Nmap Network Scanning

Chapter 4. Port Scanning Overview

Chapter 4. Port Scanning Overview

Table of Contents

Introduction to Port Scanning

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 the most commonly used 1,000 TCP ports on the host `<target>`, classifying each port into the state `open`, `closed`, `filtered`, `unfiltered`, `open|filtered`, or `closed|filtered`.

What Exactly is a Port?

Ports are simply a software abstraction, used to distinguish between communication channels. Similar to the way IP addresses are used to identify machines on networks, ports identify specific applications in use on a single machine. For example, your web browser will by default connect to TCP port 80 of machines in HTTP URLs. If you specify the secure HTTPS protocol instead, the browser will try port 443 by default.

Nmap works with two protocols that use ports: TCP and UDP. A connection for each protocol is uniquely identified by four elements: source and destination IP addresses and corresponding source and destination ports. All of these elements are simply numbers placed in the headers of each packet sent between hosts. The protocol is an eight-bit field, which specifies what type of packet is contained in the IP data (payload) section. For example, TCP is protocol number six, and UDP is 17. IPv4 addresses have a length of 32-bits, while ports are 16-bits long. IPv6 addresses are 128-bits in length. Further IP, TCP, and UDP header layout details can be found in the section called "TCP/IP Reference".

Because most popular services are registered to a well-known port number, one can often guess what services open ports represent. Nmap includes an `nmap-services` file, containing the well-known service for registered port and protocol numbers, as well as common ports for trojan backdoors and other applications that don't bother registering with the Internet Assigned Numbers Authority (IANA). Nmap prints this service name for reference along with the port number.

Because the port number field is 16-bits wide, values can reach 65,535. The lowest possible value, zero, is invalid. The Berkeley sockets API, which defines how programs are usually written for network communication, does not allow port zero to be used as such. Instead, it interprets a port zero request as a wildcard, meaning that the programmer does not care which is used. The system then chooses an available port number. For example, programmers rarely care what source port number is used for an outgoing connection. So they set it to zero and let the operating system choose one.

While port zero is invalid, nothing stops someone from specifying it in the header field. Some malicious trojan backdoors listen on port zero of compromised systems as a stealthy way to offer illegitimate access without appearing on most port scans. To combat this, Nmap does allow scanning of port zero when it is specified explicitly (e.g. `-p0-65535`).

The first class of valid ports, numbers one through 1,023, are known as reserved ports. Unix systems (unlike Windows) require that applications have special (root) privileges in order to bind to and listen on these ports. The idea is to allow remote users to trust that they are connecting to a valid service started by an administrator and not by some wicked, unprivileged user. If the registered port for SSH was 2,222 instead of 22, a malicious user could start up a rogue SSH daemon on that port, collecting passwords from anyone who connects. As most common server applications listen on reserved ports, these are often the most fruitful to scan.

The ephemeral port range is another class of ports. This pool of ports is made available by the system for allocation as needed. When an application specifies port zero (meaning "any port"), the system chooses a port from this range. The range varies by operating system, and is usually configurable. It should contain at least a couple thousand ports to avoid running out when many concurrent connections are open. The Nmap connect scan can use hundreds at a time as it scans every specified port on each target machine. On Linux, you can view or set the range using the file `/proc/sys/net/ipv4/ip_local_port_range`. Example 4.1 shows that on my Linux system, the range is 32,768 to 61,000. Such a large range should be sufficient in almost all cases, but I expand it just to demonstrate how to do so.

Example 4.1. Viewing and increasing the ephemeral port range on Linux

```
felix/# cat /proc/sys/net/ipv4/ip_local_port_range
32768   61000
felix/# echo "10000 65000" > /proc/sys/net/ipv4/ip_local_port_range
felix/# cat /proc/sys/net/ipv4/ip_local_port_range
10000   65000
felix/#
```

SunRPC ports are often found in the ephemeral range. Other applications open ephemeral ports temporarily for a file transfer or other event. FTP clients often do this when requesting an active mode transfer. Some P2P and instant messaging clients do so as well.

The IANA has their own port classification scheme, which differs slightly from the vernacular of this book. Their authoritative port list at http://www.iana.org/assignments/port-numbers divides the space into the following three classes:

well-known ports

> These are reserved ports (within the range of 1 to 1,023, as discussed above) which have been registered with the IANA for a certain service. Familiar examples are ports 22, 25, and 80 for the services SSH, SMTP, and HTTP, respectively.

registered ports

> These ports fall within the range 1,024 to 49,151 and have been registered with the IANA in the same way the well known ports have. Most of these are not as commonly used as the well-known ports. The key difference is that unprivileged users can bind to these ports and thus run the services on their registered port. Users cannot do so on most platforms for well-known ports, since they reside in the reserved port range.

dynamic and/or private ports

> The IANA reserves the port numbers from 49152 through 65535 for dynamic uses such as those discussed in the ephemeral ports section. Proprietary services that are only used within a company may also use these ports.

