NMAP - A Stealth Port Scanner

Andrew J. Bennieston

Table of Contents

1 Introduction

Nmap is a free port scanner available for both Unix and Windows. It has an optional graphical front-end, NmapFE and supports a variety of scan types, eacn one with different benefits.

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

This article assumes that you have Nmap installed on your computer (or know how to install it) and that you have the required privileges to run the scans detailed (i.e. root on Unix systems, Administrator on Windows systems).

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 information 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 below.

3.1 TCP connect() Scans [-sT]

These scans are so called because Unix sockets programming uses a system call called connect() to begin a connection to a remote site. If connect() succeeds, a connection was made, if connect() fails, the connection could not be made. This allows a basic type of port scan to attempt to connect() to every port on a system. If it succeeds, it makes a note of the port number, closes the connection, and moves on. Once it has completed its scan, the ports it could connect to are listed as "open" ports. This method of scanning is effective: it lets you know quite clearly whether a port is available to connect to from another application, because a connection attempt was made from the port scanner. However, this type of scan 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, (amongst others) the TCP Stealth Scan was developed.

3.2 SYN Stealth Scanning [-sS]

For those of you who know the basics of TCP/IP, the first part of this section will not tell you much, however it is included so that people who do not know about such things can grasp the concept of a SYN scan.

When a TCP connection is opened between two systems, a "three-way handshake" occurs in order to synchronize the connections with each other. This is part of TCP's built in error checking (refer to a good TCP/IP book for more details).

Basically, the system initiating the connection sends a TCP packet to the system it wants to connect to. A TCP packet has a header section, with a field for "flags" to be set. These flags tell the system receiving the packet what type of packet it is, and thus what they should do in response to it.

I will here introduce four of the possible flags which may be set in a TCP header. These four are SYN, ACK, FIN and RST.

SYN is short for synchronise, and tells the system it is being sent to the next TCP Sequence Number to expect for packets on that connection.

ACK is short for acknowledge, and tells the system it is being sent to that the system it came from successfully received some data (again, see a good TCP book for more details).

FIN is short for finish, and is used to tell a recipient system that the sender wants to end the connection.

RST is short for Reset, and abruptly closes the connection. No further packets are sent following receipt of an RST.

Armed with this knowledge, we can now discuss the first few moments of a TCP connections life. System A wants to open a connection to port 80 on System B. To do this, System A sends a TCP packet with the SYN flag set (and a sequence number) to System B. System B receives this packet and, if the port is open and accepting conections, responds with a TCP packet with the ACK flag set (to acknowledge receipt of the first SYN) and the SYN flag set (with its own starting sequence number). Once System A receives System B's SYN|CK packet, System A sends an ACK in response, and then data transmission can begin.

Now that the intro to TCP is over, I can explain the SYN scan type. With this scan type, the scanner does not use the TCP connect() system call, which would result in a whole connection being opened, and thus potentially logged. Instead, it generates a packet with the SYN header set (and a suitable sequence number) and sends that off to the destination. The scanner then listens for a response. If the destination port is open, a packet will come back from the target system, from that port, with both SYN and ACK set. If the port is closed, the response will be a packet with RST set, or nothing (if the port is in a state known as "filtered"). If the response is a SYN|ACK, indicating the port is open, the scanner sends an RST to forcibly close the connection before it has completed opening. This prevents it being logged in a server's connection log.

Of course, modern firewalls can still 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 types of scan work against systems where the TCP/IP implementation follows the RFC (see RFC 793). Of course, Microsoft, being Microsoft, chose to do things another way! These three scans will not work against a Windows box, although this technicality does allow you to determine that the victim is in fact a Windows system, if one of these scans doesn't show any ports open, whereas a SYN scan shows some open. Other systems which function the same way as Windows with respect to -sF, -sX and -sN are Cisco, BSDI, HP/UK, MVS and IRIX. All of those send RST's from open ports as well as closed ports.

