192.168.0.12 Running OS: "FreeBSD 4.10" (Guess probability: 100%)
[+] Host 192.168.0.12 Running OS: "FreeBSD 2.2.7" (Guess probability: 100%)
[+] Cleaning up scan engine
[+] Modules deinitialized
[+] Execution completed.
```

The only new option introduced here is `–M 8`. The `-M 8` option specifies `xprobe` to use the 8th module to carry out an ICMP Address mask request fingerprinting against the target. Needless to say, the result was terrible. `Xprobe` reveals far too many matches with 100% scores to be any useful. Even though FreeBSD 4.10 is listed as an alternative guess, the primary guess is terribly wrong. Consequently, it is highly recommended that you perform at least two different fingerprinting tests against the target host in order to obtain a reliable and an accurate result.

:.::.: PASSIVE STACK FINGERPRINTING

Passive stack fingerprinting is a much more subtle and non-intrusive way to gather hints about the OS of a remote host — based on the differences in various stack implementations. While active stack fingerprinting requires the attacker to compose and send malformed network packets to the remote host to elicit a response, the recipe to passive stack fingerprinting is very simple indeed: a sniffer and a decent OS signature database, though the signature database is rather optional.

First, the attacker needs to setup a sniffer on his or her side and then initiate a legitimate connection to the target host for the sniffer to capture the packets exchanged between the two. The attacker then carefully analyzes several attributes in the TCP header of one of those packets originating from the target host for glues about the underlying OS of the host.

This approach to fingerprinting is amazingly stealthy and supposedly undetectable. The target host would not be aware about the situation because probe packets are never

sent to the target host and the communication between the attacker and the host is purely legitimate with normal network traffic. Having said that, this approach also has several drawbacks. Firstly, the TCP header of a packet may be altered or changed by the target host or intermediary devices, such as firewalls or proxy servers, and that will have severe impacts upon the accuracy and reliability of the OS signature. Secondly, passive stack fingerprinting works with limited information from the TCP header of a packet, and for this reason, the result will be much less precise than that of active stack fingerprinting, for example, patch level of an OS cannot be identified using this passive approach.

The following are some typical fields in the TCP header whose values may vary from one OS to another. By noting the differences in these fields, you may be able to infer the OS of a remote host. Note, there are of course more for you to look at and examine in the TCP header, but for conciseness, only four will be listed.

- **Time-to-Live (TTL) –** Different OS is known to have different default TTL value set in its outbound packets. For example, Linux or FreeBSD OS normally uses a TTL value of 64 while Microsoft Windows uses a TTL value of 128.
- **Window Size –** This involves checking the TCP initial Window size reported in the packets. The Window size can vary among many OSs.
- **Don't Fragment Bit –** Some OSs are reported to set the "Don't Fragment" bit in some of their packets for performance reasons while the others are reported to not have this bit set at all. Noting these differences may give you some hints about the target OS.
- **Type of Service (TOS) –** By examining the value in the TOS field, you may be able to identify the OS that generates the message. Normally, the TOS value should be 0.

As aforementioned, the only tool required for passive stack fingerprinting is a network sniffer. The following shows you an example of a TCP header of a packet captured by `tcpdump` during a HTTP session, which you can use to deduce the OS of the host:

Besides the network sniffer, there are many other tools for you to use to automate and accelerate the task, such as `ettercap`, `p0f`, and `siphon`. Old and out-of-date, `siphon` (http://www.subterrain.net/projects/siphon) is no longer relevant for today's needs. The two other tools, `ettercap` (http://ettercap.sf.net) and `p0f` (http://stearns.org/p0f) are still being supported and widely used by many network security professionals as well as hackers all around the world. Discussions about `p0f` will be included in the Bonus Material section.

```
[bash]# tcpdump –v –n host www.scan-me.com

16:58:53.702867 IP (tos 0x0, ttl 51, id 28722, offset 0, flags [DF], length:
60) 192.168.0.12.80 > 192.168.0.16.33138: S [tcp sum ok]
1737146385:1737146385(0) ack 3911452517 win 57344

---results truncated for brevity---
```

By analyzing the captured packet, you should be able to identify the four required attributes for passive stack fingerprinting:

- **Time-to-Live (TTL) :** 51

- **Window Size :** 57344
- **Don't Fragment Bit :** Yes
- **Type of Service (TOS) :** 0

The TTL value of 51 as shown in the captured packet is not the initial TTL value that you are looking for. For this packet to get to this computer, it has to make a trip and pass through several devices and its TTL value is automatically subtracted by one at each hop. Therefore, in order to get the initial TTL value set by the originating host, you have to perform `traceroute` to calculate the number of hops and see how many times the TTL value had been altered. In this scenario, performing a `traceroute` back to 192.168.0.12 reveals that the packet had passed through a total 13 hops. That gives you the initial TTL value of 64 (51 + 13).

Both `p0f` and `ettercap` come with a decent OS fingerprint database at your disposal. You should consult with either one of those OS fingerprint database to find out which OS that has the same attributes like those above. The following uses the `p0f`'s database — `p0f.fp` — as an example:

```
[bash]$ grep 57344:64:1 p0f.fp

57344:64:1:44:M*:.:FreeBSD:4.6-4.8 (no RFC1323)
57344:64:1:60:M*,N,W0,N,N,T:.:FreeBSD:4.6-4.9
57344:64:1:64:M*,N,N,S,N,W0,N,N,T:.:OpenBSD:3.3-3.4
```

The OS of www.scan-me.com as revealed by `xprobe` earlier is FreeBSD 4.8 and the results offered by passive stack fingerprinting are not very much further from the truth either. It should be noted that `p0f.fp` combines many other different attributes in a TCP session to make up an OS signature, such as Maximum Segment Size (MSS), Window scaling, NOP option, and so forth. Therefore, the result could have been much more precise if you had provided all the available attributes.

COUNTERMEASURE: LET'S MESS UP YOUR FINGERPRINT

Preventing active OS stack fingerprinting is by no means an easy task. The unique response — generated by the TCP stack of each OS — to malformed network packets is the fingerprint that gives away clues about the OS type. Therefore, in order to reduce the threats and make the attacker's life a misery, you must be able to modify the TCP stack of the OS and control how it should react to crafted network packets. Fortunately, tools such as IP Personality, IP Scrubber, and Morph are all capable of allow you to manipulate an OS response and give the attacker mislead information.

Patching the kernel and giving you control over the TCP/IP stack behaviors, IP Personality is probably the best option. You are allowed to make changes to various aspect of the Linux kernel 2.4.x TCP/IP stack, such as the IP ID field, the Initial Window size, the Initial Sequence Number, and so forth. Those attributes, as you already know, are what many active stack fingerprinting tools use to deduce an OS type and by altering such, you can have your OS impersonate as another OS. IP Personality can still be found

here http://ippersonality.sf.net but currently the use this tool is highly deprecated. It has not been updated since 2001 and only supports Linux kernel 2.4.18 or older — which is rather old, buggy, and vulnerable to many security issues.

Morph and IP Scrubber both work based on a very similar concept to firewall technology in which they require all the network traffic to and from the local network to go through them. Any traffic traveling from the local network to the remote network will be scrutinized and any clues or hints that may give away OS information to the attacker will be "scrubbed" or removed. Morph http://www.synacklabs.net/projects/morph is highly adaptable because it doesn't rely on any specific Linux kernel version. Future development of Morph also promises to provide support for Microsoft Windows. IP Scrubber is a *BSD project which is still currently under development.

