

1.1 Introduction:

Port Scanning is one of the most popular techniques attackers use to discover services that they can exploit to break into systems. All systems that are connected to a LAN or the Internet via a modem run services that listen to well-known and not so well-known ports. By port scanning, the attacker can find the following information about the targeted systems: what services are running, what users own those services, whether anonymous logins are supported, and whether certain network services require authentication. Port scanning is accomplished by sending a message to each port, one at a time. The kind of response received indicates whether the port is used and can be probed for further weaknesses. Port scanners are important to network security technicians because they can reveal possible security vulnerabilities on the targeted system.

Just as port scans can be ran against your systems, port scans can be detected and the amount of information about open services can be limited utilizing the proper tools. Every publicly available system has ports that are open and available for use. The object is to limit the exposure of open ports to authorized users and to deny access to the closed ports. [3]

1.2 APPROACHES TO PORT SCANNING:

Based on how scanning is performed, port scan techniques can be classified into three broad categories: *single-source* port scans and *distributed* port scans and distributed many 2 many port scan, Each of these categories is illustrated in *Figure 1.1*

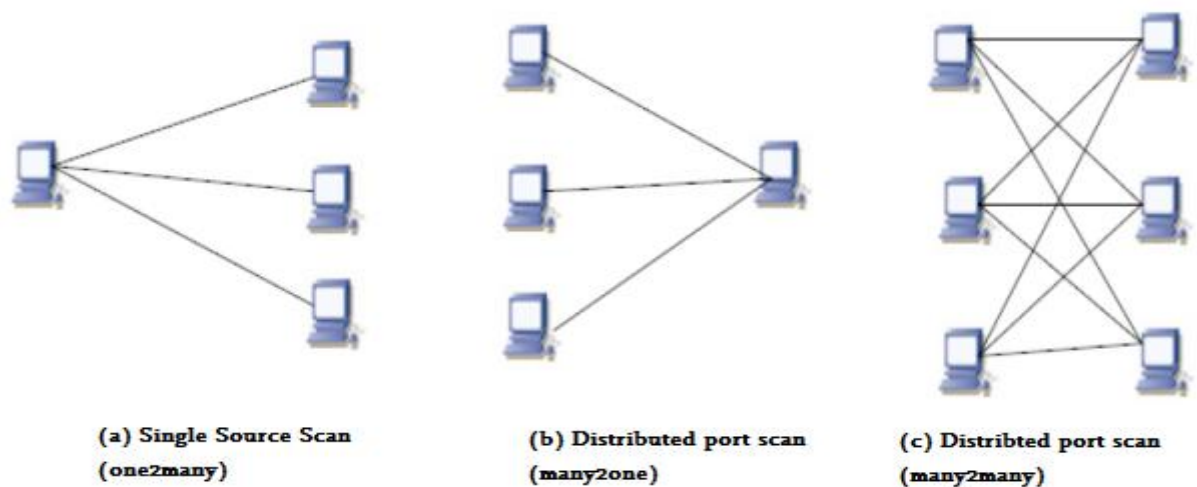


Figure 1.1 Scan categories

1.2.1 Single-source Port Scans:

The goal of port scanning from the perspective of an attacker is to gather ideas regarding where to probe for weaknesses. One can scan the network in a *one-to-many* fashion. A scan or any network attack can be detected by using NIDS. In a pattern recognition based scheme, attacks are discovered by matching network traffic with some known patterns. In the literature, a port scanner is defined as consisting of “specialized programs used to determine what TCP ports of a host have processes listening on them for possible connections”. Staniford et al further define scan footprint as the set of ports or IP combinations that the attacker is interested in characterizing. According to them, port scans can be of four types (shown in **Figure 1.2**): *vertical*, *horizontal*, *strobe* and *block*. A *vertical* scan consists of a port scan of some or all ports on a single computer. The other three types of scans are used over multiple IP addresses. A *horizontal* scan is a scan of a single port across multiple IP addresses. If the port scan is of multiple ports across multiple IP addresses, it is called a *strobe* scan. A *block* scan is a port scan against all ports on multiple IP addresses. [4]

