



Scanning and enumeration

INFORMATION IN THIS CHAPTER:

- Objectives
- Scanning
- Enumeration
- Case Studies: The Tools in Action
- Hands-On Challenge

In this chapter, we will lead you through the initial objectives and requirements for performing scanning and enumeration in support of a penetration test or vulnerability assessment. This includes discussing the final phase of reconnaissance, vitality. After that, we will dig into some scenarios in which you will see how you can use these different tools and techniques to their full advantage. Last, we'll do a hands-on challenge so you can test your new (or refined) skills in a real-world scenario.

3.1 OBJECTIVES

In a penetration test, there are implied boundaries. Depending on the breadth and scope of your testing, you may be limited to testing a certain number or specific type of host, or you may be free to test anything your client owns or operates.

To properly scan and identify systems, you need to know what the end state is for your assessment. Once the scanning and enumeration are complete, you should:

- Confirm that IP addresses found in the reconnaissance phase are reachable. This is the “vitality” phase of reconnaissance.
- Be able to identify the purpose and type of the target systems, that is, what they are and what they do.
- Have specific information about the versions of the services that are running on the systems.
- Have a concise list of targets and services which will directly feed into further penetration test activities.

3.1.1 Before you start

Now that we're moving into some penetration testing which will actually "touch" the remote systems, we need to be concerned about the rules around our testing. With any kind of functional security testing, before any packets are sent or any configurations are reviewed, make sure the client has approved all of the tasks in If any systems become unresponsive, you may need to show that management approved the tests you were conducting. It is not uncommon for system owners to be unaware when a test is scheduled for a system.

A common document to use for such approval is a "Rules of Engagement" document. This document should contain at a minimum:

- A detailed list of all parties involved, including testers and responsible system representatives, with full contact information including off-hours contact information if needed. At least one party on each side should be designated as the primary contact for any critical findings or communications.

- A complete list of all equipment and Internet Protocol (IP) addresses for testing, including any excluded systems.

- Rules around compromising systems for deeper penetration.

- Acceptable and unacceptable practices such as compromising physical site security, social-engineering employees, etc.

- Agreement of use of data from compromised systems as well as how this (often confidential) data is stored.

- The time frame for testing:

- The duration of the tests

- Acceptable times during the day or night

- Any times that are prohibited from testing

- Any specific documentation or deliverables that are expected including:

- Documentation around discoveries and methodologies (including tools) used

- Proof of successful penetration/system compromise

- Debriefing schedule

- Limitations of liability for any damage caused by the testing.

Having this type of document agreed to and in place prior to your penetration testing will help ensure that both you and your client are clear on the level and type of testing that will be performed. The more precise and extensive this document is, the less room there is for misunderstandings. One of the worst situations a penetration tester can be in is one where the client is furious because the tester brought down a production system without authorization. Agreeing on the rules and the scope of the testing up front can help prevent that type of issue.

3.1.2 Why do scanning and enumeration?

If you are given a list of targets, or subnets, some of your work has been done for you; however, you still may want to see whether other targets exist within trusted

subnets that your client does not know about. Regardless of this, you need to follow a process to ensure the following:

You are testing only the approved targets.

You are getting as much information as possible before increasing the depth of your attack.

You can identify the purposes and types of your targets, that is, what services they provide your client.

You have specific information about the versions and types of services that are running on your client's systems.

You can categorize your target systems by purpose and resource offering.

Once you figure out what your targets are and how many of them may or may not be vulnerable, you will then be able to select your tools and exploitation methods. Not only do poor system scanning and enumeration decrease the efficiency of your testing, but also the extra, unnecessary traffic increases your chances of being detected. In addition, attacking one service with a method designed for another is inefficient and may create an unwanted denial of service (DoS). In general, do not test vulnerabilities unless you have been specifically tasked with that job. The purpose of this chapter is to help you understand the need for scanning and enumeration activities after your reconnaissance is complete, and help you learn how to best perform these activities with available open source tools. We will discuss the specific tools that help reveal the characteristics of your targets, including what services they offer, and the versions and types of resources they offer. Without this foundation, your testing will lack focus, and may not give you the depth in access that you (or your customers) are seeking. Not all tools are created equal, and that is one of the things this chapter will illustrate. Performing a penetration test within tight time constraints can be difficult enough; let the right tools for the job do some of the heavy lifting.

3.2 SCANNING

No matter what kind of system you are testing, you will need to perform scanning and enumeration before you start the exploitation and increase the depth of your penetration testing. With that being said, what do scanning and enumeration activities give you? What do these terms actually mean? When do you need to vary how you perform these activities? Is there a specific way you should handle scanning or enumeration through access control devices such as routers or firewalls? In this section, we will answer these questions, and lay the foundation for understanding how to use scanning and enumeration to prepare for deeper penetration testing.

3.2.1 Approach

During the scanning phase, you will begin to gather information about the target's purposes specifically, what ports (and possibly what services) it offers. Information

gathered during this phase is also traditionally used to determine the operating system (or firmware version) of the target devices. The list of active targets gathered from the reconnaissance phase is used as the target list for this phase. This is not to say that you cannot specifically target any host within your approved ranges, but understand that you may lose time trying to scan a system that perhaps does not exist, or may not be reachable from your network location. Often your penetration tests are limited in time frame, so your steps should be as streamlined as possible to keep your time productive. Put another way: Scan only those hosts that appear to be alive, unless you literally have “time to kill.”

TIP

Although more businesses and organizations are becoming aware of the value of penetration testing, they still want to see the time/value trade-off. As a result, penetration testing often becomes less an “attacker-proof” test and more a test of the client’s existing security controls and configurations. If you have spent any time researching network attacks, you probably know that most decent attackers will spend as much time as they can spare gathering information on their target before they attack. However, as a penetration tester, your time will likely be billed on an hourly basis, so you need to be able to effectively use the time you have. Make sure your time counts toward providing the best service you can for your client.

3.2.2 Core technology

Scanning uses some basic techniques and protocols for determining the accessibility of a system and gathering some basic information on what the system is and which ports are open on it. The core technologies that we will be focusing on include Internet Control Message Protocol (ICMP) and some elements of how Transmission Control Protocol (TCP) functions and the available TCP flags.

3.2.2.1 How scanning works

The list of potential targets acquired from the reconnaissance phase can be rather expansive. To streamline the scanning process, it makes sense to first determine whether the systems are still up and responsive. Although the nonresponsive systems should not be in the list, it is possible that a system was downed after that phase and may not be answering requests when your scanning starts. You can use several methods to test a connected system’s availability, but the most common technique uses ICMP packets.

Chances are that if you have done any type of network troubleshooting, you will recognize this as the protocol that ping uses. The ICMP echo request packet is a basic one which Request for Comments (RFC) 1122 (www.ietf.org/rfc/rfc1122.txt) says every Internet host should implement and respond to. In reality, however, many networks, internally and externally, block ICMP echo requests to defend against one of the earliest DoS attacks, the ping flood. They may also block it to prevent scanning from the outside, adding an element of stealth.

If ICMP packets are blocked, you can also use TCP ACK packets. This is often referred to as a “TCP Ping.” The RFC states that unsolicited ACK packets should return a TCP RST. So, if you send this type of packet to a port that is allowed through a firewall, such as port 80, the target should respond with an RST indicating that the target is active.

When you combine either ICMP or TCP ping methods to check for active targets in a range, you perform a ping sweep. Such a sweep should be done and captured to a log file that specifies active machines which you can later input into a scanner. Most scanner tools will accept a carriage-return-delimited file of IP addresses.

3.2.2.2 Port scanning

Although there are many different port scanners, they all operate in much the same way. There are a few basic types of TCP port scans. The most common type of scan is a SYN scan (or SYN stealth scan), named for the TCP SYN flag, which appears in the TCP connection sequence or handshake. This type of scan begins by sending a SYN packet to a destination port. The target receives the SYN packet, responding with a SYN/ACK response if the port is open or an RST if the port is closed. This is typical behavior of most scans; a packet is sent, the return is analyzed, and a determination is made about the state of the system or port. SYN scans are relatively fast and relatively stealthy, because a full handshake is not made. Because the TCP handshake did not complete, the service on the target does not see a full connection and will usually not log the transaction.

Other types of port scans that may be used for specific situations, which we will discuss later in the chapter, are port scans with various TCP flags set, such as FIN, PUSH, and URG. Different systems respond differently to these packets, so there is an element of operating system detection when using these flags, but the primary purpose is to bypass access controls that specifically key on connections initiated with specific TCP flags set. Later in the chapter, we will be discussing open source tools including Nmap, a scanning and enumeration tool. In [Table 3.1](#), you can see a summary of common Nmap options along with the scan types initiated and expected response. This will help illustrate some of the TCP flags that can be set and what the expected response is.

3.2.2.3 TCP versus UDP scanning

A TCP connection involves the use of all of the steps involved in the standard TCP three-way handshake. In a standard three-way handshake, that is the following sequence:

```
Source sends SYN to target
Target responds with SYN-ACK
Source responds with ACK
```

After that sequence, a connection is considered established. As we’ve discussed already, stealth TCP scanning makes use of part of the handshake, but never

Table 3.1 Nmap Options and Scan Types

Nmap Switch	Type of Packet Sent	Response if Open	Response if Closed	Notes
-sT	OS-based connect()	Connection made	Connection refused or timeout RST	Basic nonprivileged scan type
-sS	TCP SYN packet	SYN/ACK		Default scan type with root privileges
-sN	Bare TCP packet with no flags (NULL)	Connection timeout	RST	Designed to bypass nonstateful firewalls
-sF	TCP packet with FIN flag	Connection timeout	RST	Designed to bypass nonstateful firewalls
-sX	TCP packet with FIN, PSH, and URG flags (Xmas Tree)	Connection timeout	RST	Designed to bypass nonstateful firewalls
-sA	TCP packet with ACK flag	RST	RST	Used for mapping firewall rulesets, not necessarily open system ports
-sW	TCP packet with ACK flag	RST	RST	Uses value of TCP window (positive or zero) in header to determine whether filtered port is open or closed
-sM	TCP FIN/ACK packet	Connection timeout	RST	Works for some BSD systems
-sI	TCP SYN packet	SYN/ACK	RST	Uses a “zombie” host that will show up as the scan originator
-sO	IP packet headers	Response in any protocol	ICMP unreachable (Type 3, Code 2)	Used to map out which IPs are used by the host
-b	OS-based connect()	Connection made	Connection refused or timeout	FTP bounce scan used to hide originating scan source
-sU	Blank User Datagram Protocol (UDP) header	ICMP unreachable (Type 3, Code 1, 2, 9, 10, or 13)	ICMP port unreachable (Type 3, Code 3)	Used for UDP scanning; can be slow due to timeouts from open and filtered ports

Table 3.1 Nmap Options and Scan Types (Continued)				
Nmap Switch	Type of Packet Sent	Response if Open	Response if Closed	Notes
-sV	Subprotocol-specific probe (SMTP, FTP, HTTP, etc.)	N/A	N/A	Used to determine service running on open port; uses service database; can also use banner grab information
-O	Both TCP and UDP packet probes	N/A	N/A	Uses multiple methods to determine target OS/firmware version
-sn	N/A	N/A	N/A	Skips port scan after host discovery.

completes the connection. In a stealth scan, the final ACK is never sent back to the target thus the connection is not established.

Scanning UDP is more difficult as it is a connectionless protocol and does not use a handshake like TCP. With UDP, the following sequence is used:

Source sends UDP packet to target
Target checks to see if the port/protocol is active then takes action accordingly

This makes scanning UDP ports especially challenging. If you receive a response, it will be one of three types: an ICMP type 3 message if the port is closed and the firewall allows the traffic, a disallowed message from the firewall, or a response from the service itself. Otherwise, no response could mean that the port is open, but it could also mean that the traffic was blocked or simply didn't make it to the target. While it's typically faster and more productive to perform TCP scans, it can sometimes be worth the time and effort to perform a UDP scan as well. Many administrators tend to focus more on securing TCP-based services and often don't consider UDP-based services when determining their security policies. With this in mind, you can sometimes find (and exploit) vulnerabilities in UDP-based services, giving you another potential entry point to your target system.

3.2.3 Open source tools

To start our discussion on open source tools in this chapter, we'll begin by discussing tools that aid in the scanning phase of an assessment. Remember, these tools will scan a list of targets in an effort to determine which hosts are up and which ports are open.

3.2.3.1 Nmap

Port scanners accept a target or a range as input, send a query to specified ports, and then create a list of the responses for each port. The most popular scanner is Nmap, written by Fyodor and available from www.insecure.org. Fyodor's multipurpose tool has become a standard item among pen testers and network auditors. The intent of this book is not to teach you all of the different ways to use Nmap; however, we will focus on a few different scan types and options, to make the best use of your scanning time and to return the best information to increase your attack depth.

Nmap USAGE

How to use:

`nmap [Scan Type(s)] [Options] Target(s)`

Input fields:

[Scan Type] is the type of scan to perform. Different scan options are available and are discussed throughout this chapter.

[Options] include a wide variety of configuration options including DNS resolution, use of traceroutes, and more.

Target is the target specification which can be a single host, a list of host names or IPs, or a full network.

Output:

Displays host information to the screen depending on scan type and options selected including accessibility of the host, active ports, and fingerprint data. There are also options available to output this data to a file.

