TCP Ports - Interactions and Scanning

SYN	Synchronize. Initiates a connection between hosts.
ACK	Acknowledge. Established connection between hosts.
PSH	Push. System is forwarding buffered data.
URG	Urgent. Data in packets must be processed quickly.
FIN	Finish. No more transmissions.
RST	Reset. Resets the connection.

TCP Connect and SYN stealth (aka half-open) scans
TCP connect (-sT) is the most reliable scan type but also the noisiest and most detectable since it attempts the standard 3-way TCP handshake to a port. If that port is open and unfiltered, it will think there is a connection attempt it is waiting for. When a SYN stealth scan (-sS) receives an ACK response it will shoot back a RST to slam the connection shut. It is also gives more clear results than inverse scans. To Nmap, SYN/ACK means port is open, RST means port is closed and no response (even on retrying) or ICMP back means it is filtered en route.

Packet fragmentation can evade IDS packet filtering and detection. A stealth SYN can be used in a way that splits the TCP header over several IP packets to mess with firewalls. Some firewalls may have rule sets that block IP fragmentation queues, but might not due to the adverse effect on network performance
Example: nmap -sS -T4(time delay) -A -f -v 192.168.0.2

Inverse TCP Flag scanning sends packets with various TCP flags (or no flags) enabled.

- When the port is open (or firewall-filtered), no response comes from the host.
- When the port is closed, a RST/ACK is received from the target host.

It’s often said these are called inverted since responses are sent back only by closed ports. Inverse TCP scans are named for the flags they use, NMap has -sF (FIN scan, with only the FIN flag set), -sN (Null scan with no flags set) and -sX for an Xmas scan setting all three FIN, URG, PUSH flags. Nmap doesn’t have a URG and PSH similar individual shortcut (like --flag URG), Hping has -U and -P, but these don’t yield different results than other inverse scans (NUL, FIN and Xmas all you really need). A maimon scan sends packets with both the FIN and ACK flags set. It was considered more useful but not in modern times.

NULL, FIN and Xmas scans: if we get a RST, we know the port is closed. If it's not closed, the remote host shouldn't respond so it's either open or filtered

Caveats: According to RFC 793, a RST/ACK packet must be sent for connection reset, when the port is closed on host side. RFC 793 is completely ignored in Windows, so you won’t get a RST/ACK response when trying to interact with the closed port. So, is only effective when target is not a Windows OS. If an “ICMP unreachable” is returned Nmap considers the port firewall-filtered. These might get through some firewalls (FIN most likely), and sometimes no response doesn’t mean a port is open, since instead the packet was dropped by the firewall. Finally, for ALL types of scans, just because a method can pass through a firewall does NOT necessarily mean it is slipping through undetected!

ACK scans (are not considered inverse)
Sent with just the ACK flag set, these never determines ports are closed or open (or even open/filtered). It is used to map out firewall rulesets, determining whether they are stateful or not and which ports are filtered, which usually means running other scans and cross-referencing the results. The simplest usage shown below, determines getting ***a RST back means “unfiltered”*** (RFC 793 again), and either an ICMP unreachable or no result at all is “filtered” (firewalls that block usually make no response or send back an ICMP destination unreachable)

```
# nmap -sA -T4 scanme.nmap.org
        Not shown: 994 filtered ports
        22/tcp  unfiltered ssh
        25/tcp  unfiltered smtp …
```

Forbidding anything but outbound connections is fine, but blocking ACK packets without state info doesn’t tell which side started the connection. To block unsolicited ACK packets and allow packets belonging to legitimate connections, firewalls must stateful. Either way, the stateless approach is still quite common (netfilter/iptables, basic zone-based, etc). Cross-reference a SYN scan below shows us both getting responses; SYN scan says 98/100 ports are filtered, ACK says all are filtered (all getting RST back). This is a stateless use of “iptables -A INPUT -m multiport -p tcp --destination-port 22,80 -j ACCEPT” on the firewall.

```
# nmap -sS -p1-100 -T4 para
Not shown: 98 filtered ports
22/tcp open   ssh
80/tcp closed http
Nmap done: 1 IP address (1 host up) scanned in 3.81 seconds
```

```
# nmap -sA -p1-100 -T4 para
All 100 scanned ports on para (192.168.10.191) are: unfiltered
Nmap done: 1 IP address (1 host up) scanned in 0.70 seconds
```

The ACK scan shows some packets are probably reaching the destination host- firewall forgery is always possible. Other scan types, such as FIN scan, may even be able to determine which ports are open and thus infer the purpose of the hosts. While you may not be able to establish TCP connections to those ports, they can be useful for determining which IP addresses are in use, OS detection tests, certain IP ID shenanigans, and as a channel for tunneling commands to rootkits installed on those machines. Such hosts may be useful as zombies for an IP ID idle scan.

