NAME

nmap − Network exploration tool and security scanner

SYNOPSIS

nmap [Scan Type(s)] [Options] <host or net #1 ... [#N]>

DESCRIPTION

Nmap is designed to allow system administrators and curious individuals to scan large networks to determine which hosts are up and what services they are offering. *nmap* supports a large number of scanning techniques such as: UDP, TCP connect(), TCP SYN (half open), ftp proxy (bounce attack), ICMP (ping sweep), FIN, ACK sweep, Xmas Tree, SYN sweep, IP Protocol, and Null scan. See the *Scan Types* section for more details. nmap also offers a number of advanced features such as remote OS detection via TCP/IP fingerprinting, stealth scanning, dynamic delay and retransmission calculations, parallel scanning, detection of down hosts via parallel pings, decoy scanning, port filtering detection, direct (non-portmapper) RPC scanning, fragmentation scanning, and flexible target and port specification.

Significant effort has been put into decent nmap performance for non-root users. Unfortunately, many critical kernel interfaces (such as raw sockets) require root privileges. nmap should be run as root whenever possible (not setuid root, of course).

The result of running nmap is usually a list of interesting ports on the machine(s) being scanned (if any). Nmap always gives the port's "well known" service name (if any), number, state, and protocol. The state is either "open", "closed" "filtered", or "unfiltered". Open means that the target machine will accept() connections on that port. Closed ports are not listening for connections (they have no application associated with them). Filtered means that a firewall, filter, or other network obstacle is covering the port and preventing nmap from determining whether the port is open. Unfiltered means that the port is known by nmap to be closed and no firewall/filter seems to be interfering with nmap's attempts to determine this. Unfiltered ports are the common case and are only shown when most of the scanned ports are in the filtered state. In some cases, Nmap cannot distinguish between filtered ports and those that are either open or closed. For example, a port that does not respond to a FIN Scan could be either open or filtered. In these cases, Nmap lists ports as "open|filtered" or "closed|filtered".

Depending on options used, nmap may also report the following characteristics of the remote host: OS in use, TCP sequentiality, usernames running the programs which have bound to each port, the DNS name, whether the host is a smurf address, and a few others.

OPTIONS

Options that make sense together can generally be combined. Some options are specific to certain scan modes. *nmap* tries to catch and warn the user about psychotic or unsupported option combinations.

If you are impatient, you can skip to the *examples* section at the end, which demonstrates common usage. You can also run **nmap -h** for a quick reference page listing all the options.

SCAN TYPES

−sS TCP SYN scan: This technique is often referred to as "half-open" scanning, because you don't open a full TCP connection. You send a SYN packet, as if you are going to open a real connection and you wait for a response. A SYN|ACK indicates the port is listening. A RST is indicative of a non−listener. If a SYN|ACK is received, a RST is immediately sent to tear down the connection (actually our OS kernel does this for us). The primary advantage to this scanning technique is that fewer sites will log it. Unfortunately you need root privileges to build these custom SYN packets. This is the default scan type for privileged users.

−sT TCP connect() scan: This is the most basic form of TCP scanning. The connect() system call provided by your operating system is used to open a connection to every interesting port on the machine. If the port is listening, connect() will succeed, otherwise the port isn't reachable. One strong advantage to this technique is that you don't need any special privileges. Any user on most UNIX boxes is free to use this call.

This sort of scan is easily detectable as target host logs will show a bunch of connection and error messages for the services which accept() the connection just to have it immediately shutdown.

This is the default scan type for unprivileged users.

−sF −sX −sN
Stealth FIN, Xmas Tree, or Null scan modes: There are times when even SYN scanning isn't clandestine enough. Some firewalls and packet filters watch for SYNs to restricted ports, and programs like Synlogger and Courtney are available to detect these scans. These advanced scans, on the other hand, may be able to pass through unmolested.

The idea is that closed ports are required to reply to your probe packet with an RST, while open ports must ignore the packets in question (see RFC 793 pp 64). Filered ports also tend to drop probes without a response, so Nmap considers ports "open|filtered" when it fails to elicit any response. If you add version detection (-sV), it will try to verify whether the ports are actually open and change the state as appropriate. The FIN scan uses a bare (surprise) FIN packet as the probe, while the Xmas tree scan turns on the FIN, URG, and PUSH flags. The Null scan turns off all flags. Unfortunately Microsoft (like usual) decided to completely ignore the standard and do things their own way. Thus this scan type will not work against systems running Windows95/NT. On the positive side, this is a good way to distinguish between the two platforms. If the scan finds open ports, you know the machine is not a Windows box. If a -sF,-sX,or -sN scan shows all ports closed, yet a SYN (-sS) scan shows ports being opened, you are probably looking at a Windows box. This is less useful now that nmap has proper OS detection built in. There are also a few other systems that are broken in the same way Windows is. They include Cisco, BSDI, HP/UX, MVS, and IRIX. All of the above send resets from the open ports when they should just drop the packet.

−sP Ping scanning: Sometimes you only want to know which hosts on a network are up. Nmap can do this by sending ICMP echo request packets to every IP address on the networks you specify. Hosts that respond are up. Unfortunately, some sites such as microsoft.com block echo request packets. Thus nmap can also send a TCP ack packet to (by default) port 80. If we get an RST back, that machine is up. A third technique involves sending a SYN packet and waiting for a RST or a SYN/ACK. For non-root users, a connect() method is used.

By default (for root users), nmap uses both the ICMP and ACK techniques in parallel. You can change these with the **−P** options described later.