Likewise, major OSs such as Linux, Microsoft Windows, *BSD, and Solaris, all give you an opportunity to make one or two minor changes to their TCP/IP stack implementation. For example, on a typical Linux distribution, you can easily change the default TTL value of 64 to something like 128 — which is the default TTL value for most versions of Microsoft Windows — by editing the file `/proc/sys/net/ipv4/ip_default_ttl`. It can't be emphasized enough that you must consult with the OS vendor before applying any changes to the TCP/IP stack as these changes may introduce stability issues to your network connectivity as well as normal OS operation.

If you are not fancy about hacking up the TCP/IP stack implementation of your OS and take a gamble on your network connectivity, you should consider hiding all the critical servers behind a proxy server and firewall system. The proxy server will stand in the middle and act on behalf of the servers to communicate with the remote network. As the servers are effectively invisible, the attacker will have a hard time finding the target, let alone fingerprinting it. The firewall system should be configured to drop all ICMP related traffic because such traffic gives out a great deal of information about an OS type. You must also refer to your company's security policy and only apply changes if allowed.

Of course, security is not a hide-and-seek game intrinsically. It is not about how good you are hiding your servers away from the prying eyes, but it is all about whether you have taken all the necessary steps to harden and secure the servers properly so that they will be able to withstand the attacks, irrespective of which information the attacker may know about the servers.

Instead of putting great efforts into preventing OS stack fingerprinting, it is perhaps more prudent and less hassle for you to setup an Intrusion Detection System to pick up those probe attempts. Most of the port scanning detection tools and Intrusion Detection Systems introduced previously are all well capable for such tasks, for instance, IPLog, Snort, and BlackICE.

E. FINDING VULNERABILITIES — VULNERABILITY SCANNING

Reaching this phase, the hackers should have been equipped with a lot of information

about the target network, such as, active hosts, open ports, and listening network services. However, that sort of information is still inadequate and not good enough to help the hackers break into the target network just yet. In fact, they will have to complete this last phase — vulnerability scanning — to collect the final yet decisive piece of puzzle.

Vulnerability scanning refers to the process of checking whether a system, or a network as a whole, is susceptible to known vulnerabilities. If there is a vulnerability to be found, the hackers will try to exploit and gain unauthorized access to the network through that specific vulnerability. It is worth noting that security vulnerabilities do *not* always result in instant access to the protected network upon successful exploitation. Some of them can be exploited merely to gain further information about the target network, for instance, installation path of a software or valid usernames of a FTP service.

Now that you know the simple concept behind vulnerability scanning but how exactly are you going to carry it out? Are you going to spend all days to connect to a system and perform several thousands of different vulnerability tests yourself? Of course not! Why wasting all that precious time when there are millions of vulnerability scanning tools at your disposal?

Vulnerability Scanner	License	Platform	Web Site
Saint	Commercial	Linux	http://www.wwdsi.com
Nessus	Free	Linux, Windows	http://www.nessus.org
NIKTO	Free	Linux	http://www.cirt.net/code/nikto.shtml
Retina	Commercial	Windows	http://www.eeye.com
GFI LANguard NSS	Commercial	Windows	http://www.gfi.com
ISS Security Scanner	Commercial	Windows	http://www.iss.net

Exhibit 3-4. Various Popular Vulnerability Scanners

Vulnerability scanning tools, popularly known as *vulnerability scanners*, are designed to help you automate and speed up the tedious process. Packed with a user-friendly interface, fast scan engine, and a huge vulnerability database, vulnerability scanners nowadays make scanning much more enjoyable and effective. In fact, with decent a vulnerability scanner, you can literally complete scanning a system for thousands of known vulnerabilities in just a matter of minutes. Besides, most of the renowned vulnerability scanners that you come across today will also have an option for advanced users who want to write their own scripts or plugins to enhance their scan and the scanner as a whole. Exhibit 3-4 lists some very popular and effective vulnerability scanners to date.

Showing you how to use these scanners comprehensively is certainly beyond the scope of this book though I'd love to do that very much. It is highly recommended that you download all of the scanners listed in Exhibit 3-4 and try them out on your computer. Unless you use these tools to scan your own computer, it is very important that you had the permission to scan in written form before thinking about scanning anything at all.

SAINT

SAINT is a Security Administrator's Integrated Network Tool released by World-Wide Digital Security. It is a play on the name of the passé vulnerability scanner SATAN (Security Administrator Tool for Analyzing Network). This Linux and UNIX compatible vulnerability scanner is surprisingly simple and easy to install — it isn't named SAINT for no reason obviously. During the installation, you will be prompted with the license agreement and it is important that you read and agree to all the terms mentioned there in order for the installation to continue. Should the installation complete successfully, you can simply start the program by clicking on the SAINT icon placed on your desktop or executing the script named `saint` located in the `saint-x.x/scripts` directory.

Since SAINT is a browser-based security scanner, you will need to have a browser, whether it is a graphical browser like Firefox or a text browser like Lynx, for SAINT to work with. By default, SAINT uses Netscape as its default browser but you can easily change that by editing the file named `saint-x.x/config/paths.pl` and changing the `/usr/bin/netscape` path to point to your preferred browser. You will also need to have a working PERL environment 5.004 or higher in order for SAINT to run at all.

The interface of SAINT is neat though somewhat complicated. However, if you want to get on with business fast and start scanning, you can just click on the `Scan Set-Up` tab and ignore the rest to start selecting your scan target and specifying how you want it to be scanned. The target can be denoted by a single IP address, a domain name, an IP address range, a class C subnet, or a selective file — which contains addresses of the target systems that you wish to scan. There are totally six different scanning levels for you to choose from, including Discovery, Light, Normal, Heavy, Heavy+, and SANS Top 20. Heavy+ is the most aggressive or "ruthless" scanning level which enables SAINT to perform a thorough scan against the target, irrespective of the stability of the target system. SANS Top 20 is a custom scanning level that will scan the target for the 20 most critical vulnerabilities as rated by SANS. If you do not wish to use any of the predefined scanning levels, you are more than welcome to create a custom scanning level that meets your specific needs. After you're done with selecting your scan target and scanning level, you may press Scan Now and let SAINT work its magic.

When the scan is completed, you should proceed to the Data Analysis section to configure how SAINT should display the vulnerability report. If the report is supposed to be handed over to people with limited knowledge about computer systems, such as the chief executive, then the Executive Report type should be selected. In case you may want to know, a report filled with technical and "cryptic" details is also available just for the geeks like you and me.

Overall, SAINT is a complete security scanner in every sense of the word. Its current vulnerability database (SAINT version 5) is nothing but impressive, comprising detailed information on over 1000 different vulnerabilities in many categories, including DNS, Web, Mail, Windows OS, FTP, and other well-known vulnerabilities. Not only that, its

included component SAINTwriter is so flexible that it can generate a “readable” vulnerability report virtually for everyone within the organization, regardless of his or her role. However, SAINT is a commercial security scanner and you are required to obtain a license if you intend to use SAINT to scan more than two target hosts and for a period longer than 15-day. For all the available licenses and information on how to obtain one, you can visit http://www.wwdsi.com/order.html.

