Assignment Number: 8

Name: Divya Shah; Branch: I.T (T.E.); Roll Number: 115 24/08/2023

**Aim:** Installation of NMAP and using it with different options to scan open ports, perform OS fingerprinting, ping scan, TCP port scan, UDP port scan, etc.

**LO mapped:** LO4

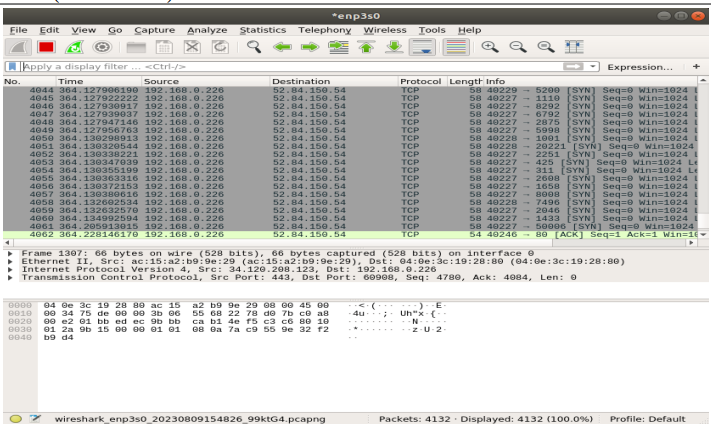
**Theory:**

1. Nmap -sP <IP address (192.168.0.\*)>

|  |
| --- |
|  |

1. -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



1. -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.

|  |
| --- |
|  |

1. -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 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

|  |
| --- |
|  |

1. -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:

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

Sets the FIN, PSH, and URG flags, lighting the packet up like a Christmas tree. These three scan types are exactly the same in behaviour 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.

1. -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 –scan flags). 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 labelled filtered

|  |
| --- |
|  |

1. -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.

|  |
| --- |
|  |

1. -O (Enable 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, IP ID sampling, and the initial window size check, Nmap compares the results to its nmap-os-db database of more than 2,600 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's name (e.g. Sun), underlying OS (e.g. Solaris), OS generation (e.g. 10), and device type (general purpose, router, switch, game console, etc). Most fingerprints also have a Common Platform Enumeration (CPE) representation, like cpe:/o:linux:linux\_kernel:2.6. 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 some other tests which make use of information that is gathered during the process anyway. One of these 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, IP ID 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

|  |
| --- |
|  |

1. nmap -sP 192.168.0.\*

|  |
| --- |
|  |

**Conclusion:** We implemented Installation of NMAP and using it with different options to scan open ports, perform OS fingerprinting, ping scan, TCP port scan, UDP port scan, etc.