Note that pinging is done by default anyway, and only hosts that respond are scanned. Only use this option if you wish to ping sweep **without** doing any actual port scans.

−sV Version detection: After TCP and/or UDP ports are discovered using one of the other scan methods, version detection communicates with those ports to try and determine more about what is actually running. A file called nmap-service-probes is used to determine the best probes for detecting various services and the match strings to expect. Nmap tries to determine the service protocol (e.g. ftp, ssh, telnet, http), the application name (e.g. ISC Bind, Apache httpd, Solaris telnetd), the version number, and sometimes miscellaneous details like whether an X server is open to connections or the SSH protocol version). If Nmap was compiled with OpenSSL support, it will connect to SSL servers to deduce the service listening behind the encryption. When RPC services are discovered, the Nmap RPC grinder is used to determine the RPC program and version numbers. Some UDP ports are left in the "open|filtered" state after a UDP scan is unable to determine whether the port is open or filtered. Version detection will try to elicit a response from these ports (just as it does with open ports), and change the state to open if it succeeds. Note that the Nmap -A option also enables this feature. For a much more detailed description of Nmap service detection, read our paper at http://www.insecure.org/nmap/versionscan.html . There is a related --version_trace option which causes Nmap to print out extensive debugging info about what version scanning is doing (this is a subset of what you would get with --packet_trace).

−sU UDP scans: This method is used to determine which UDP (User Datagram Protocol, RFC 768) ports are open on a host. The technique is to send 0 byte UDP packets to each port on the target machine. If we receive an ICMP port unreachable message, then the port is closed. If a UDP response is received to the probe (unusual), the port is open. If we get no response at all, the state is "open|filtered", meaning that the port is either open or packet filters are blocking the

communication. Versions scan (-sV) can be used to help differentiate the truly open ports from the filtered ones.

Some people think UDP scanning is pointless. I usually remind them of the Solaris rpcbind hole. Rpcbind can be found hiding on an undocumented UDP port somewhere above 32770. So it doesn't matter that 111 is blocked by the firewall. But can you find which of the more than 30,000 high ports it is listening on? With a UDP scanner you can! There is also the cDc Back Orifice backdoor program which hides on a configurable UDP port on Windows machines. Not to mention the many commonly vulnerable services that utilize UDP such as snmp, tftp, NFS, etc.

Unfortunately UDP scanning is sometimes painfully slow since most hosts implement a suggestion in RFC 1812 (section 4.3.2.8) of limiting the ICMP error message rate. For example, the Linux kernel (in net/ipv4/icmp.h) limits destination unreachable message generation to 80 per 4 seconds, with a 1/4 second penalty if that is exceeded. Solaris has much more strict limits (about 2 messages per second) and thus takes even longer to scan. *nmap* detects this rate limiting and slows down accordingly, rather than flood the network with useless packets that will be ignored by the target machine.

As is typical, Microsoft ignored the suggestion of the RFC and does not seem to do any rate limiting at all on Win95 and NT machines. Thus we can scan all 65K ports of a Windows machine **very** quickly. Whoop!

−sO IP protocol scans: This method is used to determine which IP protocols are supported on a host. The technique is to send raw IP packets without any further protocol header to each specified protocol on the target machine. If we receive an ICMP protocol unreachable message, then the protocol is not in use. Otherwise we assume it is open. Note that some hosts (AIX, HP-UX, Digital UNIX) and firewalls may not send protocol unreachable messages. This causes all of the protocols to appear "open".

Because the implemented technique is very similar to UDP port scanning, ICMP rate limit might apply too. But the IP protocol field has only 8 bits, so at most 256 protocols can be probed which should be possible in reasonable time anyway.

−sI <zombie host[:probeport]>
Idlescan: This advanced scan method allows for a truly blind TCP port scan of the target (meaning no packets are sent to the target from your real IP address). Instead, a unique side-channel attack exploits predictable "IP fragmentation ID" sequence generation on the zombie host to glean information about the open ports on the target. IDS systems will display the scan as coming from the zombie machine you specify (which must be up and meet certain criteria). I wrote an informal paper about this technique at http://www.insecure.org/nmap/idlescan.html .

Besides being extraordinarily stealthy (due to its blind nature), this scan type permits mapping out IP-based trust relationships between machines. The port listing shows open ports *from the perspective of the zombie host.* So you can try scanning a target using various zombies that you think might be trusted (via router/packet filter rules). Obviously this is crucial information when prioritizing attack targets. Otherwise, you penetration testers might have to expend considerable resources "owning" an intermediate system, only to find out that its IP isn't even trusted by the target host/network you are ultimately after.

You can add a colon followed by a port number if you wish to probe a particular port on the zombie host for IPID changes. Otherwise Nmap will use the port it uses by default for "tcp pings".

−sA ACK scan: This advanced method is usually used to map out firewall rulesets. In particular, it can help determine whether a firewall is stateful or just a simple packet filter that blocks incoming SYN packets.

This scan type sends an ACK packet (with random looking acknowledgment/sequence numbers) to the ports specified. If a RST comes back, the ports is classified as "unfiltered". If nothing comes back (or if an ICMP unreachable is returned), the port is classified as "filtered". Note that *nmap* usually doesn't print "unfiltered" ports, so getting **no** ports shown in the output is usually a

sign that all the probes got through (and returned RSTs). This scan will obviously never show ports in the "open" state.

