

Scanning networks overview

- Process of obtaining additional information about hosts, ports and services in network
- More detailed reconnaissance
- Purpose is to identify vulnerabilities in communication channels and then create an attack plan.
- Different techniques are used to identify hosts, ports, and services in the target network.
- Used by administrators to verify security policies, monitoring uptime etc.
- Can craft custom packets using different tools
 - E.g. [Colasoft Packet Builder](#), [NetScanTools Pro](#), [Packeth](#)

Scanning types

- **Port scanning:** to list open ports and services
- **Network scanning:** to list IP addresses
- **Vulnerability scanning:** to discover the presence of known vulnerabilities

Scanning in IPv6 Networks

- IP supports more addresses
 - Bigger search space
 - More difficult and complex than IPv4
 - Harder to do ping sweeps
- Supports large number of hosts in a subnet.
 - Probing takes longer time
- 💡 Attacker should find addresses from logs, e-mails, traffics etc. to identify addresses.

Common ports to scan

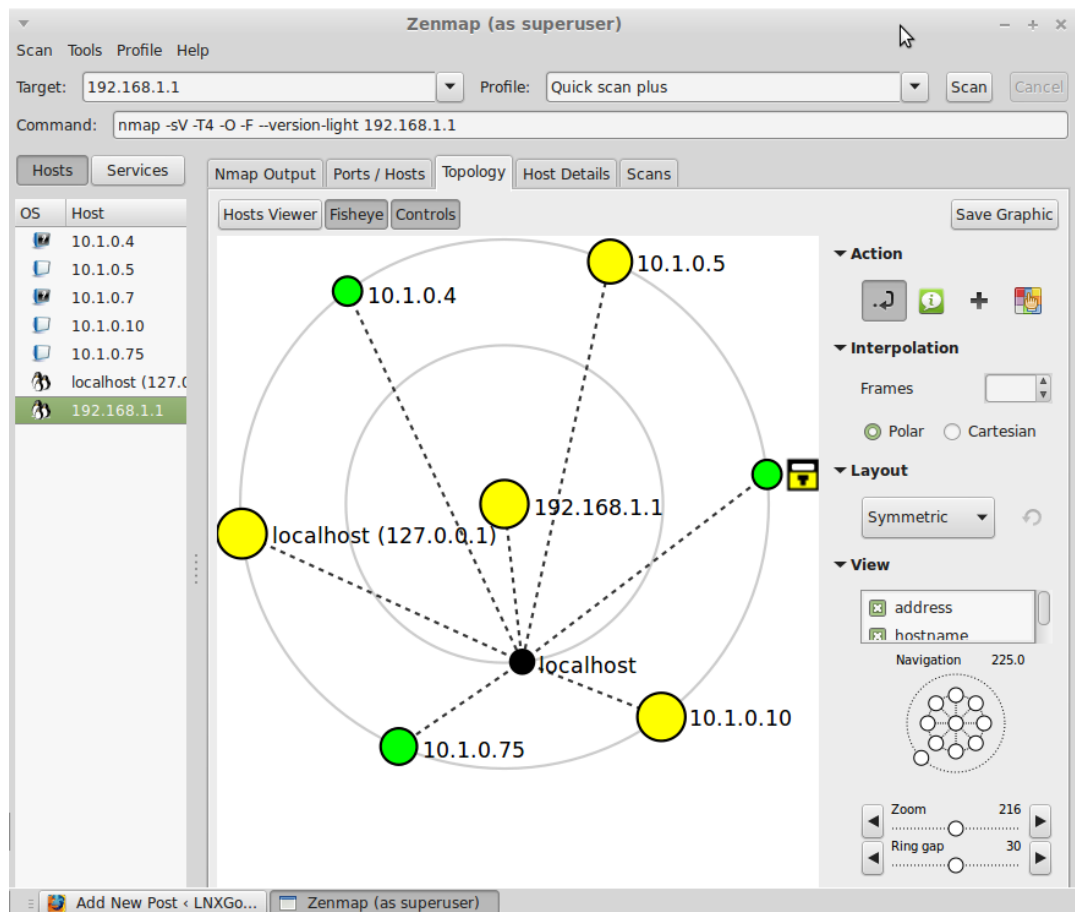
- 📝 List of TCP and UDP port numbers

Port	Protocol	Service
21	TCP	FTP (File Transfer Protocol)
22	TCP	SSH (Secure Shell)
23	TCP	Telnet
25	TCP	SMTP (Simple Mail Transfer Protocol)
53	TCP/UDP	DNS (Domain Name Server)
80	TCP	HTTP (Hypertext Transfer Protocol) ¶ HTTP/3 will run over UDP
123	TCP	NTP (Network Time Protocol)
443	TCP/UDP	HTTPS
500	TCP/UDP	IKE/IPSec (Internet Key Exchange / IPSec)
631	TCP/UDP	IPP (Internet Printing Protocol)
3389	TCP/UDP	RDP (Remote Desktop Protocol)
9100	TCP/UDP	AppSocket/JetDirect (HP JetDirect, Printer PDL Data Stream)

- Read more on [IANA ports list](#)
- See also • [Port monitoring](#) | [Malware analysis](#) • [Common ports and services to enumerate](#) | [Enumeration](#)

Drawing and mapping out network topologies

- Useful for identifying and understanding the topology of the target network.
 - The diagram can tell the attacker how firewalls, IDSes, routers, and other devices are arranged in the network
- Information can be used for vulnerability discovery and exploit.
- A popular tool is `zenmap`: A GUI for [nmap](#)




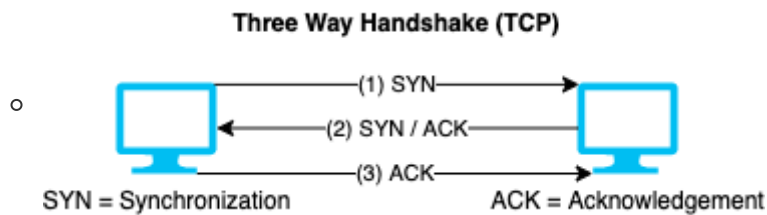
- E.g. scanning network in a coffee shop. Run `ipconfig` (windows) to get private IP address and subnet range. Then you can scan and create a map.