NESSUS

No discussion on vulnerability scanning would be complete without the mention of Nessus — a great and popular open source vulnerability scanner. Originally started by Renaud Deraison in 1998, the Nessus Project’s rationale is to provide network administrators and security professionals alike a free, powerful, and flexible security scanner that can help discover known vulnerabilities quickly and reliably. Nessus is practically available at no cost; however, that is only applicable if you are using Nessus to scan your system or network. If you plan to use Nessus to provide security consulting or auditing service, you will need to obtain the Consultant license that requires a typical annual subscription fee. Refer to Exhibit 3-4 for the download location of Nessus.

Nessus is based on the client-server model. The Nessus client is responsible for connecting to the Nessus server, configuring scanning parameters, and reporting the scan results. The Nessus server, on the other hand, is the one who as to carry out actual vulnerability tests and know nothing about reporting or configuring scanning options. While the Nessus server is only available for UNIX and Linux platforms, the Nessus client program is extremely versatile and fully compatible with Microsoft Windows or any UNIX-like operating systems that support GTK+ GUI.

Each vulnerability check in Nessus is conducted via what called a *plugin* — which is a simple script written in Nessus Attack Scripting Language (NASL) that checks for a specific vulnerability on a system. Current version of Nessus 2.2.2 comes with over 6000 different plugins, covering tests for both remote and local vulnerabilities. It can’t be emphasized enough that you must update the plugins as regularly as possible so that Nessus can check the target system for new vulnerabilities in addition to the old ones. However, do note that only *Direct Feed* Nessus users, who pay annual subscription fees, are entitled to obtain the latest and reliable plugins created by the developers of Nessus. Users who go for other options, such as the GPL (GNU Public License) Feed and Registered Feed, can still update their plugins at no cost but the new plugins are only available for download several days or even months later. If you do not have all the time in the world to wait for such updates, you are free to develop your own plugins using NASL or C.

Setting up Nessus server is straightforward with the help of a ready-made installation script. Before you can go on and start the installation, it is worth bringing to your attention that Nessus server will work much better if you had the following security-related programs preinstalled: `nmap`, `hydra`, and `nikto` or `whisker`. Without them Nessus will still work, but it just won’t work as well.

Before starting the Nessus server using the `nessusd -D` command, you will need to run the `nessus-adduser` script first to add a user. This particular feature allows administrators of the Nessus server to control Nessus' users and monitor scanning activities from a single central location. The script will ask which authentication method you'd prefer to use, password or certificate — password is the simplest and more common choice. You are then required to create simple rules for the server to control which hosts a Nessus user is allowed and not allowed to scan. Since vulnerability information is sensitive in nature, it is obvious that you don't want such information to be learnt by your adversaries. Therefore, it is imperative that you run `nessus-mkcert` afterward to generate a certificate that will be used later to encrypt the traffic between the client and the Nessus server.

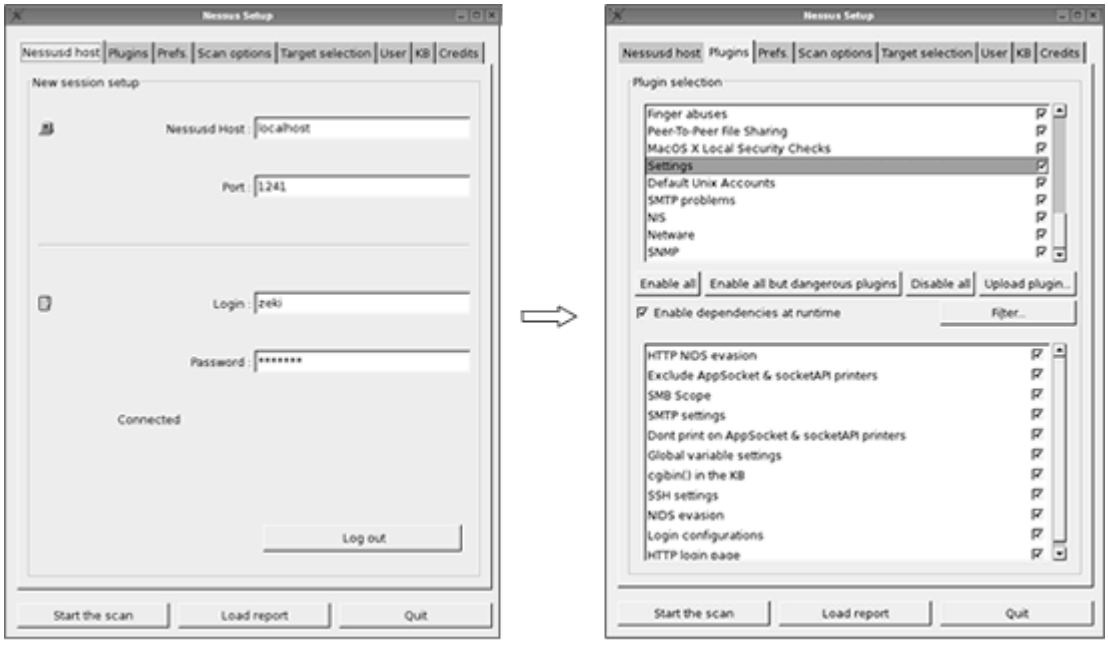

Figure 3-13. The Nessus Client (Linux)

Once you are done setting up the Nessus server, it is now time to fire up the Nessus client and start scanning. Under Linux, the Nessus client is named `nessus` — normally installed at the same time with the Nessus server. If you're using Windows however, you should download and use NessusWX instead.

The first thing to do when the Nessus client started is to login to the Nessus server using the previously created username and password or certificate (Figure 3-13). The address and port — default 1214 — of the Nessus server should also be set accordingly. In this example, the address of the Nessus server is specified as `localhost` because the client is started from the same machine that runs `nessusd` (Nessus server).

Having done logging in, you can then browse to the `Plugins` tab to choose which plugin category should be enabled for the scan. All the 6000 different plugins are divided into many small categories, helping you to manage and configure your scan more efficiently. Imagine sorting and selecting all these 6000 plugins individually every time you want to scan a network or system; it doesn't sound fun at all does it? By default, all the

"dangerous" plugins are disabled. Dangerous plugins are those aggressive vulnerability tests that may crash the remote system or network as a result and you probably don't want to do this against critical systems. Nonetheless, in real life, you can be sure hackers are not going to be lenient to your network at any rate, and therefore, having dangerous plugins disabled may only give you a false sense of security about your network.

After choosing the plugins that should be enabled for the scan, you can safely skip the `Prefs.` and `Scan options` tabs and start specifying your scan target under the `Target Selection` tab. Your scan target can be specified by a domain name, an IP address, a range of IP address, CIDR (Classless) notation, or a file containing a list of targets. When everything seems to be all set and ready, you can click on `Start the scan` and waits for Nessus to work it magic.

Upon completion of each scan, you will be prompted with a Nessus "NG" Report, featuring information that Nessus discovers about the scanned host, including open ports, security note, vulnerability details, and vulnerability remediation. From the report window, you are also given the options to save the report in many formats, such as HTML, XML, LaTeX, plain text, or HTML with Pies and Graphs.

Nessus is far more powerful than you think. In order to take full advantages of this scanner, it is recommended that you learn how to use the Nessus Attack Scripting Language to develop your own plugins. That way, you can always help yourself stay up-to-date with the latest vulnerabilities without having to rely on the developer's mercy.

