NMAP REFERENCE GUIDE
By Fyodor

Name

nmap — Network exploration tool and security / port scanner

nmap [*Scan Type* ...] [*Options*] { *target specification* }

Description

Nmap (“Network Mapper”) is an open source tool for network exploration and security auditing. It was designed to rapidly scan large networks, although it works fine against single hosts. Nmap uses raw IP packets in novel ways to determine what hosts are available on the network, what services (application name and version) those hosts are offering, what operating systems (and OS versions) they are running, what type of packet filters/firewalls are in use, and dozens of other characteristics. While Nmap is commonly used for security audits, many systems and network administrators find it useful for routine tasks such as network inventory, managing service upgrade schedules, and monitoring host or service uptime.

The output from Nmap is a list of scanned targets, with supplemental information on each depending on the options used. Key among that information is the “interesting ports table”. That table lists the port number and protocol, service name, and state. The state is either open, filtered, closed, or unfiltered. Open means that an application on the target machine is listening for connections/packets on that port. Filtered means that a firewall, filter, or other network obstacle is blocking the port so that Nmap cannot tell whether it is open or closed. Closed ports have no application listening on them, though they could open up at any time. Ports are classified as unfiltered when they are responsive to Nmap's probes, but Nmap cannot determine whether they are open or closed. Nmap reports the state combinations open|filtered and closed|filtered when it cannot determine which of the two states describe a port. The port table may also include software version details when version detection has been requested. When an IP protocol scan is requested (-sO), Nmap provides information on supported IP protocols rather than listening ports.

In addition to the interesting ports table, Nmap can provide further information on targets, including reverse DNS names, operating system guesses, device types, and MAC addresses.

A typical Nmap scan is shown in Example 1, “A representative Nmap scan”. The only Nmap arguments used in this example are -A, to enable OS and version detection, -T4 for faster execution, and then the two target hostnames.

Example 1. A representative Nmap scan

```
# nmap -A -T4 scanme.nmap.org playground

Starting nmap ( http://www.insecure.org/nmap/ )
Interesting ports on scanme.nmap.org (205.217.153.62):
(The 1663 ports scanned but not shown below are in state: filtered)
PORT    STATE  SERVICE VERSION
22/tcp  open   ssh     OpenSSH 3.9p1 (protocol 1.99)
53/tcp  open   domain
70/tcp  closed gopher
80/tcp  open   http    Apache httpd 2.0.52 ((Fedora))
113/tcp closed auth
```

```
Device type: general purpose
Running: Linux 2.4.X|2.5.X|2.6.X
OS details: Linux 2.4.7 - 2.6.11, Linux 2.6.0 - 2.6.11
Uptime 33.908 days (since Thu Jul 21 03:38:03 2005)

Interesting ports on playground.nmap.org (192.168.0.40):
(The 1659 ports scanned but not shown below are in state: closed)
PORT     STATE SERVICE       VERSION
135/tcp  open  msrpc         Microsoft Windows RPC
139/tcp  open  netbios-ssn
389/tcp  open  ldap?
445/tcp  open  microsoft-ds  Microsoft Windows XP microsoft-ds
1002/tcp open  windows-icfw?
1025/tcp open  msrpc         Microsoft Windows RPC
1720/tcp open  H.323/Q.931   CompTek AquaGateKeeper
5800/tcp open  vnc-http      RealVNC 4.0 (Resolution 400x250; VNC TCP port: 5900)
5900/tcp open  vnc           VNC (protocol 3.8)
MAC Address: 00:A0:CC:63:85:4B (Lite-on Communications)
Device type: general purpose
Running: Microsoft Windows NT/2K/XP
OS details: Microsoft Windows XP Pro RC1+ through final release
Service Info: OSs: Windows, Windows XP

Nmap finished: 2 IP addresses (2 hosts up) scanned in 88.392 seconds
```

Options Summary

This options summary is printed when Nmap is run with no arguments, and the latest version is always available at http://www.insecure.org/nmap/data/nmap.usage.txt. It helps people remember the most common options, but is no substitute for the in-depth documentation in the rest of this manual. Some obscure options aren't even included here.

Usage: nmap [Scan Type(s)] [Options] {target specification}

TARGET SPECIFICATION:
Can pass hostnames, IP addresses, networks, etc.
Ex: scanme.nmap.org, microsoft.com/24, 192.168.0.1; 10.0-255.0-255.1-254
-iL <inputfilename>: Input from list of hosts/networks
-iR <num hosts>: Choose random targets
--exclude <host1[,host2][,host3],...>: Exclude hosts/networks
--excludefile <exclude_file>: Exclude list from file

HOST DISCOVERY:
-sL: List Scan - simply list targets to scan
-sP: Ping Scan - go no further than determining if host is online
-P0: Treat all hosts as online -- skip host discovery
-PS/PA/PU [portlist]: TCP SYN/ACK or UDP discovery probes to given ports
-PE/PP/PM: ICMP echo, timestamp, and netmask request discovery probes

-n/-R: Never do DNS resolution/Always resolve [default: sometimes resolve]

SCAN TECHNIQUES:
-sS/sT/sA/sW/sM: TCP SYN/Connect()/ACK/Window/Maimon scans
-sN/sF/sX: TCP Null, FIN, and Xmas scans
--scanflags <flags>: Customize TCP scan flags
-sI <zombie host[:probeport]>: Idlescan
-sO: IP protocol scan
-b <ftp relay host>: FTP bounce scan

PORT SPECIFICATION AND SCAN ORDER:
-p <port ranges>: Only scan specified ports
Ex: -p22; -p1-65535; -p U:53,111,137,T:21-25,80,139,8080
-F: Fast - Scan only the ports listed in the nmap-services file)
-r: Scan ports consecutively - don't randomize

SERVICE/VERSION DETECTION:
-sV: Probe open ports to determine service/version info
--version_light: Limit to most likely probes for faster identification
--version_all: Try every single probe for version detection
--version_trace: Show detailed version scan activity (for debugging)

OS DETECTION:
-O: Enable OS detection
--osscan_limit: Limit OS detection to promising targets
--osscan_guess: Guess OS more aggressively

TIMING AND PERFORMANCE:
-T[0-6]: Set timing template (higher is faster)
--min_hostgroup/max_hostgroup <msec>: Parallel host scan group sizes
--min_parallelism/max_parallelism <msec>: Probe parallelization
--min_rtt_timeout/max_rtt_timeout/initial_rtt_timeout <msec>: Specifies probe round trip time.
--host_timeout <msec>: Give up on target after this long
--scan_delay/--max_scan_delay <msec>: Adjust delay between probes

FIREWALL/IDS EVASION AND SPOOFING:
-f; --mtu <val>: fragment packets (optionally w/given MTU)
-D <decoy1,decoy2[,ME],...>: Cloak a scan with decoys
-S <IP_Address>: Spoof source address
-e <iface>: Use specified interface
-g/--source_port <portnum>: Use given port number
--data_length <num>: Append random data to sent packets
--ttl <val>: Set IP time-to-live field

--spoof_mac <mac address, prefix, or vendor name>: Spoof your MAC address

OUTPUT:
-oN/-oX/-oS/-oG <file>: Output scan results in normal, XML, s|<rIpt kIddi3, and Grepable format, respectively, to the given filename.
-oA <basename>: Output in the three major formats at once
-v: Increase verbosity level (use twice for more effect)
-d[level]: Set or increase debugging level (Up to 9 is meaningful)
--packet_trace: Show all packets sent and received
--iflist: Print host interfaces and routes (for debugging)
--append_output: Append to rather than clobber specified output files
--resume <filename>: Resume an aborted scan
--stylesheet <path/URL>: XSL stylesheet to transform XML output to HTML
--no_stylesheet: Prevent Nmap from associating XSL stylesheet w/XML output

MISC:
-6: Enable IPv6 scanning
-A: Enables OS detection and Version detection
--datadir <dirname>: Specify custom Nmap data file location
--send_eth/--send_ip: Send packets using raw ethernet frames or IP packets
--privileged: Assume that the user is fully privileged
-V: Print version number
-h: Print this help summary page.

EXAMPLES:
nmap -v -A scanme.nmap.org
nmap -v -sP 192.168.0.0/16 10.0.0.0/8
nmap -v -iR 10000 -P0 -p 80

Target Specification

Everything on the Nmap command-line that isn't an option (or option argument) is treated as a target host specification. The simplest case is to specify a target IP address or hostname for scanning.

Sometimes you wish to scan a whole network of adjacent hosts. For this, Nmap supports CIDR-style addressing. You can append /*numbits* to an IP address or hostname and Nmap will scan every IP address for which the first *numbits* are the same as for the reference IP or hostname given. For example, 192.168.10.0/24 would scan the 256 hosts between 192.168.10.0 (binary: 11000000 10101000 00001010 00000000) and 192.168.10.255 (binary: 11000000 10101000 00001010 11111111), inclusive. 192.168.10.40/24 would do exactly the same thing. Given that the host scanme.nmap.org is at the IP address 205.217.153.62, the specification scanme.nmap.org/16 would scan the 65,536 IP addresses between 205.217.0.0 and 205.217.255.255. The smallest allowed value is /1, which scans half the Internet. The largest value is 32, which scans just the named host or IP address because all address bits are fixed.

CIDR notation is short but not always flexible enough. For example, you might want to scan 192.168.0.0/16 but skip any IPs ending with .0 or .255 because they are commonly broadcast addresses. Nmap supports this through octet range addressing. Rather than specify a normal IP

address, you can specify a comma separated list of numbers or ranges for each octet. For example, 192.168.0-255.1-254 will skip all addresses in the range that end in .0 and or .255. Ranges need not be limited to the final octects: the specifier 0-255.0-255.13.37 will perform an Internet-wide scan for all IP addresses ending in 13.37. This sort of broad sampling can be useful for Internet surveys and research.

IPv6 addresses can only be specified by their fully qualified IPv6 address or hostname. CIDR and octet ranges aren't supported for IPv6 because they are rarely useful.

