Understanding Nmap analysis techniques

SCANNER

How does the popular Nmap scanner identify holes in network security?

In this article, we examine some Nmap analysis techniques.

BY CHRISTIAN NEY

Network scanning is nearly as old as the network. In former times, hackers used modems to test blocks of phone numbers and record responses in a process known as wardialing. Today, port scanners transmit specially crafted IP packets across the Internet to discover and identify live systems.

Nmap (the Network Mapper [1]), which was first introduced by Fyodor in September 1997 [2], is probably one of the most comprehensive network scanning tools. Fyodor was unhappy with the features that tools such as Strobe [3] or Pscan [4] offered. He wanted a utility that would out-perform anything seen before – a feat that he certainly achieved with Nmap.

The Nmap scanner is included with many popular Linux distributions, and if your distro doesn't have it, you can obtain it from the project website [1]. Nmap uses TCP fingerprinting to identify the operating system on a scanned host. It can tell you the uptime for a host and reliably identify listening services and versions.

The sheer number of functions provided with Nmap means that Nmap has an amazing number of command line parameters. There are no less than 15 methods for scanning hosts (Table 1), and up to 20 settings for each method – some of which can produce really strange looking packets. You do not need specialist skills to use Nmap, although beginning users may be confused by some of the more granular controls. To fully leverage Nmap's power, some background knowledge is a big advantage. This article describes some of the techniques Nmap uses for discovering vulnerabilities.

THE AUTHOR

Christian Ney is a Unix and Firewall administrator with a regional airline, and contributes to various open source projects in his free time.

Three Card Trick

Nmap uses a three-stage scanning process. By default, the three steps are as follows:

- Nmap pings the target system. The user can choose between legacy ICMP Echo Requests and Nmap's own techniques for identifying live hosts.
- By default, a reverse lookup is performed to discover the hostname associated with the system's IP address.
- Nmap scans the target ports using the selected technique. Pressing [Ctrl] + [C] quits the process, if you discover that it is taking too long. If you tell Nmap to write a logfile, the tool can pick up from where it left off at a later stage.

Nmap distinguishes between four possible states for ports (Table 2). Nmap's major strength is the wide range of scanning techniques it supports. Instead of simply opening a normal TCP connection, the program transmits specially crafted packets that actually contravene the RFCs. Nmap then draws conclusions based on the reaction from the target side. For some of these techniques, the tool needs root privileges, as it needs to work with raw sockets and autonomously craft packets.

Primitive Scanning

Without root privileges (Figure 3), Nmap is restricted to simple *connect()* scanning, which uses the operating system's connect functions to establish a RFC-compliant connection based on the three-way handshake. The scanner sends a TCP packet with the SYN flag (Figure 4) and appropriate source and target ports set. If there is no listener at the target port, the target system returns a RST packet in response to close the connection; otherwise the target re-

Not Just for the Bad Guys

Although Nmap is often maligned as a Black Hat tool, it also helps administrators by simplifying the process of network analysis. Network administrators can probe for vulnerabilities before attackers have a chance to discover forgotten and neglected services. The tool also helps admins check inventory, test firewall rules, and document patch versions.

Nmap was originally developed for Linux, although it now also runs on Windows, BSD, and various flavors of Unix. There are a number of graphical front-ends for Unix (Nmapfe, Figure 1) and for Redmond-based operating systems (Nmapwin [5]). PHP-Nmap [6] (see Figure 2) even allows users to control the scanner via a web browser. The Nmap homepage has a list with more Nmap-based projects at [7], including a port to the Sharp Zaurus.

Figure 1: The Nmap front-end removes the need to learn the enormous range of options, and gives the administrator a ready-to-run command line (see the bottom of the figure.)

sponds with a packet in which the SYN and ACK flags are set. An ACK packet completes the handshake, establishing the connection, which Nmap then immediately closes by issuing a reset (Listing 1).

Half-Open Scanning

If you have root privileges, TCP-SYN scanning is preferable. This technique saves network and system resources, does not rely on a specific operating system, and is stealthy. Instead of establishing a full TCP connection, Nmap simply transmits the first SYN packet from the three-way handshake. A closed port will react with a *RST* flag to clean up the half-open connection. An open port will respond with a SYN/ACK, which Nmap will then close by transmitting a reset (Listing 2). The scanned service does not notice this process, which does not leave any traces in the logfiles.