Typical output: (extract)

```
root@bt:~/nmap_scans# nmap -sn --send-ip 192.168.1.0/24 -oA
nmap-sweep
Starting Nmap 5.30BETA1 (http://nmap.org) at 2010-08-01 10:17 CDT
Nmap scan report for 192.168.1.1
Host is up.
Nmap scan report for 192.168.1.100
Host is up (0.061s latency).
MAC Address: 00:0C:29:67:63:F5 (VMware)
Nmap scan report for 192.168.1.110
Host is up (0.0047s latency).
MAC Address: 00:0C:29:A2:C6:E6 (VMware)
Nmap done: 256 IP addresses (3 hosts up) scanned in 89.75
seconds
```

3.2.3.1.1 Nmap: ping sweep

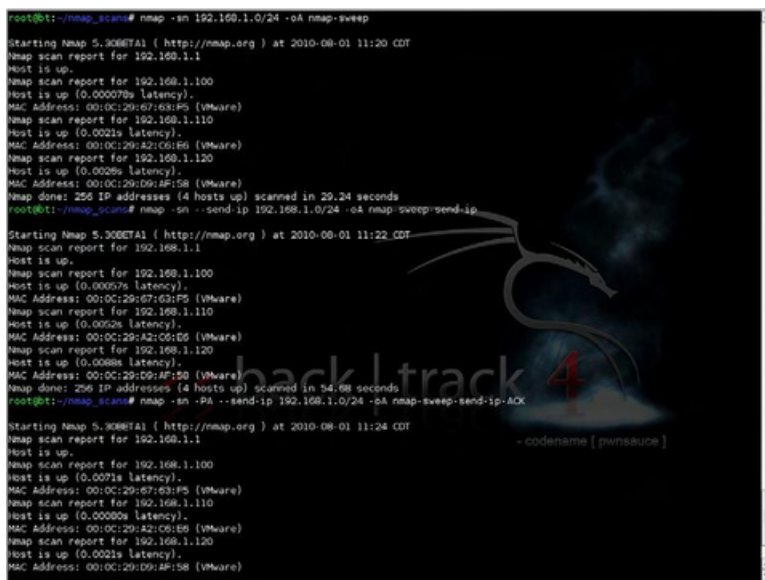
Before scanning active targets, consider using Nmap's ping sweep functionality with the `-sn` option. This option will not port-scan a target, but it will report which targets are up. When invoked as root with `nmap -sn ip_address`, Nmap will send ICMP echo and timestamp packets as well as TCP SYN and ACK packets to determine whether a host is up. If the target addresses are on a local Ethernet network, Nmap will automatically perform an ARP scan versus sending out the packets and waiting for a reply. If the ARP request is successful for a target, it will be displayed. To override this behavior and force Nmap to send IP packets use the `-send-ip` option. If the sweep needs to pass a firewall, it may also be useful to use

a TCP ACK scan in conjunction with the TCP SYN scan. Specifying `-PA` will send a single TCP ACK packet which may pass certain stateful firewall configurations that would block a bare SYN packet to a closed port. In previous Nmap releases, this type of scan was invoked using the `-sP` option.

By understanding which techniques are useful for which environments, you increase the speed of your sweeps. This may not be a big issue when scanning a handful of systems, but when scanning multiple /24 networks, or even a /16, you may need this extra time for other testing. In the example illustrated in Fig. 3.1, the standard ping sweep was the fastest for this particular environment, but that may not always be the case.

3.2.3.1.2 Nmap: ICMP options

If Nmap can't see the target, it won't scan the target unless the `-Pn` (do not ping) option is used. This option was invoked using the `-PO` and `-PN` option in previous Nmap releases. Using the `-Pn` option can create problems because Nmap will try to scan each of the target's ports, even if the target isn't up, which can waste time. To strike a good balance, consider using the `-P` option to select another type of ping behavior. For example, the `-PP` option will use ICMP timestamp requests and the `-PM` option will use ICMP netmask requests. Before you perform a full sweep of a network range, it might be useful to do a few limited tests on known IP addresses, such as Web servers, DNS, and so on, so that you can streamline your ping sweeps and cut down on the number of total packets sent, as well as the time taken for the scans.



```

root@bt:~# nmap -sn 192.168.1.0/24 -oA nmap-sweep
Starting Nmap 5.30BETA1 ( http://nmap.org ) at 2010-08-01 11:20 CDT
Nmap scan report for 192.168.1.1
Host is up.
Nmap scan report for 192.168.1.100
Host is up (0.000078s latency).
MAC Address: 00:0C:29:67:63:F5 (VMware)
Nmap scan report for 192.168.1.110
Host is up (0.0021s latency).
MAC Address: 00:0C:29:A2:C0:E6 (VMware)
Nmap scan report for 192.168.1.120
Host is up (0.0026s latency).
MAC Address: 00:0C:29:09:8F:38 (VMware)
Nmap done: 256 IP addresses (4 hosts up) scanned in 20.24 seconds
root@bt:~# nmap -sn --send-ip 192.168.1.0/24 -oA nmap-sweep-send-ip
Starting Nmap 5.30BETA1 ( http://nmap.org ) at 2010-08-01 11:22 CDT
Nmap scan report for 192.168.1.1
Host is up.
Nmap scan report for 192.168.1.100
Host is up (0.00057s latency).
MAC Address: 00:0C:29:67:63:F5 (VMware)
Nmap scan report for 192.168.1.110
Host is up (0.0026s latency).
MAC Address: 00:0C:29:A2:C0:E6 (VMware)
Nmap scan report for 192.168.1.120
Host is up (0.0008s latency).
MAC Address: 00:0C:29:09:8F:38 (VMware)
Nmap done: 256 IP addresses (4 hosts up) scanned in 34.68 seconds
root@bt:~# nmap -sn -PA --send-ip 192.168.1.0/24 -oA nmap-sweep-send-ip-ACK
Starting Nmap 5.30BETA1 ( http://nmap.org ) at 2010-08-01 11:24 CDT
Nmap scan report for 192.168.1.1
Host is up.
Nmap scan report for 192.168.1.100
Host is up (0.0071s latency).
MAC Address: 00:0C:29:67:63:F5 (VMware)
Nmap scan report for 192.168.1.110
Host is up (0.00009s latency).
MAC Address: 00:0C:29:A2:C0:E6 (VMware)
Nmap scan report for 192.168.1.120
Host is up (0.0021s latency).
MAC Address: 00:0C:29:09:8F:38 (VMware)

```

FIGURE 3.1
Nmap TCP Ping Sweep.

3.2.3.1.3 Nmap: output options

Capturing the results of the scan is extremely important, as you will be referring to this information later in the testing process, and depending on your client's requirements, you may be submitting the results as evidence of vulnerability. The easiest way to capture all the needed information is to use the `-oA` flag, which outputs scan results in three different formats simultaneously: plaintext (`.nmap`), greppable text (`.gnmap`), and XML (`.xml`). The `.gnmap` format is especially important to note, because if you need to stop a scan and resume it at a later date, Nmap will require this file to resume, by using the `-resume` switch. Note the use of the `-oA` flag in Fig. 3.1.

TIP

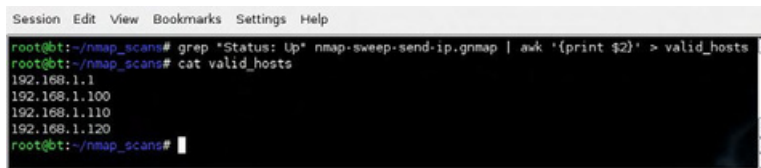
Penetration testing can take some heavy computing resources when you are scanning and querying multiple targets with multiple threads. Running all of your tools from a LiveCD directly may not be the most efficient use of your resources on an extended pen test. Consider performing a hard-drive installation of your toolset so that you can expand and fully utilize the tools. Utilizing a virtual machine is another option to help better utilize machine resources while eliminating the need to install all of your tools individually. Basically, keep your penetration test scope in mind when you are designating your resources so that you do not get caught on the job without enough resources.

3.2.3.1.4 Nmap: basic scripting

When you specify your targets for scanning, Nmap will accept specific IP addresses, address ranges in both CIDR format such as `/8`, `/16`, and `/24`, as well as ranges using `192.168.1.100e200`-style notation. If you have a hosts file, which may have been generated from your ping sweep earlier (hint, hint), you can specify it as well, using the `-iL` flag. There are other, more detailed Nmap parsing programs out there such as the `Nmap::Parser` module for Perl (<http://code.google.com/p/nmap-parser/>), but Fig. 3.2 shows how you can use the `awk` command to create a quick and dirty hosts file from an Nmap ping sweep. Scripting can be a very powerful addition to any tool, but remember to check all the available output options before doing too much work, as some of the heavy lifting may have been done for you.

3.2.3.1.5 Nmap: speed options

Nmap allows the user to specify the “speed” of the scan, or the amount of time from probe sent to reply received, and therefore, how fast packets are sent. On a fast local



```

Session Edit View Bookmarks Settings Help
root@bt:~/nmap_scans# grep "Status: Up" nmap-sweep-send-ip.gnmap | awk '{print $2}' > valid_hosts
root@bt:~/nmap_scans# cat valid_hosts
192.168.1.1
192.168.1.100
192.168.1.110
192.168.1.120
root@bt:~/nmap_scans#

```

FIGURE 3.2

Using `awk` to Parse Nmap Results.

area network (LAN), you can optimize your scanning by setting the -T option to 4, or Aggressive, usually without dropping any packets during the send. If you find that a normal scan is taking a very long time due to ingress filtering, or a firewall device, you may want to enable Aggressive scanning. If you know that an IDS sits between you and the target, and you want to be as stealthy as possible, using -T0 or Paranoid should do what you want; however, it will take a long time to finish a scan, perhaps several hours, depending on your scan parameters. Table 3.2 shows the timing template options for Nmap.

3.2.3.1.6 Nmap: port-scanning options

Besides ping sweeps, Nmap also does port scanning to identify which ports are open on a given target system. As part of our scan, we should find out which ports are open and then later determine which services (and versions) are using those ports as part of the enumeration phase. There are many options for performing this type of scan (as listed in Table 3.1), but we’re going to focus on SYN scanning for this example. By using the -sS option with Nmap, you are able to do a port scan on a target or group of targets using a SYN scan. This is the default scan mechanism used by Nmap and is one of the most commonly performed scans due to its speed, stealth, and compatibility with most target operating systems. With this type of scan, no full TCP connection is made and it is therefore considered a “half-open” scan. Fig. 3.3 shows the results of a SYN scan against some sample hosts.

This produces a listing of the open ports on the target, and possibly open/filtered ports, if the target is behind a firewall. The ports returned as open are listed with what service the ports correspond to, based on port registrations from the Internet

Table 3.2 Nmap Timing Templates		
Template Number	Template Name	Description
0	Paranoid	Used for IDS evasion. One port scanned at a time with five minutes between probes.
1	Sneaky	Used for IDS evasion. One port scanned at a time with 15 s between probes.
2	Polite	Uses less bandwidth and machine resources than normal. One port scanned at a time with 0.4 s between probes.
3	Normal	A standard scan (default if no options specified) using parallel processing. Works both locally and over the Internet.
4	Aggressive	A fast scan used with fast, stable connections. Has a 10 ms delay between probes and uses parallel processing.
5	Insane	A very fast scan used typically for very fast networks or if you’re willing to sacrifice accuracy for speed. Reduces delay between probes to 5 ms and uses parallel processing.



```

root@bt:~/nmap_scans# nmap -sS 192.168.1.110
Starting Nmap 5.30BETA1 ( http://nmap.org ) at 2010-08-01 13:32 CDT
Nmap scan report for 192.168.1.110
Host is up (0.0018s latency).
Not shown: 996 closed ports
PORT      STATE SERVICE
21/tcp    open  ftp
22/tcp    open  ssh
80/tcp    open  http
631/tcp   open  ipp
MAC Address: 00:0C:29:A2:C6:E6 (VMware)

Nmap done: 1 IP address (1 host up) scanned in 15.74 seconds
root@bt:~/nmap_scans# nmap -sS 192.168.1.120
Starting Nmap 5.30BETA1 ( http://nmap.org ) at 2010-08-01 13:33 CDT
Nmap scan report for 192.168.1.120
Host is up (0.0011s latency).
Not shown: 993 closed ports
PORT      STATE SERVICE
21/tcp    open  ftp
80/tcp    open  http
135/tcp   open  msrpc
139/tcp   open  netbios-ssn
443/tcp   open  https
445/tcp   open  microsoft-ds
3306/tcp  open  mysql
MAC Address: 00:0C:29:D9:AF:58 (VMware)

Nmap done: 1 IP address (1 host up) scanned in 15.46 seconds

```

FIGURE 3.3
Nmap TCP SYN Scan.

Assigned Numbers Authority (IANA), as well as any commonly used ports, such as 31337 for Back Orifice.

By default, Nmap 5.30 scans over 1000 ports for common services. This will catch most open TCP ports that are out there. However, sneaky system administrators may run services on uncommon ports, practicing security through obscurity. Without scanning those uncommon ports, you may be missing these services. If you have time, or you suspect that a system may be running other services, run Nmap with the `-p0-65535` parameter, which will scan all 65,536 TCP ports. Note that this may take a long time, even on a LAN with responsive systems and no firewalls, possibly up to a few hours. Performing a test such as this over the Internet may take even longer, which will also allow more time for the system owners, or watchers, to note the excessive traffic and shut you down.

3.2.3.1.7 Nmap: stealth scanning

For any scanning that you perform, it is not a good idea to use a connect scan (`-sT`), which fully establishes a connection to a port. Excessive port connections can create a DoS condition with older machines, and will definitely raise alarms on any IDS. For that reason, you should usually use a stealthy port-testing method with Nmap, such as a SYN scan. Even if you are not trying to be particularly stealthy, this is much easier on both the testing system and the target.

In addition to lowering your profile with half-open scans, you may also consider the ftp or “bounce” scan and idle scan options which can mask your IP from the

target. The ftp scan takes advantage of a feature of some FTP servers, which allow anonymous users to proxy connections to other systems. If you find during your enumeration that an anonymous FTP server exists, or one to which you have login credentials, try using the `-b` option with `user:pass@server:ftpport`. If the server does not require authentication, you can skip the `user:pass`, and unless FTP is running on a nonstandard port, you can leave out the `ftpport` option as well. This type of scan works only on FTP servers, allowing you to “proxy” an FTP connection, and many servers today disable this option by default.

The idle scan, using `-sI zombiehost:port`, has a similar result but a different method of scanning. This is detailed further at Fyodor’s web page, <http://nmap.org/book/idlescan.html>, but the short version is that if you can identify a intermediate target (zombie) with low traffic and predictable fragment identification (IP ID) values, you can send spoofed packets to your real target, with the source set to the zombie or idle target. The result is that an IDS sees the idle scan target as the system performing the scanning, keeping your system hidden. If the idle target is a trusted IP address and can bypass host-based access control lists, even better! Do not expect to be able to use a bounce or idle scan on every penetration test engagement, but keep looking around for potential targets. Older systems, which do not offer useful services, may be the best targets for some of these scan options.

NOTE

So far, we have focused on TCP-based services because most interactive services that may be vulnerable run over TCP. This is not to say that UDP-based services, such as `rpcbind`, `tftp`, `snmp`, `nfs`, and so on, are not vulnerable to attack. UDP scanning is another activity which could take a very long time, on both LANs and wide area networks (WANs). Depending on the length of time and the types of targets you are attacking, you may not need to perform a UDP scan. However, if you are attacking targets that may use UDP services, such as infrastructure devices and SunOS/Solaris machines, taking the time for a UDP scan may be worth the effort. Nmap uses the flag `-sU` to specify a UDP scan.