Two ways of interpreting other info from ACK scans can be more revealing: checking the time-to-live (TTL) field and the WINDOW field of received packets

It doesn't tell us if a port is open or closed, but it does try to tell us if the
firewall is stateful (keeps tracks of connections) or not (probably just denies
incoming SYN packets).
If the firewall is non-stateful and just drops SYN packets, an ACK will get in
because it looks like a reply to something from the other side.
If an open OR closed port receives an unexpected ACK, it should send a RST back.
So if we get a RST back, then it means the firewall is non-stateful (or there's
just not one in place). If we don't get a response, or some ICMP unreachable is
sent, it's most likely filtered.

TTL-Base Analysis:

TTL value can be used as a marker of how many systems the packet has hopped through
Below, the value on the RST packets returned by port 22 is 50, whereas the other ports return a value of 80. This suggests that port 22 is open on the target host because the TTL value returned is smaller than the TTL boundary value of 64*.

```
1: host 192.168.0.12 port 21: F:RST TTL:80 WIN:0
2: host 192.168.0.12 port 22: F:RST TTL:50 WIN:0
```

The *firewalk* assessment tool is similar http://www.packetfactory.net/projects/firewalk/.

Window-based Analysis:

If window field has non-zero value then port is open, no response, presumed filtered

```
1: host 192.168.0.20 port 22: F:RST -> ttl: 64 win: 512
2: host 192.168.0.20 port 23: F:RST -> ttl: 64 win: 0
```

The advantage of using ACK flag probe scanning is detection is difficult (for both IDS and host-based systems). The disadvantage is it relies on TCP/IP stack implementation bugs, which are prominent in BSD-derived systems but not in many other modern platforms.

```
hping3 -A 72.14.207.99 -p 80 -c 1
```

```
nmap -sW
```

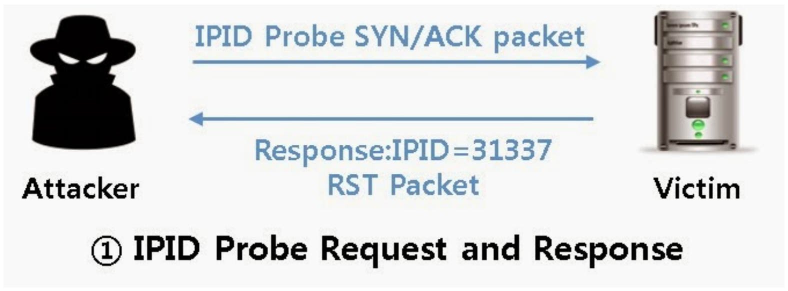
Some systems use a positive Window size for open ports, and zero for closed. Fewer systems do the exact opposite. If you scan and get tons of open ports and just a few closed ones, chances are it's the opposite. And some systems don't do either, so you can't always trust it.

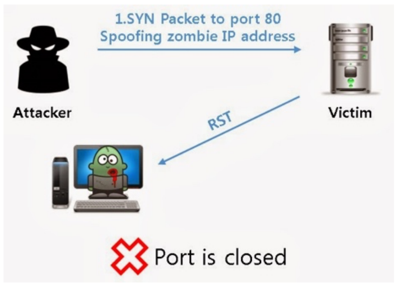
IDLE scan/ IP ID header scan - IP ID (IP fragment ID number)
Sends a spoofed source address to a computer to find out what services are available for complete blind scanning of a remote host. This is accomplished by spoofing another computer's IP address. No packet is send from your own IP address; instead, another host is used to scan the remote host and determine the open ports. This is done by expecting the sequence number of the zombie host and if the remote host checks the IP of the scanning party, the IP of the zombie machine will show up. You don't need access to the zombie machine to do any of this.