Nmap accepts multiple host specifications on the command line, and they don't need to be the same type. The command **nmap scanme.nmap.org 192.168.0.0/8 10.0.0,1,3-7.0-255** does what you would expect.

While targets are usually specified on the command lines, the following options are also available to control target selection:

-iL <inputfilename> (Input from list)

Reads target specifications from *inputfilename*. Passing a huge list of hosts is often awkward on the command line, yet it is a common desire. For example, your DHCP server might export a list of 10,000 current leases that you wish to scan. Or maybe you want to scan all IP addresses *except* for those to locate hosts using unauthorized static IP addresses. Simply generate the list of hosts to scan and pass that filename to Nmap as an argument to the -iL option. Entries can be in any of the formats accepted by Nmap on the command line (IP address, hostname, CIDR, IPv6, or octet ranges). Each entry must be separated by one or more spaces, tabs, or newlines. You can specify a hyphen (-) as the filename if you want Nmap to read hosts from standard input rather than an actual file.

-iR <num hosts> (Choose random targets)

For Internet-wide surveys and other research, you may want to choose targets at random. The *num hosts* argument tells Nmap how many IPs to generate. Undesirable IPs such as those in certain private, multicast, or unallocated address ranges are automatically skipped. The argument 0 can be specified for a never-ending scan. Keep in mind that some network administrators bristle at unauthorized scans of their networks and may complain. Use this option at your own risk! If you find yourself really bored one rainy afternoon, try the command **nmap -sS -PS80 -iR 0 -p 80** to locate random web servers for browsing.

--exclude <host1[,host2][,host3],...> (Exclude hosts/networks)

Specifies a comma-separated list of targets to be excluded from the scan even if they are part of the overall network range you specify. The list you pass in uses normal Nmap syntax, so it can include hostnames, CIDR netblocks, octet ranges, etc. This can be useful when the network you wish to scan includes untouchable mission-critical servers, systems that are known to react adversely to port scans, or subnetworks administered by other people.

--excludefile <exclude_file> (Exclude list from file)

This offers the same functionality as the --exclude option, except that the excluded targets are provided in a newline, space, or tab delimited *exclude_file* rather than on the command line.

Host Discovery

One of the very first steps in any network reconnaissance mission is to reduce a (sometimes huge) set of IP ranges into a list of active or interesting hosts. Scanning every port of every single IP address is slow and usually unnecessary. Of course what makes a host interesting depends greatly on the scan purposes. Network administrators may only be interested in hosts running a certain service, while security auditors may care about every single device with an IP address. An administrator may be comfortable using just an ICMP ping to locate hosts on his internal network, while an external penetration tester may use a diverse set of dozens of probes in an attempt to evade firewall restrictions.

Because host discovery needs are so diverse, Nmap offers a wide variety of options for customizing the techniques used. Host discovery is sometimes called ping scan, but it goes well beyond the simple ICMP echo request packets associated with the ubiquitous ping tool. Users can skip the ping step entirely with a list scan (-sL) or by disabling ping (-P0), or engage the network with arbitrary combinations of multi-port TCP SYN/ACK, UDP, and ICMP probes. The goal of these probes is to solicit responses which demonstrate that an IP address is actually active (is being used by a host or network device). On many networks, only a small percentage of IP addresses are active at any given time. This is particularly common with RFC1918-blessed private address space such as 10.0.0.0/8. That network has 16 million IPs, but I have seen it used by companies with less than a thousand machines. Host discovery can find those machines in a sparsely allocated sea of IP addresses.

If no host discovery options are given, Nmap sends a TCP ACK packet destined for port 80 and an ICMP Echo Request query to each target machine. An exception to this is that an ARP scan is used for any targets which are on a local ethernet network. For unprivileged UNIX shell users, a SYN packet is sent instead of the ack using the connect() system call. These defaults are equivalent to the -PA -PE options. This host discovery is often sufficent when scanning local networks, but a more comprehensive set of discovery probes is recommended for security auditing.

The -P* options (which select ping types) can be combined. You can increase your odds of penetrating strict firewalls by sending many probe types using different TCP ports/flags and ICMP codes. Also note that ARP discovery (-PR) is done by default against targets on a local ethernet network even if you specify other -P* options, because it is almost always faster and more effective.

The following options control host discovery.

-sL (List Scan)

The list scan is a degenerate form of host discovery that simply lists each host of the network(s) specified, without sending any packets to the target hosts. By default, Nmap still does reverse-DNS resolution on the hosts to learn their names. It is often surprising how much useful information simple hostnames give out. For example, fw.chi.playboy.com is the firewall for the Chicago office of Playboy Enterprises. Nmap also reports the total number of IP addresses at the end. The list scan is a good sanity check to ensure that you have proper IP addresses for your targets. If the hosts sport domain names you do not recognize, it is worth investigating further to prevent scanning the wrong company's network.

Since the idea is to simply print a list of target hosts, options for higher level functionality such as port scanning, OS detection, or ping scanning cannot be combined with this. If you wish to disable ping scanning while still performing such higher level functionality, read up on the -P0 option.

-sP (Ping Scan)

This option tells Nmap to *only* perform a ping scan (host discovery), then print out the available hosts that responded to the scan. No further testing (such as port scanning or OS detection) is performed. This is one step more intrusive than the list scan, and can often be used for the same purposes. It allows light reconnaissance of a target network without attracting much attention. Knowing how many hosts are up is more valuable to attackers than the list provided by list scan of every single IP and host name.

Systems administrators often find this option valuable as well. It can easily be used to count available machines on a network or monitor server availability. This is often called a ping sweep, and is more reliable than pinging the broadcast address because many hosts do not reply to broadcast queries.

The -sP option sends an ICMP echo request and a TCP packet to port 80 by default. When executed by an unprivileged user, a SYN packet is sent (using a connect() call) to port 80 on the target. When a privileged user tries to scan targets on a local ethernet network, ARP requests (-PR) are used unless --send_ip was specified. The -sP option can be combined with any of the discovery probe types (the -P* options, excluding -P0) for greater flexibility. If any of those probe type and port number options are used, the default probes (ACK and echo request) are overridden. When strict firewalls are in place between the source host running Nmap and the target network, using those advanced techniques is recommended. Otherwise hosts could be missed when the firewall drops probes or their responses.

-P0 (No ping)

This option skips the Nmap discovery stage altogether. Normally, Nmap uses this stage to determine active machines for heavier scanning. By default, Nmap only performs heavy probing such as port scans, version detection, or OS detection against hosts that are found to be up. Disabling host discovery with -P0 causes Nmap to attempt the requested scanning functions against *every* target IP address specified. So if a class B sized target address space (/16) is specified on the command line, all 65,536 IP addresses are scanned. That second option character in -P0 is a zero and not the letter O. Proper host discovery is skipped as with the list scan, but instead of stopping and printing the target list, Nmap continues to perform requested functions as if each target IP is active.

-PS [portlist] (TCP SYN Ping)

This option sends an empty TCP packet with the SYN flag set. The default destination port is 80 (configurable at compile time by changing DEFAULT_TCP_PROBE_PORT in nmap.h), but an alternate port can be specified as a parameter. A comma separated list of ports can even be specified (e.g. -PS22,23,25,80,113,1050,35000), in which case probes will be attempted against each port in parallel.

The SYN flag suggests to the remote system that you are attempting to establish a connection. Normally the destination port will be closed, and a RST (reset) packet sent back. If

the port happens to be open, the target will take the second step of a TCP 3-way-handshake by responding with a SYN/ACK TCP packet. The machine running Nmap then tears down the nascent connection by responding with a RST rather than sending an ACK packet which would complete the 3-way-handshake and establish a full connection. The RST packet is sent by the kernel of the machine running Nmap in response to the unexpected SYN/ACK, not by Nmap itself.

Nmap does not care whether the port is open or closed. Either the RST or SYN/ACK response discussed previously tell Nmap that the host is available and responsive.

On UNIX boxes, only the privileged user root is generally able to send and receive raw TCP packets. For unprivileged users, a workaround is automatically employed whereby the connect() system call is initiated against each target port. This has the effect of sending a SYN packet to the target host, in an attempt to establish a connection. If connect() returns with a quick success or an ECONNREFUSED failure, the underlying TCP stack must have received a SYN/ACK or RST and the host is marked available. If the connection attempt is left hanging until a timeout is reached, the host is marked as down. This workaround is also used for IPv6 connections, as raw IPv6 packet building support is not yet available in Nmap.

-PA [portlist] (TCP ACK Ping)

The TCP ACK ping is quite similar to the just-discussed SYN ping. The difference, as you could likely guess, is that the TCP ACK flag is set instead of the SYN flag. Such an ACK packet purports to be acknowledging data over an established TCP connection, but no such connection exists. So remote hosts should always respond with a RST packet, disclosing their existence in the process.

The -PA option uses the same default port as the SYN probe (80) and can also take a list of destination ports in the same format. If an unprivileged user tries this, or an IPv6 target is specified, the connect() workaround discussed previously is used. This workaround is imperfect because connect() is actually sending a SYN packet rather than an ACK.

The reason for offering both SYN and ACK ping probes is to maximize the chances of bypassing firewalls. Many administrators configure routers and other simple firewalls to block incoming SYN packets except for those destined for public services like the company web site or mail server. This prevents other incoming connections to the organization, while allowing users to make unobstructed outgoing connections to the Internet. This non-stateful approach takes up few resources on the firewall/router and is widely supported by hardware and software filters. The Linux Netfilter/iptables firewall software offers the --syn convenience option to implement this stateless approach. When stateless firewall rules such as this are in place, SYN ping probes (-PS) are likely to be blocked when sent to closed target ports. In such cases, the ACK probe shines as it cuts right through these rules.

Another common type of firewall uses stateful rules that drop unexpected packets. This feature was initially found mostly on high-end firewalls, though it has become much more common over the years. The Linux Netfilter/iptables system supports this through the --state option, which categorizes packets based on connection state. A SYN probe is more likely to work against such a system, as unexpected ACK packets are generally recognized as bogus and dropped. A solution to this quandary is to send both SYN and ACK probes by specifying -PS and -PA.