This simple trick will not fool intrusion detection systems such as Snort [8] or Prelude [9]. An unusually high number of simultaneous connection requests to different ports is identified as a portscan. Snort logs the following information:

```
[**] [100:1:1] spp_portscan: ➥
PORTSCAN DETECTED ➥
from 192.168.5.22 (THRESHOLD ➥
4 connections exceeded in ➥
0 seconds) [**]
10/05-19:40:49.540435
```

You can often foil an intrusion detection system by telling Nmap to slow down. This disguises the packets in normal traffic. You could also use techniques such as FIN, Xmas, or Null scanning.

Top Secret

These extremely stealthy techniques do not rely on requesting or opening a connection; instead, they send a single frame to the target system. FIN, Xmas, and Null scans differ only with respect to the TCP flags they set (Figure 4), none of which should occur in normal network traffic. None of these techniques uses the SYN flag. The response packet or the lack of a response packet tells Nmap all it needs to know about the availability of the target system. Standards-compliant TCP stacks with closed ports respond by sending an RST packet to reset the connection.

If a network service is listening on the target port, the scanned system will not typically be able to find a connection to match the requesting packet. Unfortunately, the RFCs do not give clear instructions on what to do with packets of this kind. As a consequence, different TCP stack implementations react differently in this situation; Listing 3 shows that a Linux machine would react to this scenario by ignoring unexpected packets.

Windows systems handle this problem in a different way. The response of a Windows system is to reset the connection, without disambiguating open and closed ports. This allows Nmap to identify the operating system, as this behavior is only typical of Windows and a few other exotic TCP stacks.

If the target does not respond, Nmap reports the protocol as closed or filtered (see Table 2); firewalls tend to discard packets of this kind without comment. If you need more detail, you might like to additionally enable version detection (also known as version probe): this technique does not attempt to be stealthy

Table 1: Scanning Techniques

Scanning Technique	Syntax	Use
TCP SYN	-sS	Stealth scan
TCP connect()	-sT	Scan without root privileges
FIN	-sF	Stealth scan
Xmas	-sX	Stealth scan
Null	-sN	Stealth scan
Ping	-sP	Identify live hosts
Version Detection	-sV	Identify services
UDP	-sU	Find UDP services
IP Protocol	-sO	Discover supported protocols
ACK	-sA	Identify firewalls
Window	-sW	Advanced ACK scan
RPC	-sR	Information on RPC services
List	-sL	Dummy for test purposes
Idle	-sI	Scan via third party
FTP Bounce	-b	Historic

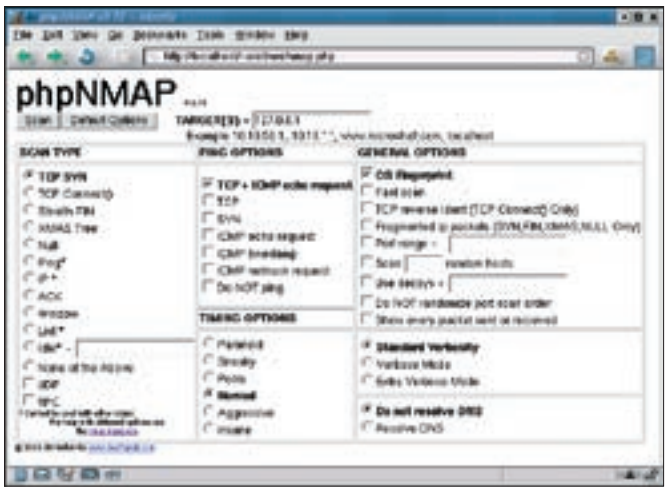

Figure 2: Nmap even has a web front-end, although the web interface does not offer the full range of Nmap capabilities.

and uses more aggressive port identification techniques.

Drilling Down

Version detection does not search for open ports, but investigates the service listening on a port. Potential candidates are identified prior to this by normal port scanning methods (Connect or SYN scan). Version detection opens a normal connection and communicates with the listening service. This typically creates a logfile entry. Nmap stores the results as *nmap-service-probes* in its database. Version 3.93 includes 2895 service signatures.

Nmap can apply the techniques used to fingerprint individual services to complete systems. OS fingerprinting, that is the technique of remotely identifying operating systems is one of Nmap's advanced features, and Nmap is the undisputed master in this field.