3.2.3.2 Netenum: ping sweep

If you need a very simple ICMP ping sweep program that you can use for scriptable applications, `netenum` might be useful. It performs a basic ICMP ping and then replies with only the reachable targets. One quirk about `netenum` is that it requires a timeout to be specified for the test. If no timeout is specified, it outputs a CR-delimited dump of the input addresses. If you have tools that will not accept a CIDR-formatted range of addresses, you might use `netenum` to simply expand that into a listing of individual IP addresses. Fig. 3.4 shows the basic usage of `netenum` in ping sweep mode with a timeout value of 5, as well as network address expansion mode showing the valid addresses for a CIDR of `192.168.1.0/24`, including the network and broadcast addresses.

Netenum USAGE

How to use:

```
netenum destination [Timeout] [Verbosity]
```

Input fields:

Destination is the target specification which can be a single host or a full network/subnet.

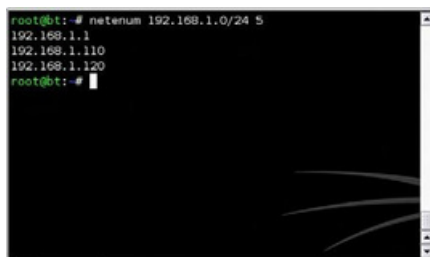
[Timeout] is a value to use for the scan. Any value greater than 0 will use pings to scan.

[Verbosity] is a value 0–3 that determines how verbose the output is.

Output:

Displays active hosts to the screen. Can be redirected to a file or to another command for scripted scans.

Typical output:



```
root@bt:~# netenum 192.168.1.0/24 5
192.168.1.1
192.168.1.110
192.168.1.120
root@bt:~#
```

FIGURE 3.4
Netenum Output.

3.2.3.3 Unicornscan: port scan and fuzzing

Unicornscan is different from a standard port-scanning program; it also allows you to specify more information, such as source port, packets per second sent, and randomization of source IP information, if needed. For this reason, it may not be the best choice for initial port scans; rather, it is more suited for later “fuzzing” or experimental packet generation and detection. However, just as Nmap has capabilities which far exceed that of a ping sweep, Unicornscan can be used for basic port scans in addition to its more complex features.

Unicornscan USAGE

How to use:

```
unicornscan [Options] Target(s):Port(s)
```

Input fields:

[Options] are very wide ranging and control the type of scan performed as well as very granular control over the packets sent. A list of all options can be seen by using the -h option.

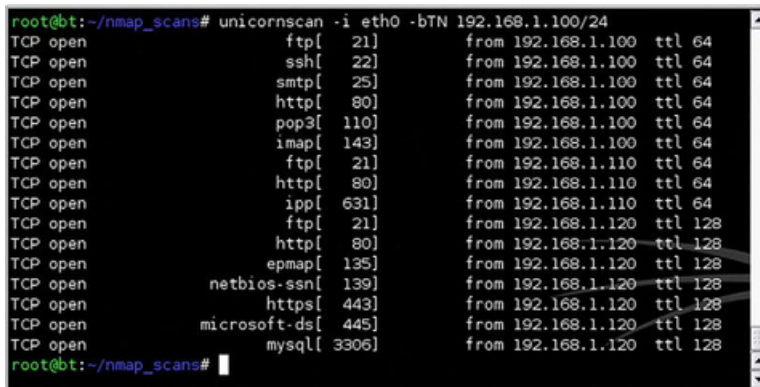
Target(s) is the target specification which can be a single host or a range using a CIDR mask.

Port(s) are the ports to scan.

Output:

Displays identified ports and their status to the screen.

Typical output:



```

root@bt:~/nmap_scans# unicornscan -i eth0 -bTN 192.168.1.100/24
TCP open      ftp[ 21]      from 192.168.1.100  ttl 64
TCP open      ssh[ 22]      from 192.168.1.100  ttl 64
TCP open      smtp[ 25]     from 192.168.1.100  ttl 64
TCP open      http[ 80]     from 192.168.1.100  ttl 64
TCP open      pop3[ 110]    from 192.168.1.100  ttl 64
TCP open      imap[ 143]    from 192.168.1.100  ttl 64
TCP open      ftp[ 21]      from 192.168.1.110  ttl 64
TCP open      http[ 80]     from 192.168.1.110  ttl 64
TCP open      ipp[ 631]     from 192.168.1.110  ttl 64
TCP open      ftp[ 21]      from 192.168.1.120  ttl 128
TCP open      http[ 80]     from 192.168.1.120  ttl 128
TCP open      epmap[ 135]   from 192.168.1.120  ttl 128
TCP open      netbios-ssn[ 139] from 192.168.1.120  ttl 128
TCP open      https[ 443]   from 192.168.1.120  ttl 128
TCP open      microsoft-ds[ 445] from 192.168.1.120  ttl 128
TCP open      mysql[ 3306]  from 192.168.1.120  ttl 128
root@bt:~/nmap_scans#

```

FIGURE 3.5

Unicornscan Port-scan Output.

Figure 3.5 shows Unicornscan in action, performing a basic SYN port scan with broken CRC values for the sent packets. This type of port scan can provide data on open ports and shows which IPs have those ports open. Due to its rich feature set, Unicornscan might be better suited for scanning during an IDS test, where the packet-forging capabilities could be put to more use.

WARNING

Tools are also available which do scanning/enumeration/vulnerability scans at the same time such as OpenVAS (www.openvas.org). Why don't we use those for the scanning phase of our penetration tests? Sure, it would be a lot easier if instead of running these granular tools, we could just fire up the big bad vulnerability scanner and have it do all the work for us. In some situations, this is perfectly acceptable; however, it always pays to know what's going on behind the scenes on those scanners. Because much of their operation is abstracted from the user (you), sometimes it can be hard to tell what is actually tested when the scanning and enumeration portion is performed. In some cases, those vulnerability scanners simply wrap a user interface around the same tool you would normally use for scanning and enumeration directly.

When you run the specific and targeted tools yourself to build up a list of valid hosts and services, you know exactly what is open at the time of scanning and what is not. If there was a bug or misconfiguration in the specification of your target addresses, you would know pretty quickly, and sometimes that is not the case with the integrated vulnerability scanners.

Vulnerability scanners serve a very important purpose in penetration testing, risk management, and functional security overall. However, during initial information gathering, as we are describing in this chapter, it is usually better to take a bit more time and run the basic tools yourself so that you have a firm understanding of what is really out there.

3.3 ENUMERATION

So, what is enumeration? Enumeration involves listing and identifying the specific services and resources that a target offers. You perform enumeration by starting with a set of parameters, such as an IP address range, or a specific domain name system (DNS) entry, and the open ports on the system. Your goal for enumeration is a list of services which are known and reachable from your source. From those services, you move further into the scanning process, including security scanning and testing, the core of penetration testing. Terms such as banner grabbing and fingerprinting fall under the category of enumeration.

3.3.1 Approach

With that goal in mind, let's talk about our approach to enumeration. An example of successful enumeration is to start with a host such as 192.168.1.100 which has Transmission Control Protocol (TCP) port 22 open. After performing enumeration on the target, you should be able to state with a reasonable level of confidence that OpenSSH v4.3 is running with protocol version 1. Moving into operating system fingerprinting, an ideal result would be determining that the host is running Linux kernel 2.6.x. Granted, sometimes your enumeration will not get to this level of detail, but you should still set that for your goal. The more information you have, the better. Remember that all the information gathered in this phase is used to deepen the penetration in later phases.

As we've already discovered, keeping good notes is very important during a penetration test, and it is especially important during enumeration. Sometimes your client may want to know the exact flags or switches you used when you ran a tool, or what the verbose output was. If you cannot provide this information upon request, at best you may lose respect in the eyes of your client. Some clients and contracts require full keylogging and output logging, so again make sure you understand the requirements upon you as the tester for all responsibilities, including documentation. This should be spelled out very clearly in your Rules of Engagement document.

TIP

If the tool you are using cannot output a log file, make sure you use tools such as `tee`, which will allow you to direct the output of a command not only to your terminal, but also to a log file.

One quick note about the `tee` command: If you need to keep detailed records about the tools and testing, you can use `date` to make a timestamp for any output files you create. In [Fig. 3.6](#), the `date` command is used to stamp with day-month-year and then hour:minute. You can use lots of other options with `date`, so if you need that level of detail, try `date -help` to get a full list of parameters.

So our approach based on this example is to take the information that we have already gathered such as the IP address (from reconnaissance) and the open ports (from scanning) and gather as much extended data about the target and the services



```

root@bt:~/nmap_scans# netenum 192.168.1.0/24 5 | tee netenum_output-$(date +%m-%d-%y\_%H:%M).log
log
192.168.1.1
192.168.1.109
192.168.1.110
192.168.1.120
root@bt:~/nmap_scans# ls netenum*
netenum_output-08-01-10_15:30.log
root@bt:~/nmap_scans# cat netenum_output-08-01-10_15:30.log
192.168.1.1
192.168.1.109
192.168.1.110
192.168.1.120
root@bt:~/nmap_scans#

```

FIGURE 3.6
Using Date with the tee Command.

running on it as possible using a variety of techniques and tools. To do this, we will be using some basic core technologies similar to but more extensive than those used in the scanning phase.

3.3.2 Core technology

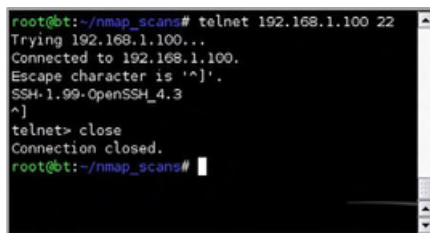
Enumeration is based on the ability to gather information from an open port. This is performed by either straightforward banner grabbing when connecting to an open port, or by inference from the construction of a returned packet. There is not much true magic here, as services are supposed to respond in a predictable manner; otherwise, they would not have much use as a service!

3.3.2.1 Active versus passive

You can perform enumeration using either active or passive methods. Proxy methods may also be considered passive, as the information you gather will be from a third source, rather than intercepted from the target itself. However, a truly passive scan should not involve any data being sent from the host system. Passive data is data that is returned from the target, without any data being sent from the testing system. A good example of a truly passive enumeration tool is p0f, which is detailed later in the chapter. Active methods are the more familiar ones, in which you send certain types of packets and then receive packets in return. Most scanning and enumeration tools are active.

3.3.2.2 Service identification

Now that the open ports are captured through your scanning efforts, you need to be able to verify what is running on them. You would normally think that the Simple Mail Transport Protocol (SMTP) is running on TCP 25, but what if the system administrator is trying to obfuscate the service and it is running Telnet instead? The easiest way to check the status of a port is a banner grab, which involves capturing the target's response after connecting to a service, and then comparing it to a list of known services, such as the response when connecting to an OpenSSH server as shown in [Fig. 3.7](#). The banner in this case is pretty evident, as is the version of the service, OpenSSH version 4.3 listening for SSH version 1.99 connections. Please



```
root@bt:~/nmap_scans# telnet 192.168.1.100 22
Trying 192.168.1.100...
Connected to 192.168.1.100.
Escape character is '^]'.
SSH-1.99-OpenSSH_4.3
^]
telnet> close
Connection closed.
root@bt:~/nmap_scans#
```

FIGURE 3.7
Basic Telnet Banner Grab.

note that just because the banner says it is one thing does not necessarily mean that it is true. System administrators and security people have been changing banners and other response data for a long time in order to fool attackers.

3.3.2.2.1 RPC enumeration

Some services are wrapped in other frameworks, such as Remote Procedure Call (RPC). On UNIX-like systems, an open TCP port 111 indicates this. UNIX-style RPC (used extensively by systems such as Solaris) can be queried with the `rpcinfo` command, or a scanner can send NULL commands on the various RPC-bound ports to enumerate what function that particular RPC service performs. Fig. 3.8 shows the output of the `rpcinfo` command used to query the portmapper on the Solaris system and return a list of RPC services available.

3.3.2.3 Fingerprinting

The goal of system fingerprinting is to determine the operating system version and type. There are two common methods of performing system fingerprinting: active and passive scanning. The more common active methods use responses sent to TCP or ICMP packets. The TCP fingerprinting process involves setting flags in the header that different operating systems and versions respond to differently. Usually several different TCP packets are sent and the responses are compared to known baselines (or fingerprints) to determine the remote OS. Typically, ICMP-based methods use fewer packets than TCP-based methods, so in an environment where you need to be stealthier and can afford a less specific fingerprint, ICMP may be the way to go. You can achieve higher degrees of accuracy by combining TCP/UDP and ICMP methods, assuming that no device in between you and the target is reshaping packets and mismatching the signatures.

For the ultimate in stealthy detection, you can use passive fingerprinting. Unlike the active method, this style of fingerprinting does not send any packets, but relies on sniffing techniques to analyze the information sent in normal network traffic. If your target is running publicly available services, passive fingerprinting may be a good way to start off your fingerprinting. Drawbacks of passive fingerprinting are that it is usually less accurate than a targeted active fingerprinting session and it relies on an

```

root@bt:~ - Shell - Konsole
root@bt:~# rpcinfo -p 144.15.17.106
program vers proto port
100000 4 tcp 111 portmapper
100000 3 tcp 111 portmapper
100000 2 tcp 111 portmapper
100000 4 udp 111 portmapper
100000 3 udp 111 portmapper
100000 2 udp 111 portmapper
100007 3 udp 32773 ypbinding
100007 2 udp 32773 ypbinding
100007 1 udp 32773 ypbinding
100007 3 tcp 32771 ypbinding
100007 2 tcp 32771 ypbinding
100007 1 tcp 32771 ypbinding
100068 2 udp 32786
100068 3 udp 32786
100068 4 udp 32786
100068 5 udp 32786
100229 1 tcp 32778
100229 2 tcp 32778
100001 2 udp 32789 rstatd
100001 3 udp 32789 rstatd
100001 4 udp 32789 rstatd
100002 2 udp 32792 rusersd
100002 3 udp 32792 rusersd
100002 2 tcp 32784 rusersd
100002 3 tcp 32784 rusersd
100153 1 udp 32798
100024 1 udp 32811 status
100024 1 tcp 32788 status
100133 1 udp 32811
100133 1 tcp 32788
100021 1 udp 4045 nlockmgr
100021 2 udp 4045 nlockmgr
100021 3 udp 4045 nlockmgr
100021 4 udp 4045 nlockmgr
100021 1 tcp 4045 nlockmgr
100021 2 tcp 4045 nlockmgr
100021 3 tcp 4045 nlockmgr
100021 4 tcp 4045 nlockmgr
100005 1 udp 32878 mountd
100005 2 udp 32878 mountd

```

FIGURE 3.8
Rpcinfo Output.

existing traffic stream to which you have access. It can also take much longer depending on how high the activity level of the target system is.

