Vulnerabilità e Difesa dei Sistemi Internet

A.K.A. ETHICAL HACKING

FRANCESCO MANCINI – francesco.mancini@uniroma2.it

PASQUALE CAPORASO - pasquale.caporaso@uniroma2.it

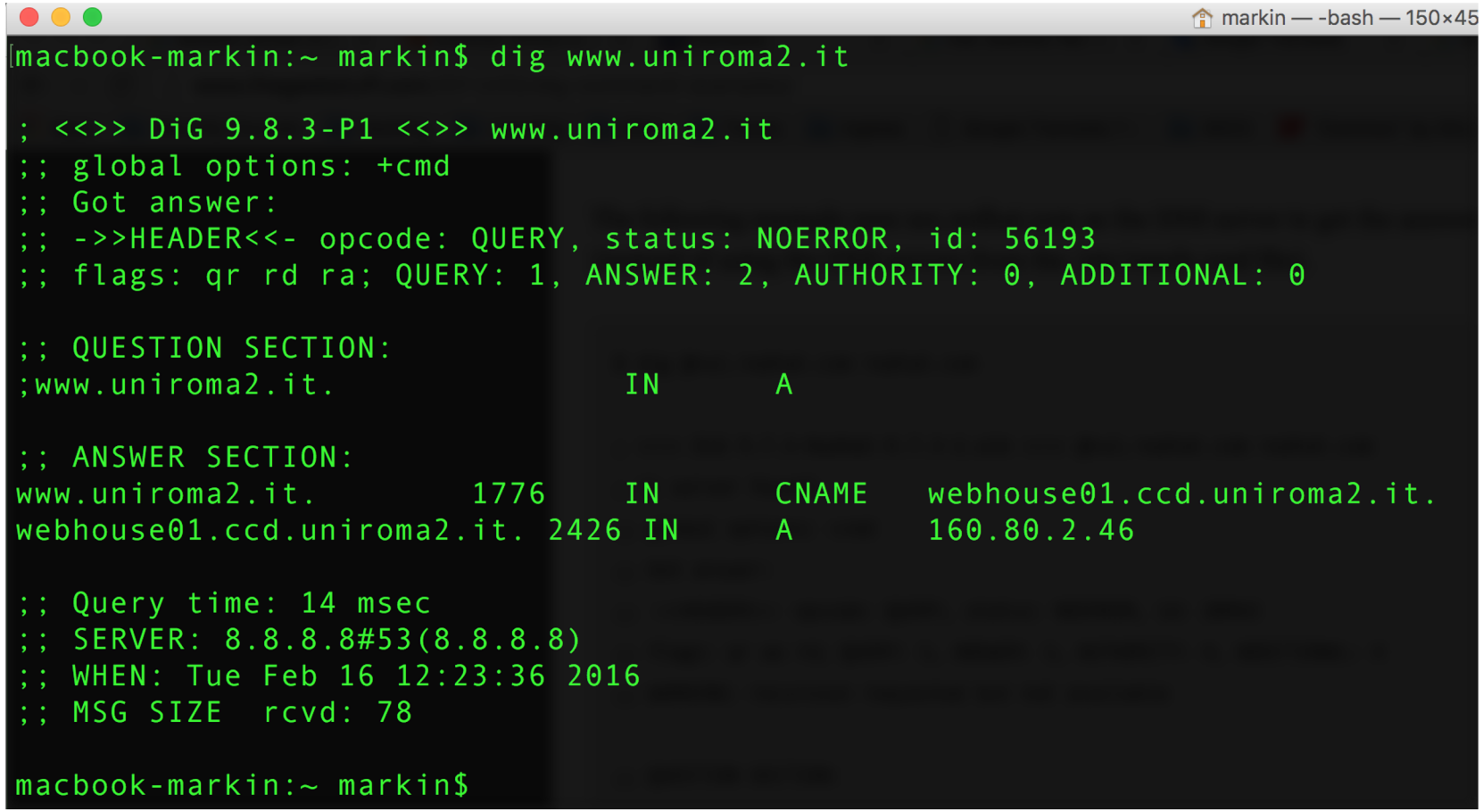
SARA DA CANAL – sara.da.canal@uniroma2.it

PIERCIRO CALIANDRO – pierciro.caliandro@uniroma2.it

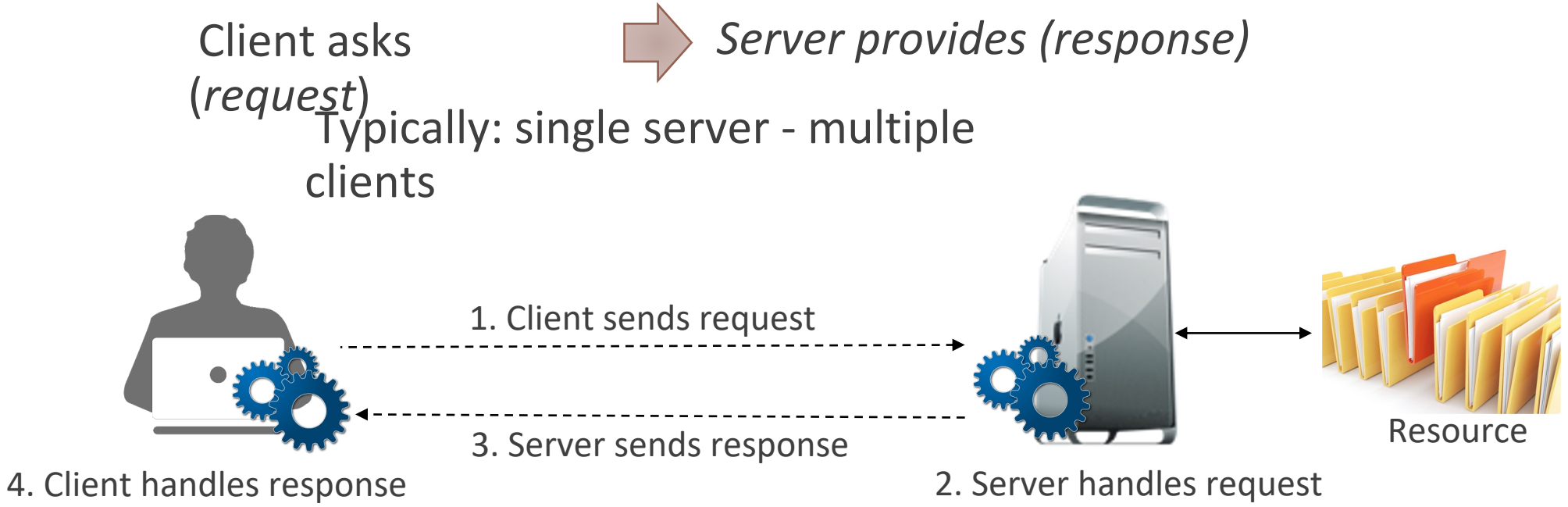
Vulnerabilità e Difesa dei Sistemi Internet

ENUMERATION

Domain name is translated to number



Client-Server Model Overview



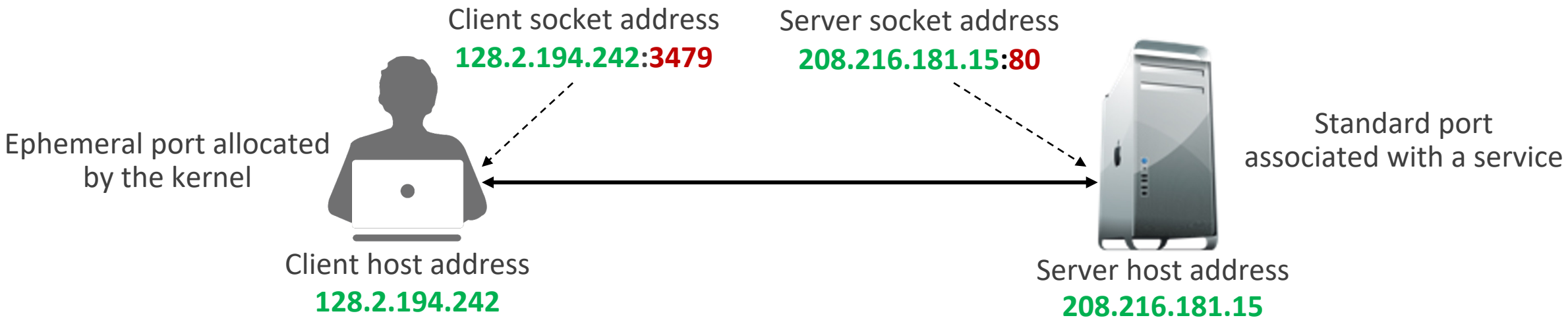
Clients and servers are processes running on physical host machines

Client-Server Model Overview

Address the machine on the network
- By IP address

Address the process
- By the "port" number

The pair of *IP-address + port* – makes up a "*socket-address*"



Client

Examples of client programs

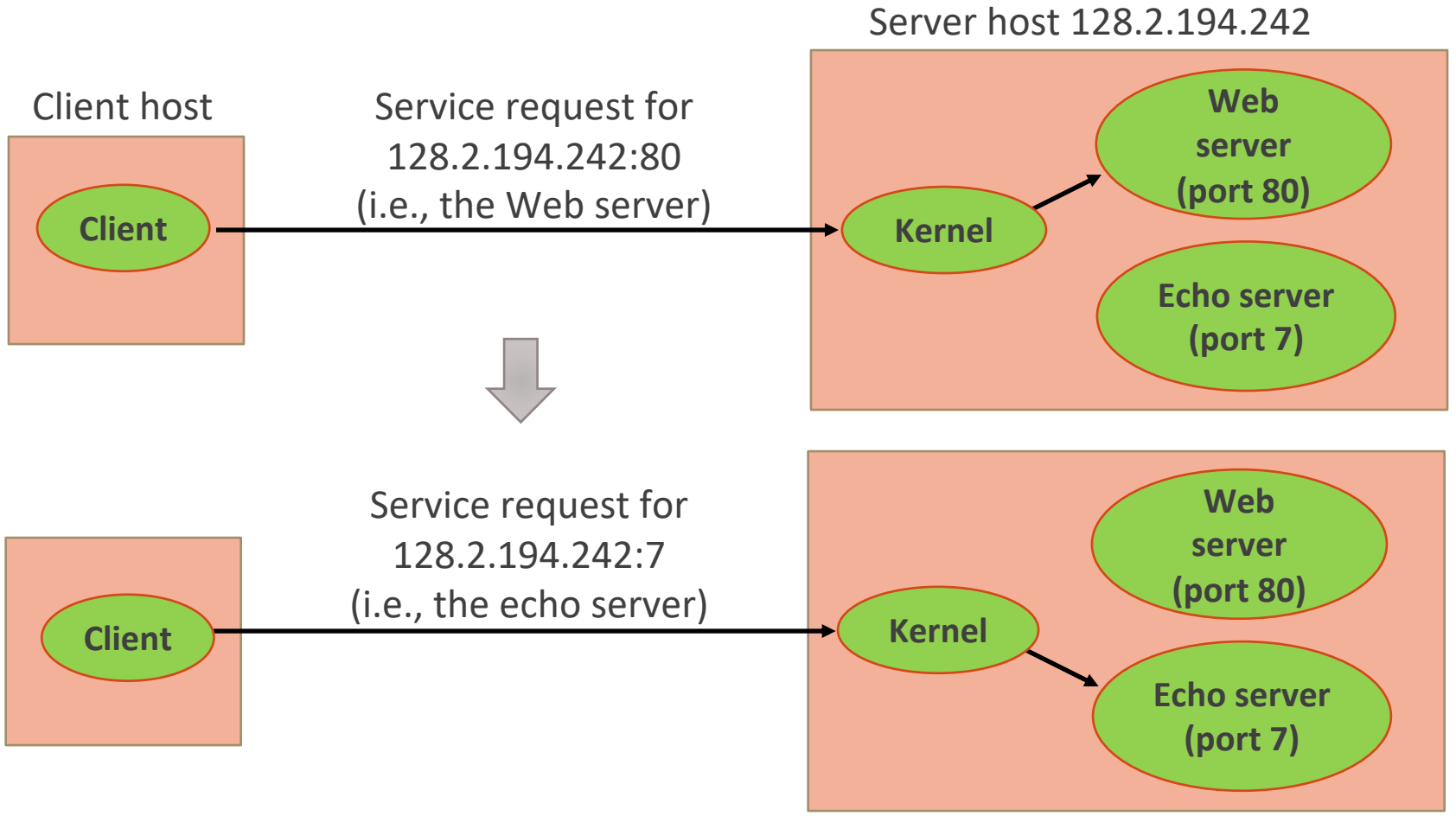
- Web browsers, ftp, telnet, ssh

How does a client find the server?