−sW Window scan: This advanced scan is very similar to the ACK scan, except that it can sometimes detect open ports as well as filtered/unfiltered due to an anomaly in the TCP window size reporting by some operating systems. Systems vulnerable to this include at least some versions of AIX, Amiga, BeOS, BSDI, Cray, Tru64 UNIX, DG/UX, OpenVMS, Digital UNIX, FreeBSD, HP-UX, OS/2, IRIX, MacOS, NetBSD, OpenBSD, OpenStep, QNX, Rhapsody, SunOS 4.X, Ultrix, VAX, and VxWorks. See the nmap-hackers mailing list archive for a full list.

−sR RPC scan. This method works in combination with the various port scan methods of Nmap. It takes all the TCP/UDP ports found open and then floods them with SunRPC program NULL commands in an attempt to determine whether they are RPC ports, and if so, what program and version number they serve up. Thus you can effectively obtain the same info as "rpcinfo -p" even if the target's portmapper is behind a firewall (or protected by TCP wrappers). Decoys do not currently work with RPC scan, at some point I may add decoy support for UDP RPC scans. This is automatically enabled as part of version scan (-sV) if you request that.

−sL List scan. This method simply generates and prints a list of IP addresses or hostnames without actually pinging or port scanning them. DNS name resolution will be performed unless you use -n.

−b <ftp relay host>

FTP bounce attack: An interesting "feature" of the ftp protocol (RFC 959) is support for "proxy" ftp connections. In other words, I should be able to connect from evil.com to the FTP server of target.com and request that the server send a file ANYWHERE on the Internet! Now this may have worked well in 1985 when the RFC was written. But in today's Internet, we can't have people hijacking ftp servers and requesting that data be spit out to arbitrary points on the Internet. As *Hobbit* wrote back in 1995, this protocol flaw "can be used to post virtually untraceable mail and news, hammer on servers at various sites, fill up disks, try to hop firewalls, and generally be annoying and hard to track down at the same time." What we will exploit this for is to (surprise, surprise) scan TCP ports from a "proxy" ftp server. Thus you could connect to an ftp server behind a firewall, and then scan ports that are more likely to be blocked (139 is a good one). If the ftp server allows reading from and writing to some directory (such as /incoming), you can send arbitrary data to ports that you do find open (nmap doesn't do this for you though).

The argument passed to the "b" option is the host you want to use as a proxy, in standard URL notation. The format is: *username:password@server:port.* Everything but *server* is optional. To determine what servers are vulnerable to this attack, you can see my article in *Phrack* 51. An updated version is available at the *nmap* URL (http://www.insecure.org/nmap).

GENERAL OPTIONS

None of these are required but some can be quite useful. Note that the -P options can now be combined -- you can increase your odds of penetrating strict firewalls by sending many probe types using different TCP ports/flags and ICMP codes.

−P0 Do not try to ping hosts at all before scanning them. This allows the scanning of networks that don't allow ICMP echo requests (or responses) through their firewall. microsoft.com is an example of such a network, and thus you should always use **−P0** or **−PS80** when portscanning microsoft.com. Note that "ping" in this context may involve more than the traditional ICMP echo request packet. Nmap supports many such probes, including arbitrary combinations of TCP, UDP, and ICMP probes. By default, Nmap sends an ICMP echo request and a TCP ACK packet to port 80.

−PA [portlist]

Use TCP ACK "ping" to determine what hosts are up. Instead of sending ICMP echo request packets and waiting for a response, we spew out TCP ACK packets throughout the target network (or to a single machine) and then wait for responses to trickle back. Hosts that are up should respond with a RST. This option preserves the efficiency of only scanning hosts that are up while

still allowing you to scan networks/hosts that block ping packets. For non root UNIX users, we use connect() and thus a SYN is actually being sent. To set the destination ports of the probe packets use -PA<port1>[,port2][...]. The default port is 80, since this port is often not filtered out. Note that this option now accepts multiple, comma-separated port numbers.

−PS [portlist]
This option uses SYN (connection request) packets instead of ACK packets for root users. Hosts that are up should respond with a RST (or, rarely, a SYN|ACK). You can set the destination ports in the same manner as −PA above.

−PR This option specifies a raw ethernet ARP ping. It cannot be used in combination with any of the other ping types. When the target machines are on the same network you are scanning from, this is the fastest and most reliable (because it goes below IP-level filters) ping method. Nmap sends an IPv4-to-Ethernet ARP request for each target IP, and watches for any ARP response. **−PU [portlist]** This option sends UDP probes to the specified hosts, expecting an ICMP port unreachable packet (or possibly a UDP response if the port is open) if the host is up. Since many UDP services won't reply to an empty packet, your best bet might be to send this to expected-closed ports rather than open ones.

−PE This option uses a true ping (ICMP echo request) packet. It finds hosts that are up and also looks for subnet-directed broadcast addresses on your network. These are IP addresses which are externally reachable and translate to a broadcast of incoming IP packets to a subnet of computers. These should be eliminated if found as they allow for numerous denial of service attacks (Smurf is the most common).

−PP Uses an ICMP timestamp request (type 13) packet to find listening hosts.

−PM Same as **−PE** and **−PP** except uses a netmask request (ICMP type 17).

−PB This is the default ping type. It uses both the ACK (**−PA**) and ICMP echo request (**−PE**) sweeps in parallel. This way you can get firewalls that filter either one (but not both). The TCP probe destination port can be set in the same manner as with −PA above. Note that this flag is now deprecated as pingtype flags can now be used in combination. So you should use both "PE" and "PA" (or rely on the default behavior) to achieve this same effect.