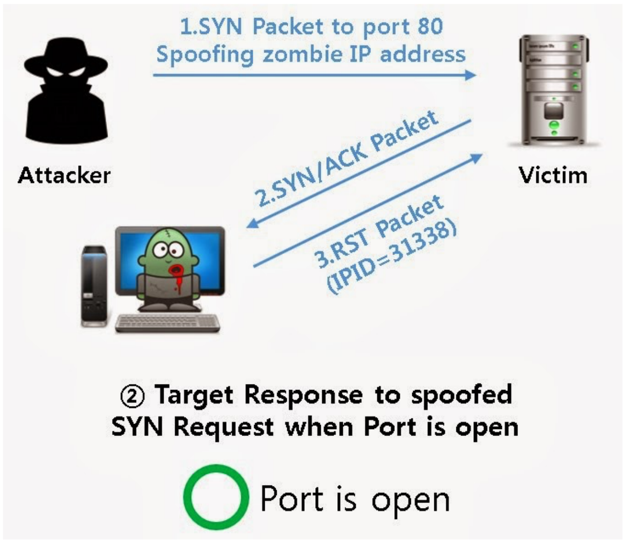
As noted, to determine if port is open, SYN scan full or stealth, get back an RST if closed, SYNACK if open.
Consider:
- An unsolicited SYNACK is responded to with a RST; An unsolicited RST will be ignored
- Each IP packet has a fragment ID (IPID), incremented for each sent. Check to get # of packets sent since probe

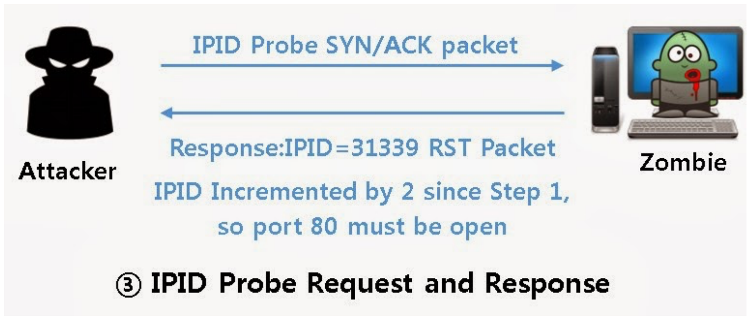
Step 1) Send an unsolicited SYNACK to the target to get a RST containing IP ID (say it's 31337)
Step 2) Spoof the IP address of your zombie machine, and send a SYN packet to the target (port 80)
- If the port is open, the target will send a SYNACK to the zombie and it will shoot a RST back to the target. (IP ID predicted is 31338)
- If the port is closed, the target will send a RST to the zombie host, and it won't respond

Step 3) Probe the target again with another SYNACK. The IP ID should have incremented by 2 from the last RST obtained in step 1 if the tested port on the target is open.








IPID Probe SYN/ACK packet
Attacker
Response:IPID=31339 RST Packet
Zombie
IPID Incremented by 2 since Step 1,
so port 80 must be open
③ IPID Probe Request and Response

```
nmap -Pn -p- -sI www.host123.com www.host345.com
Idlescan using zombie www.host123.com (192.130.18.124:80); Class: Incremental
Nmap scan report for 198.182.30.30.110
(the 40321 ports scanned but not shown below are in the state: closed)
Port      State     Service
21/tcp    open      ftp
25/tcp    open      smtp
80/tcp    open      http
Nmap done: 1 IP address (1 host up) scanned in 1931.23 seconds
```

ICMP Echo scanning/List scan
ICMP echo scanning is used to discover live machines by pinging all the machines in the target network. ICMP probes sent to the broadcast or network address are relayed to all the host addresses in the subnet. The live systems will send ICMP echo reply message to the source of ICMP echo probe.
nmap -P 192.168.0.0/24 --OR-- nmap -sn 192.168.0.2 --OR-- nmap -sL -v 192.168.2.5

UDP Scanning -sUV
Sends UDP packets to each target port, often empty but some commonly known port services need a specific payload, so Nmap tries add it in (the -V option checks Nmap's database). Some listening applications will still discard empty packets as invalid, firewalls also drop packets without responding, and UDP might just drop packets or time out. So, a UDP response means it is open, no response can be either open or filtered (can't tell!). Nmap running with application version detection (-sUV) helps there a bit.
If you get a UDP response back, it's open
If the port is closed, it will likely shoot back an ICMP port unreachable (type 3/code 3)
If it is another ICMP unreachable error (type 3, code 1, 2, 9, 10, or 13) it's likely filtered.

