NMAP - A Stealth Port Scanner

Andrew J. Bennieston

http://www.nmap-tutorial.com

Contents

1 Introduction

Nmap is a free, open-source port scanner available for both UNIX and Windows. It has an optional graphical front-end, NmapFE, and supports a wide variety of scan types, each one with different benefits and drawbacks.

This article describes some of these scan types, explaining their relative benefits and just how they actually work. It also offers tips about which types of scan would be best against which types of host.

The article assumes you have Nmap installed (or that you know how to install it. Instructions are available on the Nmap website, `http://www.insecure.org/nmap/install/inst-source.html`), and that you have the required privileges to run the scans detailed (many scans require root or Administrator privileges).

A frequently asked questions section has been added since the first version of this article, and this is included as the last section in this version. This is a fully revised and updated version of this tutorial, re-typed and converted to a TeX format, allowing more output formats to be utilised. At the time of writing, the latest Nmap version was 4.11.

2 Disclaimer

This information is provided to assist users of Nmap in scanning their own networks, or networks for which they have been given permission to scan, in order to determine the security of such networks. it is not intended to assist with scanning remote sites with the intention of breaking into or exploiting services on those sites, or for imformation gathering purposes beyond those allowed by law. I hereby disclaim any responsibility for actions taken based upon the information in this article, and urge all who seek information towards a destructive end to reconsider their life, and do something constructive instead.

3 Basic Scan Types [-sT, -sS]

The two basic scan types used most in Nmap are TCP connect() scanning [-sT] and SYN scanning (also known as half-open, or stealth scanning) [-sS].

These two types are explained in detail below.

3.1 TCP connect() Scan [-sT]

These scans are so called because UNIX sockets programming uses a system call named `connect()` to begin a TCP connection to a remote site. If `connect()` succeeds, a connection was made. If it fails, the connection could not be made (the remote system is offline, the port is closed, or some other error occurred along the way). This allows a basic type of port scan, which attempts to connect

to every port in turn, and notes whether or not the connection succeeded. Once the scan is completed, ports to which a connection could be established are listed as *open*, the rest are said to be closed.

This method of scanning is very effective, and provides a clear picture of the ports you can and cannot access. If a connect() scan lists a port as open, you can definitely connect to it - that is what the scanning computer just did! There is, however, a major drawback to this kind of scan; it is very easy to detect on the system being scanned. If a firewall or intrusion detection system is running on the victim, attempts to connect() to every port on the system will almost always trigger a warning. Indeed, with modern firewalls, an attempt to connect to a single port which has been blocked or has not been specifically "opened" will usually result in the connection attempt being logged. Additionally, most servers will log connections and their source IP, so it would be easy to detect the source of a TCP connect() scan.

For this reason, the TCP Stealth Scan was developed.

3.2 SYN Stealth Scan [-sS]

I'll begin this section with an overview of the TCP connection process. Those familiar with TCP/IP can skip the first few paragraphs.

When a TCP connection is made between two systems, a process known as a "three way handshake" occurs. This involves the exchange of three packets, and synchronises the systems with each other (necessary for the error correction built into TCP. Refer to a good TCP/IP book for more details.

The system initiating the connection sends a packet to the system it wants to connect to. TCP packets have a header section with a *flags* field. Flags tell the receiving end something about the type of packet, and thus what the correct response is.

Here, I will talk about only four of the possible flags. These are SYN (Synchronise), ACK (Acknowledge), FIN (Finished) and RST (Reset). SYN packets include a TCP sequence number, which lets the remote system know what sequence numbers to expect in subsequent communication. ACK acknowledges receipt of a packet or set of packets, FIN is sent when a communication is finished, requesting that the connection be closed, and RST is sent when the connection is to be reset (closed immediately).

To initiate a TCP connection, the initiating system sends a SYN packet to the destination, which will respond with a SYN of its own, and an ACK, acknowledging the receipt of the first packet (these are combined into a single SYN/ACK packet). The first system then sends an ACK packet to acknowledge receipt of the SYN/ACK, and data transfer can then begin.