−O This option activates remote host identification via TCP/IP fingerprinting. In other words, it uses a bunch of techniques to detect subtleties in the underlying operating system network stack of the computers you are scanning. It uses this information to create a "fingerprint" which it compares with its database of known OS fingerprints (the nmap-os-fingerprints file) to decide what type of system you are scanning.

If Nmap is unable to guess the OS of a machine, and conditions are good (e.g. at least one open port), Nmap will provide a URL you can use to submit the fingerprint if you know (for sure) the OS running on the machine. By doing this you contribute to the pool of operating systems known to nmap and thus it will be more accurate for everyone. Note that if you leave an IP address on the form, the machine may be scanned when we add the fingerprint (to validate that it works).

The −O option also enables several other tests. One is the "Uptime" measurement, which uses the TCP timestamp option (RFC 1323) to guess when a machine was last rebooted. This is only reported for machines which provide this information.

Another test enabled by −O is TCP Sequence Predictability Classification. This is a measure that describes approximately how hard it is to establish a forged TCP connection against the remote host. This is useful for exploiting source-IP based trust relationships (rlogin, firewall filters, etc) or for hiding the source of an attack. The actual difficulty number is based on statistical sampling and may fluctuate. It is generally better to use the English classification such as "worthy challenge" or "trivial joke". This is only reported in normal output with -v.

When verbose mode (−v) is on with −O, IPID Sequence Generation is also reported. Most machines are in the "incremental" class, which means that they increment the "ID" field in the IP header for each packet they send. This makes them vulnerable to several advanced information

gathering and spoofing attacks.

−-osscan_limit
OS detection is far more effective if at least one open and one closed TCP port are found. Set this option and Nmap will not even try OS detection against hosts that do not meet this criteria. This can save substantial time, particularly on -P0 scans against many hosts. It only matters when OS detection is requested (-O or -A options).

−A This option enables _a_dditional _a_dvanced and _a_ggressive options. I haven't decided exactly which it stands for yet :). Presently this enables OS Detection (-O) and version scanning (-sV). More features may be added in the future. The point is to enable a comprehensive set of scan options without people having to remember a large set of flags. This option only enables features, and not timing options (such as -T4) or verbosity options (-v) that you might wan't as well.

−6 This options enables IPv6 support. All targets must be IPv6 if this option is used, and they can be specified via normal DNS name (AAAA record) or as a literal IP address such as 3ffe:501:4819:2000:210:f3ff:fe03:4d0 . Currently, connect() TCP scan and TCP connect() Ping scan are supported. If you need UDP or other scan types, have a look at http://nmap6.sourceforge.net/ .

−-send_eth
Asks Nmap to send packets at the raw ethernet (data link) layer rather than the higher IP (network) layer. By default, Nmap chooses the one which is generally best for the platform it is running on. Raw sockets (IP layer) are generally most efficient for UNIX machines, while ethernet frames work best on the many Windows versions where Microsoft has disabled raw sockets support. Nmap still uses raw IP packets when there is no other choice (such as non-ethernet connections).

--send_ip
Asks Nmap to send packets via raw IP sockets rather than sending lower level ethernet frames. It is the complement to the --send-eth option.discussed previously.

−-spoof_mac [mac, prefix, or vendor substring]
Ask Nmap to use the given MAC address for all of the raw ethernet frames it sends. The MAC given can take several formats. If it is simply the string "0", Nmap chooses a completely random MAC for the session. If the given string is an even number of hex digits (with the pairs optionally separated by a colon), Nmap will use those as the MAC. If less than 12 hex digits are provided, Nmap fills in the remainder of the 6 bytes with random values. If the argument isn't a 0 or hex string, Nmap looks through the nmap-mac-prefixes to find a vendor name containing the given string (it is case insensitive). If a match is found, Nmap uses the vendor's OUI (3-byte prefix) and fills out the remaining 3 bytes randomly. Valid --spoof_mac argument examples are "Apple", "0", "01:02:03:04:05:06", "deadbeefcafe", "0020F2", and "Cisco".

−f This option causes the requested scan (including ping scans) to use tiny fragmented IP packets. The idea is to split up the TCP header over several packets to make it harder for packet filters, intrusion detection systems, and other annoyances to detect what you are doing. Be careful with this! Some programs have trouble handling these tiny packets. The old-school sniffer named Sniffit segmentation faulted immediately upon receiving the first fragment. Specify this option once, and Nmap splits the packets into 8 bytes or less after the IP header. So a 20-byte TCP header would be split into 3 packets.
Two with eight bytes of the TCP header, and one with the final four. Of course each fragment also has an IP header. Specify -f again to use 16 bytes per fragment (reducing the number of fragments). Or you can specify your own offset size with the --mtu option. Don't also specify -f if you use --mtu. The offset must be a multiple of 8. While fragmented packets won't get by packet filters and firewalls that queue all IP fragments, such as the CONFIG_IP_ALWAYS_DEFRAG option in the Linux kernel, some networks can't afford the performance hit this causes and thus leave it disabled. Some source systems defragment outgoing packets in the kernel. Linux with the ip tables connection tracking module is one such example. Do a scan with a sniffer such as ethereal running to ensure that sent packets are fragmented.

Note that I do not yet have this option working on all systems. It works fine for my Linux, FreeBSD, and OpenBSD boxes and some people have reported success with other *NIX variants.

−v Verbose mode. This is a highly recommended option and it gives out more information about what is going on. You can use it twice for greater effect. You can also use **−d** a few times if you really want to get crazy with scrolling the screen!