Searching for Hosts

Nmap not only supports administrators in their search for systems and services on a single machine; if needed, it can scan whole subnets for machines. The Ping scan is a useful tool in this scenario. As the name suggests, the *ping* command generates an ICMP Echo Request that prompts an ICMP Echo Reply from a live system.

As the ICMP protocol does not use any ports that Nmap could investigate, the scan needs only two frames per host, and that makes it very quick. On the downside, if you do not receive a response, there is no way of knowing if the host really is down, or if a packet filter has simply squashed the ICMP messages.

Ping scans are best used to prepare the way for more intensive investigations. A Ping scan gives the user a quick way of checking which systems react, making it easier to focus more time-consuming port scans on live targets.

Dry Run

The List scan gives users the option of checking their settings for potential sources of error before the event. The scan simply tells the user which systems Nmap would investigate, and how, without actually doing any scanning.

Pen-testers still need to take care, however: remember that Nmap will try to resolve the IP address by performing a reverse lookup, so make sure you disable this telltale behavior.

The technique leverages subtle differences in TCP stack behavior. A database records specific features as fingerprints; the *nmap-os-fingerprints* with Nmap 3.93 has no less than 1707 entries. Comparing the results with this list exposes the protocol stack and the operating system. The scan is amazingly stealthy. It does without connecting to specific applications and only uses about 30 simple frames.

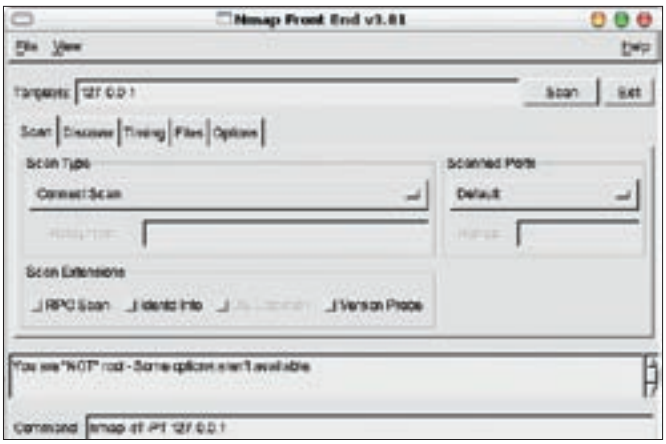

Figure 3: Without root privileges, Nmap is restricted to a small selection of techniques for identifying open ports across a network.

OS fingerprinting starts with a simple port scan, and then performs eight small tests, all of which send specially crafted packets to the target system. Some of these packets would never occur in normal network traffic, and that makes them easy prey for IDS systems. The target system does not notice that it is being scanned. At the same time, the TCP timestamp option reads the system uptime.

If Nmap is unable to identify the target system, it presents the data to the users, and if the user happens to know which OS this is, he or she can publish the signature via the Submit page at [10]. This is how Nmap users can help improve Nmap's identification feature.

No Filters

In scenarios where Nmap is unable to distinguish between a filtered port and an open port (Table 2), ACK scans can help clarify the situation. ACK scanning is a simple and stealthy technique, and although it may not help to identify a port as open, it will detect a firewall. To do this, Nmap transmits a single ACK packet to a target host, which should respond by returning a RST. If the response is a ICMP Destination Unreach-

Listing 1: Connect Scan

```
01 Port closed:
02 192.168.5.22 -> 192.168.5.10 TCP 60319 > 80 [SYN]
03 192.168.5.10 -> 192.168.5.22 TCP 80 > 60319 [RST, ACK]
04
05 Port open:
06 192.168.5.22 -> 192.168.5.10 TCP 60320 > 80 [SYN]
07 192.168.5.10 -> 192.168.5.22 TCP 80 > 60320 [SYN, ACK]
08 192.168.5.22 -> 192.168.5.10 TCP 60320 > 80 [ACK]
09 192.168.5.22 -> 192.168.5.10 TCP 60320 > 80 [RST, ACK]
```

Listing 2: TCP-SYN-Scans

```
01 Port closed:
02 192.168.5.22 -> 192.168.5.10 TCP 56522 > 80 [SYN]
03 192.168.5.10 -> 192.168.5.22 TCP 80 > 56522 [RST, ACK]
04
05 Port open:
06 192.168.5.22 -> 192.168.5.10 TCP 60420 > 80 [SYN]
07 192.168.5.10 -> 192.168.5.22 TCP 80 > 60420 [SYN, ACK]
08 192.168.5.22 -> 192.168.5.10 TCP 60420 > 80 [RST]
```

able packet rather than a reset, you can typically assume that the response was blocked by a firewall, that is, the port is filtered.