Included below are the results of an nmap -sS 127.0.0.1 and an nmap -sF 127.0.0.1, to prove that, against a Linux system, the two scan types return identical results, yet the SYN has a greater chance of being detected.

Code:

```
bash-2.05b# nmap -sS 127.0.0.1
Starting nmap 3.30 ( http://www.insecure.org/nmap/ ) at 2003-08-27 12:54 BST
Interesting ports on localhost (127.0.0.1): (The 1643 ports scanned but not shown
below are in state: closed)

Port State Service

6000/tcp open X11

Nmap run completed -- 1 IP address (1 host up) scanned in 1.986 seconds

bash-2.05b# nmap -sF 127.0.0.1
Starting nmap 3.30 ( http://www.insecure.org/nmap/ ) at 2003-08-27 12:53 BST

Interesting ports on localhost (127.0.0.1): (The 1643 ports scanned but not shown
below are in state: closed)

Port State Service

6000/tcp open X11

Nmap run completed -- 1 IP address (1 host up) scanned in 3.135 seconds
```

5 Ping Scanning [-sP]

This is useful for listing hosts on a network which are "up", rather than detecting specific open ports on them.

NMAP has four possible methods for detecting if a host is "up". The first is to send an ICMP ECHO REQUEST (ping request) to the system. If an ICMP ECHO REPLY (Ping reply) is received then the system is up, otherwise the system is down or not responding to Echo Requests.

To determine if Echo Requests are being blocked, Nmap can also use "TCP Ping" techniques. These involve sending a TCP packet with either ACK or SYN set to a port (default 80, but any will do) on a machine, and waiting to see if an RST or SYN|ACK comes back. If neither comes back, there is a high probability the machine is down (although it could be that the port chosen is in state "filtered" and is thus not responding to anything).

If you run an Nmap ping scan as root, it defaults to using the ICMP and ACK methods. Non-root users will use the connect() method (which simply tries to connect to a machine, and sees if it gets a response, similar to the SYN method for root users, but making a full connection and tearing it down as soon as it is created).

You can disable the ICMP method by setting the -P option to -P0 (yes, thats a zero, not an uppercase o).

6 UDP Scans [-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 athering 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.

Code:

```
bash-2.05b# nmap -sO 127.0.0.1

Starting nmap 3.30 ( http://www.insecure.org/nmap/ ) at 2003-08-27 14:20 BST
Interesting protocols on localhost (127.0.0.1): (The 251 protocols scanned but not
shown below are in state: closed)

Protocol State Name

1 open icmp
```

```
2 open igmp
6 open tcp
17 open udp
255 open unknown

Nmap run completed -- 1 IP address (1 host up) scanned in 3.807 seconds
```

8 Idle Scanning [-sI]

This method of scanning is an advanced and highly stealthed technique, where no packets are sent to the target from the scanning machine. You must specify a zombie host (and optionally a port) for this scan type. The host must also satisfy certain criteria.

This scan type works by expoliting "predictable IP fragmentation ID" sequence generation on the zombie host to determine open ports on the target. Basically this means it checks the TCP sequence ID on the zombie, makes the zombie try a port on the target, then checks the sequence ID on the zombie again. Depending on how many the sequence number changed by, Nmap can determine whether the port on the target was open.

This scan type requires a zombie with fairly predictable TCP sequence generation, for example a cheap or simple router. Most operating systems use randomised sequence numbers (see the OS Fingerprinting section for details on how to check a target's TCP sequence generation type). This scan type 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 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 mroe information about firewalls or packet filters between yourself and the victim.

10 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 if IPs and names (DNS resolution will be used unless the -n option is passed to Nmap) without actually pinging or scanning the hosts.

11 Timing And Hiding Scans

11.1 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, --ax_rtt_timeout, --min_rtt_timeout, --initial_rtt_timeout, --max_parallelism, --min_parallelism, and –scan_delay options. See the Nmap manual for details.

11.2 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).

11.3 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.

11.4 Turning Pings Off

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)

11.5 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!

11.6 Idle Scanning

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

12 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

13 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.