SYN or Stealth scanning makes use of this procedure by sending a SYN packet and looking at the response. If SYN/ACK is sent back, the port is open and the remote end is trying to open a TCP connection. The scanner then sends an RST

to tear down the connection before it can be established fully; often preventing the connection attempt appearing in application logs. If the port is closed, an RST will be sent. If it is filtered, the SYN packet will have been dropped and no response will be sent. In this way, Nmap can detect three port states - open, closed and filtered. Filtered ports may require further probing since they could be subject to firewall rules which render them open to some IPs or conditions, and closed to others.

Modern firewalls and Intrusion Detection Systems can detect SYN scans, but in combination with other features of Nmap, it is possible to create a virtually undetectable SYN scan by altering timing and other options (explained later).

4 FIN, Null and Xmas Tree Scans [-sF, -sN, -sX]

With the multitude of modern firewalls and IDS' now looking out for SYN scans, these three scan types may be useful to varying degrees. Each scan type refers to the flags set in the TCP header. The idea behind these type of scans is that a closed port should respond with an RST upon receiving packets, whereas an open port should just drop them (it's listening for packets with SYN set). This way, you never make even part of a connection, and never send a SYN packet; which is what most IDS' look out for.

The FIN scan sends a packet with only the FIN flag set, the Xmas Tree scan sets the FIN, URG and PUSH flags (see a good TCP/IP book for more details) and the Null scan sends a packet with no flags switched on.

These scan types will work against any system where the TCP/IP implementation follows RFC 793. Microsoft Windows does not follow the RFC, and will ignore these packets even on closed ports. This technicality allows you to detect an MS Windows system by running SYN along with one of these scans. If the SYN scan shows open ports, and the FIN/NUL/XMAS does not, chances are you're looking at a Windows box (though OS Fingerprinting is a much more reliable way of determining the OS running on a target!)

The sample below shows a SYN scan and a FIN scan, performed against a Linux system. The results are, predictably, the same, but the FIN scan is less likely to show up in a logging system.

```
1  [chaos]# nmap -sS 127.0.0.1
2
3  Starting Nmap 4.01 at 2006-07-06 17:23 BST
4  Interesting ports on chaos (127.0.0.1):
5  (The 1668 ports scanned but not shown below are in state:
6          closed)
7  PORT     STATE SERVICE
8  21/tcp   open  ftp
```

```
22/tcp   open  ssh
631/tcp  open  ipp
6000/tcp open  X11

Nmap finished: 1 IP address (1 host up) scanned in 0.207
        seconds
[chaos]# nmap -sF 127.0.0.1

Starting Nmap 4.01 at 2006-07-06 17:23 BST
Interesting ports on chaos (127.0.0.1):
(The 1668 ports scanned but not shown below are in state:
        closed)
PORT     STATE         SERVICE
21/tcp   open|filtered ftp
22/tcp   open|filtered ssh
631/tcp  open|filtered ipp
6000/tcp open|filtered X11

Nmap finished: 1 IP address (1 host up) scanned in 1.284
        seconds
```

5 Ping Scan [-sP]

This scan type lists the hosts within the specified range that responded to a ping. It allows you to detect which computers are online, rather than which ports are open. Four methods exist within Nmap for ping sweeping.

The first method sends an ICMP ECHO REQUEST (ping request) packet to the destination system. If an ICMP ECHO REPLY is received, the system is up, and ICMP packets are not blocked. If there is no response to the ICMP ping, Nmap will try a "TCP Ping", to determine whether ICMP is blocked, or if the host is really not online.

A TCP Ping sends either a SYN or an ACK packet to any port (80 is the default) on the remote system. If RST, or a SYN/ACK, is returned, then the remote system is online. If the remote system does not respond, either it is offline, or the chosen port is filtered, and thus not responding to anything.

When you run an Nmap ping scan as root, the default is to use the ICMP and ACK methods. Non-root users will use the connect() method, which attempts to connect to a machine, waiting for a response, and tearing down the connection as soon as it has been established (similar to the SYN/ACK method for root users, but this one establishes a full TCP connection!)

The ICMP scan type can be disabled by setting -P0 (that is, zero, not uppercase o).

6 UDP Scan [-sU]

Scanning for open UDP ports is done with the -sU option. With this scan type, Nmap sends 0-byte UDP packets to each target port on the victim. Receipt of an ICMP Port Unreachable message signifies the port is closed, otherwise it is assumed open.

One major problem with this technique is that, when a firewall blocks outgoing ICMP Port Unreachable messages, the port will appear open. These false-positives are hard to distinguish from real open ports.