3.3.2.4 Being loud, quiet, and all that lies between

There are always considerations to make when you are choosing what types of enumerations and scans to perform. When performing an engagement in which your client's administrators do not know that you are testing, your element of stealth is crucial. Once you begin passing too much traffic that goes outside their baseline, you may find yourself shut down at their perimeter and your testing cannot continue. Conversely, your penetration test may also serve to test the administrator's response, or the performance of an intrusion detection system (IDS) or intrusion prevention system (IPS). When that is your goal, being noisy—that is, not trying to hide your

scans and attacks may be just what you need to do. Here are some things to keep in mind when opting to use stealth.

3.3.2.4.1 Timing

Correlation is a key point when you are using any type of IDS. An IDS relies on timing when correlating candidate events. Running a port scan of 1500 ports in 30 seconds will definitely be more suspicious than one in which you take six hours to scan those same 1500 ports. Sure, the IDS might detect your slower scan by other means, but if you are trying to raise as little attention as possible, throttle your connection timing back. Also, remember that most ports lie in the “undefined” category. You can also reduce the number of ports you decide to scan if you’re interested in stealth.

Use data collected from the reconnaissance phase to supplement the scanning phase. If you found a host through a search engine such as Google, you already know that port 80 (or 443) is open. There’s no need to include that port in a scan if you’re trying to be stealthy. We discussed using Google for reconnaissance activities in Chapter 2. If you do need to create connections at a high rate, take some of the reconnaissance data and figure out when the target passes the most traffic. For example, on paydays or on the first of the month a bank should have higher traffic than on other days in the month due to the higher number of visitors performing transactions. You may even be able to find pages on the bank’s site that show trends regarding traffic. Perform your scans during those peak times and you are less likely to stand out against that background noise.

3.3.2.4.2 Bandwidth issues

When you are scanning a single target over a business broadband connection, you likely will not be affecting the destination network even if you thread up a few scans b targets, the network may start to slow down. Unless you are performing a DoS test, this is a bad idea because you may be causing negative conditions for your target and excessive bandwidth usage is one of the first things a competent system administrator will notice. Even a nonsecurity-conscious system administrator will notice when the helpdesk phone board is lit up with “I can’t reach my email!” messages. Also, sometimes you will need to scan targets that are located over connections such as satellite or microwave. In those situations, you definitely need to be aware of bandwidth issues with every action you take. Nothing is worse than shutting down the sole communications link for a remote facility due to a missed flag or option.

3.3.2.4.3 Unusual packet formation

A common source for unusual packets is active system fingerprinting programs. When the program sets uncommon flags and sends them along to a target system, although the response serves a purpose for determining the operating system, the flags may also be picked up by an IDS and firewall logs as rejections. Packets such as ICMP Source Quench coming from sources that are not in the internal network of target, especially when no communication with those sources has been

your

established, are also a warning flag. Keep in mind that whatever you send to your target can give away your intent and maybe even your testing plan.

3.3.2.5 SNMP enumeration

One of the less talked about technologies which can be used for enumeration is the Simple Network Management Protocol (SNMP). SNMP is used for monitoring and managing many systems which could exist on a network including network devices and servers. It is based on UDP and is therefore a stateless protocol.

SNMP should be included in any discussion about enumeration for three reasons. First, it is widely deployed, but often forgotten, leading to a lack of security around the community strings used for SNMP authentication. Secondly, it is typically used to monitor or control some of the most important devices or systems on any given network. Lastly, a vast amount of information about a device or system can be very rapidly gathered using some very simple SNMP queries making it a very rapid method of enumerating a host and its services.

3.3.3 Open source tools

Now, let's talk about tools that aid in the enumeration phase of an assessment. Based on the data that we gathered during our scanning, we now take our penetration testing to the next level and start gathering some in-depth information about our targets. The information we gather in this phase should include:

- Operating system
- Operating system version
- Services (ftp, http, pop3, etc.)
- Software providing those services
- Software versions

3.3.3.1 Nmap: OS fingerprinting

Let's go back to our old friend Nmap. You should be able to create a general idea of the remote target's operating system from the services running and the ports open. For example, port 135, 137, 139, or 445 often indicates a Windows-based target. However, if you want to get more specific, you can use Nmap's -O flag, which invokes Nmap's fingerprinting mode. You need to be careful here as well, as some older operating systems, such as AIX prior to 4.1, and older SunOS versions, have been known to die when presented with a malformed packet. Keep this in mind before blindly using -O across a full subnet. In [Figs 3.9 and 3.10](#), you can see the output from two fingerprint scans using nmap -O. Note that the fingerprint option without any scan types will invoke a SYN scan, the equivalent of -sS, so that ports can be found for the fingerprinting process to occur.

3.3.3.2 Nmap: banner grabbing

You invoke Nmap's version scanning feature with the -sV flag. Based on a returned banner, or on a specific response to an Nmap-provided probe, a match is made between the service response and the Nmap service fingerprints. This type of enumeration can be



```

root@bt:~/nmap_scans# nmap -O 192.168.1.120

Starting Nmap 5.30BETA1 ( http://nmap.org ) at 2010-08-01 16:05 CDT
Nmap scan report for 192.168.1.120
Host is up (0.0017s latency).
Not shown: 993 closed ports
PORT      STATE SERVICE
21/tcp    open  ftp
80/tcp    open  http
135/tcp   open  msrpc
139/tcp   open  netbios-ssn
443/tcp   open  https
445/tcp   open  microsoft-ds
3306/tcp  open  mysql
MAC Address: 00:0C:29:D9:AF:58 (VMware)
Device type: general purpose
Running: Microsoft Windows XP|2003
OS details: Microsoft Windows XP Professional SP2 or Windows Server 2003
Network Distance: 1 hop

OS detection performed. Please report any incorrect results at http://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 17.25 seconds
root@bt:~/nmap_scans#

```

FIGURE 3.9
Nmap OS Fingerprint of Windows XP System.



```

root@bt:~/nmap_scans# nmap -O 192.168.1.110

Starting Nmap 5.30BETA1 ( http://nmap.org ) at 2010-08-01 16:07 CDT
Nmap scan report for 192.168.1.110
Host is up (0.0012s latency).
Not shown: 996 closed ports
PORT      STATE SERVICE
21/tcp    open  ftp
22/tcp    open  ssh
80/tcp    open  http
631/tcp   open  ipp
MAC Address: 00:0C:29:A2:C6:E6 (VMware)
Device type: general purpose
Running: Linux 2.6.X
OS details: Linux 2.6.13 - 2.6.28
Network Distance: 1 hop

OS detection performed. Please report any incorrect results at http://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 18.26 seconds
root@bt:~/nmap_scans#

```

FIGURE 3.10
Nmap OS Fingerprint of Linux System.

very noisy as unusual packets are sent to guess the service version. As such, IDS alerts will likely be generated unless some other type of mechanism can be used to mask it.

Figure 3.11 shows a successful scan using `nmap -sS -sV -O` against a Linux server. This performs a SYN-based port scan with a version scan and uses the OS fingerprinting function. The version scanner picked up the version (4.3) and protocol (1.99) of OpenSSH in use, along with the Linux kernel level range (2.6.x), the web server type and version (Apache 2.0.55) and a mod (PHP 5.1.2), the pop3 server (Openwall), and a variety of other service and version information. Overall, we



```

Starting Nmap 5.30BETA1 ( http://nmap.org ) at 2010-08-01 16:12 CDT
Nmap scan report for 192.168.1.100
Host is up (0.0011s latency).
Not shown: 992 filtered ports
PORT      STATE SERVICE VERSION
20/tcp    closed ftp-data
21/tcp    open  ftp      vsftpd (broken: could not bind listening IPv4 socket)
22/tcp    open  ssh      OpenSSH 4.3 (protocol 1.99)
25/tcp    open  smtp      Sendmail 8.13.7/8.13.7
80/tcp    open  http      Apache httpd 2.0.55 ((Unix) PHP/5.1.2)
110/tcp   open  pop3      Openwall popa3d
143/tcp   open  imap      Uw imapd 2004.357
443/tcp   closed https
MAC Address: 00:0C:29:67:63:F5 (VMware)
Device type: general purpose
Running: Linux 2.6.X
OS details: Linux 2.6.13 - 2.6.28
Network Distance: 1 hop
Service Info: Host: slax.example.net; OS: Unix

OS and Service detection performed. Please report any incorrect results at http://nmap.org/submit/.
Nmap done: 1 IP address (1 host up) scanned in 34.92 seconds
root@bt:~/nmap_scripts#

```

FIGURE 3.11

Nmap Banner Grab.

ended up with a great deal of information about this target! Information such as this would help you to classify the system as a general infrastructure server with lots of possible targets and entry points.

With Nmap, you can still gather a little more information about your target by using the `-A` option. This option enables OS and version detection, script scanning, and a traceroute thus supplying you with extended enumeration on the target. You can see an example of the results gathered from the same target using this option in [Fig. 3.12](#).

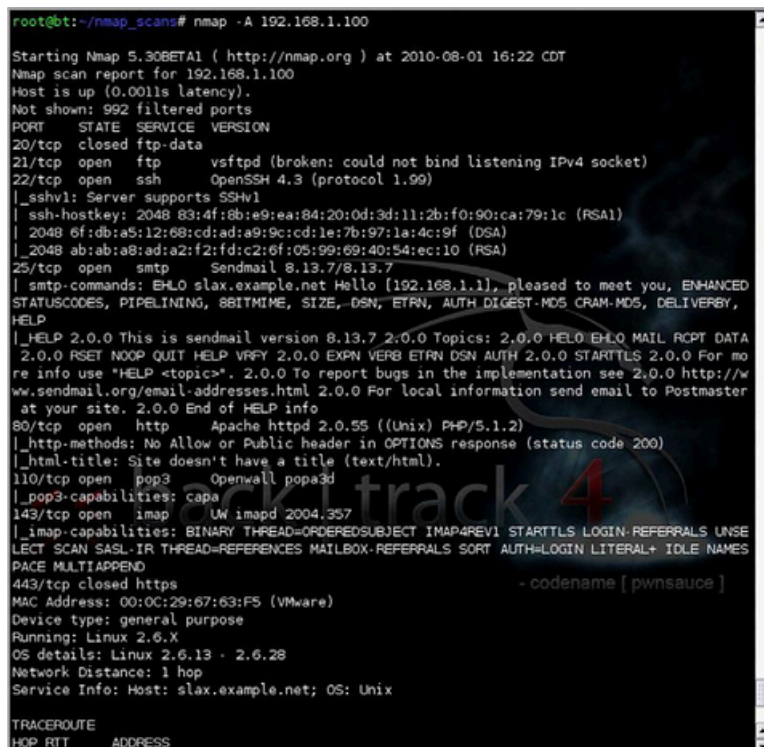
As you can see from the results, we now have information on which SMTP commands the target accepts as well as SSH host keys, POP3 and IMAP capabilities, and traceroute information. This additional level of detail can save some time later by helping us quickly identify whether a service is vulnerable to a specific attack which requires certain commands to be available.

3.3.3.3 Netcat

We used telnet for an initial example of doing a banner grab, but a more versatile tool is, quite simply, `nc`. `nc` is available for read and write on TCP and UDP ports. This Netcat seems rather vague, but that ambiguity is its greatest feature, giving it a range of flexibility beyond that which most tools offer. Netcat can run as either a client or a server using either TCP or UDP for its data transfer and allows you to perform some pretty cool tricks.

We'll examine some of Netcat's more advanced features as we dig deeper into penetration testing, but for now, we'll use its ability to connect to a TCP port and allow us to grab the banner. For this example, we'll use Netcat to connect to port 21 on our target. We received this message using Nmap:

```
21/tcp open ftp vsftpd (broken: could not bind listening IPv4 socket)
```



```

root@bt:~/nmap_scans# nmap -A 192.168.1.100

Starting Nmap 5.30BETA1 ( http://nmap.org ) at 2010-08-01 16:22 CDT
Nmap scan report for 192.168.1.100
Host is up (0.0011s latency).
Not shown: 992 filtered ports
PORT      STATE SERVICE VERSION
20/tcp    closed ftp-data
21/tcp    open  ftp      vsftpd (broken: could not bind listening IPv4 socket)
22/tcp    open  ssh      OpenSSH 4.3 (protocol 1.99)
|_sshv1: Server supports SSHv1
|_ssh-hostkey: 2048 83:4f:8b:e9:ea:84:20:0d:3d:11:2b:f0:90:ca:79:1c (RSA1)
|_ 2048 6f:db:a5:12:68:cd:ad:a9:9c:cd:1e:7b:97:1a:4c:9f (DSA)
|_ 2048 ab:ab:a8:ad:a2:f2:fd:c2:6f:05:99:69:40:54:ec:10 (RSA)
25/tcp    open  smtp      Sendmail 8.13.7/8.13.7
|_smtp-commands: EHLO slax.example.net Hello [192.168.1.1], pleased to meet you, ENHANCED
STATUSCODES, PIPELINING, 8BITMIME, SIZE, DSN, ETRN, AUTH DIGEST-MD5 CRAM-MD5, DELIVERBY,
HELP
|_HELP 2.0.0 This is sendmail version 8.13.7 2.0.0 Topics: 2.0.0 HELO EHLO MAIL RCPT DATA
2.0.0 RSET NOOP QUIT HELP VRFY 2.0.0 EXPN VERB ETRN DSN AUTH 2.0.0 STARTTLS 2.0.0 For mo
re info use "HELP <topic>". 2.0.0 To report bugs in the implementation see 2.0.0 http://w
ww.sendmail.org/email-addresses.html 2.0.0 For local information send email to Postmaster
at your site. 2.0.0 End of HELP info
80/tcp    open  http      Apache httpd 2.0.55 ((Unix) PHP/5.1.2)
|_http-methods: No Allow or Public header in OPTIONS response (status code 200)
|_html-title: Site doesn't have a title (text/html).
110/tcp   open  pop3      Openwall popa3d
|_pop3-capabilities: capa
143/tcp   open  imap      UW imapd 2004.357
|_imap-capabilities: BINARY THREAD=ORDEREDSUBJECT IMAP4REV1 STARTTLS LOGIN-REFERRALS UNSE
LECT SCAN SASL-IR THREAD=REFERENCES MAILBOX-REFERRALS SORT AUTH=LOGIN LITERAL+ IDLE NAMES
PACE MULTIAPPEND
443/tcp   closed https
MAC Address: 00:0C:29:67:63:F5 (VMware)
Device type: general purpose
Running: Linux 2.6.X
OS details: Linux 2.6.13 - 2.6.28
Network Distance: 1 hop
Service Info: Host: slax.example.net; OS: Unix

TRACEROUTE
HOP RTT ADDRESS

```

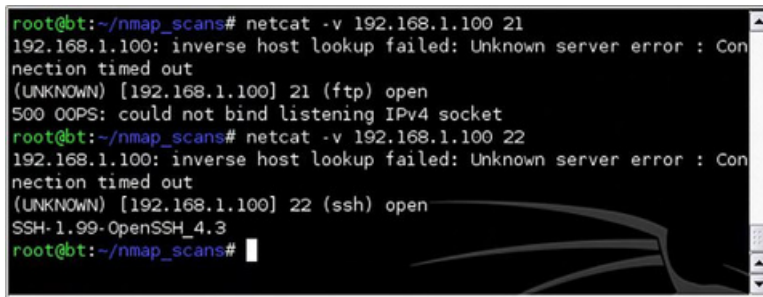
FIGURE 3.12
Nmap -A Output.