Some OSs limit the frequency of ICMP Port Unreachable messages, Nmap adjusts accordingly. Microsoft-based OSs do not usually implement any type of ICMP rate limiting. Not usually useful for most types of attack, but it can reveal information about some exploitable services (DHCP, SNMP, NFS, etc). Malware often uses UDP
nmap -sU -v 192.168.0.2

<u>Some flags available in Nmap (see https://nmap.org/book/man-briefoptions.html)</u>

-sT	TCP connect scan	-sR	RPC scan	-oN	Normal output
-sS	SYN scan	-sL	List/DNS scan	-oX	XML output
-sF	FIN scan	-sI	Idle scan	-oG	Greppable output
-sX	XMAS tree scan	-Po	Don't ping	-oA	All output
-sN	Null scan	-PT	TCP ping	-T Paranoid	Serial scan; 300 sec between scans
-sP	Ping scan	-PS	SYN ping	-T Sneaky	Serial scan; 15 sec between scans
-sU	UDP scan	-PI	ICMP ping	-T Polite	Serial scan; .4 sec between scans
-sO	Protocol scan	-PB	TCP and ICMP ping	-T Normal	Parallel scan
-sA	ACK scan	-PB	ICMP timestamp	-T Aggressive	Parallel, 300 sec timeout, 1.25 sec/probe
-sW	Windows scan	-PM	ICMP netmask	-T Insane	Parallel, 75 sec timeout, 0.3 sec/probe

IP PROTOCOL SCAN
Looks for supported IP protocols rather than open ports; sends raw IP packets with different values in the protocol field of the header. Instead of looking for "ICMP Port Unreachable", it looks for ICMP Protocol Unreachables to tell if it's unsupported (somewhat "closed"). If we get a response back in the same protocol, it's supported (somewhat "open"). If we get some different ICMP unreachable, it's probably filtered. If we don't get anything back, it's either open (and didn't reply) or filtered.

```
# nmap -sO 192.168.10.1
# hping3 -c 1 --rawip --ipproto 0 192.168.10.1
# hping3 -c 1 --icmp 192.168.10.1
# hping3 -c 1 --rawip --ipproto 2 192.168.10.1
```

FTP bounce scanning
Hosts running outdated FTP services can relay numerous TCP attacks, including port scanning. There is a flaw in the way many FTP servers handle connections using the PORT command (see RFC 959 or technical description) that allows data to be sent to user-specified hosts and ports. In their default configurations, the FTP services running on the following older Unix-based platforms are affected
The FTP bounce attack can have a far more devastating effect if a writable directory exists because a series of commands or other data can be entered into a file and then relayed via the PORT command to a specified port of a target host. For example, some- one can upload a spam email message to a vulnerable FTP server and then send this email message to the SMTP port of a target mail server. Figure 4-9 shows the parties involved in FTP bounce scanning.

1. The attacker connects to the FTP control port (TCP port 21) of the vulnerable FTP server that she is going to bounce her attack through and enters passive mode, forcing the FTP server to send data to a specific port of a specific host:
QUOTE PASV

```
	227 Entering Passive Mode (64,12,168,246,56,185).
```
2. A PORT command is issued, with an argument passed to the FTP service telling it
to attempt a connection to a specific TCP port on the target server; for example, TCP port 23 of 144.51.17.230:
```
	PORT 144,51,17,230,0,23
	200 PORT command successful.
```
3. After issuing the PORT command, a LIST command is sent. The FTP server then
attempts to create a connection with the target host defined in the PORT command issued previously:
```
LIST
	150 Opening ASCII mode data connection for file list
	226 Transfer complete.
```
If a 226 response is seen, then the port on the target host is open. If, however, a 425 response is seen, the connection has been refused:
```
LIST
	425 Can't build data connection: Connection refused
```
Nmap supports FTP bounce port scanning with the –P0 and –b flags used in the following manner:
```
    nmap –P0 –b username:password@ftp-server:port <target host>
```
The –P0 flag must be used to suppress pinging of the target host, as it may not be accessible from your location (e.g., if you are bouncing through a multihomed FTP server). Also, you may not want your source IP address to appear in logs at the target site.