- The IP address in the server socket address identifies the physical host
- The (well-known) port in the server socket address identifies the service
  - Implicitly identifies the server process that performs that service

- Examples of well known ports:
  - **Port 7**: Echo server
  - **Port 23**: Telnet server
  - **Port 25**: Mail server
  - **Port 53**: DNS
  - **Port 80**: Web server

Using Ports to Identify Services



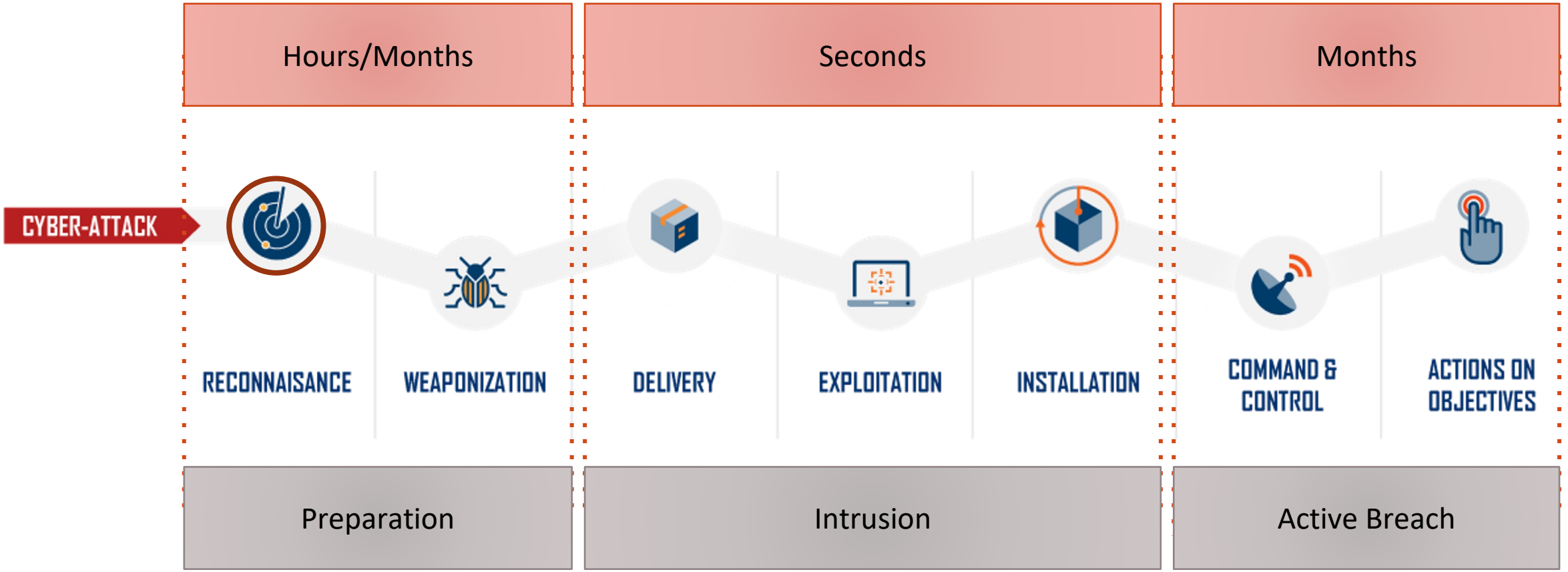
markin — -bash — 183×52

```
macbook-markin:~ markin$ sudo nmap -O -sV www.uniroma2.it

Starting Nmap 6.47 ( http://nmap.org ) at 2016-02-16 10:31 CET
Nmap scan report for www.uniroma2.it (160.80.2.46)
Host is up (0.00099s latency).
rDNS record for 160.80.2.46: webhouse01.ccd.uniroma2.it
Not shown: 994 filtered ports
PORT     STATE  SERVICE     VERSION
22/tcp   open   ssh         OpenSSH 6.0p1 Debian 4+deb7u2 (protocol 2.0)
80/tcp   open   http        Apache httpd 2.2.22
443/tcp  open   ssl/http    Apache httpd 2.2.22
2401/tcp closed cvspserver
3306/tcp open   mysql       MySQL 5.5.43-0+deb7u1-log
5432/tcp open   postgresql?
1 service unrecognized despite returning data. If you know the service/version, please submit the following fingerprint at http://www.insecure.org/cgi-bin/servicefp-submit.cgi :
SF-Port5432-TCP:V=6.47%I=7%D=2/16%Time=56C2EC82%P=x86_64-apple-darwin14.3.
SF:0%r(SMBProgNeg,90,"E\0\0\0\x8fSFATALE\0C0A000\0Mprotocollo\x20frontend\
SF:x20non\x20supportato\x2065363\.19778:\x20il\x20server\x20supporta\x20da
SF:\x201\.0\x20a\x203\.0\0Fpostmaster\.c\0L1622\0RProcessStartupPacket\0\0
SF:")%r(Kerberos,90,"E\0\0\0\x8fSFATALE\0C0A000\0Mprotocollo\x20frontend\x
SF:20non\x20supportato\x2027265\.28208:\x20il\x20server\x20supporta\x20da\
SF:x201\.0\x20a\x203\.0\0Fpostmaster\.c\0L1622\0RProcessStartupPacket\0\0"
SF:);
No exact OS matches for host (If you know what OS is running on it, see http://nmap.org/submit/ ).
TCP/IP fingerprint:
OS:SCAN(V=6.47%E=4%D=2/16%OT=22%CT=2401%CU=36235%PV=N%DS=4%DC=I%G=Y%TM=56C2
OS:ECB4%P=x86_64-apple-darwin14.3.0)SEQ(TI=Z%CI=I%TS=8)SEQ(SP=102%GCD=1%ISR
OS:=105%TI=Z%CI=I%II=I%TS=8)SEQ(SP=102%GCD=1%ISR=105%TI=Z%II=I%TS=8)OPS(O1=
OS:M5B4NNT11NW7%O2=M5B4NNT11NW7%O3=M5B4NNT11NW7%O4=M5B4NNT11NW7%O5=M5B4NNT1
OS:1NW7%O6=M5B4NNT11)WIN(W1=7120%W2=7120%W3=7120%W4=7120%W5=7120%W6=7120)EC
OS:N(R=Y%DF=Y%T=40%W=7210%O=M5B4NW7%CC=Y%Q=)T1(R=Y%DF=Y%T=40%S=O%A=S+%F=AS%
OS:RD=0%Q=)T2(R=N)T3(R=N)T4(R=Y%DF=Y%T=40%W=0%S=A%A=Z%F=R%O=%RD=0%Q=)T5(R=Y
OS:%DF=Y%T=40%W=0%S=Z%A=S+%F=AR%O=%RD=0%Q=)T5(R=N)T6(R=Y%DF=Y%T=40%W=0%S=A%
OS:A=Z%F=R%O=%RD=0%Q=)T6(R=N)T7(R=N)U1(R=Y%DF=N%T=40%IPL=164%UN=0%RIPL=G%RI
OS:D=G%RIPCK=G%RUCK=G%RUD=G)IE(R=Y%DFI=N%T=40%CD=S)

Network Distance: 4 hops
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

OS and Service detection performed. Please report any incorrect results at http://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 62.18 seconds
macbook-markin:~ markin$
```

Reconnaissance

NETWORK MAPPING AND PORT SCANNING

How a cyber-attack starts?



The Killchain model

Step 1: Reconnaissance. The attacker gathers information on the target before the actual attack starts. He can do it by looking for publicly available information on the Internet.

Step 2: Weaponization. The attacker uses an exploit and creates a malicious payload to send to the victim. This step happens at the attacker side, without contact with the victim.

Step 3: Delivery. The attacker sends the malicious payload to the victim by email or other means, which represents one of many intrusion methods the attacker can use.

Step 4: Exploitation. The actual execution of the exploit, which is, again, relevant only when the attacker uses an exploit.

Step 5: Installation. Installing malware on the infected computer is relevant only if the attacker used malware as part of the attack, and even when there is malware involved, the installation is a point in time within a much more elaborate attack process that takes months to operate.

Step 6: Command and control. The attacker creates a command and control channel in order to continue to operate his internal assets remotely. This step is relatively generic and relevant throughout the attack, not only when malware is installed.

Step 7: Exfiltration. The attacker performs the steps to achieve his actual goals inside the victim's network. This is the elaborate active attack process that takes months, and thousands of small steps, in order to achieve.

Reconnaissance

"The attacker gathers information on the target before the actual attack starts. He can do it by looking for publicly available information on the Internet."

Passive Information Gathering / Open Source Intelligence (OSINT)

- Finding, selecting, and acquiring information from publicly available sources
- Never sending any traffic to the target organization neither from one of our hosts or "anonymous" hosts or services across the Internet
- Can only use and gather archived or stored information
- Can be out of date or incorrect as we are limited to results gathered from a third party

Active Information Gathering

- During this stage we are actively mapping network infrastructure
- Actively enumerating and/or vulnerability scanning the open services
- Actively searching for unpublished directories, files, and servers

Vulnerability Assessment and Penetration Testing

ACTIVE RECONNAISSANCE: PORT SCANNING

Scanning the Network

Once a hacker knows of the existence of a network, the next step is to map the network

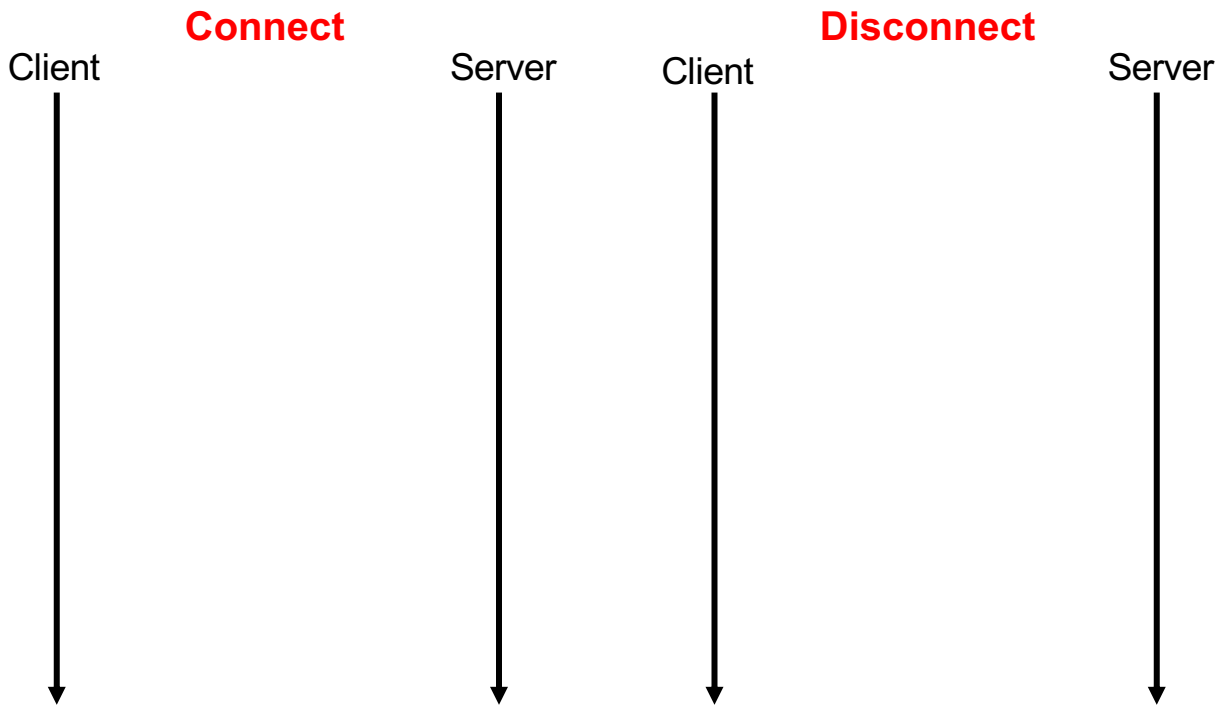
- To get a list of systems, hackers scan using a variety of tools and techniques

Not for just hackers, scanning a network can be done by the network administrator to:

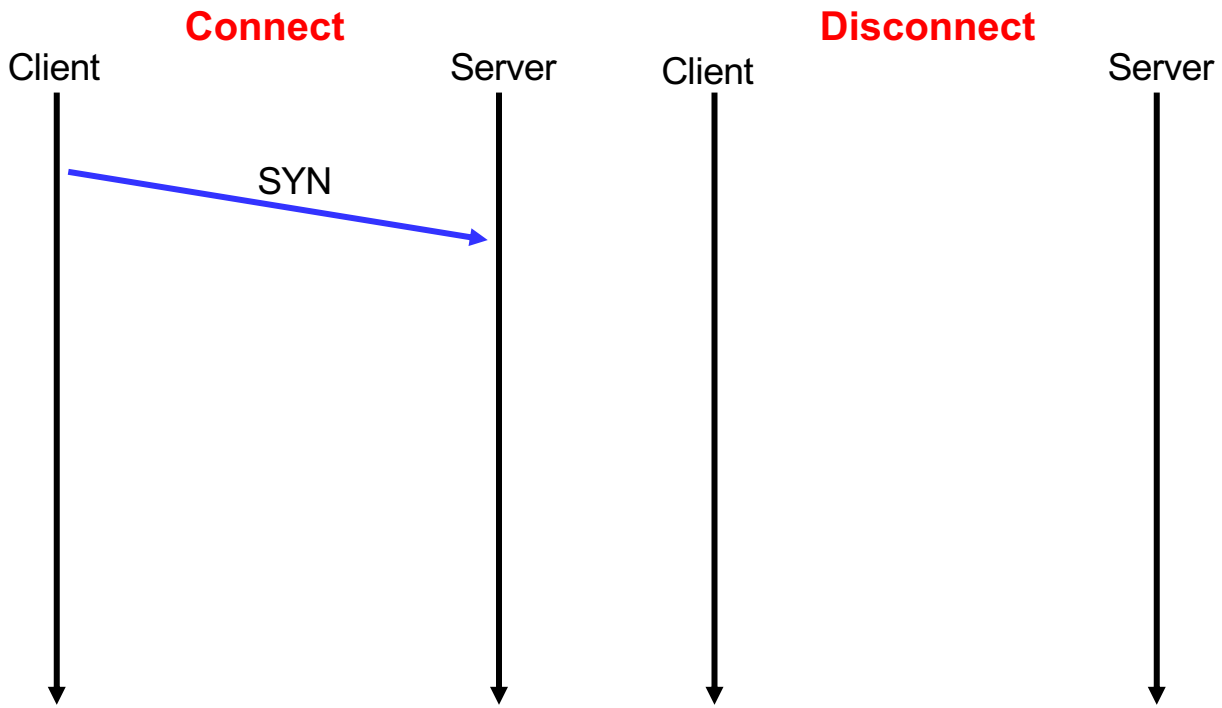
- Appreciate the hacker's perspective on the network
- Practice with, and gain an understanding of, common scanning tools
- Stress the monitoring mechanisms such as IDS
- Document the layout of the network
- Audit access control devices on the network, host configurations, and so forth