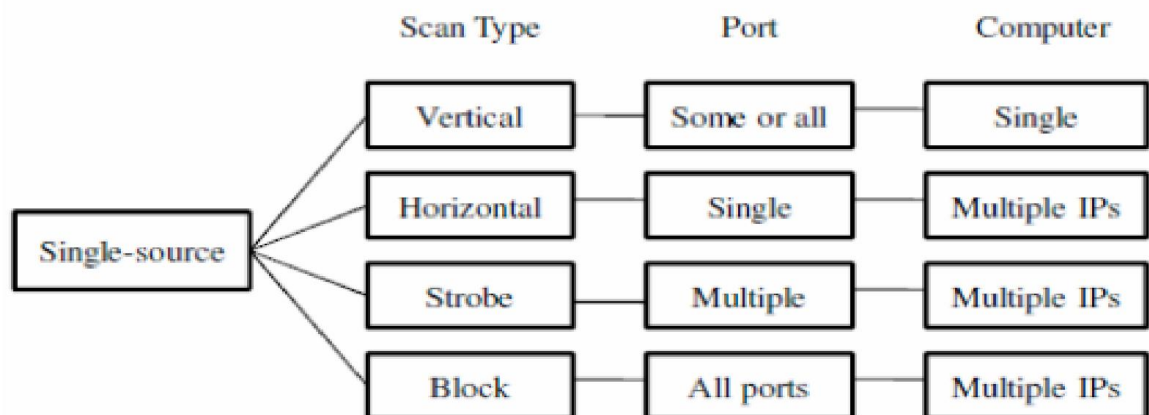


Figure 1.2 Single-source scan types with its ports detail

- **Vertical Scan** - is defined as a sequential or random scan of multiple (more than 5) ports of a single IP address from the same source during a one hour period. These are usually an attempt to survey which of several well known vulnerabilities applies to this host and are also known as *strobe* scans, based on one of the original script-kiddie tools. [5]
- **Horizontal Scan** - is a scan from a single source of one or several machines (5 or more) in a subnet aimed at the same target port, ie. The same vulnerability. In this case the attacker is searching for any machine that is running specific

service and does not care about any single machine in particular. The attacker could be just recruiting peers for launching larger distributed attacks.

- **Coordinated Scans** - are scans from multiple sources (5 or more) aimed at a particular port of destinations in the same /24 subnet within a one hour window. These are also called Distributed Scans. These scans usually come from the more aggressive/active sources that comprise several collaborative peers working in tandem.
- **Stealth Scans** - are horizontal or vertical scans initiated with a very low frequency to avoid detection. The key parameters in this definition include the maximum threshold (1 hour) and the minimum threshold for the average interscan distance. An average interscan distance below the minimum threshold indicates that the scan was not stealthy, *ie.* Not intended to evade NIDS systems. Two successive scans from the same source that are separated by more than the maximum interscan distance are considered to be unrelated or parts of different scanning episodes.

In the context of this research, the focus will be on the vertical port slow scans.

1.2.2 Distributed Port Scans:

Distributed information gathering is performed using either a *many-to-one* or a *many-to-many* model. Here, the attacker utilizes multiple hosts to execute information gathering techniques in different ways: *rate-limited* and *random* or *non-linear*. In a *rate-limited* information gathering technique, the number of packets sent by a host to scan is limited; this is based on the FreeBSD. implementation of UNIX where separate rate limits are maintained for open ports as well as closed ports. For example: TCP RST rate limited, ICMP port unreachable rate limited, and so on. On the other hand, a *random* or *non-linear* gathering technique refers to randomization of the destination IP-port pairs amongst the sources, as well as randomization of the time delay for each probe packet.