Another disadvantage with UDP scanning is the speed at which it can be performed. Most operating systems limit the number of ICMP Port Unreachable messages which can be generated in a certain time period, thus slowing the speed of a UDP scan. Nmap adjusts its scan speed accordingly to avoid flooding a network with useless packets. An interesting point to note here is that Microsoft do not limit the Port Unreachable error generation frequency, and thus it is easy to scan a Windows machine's 65,535 UDP Ports in very little time!!

UDP Scanning is not usually useful for most types of attack, but it can reveal information about services or trojans which rely on UDP, for example SNMP, NFS, the Back Orifice trojan backdoor and many other exploitable services.

Most modern services utilise TCP, and thus UDP scanning is not usually included in a pre-attack information gathering exercise unless a TCP scan or other sources indicate that it would be worth the time taken to perform a UDP scan.

7 IP Protocol Scans [-sO]

The IP Protocol Scans attempt to determine the IP protocols supported on a target. Nmap sends a raw IP packet without any additional protocol header (see a good TCP/IP book for information about IP packets), to each protocol on the target machine. Receipt of an ICMP Protocol Unreachable message tells us the protocol is not in use, otherwise it is assumed open. Not all hosts send ICMP Protocol Unreachable messages. These may include firewalls, AIX, HP-UX and Digital UNIX). These machines will report all protocols open.

This scan type also falls victim to the ICMP limiting rate described in the UDP scans section, however since only 256 protocols are possible (8-bit field for IP protocol in the IP header) it should not take too long.

Results of an -sO on my Linux workstation are included below.

```
1  [chaos]# nmap -sO 127.0.0.1
2
3  Starting Nmap 4.01 at 2006-07-14 12:56 BST
4  Interesting protocols on chaos(127.0.0.1):
```

```
5   (The 251 protocols scanned but not shown below are
6           in state: closed)
7   PROTOCOL STATE         SERVICE
8   1        open          icmp
9   2        open|filtered igmp
10  6        open          tcp
11  17       open          udp
12  255      open|filtered unknown
13
14  Nmap finished: 1 IP address (1 host up) scanned in
15          1.259 seconds
```

8 Idle Scanning [-sI]

Idle scanning is an advanced, highly stealthed technique, where no packets are sent to the target which can be identified to originate from the scanning machine. A zombie host (and optionally port) must be specified for this scan type. The zombie host must satisfy certain criteria essential to the workings of this scan.

This scan type works by exploiting "predictable IP fragmentation ID" sequence generation on the zombie host, to determine open ports on the target. The scan checks the IPID on the zombie, then spoofs a connection request to the target machine, making it appear to come from the zombie. If the target port is open, a SYN/ACK session acknowledgement will be sent from the target machine back to the zombie, which will RST the connection since it has no record of having opened such a connection. If the port on the target is closed, an RST will be sent to the zombie, and no further packets will be sent. The attacker then checks the IPID on the zombie again. If it has incremented by 2 (or changed by two steps in its sequence), this corresponds to the packet received from the target, plus the RST from the zombie, which equates to an open port on the target. If the IPID has changed by one step, an RST was received from the target and no further packets were sent.

Using this mechanism, it is possible to scan every port on a target, whilst making it appear that the zombie was the one doing the scanning. Of course, the spoofed connection attempts will likely be logged, so the target system will have the zombie IP address, and the zombie system's logs are likely to contain the attacker's IP address, so it is still possible, after acquiring logs through legal channels, to determine the attacker, but this method makes it much more difficult to do so than if the packets were sent directly from the attacker. In addition, some IDS and firewall software makes attempts to detect spoofed packets based on the network they arrive from. As long as the zombie host and the attacker are both "out on the Internet", or on the same network as each other, relative to the target, techniques to identify spoofed packets are not likely to succeed.

This scan type requires certain things of the zombie. The IPID sequence generation must be predictable (single-step increments, for example). The host must also have low traffic so that it is unlikely for other packets to hit the zombie whilst Nmap is carrying out its scan (as these will artificially inflate the IPID number!). Cheap routers or MS Windows boxes make good zombie hosts. Most operating systems use randomised sequence numbers (see the OS Fingerprinting section for details on how to check a target's sequence generation type).

The idle scan can also be used to determine IP trust based relationships between hosts (e.g. a firewall may allow a certain host to connect to port x, but not other hosts). This scan type can help to determine which hosts have access to such a system.