Nmap, Zenmap, Unicorn, Netcat, Telnet, …

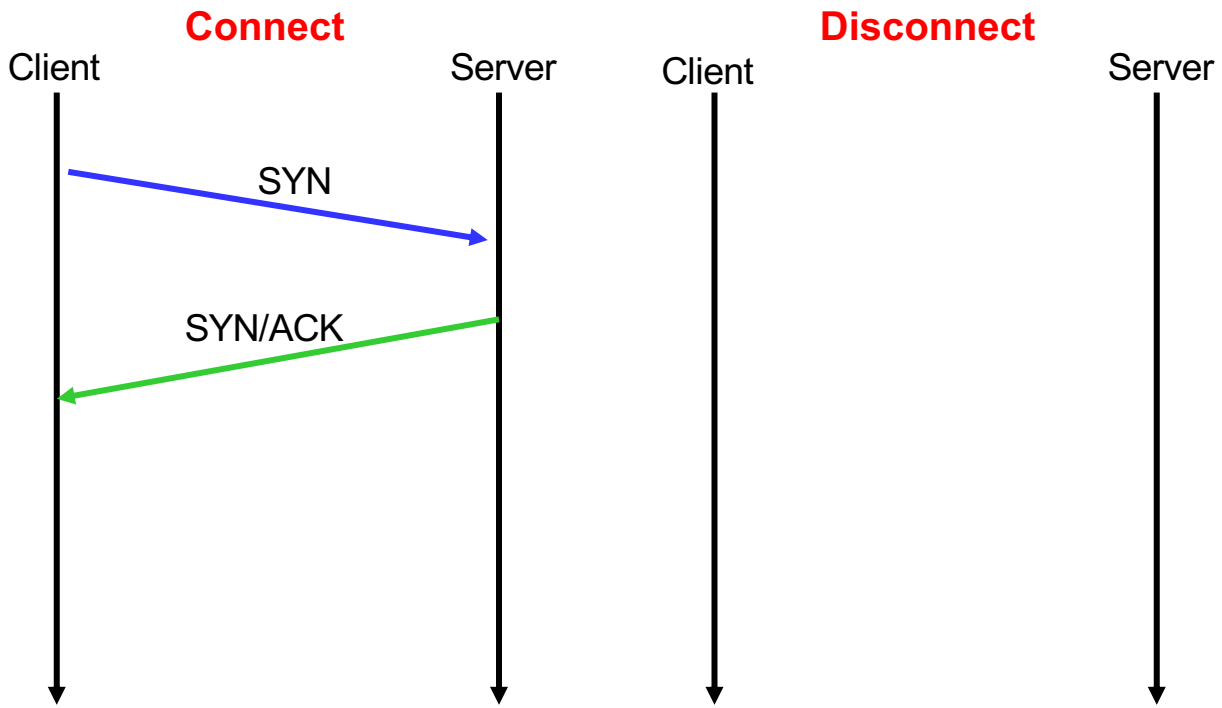
TCP Three-way handshake



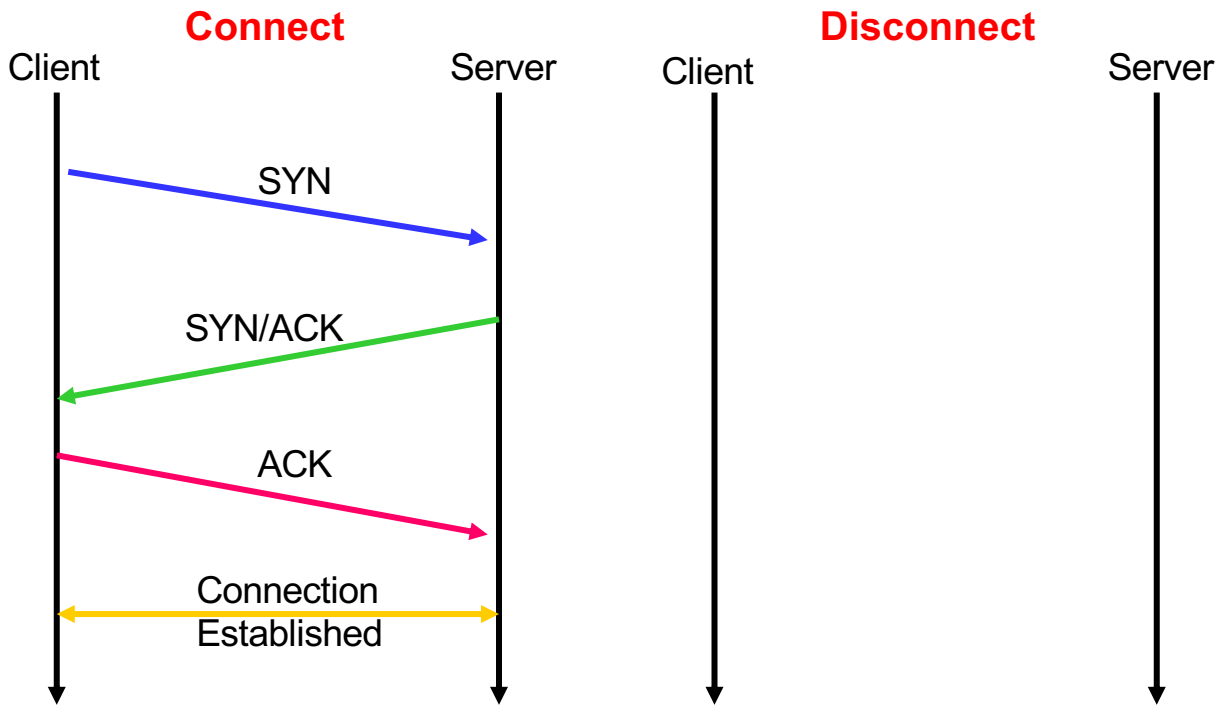
TCP Three-way handshake



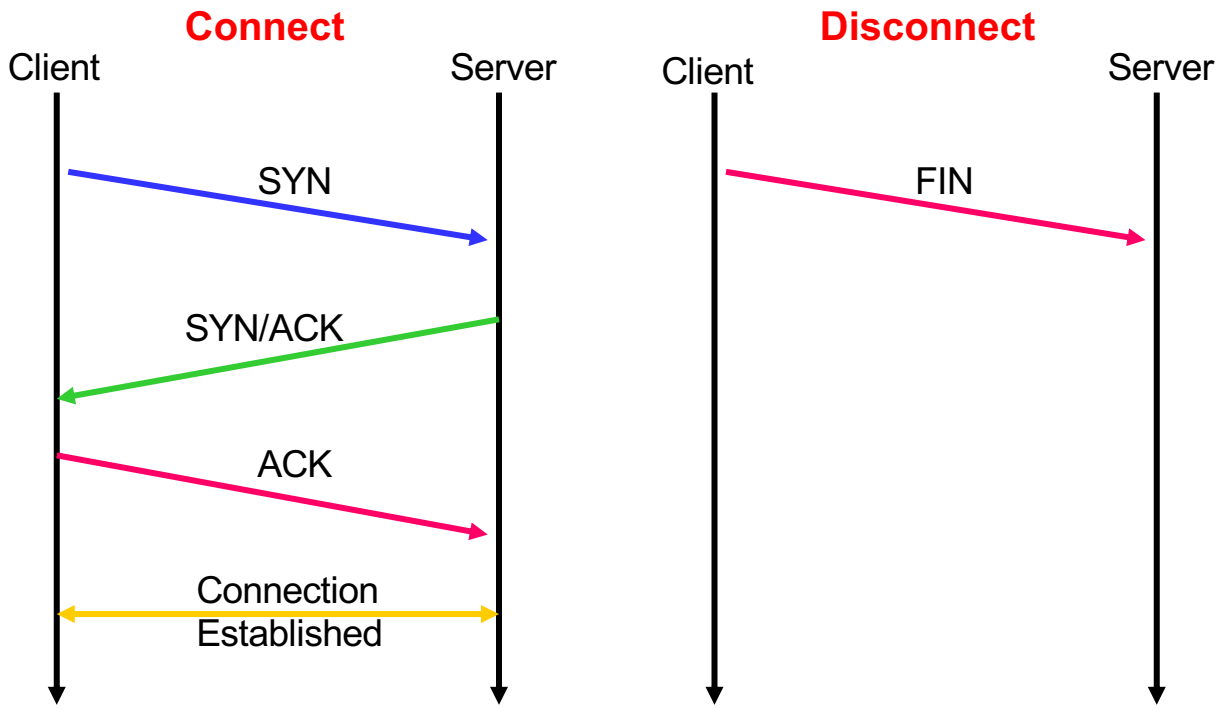
TCP Three-way handshake



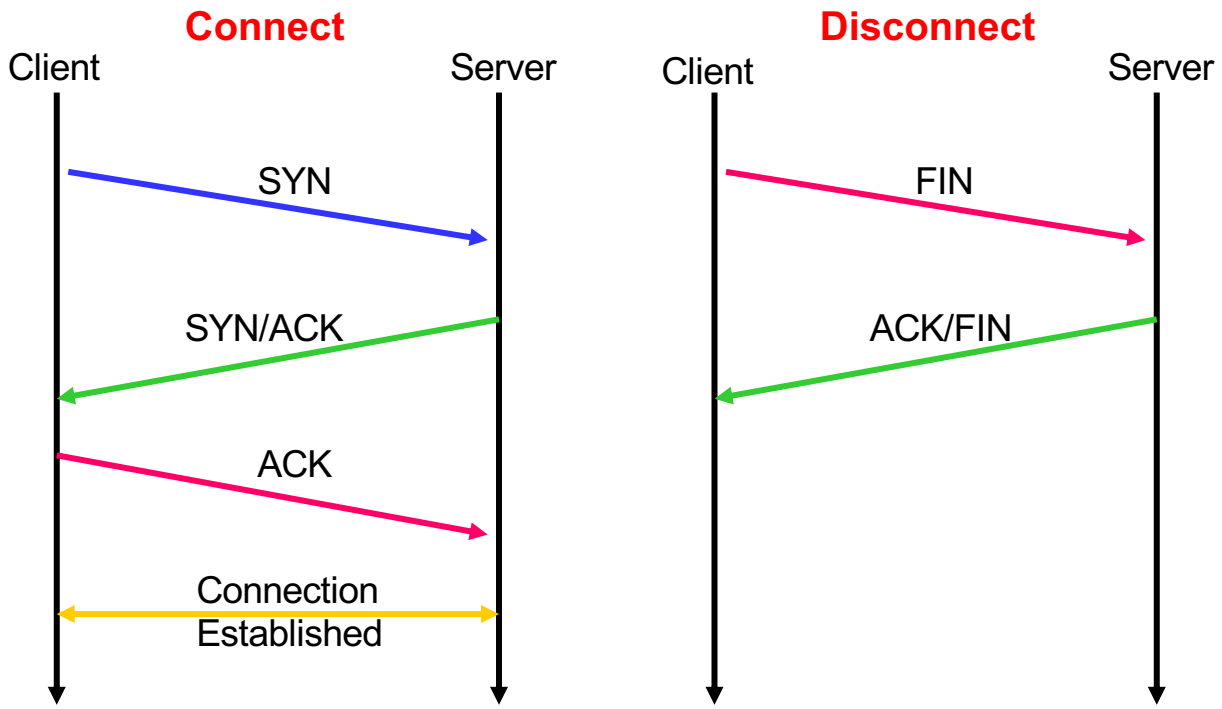
TCP Three-way handshake



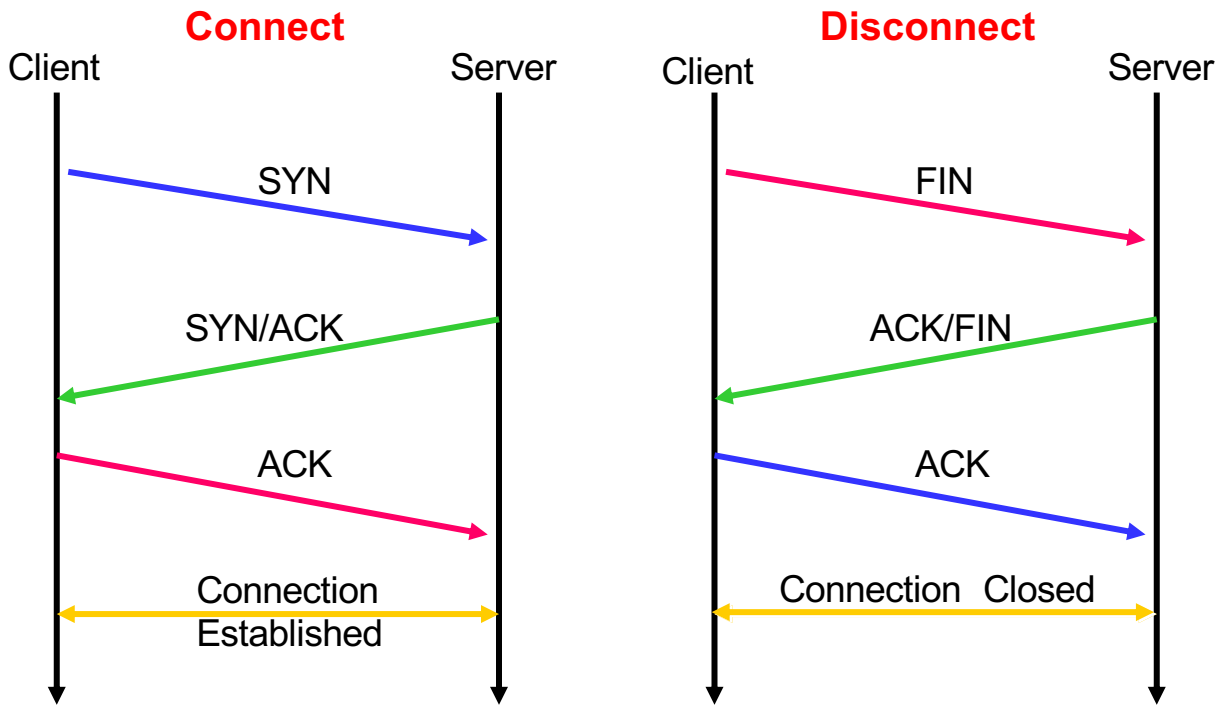
TCP Three-way handshake



TCP Three-way handshake



TCP Three-way handshake

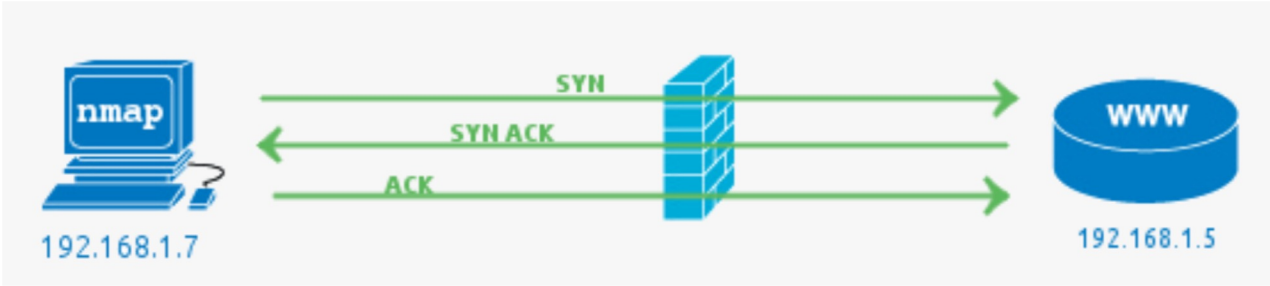


TCP SYN SCAN

Relies on the three way TCP handshake mechanism

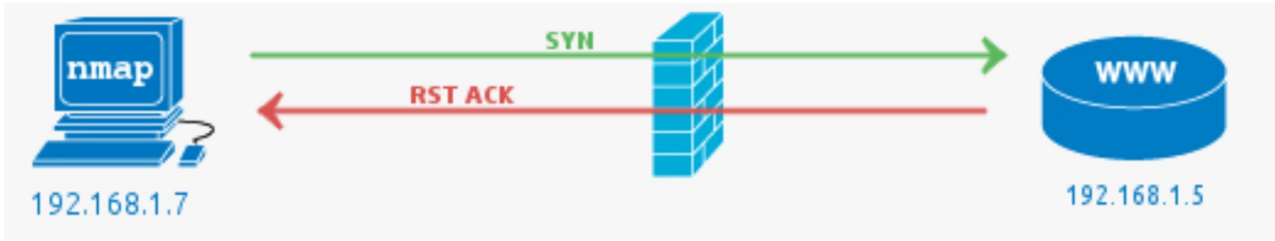
- Two hosts attempting to communicate can negotiate the parameters of the network TCP socket connection before transmitting data
- Attempt to complete a three way handshake with the target host on the specified port(s)
- If the handshake is completed, this indicates that the port is open

- **nc -n -vv -w 1 -z <ip address> <port range>**



TCP SYN SCAN

```
root@kali:~# nc -nvv -w 1 -z 10.0.0.19 3388-3390
(UNKNOWN) [10.0.0.19] 3390 (?) : Connection refused
(UNKNOWN) [10.0.0.19] 3389 (?) open
(UNKNOWN) [10.0.0.19] 3388 (?) : Connection refused
 sent 0, rcvd 0
```



No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	10.0.0.18	10.0.0.19	TCP	74	34838 > 3390 [SYN] Seq=0 Win=14600 Len=0 MSS=146
2	0.000307000	10.0.0.19	10.0.0.18	TCP	60	3390 > 34838 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
4	0.001135000	10.0.0.19	10.0.0.18	TCP	74	3389 > 48852 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=
5	0.001161000	10.0.0.18	10.0.0.19	TCP	66	48852 > 3389 [ACK] Seq=1 Ack=1 Win=14720 Len=0 TS
6	0.001365000	10.0.0.18	10.0.0.19	TCP	66	48852 > 3389 [FIN, ACK] Seq=1 Ack=1 Win=14720 Len
7	0.001756000	10.0.0.19	10.0.0.18	TCP	66	3389 > 48852 [ACK] Seq=1 Ack=2 Win=66560 Len=0 TS
9	0.001957000	10.0.0.19	10.0.0.18	TCP	60	3388 > 47524 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
10	0.005714000	10.0.0.19	10.0.0.18	TCP	60	3389 > 48852 [RST, ACK] Seq=1 Ack=2 Win=0 Len=0

UDP SCAN

The mechanism behind UDP port scanning is different

- UPD does not involve a three way handshake
- UDP is stateless
- Try using **wireshark** while UDP scanning a lab machine with **netcat** to understand the how UDP port scans work.

- **nc -nv -u -z -w 1 <ip address> <port range>**

An empty UDP packet is sent to a specific port

- If the UDP port is open, no reply is sent back from the target machine
- If the UDP port is closed, an ICMP port unreachable packet should be sent back from the target machine

Be careful when scanning

UDP port scanning is often unreliable, as firewalls and routers may drop ICMP packets. This can lead to false positives in your scan, and you will regularly see UDP port scans showing all UDP ports open on a scanned machine.

Most port scanners do not scan all available ports, and usually have a preset list of “interesting ports” that are scanned.