-PU [portlist] (UDP Ping)

Another host discovery option is the UDP ping, which sends an empty (unless --data_length is specified) UDP packet to the given ports. The portlist takes the same format as with the previously discussed -PS and -PA options. If no ports are specified, the default is 31338. This default can be configured at compile-time by changing DEFAULT_UDP_PROBE_PORT in nmap.h. A highly uncommon port is used by default because sending to open ports is often undesirable for this particular scan type.

Upon hitting a closed port on the target machine, the UDP probe should elicit an ICMP port unreachable packet in return. This signifies to Nmap that the machine is up and available. Many other types of ICMP errors, such as host/network unreachables or TTL exceeded are indicative of a down or unreachable host. A lack of response is also interpreted this way. If an open port is reached, most services simply ignore the empty packet and fail to return any response. This is why the default probe port is 31338, which is highly unlikely to be in use. A few services, such as chargen, will respond to an empty UDP packet, and thus disclose to Nmap that the machine is available.

The primary advantage of this scan type is that it bypasses firewalls and filters that only screen TCP. For example, I once owned a Linksys BEFW11S4 wireless broadband router. The external interface of this device filtered all TCP ports by default, but UDP probes would still elicit port unreachable messages and thus give away the device.

-PE; -PP; -PM (ICMP Ping Types)

In addition to the unusual TCP and UDP host discovery types discussed previously, Nmap can send the standard packets sent by the ubiquitous ping program. Nmap sends an ICMP type 8 (echo request) packet to the target IP addresses, expecting a type 0 (Echo Reply) in return from available hosts. Unfortunately for network explorers, many hosts and firewalls now block these packets, rather than responding as required by RFC 1122. For this reason, ICMP-only scans are rarely reliable enough against unknown targets over the Internet. But for system administrators monitoring an internal network, they can be a practical and efficient approach. Use the -PE option to enable this echo request behavior.

While echo request is the standard ICMP ping query, Nmap does not stop there. The ICMP standard (RFC 792) also specifies timestamp request, information request, and address mask request packets as codes 13, 15, and 17, respectively. While the ostensible purpose for these queries is to learn information such as address masks and current times, they can easily be used for host discovery. A system that replies is up and available. Nmap does not currently implement information request packets, as they are not widely supported. RFC 1122 insists that “a host SHOULD NOT implement these messages”. Timestamp and address mask queries can be sent with the -PP and -PM options, respectively. A timestamp reply (ICMP code 14) or address mask reply (code 18) discloses that the host is available. These two queries can be valuable when admins specifically block echo request packets while forgetting that other ICMP queries can be used for the same purpose.

-PR (ARP Ping)

One of the most common Nmap usage scenarios is to scan an ethernet LAN. On most LANs, especially those using RFC1918-blessed private address ranges, the vast majority of IP addresses are unused at any given time. When Nmap tries to send a raw IP packet such as an

ICMP echo request, the operating system must determine the destination hardware (ARP) address corresponding to the target IP so that it can properly address the ethernet frame. This is often slow and problematic, since operating systems weren't written with the expectation that they would need to do millions of ARP requests against unavailable hosts in a short time period.

ARP scan puts Nmap and its optimized algorithms in charge of ARP requests. And if it gets a response back, Nmap doesn't even need to worry about the IP-based ping packets since it already knows the host is up. This makes ARP scan much faster and more reliable than IP-based scans. So it is done by default when scanning ethernet hosts that Nmap detects are on a local ethernet network. Even if different ping types (such as -PI or -PS) are specified, Nmap uses ARP instead for any of the targets which are on the same LAN. If you absolutely don't want to do an ARP scan, specify --send_ip.

-n (No DNS resolution)

Tells Nmap to *never* do reverse DNS resolution on the active IP addresses it finds. Since DNS is often slow, this speeds things up.

-R (DNS resolution for all targets)

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

--system_dns (Use system DNS resolver)

By default, Nmap resolves IP addresses by sending queries directly to the name servers configured on your host and then listening for responses. Many requests (often dozens) are performed in parallel for performance. Specify this option if you wish to use your system resolver instead (one IP at a time via the getnameinfo() call). This is slower and rarely useful unless there is a bug in the Nmap DNS code -- please contact us if that is the case. The system resolver is always used for IPv6 scans.

Port Scanning Basics

While Nmap has grown in functionality over the years, it began as an efficient port scanner, and that remains its core function. The simple command **nmap *target*** scans more than 1660 TCP ports on the host *target*. While many port scanners have traditionally lumped all ports into the open or closed states, Nmap is much more granular. It divides ports into six states: open, closed, filtered, unfiltered, open|filtered, or closed|filtered.

These states are not intrinsic properties of the port itself, but describe how Nmap sees them. For example, an Nmap scan from the same network as the target may show port 135/tcp as open, while a scan at the same time with the same options from across the Internet might show that port as filtered.

The six port states recognized by Nmap

open

An application is actively accepting TCP connections or UDP packets on this port. Finding these is often the primary goal of port scanning. Security-minded people know that each open port is an avenue for attack. Attackers and pen-testers want to exploit the open ports, while administrators try to close or protect them with firewalls without thwarting legitimate users. Open ports are also interesting for non-security scans because they show services available for use on the network.

closed

A closed port is accessible (it receives and responds to Nmap probe packets), but there is no application listening on it. They can be helpful in showing that a host is up on an IP address (host discovery, or ping scanning), and as part of OS detection. Because closed ports are reachable, it may be worth scanning later in case some open up. Administrators may want to consider blocking such ports with a firewall. Then they would appear in the filtered state, discussed next.

filtered

Nmap cannot determine whether the port is open because packet filtering prevents its probes from reaching the port. The filtering could be from a dedicated firewall device, router rules, or host-based firewall software. These ports frustrate attackers because they provide so little information. Sometimes they respond with ICMP error messages such as type 3 code 13 (destination unreachable: communication administratively prohibited), but filters that simply drop probes without responding are far more common. This forces Nmap to retry several times just in case the probe was dropped due to network congestion rather than filtering. This slows down the scan dramatically.

unfiltered

The unfiltered state means that a port is accessible, but Nmap is unable to determine whether it is open or closed. Only the ACK scan, which is used to map firewall rulesets, classifies ports into this state. Scanning unfiltered ports with other scan types such as Window scan, SYN scan, or FIN scan, may help resolve whether the port is open.

open|filtered

Nmap places ports in this state when it is unable to determine whether a port is open or filtered. This occurs for scan types in which open ports give no response. The lack of response could also mean that a packet filter dropped the probe or any response it elicited. So Nmap does not know for sure whether the port is open or being filtered. The UDP, IP Protocol, FIN, Null, and Xmas scans classify ports this way.

closed|filtered

This state is used when Nmap is unable to determine whether a port is closed or filtered. It is only used for the IPID Idle scan.

Port Scanning Techniques

As a novice performing automotive repair, I can struggle for hours trying to fit my rudimentary tools (hammer, duct tape, wrench, etc.) to the task at hand. When I fail miserably and tow my jalopy to a real mechanic, he invariably fishes around in a huge tool chest until pulling out the perfect gizmo which makes the job seem effortless. The art of port scanning is similar. Experts understand the dozens of scan techniques and choose the appropriate one (or combination) for a given task. Inexperienced users and script kiddies, on the other hand, try to solve every problem with the default SYN scan. Since Nmap is free, the only barrier to port scanning mastery is knowledge. That certainly beats the automotive world, where it may take great skill to determine that you need a strut spring compressor, then you still have to pay thousands of dollars for it.

Most of the scan types are only available to privileged users. This is because they send and receive raw packets, which requires root access on UNIX systems. Using an administrator account on Windows is recommended, though Nmap sometimes works for unprivileged users on that platform when WinPcap has already been loaded into the OS. Requiring root privileges was a serious limitation when Nmap was released in 1997, as many users only had access to shared shell accounts. Now, the world is different. Computers are cheaper, far more people have always-on direct Internet access, and desktop UNIX systems (including Linux and MAC OS X) are prevalent. A Windows version of Nmap is now available, allowing it to run on even more desktops. For all these reasons, users have less need to run Nmap from limited shared shell accounts. This is fortunate, as the privileged options make Nmap far more powerful and flexible.

While Nmap attempts to produce accurate results, keep in mind that all of its insights are based on packets returned by the target machines (or firewalls in front of them). Such hosts may be untrustworthy and send responses intended to confuse or mislead Nmap. Much more common are non-RFC-compliant hosts that do not respond as they should to Nmap probes. FIN, Null, and Xmas scans are particularly susceptible to this problem. Such issues are specific to certain scan types and so are discussed in the individual scan type entries.

This section documents the dozen or so port scan techniques supported by Nmap. Only one method may be used at a time, except that UDP scan (-sU) may be combined with any one of the TCP scan types. As a memory aid, port scan type options are of the form -s*C*, where *C* is a prominent character in the scan name, usually the first. The one exception to this is the deprecated FTP bounce scan (-b). By default, Nmap performs a SYN Scan, though it substitutes a Connect() scan if the user does not have proper privileges to send raw packets (requires root access on UNIX) or if IPv6 targets were specified. Of the scans listed in this section, unprivileged users can only execute connect() and ftp bounce scans.

-sS (TCP SYN scan)

> SYN scan is the default and most popular scan option for good reasons. It can be performed quickly, scanning thousands of ports per second on a fast network not hampered by intrusive firewalls. SYN scan is relatively unobtrusive and stealthy, since it never completes TCP connections. It also works against any compliant TCP stack rather than depending on idiosyncrasies of specific platforms as Nmap's Fin/Null/Xmas, Maimon and Idle scans do. It also allows clear, reliable differentiation between the open, closed, and filtered states.
>
> 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 then wait for a response. A SYN/ACK indicates the port is listening (open), while a RST (reset) is indicative of a non-listener. If no response is received after several retransmissions, the port is

marked as filtered. The port is also marked filtered if an ICMP unreachable error (type 3, code 1,2, 3, 9, 10, or 13) is received.