NIKTO

Nikto is a Perl based security scanner that is dedicated to perform over 3100 security vulnerability tests against web servers, looking for known CGI vulnerabilities, insecure default files and configuration, misconfiguration, and outdated software. Since Nikto is written in Perl, it is therefore very flexible and can be used on almost any platform that supports Perl, including Microsoft Windows and all flavor of Linux. Note, Nikto needs the LibWhisker Perl module (by the renowned ex-hacker Rain Forest Puppy) for all of its network functionality, so make sure you have the module in place first otherwise Nikto won't work at all. If your version of Nikto somehow doesn't come with the module, you can easily download it here http://www.wiretrip.net/rfp/lw.asp.

If there is an Oscar award for the easiest-to-use web security scanner, Nikto should be nominated because it works "out of the box" with absolutely no installation or compilation needed. After unpacking `nikto-current.tar.gz`, you can start scanning right away by executing `nikto.pl` and using the `-h` switch to specify your target web server. The example below instructs `nikto.pl` to perform a generic full scan against the 192.168.0.12 web server and write the outputs to a file named `result.html` afterward. For more information on how you can use other advanced Nikto options to enhance your scan result, you should consult with the manual located in the `docs` subdirectory.

```
[bash]# ./nikto.pl -h 192.168.0.12 -g -F html -o result.html
```

Nikto optionally can perform port scanning based on its internal port scanning engine. However, if you prefer to have Nikto use `nmap` for all of its scanning activities, you need to edit the configuration file of Nikto, `config.txt`, and specify the path where `nmap` can be found. The configuration file `config.txt` is also where you can set general options that will affect all of the scans run by Nikto in future, such as proxy server, default HTTP version, CGI directory search path, and so on.

As new vulnerabilities are being discovered everyday, it is important that you keep the Nikto's vulnerability database current by downloading updates (`-update`) at least once a week or every two weeks. You may also develop your own plugins for Nikto to use if you wish.

GFI LANGUARD NETWORK SECURITY SCANNER

GFI LANguard Network Security Scanner (NSS) 6.0 is one of the most popular and powerful vulnerability scanners available for Windows platform. Even though its vulnerability database may not be as inclusive as of other scanners listed in the Exhibit 3-4, GFI LANguard NSS 6.0 performs just equally well. It can carry out more than 330 different vulnerability checks against its scan target, though most of them are to look for Linux and UNIX related vulnerabilities. Yet, GFI LANguard NSS is still worth a look because besides being a highly rated security vulnerability scanner itself, it is also a fully functional Windows patch management program.

By the looks of it, the interface of GFI LANguard NSS may seem to be confusing and hard to manipulate; but don't let that scare you off, since it only takes you a mere few minutes to setup your scanning options and get the scanner to work.

The leftmost windowpane, as can be seen in Figure 3-13, is where you can configure your scanning options and access to your vulnerability report and information gathering tools. Under the Security Scanner node, you are given options to use scan filters to pull out the information which you are explicitly looking for from the report. If that still isn't granular enough, you are free to create your own scan filter anytime you want.

The Tools node gives you access to various network reconnaissance tools, including DNS lookup, `traceroute`, `whois`, SNMP enumeration, and SQL server auditing. Using these tools before performing a full-blown vulnerability scanning is highly recommended because it will help you have a better understanding about the target network. The better you understand your target, the more effective and efficient your scan is going to be. But hey, aren't you supposed to have done this already in the earlier phases of scanning?

The Configuration node is where you must pay careful attention because the results of your vulnerability scanning depends greatly on what you do and set in the Scanning Profiles option. The Scanning Profiles allows you to define what exactly GFI LANguard NSS should scan when being asked. By default, the Default profile — the most comprehensive profile that enables all scanning types — will be used. When the Default profile

is used, the scanner will first perform TCP port scanning and then sequentially scan the target system for known CGI, DNS, Mail, FTP, RPC, and other vulnerabilities. In addition, it will also attempt to enumerate user accounts, shares, network devices, policies, and et al. Nonetheless, if the Default profile is too much for you, you can easily create new or select other specific scan profile to be used instead.

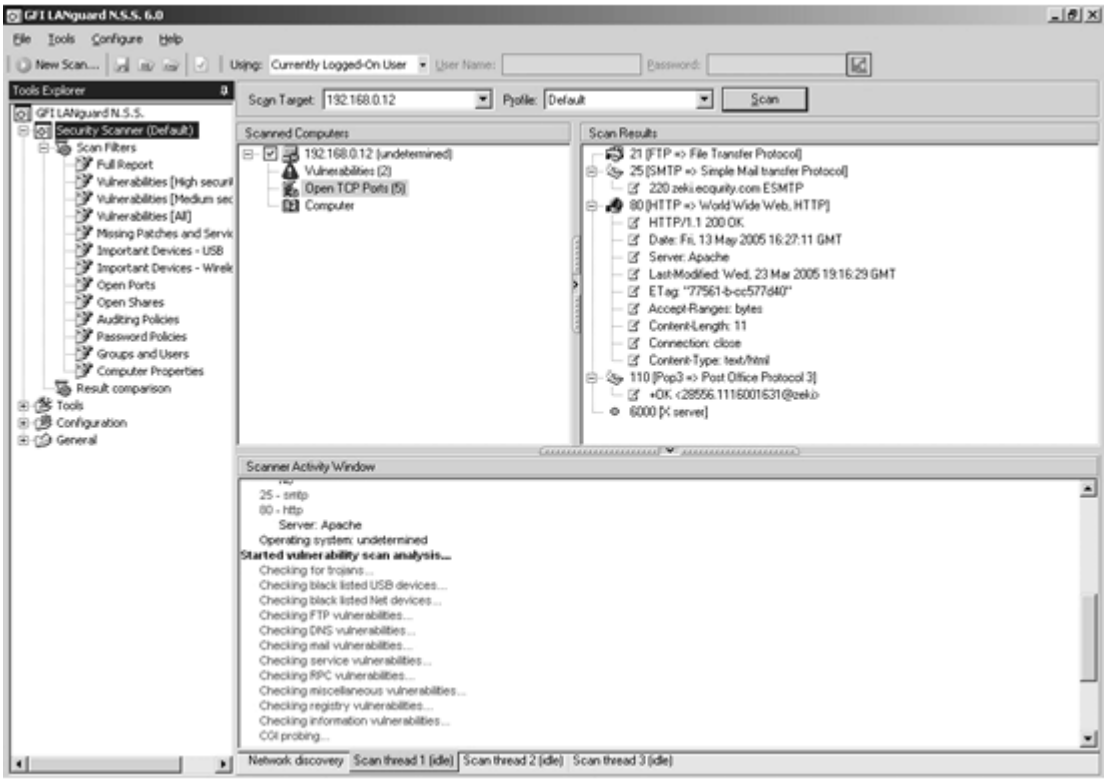

Figure 3-14. Full Scan with GFI LANguard NSS 6.0

To start scanning, you just need enter the IP address or domain name of the target into the Scan Target box located on the main interface of the scanner, choose the scanning profile that you wish to use, and then press Scan. You can also use a specific range of IP addresses or a windows domain to specify your scan target instead. For more information on how to use GFI LANguard NSS 6.0 and its supported features, you should consider reading the official manual located in the installation directory of GFI LANguard NSS.