14 Other Nmap Options

14.1 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

14.2 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.

14.3 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).

1*4.4 ResumingReading 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 stdin.

14.5 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 all65,535 ports!!

14.6 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 realize traceroute does it all for you automatically!).

15 Typical Scanning Session

First, we'll check the network to see which hosts are up:

Code:

```
Starting nmap 3.30 ( http://www.insecure.org/nmap/ ) at
2003-08-27 15:38 BST

Host 192.168.0.1 appears to be up.
Host 192.168.0.2 appears to be up.
Host chaos.bytekill.net (192.168.0.3) appears to be up.
Host 192.168.0.4 appears to be up.
Host 192.168.0.255 appears to be up.
Nmap run completed -- 256 IP addresses (5 hosts up) scanned in 9.733 seconds
```

Now, we want to take a look at 192.168.0.1. Its likely its a router of some kind (in fact I know its a router, this is my network, It's a Netgear DG814, but we'll see what Nmap makes of it anyhow...)

We'll run a SYN scan with -O set and -F to stop it taking too long, but to give it enough ports to determine an OS fingerprint!

Code:

```
bash-2.05b# nmap -sS -F -O 192.168.0.1

Starting nmap 3.30 ( http://www.insecure.org/nmap/ ) at 2003-08-27 15:41 BST

Insufficient responses for TCP sequencing (0), OS detection may be less accurate

Interesting ports on 192.168.0.1: (The 1196 ports scanned but not shown below are
in state: closed)

Port State Service

80/tcp open http Device type: WAP

Running: Compaq embedded, Netgear embedded

OS details: WAP: Compaq iPAQ Connection Point or Netgear MR814

Nmap run completed -- 1 IP address (1 host up) scanned in 6.537 seconds
```

It happens to have made quite a good guess at that! It doesn't matter that there weren't enough responses for TCP sequencing... we're not going to run an Idle scan, or attempt to expolit the routers TCP sequencing. That scan confirmed our suspicions that 192.168.0.1 was a router (the so-called "non-routable" network blocks (10.x.x.x, 192.168.x.x and the other one I can never remember) are private networks, there is almost always a router on these and it likes to configure itself with the first IP available, so its usually a fair guess that 192.168.0.1 and 10.0.0.1 are routers of some kind!)

The 192.168.0.255 is, of course, the broadcast address for this block, so we'll ignore that. Our next victim will, therefore, be one of 192.168.0.2, 192.168.0.3 or 192.168.0.4. Lets take a look at 192.168.0.2:

Code:

```
bash-2.05b# nmap -sS -P0 -O -v 192.168.0.2

Starting nmap 3.30 ( http://www.insecure.org/nmap/ ) at 2003-08-27 15:50 BST

Host 192.168.0.2 appears to be up ... good.

Initiating SYN Stealth Scan against 192.168.0.2 at 15:50

Adding open port 139/tcp
Adding open port 1025/tcp
Adding open port 445/tcp
Adding open port 135/tcp

The SYN Stealth Scan took 1 second to scan 1644 ports.
For OSScan assuming that port 135 is open and port 1 is closed and neither are
firewalled

Interesting ports on 192.168.0.2:

(The 1640 ports scanned but not shown below are in state: closed)

Port State Service

135/tcp open loc-srv

139/tcp open netbios-ssn

445/tcp open microsoft-ds

1025/tcp open NFS-or-IIS

Device type: general purpose

Running: Microsoft Windows 95/98/ME|NT/2K/XP

OS details: Microsoft Windows Millennium Edition (Me), Win 2000 profressional or
Advanced Server, or WinXP

TCP Sequence Prediction: Class=random positive increments

Difficulty=9871 (Worthy challenge)

IPID Sequence Generation: Incremental

Nmap run completed -- 1 IP address (1 host up) scanned in 2.446 seconds
```

Nmap seems to think this is a Windows computer - its right! It also said the IPID Sequence Generation is Incremental... which is useful for an Idle scan. Lets now try an idle scan against 192.168.0.3:

Code:

```
bash-2.05b# nmap -sI 192.168.0.2 192.168.0.3

WARNING: Many people use -P0 w/Idlescan to prevent pings from their true IP. On
the other hand, timing info Nmap gains from pings can allow for faster, more
reliable scans.

Starting nmap 3.30 ( http://www.insecure.org/nmap/ ) at 2003-08-27 15:54 BST
```

```
Idlescan using zombie 192.168.0.2 (192.168.0.2:80); Class: Incremental

Interesting ports on chaos.bytekill.net (192.168.0.3):

(The 1643 ports scanned but not shown below are in state: closed)

Port State Service

6000/tcp open X11

Nmap run completed -- 1 IP address (1 host up) scanned in 13.423 seconds
```

Note that I didn't use -O on this... it won't work! Nmap needs the responses from the probes to be able to do OS Fingerprinting, the responses go to the zombie in an Idle scan, and so we can't determine OS type. However, the fact that X11 is open would suggest a Unix of some kind (Its rare to find X11 open on a Windows box, although not entirely impossible).

Finally, let's take a look at 192.168.0.4. For this, we won't use -O... we'll try to make a guess at its OS based on the output of two scan types. First, a SYN scan using -F:

Code:

```
bash-2.05b# nmap -sS -F 192.168.0.4

Starting nmap 3.30 ( http://www.insecure.org/nmap/ ) at 2003-08-27 15:57 BST

Interesting ports on 192.168.0.4:

(The 1193 ports scanned but not shown below are in state: closed)

Port State Service

135/tcp open loc-srv

139/tcp open netbios-ssn

445/tcp open microsoft-ds

1025/tcp open NFS-or-IIS

Nmap run completed -- 1 IP address (1 host up) scanned in 0.731 seconds
```

This looks like a Windows, but to check, we'll run one of the three scan types Windows always returns RST's for, -sF, -sX or -sN. My personal favourite is -sX:

Code:

```
bash-2.05b# nmap -sX -F 192.168.0.4

Starting nmap 3.30 ( http://www.insecure.org/nmap/ ) at 2003-08-27 15:58 BST

All 1197 scanned ports on 192.168.0.4 are: closed

Nmap run completed -- 1 IP address (1 host up) scanned in 0.741 seconds
```

As I thought, it returns all ports closed, so its fairly likely this is a Windows. We could do a scan with -O of course, which is a lot more reliable than guessing based on this type of scan, but we're moving on here... we don't have time for things like that. Next we want to look for something exploitable. The Linux box seems only to have X open.. we could look for exploits for that, or we could hope that one of the Windows

boxes has a trojan lurking somewhere in the higher port numbers.

I'd scan with -sS -p 1024-65534 and -sU -p 1024-65534, but I know for a fact that neither have trojans running on them since they're checked regularly by myself, so it would be a waste of time and would add nothing of use to this article anyway.

16 Closing

Well, there it is... a load of information about Nmap, including some scan outputs and the reasoning behind the next steps, based on the data retrieved from them.

For more information, check out the Nmap website at http://www.insecure.org/nmap , or the various links to more specific information interspersed throughout this document.

Complete listings of the Nmap options (I didn't cover them all here, just the useful ones... there are a few more to tailor a scan, and Nmap itself, to your exacting requirements) are available in the Nmap manual.

17 Questions & Answers

This section was added as an extra to the original post as the thread 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. The posts that these questions are adapted (and generalised) from will be removed from the thread, along with the posts I've made giving a solution or answer, as the information is now in 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 is taken from a post by clipse This question assumes you used a scan command along the lines of:

Code:

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

> **Quote:**
>
> ```
> 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?

This question is taken from a post by clipse 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!

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

This question is taken from a post by clipse -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)?

This question is taken from a post by FiNaLBeTa 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.

This document is Copyright © 2003-2006 Andrew J. Bennieston.You may freely print or distribute this document in original form only.

_____

Andrew J. Bennieston

http://ajb.coldblue.net/