TCP/IP

TCP connection

Three-way handshake

- Also known as • **3-way handshake** • **three-way handshake** • **3 way handshake** • **three way handshake**
- Establishes a TCP connection
-  Sender: `SYN` → Receiver: `SYN ACK` → Sender: `ACK`



- `ACK` is then set in every packet sent after the handshake

Termination

-  Sender: `FIN` → Receiver: `ACK FIN` → Sender: `ACK`

IPv4

- IPv4 loopback address (localhost of your own machine) is `127.0.0.1`

IPv4 address types

1. Unicast

- Acted on by a single recipient

2. Multicast

- Acted on only by members of a specific group


3. Broadcast

- Acted on by everyone in the network
- Two types:
 - **Limited broadcast**
 - Delivered to every system inside a domain using:
 - IP: 255.255.255.255
 - MAC: FF:FF:FF:FF:FF:FF
 - Ignored by routers
 - **Directed broadcasts**
 - Sent to all devices on subnet
 - Use subnets broadcast address
 - E.g. if subnet is 192.168.17.0/24 then it uses 192.168.17.255
 - Routers may take action on the packets.

IPv6



- IPv6 uses a 128-bit address instead of the 32-bit IPv4 version
- Represented as eight groups of four hexadecimal digits separated by colons
 - E.g. `2001:0db8:85a3:0000:0000:8a2e:0370:7334`
- Leading zeros can be removed e.g.
 - Original: `2001:0001:0002:0003:0004:0005:0006:0007`
 - Short: `2001:1:2:3:4:5:6:7`
- The loopback address is `::1`
 - Shortened version of `0000:0000:0000:0000:0000:0000:0000:0001`

CIDR

- Method of the representing IP addresses
-  Easy way to find out CIDR ranges, remember: `/24 255.255.255.0 256`
 - So `/24` gives 256 IP addresses, `/25` gives 128, `/26` gives 64 and so on.
- IPv4 Notation

CIDR Range	Total IP Addresses	Subnet mask
<code>/32</code>	0	<code>255.255</code>
<code>/30</code>	4	<code>.225.252</code>
<code>/28</code>	16	<code>.255.240</code>
<code>/26</code>	64	<code>.255.192</code>
<code>/24</code>	256	<code>.255.0</code>
<code>/22</code>	1024	<code>.248.0</code>
<code>/20</code>	4096	<code>.240.0</code>

TCP flags

- Used to indicate a particular connection state or provide additional information
- Size of each flag is 1 bit being either `0` or `1`
-  Flag types
 - **Sequence number (`SYN`)**
 - Also known as **synchronization** flag.
 - Synchronize sequence numbers
 - First step of connection establishment (3-way handshake)
 -  Only the first packet sent from each end should have this flag set
 - **Acknowledgement (`ACK`)**
 - Confirms successful packet retrieval
 - **Push (`PSH`)**
 - Tells receiver to process packets instead of buffering them
 - **Urgent (`URG`)**

- Process packets directly before others, even if they're not complete
- **Finish (FIN):**
 - 1 indicate connection termination requests
 - Used in the last packet sent from the sender.
- **Reset (RST)**
 - 1 aborts the connection in response
 - Sent from the receiver to the sender when a packet is sent to a particular host that was not expecting it.
 - Also used as
 - DDoS attack, see [RST attack](#)
 - Scanning technique, see [RFC 793 scans](#)

Finish (FIN) vs Reset (RST)

FIN	RST
Gracefully termination	Sudden termination
Only one side of conversation is stopped	Whole conversation is stopped
No data loss	Data is discarded
Receiver of FIN can choose to continue communicating	Receiver has to stop communication


Push (PSH) vs Urgent (URG)

PSH	URG
All data in buffer are pushed	Only urgent data is pushed immediately
Data is delivered in sequence	Data is delivered out of sequence

TCP/IP sessions

- TCP uses stateful sessions
- Connection establishment must be done before data transfer
- **Session initiation**
 1. Source sends SYN packet
 2. Destination responds with SYN/ACK packet
 3. Source sends ACK packet
 - Connection stays open until closed with FIN or RST packets.
- Session termination

OSI model

- Conceptual model that characterizes and standardizes the communication functions
-  Uses seven abstraction layers:
 1. **Physical** (bits)

- Media, signal & binary transmission
 - E.g. • Cables (fiber) • Fiber • Wireless • Hubs • Repeaters
- 2. **Data link** (frames)
 - Physical addressing: MAC & LLC
 - E.g. • Ethernet • [PPP](#) • Switch • Bridge
- 3. **Network** (packets)
 - Path determination & IP
 - E.g. • IP • [ICMP](#) • [IPSec](#) • IGMP
- 4. **Transport** (segments)
 - End-to-end connections and reliability
 - E.g. • TCP • UDP
- 5. **Session** (data)
 - Sync & send to ports, inter-host communication
 - E.g. • API's • Sockets • WinSock
- 6. **Presentation** (data)
 - Syntax layer
 - Encrypts/decrypts if needed
 - E.g. • [SSL/TLS](#) (not entirely) • [SSH](#) • IMAP • [FTP](#) • MPEG • JPEG
- 7. **Application** (data)
 - End User Layer: network process to application
 - E.g. • HTTP • [FTP](#) • IRC • [SSH](#) • [DNS](#) • [SMTP](#)
- See also • [Firewall types per OSI Layer](#) | [Firewall](#) • [Vulnerability stack](#) | [Hacking web applications](#) • [Encryption types per OSI layer](#) | [Encryption algorithms](#)

TCP/IP model



- TCP/IP model defines four levels:
 1. **Link layer**: • [ARP](#) • [PPP](#) • [MAC](#)
 2. **Internet layer**: • TCP • UDP • DCCP • SCTP ...
 3. **Transport layer**: • IP • ICMP • ECN • [IPSec](#) ...
 4. **Application layer**: • [DNS](#) • HTTP • HTTPS • [FTP](#) • [SSH](#) • SMTP ...
- ¶ OSI model does not match well TCP/IP
 - [RFC 3439](#) considers layering "harmful"
- ¶ E.g. SSL/TLS does not fit in any of OSI or TCP/IP layers
 - In OSI it's in layer 6 or 7, and, at the same time, in layer 4 or below.
 - In TCP/IP it's in between the transport and the application layers.