Perhaps the only real downside of GFI LANguard NSS is that it is not free. You are required to pay $375 for a license that enables you to scan up to 25 different IP addresses. Other licensing options, such as the Consultant license that allow you to scan unlimited computers, are also available for the price of $1999. If you wish to learn more about other licensing schemes, please visit http://www.gfi.com.

RETINA

Released by eEye Digital Security — a renowned security group responsible for

discovering major vulnerabilities in multiple Microsoft products — Retina is a very versatile and powerful security scanner for Windows shipped with many outstanding features, such as fast scanning speed, flexible vulnerability report generating engine, automated vulnerability remediation solution, and the unique artificial intelligence testing engine. Since Retina is a commercial security scanner, you are required to purchase and obtain a proper license before using it. There are many licensing options available for you to choose from, ranging from consultant license to enterprise-wide license. For more information, you can visit http://www.eeye.com/html/products/retina/pricing/.

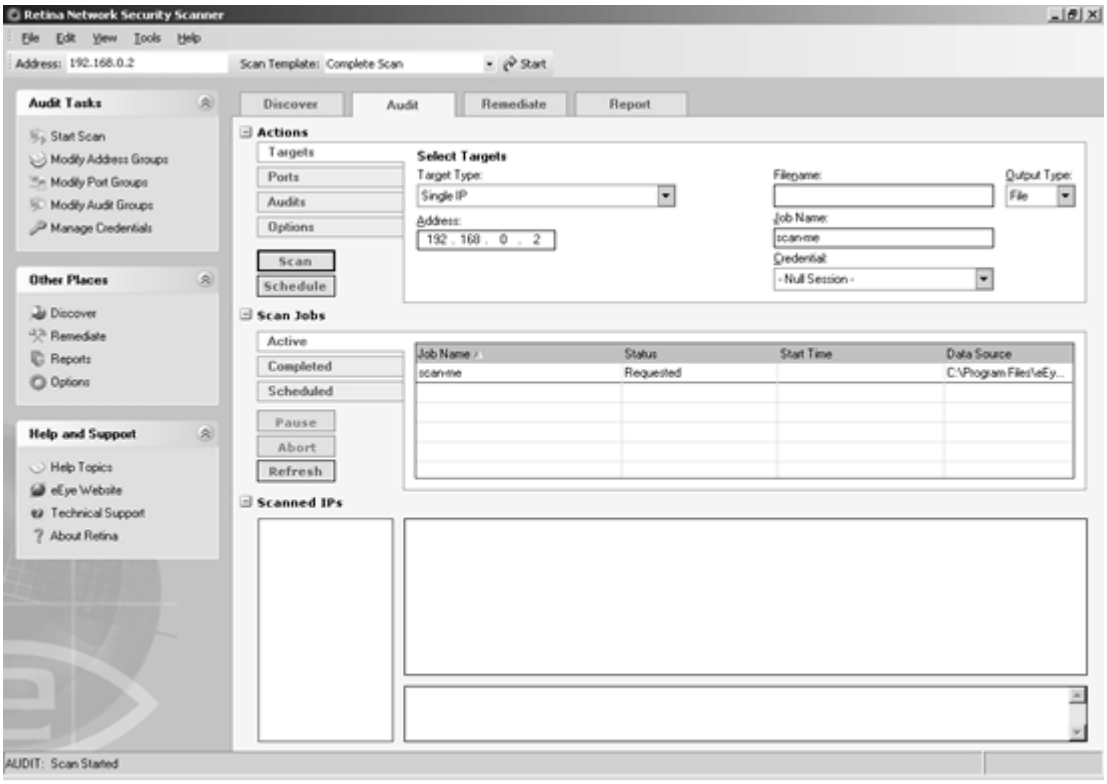

Figure 3-15. Retina Network Security Scanner in Action

Retina's interface is somewhat dense but still very well organized with the famous and "pretty" Windows XP look (Figure 3-15). The left windowpane provides shortcuts to various tasks and options, helping you get around and access the desired functions more quickly. The main window of Retina's interface, which has four different tabs to choose from, is where you will be working with most of the time.

The Discovery tab's task, as many of you would have guessed, is responsible for finding live hosts or systems on the target network. In addition to the capability of sending TCP, UDP, and ICMP packets for discovery purposes, Retina's Discovery function can also get information about the OS, Reverse DNS, MAC address, and NetBIOS Name, of the identified live hosts.

Most of your scanning options will be located under the Audit tab. From the Actions pane, you can specify your scan target, select which ports and security vulnerabilities

Retina should check for, and configure miscellaneous options. Your scan target can be denoted in the form of a single IP address, domain name, an IP range, CIDR notation, and so on. Once you're done with the configuration part, you can start scanning immediately or schedule it for a later time through the Scan Jobs pane.

As aforementioned, Retina offers a remediation solution for each of the discovered vulnerabilities. The solution can simply be an URL referring to where you can download the relevant patches but it can also be an actual fix proposed by Retina. Through the Remediate tab, you can have your remediation report generated based on the severity or category of the vulnerability. If you keen on to have Retina automatically fix some of the vulnerabilities, you can do so via the Scanned IPs pane in the Audit tab. However, you are required to have administrator privilege on the vulnerable system for Retina to access the registry and other administrative functions to remedy the vulnerabilities in question.

The Report tab helps you create a more visualize and understandable vulnerability assessment report for the executive officers — people who do not want to understand the detailed aspects of remediation and vulnerabilities. From here, you can precisely control what should be included in the report and in which format the report should be saved as.

Perhaps the feature that really sets Retina apart from other security scanners is the intelligent Common Hacking Attack Method (CHAM) engine. The CHAM engine helps Retina identify vulnerabilities that were and are completely *unknown* to the public and the respective software vendor, so called *0-day* vulnerabilities. In this prospect, Retina is made to think more like an actual hacker trying to penetrate a system rather than just an insentient security scanner. With the CHAM technology, Retina is capable of identifying potential buffer overflow, format string, and other invalid user input vulnerabilities.

ISS INTERNET SCANNER

Internet Security Systems (ISS) Internet Scanner is another leading and feature-rich vulnerability scanner for Windows platform. Initially started by Christopher Klaus as a freeware open source scanner in 1992, ISS Internet Scanner has transformed into one of the most trusted and influential commercial vulnerability scanner available to date, and being used by more than 11,000 customers worldwide is sufficed to say it all. If you think the let down of this scanner is its hefty price tag, then wait until you see its key features and you will probably say the price tag is justifiable after all.

Unlike other security scanners where vulnerability reporting is usually not weighted as much as vulnerability scanning, ISS Internet Scanner 7 emphasizes the reporting task equally well with more than 74 predefined reporting templates, giving its users the ability to distribute the information to any people at any level of the organization. The following are some additional key features of the ISS Internet Scanner (this list isn't meant to be inclusive):

- Real-time Display – As ISS Internet Scanner is scanning, this feature enables you to monitor the progress and view the scanning results live. In doing so, you can quickly

identify vulnerabilities on the scanned target without having to wait for the report or the scan to be completely stopped.

- Dynamic Check – ISS Internet Scanner performs vulnerability checks based on the target system's OS type, thus improves the scanning speed and accuracy.