Nmap also uses Window scans, a variant on the ACK scans theme, to identify open ports. The scan again starts by transmitting an ACK packet, but it additionally evaluates the window size set by the target system in its response. RST packets with a window size of zero tell the scanner that the port is open.

As the number of operating systems that will respond to this technique is quite small and will continue to dwindle in the future, the Window scan can additionally provide more insight into the host platform. And this additional insight makes this technique a good choice if you need more information on hardened systems.

Beyond TCP

Besides TCP, Nmap can also perform UDP scanning. There are very few options for this technique, as the protocol does not use SYN or other flags. This makes UDP scanning very simple: the target responds to requests for closed ports with an ICMP Port Unreachable packet, whereas open ports will typically respond with data of some kind. If there is no response, Nmap classifies the port as open or filtered (Table 2). In case of doubt, version detection might help.

As many systems reduce ICMP error messages to just a few packets per second, Nmap detects this behavior and shifts down a gear or two. This means that UDP scans can take a while to complete.

Table 2: Port States

State	Explanation
Open	Communication via this port is possible without any restrictions.
Filtered	The port is probably being blocked by a firewall. If SYN or Connect scans discover open and filtered ports, the administrator may have made the error of implementing an insecure DROP configuration.
Unfiltered	ACK or Window scans discover unfiltered ports. Communication with the port in question is possible, but other scanning techniques are required to gather more information.
Closed	The port is either correctly blocked by a firewall, or there is no listener at this address. In both cases, communication is impossible.

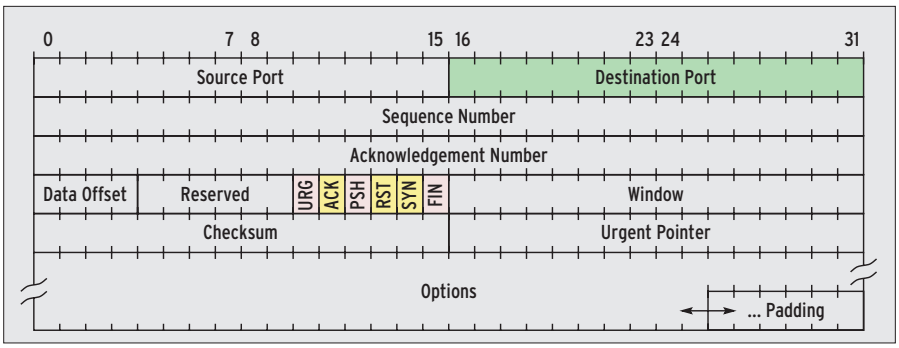

Figure 4: Nmap uses the TCP header fields to discover the details of the target systems. Besides the normal ACK, RST, and SYN (yellow) flags, some techniques use unexpected combinations of other flags (pink).

Low Level

The IP Protocol scan has nothing to do with identifying specific ports; it simply tells the user which Layer 4 protocols the target supports. With a Linux client, Nmap will typically discover ICMP (Internet Control Message Protocol), IGMP (Internet Group Multicast Protocol), TCP, UDP, and IPv6 (for an Ipv6-over-IPv4 tunnel.) It tests all protocol numbers between 1 and 255 and waits for responses.

This information allows Nmap to guess what kind of system it is scanning. Rough fingerprinting is possible based on this knowledge, as only routers and special servers will use the Virtual Router Redundancy Protocol (VRRP) or the free CARP alternative, for example.

Calling All Stations

RPC scans can identify services such as NFS and NIS, which are based on this technology, along with the ports they use. RPC scans only make sense as a follow-up to other scans that provide information on open ports. RPC scanning automatically enables version detection.

To help discover hidden RPC services, this technique additionally uses the special RPC instruction *PROC = 0*. This instruction does not actually ask the service to do anything, but it does force the service to reveal that it exists. Non-RPC services fail to identify the instruction and do not respond. As RPC scans rely on interacting with an application, they are non-stealthy, although they can reveal many details of the systems under investigation with a bit of luck.