People often forget to scan for UDP services, and stick only to TCP scanning, thereby seeing only half of the equation.

Please note that port scanning is illegal in many countries and should not be performed outside the labs

Nmap

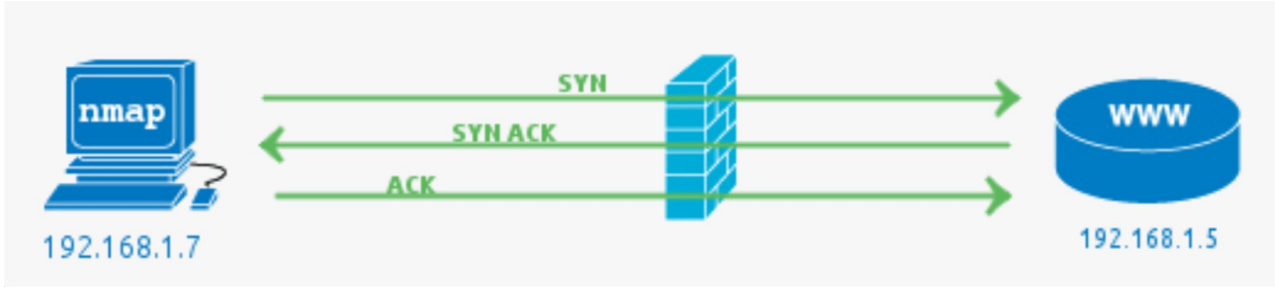
Nmap is one of the most popular, versatile, and robust port scanners to date. It has been actively developed for over a decade, and has numerous features beyond port scanning

- By experimenting with various scan options and a variety of devices you gain a sense of what devices present these network postures
- A default **nmap** TCP scan will scan the 1000 most popular ports on a given machine

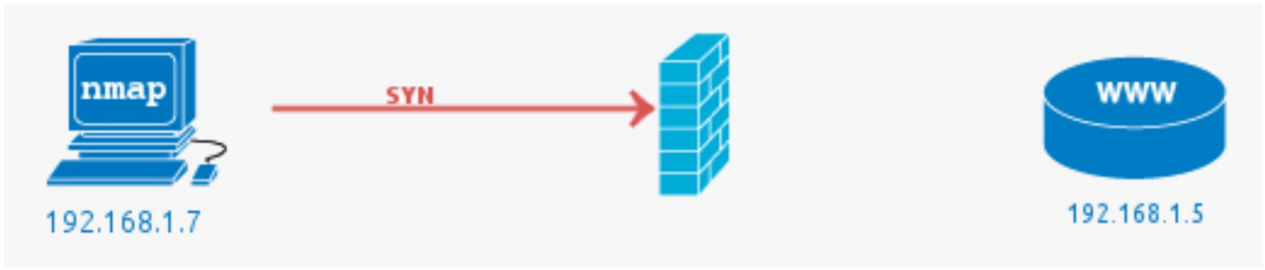
- **nmap <scantype> <options> <ip address>**

- We can have 3 different states:
  - Open
  - Filtered
  - Close

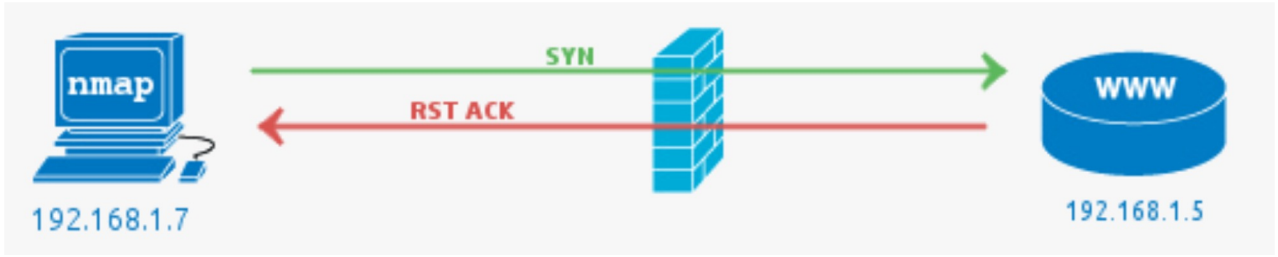
Port states



OPEN



FILTERED



CLOSED

Scan for Specific Port or Port Range

Sometimes we are looking for a specific port or a port range

- Nmap uses the **-p** switch to designate a port or port range
- If we were only looking for ports 25-150, we could use:

- **nmap <ip address> -p25-150**

Scanning a Subnet

If we want to scan more than a single IP address

- Nmap allows us to use CIDR notation to designate an entire subnet
- If we wanted to scan on entire Class C subnet (256 hosts) for port 80 open, we could type:

- **nmap <network address>/24 -p80**

Spoofing & Decoy Scan

We often want to hide our IP (our identity)

- Every packet must contain our source address or else the response from the target system will not know where to return to
- We **can** spoof our IP address (**-S**) any response and any info we are trying to gather will return to the spoofed IP
  - **Not very useful, if we are scanning for info gathering**
- We can obfuscate (**-D**) our IP address among many IP addresses
- The network/security admin can't pinpoint the source of the scan
  - Nmap allows us to use decoy IP addresses so that it looks like many IP addresses are scanning the target

- **nmap <ip address> -D <fake ip list>**

Basic Firewalls Evasion

Nmap to scan a system or network, by default, it send out a ping to see if the host is up

- Many firewalls and routers block or drop the ICMP (echo request, echo reply) ping
- We can suppress nmap's default behavior of sending out that initial ping and get past the firewall that is blocking us.
- We can do this by using the **-P0** switch

- **nmap -P0 <ip address>**

```
root@kali:~# nmap -sS -P0 192.168.89.191

Starting Nmap 6.40 ( http://nmap.org ) at 2014-09-05 16:30 EDT
mass_dns: warning: Unable to determine any DNS servers. Reverse DNS is disabled.
 Try using --system-dns or specify valid servers with --dns-servers
Nmap scan report for 192.168.89.191
Host is up (0.00072s latency).
Not shown: 982 closed ports
```