Proxy bounce scanning
Attackers bounce TCP attacks through open proxy servers. Depending on the level of poor configuration, the server will sometimes allow a full-blown TCP port scan to be relayed. Using proxy servers to perform bounce port scanning in this fashion is often time-consuming, so many attackers prefer to abuse open proxy servers more efficiently by bouncing actual attacks through to target networks.
ppscan.c, a publicly available Unix-based tool to bounce port scans, can be found in source form at:
http://examples.oreilly.com/networksa/tools/ppscan.c http://www.phreak.org/archives/exploits/unix/network-scanners/ppscan.c

----

(NMAP) When scanning hardened environments, you should use the -Pn flag to force scanning of each address within scope. A slower timing policy (such as -T2) is also useful, as an aggressive policy will trigger SYN flood protection by firewalls

SYN probes - four response variants: a packet with SYN/ACK flags indicating an open port, RST/ACK denoting closed, no response or an ICMP type 3 message implying a filter

This runs three Nmap scans to identify accessible hosts across TCP, SCTP, and UDP. Optionally, load a list of targets into Nmap from a file using the -iL flag.
```
nmap –T4 –Pn –v –n –sS –F –oG /tmp/tcp.gnmap 192.168.0.0/24
nmap –T4 –Pn –v –n –sY –F –oG /tmp/sctp.gnmap 192.168.0.0/24
nmap –T4 –Pn –v –n –sU –p53,111,123,137,161,500 -oG /tmp/udp.gnmap 192.168.0.0/24
```
These scans generate output in /tmp with gnmap file extensions. UDP results may contain false positives. If the UDP dataset looks noisy (i.e. all the hosts are reporting to have open ports), then simply disregard it. Once you're happy with the contents of these files, use grep and awk to generate a refined list of targets, as follows:
```
grep open /tmp/*.gnmap | awk '{print $2}' | sort | uniq > /tmp/targets.txt
```
This list should then be fed into four subsequent scans: A fast TCP scan of common services
```
nmap -T4 -Pn -v --open -sS -A -oA tcp_fast -iL /tmp/targets.txt
```
A TCP scan of all ports:
```
nmap -T4 -Pn -v --open -sS -A –p0-65535 -oA tcp_full -iL /tmp/targets.txt
```
An SCTP scan of all ports:
```
nmap -T4 -Pn -v --open -sY –p0-65535 -oA sctp -iL /tmp/targets.txt
```
A UDP scan of common services:
```
nmap -T3 -Pn -v --open –sU -oA udp -iL /tmp/targets.txt
```

The -oA flag will generate multiple output files for each scan type, including a gnmap file that you can easily parse, and a full text file (i.e. tcp_fast.nmap) that is human- readable. These scanning modes do not perform service fingerprinting or deep analysis of the exposed network services

Same procedure for IPv6 - first sweeping IPv6 address space for hosts running common network services, and then perform full scanning of that subset). When TCP sweeping large IPv6 networks, I recommend reducing the port list to increase speed, from -F (100 common ports) to -p22,25,53,80,111,139,443.
Upon preparing a list of targets (e.g. /tmp/targets.txt) from host discovery and sweeping, run the same four scans as before, using the -6 flag to perform the scanning over IPv6:

```
nmap -6 -T4 -Pn -v --open -sS -A -oA ipv6_tcp_fast -iL /tmp/targets.txt
nmap -6 -T4 -Pn -v --open -sS -A –p0-65535 -oA ipv6_tcp_full -iL /tmp/targets.txt
nmap -6 -T4 -Pn -v --open -sY –p0-65535 -oA ipv6_sctp -iL /tmp/targets.txt
nmap -6 -T3 -Pn -v --open -sU -oA ipv6_udp -iL /tmp/targets.txt
```

SCTP
Stream Control Transmission Protocol (SCTP) is a transport protocol that sits alongside TCP and UDP. Intended to provide transport of telephony data over IP, the protocol duplicates many of the reliability features of SS7, and underpins a larger family of protocols known as SIGTRAN. SCTP is supported by operating systems including IBM AIX, Oracle Solaris, HP-UX, Linux, Cisco IOS, and VxWorks.