TCP/IP vs OSI model

TCP/IP	Protocols and services	OSI model
Application	• HTTP • FTP • Telnet • NTP • DHCP • PING	• Application • Presentation • Session
Transport	• TCP • UDP	Transport
Network	• IP • ARP • ICMP • IGMP	Network
Network interface	• Ethernet • PPTP	• Data Link • Physical

Scanning tools


Nmap

- Scans network by sending specially crafted packets
- Allows finding hosts on network with service, OS and firewall information
- Allows custom scripts written in LUA using **NSE (Nmap Scripting Engine)**
 -  Can be used to detect and/or exploit vulnerabilities
 - E.g. can [detect shellshock using nmap scripting engine](#)
- Includes
 - [Ncat](#): reads and writes data across networks from the command
 - [ndiff](#): compares scan results
 - [nping](#): generates packets and analyzes responses
-  Used often in movies including Matrix Reloaded, see [the list](#)
- See also [Nmap | Network footprinting](#) and [Nmap | Vulnerability analysis](#).

Phases of an Nmap scan

1. **Script pre-scanning**: Runs NSE scripts that are run once per execution for each targets, e.g. `dhcp-discover`.
2. **Target enumeration**: Resolves DNS names, CIDR network notations etc. to list of IPv4 or IPv6 addresses
3. **Host discovery (ping scanning)**: Checking if a host (or which hosts are) is alive before deeper investigation
4. **Reverse-DNS resolution**: Provides IP numbers for hosts that are alive
5. **Port scanning**: Probes are sent and remote port states are classified as `open`, `closed`, `filtered`
6. **Version detection**: Determines what server software is running on remote system
7. **OS detection**: Determines OS that's running on the port
8. **Traceroute**: Usually involves another round of reverse-DNS resolution for intermediate hosts.
9. **Script scanning**: Runs most of the scripts rather than pre-scan and post-scan phases.
10. **Output**: Prints results using different options e.g. XML
11. **Script post-scanning**: Runs scripts that process results and deliver final reports and statistics

Common Nmap options

-  All options are important for a security tester to be able to use Nmap.
- `-n` (no resolution): Skips DNS resolution and scanning for DNS addresses
- `-A`: Enable • OS detection • version detection • script scanning • traceroute
- `--traceroute`: Enables trace routing
- `--script` or `-SC`: Activates custom script scanning

-S*: port scan options

- Uses [ICMP echo request](#), [TCP SYN](#) to port 443, [TCP ACK to port 80](#), and an ICMP timestamp request.
- **-sn**
 - Also known as **ping scan** or **host discovery**
 - Skips port scanning
- Common commands include:
 - TCP port scanning: **-ss** ([SYN](#)), **-sT** ([connect](#)), **-sN** ([NULL](#)), **-sF** ([FIN](#)), **-sX** ([XMAS](#))
 - UDP port scanning: **-su** ([UDP](#))
 - **-sv**: service/version detection scan
 - **-sO**
 - IP protocol scan
 - Not really a port scan
 - Lists supported IP protocols (TCP, ICMP, IGMP etc.) by target system.

-P*: ping (host discovery) options

- **-P*** options are used to select different ping methods
- User with **-sn** to skip port scanning and do host discovery only.
- Common commands include:
 - TCP: **-PS**, ([SYN](#)), **-PA** ([ACK](#))
 - Others: **-PR** ([ARP](#)), **-PO** (IP protocol ping), **-PE** ([ICMP](#)), **PU** ([UDP](#))
 - **-Pn** (**no ping**)
 - Also known as **pingless scan** or **port scan**
 - Skips host discovery and treats all hosts as online

Specifying ports

- **-p-** to scan all ports (1-65535)
- **-p**: only scan specified ports
 - E.g. **-p U:53,111,137,T:21-25,80,139,8080**
- **-r**: Scan ports consecutively - don't randomize

-O: OS fingerprinting

- **-o** is used for operating system fingerprinting
- It's Far more effective if at least one open and one closed TCP port are found.
 - Flag with **--osscan-limit** and Nmap will not try OS detection against hosts that do not meet this criteria.
- **--fuzzy** or **--osscan-guess** switch: Nmap will guess more aggressively
- Requires **sudo** privileges
- See also [banner grabbing](#)

-O* : output options

- `-oX` for XML output.
- `-oG` for `grep` able output to be able to use linux [grep command](#) to search in text
- ⚠ Not to be confused with `-O` (OS fingerprinting)

Faster scans

- `-T*`: Timing template
 - From slowest to fastest: `-T0` (paranoid), `-T1` (sneaky), `-T2` (polite), `-T3` (normal | default), `-T4` (aggressive) or `-T5` (insane)
- `-F`: Fast (limited port) scan
 - Nmap as default most common 1000 ports, `-F` reduces it to 100
- ⚠ If the scan is too fast the system can drop the packets
 - Risky because the system can cancel the whole scan when it detects for the first time.