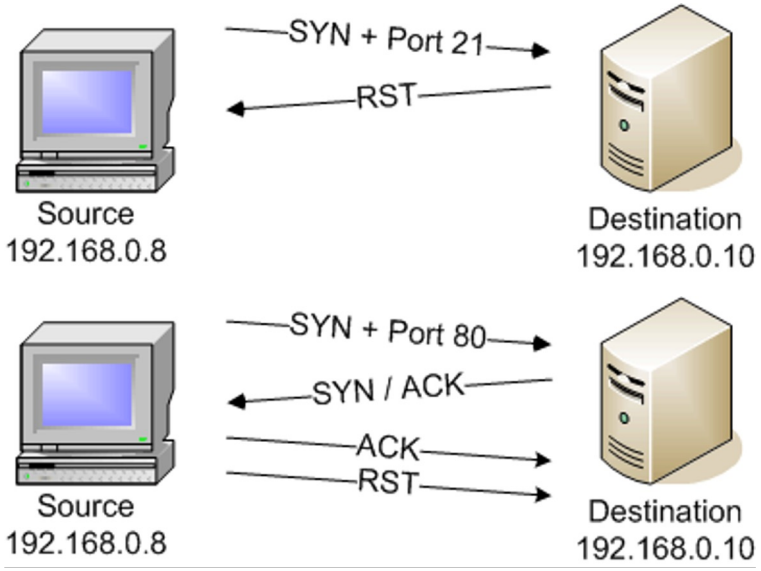
Gathering Version Info

When nmap runs a port scan,

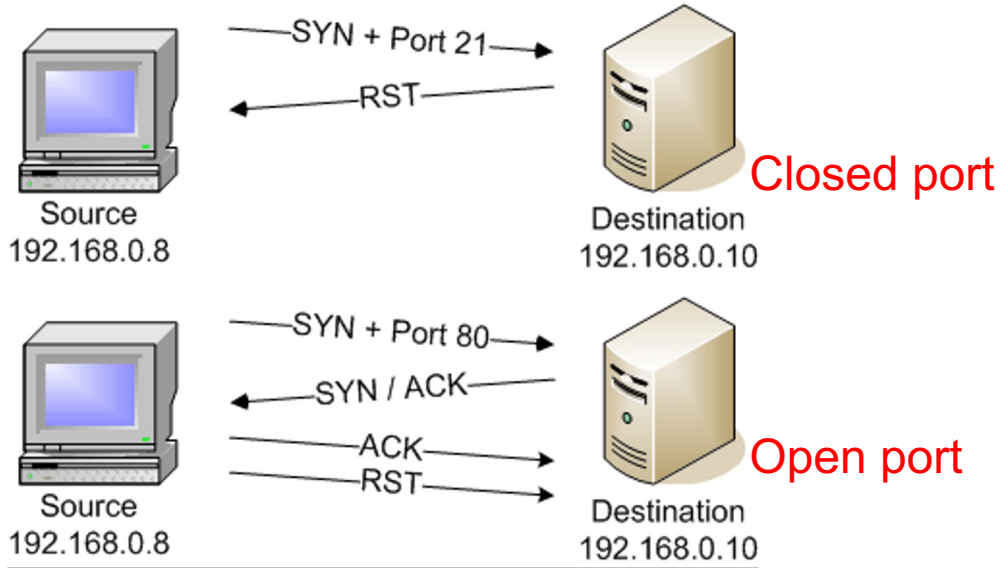
- it retrieves the port info (open/closed/filtered) and then gives us the **default** service that is running on that port
- As one can run ANY service on any port, that may not be adequate information
- We need to know what service is actually running on that port, not the default service.
- For instance, knowing that port 80 is open and running http is good to know, but if our attack is specific to Apache, then if the target has Microsoft's IIS running on that port, it won't work
- Nmap has a feature that interrogates the service running on each port scanned. It can be used with the -V switch, such as:

- **nmap -sV <ip address>**

TCP Connect Scan (-sT)



TCP Connect Scan (-sT)



TCP Connect Scan: Disadvantages

The disadvantage of this scan is that it’s a very noisy scan

- Check application connection logs
- Since the TCP connect() scan is completing a TCP connection, normal application processes immediately follow
- These applications are immediately met with a RST packet
  - but the application has already provided the appropriate login screen or introductory page
- By the time the RST is received
  - the application initiation process is already well underway and additional system resources are used

It might be considered the TCP scan of last resort

If privileged access isn't available

- Connect scan may be the only method available!

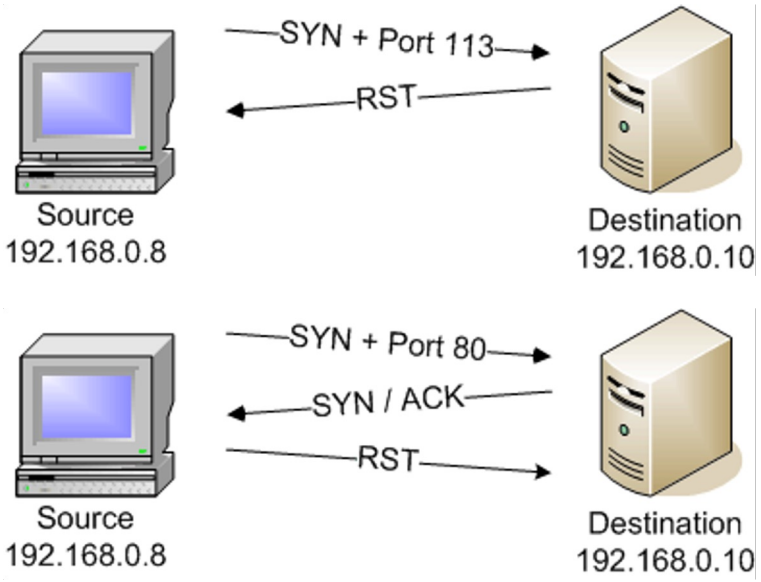
TCP SYN Scan (-sS)

The TCP SYN scan uses common methods of port-identification that allow nmap to gather information about open ports without completing the TCP handshake process

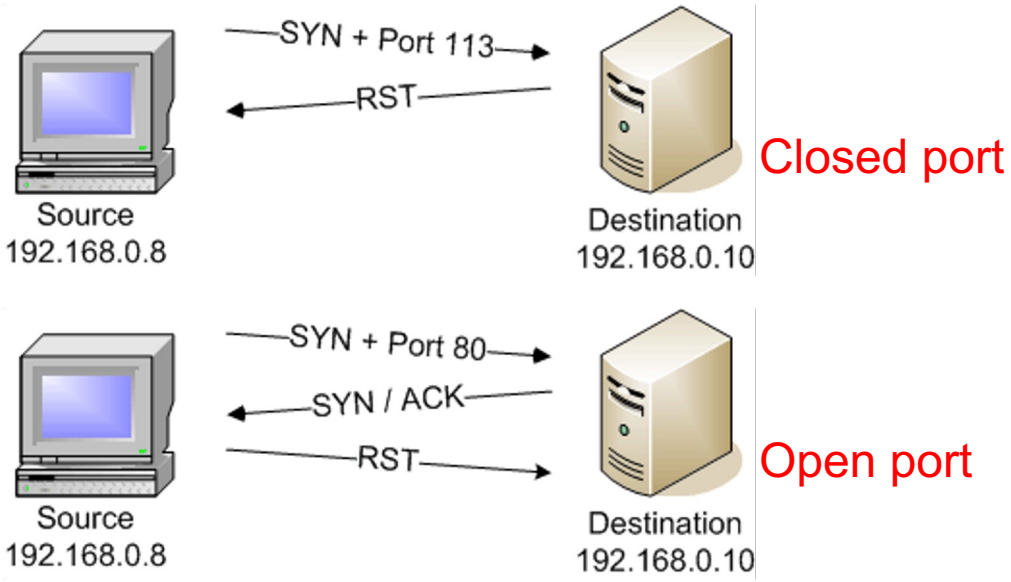
When an open port is identified

- The TCP handshake is reset before it can be completed
- This technique is often referred to as "**half open**" scanning

TCP SYN Scan


SYN + Port 113
RST
Source
192.168.0.8
Destination
192.168.0.10
SYN + Port 80
SYN / ACK
RST
Source
192.168.0.8
Destination
192.168.0.10

TCP SYN Scan


SYN + Port 113
RST
Source
192.168.0.8
Destination
192.168.0.10
Closed port
SYN + Port 80
SYN / ACK
RST
Source
192.168.0.8
Destination
192.168.0.10
Open port

TCP SYN Scan: Advantages

The TCP SYN scan never actually creates a TCP session

- so may be not logged by the destination host's applications

This is a much "quieter" scan than the TCP connect() scan

- Less visibility in the destination system's application logs since no sessions are ever initiated

Since an application session is never opened

- SYN scan is also less stressful to the application service

Stealth Scans

FIN Scan (-sF), Xmas Scan (-sX), and NULL Scan (-sN)

Called "stealth" scans because they send a single frame to a TCP port without any TCP handshaking or additional packet transfers

- This is a scan type that sends a single frame with the expectation of a single response

These scans operate by manipulating the bits of the TCP header to induce a response from the remote station

- The forged packet is an invalid packet (does not work on Windows)

Except for the FIN scan, nmap creates TCP headers that combine bit options that should never occur in the real world

Stealth Scans

TCP protocol states that stations receiving information on a closed TCP port should send a RST frame and an available TCP port should not respond at all

During any of these stealth scans, nmap categorizes the responses as either closed, or open|filtered

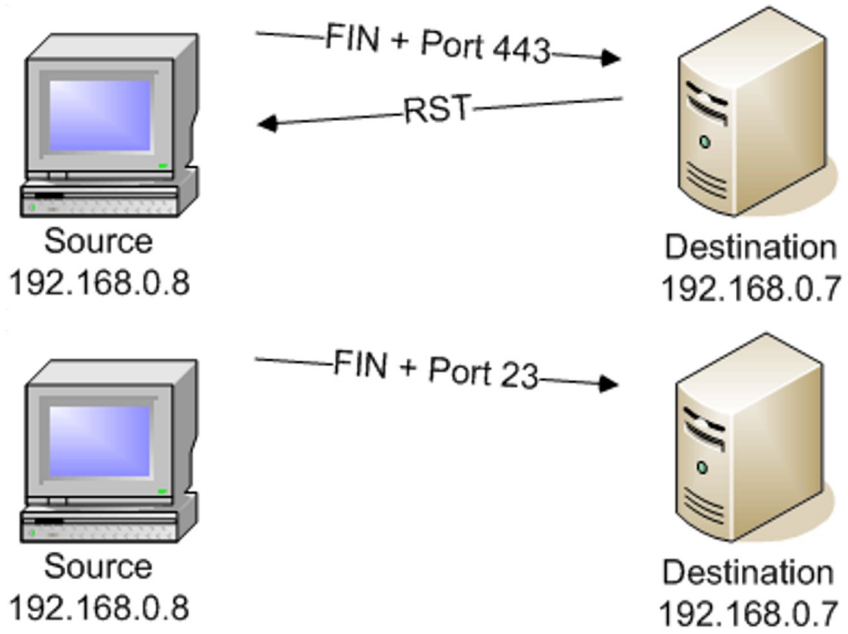
It's impossible to determine if a missing response was due to an open port or a filtered network connection

- **there's no way to differentiate between an open port and an administratively dropped frame**.

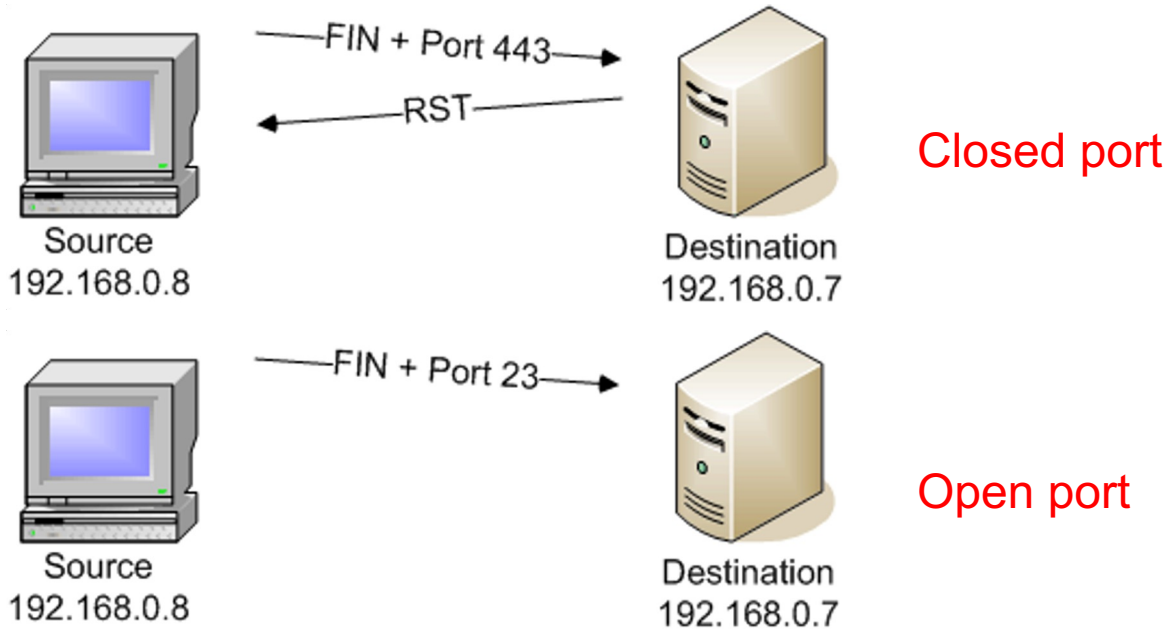
Because of these "**customized**" packets