- Wide Test Coverage – ISS Internet Scanner can identify more than 1,300 types of network devices and thousands of vulnerabilities spanned across 36 different categories.
- Flexible Scanning Options – 20 predefined scanning policies allow you to carry out your scan quickly. Advanced users can customize the scanning policy and write their own vulnerability checks with FlexiCheck.
- Detailed Vulnerability Information – A thorough description and step-by-step remediation solution are provided for each vulnerability listed in the database.

For a comprehensive feature list and an evaluation copy of the current ISS Internet Scanner, you should visit the ISS homepage as listed in Exhibit 3-4 on page 58. Be warn, ISS Internet Scanner is a resource hunger so make sure your computer has enough CPU and memory power for it to burn.

I FINISH SCANNING, NOW WHAT?

Now that you finish scanning a system for vulnerability, it is time for you to analyze the results and find or write the right exploit to take advantages of the vulnerability. Easier to say than done, where can I find exploit and what does it take to write one? Well, you can easily find exploit code at many popular security sites, such as Packet Storm Security (http://packetstormsecurity.org), SecurityFocus (http://www.securityfocus.com), and the prolific French security site FrSIRT (http://www.frsirt.com/exploits/). Writing or creating exploits require you to have an in-depth knowledge of computer programming and networking and you will be shown how to do all this later in the book.

COUNTERMEASURE: STAYING ALERT AND UPDATED AT ALL TIME

Keeping all your servers and network up-to-dated with the latest fixes and patches is perhaps the best and most cost-effective countermeasure there is to vulnerability scanning. By applying security patches to the defective and vulnerable network service or device on your network, the known vulnerabilities are practically removed and their associated risks are therefore mitigated tremendously. For the latest information on new vulnerabilities, security threats, and patches, it is highly recommended that you subscribe to SecurityFocus — the de facto mailing list for security professionals and enthusiasts.

Misconfiguration and insecure default configuration also play their parts to many successful hack incidents over the past years, most likely due to the lack of technical knowledge and skills of the network administrators. Consequently, it is important that the company provide its network administrators with access to the appropriate software-training workshop for them to update and possess the necessary skills. Much of the knowledge on configuration or hardening a service can also be gained from public security forums or mailing lists such as SecurityFocus as mentioned above.

Having said known vulnerabilities can be eradicated by staying updated, it is worth bringing to your attention that 0-day vulnerabilities do exist and staying updated is only one of the many steps that you must take to strengthen the security of your network. To fight against 0-day vulnerabilities you will need to use Intrusion Prevent Systems (IPS). IPS is a new concept in this security arena, differing from Intrusion Detection Systems (IDS) in that they can actually predict and prevent the intrusion before it happens rather than just letting you know about it — which is not very helpful to many people. Besides, IPS do not rely on signature files, and therefore, they are very flexible and can protect a system from both known and unknown vulnerabilities. For Linux and UNIX servers, GRsecurity (http://www.grsecurity.net) is the most popular and powerful host-based IPS there is. It can protect the servers from buffer overflow, race condition, and other types of exploitation. There are also several host-based IPS available for Windows platform, including NGSec StackDefender (http://www.ngsec.com/products/stackdefender), Prevx (http://www.prevx.com), and Qwik-Fix Pro by Pivx (http://www.pivx.com).

Like other scanning methods, vulnerability scanning can be detected with the help of any decent host-based and network-based IDS, such as BlackICE or Snort — a much more favorite candidate for the job mainly because it's free, regularly updated, and has many more detection rules than BlackICE. Regularly examining log files is another way to determine if an intrusion is about to happen, because security scanners like Nikto or Nessus tend to leave thousands of different entries in the log files as the aftermath.

Assessing the security of your network from a third point of view, using any of the vulnerability scanners introduced previously, is a worthy course of action. Vulnerability scanning should be conducted on a regular basis but with great cares as some of the more intrusive checks may cause Denial of Service (DoS) or interrupt normal network's operation. Without prior written permission from the upper management, you should probably forget about scanning and go enjoy your frappuccino, as it is much safer and more enjoyable than scanning a network without authorization.

MISCELLANEOUS

PROXY & ET AL – SURFING AND SCANNING ANONYMOUSLY

You have seen the term "proxy server" mentioned repeatedly throughout the whole chapter but what exactly is a proxy server? A proxy server is an intermediary device between two computers, client and server. It is responsible for validating and processing a connection request from the client and then relaying such request to the specified server on the client's behalf. Upon receiving the response, the proxy server in turn acts on the behalf of the server and forwards the response to the client. By doing this, the connection between the client and the server can still be established but the two computers just won't be able to see each other or communicate in a direct, an end-to-end fashion. Though in practice proxy servers may be used for a variety of purposes (beneficial), the four listed below are the most common ones:

- **Content Filtering and Security** – A typical implementation of a proxy server

will work at the application layer, layer 7, of the OSI model. As a result, the proxy can work much more intelligently by dissecting a packet and making a filtering decision based on its contents, rather than just its header.

- **Caching** – The web browsing performance is dramatically improved as the proxy server can provide the requested page to the client instantly from its cache. If the page cannot be found in the cache, the proxy server will attempt to fetch the page from the remote server, returns the page to the client, and then put the page in cache for later use.
- **Sharing Connection and NAT** – A proxy server allows many computers on a local network to connect to the Internet through it using a single IP address — an efficient IP allocation solution for current shortage of IPv4 addresses. Since all the connections between the local and remote network are established through the proxy server, the proxy server can provide NAT (Network Address Translation) functionality, help obscure and prevent leakage of internal IP addresses.
- **Anonymizing** – As two computers virtually can't see each other, the proxy server can offer some levels of anonymity, particularly for web browsing.

For the wary web surfers and hackers, the anonymity and protection offered by the proxy server is simply too hard to resist. How much protection a proxy server offering is dependent much on which mode it is willing to operate, and of course, the generosity of the administrators. A proxy server typically works in either one of these two modes: *transparent* and *anonymous.*

In transparent mode, many fields or variables in the client request's packet header, such as the HTTP_FORWARDED_FOR and REMOTE_ADDR fields, stay unchanged and are forwarded along with the proxy's own header to the remote server. As these two fields would contain the real IP address of the client, it is obvious that transparent proxy servers only make a little effort to help the client stay anonymous.

Things look much brighter for the client when the proxy server works in anonymous mode, however. The proxy server replaces the header of the request packet with its own before handing the request over to the server, and therefore, the IP address and other information about the client would be completely unknown to the target server. To check whether a proxy server is operating in transparent or anonymous mode, you should visit http://stealthtests.lockdowncorp.com and pick the "All Remote Environmental" test — a thorough and complete proxy test.

If you use a proxy server to protect your privacy, hackers also use it for the exact same reason — covering up their real IP addresses and malicious activities. In practice, hackers often perform vulnerability scanning behind a proxy server so that their real IP addresses would not be logged by the remote server and IDS. Besides, by exploiting the advantages of a proxy server, internal hackers or naughty network users can evade firewall rules at relative-ease and access sites that would normally be blocked otherwise.

ANONYMIZER

Anonymizer is a service providing a similar protection to a proxy server to help keep your identity private from the hostile Internet. While using a proxy server usually entails some changes in your browser settings, anonymizer requires absolutely nothing else than a browser. All your web surfing activities will be started from the web interface of the anonymizer service. There are many Internet companies providing this kind of service, but http://www.anonymizer.com and http://www.safeweb.net are the typical ones.