Let's see what response we get with Netcat. You can see these results in Fig. 3.13. It looks like we ended up with an identical result which validates our Nmap scan results and indicates that there is an issue with connecting to the FTP server on that host. However, the additional results shown in Fig. 3.13 for a connection to port 22 give us the banner for SSH on the host. This also matches the Nmap results but shows another way to gather that type of data.

3.3.3.4 POf: passive OS fingerprinting

POf is one of the few open source passive fingerprinting tools. If you want to be extremely stealthy in your initial scan and enumeration processes, and you don't mind getting high-level results for OS fingerprinting, pOf is the tool for you. It works by analyzing the responses from your target on innocuous queries, such as web traffic, ping replies, or normal operations. POf gives the best estimation on operating system based on those replies, so it may not be as precise as other active tools, but it can still give a good starting point.

While the accuracy may not be as high as with an active tool, the benefit of using pOf is in its stealth and its ability to fingerprint systems based on packet captures. If you happen to have a sniffer capture of a target environment, pOf can analyze that data and attempt to fingerprint the hosts.



```

root@bt:~/nmap_scans# netcat -v 192.168.1.100 21
192.168.1.100: inverse host lookup failed: Unknown server error : Connection timed out
(UNKNOWN) [192.168.1.100] 21 (ftp) open
500 OOPS: could not bind listening IPv4 socket
root@bt:~/nmap_scans# netcat -v 192.168.1.100 22
192.168.1.100: inverse host lookup failed: Unknown server error : Connection timed out
(UNKNOWN) [192.168.1.100] 22 (ssh) open
SSH-1.99-OpenSSH_4.3
root@bt:~/nmap_scans#

```

FIGURE 3.13

Netcat Connection Results.

Figure 3.14 shows the results of using p0f to monitor network traffic on eth0 and attempt to fingerprint hosts based on the traffic that it sees. Fig. 3.15 shows the traffic that p0f was monitoring at the time it fingerprinted the host. As you can see, if you were monitoring a live network the chances that this type of connection would be made at some point is very high and thus you'd have fingerprint data on your target in short order.

p0f USAGE

How to use:

p0f [Options]

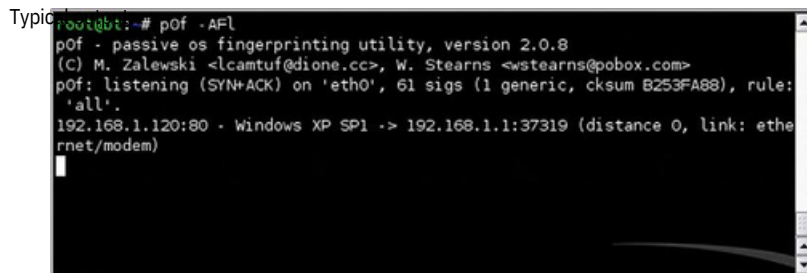
Input fields:

[Options] are very wide ranging and include the following:

A list of all options can be seen by using the -h option.

Output:

Displays packets matching the scan criteria and any identified OS versions.



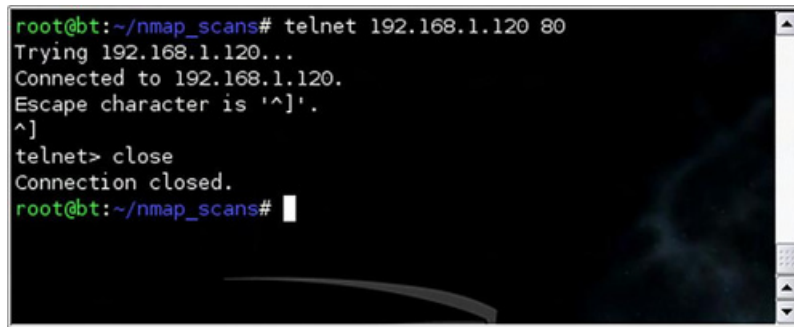
```

root@bt:~# p0f -AF
p0f - passive os fingerprinting utility, version 2.0.8
(C) M. Zalewski <lcamtuf@diene.cc>, W. Stearns <wstearns@pobox.com>
p0f: listening (SYN+ACK) on 'eth0', 61 sigs (1 generic, cksum B253FA88), rule:
'all'.
192.168.1.120:80 - Windows XP SP1 -> 192.168.1.1:37319 (distance 0, link: ethernet/modem)

```

FIGURE 3.14

p0f Fingerprinting Results.



```

root@bt:~/nmap_scans# telnet 192.168.1.120 80
Trying 192.168.1.120...
Connected to 192.168.1.120.
Escape character is '^]'.
^]
telnet> close
Connection closed.
root@bt:~/nmap_scans#

```

FIGURE 3.15

Sample Data for pOf Fingerprinting.

It should be noted, however, that while this tool is very useful, it has been a long time (2006) since an update has been published and signature files are becoming more and more out of date. Fortunately, you can add signatures to a custom file and have pOf read from that file to update its fingerprinting capabilities.

3.3.3.5 Xprobe2: OS fingerprinting

Xprobe2 is primarily an OS fingerprinter, but it also has some basic port-scanning functionality built in to identify open or closed ports. You can also specify known open or closed ports, to which Xprobe2 performs several different TCP, UDP, and ICMP-based tests to determine the remote OS. Although you can provide Xprobe2 with a known open or closed port for it to determine the remote OS, you can also tell it to “blindly” find an open port for fingerprinting using the -B option, as shown in [Fig. 3.16](#).

Xprobe2 USAGE

How to use:

xprobe2 [Options] target

Input fields:

[Options] are very wide ranging and include the following:

A list of all options can be seen by using the -h option.

Output:

Displays packets matching the scan criteria and any identified OS versions.

Typical output:

```

root@bt:/pentest# xprobe2 -B 192.168.1.120

Xprobe2 v.0.3 Copyright (c) 2002-2005 fyodor@o0o.nu, ofir@sys-security.com, meder@o0o.nu

[+] Target is 192.168.1.120
[+] 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:tll_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 module
[x] [8] fingerprint:icmp_amask - ICMP Address mask request fingerprinting module
[x] [9] fingerprint:icmp_port_unreach - ICMP port unreachable fingerprinting module
[x] [10] fingerprint:tcp_hshake - TCP Handshake fingerprinting module
[x] [11] fingerprint:tcp_rst - TCP RST fingerprinting module
[x] [12] fingerprint:smb - SMB fingerprinting module
[x] [13] fingerprint:snmp - SNMPv2c fingerprinting module
[+] 13 modules registered
[+] Initializing scan engine
[+] Running scan engine
[-] ping:tcp_ping module: no closed/open TCP ports known on 192.168.1.120. Module test failed
[-] ping:udp_ping module: no closed/open UDP ports known on 192.168.1.120. Module test failed
[-] No distance calculation. 192.168.1.120 appears to be dead or no ports known
[+] Host: 192.168.1.120 is up (Guess probability: 50%)
[+] Target: 192.168.1.120 is alive. Round-Trip Time: 0.02027 sec
[+] Selected safe Round-Trip Time value is: 0.04053 sec
[-] icmp_port_unreach::build_dns_reply(): gethostbyname() failed! Using static ip for www.securityfocus.com in UDP probe
[-] fingerprint:smb need either TCP port 139 or 445 to run
[-] fingerprint:snmp: need UDP port 161 open
[+] Primary guess:
[+] Host 192.168.1.120 Running OS: "Microsoft Windows 2003 Server Enterprise Edition" (Guess probability: 100%)
[+] Other guesses:
[+] Host 192.168.1.120 Running OS: "Microsoft Windows 2003 Server Standard Edition" (Guess probability: 100%)

```

FIGURE 3.16
Xprobe2 Fingerprinting Results.

3.3.3.6 Httpprint

Suppose you run across a Web server and you want to know the HTTP daemon running, without loading a big fingerprinting tool that might trip IDS sensors. Httpprint is designed for just such a purpose. It only fingerprints HTTP servers, and it does both banner grabbing as well as signature matching against a signature file. In Fig. 3.17, you can see where httpprint is run against the Web server for a test system, using -h for the host and -PO for no ICMP ping, and where it designates the signatures with -s signatures.txt.

Httpprint is not in the standard path for the root user if you're using the BackTrack toolset, so you must run it via the program list or CD into the directory /pentest/enumeration/www/httpprint_301/linux. The resulting banner specifies Apache 2.0.55 and the nearest signature match is Apache 2.0.x, which matches up. Listed beneath that output are all signatures that were included, and then a score and confidence rating for that particular match.

Httpprint USAGE

How to use:

```
httpprint {-h <host> -j <input file> -x <nmap xmlfile>} -s
<signatures> [Options]
```

Input fields:

Target Specification:

- h can be used where <host> is a DNS hostname or IP address
- j can be used to read in data from a specific <input file>
- x will use an Nmap-generated XML file for input as specified by <nmap xmlfile>

-s specifies the file where the signatures are stored using the identifier <signatures>

[Options] are very wide ranging and include the following:

- o <outputfile>—Output file for HTML results
- t <timeout>—Connection/read timeout
- P0—Turn off ICMP ping
- th <threads>—Number of threads
- B—Blindly guess open TCP ports

A list of all options can be seen by using the -? option.

Output:

Displays web host signature and banner information as well as other potential matches and confidence levels.

Typical output:

```
root@bt:~/pentest/enumeration/www/httpprint/linux# ./httpprint -h 192.168.1.100 -P0 -s s
signatures.txt
httpprint v0.301 (beta) - web server fingerprinting tool
(c) 2003-2005 net-square solutions pvt. ltd. - see readme.txt
http://net-square.com/httpprint/
httpprint@net-square.com

Finger Printing on http://192.168.1.100:80/
Finger Printing Completed on http://192.168.1.100:80/
.....
Host: 192.168.1.100
Derived Signature:
Apache/2.0.55 (Unix) PHP/5.1.2
811c90c56ed3c295811c90c5811c90c5811c90c5505fcfe84276e4b8811c90c5
0d764585811c90c5811c90c5cd37187c811c90c5811c90c5811c90c5811c90c5
6ED3C2956ED3C2956ED3C295811c90c5E2CE6927050C50336ED3C2959E4318C8
6ED3C2956ED3C2952A20084C6ED3C2956ED3C2956ED3C2956ED3C295E2CE6923
E2CE69236ED3C29500614824E2CE6927E2CE6923

Banner Reported: Apache/2.0.55 (Unix) PHP/5.1.2
Banner Deduced: Apache/2.0.x
Score: 105
Confidence: 63.25
.....
Scores:
Apache/2.0.x: 105 63.25
Apache/1.3.[4-24]: 97 47.71
Apache/1.3.27: 96 45.97
Apache/1.3.26: 96 45.97
Apache/1.3.[1-3]: 92 39.40
TUX/2.0 (Linux): 88 33.47
Apache/1.2.6: 82 25.70
Com21 Cable Modem: 70 13.83
WebSitePro/2.3.18: 70 13.83
Oracle Servlet Engine: 69 13.04
```

FIGURE 3.17

Httpprint Fingerprinting Results.

3.3.3.7 Ike-scan: VPN assessment

One of the more common virtual private network (VPN) implementations involves the use of IPsec tunnels. Different manufacturers have slightly different usages of IPsec, which can be discovered and fingerprinted using ike-scan. IKE stands for Internet Key Exchange, and you use it to provide a secure basis for establishing an IPsec-secured tunnel. You can run ike-scan in two different modes, Main and Aggressive (-A), each which can identify different VPN implementations. Both operate under the principle that VPN servers will attempt to establish communications to a client that sends only the initial portion of an IPsec handshake. An initial IKE packet is sent (with Aggressive mode, a User ID can also be specified), and based on the time elapsed and types of responses sent, the VPN server can be identified based on service fingerprints.

In addition to the VPN fingerprinting functionality, ike-scan also includes psk-crack, which is a program that is used to dictionary-crack Pre-Shared Keys (psk) used for VPN logins. Ike-scan does not have fingerprints for all VPN vendors, and because the fingerprints change based on version increases as well, you may not find a fingerprint for your specific VPN. However, you can still gain useful information, such as the authentication type and encryption algorithm used. Fig. 3.18 shows ike-scan running against a Cisco VPN server. The default type of scan, Main, shows that an IKE-enabled VPN server is running on the host. When using the Aggressive mode (-A), the scan returns much more information, including the detected VPN based on the fingerprint. The -M flag is used to split the output into multiple lines for easier readability.

Ike-scan USAGE

How to use:

ike-scan [Options] [Hosts]

Input fields:

[Options] are very extensive and a list of all options can be seen by using the -h option.

Output:

```

root@bt:~# ike-scan -v -M 144.15.
DEBUG: pkt len=336 bytes, bandwidth=56000 bps, int=52000 us
Starting ike-scan 1.9 with 1 hosts (http://www.gna-monster.com/tools/ike-scan/)
144.15.      Main Mode Handshake returned
HDR=(CKY-R=0bd3b75e4077709c)
SA=(Enc=3DES Hash=MD5 Group=2:modp1024 Auth=PSK LifeType=Seconds LifeDurati
on=28800)
VID=4048b7d56ebce88525e7de7f00d6c2d3c0000000 (IKE Fragmentation)

Ending ike-scan 1.9: 1 hosts scanned in 0.081 seconds (12.37 hosts/sec). 1 returne
d handshake; 0 returned notify
root@bt:~#
```

FIGURE 3.18

Ike-scan Results.