- nmap **requires privileged access** to perform stealth scans

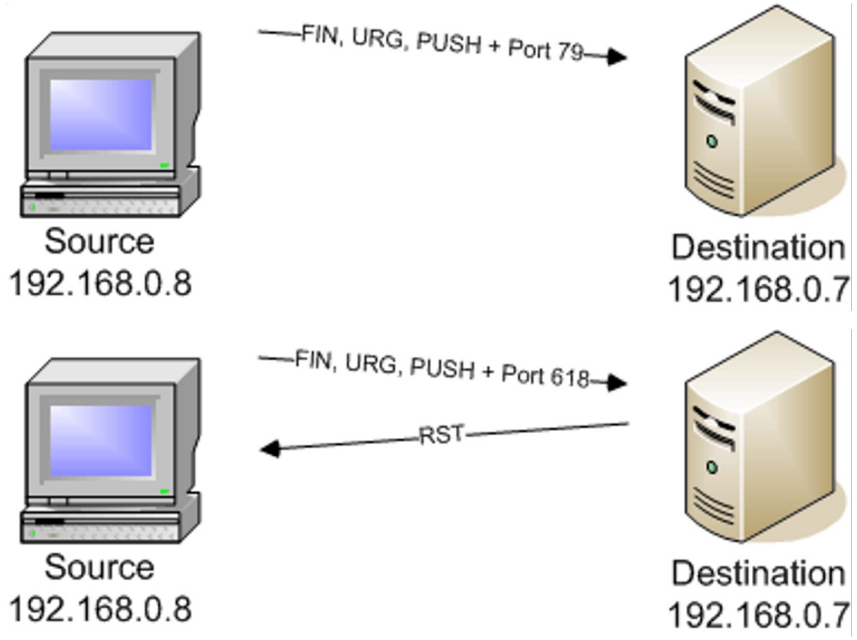
FIN Scan (-sF)


FIN + Port 443
RST
Source
192.168.0.8
Destination
192.168.0.7
FIN + Port 23
Source
192.168.0.8
Destination
192.168.0.7

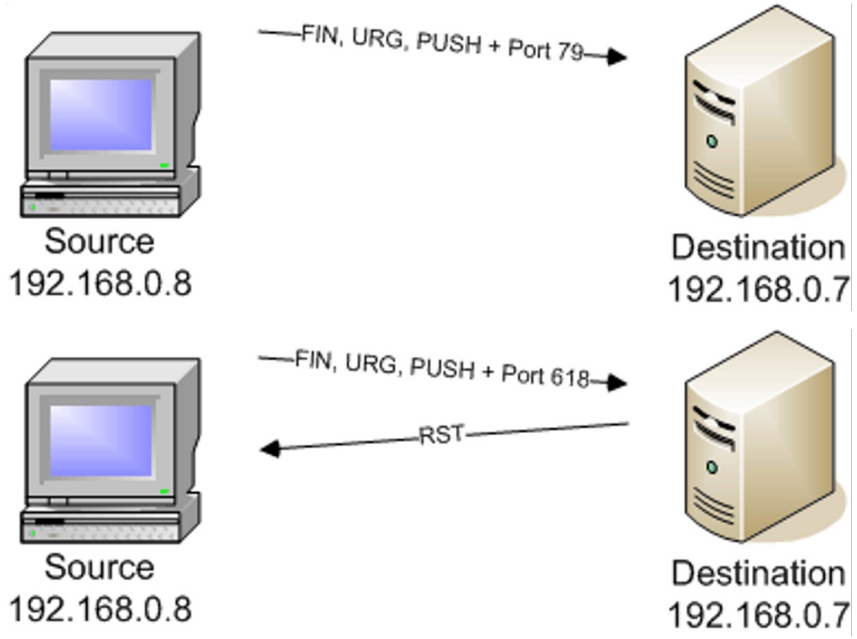
FIN Scan (-sF)


FIN + Port 443
RST
Source
192.168.0.8
Destination
192.168.0.7
Closed port
FIN + Port 23
Source
192.168.0.8
Destination
192.168.0.7
Open port

XMas Scan (-sX)


FIN, URG, PUSH + Port 79
Source
192.168.0.8
Destination
192.168.0.7
FIN, URG, PUSH + Port 618
RST
Source
192.168.0.8
Destination
192.168.0.7

XMas Scan (-sX)

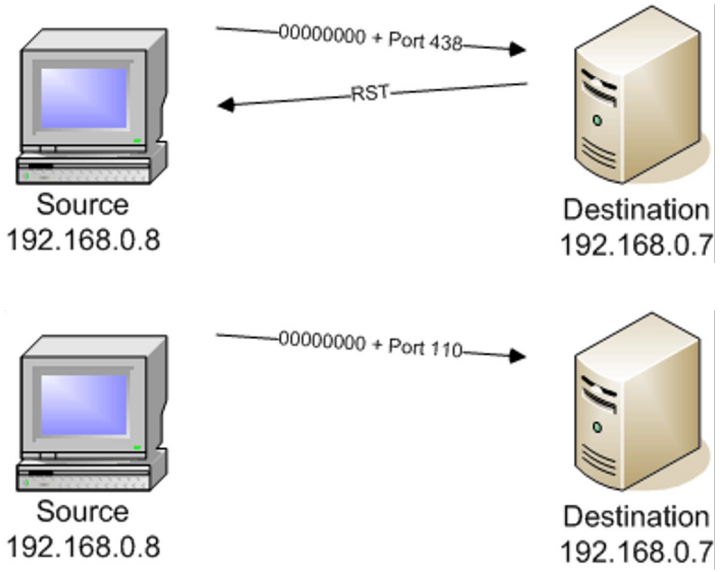


Open port

Closed port

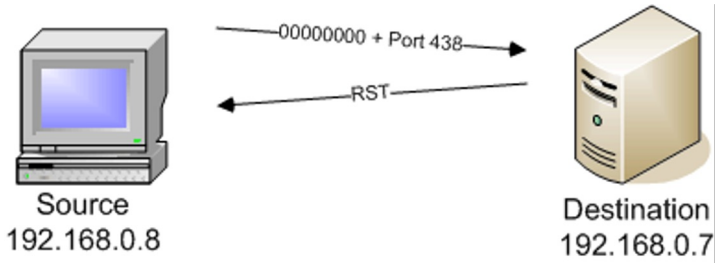
NULL Scan (-sN)

The null scan turns off all flags, creating a lack of TCP flags that should never occur in the real world.



NULL Scan (-sN)

The null scan turns off all flags, creating a lack of TCP flags that should never occur in the real world.



Closed port

Open port

Advantages and Disadvantages of Stealth Scans

Since no TCP sessions are created for any of these scans

- Remarkably quiet from the perspective of the remote device's applications
- None of these scans should appear in any of the application logs

Most minimal port-level scans

- Require 2 packet transfer for a closed port
- Require 1 packet transfer to find an open port

Advantages and Disadvantages of Stealth Scans

On a Windows-based computer

- All ports will appear to be closed regardless of their actual state
- Any device showing open ports must not be a Windows-based device

The user running stealth scans needs to have root privilege

When to Use Stealth Scans

Although TCP SYN scans are relatively subtle

- The FIN, Xmas tree, and Null scans are even more invisible on the network

Don't show up in application log files

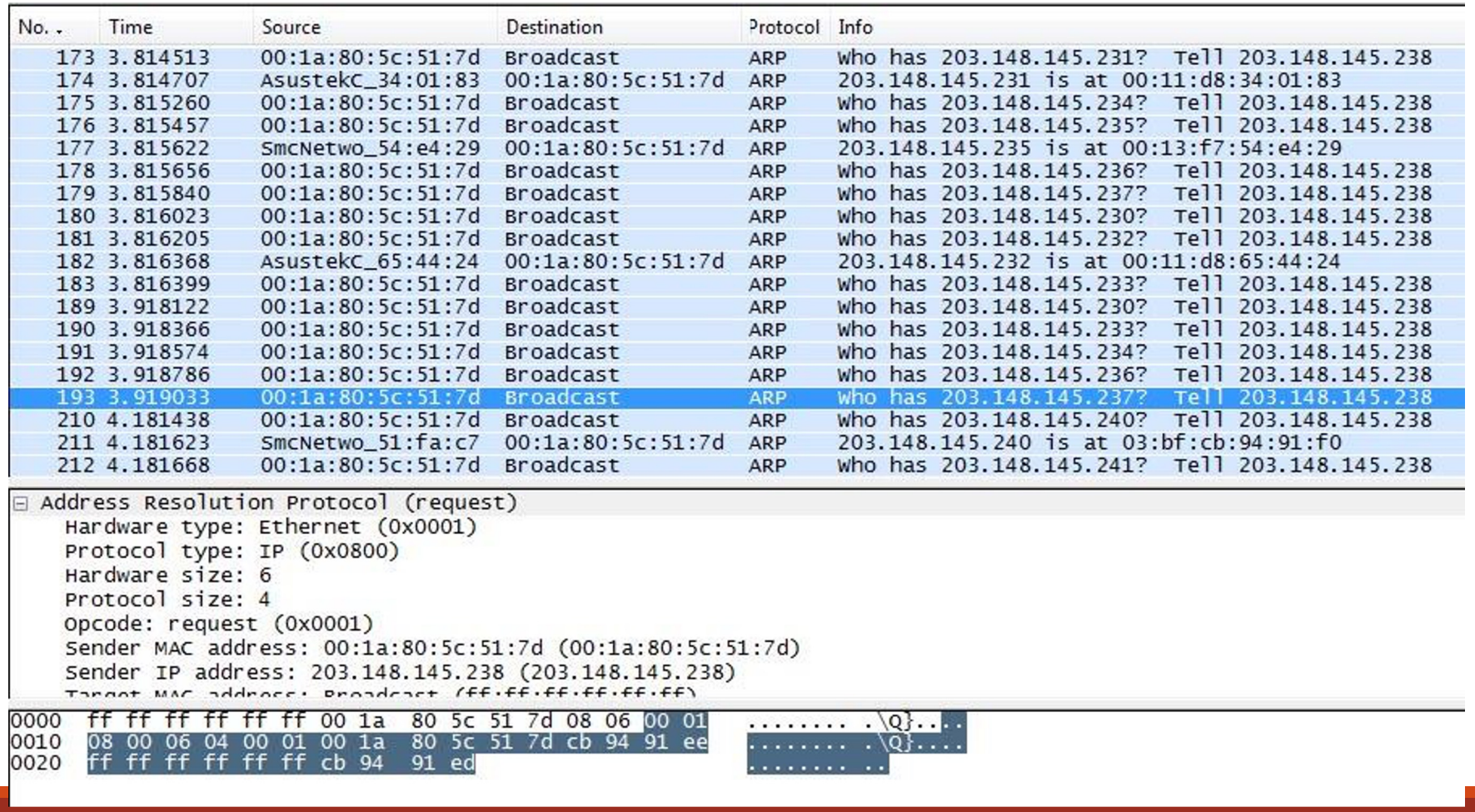
Take little network bandwidth

Provide extensive port information on non-Windows based systems

If the scanned device is susceptible to these odd TCP packets

- Information can be gathered with only a whisper of network communication!

Nmap –sP (same subnet)



No.	Time	Source	Destination	Protocol	Info
173	3.814513	00:1a:80:5c:51:7d	Broadcast	ARP	who has 203.148.145.231? Tell 203.148.145.238
174	3.814707	AsustekC_34:01:83	00:1a:80:5c:51:7d	ARP	203.148.145.231 is at 00:11:d8:34:01:83
175	3.815260	00:1a:80:5c:51:7d	Broadcast	ARP	who has 203.148.145.234? Tell 203.148.145.238
176	3.815457	00:1a:80:5c:51:7d	Broadcast	ARP	who has 203.148.145.235? Tell 203.148.145.238
177	3.815622	SmcNetwo_54:e4:29	00:1a:80:5c:51:7d	ARP	203.148.145.235 is at 00:13:f7:54:e4:29
178	3.815656	00:1a:80:5c:51:7d	Broadcast	ARP	who has 203.148.145.236? Tell 203.148.145.238
179	3.815840	00:1a:80:5c:51:7d	Broadcast	ARP	who has 203.148.145.237? Tell 203.148.145.238
180	3.816023	00:1a:80:5c:51:7d	Broadcast	ARP	who has 203.148.145.230? Tell 203.148.145.238
181	3.816205	00:1a:80:5c:51:7d	Broadcast	ARP	who has 203.148.145.232? Tell 203.148.145.238
182	3.816368	AsustekC_65:44:24	00:1a:80:5c:51:7d	ARP	203.148.145.232 is at 00:11:d8:65:44:24
183	3.816399	00:1a:80:5c:51:7d	Broadcast	ARP	who has 203.148.145.233? Tell 203.148.145.238
189	3.918122	00:1a:80:5c:51:7d	Broadcast	ARP	who has 203.148.145.230? Tell 203.148.145.238
190	3.918366	00:1a:80:5c:51:7d	Broadcast	ARP	who has 203.148.145.233? Tell 203.148.145.238
191	3.918574	00:1a:80:5c:51:7d	Broadcast	ARP	who has 203.148.145.234? Tell 203.148.145.238
192	3.918786	00:1a:80:5c:51:7d	Broadcast	ARP	who has 203.148.145.236? Tell 203.148.145.238
193	3.919033	00:1a:80:5c:51:7d	Broadcast	ARP	who has 203.148.145.237? Tell 203.148.145.238
210	4.181438	00:1a:80:5c:51:7d	Broadcast	ARP	who has 203.148.145.240? Tell 203.148.145.238
211	4.181623	SmcNetwo_51:fa:c7	00:1a:80:5c:51:7d	ARP	203.148.145.240 is at 03:bf:cb:94:91:f0
212	4.181668	00:1a:80:5c:51:7d	Broadcast	ARP	who has 203.148.145.241? Tell 203.148.145.238

```
⊟ Address Resolution Protocol (request)
    Hardware type: Ethernet (0x0001)
    Protocol type: IP (0x0800)
    Hardware size: 6
    Protocol size: 4
    Opcode: request (0x0001)
    Sender MAC address: 00:1a:80:5c:51:7d (00:1a:80:5c:51:7d)
    Sender IP address: 203.148.145.238 (203.148.145.238)
    Target MAC address: Broadcast (ff:ff:ff:ff:ff:ff)
```

```
0000  ff ff ff ff ff ff 00 1a  80 5c 51 7d 08 06 00 01   ........ .\Q}....
0010  08 00 06 04 00 01 00 1a  80 5c 51 7d cb 94 91 ee   ........ .\Q}....
0020  ff ff ff ff ff ff cb 94  91 ed                     ........ ..
```