For more information about this scan type, read `http://www.insecure.org/nmap/idlescan.html`

9 Version Detection [-sV]

Version Detection collects information about the specific service running on an open port, including the product name and version number. This information can be critical in determining an entry point for an attack. The -sV option enables version detection, and the -A option enables both OS fingerprinting and version detection, as well as any other advanced features which may be added in future releases.

Version detection is based on a complex series of probes, detailed in the Version Detection paper at `http://www.insecure.org/nmap/vscan/`

10 ACK Scan [-sA]

Usually used to map firewall rulesets and distinguish between stateful and stateless firewalls, this scan type sends ACK packets to a host. If an RST comes back, the port is classified "unfiltered" (that is, it was allowed to send its RST through whatever firewall was in place). If nothing comes back, the port is said to be "filtered". That is, the firewall prevented the RST coming back from the port. This scan type can help determine if a firewall is stateless (just blocks incoming SYN packets) or stateful (tracks connections and also blocks unsolicited ACK packets).

Note that an ACK scan will never show ports in the "open" state, and so it should be used in conjunction with another scan type to gain more information about firewalls or packet filters between yourself and the victim.

11 Window Scan, RPC Scan, List Scan [-sW, -sR, -sL]

The TCP Window scan is similar to the ACK scan but can sometimes detect open ports as well as filtered/unfiltered ports. This is due to anomalies in TCP Window size reporting by some operating systems (see the Nmap manual for a list, or the nmap-hackers mailing list for the full list of susceptible OS').

RPC Scans can be used in conjunction with other scan types to try to determine if an open TCP or UDP port is an RPC service, and if so, which program, and version numbers are running on it. Decoys are not supported with RPC scans (see section on Timing and Hiding Scans, below).

List scanning simply prints a list of IPs and names (DNS resolution will be used unless the -n option is passed to Nmap) without actually pinging or scanning the hosts.

12 Timing and Hiding Scans

12.1 Timing

Nmap adjusts its timings automatically depending on network speed and response times of the victim. However, you may want more control over the timing in order to create a more stealthy scan, or to get the scan over and done with quicker.

The main timing option is set through the -T parameter. There are six predefined timing policies which can be specified by name or number (starting with 0, corresponding to Paranoid timing). The timings are Paranoid, Sneaky, Polite, Normal, Aggressive and Insane.

A -T Paranoid (or -T0) scan will wait (generally) at least 5 minutes between each packet sent. This makes it almost impossible for a firewall to detect a port scan in progress (since the scan takes so long it would most likely be attributed to random network traffic). Such a scan will still show up in logs, but it will be so spread out that most analysis tools or humans will miss it completely.

A -T Insane (or -T5) scan will map a host in very little time, provided you are on a very fast network or don't mind losing some information along the way.

Timings for individual aspects of a scan can also be set using the –host_timeout, –max_rtt_timeout, –min_rtt_timeout, –initial_rtt_timeout, –max_parallelism, –min_parallelism, and –scan_delay options. See the Nmap manual for details.

12.2 Decoys

The -D option allows you to specify Decoys. This option makes it look like those decoys are scanning the target network. It does not hide your own IP, but it makes your IP one of a torrent of others supposedly scanning the victim at the

same time. This not only makes the scan look more scary, but reduces the chance of you being traced from your scan (difficult to tell which system is the "real" source).

12.3 FTP Bounce

The FTP protocol (RFC 959) specified support for a "proxy" ftp, which allowed a connection to an FTP server to send data to anywhere on the internet. This tends not to work with modern ftpds, in which it is an option usually disabled in the configuration. If a server with this feature is used by Nmap, it can be used to try to connect to ports on your victim, thus determining their state.

This scan method allows for some degree of anonymity, although the FTP server may log connections and commands sent to it.

12.4 Turning Off Ping

The -P0 (that's a zero) option allows you to switch off ICMP pings. The -PT option switches on TCP Pings, you can specify a port after the -PT option to be the port to use for the TCP ping.

Disabling pings has two advantages: First, it adds extra stealth if you're running one of the more stealthy attacks, and secondly it allows Nmap to scan hosts which don't reply to pings (ordinarily, Nmap would report those hosts as being "down" and not scan them).