Red Herrings

In some situations, even the most stealthy of Nmap scanning techniques leave too many traces. A tester might need detailed information that only an easily identifiable scan can reveal. The trick in this case is to feed IDS systems and administrators decoys to keep them busy.

When Nmap uses a decoy, it simulates various other scans from a preset number of spoofed IP addresses to disguise the genuine scanning process. Although this makes it more likely for the scans to be detected, the noise from the spoofed attacks makes it more difficult to identify the genuine offender.

From the system administrator's point of view, decoy scanning is a good test of

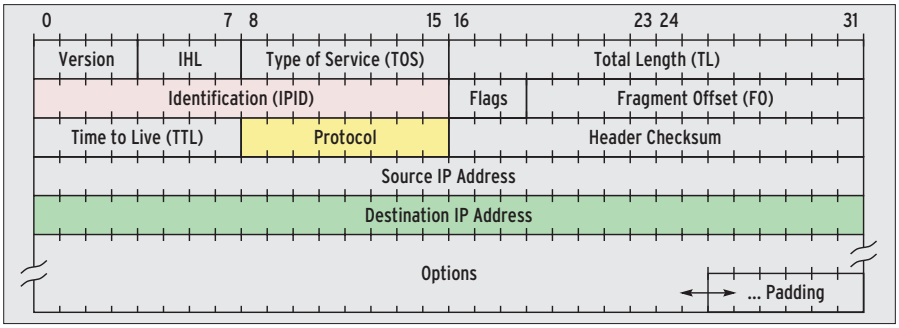

Figure 5: The IP header contains the IPID field (pink), which specifies the number of contiguous fragments.

Listing 3: FIN, Xmas, and Null

```
01 FIN scan, Port closed:
02 192.168.5.22 -> 192.168.5.10 TCP 56485 > 80 [FIN]
03 192.168.5.10 -> 192.168.5.22 TCP 80 > 56485 [RST, ACK]
04
05 FIN Scan, Port open:
06 192.168.5.22 -> 192.168.5.10 TCP 43406 > 80 [FIN]
07 192.168.5.22 -> 192.168.5.10 TCP 43407 > 80 [FIN]
08
09 Xmas Scan, Port closed:
10 192.168.5.22 -> 192.168.5.10 TCP 49499 > 80 [FIN, PSH, URG]
11 192.168.5.10 -> 192.168.5.22 TCP 80 > 49499 [RST, ACK]
12
13 Xmas Scan, Port open:
14 192.168.5.22 -> 192.168.5.10 TCP 47109 > 80 [FIN, PSH, URG]
15 192.168.5.22 -> 192.168.5.10 TCP 47110 > 80 [FIN, PSH, URG]
16
17 Null Scan, Port closed:
18 192.168.5.22 -> 192.168.5.10 TCP 50508 > 80 []
19 192.168.5.10 -> 192.168.5.22 TCP 80 > 50508 [RST, ACK]
20
21 Null Scan, Port open:
22 192.168.5.22 -> 192.168.5.10 TCP 55971 > 80 []
23 192.168.5.22 -> 192.168.5.10 TCP 55972 > 80 []
```

host, firewall, and Intrusion Detection System performance. By default, Nmap can simulate up to 128 hosts at the same time. If you have a powerful machine with a good network connection, you might like to increase the value for the *MAX_DECOYS* constant in *nmap.h* and rebuild Nmap.

Zombies

Idle scans are performed by the scanning machine misusing an uninvolved third party. An idle scan removes the need for the scanning machine to exchange packets with the target at any time in the process. Nmap handles the required IP spoofing autonomously. The following conditions must apply for this technique to work:

- The zombie host (the uninvolved third party, also referred to as a proxy) should have little to no network load to avoid modifying the IPID (the identification field in the IP header, see Figure 5).
- The IPID of the zombie must be predictable. A suitable system will increase the IPID by one for each new packet.

Nmap quickly identifies suitable zombie systems. The program sends six SYN/ACK packets to the zombie host and checks for linear incrementation of the IPIDs in the RST packets in the response. If this does not check out, the scan quits, telling the user: *Idlescan is unable to obtain meaningful results from proxy 192.168.5.99 (192.168.5.99). I'm sorry it didn't work out. QUITTING!*.