SOCKSCHAIN

SocksChain by Ufasoft (http://www.ufasoft.com) is a program guaranteed to satisfy even the privacy fanatics. It allows you to establish a connection, with a high-degree of anonymity, to another computer through many proxy servers, rather than just one. By having the attack routed through many different proxy servers, as you will see in the discussion of SocksChain, the hackers literally make it impossible for the authority to determine where the attack really originated.

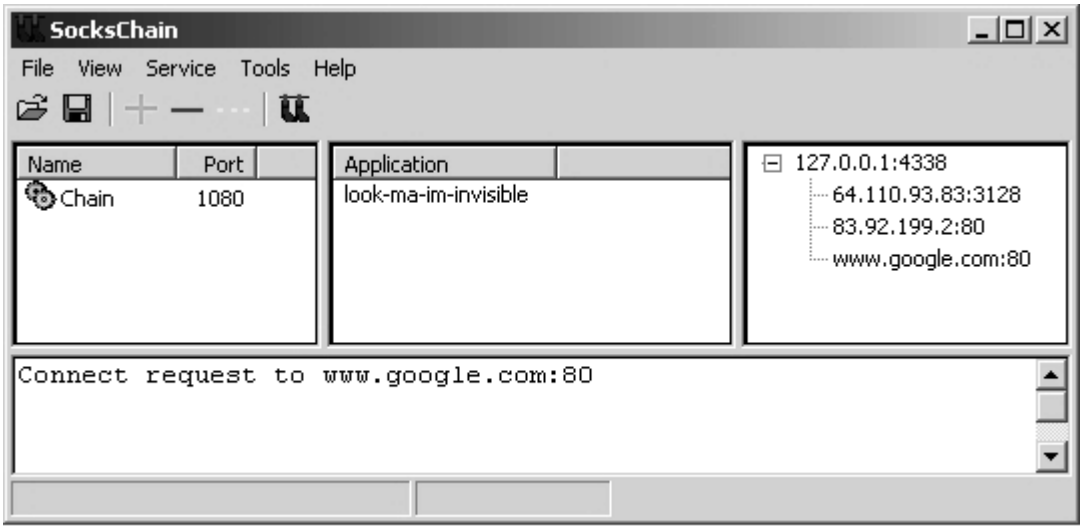

Figure 3-16. Reaching Google anonymously with SocksChain

SocksChain works by running as a local proxy server on your computer and accepting TCP connection queries on the default port 1080. When a query received, the program sends the query through a chain of proxy servers. The only proxy that has anything to do with the client's IP address is the first proxy in the chain. That proxy in turn replaces the client's IP address and other identifying information with its own and forwards the query to the second proxy in the chain, the second proxy then forwards the query to the third one, and the forwarding goes on and on until the end of the proxy chain is reached.

The premise behind this is that if one computer on the other end wants to trace back to where the query originated, it has to do so in a reverse order and examine all the log files in-between the end of the chain and the very top. If any proxy server in the chain does not keep its log file, voila, the chasing and tracing game is virtually over.

Figure 3-16 shows an Internet application named `look-ma-im-invisible` using SocksChain to reach www.google.com. As you can see from the top right window, the query is routed through two proxy servers before it can reach its final destination — Google. SocksChain can of course work with as many proxies as you wish but keep in mind that going through so many proxy servers can slow down your connection tremendously. To learn how to use SocksChain, you should study the help file come with the program.

HTTP TUNNEL – THE FIREWALL'S SUCKER PUNCH

Over the years, many people have been diving into the misconception that firewall is a perfect solution to solve all the security problems, yet nothing can be further from the truth. Although it is unquestionable about the contribution of a firewall to the security of a network, you simply cannot lay all the trusts and rely solely on the firewall because just like any other products, firewall is not impenetrable nor can it outsmart the hacker's brain and tactics. HTTP tunnel, as discussed below, is one of the many examples showing why you must remain attentive at any rate and where the firewall may fall.

HTTPORT

HTTPort is a firewall-bypassing tool that creates a transparent tunnel through the firewall or proxy server to allow you to use any of your favorite Internet applications, such as eMule or IRC, through such tunnel. However, it is important that the firewall must at least allow access to HTTP or web service in order for HTTPort to work at all, though it's perfectly ok for the firewall to block everything else. HTTPort is a giftware, which practically means you can download it for free at http://www.htthost.com/httport.boa.

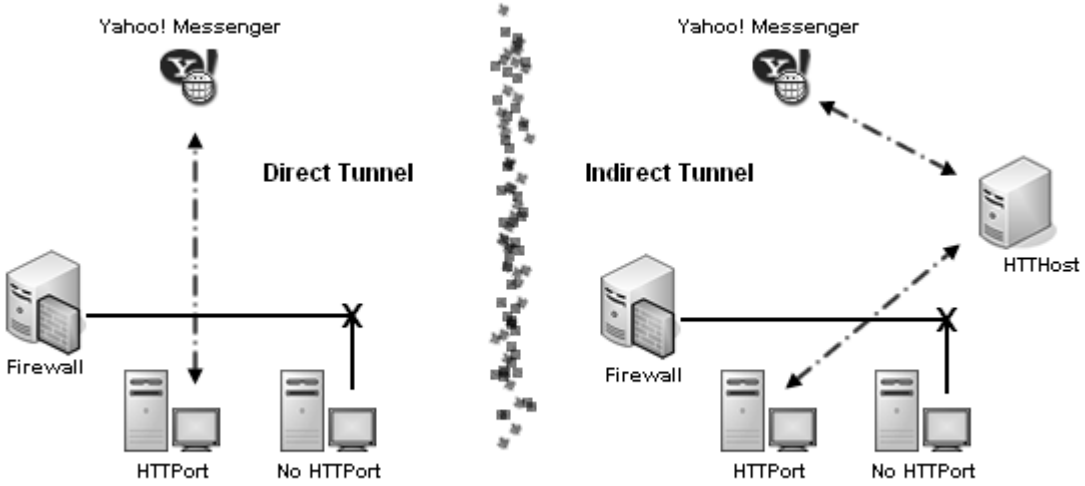

Figure 3-17. Tunnel with HTTPort

HTTPort helps you build either a *direct* or an *indirect* tunnel through a proxy server. For a direct tunnel, HTTPort creates a tunnel itself and requires no server program, but the catch here is that the proxy server must support the CONNECT HTTP method — which is a very rare setup that you will hardly find one these days. Hence, to keep things

in context, this section will only discuss the use of indirect tunnel.

For an indirect tunnel, HTTPort directs traffic to another computer that has HTTHost installed and located out-side the firewall-protected network. HTTHost is a special server program that is supposed to interpret and forward all the traffics that it receives from HTTPort to the appropriate destination. Figure 3-17 illustrates the difference between direct tunnel and indirect tunnel and it should help those who are still confused by now.

HTTPort only works well when configured properly so it is important that you give yourself enough time to read and understand what it says in the manual. If you don't like to read the manual, then well, here are some quick steps (listed in sequential order) that can help you get HTTPort work quickly:

- First, you need to have a HTTHost server ready, by either setting up one on your own or let HTTPort help you find and use a public server. HTTHost can be downloaded from the same web site as HTTPort.