In conjunction with -PT, you can use -PS to send SYN packets instead of ACK packets for your TCP Ping.

The -PU option (with optional port list after) sends UDP packets for your "ping". This may be best to send to suspected-closed ports rather than open ones, since open UDP ports tend not to respond to zero-length UDP packets.

Other ping types are -PE (Standard ICMP Echo Request), -PP (ICMP Timestamp Request), -PM (Netmask Request) and -PB (default, uses both ICMP Echo Request and TCP ping, with ACK packets)

12.5 Fragmenting

The -f option splits the IP packet into tiny fragments when used with -sS, -sF, -sX or -sN. This makes it more difficult for a firewall or packet filter to determine the packet type. Note that many modern packet filters and firewalls (including iptables) feature optional defragmenters for such fragmented packets, and will thus reassemble the packet to check its type before sending it on. Less complex firewalls will not be able to cope with fragmented packets this small and will most likely let the OS reassemble them and send them to the port they were intended to reach. Using this option could crash some less stable software and hardware since packet sizes get pretty small with this option!

12.6 Idle Scanning

See the section on -sI for information about idle scans.

13 OS Fingerprinting

The -O option turns on Nmap's OS fingerprinting system. Used alongside the -v verbosity options, you can gain information about the remote operating system and about its TCP Sequenmce Number generation (useful for planning Idle scans).

An article on OS detection is available at `http://www.insecure.org/nmap/nmap-fingerprinting-article.html`

14 Outputting Logs

Logging in Nmap can be provided by the -oN, -oX or -oG options. Each one is followed by the name of the logfile. -oN outputs a human readable log, -oX outputs an XML log and -oG outputs a grepable log. The -oA option outputs in all 3 formats, and -oS outputs in a format I'm sure none of you would ever want to use (try it; you'll see what I mean!)

The –append-output option appends scan results to the output files you specified instead of overwriting their contents.

15 Other Nmap Options

15.1 IPv6

The -6 option enables IPv6 in Nmap (provided your OS has IPv6 support). Currently only TCP connect, and TCP connect ping scan are supported. For other scantypes, see `http://nmap6.sourceforge.net`

15.2 Verbose Mode

Highly recommended, -v

Use -v twice for more verbosity. The option -d can also be used (once or twice) to generate more verbose output.

15.3 Resuming

Scans cancelled with Ctrl+C can be resumed with the –resume ¡logfilename¿ option. The logfile must be a Normal or Grepable logfile (-oN or -oG).

15.4 Reading Targets From A File

-iL ¡inputfilename¿ reads targets from inputfilename rather than from the command-line.

The file should contain a hostlist or list of network expressions separated by spaces, tabs or newlines. Using a hyphen as inputfile makes Nmap read from standard input.

15.5 Fast Scan

The -F option scans only those ports listed in the nmap_services file (or the protocols file if the scan type is -sO). This is far faster than scanning all 65,535 ports!!

15.6 Time-To-Live

The -ttl ¡value¿ option sets the IPv4 packets time-to-live. The usefulness of this is in mapping paths through networks and determining ACL's on firewalls (setting the ttl to one past the packet filter can help to determine information about the filtering rules themselves). Repeated Nmap scans to a single port using differing ttl values will emulate a traceroute style network path map (Try it, its great fun for a while, until you get bored and realise traceroute does it all for you automatically!).

16 Typical Scanning Session

First, we'll sweep the network with a simple Ping scan to determine which hosts are online.

```
1   [chaos]# nmap -sP 10.0.0.0/24
2
3   Starting Nmap 4.01 ( http://www.insecure.org/nmap/ ) at
4           2006-07-14 14:19 BST
5   Host 10.0.0.1 appears to be up.
6   MAC Address: 00:09:5B:29:FD:96 (Netgear)
7   Host 10.0.0.2 appears to be up.
8   MAC Address: 00:0F:B5:96:38:5D (Netgear)
9   Host 10.0.0.4 appears to be up.
10  Host 10.0.0.5 appears to be up.
11  MAC Address: 00:14:2A:B1:1E:2E (Elitegroup Computer System Co.)
12  Nmap finished: 256 IP addresses (4 hosts up) scanned in 5.399 seconds
```

Now we're going to take a look at 10.0.0.1 and 10.0.0.2, both listed as Netgear in the ping sweep. These IPs are good criteria for routers (in fact I know that 10.0.0.1 is a router and 10.0.0.2 is a wireless access point, since it's my network, but lets see what Nmap makes of it...)