−h This handy option display a quick reference screen of nmap usage options. As you may have noticed, this man page is not exactly a "quick reference" :)

−oN <logfilename>
This logs the results of your scans in a normal **human readable** form into the file you specify as an argument.

−oX <logfilename>
This logs the results of your scans in **XML** form into the file you specify as an argument. This allows programs to easily capture and interpret Nmap results. You can give the argument "-" (without quotes) to shoot output into stdout (for shell pipelines, etc). In this case normal output will be suppressed. Watch out for error messages if you use this (they will still go to stderr). Also note that "-v" may cause some extra information to be printed. The Document Type Definition (DTD) defining the XML output structure is available at http://www.insecure.org/nmap/data/nmap.dtd .

−-stylesheet <filename>
Nmap ships with an XSL stylesheet named nmap.xsl for viewing or translating XML output to HTML. The XML output includes an xml-stylesheet directive which points to nmap.xml where it was initially installed by Nmap (or in the current working directory on Windows). Simply load Nmap's XML output in a modern web browser and it should retrieve nmap.xsl from the filesystem and use it to render results. If you wish to use a different stylesheet, specify it as the argument to --stylesheet. You must pass the full pathname or URL. One common invocation is --stylesheet http://www.insecure.org/nmap/data/nmap.xsl . This tells a browser to load the latest version of the stylesheet from Insecure.Org. This makes it easier to view results on a machine that doesn't have Nmap (and thus nmap.xsl) installed. So the URL is often more useful, but the local filesystem locaton of nmap.xsl is used by default for privacy reasons.

−-no_stylesheet
Specify this option to prevent Nmap from associating any XSL stylesheet with its XML output. The xml-stylesheet directive is omitted.

−oG <logfilename>
This logs the results of your scans in a **grepable** form into the file you specify as an argument. This simple format provides all the information on one line (so you can easily grep for port or OS information and see all the IPs. This used to be the preferred mechanism for programs to interact with Nmap, but now we recommend XML output (-oX instead). This simple format may not contain as much information as the other formats. You can give the argument "-" (without quotes) to shoot output into stdout (for shell pipelines, etc). In this case normal output will be suppressed. Watch out for error messages if you use this (they will still go to stderr). Also note that "-v" will cause some extra information to be printed.

−oA <basefilename>
This tells Nmap to log in ALL the major formats (normal, grepable, and XML). You give a base for the filename, and the output files will be base.nmap, base.gnmap, and base.xml.

−oS <logfilename>
thIs l0gz th3 r3suLtS of YouR ScanZ iN a **s|<ipT kiDd|3** f0rM iNto THe fiL3 U sPecfy 4s an arGuMEnT! U kAn gIv3 the 4rgument "-" (wItHOUt qUOteZ) to sh00t output iNT0 stDouT!@!!

−-resume <logfilename>
A network scan that is canceled due to control-C, network outage, etc. can be resumed using this option. The logfilename must be either a normal (-oN) or grepable (-oG) log from the aborted

scan. No other options can be given (they will be the same as the aborted scan). Nmap will start on the machine after the last one successfully scanned in the log file.

−-exclude <host1 [,host2][,host3],...">
Specifies a list of targets (hosts, ranges, netblocks) that should be excluded from a scan. Useful to keep from scanning yourself, your ISP, particularly sensitive hosts, etc.

−-excludefile <exclude_file>
Same functionality as the --exclude option, only the excluded targets are provided in an newline-delimited exclude_file rather than on the command line.

−-allports
Causes version detection (-sV) to scan all open ports found, including those excluded as dangerous (likely to cause crashes or other problems) in nmap-service-probes.

−-append_output
Tells Nmap to append scan results to any output files you have specified rather than overwriting those files.

−iL <inputfilename>
Reads target specifications from the file specified RATHER than from the command line. The file should contain a list of host or network expressions separated by spaces, tabs, or newlines. Use a hyphen (-) as *inputfilename* if you want nmap to read host expressions from stdin (like at the end of a pipe). See the section *target specification* for more information on the expressions you fill the file with.

−iR <num hosts>
This option tells Nmap to generate its own hosts to scan by simply picking random numbers :). It will never end after the given number of IPs has been scanned -- use 0 for a never-ending scan. This option can be useful for statistical sampling of the Internet to estimate various things. If you are ever really bored, try *nmap −sS −PS80 −iR 0 −p 80* to find some web servers to look at.

−p <port ranges>
This option specifies what ports you want to specify. For example "-p 23" will only try port 23 of the target host(s). "−p 20-30,139,60000-" scans ports between 20 and 30, port 139, and all ports greater than 60000. The default is to scan all ports between 1 and 1024 as well as any ports listed in the services file which comes with nmap. For IP protocol scanning (-sO), this specifies the protocol number you wish to scan for (0-255).

When scanning both TCP and UDP ports, you can specify a particular protocol by preceding the port numbers by "T:" or "U:". The qualifier lasts until you specify another qualifier. For example, the argument "-p U:53,111,137,T:21-25,80,139,8080" would scan UDP ports 53,111,and 137, as well as the listed TCP ports. Note that to scan both UDP & TCP, you have to specify -sU and at least one TCP scan type (such as -sS, -sF, or -sT). If no protocol qualifier is given, the port numbers are added to all protocol lists.

−F Fast scan mode.
Specifies that you only wish to scan for ports listed in the services file which comes with nmap (or the protocols file for -sO). This is obviously much faster than scanning all 65535 ports on a host.