A *co-ordinated* attack has a more generic form of a distributed scan described by Staniford-Chen et al. It is defined as multi-step exploitation using parallel sessions with the objective of obscuring the unified nature of the attack or allowing the attackers to proceed more quickly. However, Green et al. define a *co-ordinated* attack as “multiple

IP addresses working together towards a common goal”. They also add that a co-ordinated attack can be viewed as multiple attackers working together to execute a distributed scan on many internal addresses or services. Staniford et al. later define a distributed scan as one that is launched from a number of different real IP addresses, so that the scanner can investigate different parts of the footprint from different places. An attacker can scan the Internet using a few dozen to a few thousand zombies. A zombie is a compromised host, whose owner is unaware that the computer is being exploited (a remote attacker has accessed and set up to forward transmissions (spams or viruses) to other computers on the network) by the external party. Yegneswaran et al. define *co-ordinated* scans as being scans from multiple sources aimed at a particular port of destinations within a one hour window. These scans usually come from more aggressive or active sources that comprise several collaborative peers working in tandem. Finally, Robertson et al. group source addresses together as forming a potentially distributed port scan if they are sufficiently close, where the scanner simply obtains multiple IP addresses from his ISP. It should be noted that all of these definitions imply some level of co-ordination among the single sources used in the scan. [4]

1.3 Port Scanning Techniques:

There are 65,536 standardly defined ports on a computer. They can be categorized into three large ranges: (i) Well Known Ports (0 - 1023), (ii) Registered Ports (1024 - 49151), and (iii) Dynamic and/or Private Ports (49152 - 65535). Normally, a port scan does not directly damage the system, but potentially a port scan helps the attacker in finding those ports that are available to launch attacks. Essentially, a port scan consists of sending a message to each port, one at a time and listening for an answer. The kind of response received indicates whether the port is being used and can therefore be probed further for weakness to launch future attacks.

Port scanning usually happens on TCP ports, i.e., ports that use a connection-oriented protocol; such ports return good feedback to the attacker. It also happens on UDP ports, but they provide connectionless services that may not readily give relevant information to attackers. a UDP port may be easily blocked by network defenders. Following are the various techniques of port scans (shown in **Figure 1.3**). Each of these scanning techniques is introduced in brief below. [4]