-sT (TCP connect() scan)

TCP Connect() scan is the default TCP scan type when SYN scan is not an option. This is the case when a user does not have raw packet privileges or is scanning IPv6 networks. Instead of writing raw packets as most other scan types do, Nmap asks the underlying operating system to establish a connection with the target machine and port by issuing the connect() system call. This is the same high-level system call that web browsers, P2P clients, and most other network-enabled applications use to establish a connection. It is part of a programming interface known as the Berkeley Sockets API. Rather than read raw packet responses off the wire, Nmap uses this API to obtain status information on each connection attempt.

When SYN scan is available, it is usually a better choice. Nmap has less control over the high level connect() call than with raw packets, making it less efficient. The system call completes connections to open target ports rather than performing the half-open reset that SYN scan does. Not only does this take longer and require more packets to obtain the same information, but target machines are more likely to log the connection. A decent IDS will catch either, but most machines have no such alarm system. Many services on your average UNIX system will add a note to syslog, and sometimes a cryptic error message, when Nmap connects and then closes the connection without sending data. Truly pathetic services crash when this happens, though that is uncommon. An administrator who sees a bunch of connection attempts in her logs from a single system should know that she has been connect scanned.

-sU (UDP scans)

While most popular services on the Internet run over the TCP protocol, UDP services are widely deployed. DNS, SNMP, and DHCP (registered ports 53, 161/162, and 67/68) are three of the most common. Because UDP scanning is generally slower and more difficult than TCP, some security auditors ignore these ports. This is a mistake, as exploitable UDP services are quite common and attackers certainly don't ignore the whole protocol. Fortunately, Nmap can help inventory UDP ports.

UDP scan is activated with the -sU option. It can be combined with a TCP scan type such as SYN scan (-sS) to check both protocols during the same run.

UDP scan works by sending an empty (no data) UDP header to every targeted port. If an ICMP port unreachable error (type 3, code 3) is returned, the port is closed. Other ICMP unreachable errors (type 3, codes 1, 2, 9, 10, or 13) mark the port as filtered. Occasionally, a service will respond with a UDP packet, proving that it is open. If no response is received after retransmissions, the port is classified as open|filtered. This means that the port could be open, or perhaps packet filters are blocking the communication. Versions scan (-sV) can be used to help differentiate the truly open ports from the filtered ones.

A big challenge with UDP scanning is doing it quickly. Open and filtered ports rarely send any response, leaving Nmap to time out and then conduct retransmissions just in case the probe or response were lost. Closed ports are often an even bigger problem. They usually send back an ICMP port unreachable error. But unlike the RST packets sent by closed TCP ports in response to a SYN or Connect scan, many hosts rate limit ICMP port unreachable messages

by default. Linux and Solaris are particularly strict about this. For example, the Linux 2.4.20 kernel limits destination unreachable messages to one per second (in net/ipv4/icmp.c).

Nmap detects rate limiting and slows down accordingly to avoid flooding the network with useless packets that the target machine will drop. Unfortunately, a Linux-style limit of one packet per second makes a 65,536-port scan take more than 18 hours. Ideas for speeding your UDP scans up include scanning more hosts in parallel, doing a quick scan of just the popular ports first, scanning from behind the firewall, and using --host_timeout to skip slow hosts.

-sN; -sF; -sX (TCP Null, FIN, and Xmas scans)

These three scan types (even more are possible with the --scanflags option described in the next section) exploit a subtle loophole in the TCP RFC to differentiate between open and closed ports. Page 65 says that “if the [destination] port state is CLOSED .... an incoming segment not containing a RST causes a RST to be sent in response.” Then the next page discusses packets sent to open ports without the SYN, RST, or ACK bits set, stating that: “you are unlikely to get here, but if you do, drop the segment, and return.”

When scanning systems compliant with this RFC text, any packet not containing SYN, RST, or ACK bits will result in a returned RST if the port is closed and no response at all if the port is open. As long as none of those three bits are included, any combination of the other three (FIN, PSH, and URG) are OK. Nmap exploits this with three scan types:

- Null scan (-sN) : Does not set any bits (tcp flag header is 0)
- FIN scan (-sF) : Sets just the TCP FIN bit.
- Xmas scan (-sX) : Sets the FIN, PSH, and URG flags, lighting the packet up like a Christmas tree.

These three scan types are exactly the same in behavior except for the TCP flags set in probe packets. If a RST packet is received, the port is considered closed, while no response means it is open|filtered. The port is marked filtered if an ICMP unreachable error (type 3, code 1, 2, 3, 9, 10, or 13) is received.

The key advantage to these scan types is that they can sneak through certain non-stateful firewalls and packet filtering routers. Another advantage is that these scan types are a little more stealthy than even a SYN scan. Don't count on this though -- most modern IDS products can be configured to detect them. The big downside is that not all systems follow RFC 793 to the letter. A number of systems send RST responses to the probes regardless of whether the port is open or not. This causes all of the ports to be labeled closed. Major operating systems that do this are Microsoft Windows, many Cisco devices, BSDI, and IBM OS/400. This scan does work against most UNIX-based systems though. Another downside of these scans is that they can't distinguish open ports from certain filtered ones, leaving you with the response open|filtered.

-sA (TCP ACK scan)

This scan is different than the others discussed so far in that it never determines open (or even open|filtered) ports. It is used to map out firewall rulesets, determining whether they are stateful or not and which ports are filtered.

The ACK scan probe packet has only the ACK flag set (unless you use --scanflags). When scanning unfiltered systems, open and closed ports will both return a RST packet. Nmap then labels them as unfiltered, meaning that they are reachable by the ACK packet, but whether they are open or closed is undetermined. Ports that don't respond, or send certain ICMP error messages back (type 3, code 1, 2, 3, 9, 10, or 13), are labeled filtered.

-sW (TCP Window scan)

Window scan is exactly the same as ACK scan except that it exploits an implementation detail of certain systems to differentiate open ports from closed ones, rather than always printing unfiltered when a RST is returned. It does this by examining the TCP Window field of the RST packets returned. On some systems, open ports use a positive window size (even for RST packets) while closed ones have a zero window. So instead of always listing a port as unfiltered when it receives a RST back, Window scan lists the port as open or closed if the TCP Window value in that reset is positive or zero, respectively.

This scan relies on an implementation detail of a minority of systems out on the Internet, so you can't always trust it. Systems that don't support it will usually return all ports closed. Of course, it is possible that the machine really has no open ports. If most scanned ports are closed but a few common port numbers (such as 22, 25, 53) are filtered, the system is most likely susceptible. Occasionally, systems will even show the exact opposite behavior. If your scan shows 1000 open ports and 3 closed or filtered ports, then those three may very well be the truly open ones.

-sM (TCP Maimon scan)

The Maimon scan is named after its discoverer, Uriel Maimon. He described the technique in Phrack Magazine issue #49 (November 1996). Nmap, which included this technique, was released two issues later. This technique is exactly the same as Null, FIN, and Xmas scans, except that the probe is FIN/ACK. According to RFC 793 (TCP), a RST packet should be generated in response to such a probe whether the port is open or closed. However, Uriel noticed that many BSD-derived systems simply drop the packet if the port is open.

--scanflags (Custom TCP scan)

Truly advanced Nmap users need not limit themselves to the canned scan types offered. The --scanflags option allows you to design your own scan by specifying arbitrary TCP flags. Let your creative juices flow, while evading intrusion detection systems whose vendors simply paged through the Nmap man page adding specific rules!

The --scanflags argument can be a numerical flag value such as 9 (PSH and FIN), but using symbolic names is easier. Just mash together any combination of URG, ACK, PSH, RST, SYN, and FIN. For example, --scanflags URGACKPSHRSTSYNFIN sets everything, though it's not very useful for scanning. The order these are specified in is irrelevant.

In addition to specifying the desired flags, you can specify a TCP scan type (such as -sA or -sF). That base type tells Nmap how to interpret responses. For example, a SYN scan

considers no-response to indicate a filtered port, while a FIN scan treats the same as open|filtered. Nmap will behave the same way it does for the base scan type, except that it will use the TCP flags you specify instead. If you don't specify a base type, SYN scan is used.

-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). This fascinating scan type is too complex to fully describe in this reference guide, so I wrote and posted an informal paper with full details 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).

You can add a colon followed by a port number to the zombie host if you wish to probe a particular port on the zombie for IPID changes. Otherwise Nmap will use the port it uses by default for tcp pings (80).

-sO (IP protocol scan)

IP Protocol scan allows you to determine which IP protocols (TCP, ICMP, IGMP, etc.) are supported by target machines. This isn't technically a port scan, since it cycles through IP protocol numbers rather than TCP or UDP port numbers. Yet it still uses the -p option to select scanned protocol numbers, reports its results within the normal port table format, and even uses the same underlying scan engine as the true port scanning methods. So it is close enough to a port scan that it belongs here.

Besides being useful in its own right, protocol scan demonstrates the power of open source software. While the fundamental idea is pretty simple, I had not thought to add it nor received any requests for such functionality. Then in the summer of 2000, Gerhard Rieger conceived the idea, wrote an excellent patch implementing it, and sent it to the nmap-hackers mailing list. I incorporated that patch into the Nmap tree and released a new version the next day. Few pieces of commercial software have users enthusiastic enough to design and contribute their own improvements!

Protocol scan works in a similar fashion to UDP scan. Instead of iterating through the port number field of a UDP packet, it sends IP packet headers and iterates through the 8-bit IP protocol field. The headers are usually empty, containing no data and not even the proper header for the claimed protocol. The three exceptions are TCP, UDP, and ICMP. A proper protocol header for those is included since some systems won't send them otherwise and because Nmap already has functions to create them. Instead of watching for ICMP port unreachable messages, protocol scan is on the lookout for ICMP *protocol* unreachable messages. If Nmap receives any response in any protocol from the target host, Nmap marks that protocol as open. An ICMP protocol unreachable error (type 3, code 2) causes the protocol to be marked as closed Other ICMP unreachable errors (type 3, code 1, 3, 9, 10, or

13) cause the protocol to be marked filtered (though they prove that ICMP is open at the same time). If no response is received after retransmissions, the protocol is marked open|filtered.