−D <decoy1 [,decoy2][,ME],...>
Causes a decoy scan to be performed which makes it appear to the remote host that the host(s) you specify as decoys are scanning the target network too. Thus their IDS might report 5-10 port scans from unique IP addresses, but they won't know which IP was scanning them and which were innocent decoys. While this can be defeated through router path tracing, response-dropping, and other "active" mechanisms, it is generally an extremely effective technique for hiding your IP address.

Separate each decoy host with commas, and you can optionally use "ME" as one of the decoys to represent the position you want your IP address to be used. If you put "ME" in the 6th position or later, some common port scan detectors (such as Solar Designer's excellent scanlogd) are unlikely to show your IP address at all. If you don't use "ME", nmap will put you in a random position.

Note that the hosts you use as decoys should be up or you might accidentally SYN flood your targets. Also it will be pretty easy to determine which host is scanning if only one is actually up on the network. You might want to use IP addresses instead of names (so the decoy networks don't see you in their nameserver logs).

Also note that some "port scan detectors" will firewall/deny routing to hosts that attempt port scans. The problem is that many scan types can be forged (as this option demonstrates). So attackers can cause such a machine to sever connectivity with important hosts such as its internet gateway, DNS TLD servers, sites like Windows Update, etc. Most such software offers whitelist capabilities, but you are unlikely to enumerate all of the critical machines. For this reason we never recommend taking action against port scans that can be forged, including SYN scans, UDP scans, etc. The machine you block could just be a decoy.

Decoys are used both in the initial ping scan (using ICMP, SYN, ACK, or whatever) and during the actual port scanning phase. Decoys are also used during remote OS detection (**−O**).

It is worth noting that using too many decoys may slow your scan and potentially even make it less accurate. Also, some ISPs will filter out your spoofed packets, although many (currently most) do not restrict spoofed IP packets at all.

−S <IP_Address>

In some circumstances, *nmap* may not be able to determine your source address (*nmap* will tell you if this is the case). In this situation, use −S with your IP address (of the interface you wish to send packets through).

Another possible use of this flag is to spoof the scan to make the targets think that **someone else** is scanning them. Imagine a company being repeatedly port scanned by a competitor! This is not a supported usage (or the main purpose) of this flag. I just think it raises an interesting possibility that people should be aware of before they go accusing others of port scanning them. **−e** would generally be required for this sort of usage.

−e <interface>

Tells nmap what interface to send and receive packets on. Nmap should be able to detect this but it will tell you if it cannot.

−-source_port <portnumber>

Sets the source port number used in scans. Many naive firewall and packet filter installations make an exception in their ruleset to allow DNS (53) or FTP-DATA (20) packets to come through and establish a connection. Obviously this completely subverts the security advantages of the firewall since intruders can just masquerade as FTP or DNS by modifying their source port. Obviously for a UDP scan you should try 53 first and TCP scans should try 20 before 53. Note that this is only a request -- nmap will honor it only if and when it is able to. For example, you can't do TCP ISN sampling all from one host:port to one host:port, so nmap changes the source port even if you used this option. This is an alias for the shorter, but harder to remember, -g option.

Be aware that there is a small performance penalty on some scans for using this option, because I sometimes store useful information in the source port number.

−-data_length <number>

Normally Nmap sends minimalistic packets that only contain a header. So its TCP packets are generally 40 bytes and ICMP echo requests are just 28. This option tells Nmap to append the given number of random bytes to most of the packets it sends. OS detection (−O) packets are not affected, but most pinging and portscan packets are. This slows things down, but can be slightly less conspicuous.

−n Tells Nmap to **NEVER** do reverse DNS resolution on the active IP addresses it finds. Since DNS is often slow, this can help speed things up.

−R Tells Nmap to **ALWAYS** do reverse DNS resolution on the target IP addresses. Normally this is only done when a machine is found to be alive.

−r Tells Nmap **NOT** to randomize the order in which ports are scanned.

−−ttl <value>
Sets the IPv4 time to live field in sent packets to the given value.

−−privileged
Tells Nmap to simply assume that it is privileged enough to perform raw socket sends, packet sniffing, and similar operations that usually require root privileges on UNIX systems. By default Nmap bails if such operations are requested but geteuid() is not zero. --privileged is useful with Linux kernel capabilities and similar systems that may be configured to allow unprivileged users to perform raw-packet scans. Be sure to provide this option flag before any flags for options that require privileges (SYN scan, OS detection, etc.). The NMAP_PRIVILEGED variable may be set as an equivalent alternative --privileged.

−−interactive
Starts Nmap in interactive mode, which offers an interactive Nmap prompt allowing easy launching of multiple scans (either synchronously or in the background). This is useful for people who scan from multi-user systems -- they often want to test their security without letting everyone else on the system know exactly which systems they are scanning. Use --interactive to activate this mode and then type usually more familiar and feature-complete.

−−randomize_hosts
Tells Nmap to shuffle each group of up to 2048 hosts before it scans them. This can make the scans less obvious to various network monitoring systems, especially when you combine it with slow timing options (see below).

−M <max sockets>
Sets the maximum number of sockets that will be used in parallel for a TCP connect() scan (the default). This is useful to slow down the scan a little bit and avoid crashing remote machines. Another approach is to use −sS, which is generally easier for machines to handle.

--packet_trace
Tells Nmap to show all the packets it sends and receives in a tcpdump-like format. This can be tremendously useful for debugging, and is also a good learning tool.