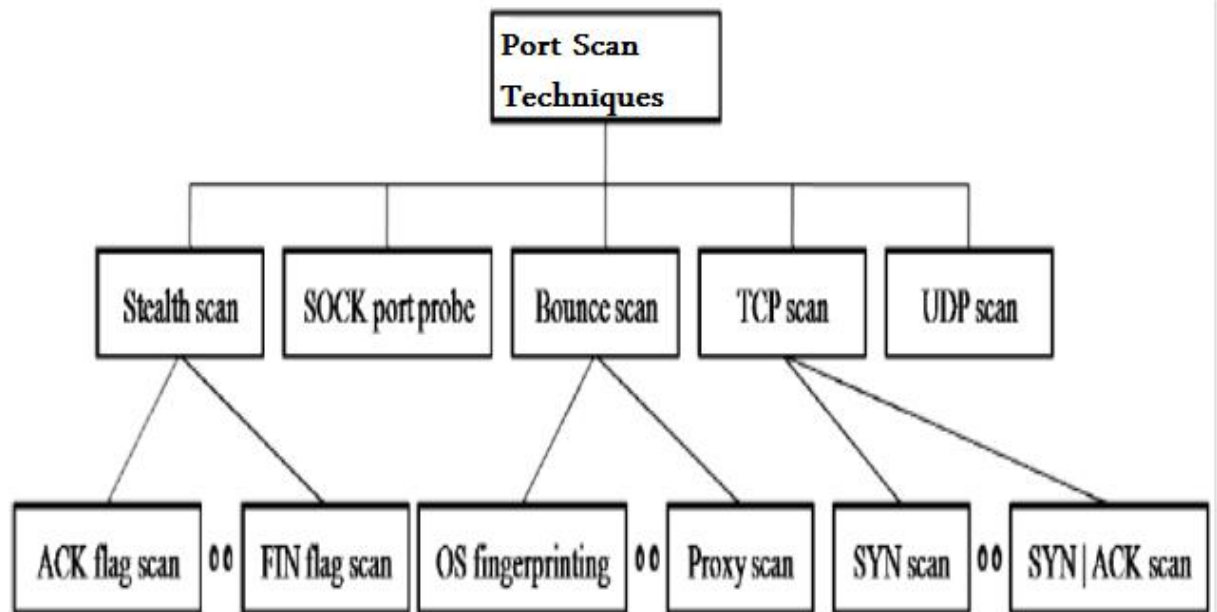


Figure 1.3 Techniques of port scan

- a. **Stealth Scan:** Such a scan is designed to go undetected by auditing tools. It sends TCP packets to the destination host with stealth flags. Some of the flags are SYN, FIN and NULL.
- b. **SOCKS Port Probe:** A SOCKS port allows sharing of Internet connections on multiple machines. Attackers scan these ports because a large percentage of users misconfigure SOCKS ports, potentially permitting arbitrarily chosen sources and destinations to communicate. A SOCKS port on a system may allow the attacker to access other Internet hosts while hiding his or her true location.
- c. **Bounce Scan:** It takes advantage of a vulnerability of the FTP protocol itself. Some applications that potentially allow bounce scans are email servers and HTTP Proxies.
- d. **TCP Scanning:** A TCP connection is never fully established during this type of scanning. So, it is used by smart attackers. If the attacker can clearly know that a remote port is accepting connections, the attacker can launch an attack immediately. It is much more difficult for network defenders to detect since this kind of connection attempts are not logged by the server's logging system. Some TCP scans are *TCP Connect ()*, Reverse Identification, IP Header Dump scan, SYN, FIN, ACK, XMAS, NULL and TCP Fragment.

- e. **UDP Scanning:** It attempts to find open ports related to the UDP protocol. However, UDP is a connectionless protocol and thus, it is not often used by attackers since it can be easily blocked.

1.4 TCP/IP Relevant Issues:

Before we move on to discussing the various techniques, we need to make clear in our mind, some basic things.

1.4.1 Packet Flags:

Almost all Port Scans are based on the client sending a packet to the target port of the remote system, containing a particular flag. A TCP Packet header contains six different types of flags which are [6] :

TCP Flags	Full Names	Descriptions
<i>URG</i>	Urgency pointer	Indicates the TCP priority of the packets.
<i>ACK</i>	Acknowledgment	Designates this packet as an acknowledgment of receipt.
<i>PSH</i>	Push	Flushes queued data from buffers.
<i>RST</i>	Reset	Resets a TCP connection on completion or being aborted.
<i>SYN</i>	Synchronization	Synchronizes a connection.
<i>FIN</i>	Finished	Finishes a transmission.

Table 1.1 TCP Flag Fields

1.4.2 TCP Connection Establishment:

TCP is a connection-oriented, reliable transport protocol. Connection-oriented means the two applications using TCP must establish a TCP connection with each other before they can exchange data. Reliability is provided in TCP by the use of checksums, timers, data sequencing and acknowledgments. By assigning a sequence number to **every** byte transferred, and requiring an acknowledgment from the other end upon receipt, TCP can guarantee reliable delivery. Sequence numbers are used to ensure proper ordering of the data and to eliminate duplicate data bytes. Note that in a TCP session there are usually

two streams of data (every end point is receiving from one of them and sending through the other). So an *ISN* must be assigned to each stream when the connection is being established. To see how it is done, let's suppose that **C** is a client wishing to connect to the server **S**, and analyze the connection establishment process (often called the *three-way handshake*) [6]:

1	C	<---SYN XX--->	S
2	C	<---SYN YY/ ACK XX+1--->	S
3	C	<---ACK YY+1--->	S

1. **C** sends a TCP message (known as SYN request) to **S** with the special flag SYN set to ON. A SYN request specifies the port number of the server that the client wants to connect to, and the client's ISN (XX in this example).

2. The server responds with its own SYN message containing **S**'s ISN (YY) and acknowledging **C**'s SYN by specifying that the **next** byte expected from **C** is the byte numbered XX+1.

3. **C** acknowledges the SYN message from **S**, and data transfer may take place.