-b <ftp relay host> (FTP bounce scan)

An interesting feature of the FTP protocol (RFC 959) is support for so-called proxy ftp connections. This allows a user to connect to one FTP server, then ask that files be sent to a third-party server. Such a feature is ripe for abuse on many levels, so most servers have ceased supporting it. One of the abuses this feature allows is causing the FTP server to port scan other hosts. Simply ask the FTP server to send a file to each interesting port of a target host in turn. The error message will describe whether the port is open or not. This is a good way to bypass firewalls because organizational FTP servers are often placed where they have more access to other internal hosts than any old Internet host would. Nmap supports ftp bounce scan with the -b option. It takes an argument of the form *username*:*password*@*server*:*port*. *Server* is the name or IP address of a vulnerable FTP server. As with a normal URL, you may omit *username*:*password*, in which case anonymous login credentials (user: anonymous password:-wwwuser@) are used. The port number (and preceding colon) may be omitted as well, in which case the default FTP port (21) on *server* is used.

This vulnerability was widespread in 1997 when Nmap was released, but has largely been fixed. Vulnerable servers are still around, so it is worth trying when all else fails. If bypassing a firewall is your goal, scan the target network for open port 21 (or even for any ftp services if you scan all ports with version detection), then try a bounce scan using each. Nmap will tell you whether the host is vulnerable or not. If you are just trying to cover your tracks, you don't need to (and, in fact, shouldn't) limit yourself to hosts on the target network. Before you go scanning random Internet addresses for vulnerable FTP servers, consider that sysadmins may not appreciate you abusing their servers in this way.

Port Specification and Scan Order

In addition to all of the scan methods discussed previously, Nmap offers options for specifying which ports are scanned and whether the scan order is randomized or sequential. By default, Nmap scans all ports up to and including 1024 as well as higher numbered ports listed in the nmap-services file for the protocol(s) being scanned.

-p <port ranges> (Only scan specified ports)

This option specifies which ports you want to scan and overrides the default. Individual port numbers are OK, as are ranges separated by a hyphen (e.g. 1-1023). The beginning and/or end values of a range may be omitted, causing Nmap to use 1 and 65535, respectively. So you can specify -p- to scan ports from 1 through 65535. Scanning port zero is allowed if you specify it explicitly. For IP protocol scanning (-sO), this option specifies the protocol numbers 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 (limited port) scan)

Specifies that you only wish to scan for ports listed in the nmap-services file which comes with nmap (or the protocols file for -sO). This is much faster than scanning all 65535 ports on a host. Because this list contains so many TCP ports (more than 1200), the speed difference from a default TCP scan (about 1650 ports) isn't dramatic. The difference can be enormous if you specify your own tiny nmap-services file using the --datadir option.

-r (Don't randomize ports)

By default, Nmap randomizes the scanned port order (except that certain commonly accessible ports are moved near the beginning for efficiency reasons). This randomization is normally desirable, but you can specify -r for sequential port scanning instead.

Service and Version Detection

Point Nmap at a remote machine and it might tell you that ports 25/tcp, 80/tcp, and 53/udp are open. Using its nmap-services database of about 2,200 well-known services, Nmap would report that those ports probably correspond to a mail server (SMTP), web server (HTTP), and name server (DNS) respectively. This lookup is usually accurate -- the vast majority of daemons listening on TCP port 25 are, in fact, mail servers. However, you should not bet your security on this! People can and do run services on strange ports.

Even if Nmap is right, and the hypothetical server above is running SMTP, HTTP, and DNS servers, that is not a lot of information. When doing vulnerability assessments (or even simple network inventories) of your companies or clients, you really want to know which mail and DNS servers and versions are running. Having an accurate version number helps dramatically in determining which exploits a server is vulnerable to. Version detection helps you obtain this information.

After TCP and/or UDP ports are discovered using one of the other scan methods, version detection interrogates those ports to determine more about what is actually running. The nmap-service-probes database contains probes for querying various services and match expressions to recognize and parse responses. 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, hostname, device type (e.g. printer, router), the OS family (e.g. Windows, Linux) and sometimes miscellaneous details like whether an X server is open to connections, the SSH protocol version, or the KaZaA user name). Of course, most services don't provide all of this information. If Nmap was compiled with OpenSSL support, it will connect to SSL servers to deduce the service listening behind that encryption layer. When RPC services are discovered, the Nmap RPC grinder (-sR) is automatically used to determine the RPC program and version numbers. Some UDP ports are left in the open|filtered state after a UDP port 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. open|filtered TCP ports are treaded the same way. Note that the Nmap -A option enables version detection among other things. A paper documenting the workings, usage, and customization of version detection is available at http://www.insecure.org/nmap/vscan/.

When Nmap receives responses from a service but cannot match them to its database, it prints out a special fingerprint and a URL for you to submit if to if you know for sure what is running on the port. Please take a couple minutes to make the submission so that your find can benefit everyone. Thanks to these submissions, Nmap has about 3,000 pattern matches for more than 350 protocols such as smtp, ftp, http, etc.

Version detection is enabled and controlled with the following options:

-sV (Version detection)

Enables version detection, as discussed above. Alternatively, you can use -A to enable both OS detection and version detction.

--allports (Don't exclude any ports from version detection)

By default, Nmap version detection skips TCP port 9100 because some printers simply print anything sent to that port, leading to dozens of pages of HTTP get requests, binary SSL session requests, etc. This behavior can be changed by modifying or removing the Exclude directive in nmap-service-probes, or you can specify --allports to scan all ports regardless of any Exclude directive.

--version_intensity <intensity> (Set version scan intensity)

When performing a version scan (-sV), nmap sends a series of probes, each of which is assigned a rarity value between 1 and 9. The lower-numbered probes are effective against a wide variety of common services, while the higher numbered ones are rarely useful. The intensity level specifies which probes should be applied. The higher the number, the more likely it is the service will be correctly identified. However, high intensity scans take longer. The intensity must be between 0 and 9. The default is 7. When a probe is registered to the target port via the nmap-service-probes ports directive, that probe is tried regardless of intensity level. This ensures that the DNS probes will always be attempted against any open port 53, the SSL probe will be done against 443, etc.

--version_light (Enablie light mode)

This is a convenience alias for --version_intensity 2. This light mode makes version scanning much faster, but it is slightly less likely to identi**fy services.**

--version_all (Try every single probe)

An alias for --version_intensity 9, ensuring that every single probe is attempted against each port.

--version_trace (Trace version scan activity)

This causes Nmap to print out extensive debugging info about what version scanning is doing. It is a subset of what you get with --packet_trace.

-sR (RPC scan)

This method works in conjunction with the various port scan methods of Nmap. It takes all the TCP/UDP ports found open and 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. This is automatically enabled as part of version scan (-sV) if you request that. As version detection includes this and is much more comprehensive, -sR is rarely needed.

OS Detection

One of Nmap's best-known features is remote OS detection using TCP/IP stack fingerprinting. Nmap sends a series of TCP and UDP packets to the remote host and examines practically every bit in the responses. After performing dozens of tests such as TCP ISN sampling, TCP options support and ordering, IPID sampling, and the initial window size check, Nmap compares the results to its nmap-os-fingerprints database of more than 1500 known OS fingerprints and prints out the OS details if there is a match. Each fingerprint includes a freeform textual description of the OS, and a classification which provides the vendor name (e.g. Sun), underlying OS (e.g. Solaris), OS generation (e.g. 10), and device type (general purpose, router, switch, game console, etc).

If Nmap is unable to guess the OS of a machine, and conditions are good (e.g. at least one open port and one closed port were found), 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.

OS detection enables several other tests which make use of information that is gathered during the process anyway. One of these is 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 is TCP Sequence Predictability Classification. This measures approximately how hard it is to establish a forged TCP connection against the remote host. It is useful for exploiting source-IP based trust relationships (rlogin, firewall filters, etc) or for hiding the source of an attack. This sort of spoofing is rarely performed any more, but many machines are still vulnerable to it. 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 in verbose (-v) mode. When verbose mode is enabled along 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.

A paper documenting the workings, usage, and customization of version detection is available in more than a dozen languages at http://www.insecure.org/nmap/nmap-fingerprinting-article.html.

OS detection is enabled and controlled with the following options:

-O (Enable OS detection)

Enables OS detection, as discussed above. Alternatively, you can use -A to enable both OS detection and version detction.

--osscan_limit (Limit OS detection to promising targets)

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 with -O or -A.

--osscan_guess; --fuzzy (Guess OS detection results)

When Nmap is unable to detect a perfect OS match, it sometimes offers up near-matches as possibilities. The match has to be very close for Nmap to do this by default. Either of these (equivalent) options make Nmap guess more aggressively.

Timing and Performance

One of my highest Nmap development priorities has always been performance. A default scan (**nmap *hostname***) of a host on my local network takes a fifth of a second. That is barely enough time to blink, but adds up when you are scanning tens or hundreds of thousands of hosts. Moreover, certain scan options such as UDP scanning and version detection can increase scan times substantially. So can certain firewall configurations, particularly response rate limiting. While Nmap utilizes parallelism and many advanced algorithms to accelerate these scans, the user has ultimate control over how Nmap runs. Expert users carefully craft Nmap commands to obtain only the information they care about while meeting their time constraints.

Techniques for improving scan times include omitting non-critical tests, and upgrading to the latest version of Nmap (performance enhancements are made frequently). Optimizing timing parameters can also make a substantial difference. Those options are listed below.