Ping Scan (diff subnet)

ARP request will not work when scanning devices in a different broadcast domains

As root, running nmap with –sP option sends

- ICMP echo requests
- TCP segments targeting port 80 with ACK bit set

As non root, nmap sends only TCP segments

If the port 80 on the remote host is not open

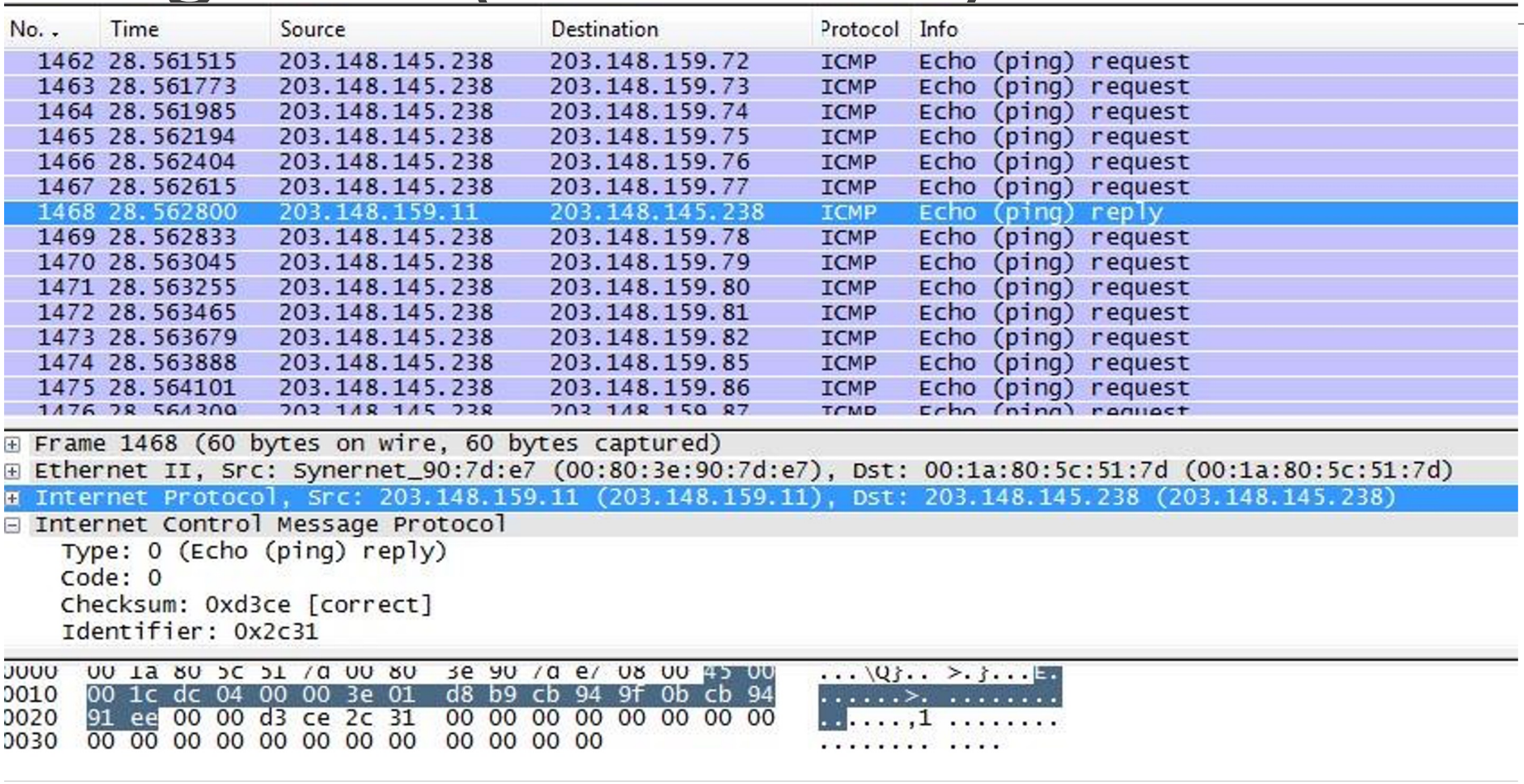
- It will receive response with RST bit set

If the port 80 on the remote host is open

- The nmap host will receive SYN and ACK bits set, but it will reply with a segment with RST bit set

This is because the nmap host does not want to establish a session with the target, just want to scan

Ping Scan (diff subnet)



No.	Time	Source	Destination	Protocol	Info
1462	28.561515	203.148.145.238	203.148.159.72	ICMP	Echo (ping) request
1463	28.561773	203.148.145.238	203.148.159.73	ICMP	Echo (ping) request
1464	28.561985	203.148.145.238	203.148.159.74	ICMP	Echo (ping) request
1465	28.562194	203.148.145.238	203.148.159.75	ICMP	Echo (ping) request
1466	28.562404	203.148.145.238	203.148.159.76	ICMP	Echo (ping) request
1467	28.562615	203.148.145.238	203.148.159.77	ICMP	Echo (ping) request
1468	28.562800	203.148.159.11	203.148.145.238	ICMP	Echo (ping) reply
1469	28.562833	203.148.145.238	203.148.159.78	ICMP	Echo (ping) request
1470	28.563045	203.148.145.238	203.148.159.79	ICMP	Echo (ping) request
1471	28.563255	203.148.145.238	203.148.159.80	ICMP	Echo (ping) request
1472	28.563465	203.148.145.238	203.148.159.81	ICMP	Echo (ping) request
1473	28.563679	203.148.145.238	203.148.159.82	ICMP	Echo (ping) request
1474	28.563888	203.148.145.238	203.148.159.85	ICMP	Echo (ping) request
1475	28.564101	203.148.145.238	203.148.159.86	ICMP	Echo (ping) request
1476	28.564309	203.148.145.238	203.148.159.87	ICMP	Echo (ping) request

```
⊞ Frame 1468 (60 bytes on wire, 60 bytes captured)
⊞ Ethernet II, Src: Synernet_90:7d:e7 (00:80:3e:90:7d:e7), Dst: 00:1a:80:5c:51:7d (00:1a:80:5c:51:7d)
⊞ Internet Protocol, Src: 203.148.159.11 (203.148.159.11), Dst: 203.148.145.238 (203.148.145.238)
⊟ Internet Control Message Protocol
    Type: 0 (Echo (ping) reply)
    Code: 0
    Checksum: 0xd3ce [correct]
    Identifier: 0x2c31
```

```
0000  00 1a 80 5c 51 7d 00 80  3e 90 7d e7 08 00 45 00   ...\Q}.. >.}...E.
0010  00 1c dc 04 00 00 3e 01  d8 b9 cb 94 9f 0b cb 94   ......>. ........
0020  91 ee 00 00 d3 ce 2c 31  00 00 00 00 00 00 00 00   ......,1 ........
0030  00 00 00 00 00 00 00 00  00 00 00 00               ........ ....
```

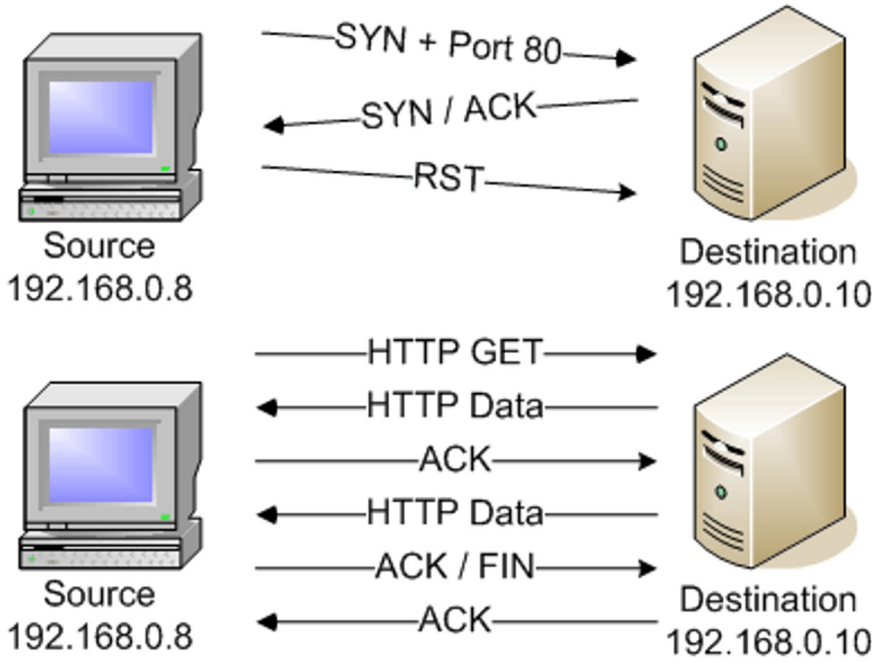
Ping Scan (diff subnet)

No.	Time	Source	Destination	Protocol	Info
1394	28.471175	203.148.145.238	203.148.159.34	ICMP	Echo (ping) request
1395	28.471397	203.148.145.238	203.148.159.35	ICMP	Echo (ping) request
1396	28.471609	203.148.145.238	203.148.159.36	ICMP	Echo (ping) request
1397	28.471819	203.148.145.238	203.148.159.37	ICMP	Echo (ping) request
1398	28.472029	203.148.145.238	203.148.159.38	ICMP	Echo (ping) request
1399	28.472242	203.148.145.238	203.148.159.39	ICMP	Echo (ping) request
1400	28.472452	203.148.145.238	203.148.159.40	ICMP	Echo (ping) request
1401	28.472665	203.148.145.238	203.148.159.43	ICMP	Echo (ping) request
1402	28.472874	203.148.145.238	203.148.159.44	ICMP	Echo (ping) request
1403	28.473083	203.148.145.238	203.148.159.45	ICMP	Echo (ping) request
1404	28.473296	203.148.145.238	203.148.159.46	TCP	35126 > http [ACK] Seq=0 Ack=0 Win=4096 Len=0
1405	28.473532	203.148.145.238	203.148.159.47	TCP	35126 > http [ACK] Seq=0 Ack=0 Win=3072 Len=0
1406	28.473746	203.148.145.238	203.148.159.48	TCP	35126 > http [ACK] Seq=0 Ack=0 Win=3072 Len=0
1407	28.473959	203.148.145.238	203.148.159.49	TCP	35126 > http [ACK] Seq=0 Ack=0 Win=2048 Len=0
1408	28.474171	203.148.145.238	203.148.159.50	TCP	35126 > http [ACK] Seq=0 Ack=0 Win=4096 Len=0
1409	28.474383	203.148.145.238	203.148.159.51	TCP	35126 > http [ACK] Seq=0 Ack=0 Win=2048 Len=0
1410	28.474605	203.148.145.238	203.148.159.52	TCP	35126 > http [ACK] Seq=0 Ack=0 Win=1024 Len=0
1411	28.474818	203.148.145.238	203.148.159.53	TCP	35126 > http [ACK] Seq=0 Ack=0 Win=3072 Len=0

```
⊞ Frame 1436 (54 bytes on wire, 54 bytes captured)
⊞ Ethernet II, Src: 00:1a:80:5c:51:7d (00:1a:80:5c:51:7d), Dst: Synernet_90:7d:e7 (00:80:3e:90:7d:e7)
⊞ Internet Protocol, Src: 203.148.145.238 (203.148.145.238), Dst: 203.148.159.64 (203.148.159.64)
⊞ Transmission Control Protocol, Src Port: 35126 (35126), Dst Port: http (80), Seq: 0, Ack: 0, Len: 0
```

```
0000  00 80 3e 90 7d e7 00 1a  80 5c 51 7d 08 00 45 00   ..>.}... .\Q}..E.
0010  00 28 25 5a 00 00 2b 06  a2 1e cb 94 91 ee cb 94   .(%Z..+. ........
0020  9f 40 89 36 00 50 11 e3  17 88 05 5c cb fd 50 10   .@.6.P.. ...\..P.
0030  10 00 53 31 00 00                                  ..S1..
```

Version Detection (nmap –sV)