Target specification

- `nmap <target>`
- Everything that isn't an option (or option argument) is treated as a target host specification
- Target can be IP address(es) or hostname(s) (resolved via DNS)
- Target can be specify single or multiple hosts:
 - Scanning single host:
 - E.g. `nmap 192.168.10.0` (IP address) or `nmap localhost` (hostname)
 - Scanning many hosts:
 - CIDR style addressing
 - E.g. `192.168.10.0/24` would scan the 256 hosts
 - Octet range addressing (more flexible)
 - E.g. `192.168.0-255.1-254`
 - Full octet scan: `192.168.0.*`
 - Using target list: `nmap -iL targets`
 - Scan multiple addresses using `nmap <target-1>, <target-2> ...`
 - E.g. `nmap privacy.sexy cloudarchitecture.io`

Hping

- [Open-source](#) port scanner
- Sends custom ICMP, UDP, or TCP packets and then displays any replies

Hping vs Nmap

- `nmap` can scan a range of IP addresses
 - `hping` can only port scan one individual IP address
- `hping` is more lower level and stealthier than `nmap`
- `hping` does not support IPv6 while `nmap` does.

Common hping commands

- `--tcp-timestamp`
 - Enables TCP timestamps
 - Tries to guess the timestamp update frequency and the remote system uptime.
 - ¶ Many firewalls drop packets without timestamp.
- `-Q` or `--seqnum`
 - Collects sequence numbers generated by target host
 - Useful when you need to analyze whether TCP sequence number is predictable.
- Setting flags using
 - `-F` (FIN), `-S` (SYN), `-R` (RST), `-P` (PUSH), `-A` (ACK), `-U` (URG)
- Scanning entire subnet: `hping3 -1 10.0.1.x`
- Listen to traffic (e.g. to sniff): `hping3 -9 HTTP -I eth0`
- See also its [man page](#)

Mobile tools

- [IP Scanner](#) for IOS
- [Fing](#) for IOS and Android

Scanning techniques

- Used to
 - Identify open ports also i.e. **port scanning**
 - Identify which hosts are alive i.e. **host discovery**
- Same techniques can be used for both port scanning and host scanning
 - But can require different tools or commands
 - See: Nmap [port scanning](#) vs [host discovery](#) commands

Techniques per protocol

Scanning ICMP

- Also called **ping scan** or **ICMP scan**
- Typically used for checking if a host is alive or has errors
- 💡 It saves time before running a port scan
- 🚫 Often incoming ICMP requests are disabled by firewalls.
- On Network Layer 3, so does not have port abstraction like TCP/UDP (Transport layer 4)
- In Nmap, ping commands start with `-P*` e.g. `-PE` for `ICMP Echo`
- **Round-trip time**
 - Time it takes for a packet to make a complete trip
- If blocked, usually returns no response or sometimes ICMP errors such as type 3 code 13 (destination unreachable: communication administratively prohibited)

Broadcast ICMP ping

- Sends packet to every possible address on the network
- E.g. `ping -b 192.168.135.255` (subnet broadcast)
 - Pings to `192.168.129.19`, `192.168.129.20`, `192.168.129.21` ...
 - Or all broadcast through `ping -b 192.168.129.255`
 - Alive hosts will populate local arp table, can be seen using `arp -a`
- Usually discarded and a useless method
- See also [Smurf attack](#) | [Denial of Service](#)

ICMP ping packet types

- **Echo**
 - Sends `ICMP ECHO` (type 8) expecting `ICMP ECHO` reply
 - Also known as **ICMP echo scanning** or [ICMP ping sweep](#)
 - 📝 **Tools**
 - Nmap: `-PE`
 - hping: `hping3 -1 <target>`
- **Timestamp**
 - Sends `ICMP TIMESTAMP` (type 13)

- **Tools**

- Nmap: `-PP`
- hping: `hping3 -1 <target> --icmp-type 13`


- **Address mask**

- Sends `ICMP ADDRESS MASK` (type 17)

- **Tools**

- Nmap: `-PM`
- hping: `hping3 -1 <target> --icmp-type 17`

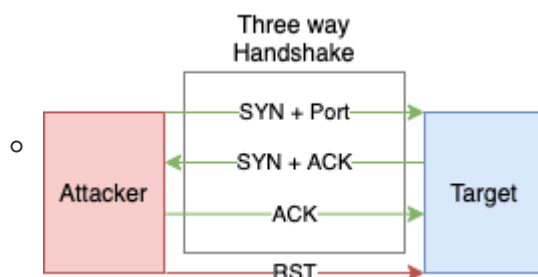
ICMP ping sweep

- Also called **ICMP sweep** or **ICMP echo scanning**
-  Used to determine active hosts in target network range.
- Tools include
 - [Angry_ip](#)
 - [Nmap](#): `nmap -sn 192.168.0.*`
 - or `nmap -sn <ip-address>/<subnet-range>`
 - or `nmap -sn 192.168.0.1-30` to scan e.g. between `1-30` in last octet
 - `-sn`: (optional) skip port scan


Scanning TCP

TCP connect



- Also known as **full open scan** or **TCP Connect() scan**
- Used for detecting open ports upon the completion of the three-way handshake.
- Works by establishing a full connection and then dropping it by sending a RST packet.



- **Pros**

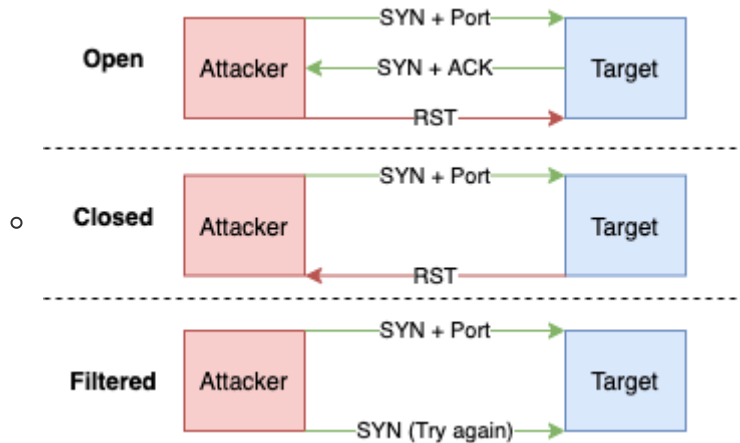
-  Accurate, most reliable
- Requires no additional user privileges
- Allows scanning behind a firewall.