--min_hostgroup <milliseconds>; --max_hostgroup <milliseconds> (Adjust parallel scan group sizes)

Nmap has the ability to port scan or version scan multiple hosts in parallel. Nmap does this by dividing the target IP space into groups and then scanning one group at a time. In general, larger groups are more efficient. The downside is that host results can't be provided until the whole group is finished. So if Nmap started out with a group size of 50, the user would not receive any reports (except for the updates offered in verbose mode) until the first 50 hosts are completed.

By default, Nmap takes a compromise approach to this conflict. It starts out with a group size as low as five so the first results come quickly and then increases the groupsize to as high as 1024. The exact default numbers depend on the options given. For efficiency reasons, Nmap uses larger group sizes for UDP or few-port TCP scans.

When a maximum group size is specified with --max_hostgroup, Nmap will never exceed that size. Specify a minimum size with --min_hostgroup and Nmap will try to keep group sizes above that level. Nmap may have to use smaller groups than you specify if there are not enough target hosts left on a given interface to fulfill the specified minimum. Both may be set to keep the group size within a specific range, though this is rarely desired.

The primary use of these options is to specify a large minimum group size so that the full scan runs more quickly. A common choice is 256 to scan a network in Class C sized chunks. For a

scan with many ports, exceeding that number is unlikely to help much. For scans of just a few port numbers, host group sizes of 2048 or more may be helpful.

--min_parallelism <milliseconds>; --max_parallelism <milliseconds> (Adjust probe parallelization)

These options control the total number of probes that may be outstanding for a host group. They are used for port scanning and host discovery. By default, Nmap calculates an ever-changing ideal parallelism based on network performance. If packets are being dropped, Nmap slows down and allows fewer outstanding probes. The ideal probe number slowly rises as the network proves itself worthy. These options place minimum or maximum bounds on that variable. By default, the ideal parallelism can drop to 1 if the network proves unreliable and rise to several hundred in perfect conditions.

The most common usage is to set --min_parallelism to a number higher than one to speed up scans of poorly performing hosts or networks. This is a risky option to play with, as setting it too high may affect accuracy. Setting this also reduces Nmap's ability to control parallelism dynamically based on network conditions. A value of ten might be reasonable, though I only adjust this value as a last resort.

The --max_parallelism option is sometimes set to one to prevent Nmap from sending more than one probe at a time to hosts. This can be useful in combination with --scan_delay (discussed later), although the latter usually serves the purpose well enough by itself.

--min_rtt_timeout <milliseconds>, --max_rtt_timeout <milliseconds>, --initial_rtt_timeout <milliseconds> (Adjust probe timeouts)

Nmap maintains a running timeout value for determining how long it will wait for a probe response before giving up or retransmitting the probe. This is calculated based on the response times of previous probes. If the network latency shows itself to be significant and variable, this timeout can grow to several seconds. It also starts at a conservative (high) level and may stay that way for a while when Nmap scans unresponsive hosts.

These options take a value in milliseconds. Specifying a lower --max_rtt_timeout and --initial_rtt_timeout than the defaults can cut scan times significantly. This is particularly true for pingless (-P0) scans, and those against heavily filtered networks. Don't get too aggressive though. The scan can end up taking longer if you specify such a low value that many probes are timing out and retransmitting while the response is in transit.

If all the hosts are on a local network, 100 milliseconds is a reasonable aggressive --max_rtt_timeout value. If routing is involved, ping a host on the network first with the ICMP ping utility, or with a custom packet crafter such as hping2 that is more likely to get through a firewall. Look at the maximum round trip time out of ten packets or so. You might want to double that for the --initial_rtt_timeout and triple or quadruple it for the --max_rtt_timeout. I generally do not set the maximum rtt below 100ms, no matter what the ping times are. Nor do I exceed 1000ms.

--min_rtt_timeout is a rarely used option that could be useful when a network is so unreliable that even Nmap's default is too aggressive. Since Nmap only reduces the timeout down to the minimum when the network seems to be reliable, this need is unusual and should be reported as a bug to the nmap-dev mailing list.

--host_timeout <milliseconds> (Give up on slow target hosts)

Some hosts simply take a *long* time to scan. This may be due to poorly performing or unreliable networking hardware or software, packet rate limiting, or a restrictive firewall. The slowest few percent of the scanned hosts can eat up a majority of the scan time. Sometimes it is best to cut your losses and skip those hosts initially. This can be done by specifying --host_timeout with the number of milliseconds you are willing to wait. I often specify 1800000 to ensure that Nmap doesn't waste more than half an hour on a single host. Note that Nmap may be scanning other hosts at the same time during that half an hour as well, so it isn't a complete loss. A host that times out is skipped. No port table, OS detection, or version detection results are printed for that host.

--scan_delay <milliseconds>; --max_scan_delay <milliseconds> (Adjust delay between probes)

This option causes Nmap to wait at least the given number of milliseconds between each probe it sends to a given host. This is particularly useful in the case of rate limiting. Solaris machines (among many others) will usually respond to UDP scan probe packets with only one ICMP message per second. Any more than that sent by Nmap will be wasteful. A --scan_delay of 1000 will keep Nmap at that slow rate. Nmap tries to detect rate limiting and adjust the scan delay accordingly, but it doesn't hurt to specify it explicitly if you already know what rate works best.

Another use of --scan_delay is to evade threshold based intrusion detection and prevention systems (IDS/IPS).

-T <Paranoid|Sneaky|Polite|Normal|Aggressive|Insane> (Set a timing template)

While the fine grained timing controls discussed in the previous section are powerful and effective, some people find them confusing. Moreover, choosing the appropriate values can sometimes take more time than the scan you are trying to optimize. So Nmap offers a simpler approach, with six timing templates. You can specify them with the -T option and their number (0 - 5) or their name. The template names are paranoid (0), sneaky (1), polite (2), normal (3), aggressive (4), and insane (5). The first two are for IDS evasion. Polite mode slows down the scan to use less bandwidth and target machine resources. Normal mode is the default and so -T3 does nothing. Aggressive mode speeds scans up by making the assumption that you are on a reasonably fast and reliable network. Finally Insane mode assumes that you are on an extraordinarily fast network or are willing to sacrifice some accuracy for speed.

These templates allow the user to specify how aggressive they wish to be, while leaving Nmap to pick the exact timing values. The templates also make some minor speed adjustments for which fine grained control options do not currently exist. For example, -T4 prohibits the dynamic scan delay from exceeding 10ms for TCP ports and -T5 caps that value at 5 milliseconds. Templates can be used in combination with fine grained controls, as long as the template is specified first. Otherwise the standard values for the template may override the values you specify. I recommend using -T4 when scanning reasonably modern and reliable networks. Keep that option (at the beginning of the command line) even when you add fine grained controls so that you benefit from those extra minor optimizations that it enables.

If you are on a decent broadband or ethernet connection, I would recommend always using -T4. Some people love -T5 though it is too aggressive for my taste. People sometimes specify -T2 because they think it is less likely to crash hosts or because they consider themselves to

be polite in general. They often don't realize just how slow -T Polite really is. They scan may take ten times longer than a default scan. Machine crashes and bandwidth problems are rare with the default timing options (-T3) and so I normally recommend that for cautious scanners. Omitting version detection is far more effective than playing with timing values at reducing these problems.

While -T0 and -T1 may be useful for avoiding IDS alerts, they will take an extraordinarily long time to scan thousands of machines or ports. For such a long scan, you may prefer to set the exact timing values you need rather than rely on the canned -T0 and -T1 values.

The main effects of T0 are serializing the scan so only one port is scanned at a time, and waiting five minutes between sending each probe. T1 and T2 are similar but they only wait 15 seconds and 0.4 seconds, respectively, between probes. T3 is Nmap's default behavior, which includes parallelization. T4 does the equivalent of --max_rtt_timeout 1250 --initial_rtt_timeout 500 and sets the maximum TCP scan delay to 10 milliseconds. T5 does the equivalent of --max_rtt_timeout 300 --min_rtt_timeout 50 --initial_rtt_timeout 250 --host_timeout 900000 as well as setting the maximum TCP scan delay to 5ms.

Firewall/IDS Evasion and Spoofing

Many Internet pioneers envisioned a global open network with a universal IP address space allowing virtual connections between any two nodes. This allows hosts to act as true peers, serving and retrieving information from each other. People could access all of their home systems from work, changing the climate control settings or unlocking the doors for early guests. This vision of universal connectivity has been stifled by address space shortages and security concerns. In the early 1990s, organizations began deploying firewalls for the express purpose of reducing connectivity. Huge networks were cordoned off from the unfiltered Internet by application proxies, network address translation, and packet filters. The unrestricted flow of information gave way to tight regulation of approved communication channels and the content that passes over them.

Network obstructions such as firewalls can make mapping a network exceedingly difficult. It will not get any easier, as stifling casual reconnaissance is often a key goal of implementing the devices. Nevertheless, Nmap offers many features to help understand these complex networks, and to verify that filters are working as intended. It even supports mechanisms for bypassing poorly implemented defenses. One of the best methods of understanding your network security posture is to try to defeat it. Place yourself in the mindset of an attacker, and deploy techniques from this section against your networks. Launch an FTP bounce scan, Idle scan, fragmentation attack, or try to tunnel through one of your own proxies.

In addition to restricting network activity, companies are increasingly monitoring traffic with intrusion detection systems (IDS). All of the major IDSs ship with rules designed to detect Nmap scans because scans are sometimes a precursor to attacks. Many of these products have recently morphed into intrusion *prevention* systems (IPS) that actively block traffic deemed malicious. Unfortunately for network administrators and IDS vendors, reliably detecting bad intentions by analyzing packet data is a tough problem. Attackers with patience, skill, and the help of certain Nmap options can usually pass by IDSs undetected. Meanwhile, administrators must cope with large numbers of false positive results where innocent activity is misdiagnosed and alerted on or blocked.

Occasionally people suggest that Nmap should not offer features for evading firewall rules or sneaking past IDSs. They argue that these features are just as likely to be misused by attackers as used by administrators to enhance security. The problem with this logic is that these methods would

still be used by attackers, who would just find other tools or patch the functionality into Nmap. Meanwhile, administrators would find it that much harder to do their jobs. Deploying only modern, patched FTP servers is a far more powerful defense than trying to prevent the distribution of tools implementing the FTP bounce attack.