If the IPIDs increase in a predictable way, Nmap repeats the process four times, using packets with the source address of the system to be investigated in the process. The zombie then sends reset packets to the target host rather than to the scanner. To obtain results, Nmap sends a further SYN/ACK packet to the zombie, using its own source IP this time. Nmap will only attempt to perform an Idle (or Blind) scan if the IPID in the resulting reset has increased by a count of five.

Blind Vision

Nmap uses a similar approach to investigate the target: it sends SYN packets to the target, using the zombie's address as the source. The scan target will respond to requests to closed ports with a RST, which the zombie ignores. In contrast to this, an open port will attempt to move to the next stage of the handshake by sending a SYN/ACK packet. The zombie does not know anything about the connection, and thus responds with a reset, increasing the IPID at the same time – the scanning host can then query the zombie for the counter value (Figure 6).

To speed up this process, Nmap intelligently assumes that most ports will be closed. It starts with 30 randomly selected TCP ports, and sends a SYN packet to each of them. If the IPID increases, Nmap now knows how many ports in the scanning range are open. In phase two, Nmap reduces the number of ports, and keeps on restricting the range until it can identify the port numbers.

Examples

The examples in the following are designed to show that Nmap is not just a tool for attackers, but also a useful aid to administrators. Before you decide to experiment with these techniques, you should be aware that protocol trickery is not completely risk-free: mission-critical systems can react in unforeseen ways. Taking down an important system is a sure-fire way of identifying a critical vulnerability, but in this case, the test does you more harm than good. Just imagine a scan knocking out the TCP stack on your Internet telephone system, for example, and taking down your landlines.

For routine checks, you might like to add the IP addresses of your critical systems to a text file, and then specify the *--excludefile* option with this file to scan the whole network but without endangering your key systems.

Nmap has a wide range of logging features for documenting and comparing scans. The *-oA* option enables three useful formats, and prepares the way for manual or automated evaluation of the scan data. NDiff [11] is useful for comparing data from various scans.

Licensing Issues

Organizations with geographically distributed branch offices may not have administrative staff at each branch. Nmap can give you a clearer picture of what is going on at your branch offices. This is not only a good idea security-wise; it also makes sense to check what pro-

grams are running on your machines to avoid licensing issues.

A simple Ping scan can give you an overview of the machines; more complex techniques provide information on the patch status, and help you identify rogue hosts:

```
nmap -vv -sS -O -T Polite -n ➥
-oA remote
192.168.6.0/24
```

This tells Nmap to use a SYN scan *-sS* to investigate the Class C subnet 192.168.6.0 and collect information about operating systems (OS detection *-O*). It does not make sense to resolve IP addresses to hostnames here; *-n* disables the reverse lookup. To avoid overloading the WAN connection, *-T Polite* switches to a less aggressive timing variant. *-oA remote* gives you more verbose logging. Nmap has three standard logging formats: an easily readable *remote.nmap*; *remote.gnmap*, which lends itself to investigation using Grep; and an XML file, *remote.xml*.

Outbreak

Worm outbreaks are a common occurrence in Windows-based networks, and Spyware tools such as Backorifice are still encountered in the wild. Nmap helps creative administrators to set up a "poor man's IDS" by searching for ports used by common malware programs:

```
nmap -vv -sS -n --excludefile ➥
exceptions -p wormports -oA ➥
infected 192.168.5.0/24
```

The preceding command tells Nmap to be as verbose as possible, *-vv*. A list of known, potentially dangerous ports follows the *-p* parameter. Worms use these ports to follow commands, load dangerous code, or propagate. The hosts in the 192.168.5.0/24 network are scanned using a SYN scan *-sS*, but without name resolution *-n*. The hosts in the exceptions file are omitted from the scan. The results of the scan end up in files titled *infected.nmap*, *infected.gnmap* and *infected.xml*.

Nmap's version detection *-sV* feature does a good job of detecting malware listening on standard ports. Nmap references its internal database to identify the services in question. However, the process is time-consuming and it can cause heavy network load, which makes it impractical in many cases.