SCTP chunk types

ID	*Value*	*Description*
0	DATA	Payload data
1	INIT	Initiation
2	INIT ACK	Initiation acknowledgement
3	SACK	Selective acknowledgement
4	HEARTBEAT	Heartbeat request
5	HEARTBEAT ACK	Heartbeat acknowledgement
6	ABORT	Abort
7	SHUTDOWN	Shutdown
8	SHUTDOWN ACK	Shutdown acknowledgement
9	ERROR	Operation error
10	COOKIE ECHO	State cookie
11	COOKIE ACK	Cookie acknowledgement
12	ECNE	Explicit congestion notification echo
13	CWR	Congestion window reduced
14	SHUTDOWN COMPLETE	Shutdown complete

Tools such as Nmap and SING don’t identify these responses from private addresses (behind NAT), as low- level stateful analysis of the traffic flowing into and out of a network is required. A quick and simple example of this behavior can be seen in the ISS BlackICE personal firewall event log in Figure 4-1 as a simple ICMP ping sweep is performed.

It is beneficial to run a network sniffer such as Ethereal or tcpdump during testing to pick up on unsolicited ICMP responses, including “ICMP TTL exceeded” (type 11 code 0) messages, indicating a routing loop, and “ICMP administratively prohibited” (type 3 code 13) messages, indicating an ACL in use on a router or firewall.

OS Fingerprinting Using ICMP
Ofir Arkin’s Xprobe2 utility performs OS fingerprinting primarily by analyzing responses to ICMP probes.

Another SYN port scanner worth mentioning is Scanrand, a component of the Paketto Keiretsu suite. Paketto Keiretsu contains a number of useful networking utilities that are available at http://www.doxpara.com/read.php/code/paketto.html. For Windows, Foundstone’s SuperScan is an excellent port

scanning utility with good functionality, including banner grabbing. SuperScan is available from http://
examples.oreilly.com/networksa/tools/superscan4.zip

Scanrand is well designed, with distinct SYN probing and background listening components that allow for very fast scanning. Inverse SYN cookies (using SHA1) tag out- going probe packets, so that false positive results become nonexistent, as the listening component only registers responses with the correct SYN cookies. Scanrand is much faster than bulkier scanners, such as Nmap.

Unicornscan (http://www.unicornscan.org) is another tool that performs fast half- open scanning. It has some unique and very useful features, and it is recommended for advanced users.
(UDP payload scan) against the 10.3.0.1 candidate within my environment, results are as follows:

```
root@kali:~# unicornscan -mU 10.3.0.1
UDP open domain[ 53] from 10.3.0.1 ttl 128
UDP open netbios-ns[ 137] from 10.3.0.1 ttl 128
```

UDP scanning results across tools may vary. Nmap provides a comprehensive option with -sV, but testing of a single host using the -F option (scanning 100 ports) takes around 10 minutes to complete.

Using malformed TCP flags to probe a target is known as an inverted technique because responses are sent back only by closed ports. RFC 793 states that if a port is closed on a host, an RST/ACK packet should be sent to reset the connection. To take advantage of this feature, attackers send TCP probe packets with various TCP flags set.

Vscan is another Windows tool you can use to perform inverse TCP flag scanning. The utility doesn't require installation of WinPcap network drivers; instead it uses raw sockets within Winsock 2 (present in Windows itself). Vscan is available from http://examples.oreilly.com/networksa/tools/vscan.zip.

