NMAP.ORG
Npcap.com Seclists.org Sectools.org Insecure.org
Site Search
Download | Reference Guide | Book | Docs | Zenmap GUI | In the Movies
Nmap Network Scanning / Chapter 15. Nmap Reference Guide / Port Scanning Techniques
◀ Prev Next ▶

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 Npcap 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`) and any one of the SCTP scan types (`-sY`, `-sZ`) 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). 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 restrictive firewalls. It is also relatively unobtrusive and stealthy since it never completes TCP connections. SYN scan 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 0, 1, 2, 3, 9, 10, or 13) is received. The port is also considered open if a SYN packet (without the ACK flag) is received in response. This can be due to an extremely rare TCP feature known as a simultaneous open or split handshake connection (see `https://nmap.org/misc/split-handshake.pdf`).

`-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. 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 a UDP packet to every targeted port. For some common ports such as 53 and 161, a protocol-specific payload is sent to increase response rate, but for most ports the packet is empty unless the `--data`, `--data-string`, or `--data-length` options are specified. If an ICMP port unreachable error (type 3, code 3) is returned, the port is `closed`. Other ICMP unreachable errors (type 3, codes 0, 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. Version detection (`-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.

`-sY` (SCTP INIT scan)

> SCTP is a relatively new alternative to the TCP and UDP protocols, combining most characteristics of TCP and UDP, and also adding new features like multi-homing and multi-streaming. It is mostly being used for SS7/SIGTRAN related services but has the potential to be used for other applications as well. SCTP INIT scan is the SCTP equivalent of a TCP SYN scan. It can be performed quickly, scanning thousands of ports per second on a fast network not hampered by restrictive firewalls. Like SYN scan, INIT scan is relatively unobtrusive and stealthy, since it never completes SCTP associations. 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 SCTP association. You send an INIT chunk, as if you are going to open a real association and then wait for a response. An INIT-ACK chunk indicates the port is listening (open), while an ABORT chunk 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 0, 1, 2, 3, 9, 10, or 13) is received.

`-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 of RFC 793 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 0, 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 0, 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 1,000 open ports and three 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.

`-sZ` (SCTP COOKIE ECHO scan)

> SCTP COOKIE ECHO scan is a more advanced SCTP scan. It takes advantage of the fact that SCTP implementations should silently drop packets containing COOKIE ECHO chunks on open ports, but send an ABORT if the port is closed. The advantage of this scan type is that it is not as obvious a port scan than an INIT scan. Also, there may be non-stateful firewall rulesets blocking INIT chunks, but not COOKIE ECHO chunks. Don't be fooled into thinking that this will make a port scan invisible; a good IDS will be able to detect SCTP COOKIE ECHO scans too. The downside is that SCTP COOKIE ECHO scans cannot differentiate between `open` and `filtered` ports, leaving you with the state `open|filtered` in both cases.

`-sI <zombie host>[:<probeport>]` (idle scan)

> 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). Full details of this fascinating scan type are in the section called “TCP Idle Scan (`-sI`)”.
>
> 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 IP ID 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 *announce* mailing list (then called *nmap-hackers*). 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 eight-bit IP protocol field. The headers are usually empty, containing no data and not even the proper header for the claimed protocol. The exceptions are TCP, UDP, ICMP, SCTP, and IGMP. 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` while port unreachable (type 3, code 3) marks the protocol `open`. Other ICMP unreachable errors (type 3, code 0, 1, 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 port 21 (or even for any FTP services if you scan all ports with version detection) and use the `ftp-bounce` NSE script. 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.

◀ Prev ▲ Up Home Next ▶
Port Scanning Basics Chapter 15. Nmap Reference Guide Port Specification and Scan Order

Site Search

Nmap Security Scanner: Ref Guide, Install Guide, Docs, Download, Nmap OEM
Npcap packet capture: User's Guide, API docs, Download, Npcap OEM
Security Lists: Nmap Announce, Nmap Dev, Full Disclosure, Open Source Security, BreachExchange
Security Tools: Vuln scanners, Password audit, Web scanners, Wireless, Exploitation
About: About/Contact, Privacy, Advertising, Nmap Public Source License