3.3.3.8 SNMP

SNMP is one of the protocols which can be used for enumeration but is often forgotten by penetration testers and system administrators alike. That generally means that there is an opportunity there to gather a great deal of system information from a source that may not be secured very well. For example, the SNMP community string “public” is frequently used to monitor network devices and servers. Using a few simple tools, we can view extensive and useful information on many systems. More frightening than that is that the community string “private” is often the default for allowing modification of system configurations!

3.3.3.8.1 Snmpwalk

Snmpwalk is a tool which allows you to pull detailed information using SNMP from a supporting device or system. Many different options are available for snmpwalk, but to start, let’s take a look at some basic commands. First, let’s see what happens if we scan a Windows system using the default community string:

```
snmpwalk -c public -v1 192.168.1.120 1
```

Figure 3.19 shows the result of this scan. As you can see, there is a huge amount of data presented. By using some of the options available with snmpwalk, you can prune down the amount of data to some of the more useful nuggets. For example, consider the following syntax instead:

```
snmpwalk -c public -v1 192.168.1.120 SNMPv2-MIB:: sysDescr .0
```

The results of this are shown in Fig. 3.20 and are much more useful to us for a quick look at the host.

Snmpwalk USAGE

How to use: <agent>
snmpwalk [Options]

Input fields:
-v<version>--SNMP version designator
[Options] are very extensive and include:
-c<string>--Community string
-t<value>--Timeout

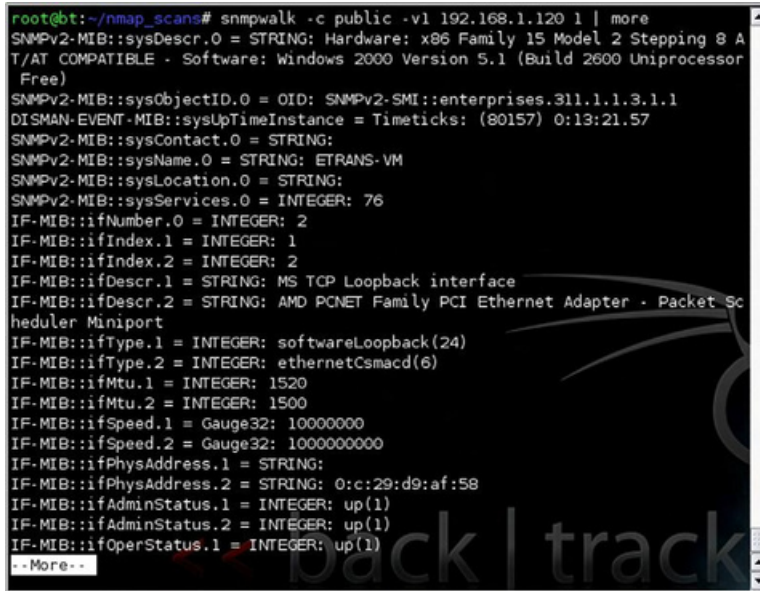
A list of all options can be seen by using the -h option.

Agent is the host and MIB to use.

Output:

Displays all data gathered from the SNMP MIB.

Typical output:



```

root@bt:~/nmap_scans# snmpwalk -c public -v1 192.168.1.120 1 | more
SNMPv2-MIB::sysDescr.0 = STRING: Hardware: x86 Family 15 Model 2 Stepping 8 A
T/AT COMPATIBLE - Software: Windows 2000 Version 5.1 (Build 2600 Uniprocessor
Free)
SNMPv2-MIB::sysObjectID.0 = OID: SNMPv2-SMI::enterprises.311.1.1.3.1.1
DISMAN-EVENT-MIB::sysUpTimeInstance = Timeticks: (80157) 0:13:21.57
SNMPv2-MIB::sysContact.0 = STRING:
SNMPv2-MIB::sysName.0 = STRING: ETRANS-VM
SNMPv2-MIB::sysLocation.0 = STRING:
SNMPv2-MIB::sysServices.0 = INTEGER: 76
IF-MIB::ifNumber.0 = INTEGER: 2
IF-MIB::ifIndex.1 = INTEGER: 1
IF-MIB::ifIndex.2 = INTEGER: 2
IF-MIB::ifDescr.1 = STRING: MS TCP Loopback interface
IF-MIB::ifDescr.2 = STRING: AMD PCNET Family PCI Ethernet Adapter - Packet Sc
heduler Miniport
IF-MIB::ifType.1 = INTEGER: softwareLoopback(24)
IF-MIB::ifType.2 = INTEGER: ethernetCsmacd(6)
IF-MIB::ifMtu.1 = INTEGER: 1520
IF-MIB::ifMtu.2 = INTEGER: 1500
IF-MIB::ifSpeed.1 = Gauge32: 100000000
IF-MIB::ifSpeed.2 = Gauge32: 1000000000
IF-MIB::ifPhysAddress.1 = STRING:
IF-MIB::ifPhysAddress.2 = STRING: 0:c:29:d9:af:58
IF-MIB::ifAdminStatus.1 = INTEGER: up(1)
IF-MIB::ifAdminStatus.2 = INTEGER: up(1)
IF-MIB::ifOperStatus.1 = INTEGER: up(1)
.. More ..

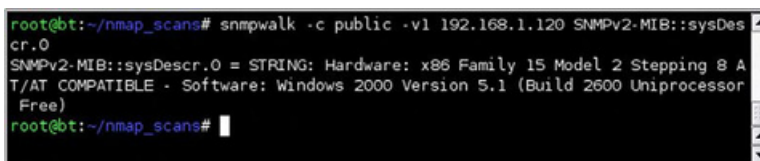
```

FIGURE 3.19

Snmpwalk Full Results.

What else can we do with this? There are many options. Take a look at the Management Information Base (MIB) support options from Microsoft at <http://support.microsoft.com/kb/237295>. This details out the MIBs supported by each OS which can help you see what options are available to you. For another example, try this command:

```
snmpwalk -c public -v1 192.168.1.120 1 | grep
hrSWI nstal ledName
```



```

root@bt:~/nmap_scans# snmpwalk -c public -v1 192.168.1.120 SNMPv2-MIB::sysDes
cr.0
SNMPv2-MIB::sysDescr.0 = STRING: Hardware: x86 Family 15 Model 2 Stepping 8 A
T/AT COMPATIBLE - Software: Windows 2000 Version 5.1 (Build 2600 Uniprocessor
Free)
root@bt:~/nmap_scans#

```

FIGURE 3.20

Snmpwalk System Description.

3.3.3.8.2 snmpenum.pl

The snmpenum.pl tool can be used to quickly enumerate most of the useful information available through the MIBs available on a variety of systems. By executing

this tool against a host, it will send the appropriate SNMP packets, gather the resulting data, and format it in a nicely readable form for you to make use of. An example of the use of `snmpenum.pl` is shown in Fig. 3.21.

snmpenum.pl USAGE

How to use: `<host> <community string> <config file>`

`snmpenum.pl`

`<host>` is the IP address to scan.

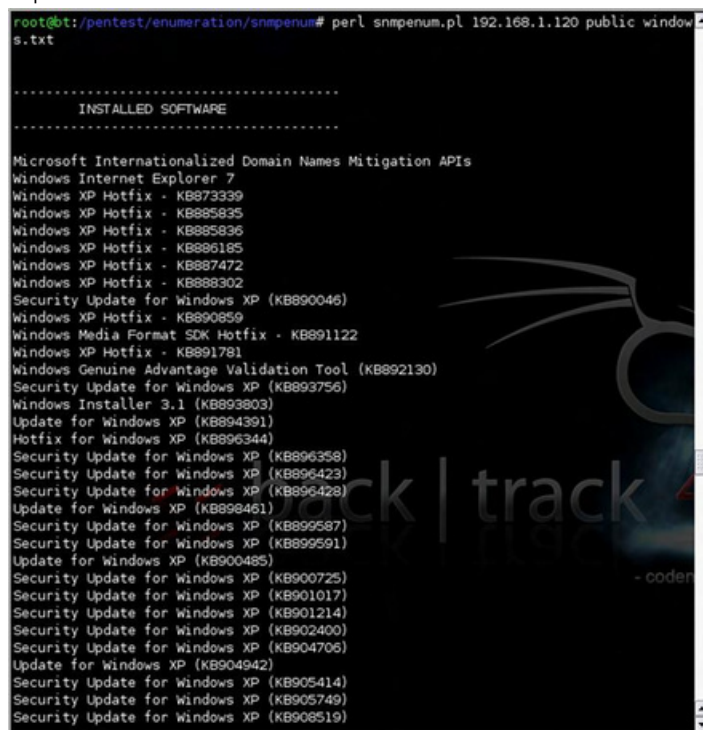
`<community string>` is the community string to use for authentication.

`<config file>` specifies the config file to use for the scan which differs based on the type of system being scanned.

Output:

Displays all data gathered from the SNMP MIB in an easy to read format.

Typical output:



```

root@bt: /pentest/enumeration/snmpenum# perl snmpenum.pl 192.168.1.120 public windows.txt
-----
INSTALLED SOFTWARE
-----

Microsoft Internationalized Domain Names Mitigation APIs
Windows Internet Explorer 7
Windows XP Hotfix - KB873339
Windows XP Hotfix - KB885835
Windows XP Hotfix - KB885836
Windows XP Hotfix - KB886185
Windows XP Hotfix - KB887472
Windows XP Hotfix - KB888302
Security Update for Windows XP (KB890046)
Windows XP Hotfix - KB890859
Windows Media Format SDK Hotfix - KB891122
Windows XP Hotfix - KB891781
Windows Genuine Advantage Validation Tool (KB892130)
Security Update for Windows XP (KB893756)
Windows Installer 3.1 (KB893803)
Update for Windows XP (KB894391)
Hotfix for Windows XP (KB896344)
Security Update for Windows XP (KB896358)
Security Update for Windows XP (KB896423)
Security Update for Windows XP (KB896428)
Update for Windows XP (KB896461)
Security Update for Windows XP (KB899587)
Security Update for Windows XP (KB899591)
Update for Windows XP (KB900485)
Security Update for Windows XP (KB900725)
Security Update for Windows XP (KB901017)
Security Update for Windows XP (KB901214)
Security Update for Windows XP (KB902400)
Security Update for Windows XP (KB904706)
Update for Windows XP (KB904942)
Security Update for Windows XP (KB905414)
Security Update for Windows XP (KB905749)
Security Update for Windows XP (KB908519)

```

FIGURE 3.21
snmpenum.pl Output.

As you can see from the results shown in [Fig. 3.21](#), `snmpenum.pl` can save a lot of time spent analyzing the SNMP results and allows you to quickly get some great information about your target system. It is very valuable to use this often forgotten service to enumerate massive amounts of usable data.

TIP

What about SMB? Since the MS Blaster, Nimda, Code-Red, and numerous LSASS.EXE worms spread with lots of media attention, it seems that users and system administrators alike are getting the word that running NetBIOS, SMB, and Microsoft-ds ports open to the Internet is a Bad Thing. Because of that, you will not see many external penetration tests where lots of time is spent enumerating for NetBIOS and SMB unless open ports are detected. Keep this in mind when you are scanning. Although the security implications are huge for finding those open ports, do not spend too much time looking for obvious holes that most administrators already know about.

3.3.3.9 Nbtscan

When you encounter Windows systems (remember, TCP ports such as 135, 137, 139, and 445) on the target network, you may be able to use a NetBIOS broadcast to query target machines for information. Nbtscan acts as a Windows system by querying local systems for NetBIOS resources. Usage is rather simple; you can launch `nbtscan` at either a single IP address or an entire range. Scanning for resources is a fairly quick affair, as it has to broadcast only one query and then wait for the responses. [Fig. 3.22](#) shows `nbtscan`'s output from a class C network scan.

Nbtscan USAGE

How to use: `<scan range>`

`nbtscan [Options]`

[Options] are extensive and include:

Input fields:

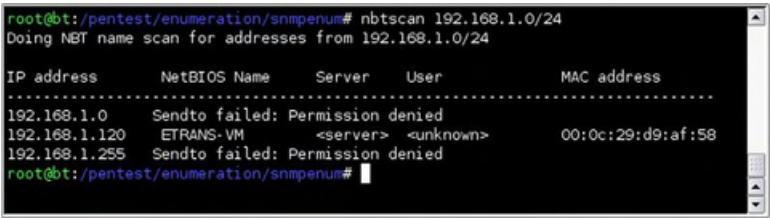
- v—Output verbosity
- s<separator>—Output in script-friendly format using designated separator
- h—Use human-readable format for services
- t<value>—Timeout

A list of all options can be seen by running `nbtscan` with no options.

Output:

Displays all data systems which respond to the scan including their IP address, name, services, user, and MAC address.

Typical output:



```
root@bt:/pentest/enumeration/snmpenum# nbtscan 192.168.1.0/24
Doing NBT name scan for addresses from 192.168.1.0/24

IP address      NetBIOS Name    Server  User      MAC address
-----
192.168.1.0      Sendto failed:  Permission denied
192.168.1.120    ETRANS-VM       <server>  <unknown>  00:0c:29:d9:af:58
192.168.1.255    Sendto failed:  Permission denied
root@bt:/pentest/enumeration/snmpenum#
```

FIGURE 3.22
Nbtscan Output.

3.3.3.10 Nmap scripting

One of the more advanced features recently added to Nmap is the ability to create scripts enabling automation. These scripts can be used to automate a wide variety of functions including enumeration, vulnerability scans, and even exploitation. For example, the Nsploit tool (<http://trac.happypacket.net/>) has the ability to use Nmap to scan a target, and then automatically call Metasploit to attempt to exploit any identified vulnerabilities.

For the purposes of enumeration, these Nmap scripts can help automate some of your work and speed up your penetration testing process. More scripts are being developed constantly, but most security toolsets such as BackTrack include a number of basic scripts. In most cases, these scripts will be stored in the /usr/share/nmap/scripts or /usr/local/share/nmap/scripts directory.

To call one of the scripts, we will use the --script option for Nmap. Fig. 3.23 shows an example using the script “http-enum.nse” to enumerate some additional http information on a remote web server. In this example, the script was able to expand on the basic port and fingerprint data and provide us some details on directories which exist within the web server.

As you can see, the scripting capability of Nmap can be very useful. By looking at the source code for existing scripts, you can see how the scripts work as well as modify them for your own needs.