- **Cons**

-  Easily detectable by IDSes and logged
-  Use [SYN](#) which would be faster and stealthier
- **Tools**
 - [Nmap](#): `nmap -sT <ip-or-host>`

SYN scanning

- Also known as **TCP SYN ping**, **SYN stealth**, **stealth scan**, **half-open scan** or **TCP ping scan**
- Default and most popular scan
- Works by resetting the TCP connection before the three-way handshake is completed, which in turn makes the connection half open.




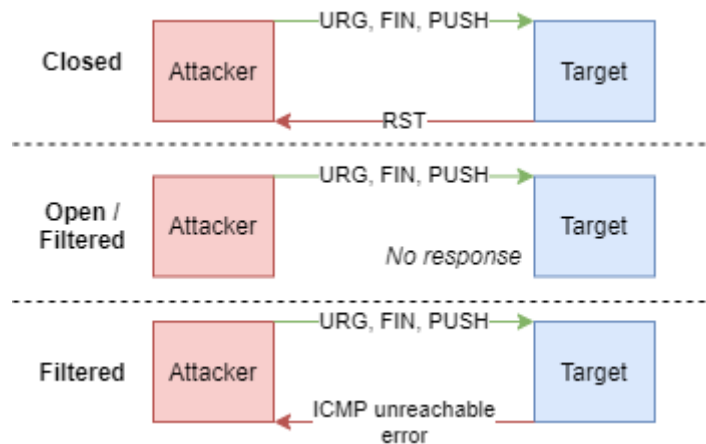
- **Pros**
 - Fast
 - Reliable
 - Works on any compliant TCP stack
 - Stealthy because it never completes TCP connections (can bypass firewalls and logging)
- **Cons**
 - Require root privileges
 - Rulesets block many SYN scan attempts
- **Tools**
 - [Nmap](#): `-PS` (host discovery) or `-ss` (port scan)
 - Hping: `hping3 -8 <port-list e.g. 50-60> -S <ip-address> -V`


RFC 793 scans

- According to [RFC 793](#), any packet not containing `SYN`, `RST`, or `ACK` bits will result in
 - `RST` if the port is closed
 - no response if the port is open
- Works by sending TCP probe packets with or without `SYN`, `RST`, `ACK` flags.
- **Pros**
 - Very stealthy (more than `SYN`) against IDS and logging systems
 - Avoids TCP three-way handshake
- **Cons**
 - Require root privileges
 - Unreliable (false positives) against systems that do not follow the RFC
 - Unix-only
 - Ineffective against Windows and many IBM/Cisco devices
- Using Nmap, one could utilize `--scanflags` to test with permutations of each case (total: 8) to check for `RST` response, but there are easier ways for most popular combinations: • [XMAS](#) • [NULL scan](#) • [FIN scan](#)

XMAS Scan

- Works by sending a TCP frame with `FIN`, `URG`, and `PUSH` flags set.
-  **Christmas tree packet**: packet with every option set, like bulbs on a Christmas tree
-



- Tools
 - Hping: `hping3 -F -P -U <ip-address> -p <port-number>`
 - `-F` for `FIN`, `-P` for `PUSH`, `-U` for `URG`
 -  [Nmap](#): `-sX`

Inverse TCP flag scan

- An inverse TCP scan has `PSH` or `URG` or `FIN` or none flag set, i.e. a single flag or no flag.
- As opposed to [XMAS](#) that sets `PSH` and `URG` and `FIN`, i.e. all three flags at once

NULL Scan

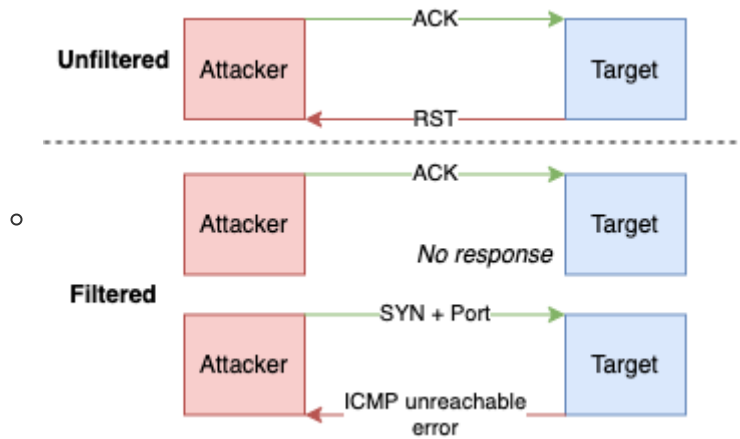
- Also known as **NULL probe**
- Does not set any bits (TCP flag header is 0)
- [Nmap](#): `-sN`

FIN Scan

- Also known as **FIN probe**
- Sets just the TCP `FIN` bit.
- [Nmap](#): `-sF`

ACK scanning

- Also known as **ACK flag scanning**, **ACK flag probe scanning** or **TCP ACK scan**
- Used to detect existence of firewalls, cannot detect open ports
- Works by sending TCP packet with `ACK` flag set.
 - `ACK` (acknowledgment) is used to acknowledge the successful receipt of a packet



- **Pros**

- Difficult to log, avoids IDS detection
- Helps detecting existence of stateful firewalls
 - As they would not allow `ACK` without `SYN`, see [TCP sessions](#)

- **Disadvantages**

- Relies on BSD network code bug in older versions
- Slow

- **Tools**

- [Nmap](#): `-PA` (host discovery) or `-SA` (port scan)
- Hping: `hping3 -A <ip-address> -p <port>`