SING (Send ICMP Nasty Garbage) - http://sourceforge.net/projects/sing
Ability to transmit and receive spoofed packets, send MAC-spoofed packets, and support the transmission of many other message types, including ICMP address mask, timestamp, and information requests, as well as router solicitation and router advertisement messages

ICMPScan - http://www.bindshell.net/tools/icmpscan
Bulk scanner derived from Nmap that sends type 8, 13, 15, and 17 ICMP; can process inbound responses by placing the network interface into promiscuous mode, thereby identifying internal IP addresses and machines that respond from probes sent to subnet network and broadcast addresses. Because ICMP is a connectionless protocol, it is best practice to resend each probe (using –r 1) and set the timeout to 500 milliseconds (using -t 500). We also set the tool to listen in promiscuous mode for unsolicited responses (using the –c flag).

```
Nmap 7.01SVN ( https://nmap.org )
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; 10.0.0-255.1-254
  -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
  -sn: Ping Scan - disable port scan
  -Pn: Treat all hosts as online -- skip host discovery
  -PS/PA/PU/PY[portlist]: TCP SYN/ACK, UDP or SCTP discovery to given ports
  -PE/PP/PM: ICMP echo, timestamp, and netmask request discovery probes
  -PO[protocol list]: IP Protocol Ping
  -n/-R: Never do DNS resolution/Always resolve [default: sometimes]
  --dns-servers <serv1[,serv2],...>: Specify custom DNS servers
  --system-dns: Use OS's DNS resolver
  --traceroute: Trace hop path to each host
SCAN TECHNIQUES:
  -sS/sT/sA/sW/sM: TCP SYN/Connect()/ACK/Window/Maimon scans
  -sU: UDP Scan
  -sN/sF/sX: TCP Null, FIN, and Xmas scans
  --scanflags <flags>: Customize TCP scan flags
  -sI <zombie host[:probeport]>: Idle scan
  -sY/sZ: SCTP INIT/COOKIE-ECHO scans
  -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,S:9
  --exclude-ports <port ranges>: Exclude the specified ports from scanning
  -F: Fast mode - Scan fewer ports than the default scan
  -r: Scan ports consecutively - don't randomize
  --top-ports <number>: Scan <number> most common ports
  --port-ratio <ratio>: Scan ports more common than <ratio>
SERVICE/VERSION DETECTION:
  -sV: Probe open ports to determine service/version info
  --version-intensity <level>: Set from 0 (light) to 9 (try all probes)
  --version-light: Limit to most likely probes (intensity 2)
  --version-all: Try every single probe (intensity 9)
  --version-trace: Show detailed version scan activity (for debugging)
SCRIPT SCAN:
  -sC: equivalent to --script=default
  --script=<Lua scripts>: <Lua scripts> is a comma separated list of
           directories, script-files or script-categories
  --script-args=<n1=v1,[n2=v2,...]>: provide arguments to scripts
  --script-args-file=filename: provide NSE script args in a file
  --script-trace: Show all data sent and received
  --script-updatedb: Update the script database.
  --script-help=<Lua scripts>: Show help about scripts.
           <Lua scripts> is a comma-separated list of script-files or
           script-categories.
OS DETECTION:
  -O: Enable OS detection
  --osscan-limit: Limit OS detection to promising targets
  --osscan-guess: Guess OS more aggressively
TIMING AND PERFORMANCE:
  Options which take <time> are in seconds, or append 'ms' (milliseconds),
  's' (seconds), 'm' (minutes), or 'h' (hours) to the value (e.g. 30m).
  -T<0-5>: Set timing template (higher is faster)
  --min-hostgroup/max-hostgroup <size>: Parallel host scan group sizes
  --min-parallelism/max-parallelism <numprobes>: Probe parallelization
  --min-rtt-timeout/max-rtt-timeout/initial-rtt-timeout <time>: Specifies
      probe round trip time.
  --max-retries <tries>: Caps number of port scan probe retransmissions.
  --host-timeout <time>: Give up on target after this long
  --scan-delay/--max-scan-delay <time>: Adjust delay between probes
  --min-rate <number>: Send packets no slower than <number> per second
  --max-rate <number>: Send packets no faster than <number> per second
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
  --proxies <url1,[url2],...>: Relay connections through HTTP/SOCKS4 proxies
  --data <hex string>: Append a custom payload to sent packets
  --data-string <string>: Append a custom ASCII string to sent packets
  --data-length <num>: Append random data to sent packets
  --ip-options <options>: Send packets with specified ip options
  --ttl <val>: Set IP time-to-live field
  --spoof-mac <mac address/prefix/vendor name>: Spoof your MAC address
  --badsum: Send packets with a bogus TCP/UDP/SCTP checksum
OUTPUT:
  -oN/-oX/-oS/-oG <file>: Output scan 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 -vv or more for greater effect)
  -d: Increase debugging level (use -dd or more for greater effect)
  --reason: Display the reason a port is in a particular state
  --open: Only show open (or possibly open) ports
  --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
  --webxml: Reference stylesheet from Nmap.Org for more portable XML
  --no-stylesheet: Prevent associating of XSL stylesheet w/XML output
MISC:
  -6: Enable IPv6 scanning
  -A: Enable OS detection, version detection, script scanning, and traceroute
  --datadir <dirname>: Specify custom Nmap data file location
  --send-eth/--send-ip: Send using raw ethernet frames or IP packets
  --privileged: Assume that the user is fully privileged
  --unprivileged: Assume the user lacks raw socket privileges
  -V: Print version number
  -h: Print this help summary page.
EXAMPLES:
  nmap -v -A scanme.nmap.org
  nmap -v -sn 192.168.0.0/16 10.0.0.0/8
  nmap -v -iR 10000 -Pn -p 80
SEE THE MAN PAGE (https://nmap.org/book/man.html)
```