3.4 CASE STUDIES: THE TOOLS IN ACTION

Okay, here is where it all comes together, the intersection of the tools and the methodology. We will run through a series of scenarios based on external and internal penetration tests, including a very stealthy approach and a noisy IDS test. We will treat these scenarios as the initial rounds in a penetration test and will give a scope for each engagement. The goal for these case studies is to determine enough information about the targets to move intelligently into the exploitation

```

root@bt:~# nmap -sS -sV 192.168.1.120 --script http-enum.nse

Starting Nmap 5.30BETA1 ( http://nmap.org ) at 2010-08-07 09:11 CDT
NSE: Script Scanning completed.
Nmap scan report for 192.168.1.120
Host is up (0.0014s latency).
Not shown: 988 closed ports
PORT      STATE SERVICE      VERSION
21/tcp    open  ftp          FileZilla ftpd
25/tcp    open  smtp         Mercury/32 smtpd (Mail server account Maiser)
79/tcp    open  finger       Mercury/32 fingerd
80/tcp    open  http         Apache httpd 2.2.14 ((win32) DAV/2 mod_ssl/2.2.14 Open
SSL/0.9.8l mod_autoindex_color PHP/5.3.1 mod_apreq2-20090110/2.7.1 mod_perl/2.0.4
Perl/v5.10.1)
| http-enum:
| /icons/: Icons and images
|_ /phpmyadmin/: phpMyAdmin
106/tcp   open  pop3pw       Mercury/32 poppass service
110/tcp   open  pop3         Mercury/32 pop3d
135/tcp   open  msrpc        Microsoft Windows RPC
139/tcp   open  netbios-ssn  Microsoft Windows RPC
143/tcp   open  imap         Mercury/32 imapd 4.72
443/tcp   open  ssl/http     Apache httpd 2.2.14 ((win32) DAV/2 mod_ssl/2.2.14 Open
SSL/0.9.8l mod_autoindex_color PHP/5.3.1 mod_apreq2-20090110/2.7.1 mod_perl/2.0.4
Perl/v5.10.1)
| http-enum:
| /icons/: Icons and images
|_ /phpmyadmin/: phpMyAdmin
445/tcp   open  microsoft-ds Microsoft Windows XP microsoft-ds
3306/tcp   open  mysql        MySQL (unauthorized)
MAC Address: 00:0C:29:D9:AF:58 (VMware)
Service Info: Host: localhost; OS: Windows

Service detection performed. Please report any incorrect results at http://nmap.org/submit/.
Nmap done: 1 IP address (1 host up) scanned in 24.75 seconds
root@bt:~#

```

FIGURE 3.23

Nmap http-enum.nse Script Results.

phase. IP addresses have been changed or obfuscated to protect the (clueless) innocent.

3.4.1 External

The target for this attack is a single address provided by the client. There is no IDS, but a firewall is involved. The target DNS name is faircloth.is-a-geek.org.

The first step is to perform a WHOIS lookup, ping, and host queries to make sure the system is truly the target. Running WHOIS faircloth.is-a-geek.org returns NOT FOUND, so we do a WHOIS on the domain only, is-a-geek.org. This returns registration information for DynDNS.org, which means that the target is likely a dynamic IP address using DynDNS for an externally reachable DNS name. This is commonly used for home systems, or those that may not be reachable 100 percent of the time. A dig faircloth.is-a-geek.org returns the IP address of 68.89.112.40, the target IP address.

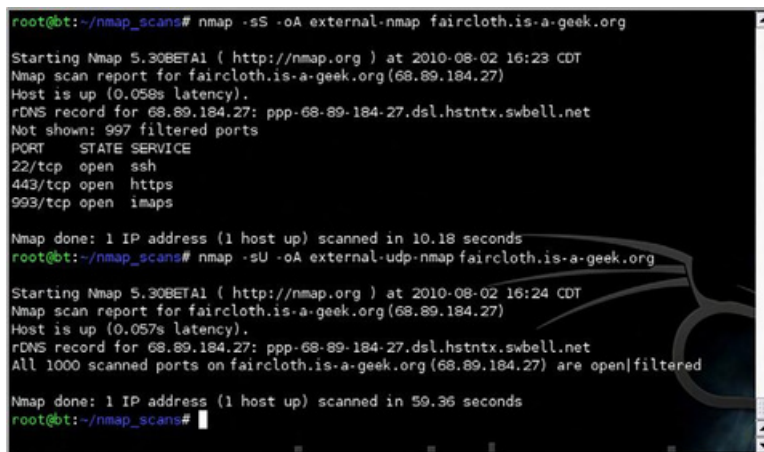
Performing a reverse lookup with host 68.89.112.40 gives a different host name than the one provided: adsl-68-89-172-40.dsl.hstntx.swbell.net. SWBell.net is

the domain for SBC Communications, an ISP, and “hstntx” in the domain name leads us to believe that the IP address may be terminated in Houston, TX. This may not be useful information right now, but any information about the target could be useful further into the test. Also note that at this point, not a single ping has been sent to the target, so all reconnaissance thus far has been totally indirect.

In Fig. 3.24, we run `nmap -sS -oA external-nmap faircloth.is-a-geek.org`, which performs a SYN scan, writing the output to the files `external-nmap`. This scan returns three TCP ports `open22`, `443`, and `993`. To check for any UDP-based services, we also run `nmap -sU -oA external-udp-nmap faircloth.is-a-geek.org`, which returns indicating that all scanned ports are open or filtered as shown in Fig. 3.24.

To identify what those open ports are running, we can use Nmap again using the `-sV` and `-O` options to do some fingerprinting. This reveals that the target is running OpenSSH 5.1-p1, with protocol version 2.0; port 443 shows as Apache 2.2.11 (Ubuntu) with PHP 5.2.6; and 993 returns as SSL (however, it is also the IANA-assigned port for IMAP over Secure Sockets Layer [SSL]) and looks to be running Courier Imapd. OS detection is a little questionable, but based on the service information, we can assume that we’re dealing with Ubuntu. Fig. 3.25 shows the exact output and execution of the Nmap command.

Although this process was very direct and simple, the point of this case study is to show how straightforward a basic external scan and enumeration can be. Each discovered software product would be investigated to search for known vulnerabilities, and further testing would be performed against the software to determine any misconfigurations.



```

root@bt:~/nmap_scans# nmap -sS -oA external-nmap faircloth.is-a-geek.org

Starting Nmap 5.30BETA1 ( http://nmap.org ) at 2010-08-02 16:23 CDT
Nmap scan report for faircloth.is-a-geek.org (68.89.184.27)
Host is up (0.058s latency).
rDNS record for 68.89.184.27: ppp-68-89-184-27.dsl.hstntx.swbell.net
Not shown: 997 filtered ports
PORT      STATE SERVICE
22/tcp    open  ssh
443/tcp    open  https
993/tcp    open  imap

Nmap done: 1 IP address (1 host up) scanned in 10.18 seconds
root@bt:~/nmap_scans# nmap -sU -oA external-udp-nmap faircloth.is-a-geek.org

Starting Nmap 5.30BETA1 ( http://nmap.org ) at 2010-08-02 16:24 CDT
Nmap scan report for faircloth.is-a-geek.org (68.89.184.27)
Host is up (0.057s latency).
rDNS record for 68.89.184.27: ppp-68-89-184-27.dsl.hstntx.swbell.net
All 1000 scanned ports on faircloth.is-a-geek.org (68.89.184.27) are open|filtered

Nmap done: 1 IP address (1 host up) scanned in 59.36 seconds
root@bt:~/nmap_scans#

```

FIGURE 3.24

Nmap Results for `faircloth.is-a-geek.org`.

```

root@bt:~/nmap_scans# nmap -sS -sV -O -oA external-enum-nmap faircloth.is-a-geek.org

Starting Nmap 5.30BETA1 ( http://nmap.org ) at 2010-08-02 16:31 CDT
Nmap scan report for faircloth.is-a-geek.org (68.89.184.27)
Host is up (0.059s latency).
rDNS record for 68.89.184.27: ppp-68-89-184-27.dsl.hstntx.swbell.net
Not shown: 996 filtered ports
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 5.1p1 Debian Subuntul (protocol 2.0)
443/tcp    open  ssl/http Apache httpd 2.2.11 ((Ubuntu) PHP/5.2.6-3ubuntu4.5 with Suhosin-Patch mod_ssl/2.2.11 OpenSSL/0.9.8g)
465/tcp    open  smtps?
993/tcp    open  ssl/imap Courier Imapd (released 2008)
Warning: OSScan results may be unreliable because we could not find at least 1 open and 1 closed port
Aggressive OS guesses: AVM FRITZ!Box FON WLAN 7170 WAP (91%), HP Brocade 4Gb SAN switch (91%), Linux 2.4.20 (91%), Linux 2.4.21 (embedded) (91%), Linux 2.6.24 (Ubuntu 8.04, x86) (91%), Acorp W400G or W422G wireless ADSL modem (MontaVista embedded Linux 2.4.17) (90%), MontaVista embedded Linux 2.4.17 (90%), Google Mini search appliance (89%), Link sys WRV200 wireless broadband router (89%), Linux 2.6.20 (Ubuntu 7.04 server, x86) (89%)
No exact OS matches for host (test conditions non-ideal).
Service Info: OS: Linux

OS and Service detection performed. Please report any incorrect results at http://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 100.90 seconds
root@bt:~/nmap_scans#

```

FIGURE 3.25

Nmap Fingerprinting Results for faircloth.is-a-geek.org.

3.4.2 Internal

For our internal case study, we will scan and enumerate the 192.168.1.0/24 network. No internal network firewalls exist, but host firewalls are installed.

Performing a ping sweep using `nmap -sP -PA -oA intcase-nmap-sweep`

192.168.1.0/24 reveals four targets, shown in Fig. 3.26.

and given the IP address of 192.168.1.200,

To provide a thorough scan, we ran `nmap -sS -sV -O -iL valid-hosts -oA full-internal-scan`, where `valid-hosts` was created through the use of the earlier `awk` command shown in Fig. 3.2. Interesting items of note from this scan include an IIS 6.0 web server on 10.0.0.99 (a Windows 2003 Server system) and a mail server running SMTP and IMAP on 10.0.0.9 (a Linux system). These two



```

root@bt:~/nmap_scans# nmap -sP -PA -oA intcase-nmap-sweep 192.168.1.0/24
Starting Nmap 5.30BETA1 ( http://nmap.org ) at 2010-08-02 16:13 CDT
Nmap scan report for 192.168.1.1
Host is up.
Nmap scan report for 192.168.1.100
Host is up (0.0017s latency).
MAC Address: 00:0C:29:67:63:F5 (VMware)
Nmap scan report for 192.168.1.110
Host is up (0.0016s latency).
MAC Address: 00:0C:29:A2:C6:E8 (VMware)
Nmap scan report for 192.168.1.120
Host is up (0.0086s latency).
MAC Address: 00:0C:29:D9:AF:58 (VMware)
Nmap done: 256 IP addresses (4 hosts up) scanned in 28.04 seconds
root@bt:~/nmap_scans#

```

FIGURE 3.26

Ping Sweep.

servers seem to comprise most of the infrastructure needed for a small network. Information such as this will set up further attack scenarios. See the following output for the Nmap results:

```

# Nmap 5.30BETA1 scan initiated Mon Aug 2 16:56:37 2010
as: nmap -sS -sV -O -iL valid_hosts -oA full-internal-
scan
Nmap scan report for 192.168.1.100
Host is up (0.0051s latency).
Not shown: 992 filtered ports
PORT      STATE SERVICE VERSION
20/tcp    closed ftp-data
21/tcp    open  ftp      vsftpd(broken:couldnotbind
to listening IPv4 socket)
22/tcp    open  ssh      OpenSSH4.3(protocol1.99)
25/tcp    n     smt      Sendmail8.13.7/8.13.7
80/tcp    open  http     Apache/2.0.55((Unix)PHP/
5.2.6)
110/tcp   open  pop3     Openwallpopa3d
143/tcp   open  imap     UWimapd2004.357
443/tcp   closed https
MAC Address: 00:0C:29:67:63:F5 (VMware)
Device type: general purpose
Running: Linux 2.6.X
OS details: Linux 2.6.13 - 2.6.28
Network Distance: 1 hop
Service Info: Host: slax.example.net; OS: Unix
Nmap scan report for 192.168.1.110

```

```

Host is up (0.0046s latency).
Not shown: 996 closed ports
PORT      STATE SERVICE VERSION
21/tcp    open  ftp    vsftpd 2.0.4
22/tcp    open  ssh?
80/tcp    open  http?
631/tcp   open  ipp     CUPS1.1
MAC Address: 00:0C:29:A2:C6:E6 (VMware)
Device type: general purpose
Running: Linux 2.6.X
OS details: Linux 2.6.13 - 2.6.28
Network Distance: 1 hop
Service Info: OS: Unix
Nmap scan report for 192.168.1.120
Host is up (0.0064s latency).
Not shown: 988 closed ports
PORT STATE SERVICE VERSION
21/tcp    open  ftp      FileZillaftpd
25/tcp    open  smtp     Mercury/32smtpd(Mailserver
25/tcp    open  smtp     Mercury/32smtpd(Mailserver
79/tcp    open  finger   Mercury/32fingerd
80/tcp    open  http     Apachehttpd2.2.14((Win32)DAV/2
80/tcp    open  ssl/http OpenSSL/0.9.8l mod_autoindex_color
PHP/ 5.3.11 mod_apreq2-20090110/2.7.1 mod_perl/2.0.4
Perl/ v5.10 .1) 106/tcp 110/tcp 135/tcp 139/tcp 143/tcp
443/tcp   open  pop3pw   Mercury/32 poppass service
         open  pop3     Mercury/32pop3d
         open  msrpc    MicrosoftWindowsRPC
         open  netbios-ssn
         open  imap     Mercury/32 imapd 4.72
         open  ssl/http Apachehttpd2.2.14((Win32)
DAV/2 mod_ssl/2.2.14 OpenSSL/0.9.8l mod_autoindex_color
PHP/5.3.11 mod_apreq2-20090110/2.7.1 mod_perl/2.0.4 Perl/
v5.10 .1)
445/tcp   open  microsoft-ds Microsoft Windows XP
micro soft- ds
3306/tcp   open  mysql    MySQL (unauthorized)
MAC Address: 00:0C:29:D9:AF:58 (VMware)
Device type: general purpose
Running: Microsoft Windows XP
OS details: Microsoft Windows XP Professional SP2 or
Windows Server 2003
Network Distance: 1 hop
Service Info: Host: localhost; OS: Windows

```

OS and Service detection performed. Please report any incorrect results at <http://nmap.org/submit/>.
 # Nmap done at Mon Aug 2 16:59:30 2010 -- 3 IP addresses
 (3 hosts up) scanned in 173.53 seconds


As a server running Windows was detected, we could use `nbtscan` to pull any information from that target. The NetBIOS name detected was `ETRANS-VM`. As some of these targets also have DNS names registered and others do not, dynamic DNS may not be enabled for this particular network. The `-v` option is used for `nbtscan` to show the full and verbose NBT resources offered, as well as the Media Access Control (MAC) address of the targets. Fig. 3.27 shows the results from `nbtscan`.