ACK Classification

- Two fields are inspected to classify `RST` response: `WINDOW` and `TTL`
- `TTL`
 - It's open if TTL has less than `64` while others have higher values

```
packet 1: host X.X.X.X port 20: F:RST -> ttl: 70 win: 0 => closed
packet 2: host X.X.X.X port 21: F:RST -> ttl: 70 win: 0 => closed
packet 3: host X.X.X.X port 22: F:RST -> ttl: 40 win: 0 => open
packet 4: host X.X.X.X port 23: F:RST -> ttl: 70 win: 0 => closed
```

- `WINDOW`

- It's open if it has non-zero value.
- Works for older BSD and UNIX but has been patched

```
packet 1: host X.X.X.X port 20: F:RST -> ttl: 64 win: 0 => closed
packet 2: host X.X.X.X port 21: F:RST -> ttl: 64 win: 0 => closed
packet 3: host X.X.X.X port 22: F:RST -> ttl: 64 win: 512 => open
packet 4: host X.X.X.X port 23: F:RST -> ttl: 64 win: 0 => closed
```

IDLE scan

- Also known as **TCP Idle Scan**, **IDLE header scan**, **header scan**, **IDLE/IPID scan**
- Allows for blind port scanning (without sending any packet with own IP)
- Utilizes IP address of a zombie machine through spoofed packets
- 🧑‍🔬 Found by author of [hping2](#) tool
- 📄 Flow

1. Probe the zombie's IP ID and record it.

- **IP ID**

- Every IP packet on the Internet has a fragment identification number
- Incremented by OSes for each packet sent
- Zombie should be
 - idle as no other traffic would increase the traffic
 - assigning IP ID packets incrementally on global level instead of per-host basis.

2. Forge a `SYN` packet from the zombie and send it to the desired port on the target.

3. Probe the zombie's IP ID again.

- If it's increased compared to one in step 1, port is open (it has received)


- **Pros**

- Ultimate stealth scan as attackers IP would not be revealed
- Can be used for framing as IDS will report zombie as the attacker

- **Cons**

- It takes far longer
- Many ISPs implement egress filtering to prevent the packet spoofing


- **Tools**

- [Nmap](#): `nmap -Pn -sI <zombie-ip/domain> <target-ip/domain>`
 -  `-sI`: Idle scan
 - `-Pn`: no ping to be stealthy

Scanning UDP

- Connectionless stream protocol, so no handshakes
- UDP is used by e.g. DNS (port 53), SNMP (port 161/162), and DHCP (port 67/68)

UDP Scanning

- Also known as **UDP ping**, **UDP/ICMP error scan**, **UDP port scan** or **UDP ICMP_PORT_UNREACHABLE scan**
- Exploits UDP behavior where the receiver sends an ICMP packet with error code when a port is unreachable.
- No response is interpreted as "open" or "filtered" behind firewall
- **Pros**
 - Avoids TCP IDS
 - Scans non-TCP ports that are quite common
- **Cons**
 - Provides port information only
 - ICMP is rate-limited by Linux kernel however not by Windows
 - Require root privileges
 - Slower and more difficult than TCP
- **Tools**
 - Hping: `hping3 -2 <ip-address> -p <port>`
 -  [Nmap](#): `-PU` (host discovery) or `-sU` (port scan)

Other techniques

List Scanning

- Performing a reverse DNS resolution to identify the names of the hosts.
- No packets are sent to hosts
- **Pros**
 - Sanity check to ensure that target IP addresses are proper
 - Stealthy, unobtrusive, does not trigger IDS alerts
- Tools
 - [Nmap](#): `nmap -sL <ip-address>`

SSDP Scanning

- Used to detect UPnP vulnerabilities and carry out buffer overflow or DoS attacks
- **SSDP**
 - Simple Service Discovery Protocol
 - Network protocol based on for discovery of devices on same network
 - Basis of UPnP (Universal Plug and Play) protocol
- Steps:
 1. Attacker discovers networked devices and create a list of all the devices that respond.
 2. Attacker creates a UDP packet with the spoofed IP address of the targeted victim.
 3. Attacker then uses a botnet to send a spoofed discovery packet to each plug-and-play device
 4. Each device will send a reply to the targeted victim with an amount of data up to about 30 times larger than the attacker's request.
 5. The target then receives a large volume of traffic from all the devices and becomes overwhelmed, potentially resulting in denial-of-service to legitimate traffic
- Tools
 - [Metasploit](#): `use auxiliary/scanner/upnp/ssdp_msearch`

ARP Scan

- Useful when scanning an ethernet LAN
- **ARP**: Regular ARP packet request looks for the MAC address using the device's IP address.
- Tools
 - `nping`: Learn router MAC address: `nping --arp-type ARP <router-ip-address>`
 - [nmap](#): `nmap -sn -PR 192.168.0.*`
 - `-PR`: ARP discovery
 - `-sn`: skip port scan

Countermeasures

- Configuring firewall and IDS to detect and block scans
- Protecting mechanisms for filtering and routing from being bypassed using certain ports
- Updating the router, IDS, and firewall
- Keeping as few ports open as necessary and filtering the rest
- Checking the configuration of the network and its ports
- Ensuring the proper configuration of anti- scanning and spoofing rulesets