https://techgenix.com/packet-analysis-tools-methodology-part1/

https://sourceforge.net/software/product/Essential-NetTools/

https://medium.com/100-days-of-linux/10-curl-commands-that-you-should-know-ee3d032eb351

nmap hping switches
nmap and hping switches

TERM	DEFINITION
-sT	nmap TCP connect scan

-sS	nmap SYN scan
-sF	nmap FIN scan
-sX	nmap XMAS tree scan
-sN	nmap Null scan
-sP	nmap Ping scan
-sU	nmap UDP scan
-sO	nmap Protocol scan
-sA	nmap ACK scan
-sW	nmap Windows scan
-sR	nmap RPC scan
-sL	nmap List/DNS scan
-sI	nmap Idle scan
-Po	nmap Don't ping
-PT	nmap TCP ping
-PS	nmap SYN ping
-PI	nmap ICMP ping
-PB	nmap TCP and ICMP ping
-PB	nmap ICMP timestamp
-PM	nmap ICMP netmask
-oN	nmap normal output
-oX	nmap xml output
-oG	nmap greppable output

-oA	nmap all output
-T Paranoid	nmap serial scan; 300 sec between scans
-T Sneaky	nmap serial scan; 15 sec between scans
-T Polite	nmap serial scan; .4 sec between scans
-T Normal	nmap parallel scan
-T Aggressive	nmap parallel scan, 300 sec timeout, and 1.25 sec/probe
-T Insane	nmap parallel scan, 75 sec timeout, and .3 sec/probe
-1	Sets ICMP mode. For example, hping3 -1 172.17.15.12 performs an ICMP ping.
-2	Sets UDP mode. For example, hping3 -2 192.168.12.55 -p 80 performs a UDP scan on port 80 for 192.168.12.55.
-8	Sets scan mode, expecting an argument for the ports to be scanned (single, range [1–1000], or "all"). For example, hping3 -8 20-100 scans ports 20 through 100.
-9	Sets Hping in listen mode, to trigger on a signature argument when it sees it come through. For example, hping3 -9 HTTP -I eth0 looks for HTTP signature packets on eth0.
--flood	hping3 Will send packets as fast as possible, without taking care to show incoming replies. For example, a SYN flood from 192.168.10.10 against .22 could be kicked off with hping3 -S 192.168.10.10 -a 192.168.10.22 -p 22 --flood.
-Q --seqnum	This option can be used in order to collect sequence numbers generated by the target host. This can be useful when you need to analyze whether a TCP sequence number is predictable (for example, hping3 172.17.15.12 -Q -p 139 -s).
-F	hping3 Sets the FIN flag.
-S	hping3 Sets the SYN flag.
-Rhping3	hping3 Sets the RST flag.
-P	hping3 Sets the PSH flag.
-A	hping3 Sets the ACK flag.
-U	hping3 Sets the URG flag.

-X	hping3 Sets the XMAS scan flags.