When this book mentions registered or well-known ports without any reference to the IANA, it usually means ports registered with Nmap in the `nmap-services` file, regardless of whether they fall in the reserved port range.

Nmap's port registration file (`nmap-services`) contains empirical data about how frequently each TCP or UDP port is found to be open. By default, Nmap scans the 1,000 most popular ports of each protocol it is asked to scan. There are many options for specifying an alternate set of ports (by frequency or by listing them explicitly), as described in the section called “Selecting Ports to Scan”.

What Are the Most Popular Ports?

I spent the Summer of 2008 scanning tens of millions of Internet hosts and collecting data from enterprises to determine how frequently each port number is found open. It is important to be familiar with the most common service ports, and also interesting to see which ones made the list. The following two lists provide the top TCP and UDP ports as determined by our empirical scan data. The listed service is the one found in our `nmap-services` file. We try to list the most common service for each port there, though of course it is possible for a port to be used for different things.

Top 20 (most commonly open) TCP ports

1. Port 80 (HTTP)—If you don't even know this service, you're reading the wrong book. This accounted for more than 14% of the open ports we discovered.
2. Port 23 (Telnet)—Telnet lives on (particularly as an administration port on devices such as routers and smart switches) even though it is insecure (unencrypted).
3. Port 443 (HTTPS)—SSL-encrypted web servers use this port by default.
4. Port 21 (FTP)—FTP, like Telnet, is another insecure protocol which should die. Even with anonymous FTP (avoiding the authentication sniffing worry), data transfer is still subject to tampering.
5. Port 22 (SSH)—Secure Shell, an encrypted replacement for Telnet (and, in some cases, FTP).
6. Port 25 (SMTP)—Simple Mail Transfer Protocol (also insecure).
7. Port 3389 (ms-term-server)—Microsoft Terminal Services administration port.
8. Port 110 (POP3)—Post Office Protocol version 3 for email retrieval (insecure).
9. Port 445 (Microsoft-DS)—For SMB communication over IP with MS Windows services (such as file/printer sharing).
10. Port 139 (NetBIOS-SSN)—NetBIOS Session Service for communication with MS Windows services (such as file/printer sharing). This has been supported on Windows machines longer than 445 has.
11. Port 143 (IMAP)—Internet Message Access Protocol version 2. An insecure email retrieval protocol.
12. Port 53 (Domain)—Domain Name System (DNS), an insecure system for conversion between host/domain names and IP addresses.
13. Port 135 (MSRPC)—Another common port for MS Windows services.
14. Port 3306 (MySQL)—For communication with MySQL databases.
15. Port 8080 (HTTP-Proxy)—Commonly used for HTTP proxies or as an alternate port for normal web servers (e.g. when another server is already listening on port 80, or when run by unprivileged UNIX users who can only bind to high ports).
16. Port 1723 (PPTP)—Point-to-point tunneling protocol (a method of implementing VPNs which is often required for broadband connections to ISPs).
17. Port 111 (RPCBind)—Maps SunRPC program numbers to their current TCP or UDP port numbers.
18. Port 995 (POP3S)—POP3 with SSL added for security.
19. Port 993 (IMAPS)—IMAPv2 with SSL added for security.
20. Port 5900 (VNC)—A graphical desktop sharing system (insecure).