Patchwork

Worms, viruses and other kinds of malware leverage unpatched, publicly known vulnerabilities. SQL Slammer is an example of a worm that gained notoriety due to its widespread success, but OpenSSL also got the wrong kind of publicity thanks to the Scalper worm. To test a system for a legacy vulnerability of this kind, follow the procedure outlined in the previous example. You might prefer to perform the version detection and OS detection phases separately to avoid network load:

```
nmap -vv -sS -A -n ➥
--excludefile exceptions -oA ➥
version 192.168.5.0/24
```

Again, the basic underlying technique is a SYN scan *-sS*, and the options are more or less unchanged. The version detection and OS fingerprinting techniques, combined as the *-A* option are new. The results give you more insight into missing patches or forgotten updates. But be aware that some security patches do not increment the version number – Nmap is no substitute for genuine patch management.

Versatile and Powerful

Nmap handles an enormous range of sophisticated and stealthy scanning techniques. The fact that the tool can be misused by attackers should not deter administrators from using it. Doing so gives you comprehensive, and in some cases, astonishingly detailed information about your network. ■

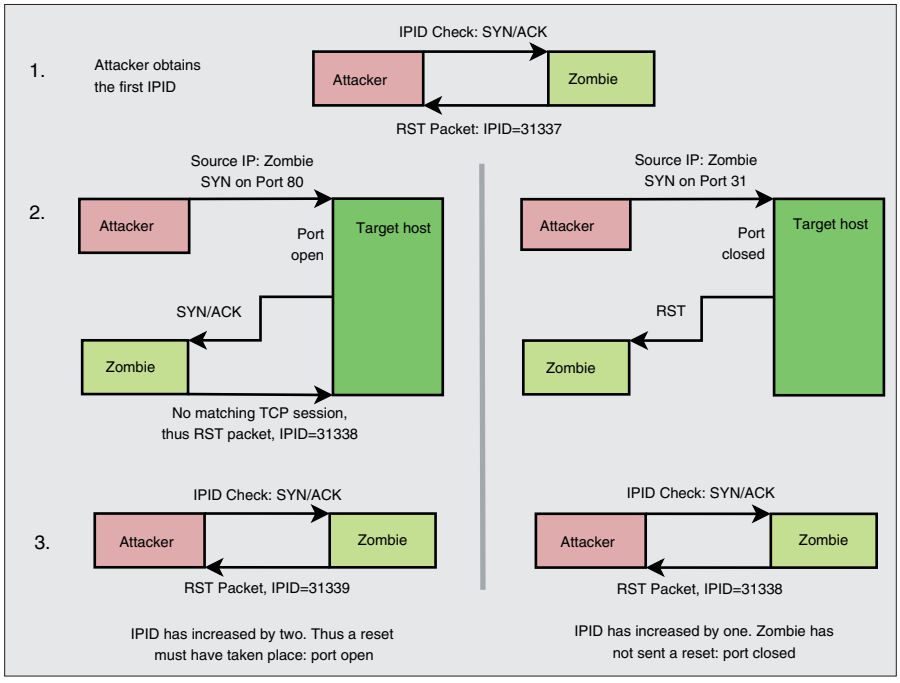

Figure 6: Idle scanning is a crafty technique that involves an uninvolved third party known as a zombie or proxy. The packets sent to the target host originate with the zombie. The attacker draws conclusions by checking for changes in the IPID (the identification field in the IP header.)

INFO

[1] Nmap: *http://www.insecure.org/nmap/*

[2] Fyodor, "The Art of Port Scanning", Phrack 51: *http://www.phrack.org/phrack/51/P51-11*

[3] Strobe: *http://ftp.surfnet.nl/security/coast/scanners/strobe/*

[4] Pscan: *http://www.packetstormsecurity.com/UNIX/scanners/pscan.c*

[5] Nmapwin: *http://nmapwin.sourceforge.net*

[6] PHP-Nmap: *http://phpnmap.sourceforge.net*

[7] Nmap-related projects: *http://www.insecure.org/nmap/nmap_relatedprojects.html*

[8] Snort IDS: *http://www.snort.org*

[9] Prelude IDS: *http://www.prelude-ids.org*

[10] Identifying services and ports: *http://www.insecure.org/cgi-bin/nmap-submit.cgi*

[11] NDiff: *http://www.vinecorp.com/ndiff/*

[12] OpenSSL: *http://www.openssl.org*