There is no magic bullet (or Nmap option) for detecting and subverting firewalls and IDS systems. It takes skill and experience. A tutorial is beyond the scope of this reference guide, which only lists the relevant options and describes what they do.

-f (fragment packets); --mtu (using the specified MTU)

The -f 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. Others can't enable this because fragments may take different routes into their networks. Some source systems defragment outgoing packets in the kernel. Linux with the iptables connection tracking module is one such example. Do a scan while a sniffer such as Ethereal is running to ensure that sent packets are fragmented. If your host OS is causing problems, try the --send_eth option to bypass the IP layer and send raw ethernet frames.

-D <decoy1 [,decoy2][,ME],...> (Cloak a scan with decoys)

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 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 for your real IP address. 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).

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). Decoys do not work with version detection or TCP connect() scan.

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, but many do not restrict spoofed IP packets at all.

-S <IP_Address> (Spoof source 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 the 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! The -e option would generally be required for this sort of usage, and -P0 would normally be advisable as well.

-e <interface> (Use specified interface)

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

--source_port <portnumber>; -g <portnumber> (Spoof source port number)

One surprisingly common misconfiguration is to trust traffic based only on the source port number. It is easy to understand how this comes about. An administrator will set up a shiny new firewall, only to be flooded with complains from ungrateful users whose applications stopped working. In particular, DNS may be broken because the UDP DNS replies from external servers can no longer enter the network. FTP is another common example. In active FTP transfers, the remote server tries to establish a connection back to the client to transfer the requested file.

Secure solutions to these problems exist, often in the form of application-level proxies or protocol-parsing firewall modules. Unfortunately there are also easier, insecure solutions. Noting that DNS replies come from port 53 and active ftp from port 20, many admins have fallen into the trap of simply allowing incoming traffic from those ports. They often assume that no attacker would notice and exploit such firewall holes. In other cases, admins consider this a short-term stop-gap measure until they can implement a more secure solution. Then they forget the security upgrade.

Overworked network administrators are not the only ones to fall into this trap. Numerous products have shipped with these insecure rules. Even Microsoft has been guilty. The IPsec filters that shipped with Windows 2000 and Windows XP contain an implicit rule that allows all TCP or UDP traffic from port 88 (Kerberos). In another well-known case, versions of the Zone Alarm personal firewall up to 2.1.25 allowed any incoming UDP packets with the source port 53 (DNS) or 67 (DHCP).

Nmap offers the -g and --source_port options (they are equivalent) to exploit these weaknesses. Simply provide a port number and Nmap will send packets from that port where possible. Nmap must use different port numbers for certain OS detection tests to work properly, and DNS requests ignore the --source_port flag because Nmap relies on system libraries to handle those. Most TCP scans, including SYN scan, support the option completely, as does UDP scan.

--data_length <number> (Append random data to sent packets)

Normally Nmap sends minimalist packets containing only 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 make a scan slightly less conspicuous.

--ttl <value> (Set IP time-to-live field)

Sets the IPv4 time-to-live field in sent packets to the given value.

--randomize_hosts (Randomize target host order)

Tells Nmap to shuffle each group of up to 8096 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. If you want to randomize over larger group sizes, increase PING_GROUP_SZ in nmap.h and recompile. An alternative solution is to generate the target IP list with a list scan (-sL -n -oN *filename*), randomize it with a Perl script, then provide the whole list to Nmap with -iL.

--spoof_mac <mac address, prefix, or vendor name> (Spoof MAC address)

Asks Nmap to use the given MAC address for all of the raw ethernet frames it sends. This option implies --send_eth to ensure that Nmap actually sends ethernet-level packets. 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 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.

Output

Any security tools is only as useful as the output it generates. Complex tests and algorithms are of little value if they aren't presented in an organized and comprehensible fashion. Given the number of ways Nmap is used by people and other software, no single format can please everyone. So Nmap offers several formats, including the interactive mode for humans to read directly and XML for easy parsing by software.

In addition to offering different output formats, Nmap provides options for controlling the verbosity of output as well as debugging messages. Output types may be sent to standard output or to named files, which Nmap can append to or clobber. Output files may also be used to resume aborted scans.

Nmap makes output available in five different formats. The default is called interactive output, and it is sent to standard output (stdout). There is also normal output, which is similar to interactive except that it displays less runtime information and warnings since it is expected to be analyzed after the scan completes rather than interactively.

XML output is one of the most important output types, as it can be converted to HTML, easily parsed by programs such as Nmap graphical user interfaces, or imported into databases.

The two remaining output types are the simple grepable output which includes most information for a target host on a single line, and sCRiPt KiDDi3 0utPUt for users who consider themselves |<-r4d.

While interactive output is the default and has no associated command-line options, the other four format options use the same syntax. They take one argument, which is the filename that results should be stored in. Multiple formats may be specified, but each format may only be specified once. For example, you may wish to save normal output for your own review while saving XML of the same scan for programmatic analysis. You might do this with the options -oX myscan.xml -oN myscan.nmap. While this chapter uses the simple names like myscan.xml for brevity, more descriptive names are generally recommended. The names chosen are a matter of personal preference, though I use long ones that incorporate the scan date and a word or two describing the scan, placed in a directory named after the company I'm scanning.

While these options save results to files, Nmap still prints interactive output to stdout as usual. For example, the command **nmap -oX myscan.xml target** prints XML to myscan.xml and fills standard output with the same interactive results it would have printed if -oX wasn't specified at all. You can change this by passing a hyphen character as the argument to one of the format types. This causes Nmap to deactivate interactive output, and instead print results in the format you specified to the standard output stream. So the command **nmap -oX - target** will send only XML output to stdout. Serious errors may still be printed to the normal error stream, stderr.

Unlike some Nmap arguments, the space between the logfile option flag (such as -oX) and the filename or hyphen is mandatory. If you omit the flags and give arguments such as -oG- or -oXscan.xml, a backwards compatibility feature of Nmap will cause the creation of *normal format* output files named G- and Xscan.xml respectively.

Nmap also offers options to control scan verbosity and to append to output files rather than clobbering them. All of these options are described belowe.

Nmap Output Formats

-oN <filespec> (Normal output)

> Requests that normal output be directed to the given filename. As discussed above, this differs slightly from interactive output.

-oX <filespec> (XML output)

Requests that XML output be directed to the given filename. Nmap includes a document type definition (DTD) which allows XML parsers to validate Nmap XML output. While it is primarily intended for programmatic use, it can also help humans interpret Nmap XML output. The DTD defines the legal elements of the format, and often enumerates the attributes and values they can take on. The latest version is always available from http://www.insecure.org/nmap/data/nmap.dtd.

XML offers a stable format that is easily parsed by software. Free XML parsers are available for all major computer languages, including C/C++, Perl, Python, and Java. People have even written bindings for most of these languages to handle Nmap output and execution specifically. Examples are Nmap::Scanner and Nmap::Parser in Perl CPAN. In almost all cases that a non-trivial application interfaces with Nmap, XML is the preferred format.

The XML output references an XSL stylesheet which can be used to format the results as HTML. The easiest way to use this is simply to load the XML output in a web browser such as Firefox or IE. By default, this will only work on the machine you ran Nmap on (or a similarly configured one) due to the hard-coded nmap.xsl filesystem path. See the --stylesheet option for a way to create a portable XML file that renders as HTML on any web-connected machine.

-oS <filespec> (ScRipT KIdd|3 oUTpuT)

Script kiddie output is like interactive output, except that it is post-processed to better suit the 'l33t HaXXorZ who previously looked down on Nmap due to its consistent capitalization and spelling. Humor impaired people should note that this option is making fun of the script kiddies before flaming me for supposedly “helping them”.

-oG <filespec> (Grepable output)

This output format is covered last because it is deprecated. The XML output format is far more powerful, and is nearly as convenient for experienced users. XML is a standard for which dozens of excellent parsers are available, while grepable output is my own simple hack. XML is extensible to support new Nmap features as they are released, while I often must omit those features from grepable output for lack of a place to put them.

Nevertheless, grepable output is still quite popular. It is a simple format that lists each host on one line and can be trivially searched and parsed with standard UNIX tools such as grep, awk, cut, sed, diff, and Perl. Even I usually use it for one-off tests done at the command line. Finding all the hosts with the ssh port open or that are running Solaris takes only a simple grep to identify the hosts, piped to an awk or cut command to print the desired fields.

Grepable output consists of comments (lines starting with a pound (#)) and target lines. A target line includes a combination of 6 labeled fields, separated by tabs and followed with a colon. The fields are Host, Ports, Protocols, Ignored State, OS, Seq Index, IPID, and Status.

The most important of these fields is generally Ports, which gives details on each interesting port. It is a comma separated list of port entries. Each port entry represents one interesting port, and takes the form of seven slash (/) separated subfields. Those subfields are: Port number, State, Protocol, Owner, Service, SunRPC info, and Version info.

As with XML output, this man page does not allow for documenting the entire format. A more detailed look at the Nmap grepable output format is available from http://www.unspecific.com/nmap-oG-output.

-oA <basename> (Output to all formats)

As a convenience, you may specify -oA *basename* to store scan results in normal, XML, and grepable formats at once. They are stored in *basename*.nmap, *basename*.xml, and *basename*.gnmap, respectively. As with most programs, you can prefix the filenames with a directory path, such as ~/nmaplogs/foocorp/ on UNIX or c:\hacking\sco on Windows.

Verbosity and debugging options

-v (Increase verbosity level)

Increases the verbosity level, causing Nmap to print more information about the scan in progress. Open ports are shown as they are found and completion time estimates are provided when Nmap thinks a scan will take more than a few minutes. Use it twice for even greater verbosity. Using it more than twice has no effect.