We'll scan 10.0.0.1 using a SYN scan [-sS] and -A to enable OS fingerprinting and version detection.

```
1  [chaos]# nmap -sS -A 10.0.0.1
2
3  Starting Nmap 4.01 ( http://www.insecure.org/nmap/ ) at
4          2006-07-14 14:23 BST
5  Insufficient responses for TCP sequencing (0),
6          OS detection may be less accurate
7  Interesting ports on 10.0.0.1:
8  (The 1671 ports scanned but not shown below are in state:
9          closed)
10 PORT   STATE SERVICE    VERSION
11 80/tcp open  tcpwrapped
12 MAC Address: 00:09:5B:29:FD:96 (Netgear)
13 Device type: WAP
14 Running: Compaq embedded, Netgear embedded
15 OS details: WAP: Compaq iPAQ Connection Point or
16         Netgear MR814
17
18 Nmap finished: 1 IP address (1 host up) scanned in
19         3.533 seconds
```

The only open port is 80/tcp - in this case, the web admin interface for the router. OS fingerprinting guessed it was a Netgear Wireless Access Point - in fact this is a Netgear (wired) ADSL router. As it said, though, there were insufficient responses for TCP sequencing to accurately detect the OS.

Now we'll do the same for 10.0.0.2...

```
1  [chaos]# nmap -sS -A 10.0.0.2
2
3  Starting Nmap 4.01 ( http://www.insecure.org/nmap/ )
4          at 2006-07-14 14:26 BST
5  Interesting ports on 10.0.0.2:
6  (The 1671 ports scanned but not shown below are in state:
7          closed)
8  PORT   STATE SERVICE VERSION
9  80/tcp open  http    Boa HTTPd 0.94.11
10 MAC Address: 00:0F:B5:96:38:5D (Netgear)
```

```
11  Device type: general purpose
12  Running: Linux 2.4.X|2.5.X
13  OS details: Linux 2.4.0 - 2.5.20
14  Uptime 14.141 days (since Fri Jun 30 11:03:05 2006)
15
16  Nmap finished: 1 IP address (1 host up) scanned in 9.636
17          seconds
```

Interestingly, the OS detection here listed Linux, and the version detection was able to detect the httpd running. The accuracy of this is uncertain, this is a Netgear home wireless access point, so it could be running some embedded Linux!

Now we'll move on to 10.0.0.4 and 10.0.0.5, these are likely to be normal computers running on the network...

```
1   [chaos]# nmap -sS -P0 -A -v 10.0.0.4
2
3   Starting Nmap 4.01 ( http://www.insecure.org/nmap/ ) at
4           2006-07-14 14:31 BST
5   DNS resolution of 1 IPs took 0.10s. Mode:
6           Async [#: 2, OK: 0, NX: 1, DR: 0, SF: 0, TR: 1, CN: 0]
7   Initiating SYN Stealth Scan against 10.0.0.4 [1672 ports] at 14:31
8   Discovered open port 21/tcp on 10.0.0.4
9   Discovered open port 22/tcp on 10.0.0.4
10  Discovered open port 631/tcp on 10.0.0.4
11  Discovered open port 6000/tcp on 10.0.0.4
12  The SYN Stealth Scan took 0.16s to scan 1672 total ports.
13  Initiating service scan against 4 services on 10.0.0.4 at 14:31
14  The service scan took 6.01s to scan 4 services on 1 host.
15  For OSScan assuming port 21 is open, 1 is closed, and neither are
16          firewalled
17  Host 10.0.0.4 appears to be up ... good.
18  Interesting ports on 10.0.0.4:
19  (The 1668 ports scanned but not shown below are in state: closed)
20  PORT     STATE SERVICE VERSION
21  21/tcp   open  ftp     vsftpd 2.0.3
22  22/tcp   open  ssh     OpenSSH 4.2 (protocol 1.99)
23  631/tcp  open  ipp     CUPS 1.1
24  6000/tcp open  X11      (access denied)
25  Device type: general purpose
26  Running: Linux 2.4.X|2.5.X|2.6.X
27  OS details: Linux 2.4.0 - 2.5.20, Linux 2.5.25 - 2.6.8 or
28          Gentoo 1.2 Linux 2.4.19 rc1-rc7
```

```
29  TCP Sequence Prediction: Class=random positive increments
30                           Difficulty=4732564 (Good luck!)
31  IPID Sequence Generation: All zeros
32  Service Info: OS: Unix
33
34  Nmap finished: 1 IP address (1 host up) scanned in 8.333 seconds
35                 Raw packets sent: 1687 (74.7KB) | Rcvd: 3382 (143KB)
```