Bypassing IDS and firewall

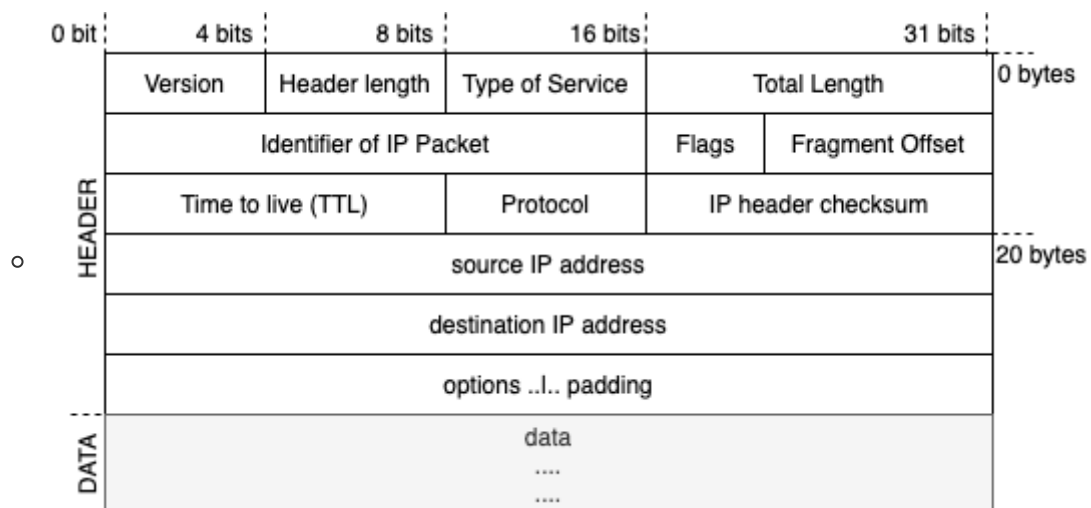
- Read more on [Intrusion Detection System Detection | Nmap](#)
- See also • [Evading IDS](#) and [Evading Firewalls](#)

Packet fragmentation

- Also known as **IP fragment scanning** or **IP fragmentation**
- 📄 Splitting up TCP header to several smaller (fragmented) packets when sending
- Server then reassembles them once all packets are received.
- Usually ignored by IDSes as processing them requires a lot of computer resources
- Any IP datagram can be fragmented: including UDP, TCP, ICMP, etc.
- See also [session splicing](#) for HTTP header variant.
- Tools
 - [Nmap](#): `-f` flag e.g. `nmap -f <ip-or-host>`
 - splits the packets into 8 bytes or less after the IP header
 - Alternatively can use `--mtu` option allows to set bytes e.g. `--mtu 16`
 - [fragroute](#)
 - Usage: `fragroute <domain-name>`
 - Intercept, modify, and rewrite egress traffic to use fragments

Source routing

- Also called **path addressing**
- Specifying which path the malformed packet will take to get to the target host.
- Used to skip routes (routers/gateways) where firewalls exist
 - Disregards what route tables say
- 🚫 Almost always blocked
- Done by modifying IP address field in IP Options field



- Using Nmap:
 - **Loose routing**
 - Specifying packet to be loose source routed through given IP way points

- E.g. `--ip-options "L 192.168.0.7 192.168.30.9"`
- **Strict routing**
 - You will have to specify every single hop along the path.
 - E.g. `--ip-options "S 192.168.0.7 192.168.0.9 .. 192.168.30.9"`
- See also [IP address spoofing through source routing | Session hijacking](#).

IP address decoy

- Also known as **decoy scan**
- All packets originate from the scanning machine, but some have spoofed source addresses.
- **Pros:** Helps to confuse port scan detection
- **Cons**
 - Does not offer any information beyond a regular scan.
 - Slows down the scanning process
- Using Nmap:
 - `nmap -D decoy1,decoy2,ME,decoy3... <target>`: Manual list with custom positioned ME
 - or `nmap -D RND:10 <target>` to randomize 10 decoy IP addresses

IP address spoofing

- Used to make packet appear to come from someone else
- Done by changing address information in IP packet header field
- Replies go back to the spoofed address not to the attacker
- Mostly used for DoS attacks
- Tools:
 - hping: `hping3 <target> -a <spoofed-ip>`
 - Nmap: `nmap <target> -S <spoofed-ip>`

IP address spoofing detection techniques


- **Direct TTL probes**
 - Ping to suspect IP
 - If `TTL` in reply is not same as suspect packet, it may be spoofed
- **IP Identification Number**
 - Ping to suspect IP
 - `IPID` should be close to but higher than suspect packet
 - OS increases IP incrementally for each packet sent
- **TCP Flow Control Method**
 - **Sliding window protocol**
 - In TCP, for each packet `ACK` is expected back before sending another packet.
 - *Window* packets are allowed to be sent without having to wait for an ACK.
 - Allows packets to arrive out of order
 - Window size field in TCP header
 - Tells maximum amount of data sender can transmit without waiting for `ACK`
 - **Windows update packet** is used to negotiate a different window size.

- Attacker that uses spoof IP's do not receive window size information
- If victims receives data packets beyond the window size, they are spoofed packets

IP address spoofing countermeasures

- Use encryption
 - Best prevention against IP spoofing attacks
- Avoid trust relationships
 - Also known as **no trust architecture**
 - Do not rely on IP-based authentication
 - Test all packets, even when they come from one of your trusted hosts
 - E.g. through password authentication, OAuth etc.
- Use firewalls and filtering mechanisms
- Use random initial sequence numbers (ISN)
 - As `SYN` is `ISN+1` it allows malicious connections.
 - E.g. if it's based on timed counter it's predictable
- Ingress filtering
 - Blocking incoming traffic based [access control lists \(ACLs\)](#)
 - Good against blocking unauthorized access
- Egress filtering against insider attacks
 - Blocking outgoing traffic
 - Good against insider attacks where e.g. malware can send information
- [SYN flooding countermeasures](#)

Encryption

-  Encryption over e.g. SSL/TLS or SSH is a good evasion as it "cripples" payload inspection.
- One way to be able to inspect is through **MITM attacks**
 1. The server-side encryption is terminated at the inspecting firewall
 2. The firewall re-encrypts the client-side connection and passes data in between.
- Other solution is to use **Network Traffic Analysis (NTA)**
 - It focuses more on unencrypted metadata rather than encrypted payload.
 - It uses behavioral analysis to sort through network traffic and identify dangerous or suspicious traffic based on what the traffic/host does on the network.