3.4.3 Stealthy

To demonstrate a stealthy approach, we will target an internal host that may or may not have an IDS or a firewall. Either way, we will attempt to avoid tripping sensors until we know more information about the system. The IP address of this target is `192.168.1.100`.

First, we will need to perform a port scan, but one that an IDS will not notice. To do this we will be combining a slow targeted Nmap scan with a firewall rule that will drop the automatic RST packet sent back to the target, by creating an iptables rule using `iptables -A OUTPUT -p tcp --tcp-flags RST RST -d`

`192.168.1.100 -j DROP`. By expanding on the same principle, you can create rules that will drop packets depending on the scan type, such as a FIN scan; `iptables -A OUTPUT -p tcp --tcp-flags FIN FIN -d 192.168.1.100` will trigger the rule creation, dropping FIN packets once they are detected by the scan.



```

root@bt:~# nbtscan -v 192.168.1.120
Doing NBT name scan for addresses from 192.168.1.120

NetBIOS Name Table for Host 192.168.1.120:

Name                Service          Type
-----
ETRANS-VM            <00>              UNIQUE
MSHOME               <00>              GROUP
ETRANS-VM            <20>              UNIQUE
MSHOME               <1e>              GROUP
MSHOME               <1d>              UNIQUE
__MSBROWSE__         <01>              GROUP

Adapter address: 00:0c:29:d9:af:58
root@bt:~#

```

FIGURE 3.27

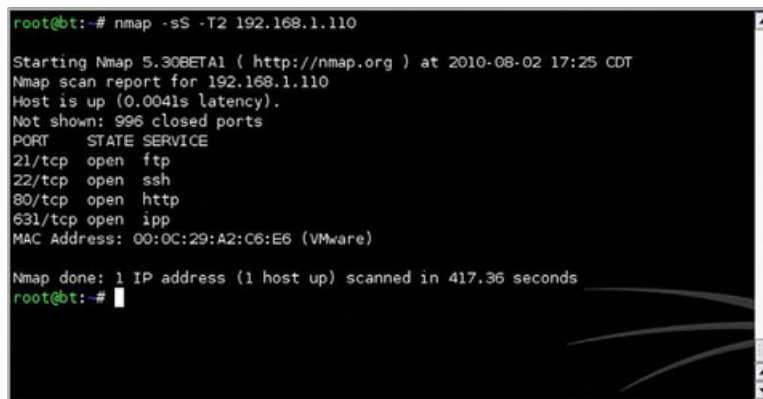
nbtscan Results.

If you want to use iptables to automate this process, perhaps on a standing scan system, you may also investigate the use of the iptables RECENT module, which allows you to specify limits and actions on the reception of specific packets. Something similar to the following code might be useful for this purpose. This should drop any FIN packets outbound from the scanner, except for one every 10 s. Legitimate traffic should resend without much trouble, but the scanner should not resend. Note that this will work for only one port checked every 10 s.

```
iptables -A OUTPUT -m recent --name FIN-DROP --rcheck
--rdest --proto tcp --tcp-flags FIN FIN --seconds 10 -j
DROP
iptables -A OUTPUT -m recent --name FIN-DROP --set
--rdest --proto tcp --tcp-flags FIN FIN -j ACCEPT
```

Now that the iptables rules are set up, we launch a SYN scan directly to the target with no additional scans, such as version or fingerprint. We do, however, slow down the scan by using Nmap's "Polite" timing template. We could also use the "Sneaky" timing template for this to slow the scan down further and reduce the possibilities of being identified. The resultant commands used are `nmap -sS -T2 192.168.1.110`. Fig. 3.28 shows the results from the scan.

As far as the results go, they show FTP, SSH, HTTP, and IPP being available on the target system. With this variety of services, it would be difficult to fingerprint from this information alone. To get a more complete picture of the system, we launch a targeted service identification scan using Nmap against three services that should give a more proper view of the system fingerprint. SSH, SMTP, and IMAP are targeted and send packets only once every 15 s, using the command `nmap -sV -T1 -p21,22,80 192.168.1.100`. Fig. 3.29 shows the results from that



```
root@bt:~# nmap -sS -T2 192.168.1.110

Starting Nmap 5.30BETA1 ( http://nmap.org ) at 2010-08-02 17:25 CDT
Nmap scan report for 192.168.1.110
Host is up (0.0041s latency).
Not shown: 996 closed ports
PORT      STATE SERVICE
21/tcp    open  ftp
22/tcp    open  ssh
80/tcp    open  http
631/tcp   open  ipp
MAC Address: 00:0C:29:A2:C6:E6 (VMware)

Nmap done: 1 IP address (1 host up) scanned in 417.36 seconds
root@bt:~#
```

FIGURE 3.28

Stealth Nmap Scan Results.



```

root@bt:~# nmap -sV -T1 -p21,22,80 192.168.1.110

Starting Nmap 5.30BETA1 ( http://nmap.org ) at 2010-08-02 17:33 CDT
Nmap scan report for 192.168.1.110
Host is up (0.0015s latency).
PORT      STATE SERVICE VERSION
21/tcp    open  ftp      vsftpd 2.0.4
22/tcp    open  ssh      OpenSSH 4.3 (protocol 1.99)
80/tcp    open  http?
MAC Address: 00:0C:29:A2:C6:E6 (VMware)
Service Info: OS: Unix

Service detection performed. Please report any incorrect results at http://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 225.52 seconds
root@bt:~#

```

FIGURE 3.29
Stealth Targeted Nmap Scan Results.

slow, targeted scan. From these results, we can guess with a high confidence level that this is a Linux server running as a VMware virtual machine.

Because this is a stealthy test, p0f would be useful if we simply wanted to get a system fingerprint. However, because we are doing an Nmap scan, p0f would be a bit redundant and would not provide much value to the scan.

3.4.4 Noisy (IDS) testing

For this example, the target (192.168.1.100) will have an IDS in-line so that all traffic will pass the IDS. The goal for this scan is to test that the IDS will pick up the “basics” by hammering the network with lots of malicious traffic.

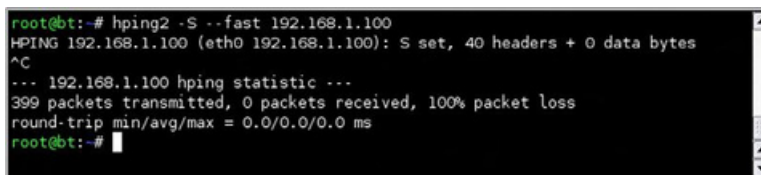
During this test, we will initiate a SYN flood from the scanner to the target, and a SYN scan with version scanning and OS fingerprinting will be performed during that scan. The hope is that the IDS does not detect the targeted scan due to the flood of traffic coming in from the scanner.

WARNING

Please note that testing of this type can be harmful to the network on which you are testing. Never do any type of testing that can create a DoS condition without explicitly getting permission or allowances for it first.

To initiate the SYN flood, we will use a tool called hping to send out SYN packets at a very fast rate. We do this with the command `hping2 -S --fast 192.168.1.100`, as shown in Fig. 3.30.

Once the flooding has started, launch an Nmap scan that will hopefully be masked in the torrent of SYN packets currently being sent. This scan uses a standard



```

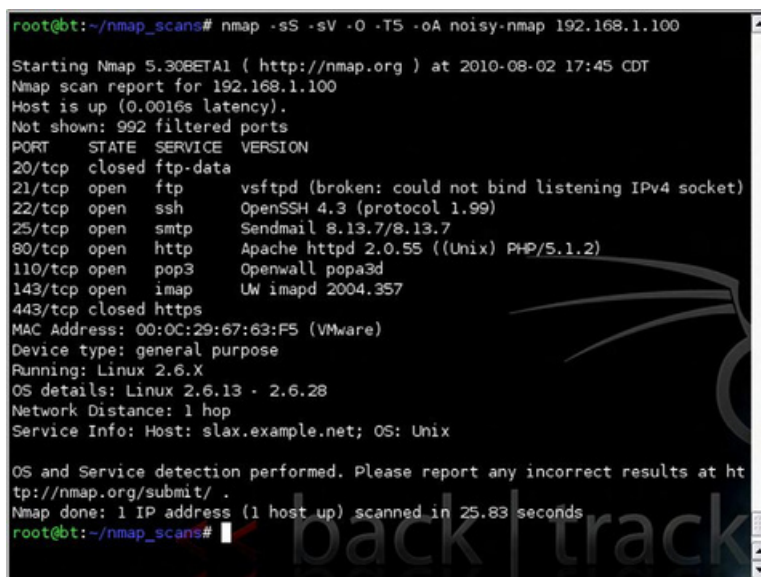
root@bt:~# hping2 -S --fast 192.168.1.100
HPING 192.168.1.100 (eth0 192.168.1.100): S set, 40 headers + 0 data bytes
^C
--- 192.168.1.100 hping statistic ---
399 packets transmitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
root@bt:~#

```

FIGURE 3.30
Hping SYN Flood.

SYN scan while performing service version matching and OS fingerprinting, all set at the highest rate of send for Nmap, `-T5` or `Insane`. Just in case the target is not returning ICMP pings, ping checking is disabled. Fig. 3.31 shows the output from this scan.

Since our scan was successful while we were flooding the target, the next step for the client would be to take a look at their IDS and see if they at least logged our scan. It's obvious that we weren't blocked, but we could have set off some alarms. This example shows one of the reasons that your documentation must be extensive and precise. The client may need to know the timestamp or source IP from your scan in order to correlate the data in their IDS logs.



```

root@bt:~/nmap_scans# nmap -sS -sV -O -T5 -oA noisy-nmap 192.168.1.100

Starting Nmap 5.30BETA1 ( http://nmap.org ) at 2010-08-02 17:45 CDT
Nmap scan report for 192.168.1.100
Host is up (0.0016s latency).
Not shown: 992 filtered ports
PORT      STATE SERVICE VERSION
20/tcp    closed ftp-data
21/tcp    open  ftp      vsftpd (broken: could not bind listening IPv4 socket)
22/tcp    open  ssh      OpenSSH 4.3 (protocol 1.99)
25/tcp    open  smtp     Sendmail 8.13.7/8.13.7
80/tcp    open  http     Apache httpd 2.0.55 ((Unix) PHP/5.1.2)
110/tcp   open  pop3     Openwall popa3d
143/tcp   open  imap     UW imapd 2004.357
443/tcp   closed https
MAC Address: 00:0C:29:67:63:F5 (VMware)
Device type: general purpose
Running: Linux 2.6.X
OS details: Linux 2.6.13 - 2.6.28
Network Distance: 1 hop
Service Info: Host: slax.example.net; OS: Unix

OS and Service detection performed. Please report any incorrect results at http://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 25.83 seconds
root@bt:~/nmap_scans#

```

FIGURE 3.31
Nmap SYN Scan with Background Noise.

EPIC FAIL

Sometimes during a penetration test your approach or attack vector may not work out. IP addresses may change, routes may vary or drop, or tools may stop working without any warning. Sometimes the test may succeed, but it will give unusual results. Even negative results may yield positive information, such as the fact that the firewall mimics open ports for closed ports. Make sure that when you find unusual information, you log it using as much detail as you would for expected information. The only bad information is not enough information.

Although this chapter represented just a simple use of the tools to perform an IDS test, the premise is the same no matter what. Try to overload the network with traffic while sneaking in your tool “under the radar” to get it past the alerts. If possible, encode any input you send through a system in a different character set than normal or even UTF-8 to avoid common ASCII string matches. If that is not an option, closely analyze the specific target you are assessing. Sometimes specific products have vulnerabilities reported that could allow you to configure your scanning tool in such a way that it will not trip any sensors when run.

3.5 HANDS-ON CHALLENGE

Throughout this chapter, we’ve studied scanning and enumeration for penetration testing of target systems. You should now have a good understanding of the approaches that we take with each as well as the core technologies used for this phase of penetration testing. In addition, we’ve looked at some tools you can use to perform these tasks efficiently and effectively. Lastly, we went through four real-world scenarios where we would use these techniques and tools to gather data on our targets.

With that in mind, it’s time to try it out in your world. Using a test lab, not a live production network, try performing some scanning and enumeration using the tools that we have discussed. This could be your home network or a dedicated lab environment depending on the resources that you have available. Again, documentation is key, so this is what you should be putting together as the results of your testing:

A list of “live” systems within your target environment
 The operating system type and version for each system
 A list of open ports on those systems
 The exact service, software, and version for each open port

This documentation should be added to the information you accumulated during the reconnaissance phase (if you used the same target for these challenges) and will be used for future penetration testing phases. Cumulatively, you should now have a list of DNS names, IP addresses, identified “live” or reachable IP addresses, as well as the details associated with those hosts.

SUMMARY

This chapter has focused on taking the data we gathered during the reconnaissance phases and expanding on them by using scanning and enumeration. This also covers

the “vitality” phase of reconnaissance. We focused first on our objectives related to scanning and enumeration. This includes availability of target hosts as well as gathering details about those hosts and the services offered by them.

We then moved on to the concept of scanning. We talked about the general approach to scanning and why scanning should be done. We also talked about methods to ensure that you’re making the most effective use of your time by scanning for the most common ports first and then expanding your scanning if you have additional time available. The core technologies used for scanning were our next topic and we went over these in some detail as those same technologies apply many times over in penetration testing. We went over a variety of open source tools which are available to help you in performing those important scanning operations and speeding up your penetration testing process.

Next we went

Our next topic was discussing the real-world scenarios that could be presented through a series of case studies. These case studies illustrated real scenarios that you could run into when doing penetration testing professionally. For each case study, we examined a method for accomplishing our goals and demonstrated the use of a number of tools and options for those tools that helped us to get the job done.

Finally, you got to try it yourself through our hands-on challenge and were presented with a task and appropriate deliverables for demonstrating your ability to use these techniques and tools.

Now that we’ve finished up with enumeration, we will have a list of targets that we can use for the next penetration testing stagedvulnerability scanning. We needed to have knowledge about specific services that are running, versions of those services, and any host or system fingerprinting that we could determine to successfully move to this next stage. Moving forward without that information would really hamper our efforts in exploitation.