Top 20 (most commonly open) UDP ports

1. Port 631 (IPP)—Internet Printing Protocol.
2. Port 161 (SNMP)—Simple Network Management Protocol.
3. Port 137 (NETBIOS-NS)—One of many UDP ports for Windows services such as file and printer sharing.
4. Port 123 (NTP)—Network Time Protocol.
5. Port 138 (NETBIOS-DGM)—Another Windows service.
6. Port 1434 (MS-SQL-DS)—Microsoft SQL Server.
7. Port 445 (Microsoft-DS)—Another Windows Services port.
8. Port 135 (MSRPC)—Yet Another Windows Services port.
9. Port 67 (DHCPS)—Dynamic Host Configuration Protocol Server (gives out IP addresses to clients when they join the network).
10. Port 53 (Domain)—Domain Name System (DNS) server.
11. Port 139 (NETBIOS-SSN)—Another Windows Services port.
12. Port 500 (ISAKMP)—The Internet Security Association and Key Management Protocol is used to set up IPsec VPNs.
13. Port 68 (DHCPC)—DHCP client port.
14. Port 520 (Route)—Routing Information Protocol (RIP).
15. Port 1900 (UPNP)—Microsoft Simple Service Discovery Protocol, which enables discovery of Universal plug-and-play devices.
16. Port 4500 (nat-t-ike)—For negotiating Network Address Translation traversal while initiating IPsec connections (during Internet Key Exchange).
17. Port 514 (Syslog)—The standard UNIX log daemon.
18. Port 49152 (Varies)—The first of the IANA-specified dynamic/private ports. No official ports may be registered from here up until the end of the port range (65536). Some systems use this range for their ephemeral ports, so services which bind a port without requesting a specific number are often allocated 49152 if they are the first program to do so.
19. Port 162 (SNMPTrap)—Simple Network Management Protocol trap port (An SNMP agent typically uses 161 while an SNMP manager typically uses 162).
20. Port 69 (TFTP)—Trivial File Transfer Protocol.

What is Port Scanning?

Port scanning is the act of remotely testing numerous ports to determine what state they are in. The most interesting state is usually open, meaning that an application is listening and accepting connections on the port. Many techniques are available for conducting such a scan. Chapter 5, *Port Scanning Techniques and Algorithms* explains the circumstances under which each is most appropriate.

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. 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. Before you get too excited about an open port, note that it is possible that the application is protected with a TCP wrapper (tcpd) or that the application itself is configured to only service approved client IP addresses. Such cases still leave more attack surface than a closed port.

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 online and using an IP address (host discovery, or ping scanning), and as part of OS detection. Because closed ports are reachable, they may be worth scanning later in case some open up. Administrators may want to consider blocking such ports with a firewall so they 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 sort of filtering slows scans down 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 IP ID Idle scan discussed in the section called “TCP Idle Scan (-sI)”.

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 relevant sections of Chapter 5, *Port Scanning Techniques and Algorithms*.

Why Scan Ports?

Port scanning is not only performed for fun and amusement. There are numerous practical benefits to regularly scanning your networks. Foremost among these is security. One of the central tenets of network security is that reducing the number and complexity of services offered reduces the opportunity for attackers to break in. Most remote network compromises come from exploiting a server application listening on a TCP or UDP port. In many cases, the exploited application is not even used by the targeted organization, but was enabled by default when the machine was set up. Had that service been disabled, or protected by a firewall, the attack would have been thwarted.

Realizing that every open port is an opportunity for compromise, attackers regularly scan targets, taking an inventory of all open ports. They compare this list of listening services with their list of favorite exploits for vulnerable software. It takes just one match to compromise a machine, creating a foothold that is often used to infest the whole network. Attackers who are less discriminate about who they target will often scan for just the default port of an exploitable application. This is much faster than scanning every port, though the service will be missed when running on a non-default port. Such attackers are often derided as “script kiddies”, because they often know little more about security than how to run an exploit script written by someone more skilled. Across many organizations, such attackers are bound to find vulnerable hosts. They can be quite a nuisance, though their sheer numbers and relentless pounding against Internet-accessible machines often drive people to patch systems quickly. This reduces the likelihood of more serious, targeted attacks succeeding.

An important defense against these crackers is for systems administrators to scan their own networks regularly with tools such as Nmap. Take the list of open ports, and shut down any services that aren't used. Ensure that those which must remain available are fully patched and that you are on the vendor's security notification list. Firewall rules should be added where possible, limiting access to only legitimate users. Hardening instructions are available on the Web for most popular applications, reducing the cracker's opportunity even further. Nmap cannot do most of this for you, but it creates the list of available services to start out with. Some administrators try to use **netstat** instead, but that doesn't scale well. It requires access to every machine, and some mobile machines are easy to miss. Plus, you can't run **netstat** on your average wireless access point, VoIP phone, or printer. In addition, there is always the risk that a compromised machine will have a trojaned **netstat** which gives out false information. Most of the modern rootkits installed by attackers include this functionality. Relying solely on Nmap is a mistake too. A combination of careful design, configuration auditing, and regular scanning is well advised.

While security is the most common reason for port scanning, administrators often find that it suits other purposes as well. Creating an inventory of machines and the services they offer can be useful for asset tracking, network design, policy compliance checks, software license tracking, availability testing, network debugging, and more.

Host Discovery Code Algorithms

A Quick Port Scanning Tutorial

[Nmap | Sec Tools | Mailing Lists | Site News | About/Contact | Advertising | Privacy]

Custom Search