--datadir [directoryname]
Nmap obtains some special data at runtime in files named nmap-service-probes, nmap-services, nmap-protocols, nmap-rpc, nmap-mac-prefixes, and nmap-os-fingerprints. Nmap first searches these files in the directory option to --datadir. Any files not found there, are searched for in the directory specified by the NMAPDIR environmental variable. Next comes ~/.nmap for real and effective UIDs (POSIX systems only) or location of the Nmap executable (Win32 only), and then a compiled-in location such as /usr/local/share/nmap or /usr/share/nmap . As a last resort, Nmap will look in the current directory.

TIMING OPTIONS
Generally Nmap does a good job at adjusting for Network characteristics at runtime and scanning as fast as possible while minimizing that chances of hosts/ports going undetected. However, there are same cases where Nmap's default timing policy may not meet your objectives. The following options provide a fine level of control over the scan timing:

-T <Paranoid|Sneaky|Polite|Normal|Aggressive|Insane>
These are canned timing policies for conveniently expressing your priorities to Nmap. **Paranoid** mode scans **very** slowly in the hopes of avoiding detection by IDS systems. It serializes all scans (no parallel scanning) and generally waits at least 5 minutes between sending packets. **Sneaky** is similar, except it only waits 15 seconds between sending packets. **Polite** is meant to ease load on the network and reduce the chances of crashing machines. It serializes the probes and waits **at least** 0.4 seconds between them. Note that this is generally at least an order of magnitude slower than default scans, so only use it when you need to. **Normal** is the default Nmap behavior, which tries to run as quickly as possible without overloading the network or missing hosts/ports. **Aggressive** This option can make certain scans (especially SYN scans against heavily filtered

hosts) much faster. It is recommended for impatient folks with a fast net connection. **Insane** is only suitable for very fast networks or where you don't mind losing some information. It times out hosts in 15 minutes and won't wait more than 0.3 seconds for individual probes. It does allow for very quick network sweeps though :).

You can also reference these by number (0-5). For example, "-T0" gives you Paranoid mode and "-T5" is Insane mode. If you wish to set specific timing values such as --max_rtt_timeout or --host_timeout, place them after any -T option on the command line. Otherwise the defaults for the selected timing mode will override your choices.

--host_timeout <milliseconds>
Specifies the amount of time Nmap is allowed to spend scanning a single host before giving up on that IP. The default timing mode has no host timeout.

--max_rtt_timeout <milliseconds>
Specifies the maximum amount of time Nmap is allowed to wait for a probe response before retransmitting or timing out that particular probe. The default mode sets this to about 9000.

--min_rtt_timeout <milliseconds>
When the target hosts start to establish a pattern of responding very quickly, Nmap will shrink the amount of time given per probe. This speeds up the scan, but can lead to missed packets when a response takes longer than usual. With this parameter you can guarantee that Nmap will wait at least the given amount of time before giving up on a probe.

--initial_rtt_timeout <milliseconds>
Specifies the initial probe timeout. This is generally only useful when scanning firewalled hosts with -P0. Normally Nmap can obtain good RTT estimates from the ping and the first few probes. The default mode uses 6000.

--max_hostgroup <numhosts>
Specifies the maximum number of hosts that Nmap is allowed to scan in parallel. Most of the port scan techniques support multi-host operation, which makes them much quicker. Spreading the load among multiple target hosts makes the scans gentler. The downside is increased results latency. You need to wait for all hosts in a group to finish, rather than having them pop up one by one. Specify an argument of one for old-style (one host at a time) Nmap behavior. Note that the ping scanner handles its own grouping, and ignores this value.

--min_hostgroup <numhosts>
Specifies the minimum host group size (see previous entry). Large values (such as 50) are often beneficial for unattended scans, though they do take up more memory. Nmap may override this preference when it needs to, because a group must all use the same network interface, and some scan types can only handle one host at a time.

--max_parallelism <number>
Specifies the maximum number of scans Nmap is allowed to perform in parallel. Setting this to one means Nmap will never try to scan more than 1 port at a time. It also effects other parallel scans such as ping sweep, RPC scan, etc.

--min_parallelism <number>
Tells Nmap to scan at least the given number of ports in parallel. This can speed up scans against certain firewalled hosts by an order of magnitude. But be careful -- results will become unreliable if you push it too far.

--scan_delay <milliseconds>
Specifies the **minimum** amount of time Nmap must wait between probes. This is mostly useful to reduce network load or to slow the scan way down to sneak under IDS thresholds. Nmap will sometimes increase the delay itself when it detects many dropped packets. For example, Solaris systems tend to respond with only one ICMP port unreachable packet per second during a UDP scan. So Nmap will try to detect this and lower its rate of UDP probes to one per second.

--max_scan_delay <milliseconds>

As noted above, Nmap will sometimes enforce a special delay between sending packets. This can provide more accurate results while reducing network congestion, but it can slow the scans down substantially. By default (with no -T options specified), Nmap allows this delay to grow to one second per probe. This option allows you to set a lower or higher maximum. Even if you set it to zero, Nmap will have some delay between packet sends so that it can wait for responses and avoid having too many outstanding probes in parallel.

TARGET SPECIFICATION

Everything that isn't an option (or option argument) in nmap is treated as a target host specification. The simplest case is listing single hostnames or IP addresses on the command line. If you want to scan a subnet of IP addresses, you can append **/mask** to the hostname or IP address. **mask** must be between 0 (scan the whole Internet) and 32 (scan the single host specified). Use /24 to scan a class "C" address and /16 for a class "B".