Proxy servers

- Acts as an intermediary for requests from clients and servers
- `client <--> proxy <--> server`
- You can use [free public proxies](#) for educational purposes
 - ⚠ Careful with public proxies as they can be malicious e.g. inject stuff into your traffic.
 - 🗣 Have your own proxies set-up, see [steps for creating a proxy server](#)

Usage by defenders

- As a firewall to protect internal network
- As a IP address multiplexer hiding IP addresses of internal servers.
- Anonymize web surfing to some extent e.g. with VPN services or [Tor](#)
 - Mobile tools include [Shadowsocks](#) (cross-platform sock5 proxy), [proxydroid](#) (http / sock4 / sock5 on Android)
- Filtering unwanted content such as ads
- Save bandwidth on local networks


Usage by attackers

- To conceal the real source of the scan.
- To remotely access intranets/corporate resources using victim as proxy
- To do man-in-the-middle attacks by redirecting user traffic to the proxy
- To sniff user data
- Tools include
 - [Parox.proxy](#) (outdated), or updated fork of it: [OWASP ZAP](#)

Proxy chaining

- Helps to increase anonymity by using two or more proxies in chain:
 - like `your_host <--> proxy1 <--> proxy2 <--> target_host`

ProxyChains

-  [Open source](#) unix tool to setup a proxy.
- Download a file using e.g. `proxychains4 wget <fileurl>`
- Configure through its configuration file `vim /etc/proxychains.conf`
 - E.g.

```
sock4 127.0.0.1 9050
http 159.65.133.175 3128
http 50.198.134.65 80
```

Anonymizer

- Proxy server that acts as an intermediary and privacy shield between a client computer and Internet provider
- Use-cases include
 - Bypassing internet censors
 - Ensuring privacy
 - Having protection against online attacks
 - Bypassing IDS and firewall rules (of e.g. companies or schools)







Types of anonymizers

- Networked anonymizers
 - Information is transferred through network of computers
 - **Pros:** Harder to analyze traffic
 - **Cons:** Nodes in the sequence can be compromised
- Single-point anonymizers
 - Information is transferred through single website
 - **Pros:** Protects IP and related information
 - **Cons:** Less resistance to sophisticated analysis

Anonymizer tools

- OS
 - [Tails](#): Live OS to protect against surveillance and censorship
 - [Whonix](#): Designed to run in VMs to increase security and privacy
- Software
 - [Invisible Internet Project \(I2P\)](#)
 - [TunnelBear](#)
- Mobile apps
 - [Orbot](#): Proxy with Tor
 - [Psiphon](#): Tunnel through VPN on Android


Tor

- Free and open-source software for enabling anonymous communication
-  Stronger privacy than single [node proxies](#) or [proxy chains](#) without encryption
-  Anonymous as long as guard and exit nodes are not the same party and the traffic is over HTTPS
- VPN / Proxies (non-chained): Single point of failure
 - Provider knows who you are and what you browse
 - Provider may inject packets (e.g. ads) into traffic or record personal details.
- Tor traffic goes through at least 3 different servers (up to 5000) before destination
 - Provides separate encryption layer for each server
 - An observer cannot read or modify packets
 - Second server (middle relay) passes encrypted data without knowing more than its predecessor and descendant
 - Weaknesses
 -  A bad first server (guard node) knows your IP but not what you sent or where.
 -  A bad third server (exit node) sees your traffic but cannot know who sent the traffic.
 -  Exit node can eavesdrop if the packet is not using HTTPS but HTTP
 -  Will be no privacy if both the entry and exit nodes are hijacked by same hostile party.

I2P (Invisible Internet Project)

- Peer-to-peer alternative to [Tor](#)
- Each participant acts both as a client and as a router

Banner grabbing

- **Banner information** = name + version
-  Used to gain banner information about
 - a computer (e.g. OS information)
 - the services running on its open ports (e.g. nginx server)
- Allows attackers to exploit known vulnerabilities and form an attack plan.
- Can be prevented by disabling banners and by hiding web page extensions.

Passive banner grabbing

- Uses sniffing to determine the operating system.
- E.g. analyzing error messages, sniffing the traffic on the network, and examining page extensions.


Active banner grabbing

- Sending a packet to the OS and then analyzing the responses.
- E.g. each OS have different TCP/IP stack implementations.
 - Values of TTL (time to live) and TCP window size in the IP header of the first packet are different in each OS.
- HTTP (80), FTP (21), SMTP (25) are common ports.
- Tools include `telnet`, `nmap`, `zgrab` and `netcat`.

Banner grabbing tools

- `nmap -O` for OS automatic fingerprinting, see also [-O: OS fingerprinting.](#) | [Scanning tools](#)

Netcat

- Networking utility for reading from and writing to network connections using TCP or UDP
-  Also known as TCP/IP swiss army knife
- **FTP**
 - `nc 178.128.203.1 21`: Checks FTP port 21
 - 💡 You can use `nmap 178.128.203.1` to find out open ports.
 - Usual for FTP servers to return something like `Welcome to XXX FTP Server`
 - gives you insights about the owner and that it's a FTP server
- **HTTP**
 - E.g.
 1. `nc 178.128.203.1 80`
 2. Type e.g. `GET /index.html HTTP 1.0` or `HEAD / HTTP/1.1`
 3. Press `[ENTER]`
 - And also usually useful headers such as `Server: Apache/2.4.6 (CentOS)`
- [CryptCat](#) is the encrypted version of netcat

Telnet

- Application protocol for bidirectional interactive text-oriented communication using a virtual terminal connection
- Historically used to access to a command-line interface on a remote host but now replaced with [SSH](#) as it's unsecure.
- Built into many operative systems as default
- Runs on port 23 so `telnet <target>` sends TCP to targets port 23.
 - You can send to different ports using `telnet`
- If port is open a banner response received looking like:
 - `Server: Microsoft-IIS/5.0 Date: Fri, 14 Aug 2009 1:14:42 GMT Content-Length:340 Content-Type: text/html`