Most changes only affect interactive output, and some also affect normal and script kiddie output. The other output types are meant to be processed by machines, so Nmap can give substantial detail by default in those formats without fatiguing a human user. However, there are a few changes in other modes where output size can be reduced substantially by omitting some detail. For example, a comment line in the grepable output that provides a list of all ports scanned is only printed in verbose mode because it can be quite long.

-d [level] (Increase or set debugging level)

When even verbose mode doesn't provide sufficient data for you, debugging is available to flood you with much more! As with the verbosity option (-v), debugging is enabled with a command-line flag (-d) and the debug level can be increased by specifying it multiple times. Alternatively, you can set a debug level by giving an argument to -d. For example, -d9 sets level nine. That is the highest effective level and will produce thousands of lines unless you run a very simple scan with very few ports and targets.

Debugging output is useful when a bug is suspected in Nmap, or if you are simply confused as to what Nmap is doing and why. As this feature is mostly intended for developers, debug lines aren't always self-explanatory. You may get something like: Timeout vals: srtt: -1 rttvar: -1 to: 1000000 delta 14987 ==> srtt: 14987 rttvar: 14987 to: 100000. If you don't understand a line, your only recourses are to ignore it, look it up in the source code, or request help from the development list (nmap-dev). Some lines are self explanatory, but the messages become more obscure as the debug level is increased.

--packet_trace (Trace packets and data sent and received)

Causes Nmap to print a summary of every packet sent or received. This is often used for debugging, but is also a valuable way for new users to understand exactly what Nmap is doing under the covers. To avoid printing thousands of lines, you may want to specify a limited

number of ports to scan, such as -p20-30. If you only care about the goings on of the version detection subsystem, use --version_trace instead.

--iflist (List interfaces and routes)

Prints the interface list and system routes as detected by Nmap. This is useful for debugging routing problems or device mischaracterization (such as Nmap treating a PPP connection as Ethernet).

Miscellaneous output options

--append_output (Append to rather than clobber output files)

When you specify a filename to an output format flag such as -oX or -oN, that file is overwritten by default. If you prefer to keep the existing content of the file and append the new results, specify the --append_output option. All output filenames specified in that Nmap execution will then be appended to rather than clobbered. This doesn't work well for XML (-oX) scan data as the resultant file generally won't parse properly until you fix it up by hand.

--resume <filename> (Resume aborted scan)

Some extensive Nmap runs take a very long time -- on the order of days. Such scans don't always run to completion. Restrictions may prevent Nmap from being run during working hours, the network could go down, the machine Nmap is running on might suffer a planned or unplanned reboot, or Nmap itself could crash. The admin running Nmap could cancel it for any other reason as well, by pressing **ctrl-C**. Restarting the whole scan from the beginning may be undesirable. Fortunately, if normal (-oN) or grepable (-oG) logs were kept, the user can ask Nmap to resume scanning with the target it was working on when execution ceased. Simply specify the --resume option and pass the normal/grepable output file as its argument. No other arguments are permitted, as Nmap parses the output file to use the same ones specified previously. Simply call Nmap as **nmap --resume *logfilename***. Nmap will append new results to the data files specified in the previous execution. Resumption does not support the XML output format because combining the two runs into one valid XML file would be difficult.

--stylesheet <path or URL> (Set XSL stylesheet to transform XML output)

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 location of nmap.xsl is used by default for privacy reasons.

--no_stylesheet (Omit XSL stylesheet declaration from XML)

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

Miscellaneous Options

This section describes some important (and not-so-important) options that don't really fit anywhere else.

-6 (Enable IPv6 scanning)

Since 2002, Nmap has offered IPv6 support for its most popular features. In particular, ping scanning (TCP-only), connect() scanning, and version detection all support IPv6. The command syntax is the same as usual except that you also add the -6 option. Of course, you must use IPv6 syntax if you specify an address rather than a hostname. An address might look like 3ffe:7501:4819:2000:210:f3ff:fe03:14d0, so hostnames are recommended. The output looks the same as usual, with the IPv6 address on the “interesting ports” line being the only IPv6 give away.

While IPv6 hasn't exactly taken the world by storm, it gets significant use in some (usually Asian) countries and most modern operating systems support it. To use Nmap with IPv6, both the source and target of your scan must be configured for IPv6. If your ISP (like most of them) does not allocate IPv6 addresses to you, free tunnel brokers are widely available and work fine with Nmap. One of the better ones is run by BT Exact at https://tb.ipv6.btexact.com/. I have also used one that Hurricane Electric provides at http://ipv6tb.he.net/. 6to4 tunnels are another popular, free approach.

-A (Aggressive scan options)

This option enables additional advanced and aggressive 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 want as well.

--datadir <directoryname> (Specify custom Nmap data file location)

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 specified with the --datadir option (if any). 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.

--send_eth (Use raw ethernet sending)

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 are required for Windows operation since Microsoft disabled raw socket support. Nmap still uses raw IP packets on UNIX despite this option when there is no other choice (such as non-ethernet connections).

--send_ip (Send at raw IP level)

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.

--privileged (Assume that the user is fully 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 quits 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 to --privileged.

--interactive (Start in interactive mode)

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 as 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 **h** for help. This option is rarely used because proper shells are usually more familiar and feature-complete. This option includes a bang (!) operator for executing shell commands, which is one of many reasons not to install Nmap setuid root.

-V; --version (Print version number)

Prints the Nmap version number and exits.

-h; --help (Print help summary page)

Prints a short help screen with the most common command flags. Running Nmap without any arguments does the same thing.

Runtime Interaction

This feature does not yet exist in Nmap. I need to either add it or remove this section

During the execution of nmap, all key presses are captured. This allows you to interact with the program without aborting and restarting it. Certain special keys will change options, while any other keys will print out a status message telling you about the scan. The convention is that *lowercase letters increase* the amount of printing, and *uppercase letters decrease* the printing.

v / V

Increase / Decrease the Verbosity

d / D

Increase / Decrease the Debugging Level

p / P

Turn on / off Packet Tracing

Anything else

Print out a status message like this:

Stats: 0:00:08 elapsed; 111 hosts completed (5 up), 5 undergoing Service Scan

Service scan Timing: About 28.00% done; ETC: 16:18 (0:00:15 remaining)

Examples

Here are some Nmap usage examples, from the simple and routine to a little more complex and esoteric. Some actual IP addresses and domain names are used to make things more concrete. In their place you should substitute addresses/names from *your own network.*. While I don't think port scanning other networks is or should be illegal, some network administrators don't appreciate unsolicited scanning of their networks and may complain. Getting permission first is the best approach.

For testing purposes, you have permission to scan the host scanme.nmap.org. This permission only includes scanning via Nmap and not testing exploits or denial of service attacks. To conserve bandwidth, please do not initiate more than a dozen scans against that host per day. If this free scanning target service is abused, it will be taken down and Nmap will report Failed to resolve given hostname/IP: scanme.nmap.org. These permissions also apply to the hosts scanme2.nmap.org, scanme3.nmap.org, and so on, though those hosts do not currently exist.

nmap -v scanme.nmap.org

This option scans all reserved TCP ports on the machine scanme.nmap.org . The -v option enables verbose mode.

nmap -sS -O scanme.nmap.org/24

Launches a stealth SYN scan against each machine that is up out of the 255 machines on “class C” network where Scanme 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 OS detection.

nmap -sV -p 22,53,110,143,4564 198.116.0-255.1-127

Launches host enumeration and a TCP scan at the first half of each of the 255 possible 8 bit subnets in the 198.116 class B address space. This tests whether the systems run sshd, DNS, pop3d, imapd,

or port 4564. For any of these ports found open, version detection is used to determine what application is running.

nmap -v -iR 100000 -P0 -p 80

Asks Nmap to choose 100,000 hosts at random and scan them for web servers (port 80). Host enumeration is disabled with -P0 since first sending a couple probes to determine whether a host is up is wasteful when you are only probing one port on each target host anyway.

nmap -P0 -p80 -oX logs/pb-port80scan.xml -oG logs/pb-port80scan.gnmap 216.163.128.20/20

This scans 4096 IPs for any webservers (without pinging them) and saves the output in grepable and XML formats.

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 -- other systems have different commands for performing a zone transfer.

Bugs

Like its author, Nmap isn't perfect. But you can help make it better by sending bug reports or even writing patches. If Nmap doesn't behave the way you expect, first upgrade to the latest version available from http://www.insecure.org/nmap/. If the problem persists, do some research to determine whether it has already been discovered and addressed. Try Googling the error message or browsing the Nmap-dev archives at http://seclists.org/. Read this full munaual page as well. If nothing comes of this, mail a bug report to <nmap-dev@insecure.org>. Please include everything you have learned about the problem, as well as what version of Nmap you are running and what operating system version it is running on. Problem reports and Nmap usage questions sent to nmap-dev@insecure.org are far more likely to be answered than those sent to Fyodor directly.

Code patches to fix bugs are even better than bug reports. Basic instructions for creating patch files with your changes are available at http://www.insecure.org/nmap/data/HACKING. Patches may be sent to nmap-dev (recommended) or to Fyodor directly.

Author

Fyodor <fyodor@insecure.org> (http://www.insecure.org)

Hundreds of people have made valuable contributions to Nmap over the years. These are detailed in the CHANGELOG file which is distributed with Nmap and also available from http://www.insecure.org/nmap/nmap_changelog.html.

Legal Notices (Copyright, Licensing, (lack of) Warranty, Export Control)

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

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 of 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 (e.g. suid root) for security reasons.

This product includes software developed by the Apache Software Foundation. A modified version of the Libpcap portable packet capture library is distributed along with nmap. The Windows version of Nmap utilized the libpcap-derived WinPcap library instead. Regular expression support is provided by the PCRE library, which is open source software, written by Philip Hazel. Certain raw networking functions use the Libdnet networking library, which was written by Dug Song. A modified version is distributed with Nmap. Nmap can optionally link with the OpenSSL cryptography toolkit for SSL version detection support. All of the third-party software described in this paragraph is freely redistributable under BSD-style software licenses.

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.