From this, we can deduce that 10.0.0.4 is a Linux system (in fact, the one I'm typing this tutorial on!) running a 2.4 to 2.6 kernel (Actually, Slackware Linux 10.2 on a 2.6.19.9 kernel) with open ports 21/tcp, 22/tcp, 631/tcp and 6000/tcp. All but 6000 have version information listed. The scan found the IPID sequence to be all zeros, which makes it useless for idle scanning, and the TCP Sequence prediction as random positive integers. The -v option is needed to get Nmap to print the IPID information out!

Now, onto 10.0.0.5...

```
1   [chaos]# nmap -sS -P0 -A -v 10.0.0.5
2
3   Starting Nmap 4.01 ( http://www.insecure.org/nmap/ )
4           at 2006-07-14 14:35 BST
5   Initiating ARP Ping Scan against 10.0.0.5 [1 port] at 14:35
6   The ARP Ping Scan took 0.01s to scan 1 total hosts.
7   DNS resolution of 1 IPs took 0.02s. Mode: Async
8           [#: 2, OK: 0, NX: 1, DR: 0, SF: 0, TR: 1, CN: 0]
9   Initiating SYN Stealth Scan against 10.0.0.5 [1672 ports] at 14:35
10  The SYN Stealth Scan took 35.72s to scan 1672 total ports.
11  Warning:  OS detection will be MUCH less reliable because we did
12          not find at least 1 open and 1 closed TCP port
13  Host 10.0.0.5 appears to be up ... good.
14  All 1672 scanned ports on 10.0.0.5 are: filtered
15  MAC Address: 00:14:2A:B1:1E:2E (Elitegroup Computer System Co.)
16  Too many fingerprints match this host to give specific OS details
17  TCP/IP fingerprint:
18  SInfo(V=4.01%P=i686-pc-linux-gnu%D=7/14%Tm=44B79DC6%O=-1%C=-1%M=00142A)
19  T5(Resp=N)
20  T6(Resp=N)
21  T7(Resp=N)
22  PU(Resp=N)
23
24  Nmap finished: 1 IP address (1 host up) scanned in 43.855 seconds
25                 Raw packets sent: 3369 (150KB) | Rcvd: 1 (42B)
```

No open ports, and Nmap couldn't detect the OS. This suggests that it is a firewalled or otherwise protected system, with no services running (and yet it responded to ping sweeps).

We now have rather more information about this network than we did when we started, and can guess at several other things based on these results. Using that information, and the more advanced Nmap scans, we can obtain further scan results which will help to plan an attack, or to fix weaknesses, in this network.

17 Frequently Asked Questions

This section was added as an extra to the original tutorial as it became popular and some questions were asked about particular aspects of an nmap scan. I'll use this part of the tutorial to merge some of those into the main tutorial itself.

17.1 I tried a scan and it appeared in firewall logs or alerts. What else can I do to help hide my scan?

This question assumes you used a scan command along the lines of:

```
1   nmap -sS -P0 -p 1-140 -O -D xxx.xxx.xxx.xxx,
2           xxx.xxx.xxx.xxx, xxx.xxx.xxx.xxx -sV xxx.xx.xxx.xxx
```

Note: Each xxx corresponds to an octet of the IP address/addresses. This is instructing NMAP to run a Stealth scan (-sS) without pinging (-P0) on ports 1 to 140 (-p 1-140), to use OS Detection (-O) and to use Decoys (-D). The three comma-separated IPs are the decoy IPs to use. It also specifies to use version scanning (-sV) which attempts to determine precisely which program is running on a port.

Now, heres the analysis of this command: A stealth scan (-sS) is often picked up by most firewalls and IDS systems nowdays. It was originally designed to prevent logging of a scan in the logs for whatever server is running on the port the scanner connects to. In other words, if the scan connects to port 80 to test if its open, Apache (or whatever other webserver they may be using) will log the connection in its logfiles.