Nmap also has a more powerful notation which lets you specify an IP address using lists/ranges for each element. Thus you can scan the whole class "B" network 192.168.*.* by specifying "192.168.*.*" or "192.168.0-255.0-255" or even "192.168.1-50,51-255.1,2,3,4,5-255". And of course you can use the mask notation: "192.168.0.0/16". These are all equivalent. If you use asterisks ("*"), remember that most shells require you to escape them with back slashes or protect them with quotes.

Another interesting thing to do is slice the Internet the other way. Instead of scanning all the hosts in a class "B", scan "*.*.5.6-7" to scan every IP address that ends in .5.6 or .5.7 Pick your own numbers. For more information on specifying hosts to scan, see the *examples* section.

EXAMPLES

Here are some examples of using nmap, from simple and normal to a little more complex/esoteric. Note that actual numbers and some actual domain names are used to make things more concrete. In their place you should substitute addresses/names from **your own network.** I do not think portscanning other networks is illegal; nor should portscans be construed by others as an attack. I have scanned hundreds of thousands of machines and have received only one complaint. But I am not a lawyer and some (anal) people may be annoyed by *nmap* probes. Get permission first or use at your own risk.

nmap -v target.example.com

This option scans all reserved TCP ports on the machine target.example.com . The −v means turn on verbose mode.

nmap -sS -O target.example.com/24

Launches a stealth SYN scan against each machine that is up out of the 255 machines on class "C" where target.example.com resides. It also tries to determine what operating system is running on each host that is up and running. This requires root privileges because of the SYN scan and the OS detection.

nmap -sX -p 22,53,110,143,4564 198.116.*.1-127

Sends an Xmas tree scan to the first half of each of the 255 possible 8 bit subnets in the 198.116 class "B" address space. We are testing whether the systems run sshd, DNS, pop3d, imapd, or port 4564. Note that Xmas scan doesn't work on Microsoft boxes due to their deficient TCP stack. Same goes with CISCO, IRIX, HP/UX, and BSDI boxes.

nmap -v --randomize_hosts -p 80 *.*.2.3-5

Rather than focus on a specific IP range, it is sometimes interesting to slice up the entire Internet and scan a small sample from each slice. This command finds all web servers on machines with IP addresses ending in .2.3, .2.4, or .2.5. If you are root you might as well add -sS. Also you will find more interesting machines starting at 127. so you might want to use "127-222" instead of the first asterisks because that section has a greater density of interesting machines (IMHO).

host -l company.com | cut -d -f 4 | ./nmap -v -iL -

Do a DNS zone transfer to find the hosts in company.com and then feed the IP addresses to *nmap.* The

above commands are for my GNU/Linux box. You may need different commands/options on other operating systems.

BUGS

Bugs? What bugs? Send me any that you find. Patches are nice too :) Remember to also send in new OS fingerprints so we can grow the database. Nmap will give you a submission URL when an appropriate fingerprint is found.

AUTHOR

Fyodor <*fyodor@insecure.org*>

DISTRIBUTION

The newest version of *nmap* can be obtained from *http://www.insecure.org/nmap/*

above, then that alternative license agreement takes precedence over these comments.

Source is provided to this software because we believe users have a right to know exactly what a program is going to do before they run it. This also allows you to audit the software for security holes (none have been found so far).

Source code also allows you to port Nmap to new platforms, fix bugs, and add new features. You are highly encouraged to send your changes to fyodor@insecure.org for possible incorporation into the main distribution. By sending these changes to Fyodor or one the Insecure.Org development mailing lists, it is assumed that you are offering Fyodor and Insecure.Com LLC the unlimited, non-exclusive right to reuse, modify, and relicense the code. Nmap will always be available Open Source, but this is important because the inability to relicense code has caused devastating problems for other Free Software projects (such as KDE and NASM). We also occasionally relicense the code to third parties as discussed above. If you wish to specify special license conditions of your contributions, just say so when you send them.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details at http://www.gnu.org/copyleft/gpl.html , or in the COPYING file included with Nmap.

It should also be noted that Nmap has been known to crash certain poorly written applications, TCP/IP stacks, and even operating systems. **Nmap should never be run against mission critical systems** unless you are prepared to suffer downtime. We acknowledge here that Nmap may crash your systems or networks and we disclaim all liability for any damage or problems Nmap could cause.

Because of the slight risk of crashes and because a few black hats like to use Nmap for reconnaissance prior to attacking systems, there are administrators who become upset and may complain when their system is scanned. Thus, it is often advisable to request permission before doing even a light scan of a network.

Nmap should never be installed with special privileges (eg suid root) for security reasons.

This product includes software developed by the Apache Software Foundation (http://www.apache.org/). The *Libpcap* portable packet capture library is distributed along with nmap. Libpcap was originally copyrighted by Van Jacobson, Craig Leres and Steven McCanne, all of the Lawrence Berkeley National Laboratory, University of California, Berkeley, CA. It is now maintained by http://www.tcpdump.org .

Regular expression support is provided by the PCRE library package, which is open source software, written by Philip Hazel, and copyright by the University of Cambridge, England. See http://www.pcre.org/ .

Nmap can optionally link to the OpenSSL cryptography toolkit, which is available from http://www.openssl.org/ .

US Export Control: Insecure.Com LLC believes that Nmap falls under US ECCN (export control classification number) 5D992. This category is called '"Information Security" "software" not controlled by 5D002'. The only restriction of this classification is AT (anti-terrorism), which applies to almost all goods and denies export to a handful of rogue nations such as Iran and North Korea. Thus exporting Nmap does not require any special license, permit, or other governmental authorization.