1.4.3 Some Implementation details:

Most TCP/IP implementations (with important exceptions, though) follow these rules [6]:

- When a **SYN** (or **FIN**) segment arrives for a *closed port* TCP drops the segment and sends a **RST**.
- When a **RST** segment arrives for a *listening port*, it is simply dropped.
- When a **RST** segment arrives for a *closed port*, it is simply dropped.
- When a segment containing an **ACK** arrives for a *listening port*, it is dropped and a **RST** is sent.
- When a segment with the **SYN** bit **off** arrives for a *listening port*, it is simply dropped.
- When a **SYN** segment arrives for a *listening port*, the normal *three-way handshake* continues by replying **SYN|ACK**.
- When a **FIN** segment arrives for a *listening port*, it is simply dropped. “**FIN** behaviour” (closed port = **RST**, listening port = dropped) can also be seen with

the **PSH** and **URG** flags, with a TCP segment with no flags, and with all the combinations of **FIN|PSH|URG**.

1.5 Nmap:

There is plethora of tools to perform reconnaissance scans. Most of them are available for free download. They are Nmap, curl, hping, hydra, nessus, netcat, stunnel, tcpdump, Angry IP Scanner, Super Scan, FoundStone Vision, FPort, Scanline, PortScan 2000, Blue's Port Scanner and more. However not a single tool offers all the features offered by the NMAP.

Nmap. is a free and open source license utility for network discovery and security auditing. Many systems and network administrators also find it useful for tasks such as network inventory, managing service upgrade schedules, and monitoring host or service uptime. 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. It was designed to rapidly scan large networks, but works fine against single hosts. Nmap runs on all major computer operating systems, and official binary packages are available for Linux, Windows, and Mac OS X. [7]

1.5.1 Nmap Port Scanning Techniques:

Over time, a number of techniques have been developed for surveying the protocols and ports on which a target machine is listening. They all offer different benefits and problems. Here is a line up of the most common: [8]

1.5.1.1 TCP SYN scan (-sS):

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 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. [9]

1.5.1.2 TCP connect scan (-sT) :

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. 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.

1.5.1.3 UDP scans (-sU) :

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, but for most ports the packet is empty. The --data-length option can be used to send a fixed-length random payload to every port or (if you specify a value of 0) to disable payloads. 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. 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 was 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 takes 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.

1.5.1.4 Sctp INIT scan (-sY) :

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 1, 2, 3, 9, 10, or 13) is received.

1.5.1.5 TCP NULL, FIN, and Xmas scans(-sN; -sF; -sX):

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 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.

1.5.1.6 TCP ACK scan(-sA):

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.

1.5.1.7 TCP Window scan (-sW):

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.

1.5.1.8 TCP Maimon scan (-sM) :

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.

1.5.1.9 Custom TCP scan (--scanflags) :

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.

1.5.1.10 SCTP COOKIE ECHO scan (-sZ):

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 as 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; 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.

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

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).

1.5.1.12 IP protocol scan (-sO) :

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. 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.

1.5.1.13 FTP bounce scan (-b <FTP relay host>):

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.

We summarize the various port scan types discussed in this section along with firewall detection possibilities [4]

PST*	Protocol	TCP flag	VR*(OP*)	VR*(CP*)	FLDP*
TCP <i>Connect()</i>	TCP	SYN	ACK	RST	Yes
Reverse Ident	TCP	No	No	No	No
SYN Scan	TCP	SYN	ACK	RST	Yes
IP Header Dump Scan	TCP	No	No	No	No
SYN ACK Scan	TCP	SYN ACK	RST	RST	Yes
FIN Scan	TCP	FIN	No	RST	No
ACK Scan	TCP	ACK	No	RST	No
NULL Scan	TCP	No	No	RST	No
XMAS Scan	TCP	All flags	No	RST	No
TCP Fragment	TCP	No	No	No	No
UDP Scan	UDP	No	No	Port Unreachable	No
FTP Bounce Scan	FTP	Arbitrary Flag Set	No	No	No

Table 1.2 Details of port scan types and its firewall level detection possibilities

Note: PST (Port Scanning Technique), VR (Victim's Reply), OP (Open Port), CP (Closed Port), FLDP (Firewall Level Detection Possibility)

1.6 Conclusion:

Port scanning constitutes an important piece in hacker's attack puzzle it is method used by hackers in order to uncover and exploit weaknesses in networked computer hosts .in this chapter we present an overview of port scanning. It provides an overview of the most commonly used techniques in scanning, a categorization of scans, and scans, stealthy scans. In the next chapter we present an overview of some methods designed to detect port scanning.