The -sS scan option doesn't make a full TCP connect (which can be achieved with the -sT option, or by not running as root) but resets the connection before it can be fully established. As such, most servers will not log the connection, but an IDS or firewall will recognise this behaviour (in repeated cases) as typical of a port scan. This will mean that the scan shows up in firewall or IDS logs and alerts. There are few ways around this, to be honest. Most firewall/IDS software nowdays is quite good at detecting these things; particularly if its running on the same host as the victim (the system you are scanning).

Note also, that decoys will not prevent your IP showing entirely; it just lists the others as well. A particularly well designed IDS may even be able to figure out which is the real source of the scans.

Where speed of scan isn't essential, the -P0 option is a good idea. Nmap gains timing information from pinging the host, and can often complete its scans faster with this information, but the ping packets will be sent to the victim from your IP, and any IDS worth its CPU cycles will pick up on the pattern of a few pings followed by connects to a variety of ports. -P0 also allows scanning of hosts which do not respond to pings (i.e. if ICMP is blocked by a firewall or by in-kernel settings).

I mentioned timing in the above paragraph. You can use the -T timing option to slow the scan down. The slower a scan is, the less likely it is to be detected by an IDS. There are bound to be occasional random connects occurring, people type an IP in wrong or try to connect and their computer crashes half way through the connect. These things happen, and unless an IDS is configured extremely strictly, they generally aren't reported (at least, not in the main alert logs, they may be logged if logging of all traffic is enabled, but typically these kind of logs are only checked if theres evidence of something going on). Setting the timing to -T 0 or -T 1 (Paranoid or Sneaky) should help avoid detection. As mentioned in my main tutorial, you can also set timing options for each aspect of a scan,

> Timings for individual aspects of a scan can also be set using the –host_timeout, –max_rtt_timeout, –min_rtt_timeout, –initial_rtt_timeout, –max_parallelism, –min_parallelism, and –scan_delay options. See the Nmap manual for details.

The final note I will add to this answer is that use of the Idle scan method (-sI) means that not a single packet is sent to the victim from your IP (provided you also use the -P0 option to turn off pings). This is the ultimate in stealth as there is absolutely no way the victim can determine that your IP is responsible for the scan (short of obtaining log information from the host you used as part of your idle scan).

17.2 NMAP seems to have stopped, or my scan is taking a very long while. Why is this?

The timing options can make it take a very long time. I believe the -T Paranoid (-T 0)option waits up to 5 minutes between packets... now, for 65000 ports, thats 65000 x 5 = 325000 minutes = 225 days!!

-T Sneaky (-T 1) waits up to 15 seconds between scans, and is therefore more useful; but scans will still take a long while! You can use -v to get more verbose output, which will alert you as to the progress of the scan. Using -v twice makes the output even more verbose.

17.3 Will -sN -sX and -sF work against any host, or just Windows hosts?

-sN -sX and -sF scans will work against any host, but Windows computers do not respond correctly to them, so scanning a Windows machine with these scans results in all ports appearing closed. Scanning a *nix or other system should work just fine, though. As I said in the main tutorial, -sX -sF and -sN are commonly used to determine if you're scanning a Windows host or not, without using the -O fingerprinting option.

The Nmap manual page should help to determine which scans work alongside which options, and on which target systems they are most effective.

17.4 How do I find a dummy host for the Idle Scan (-sI)?

You simply have to scan for hosts using sequential IPID sequences, these are (often) suitable for use as a dummy host for the -sI Idle Scan.

17.5 What does "Host seems down. If it is really up, but blocking our ping probes, try -P0" mean?

When Nmap starts, it tries to ping the host to check that it is online. Nmap also gains timing information from this ping. If the remote host, or a system on the path between you and the remote host, is blocking pings, this ping will not be replied to, and Nmap will not start scanning. Using the -P0 option, you can turn off ping-on-start and have Nmap try to scan anyway.

17.6 Where can I find NmapFE?

NmapFE is a graphical front-end for Nmap.

NmapFE for UNIX/Linux is included in the Nmap source. NmapFE for OSX is available at `http://faktory.org/m/software/nmap/` NmapFE for Windows is under development as part of NmapFE++, a new frontend for Linux, OSX and Windows. Information is available at `http://www.insecure.org/nmap/SoC/NmapFE.html`

18 About This Document