- You then have to start and configure HTTPort installed on your computer:
    - Under the `Proxy` tab, you need to specify the proxy server that you want to bypass, leave blank if you have direct web access. If you intend to use your own HTTHost, you must specify so in the `Use personal remote host at` box or else HTTPort will try to find and use a public HTTHost instead.
    - HTTPort works by tricking your Internet application into thinking that your PC is the remote server that it needs. When the application sends query to its "remote server", it actually sends the query to HTTPort on your PC, which intercepts and forwards the query through the tunnel. Hence, under the `Port mapping` tab, you must supply correct information for HTTPort to create tunnels for your Internet applications. Typically, there are three parameters that you need to look out for: local port, remote host, and remote port. Remote host and remote port should be the address and port of the server that your Internet application needs to use. Local port is the port which HTTPort will listen for queries. Screenshots of this configuration process are shown in Figure 3-18. When things are all set, you can go back to the `Proxy` tab and start the program.
- Finally, you should start your Internet application and change its settings so that it points to 127.0.0.1: "local port" as its remote server. If you believe you have done everything correctly, the tunnel and your Internet application should work by now and make the firewall or restrictive proxy server look rather ridiculous.

Note, not all applications can give you the luxury to specify a custom address for its remote server, for example, Yahoo! Messenger. In that case, you will need to obtain the address of the remote server (in the form of resolvable hostname) and manually edit the file `c:\windows\system32\drivers\etc\hosts` (in Windows XP) to make the address of the server resolvable to localhost — your computer with HTTPort installed. Hence, if the address of Yahoo! Messenger server is scs.msg.yahoo.com then you should add the following line to the `hosts` file: `127.0.0.1 scs.msg.yahoo.com`.

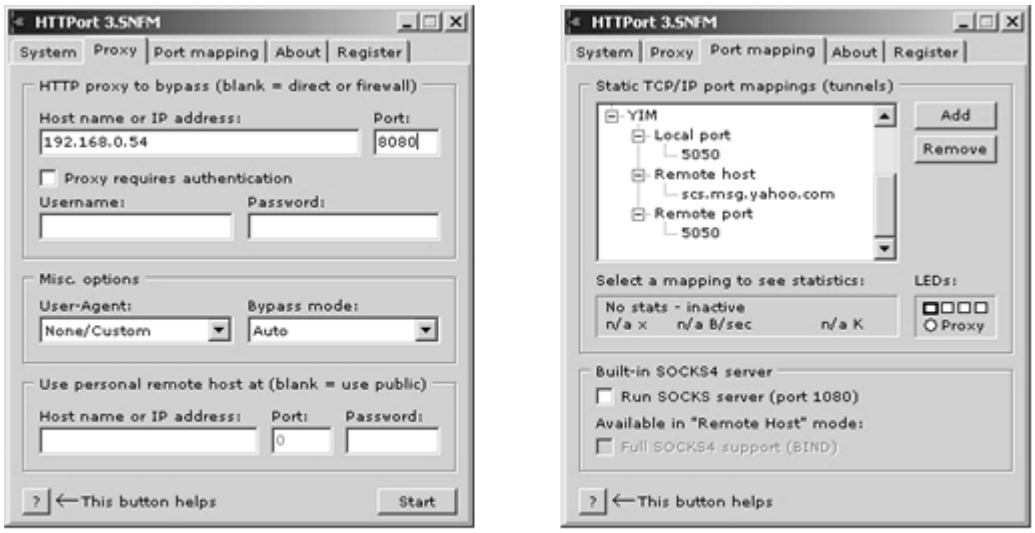

Figure 3-18. Configuring HTTPort

HTTPTUNNEL

HTTPTunnel by Lars Brinkhoff is a similar tool like HTTPort in which it provides users a way to avoid the restrictive firewall or proxy server and connect to outside computers through a bidirectional virtual tunnel. HTTPTunnel can be compiled on almost any Linux platform, available at http://www.nocrew.org/software/httptunnel.html. On the web site, you will also find link to Windows NT port version of HTTPTunnel.

HTTPTunnel is based on a client-server model, comprising of two main programs: `hts` and `htc`. `Hts` is the server component that listens for incoming HTTPTunnel queries, and when a query is made, forwards the query to the defined host and port. It must be installed on a remote computer located outside the firewall protected network, and more importantly, it must be set to listen on a port that is not blocked by the firewall, such as port 80 (where do you think the name HTTPTunnel come from??). For example, the following command will set `hts` to listen on port 80 and forward `-F` any query it receives to port 21 on ftp.block-who.com (you need root privilege in order to bind `hts` to any port smaller than 1024):

```
[bash]# hts -F ftp.block-who.com:21 80
```

The client component, `htc`, is responsible for setting up tunnel through the firewall to the remote computer running `hts`. It is worth to remind you that only one connection is allow using the tunnel at a time. If you want to use two connections, you will need to create two tunnels by running two instances of `hts`.

```
[bash]$ htc -F 2121 -P 192.168.0.54:8080 12.13.14.15:80
```

The preceding command sets `htc` to listen on port 2222 and forward `-F` any query from the *local* computer to port 80 on the `hts` server 12.13.14.15. The `-P` option allows

you to specify the address and port of the restrictive proxy server, omit it if you have direct web access.

Now that the tunnel has been created, you can start using your program and have it connect to its server through the tunnel, which should look something like this:

```
[bash]$ ftp localhost 2121
Connected to localhost.
220 Serv-U FTP Server v6.0 for WinSock ready...
Name (localhost:xxx):
```

SUMMARY

Scanning is the process of establishing a direct connection to a target network and use a wide range of active reconnaissance techniques to extract further information about the target. Scanning is typically divided into five distinctive phases: sweeping, port scanning, application mapping, OS fingerprinting, and vulnerability scanning.

War-dialing and ping sweeping are the two most common sweeping techniques that help hackers discover live hosts on the network. War-dialing involves using a modem to dial a defined range of phone numbers to find a responsive computer that is connected to a modem. Ping sweeping, on the other hand, purely searches for systems that are active or reachable from the Internet by sending ICMP, TCP, or UDP probe packets.

After sweeping, the hackers then proceed to carry out port scanning and application mapping to inventory open ports on the active system and their associated network services. Port scanning is available in many different types, ranging from the least stealthy to the stealthiest, such as, TCP Connect(), SYN scanning, and so on.

As vulnerability scanning and exploit code highly depending on OS information, the hackers will have to find out what OS the remote system is running by performing either passive or active OS stack fingerprinting. With passive stack fingerprinting, the hackers can guess the remote OS merely by sniffing and it is extraordinary stealthy. Contrarily, active stack fingerprinting requires the hackers to proactively send malformed network packets to the remote system to elicit responses and this method of OS fingerprinting is somewhat intrusive and easy to detect.

Vulnerability scanning is the last phase of scanning where the hackers will use automated scanners to identify whether a particular vulnerability is present on the system or the network as whole. Along the way, you were introduced to powerful scanning tools that hackers often use to accelerate and automate the scanning process.

Finally, the chapter ended with introduction of proxy servers and HTTP tunnels. Hackers normally scan a network through a proxy server in an effort to cover up their real IP addresses and avoid being logged or traced down by the target. HTTP tunnel is used to help one pierce through the firewall and use services that should have been blocked by the rules or policy otherwise.