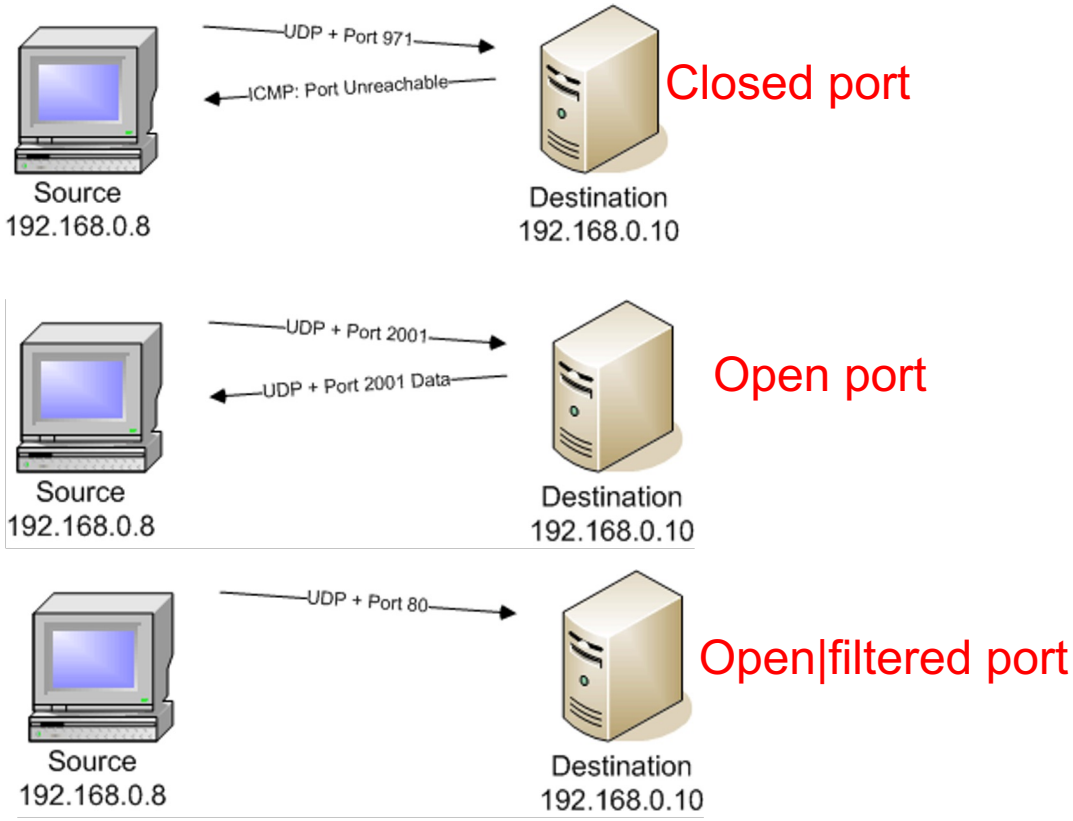
UDP Scan (-sU)

UDP has no need for SYNs, FINs, or any other fancy handshaking

With the UDP protocol, packets are sent and received without warning and prior notice is not usually expected

This lack of a formal communications process greatly simplifies UDP scanning!

UDP Scan


UDP + Port 971
ICMP: Port Unreachable
Source
192.168.0.8
Destination
192.168.0.10
Closed port
UDP + Port 2001
UDP + Port 2001 Data
Source
192.168.0.8
Destination
192.168.0.10
Open port
UDP + Port 80
Source
192.168.0.8
Destination
192.168.0.10
Open|filtered port

UDP Scan Advantages

No overhead of a TCP handshake

- less "chatty" once it finds an open port

However, if ICMP is responding to each unavailable port, the number of total frames can exceed a TCP scan by about 30%!

Very efficiently on Windows-based devices

- Microsoft-based OSes do not usually implement any type of ICMP rate limiting

UDP Scan Disadvantages

The UDP scan provides port information only

- If additional version information is needed
  - The scan must be supplemented with a version detection scan (-sV) or the operating system fingerprinting option (-O).

UDP scan requires privileged access

When to Use UDP Scan

Because of the huge amount of TCP traffic on most networks, the usefulness of the UDP scan is often incorrectly discounted

There are numerous examples of open UDP ports caused by spyware applications, Trojan horses, and other malicious software

The UDP scan will locate these open ports and provide the security manager with valuable information that can be used to identify and contain these infestations

ACK Scan (-sA)

Nmap's unique ACK scan will never locate an open port

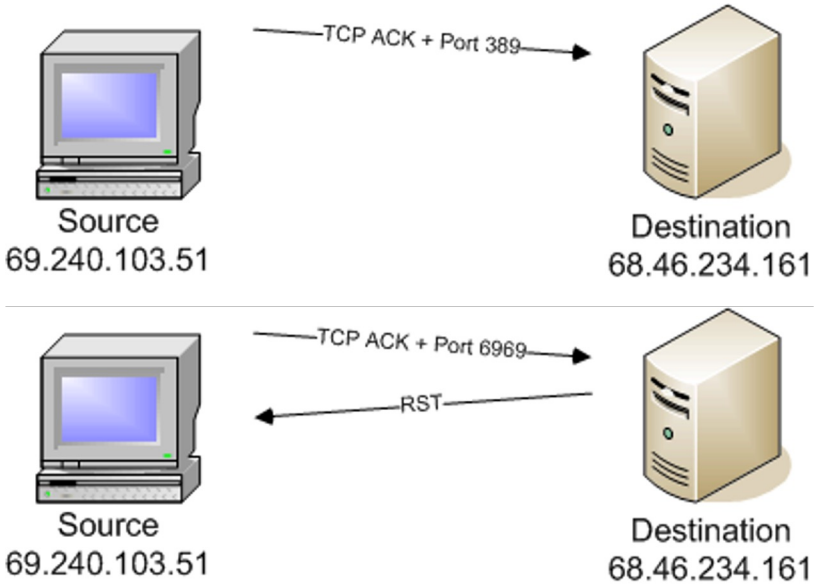
The ACK scan only provides a "**filtered**" or "**unfiltered**" disposition

- It never connects to an application to confirm an "open" state

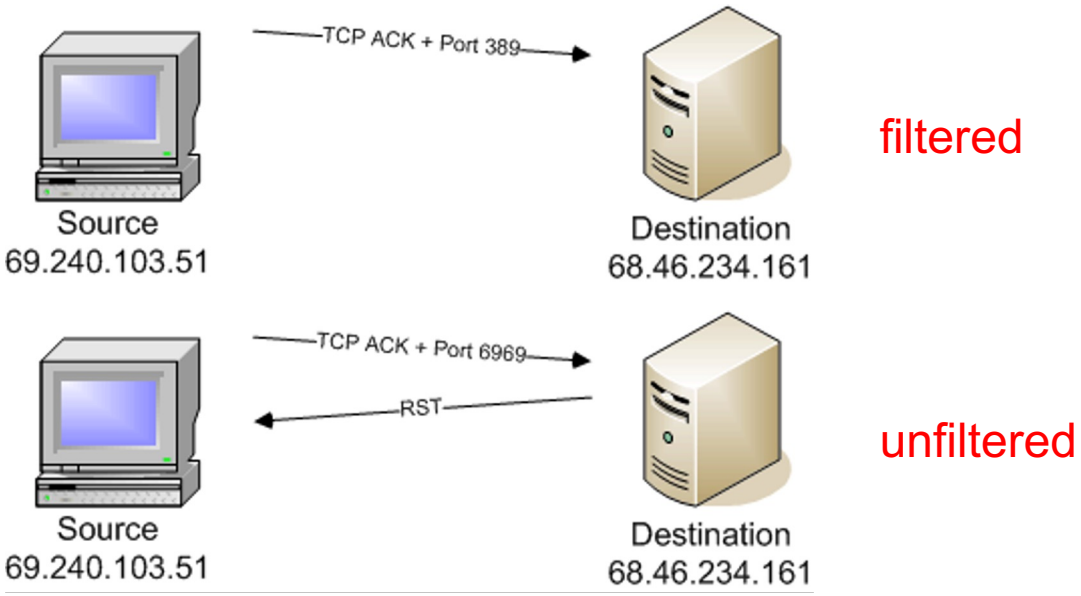
At face value this appears to be rather limiting

- The ACK scan can characterize the ability of a packet to traverse firewalls or packet filtered links

ACK Scan


TCP ACK + Port 389
Source
69.240.103.51
Destination
68.46.234.161
TCP ACK + Port 6969
RST
Source
69.240.103.51
Destination
68.46.234.161

ACK Scan


TCP ACK + Port 389
Source
69.240.103.51
Destination
68.46.234.161
filtered
TCP ACK + Port 6969
RST
Source
69.240.103.51
Destination
68.46.234.161
unfiltered

ACK Scan: Advantages and Disadvantages

Since the ACK scan doesn't open any application sessions

- The conversation between nmap and the remote device is relatively simple
- This scan of a single port is almost invisible when combined with the other network traffic

The ACK scan's simplicity is also its largest disadvantage

- Because it never tries to connect to a remote device, it can never definitively identify an open port

When to Use ACK Scan

Although the ACK scan doesn't identify open ports

- It does a masterful job of identifying ports that are filtered through a firewall.

This list of filtered and unfiltered port numbers is useful as reconnaissance for a more detailed scan that focuses on specific port numbers

Scan Timing(-T <mode>)

name	Probe Response Timeout	Time Spent on One Host	Time between Probes	Use Parallelized Probes
Paranoid	5 min	Unlimited	5 min	No
Sneaky	15 sec	Unlimited	12 sec	No
Polite	6 sec	Unlimited	0.4 sec	No
Normal	6 sec	Unlimited	None	No
Aggressive	1 sec	5 min	None	Yes
Insane	0.3 sec	75 sec	None	Yes

Timing Your Scan (-T <mode>)

```
[root@eea340 init.d]# nmap -T insane 140.130.19.169

Starting nmap V. 2.54BETA31 ( www.insecure.org/nmap/ )
Interesting ports on  (140.130.19.169):
(The 1548 ports scanned but not shown below are in state: closed)
Port       State       Service
135/tcp    open        loc-srv
139/tcp    open        netbios-ssn
445/tcp    open        microsoft-ds
1025/tcp   open        listen
1026/tcp   open        nterm
1723/tcp   open        pptp


Nmap run completed -- 1 IP address (1 host up) scanned in 0 seconds
[root@eea340 init.d]#
```

OS Fingerprinting

Once a network has been identified

- The next step is to discover the systems that are attached to it

Known network exploits tend to be very specific with respect to the host **OS** in conjunction with specific **versions** of specific **applications**

From network administration standpoint, we can document our network automatically

- We can also detect unapproved or unexpected devices

How OS Discovery Works

Active (nmap)

- Send several probes or triggers and analyze the responses to possibly guess the OS
- Commonly used OSes present an identifiable signature when stimulated this way

Passive (p0f)

- Monitor traffic, looking for patterns that are characteristic of known OSes
- More attractive stealth and low network impact
- Best result when connect directly to the network being observed

Active OS Fingerprinting with Nmap

The OS fingerprinting process is not a port scan

- It works in conjunction with nmap's scanning processes.

Nmap's OS fingerprinting is based on a remote device's responses when it's sent a group of very specific packets.

If a particular OS receives a TCP ACK frame to a closed port, it may react differently than other operating systems receiving the same frame.

It's these minor response variations that allow nmap to build detailed "**fingerprints**" for different operating systems and devices.

Active OS Fingerprinting with Nmap

Different from a version detection scan (-sV)

- although many methodologies are similar

For example, both the version scan and the OS fingerprinting scan rely on the nmap scanning process to identify active devices and their available ports

However, The OS fingerprinting process uses techniques not found in version detection

- Such as a standard method of operating system probing and a modular operating system definition file.

OS Fingerprinting Operation

1. Before the operating system fingerprinting process begins, nmap performs a normal ping and scan
   - During the nmap scan, nmap determines device availability and categorizes the ports on the remote device as open, closed, or filtered.

2. Once the open and closed ports are identified, nmap begins the OS fingerprinting procedure
   1. Sending an OS probe,
   2. followed by series of TCP handshakes that are used for testing responses to the TCP uptime measurement options, TCP sequence predictabilities, and IP identification sequence generation.

OS Fingerprinting Operation

A normal OS fingerprinting process will uncover the following information:

```
Device type: general purpose Running: Microsoft Windows NT/2K/XP

OS details: Microsoft Windows XP SP2

TCP Sequence Prediction: Class=truly random
      Difficulty=9999999 (Good luck!)

IPID Sequence Generation: Incremental
```

TCP Sequence Prediction Analysis

If the TCP sequences of a remote device are understood, then that remote device is more susceptible to malicious activity such as TCP hijacking.

TCP hijacking is a technique that allows a third-party to "**interrupt**" an existing TCP connection between two devices.

The attacker can then masquerade as one of the original stations, allowing them to send unwanted information to the other device.

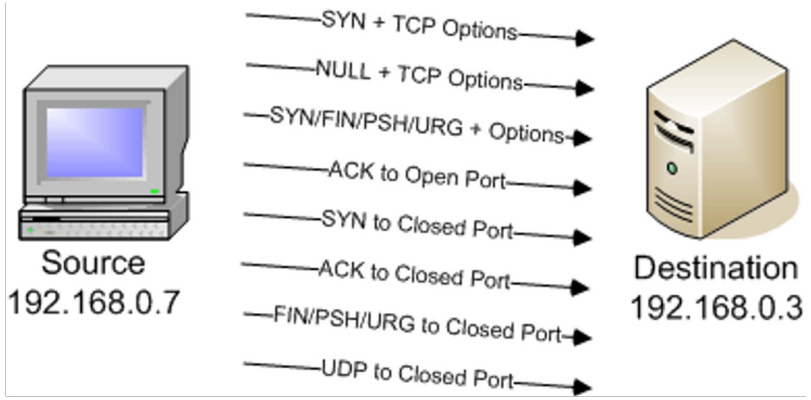
A major technical aspect of the hijacking process is the ability of the attacking station to predict the TCP sequence numbers.

OS Fingerprinting Process (-O)

The operating system fingerprinting probes begin with Test 1 through Test 7

- followed immediately by the UDP-based ICMP port unreachable test

The responses to these probes are compared to the T1-T7 fingerprints in the hopes of locating some likely matches

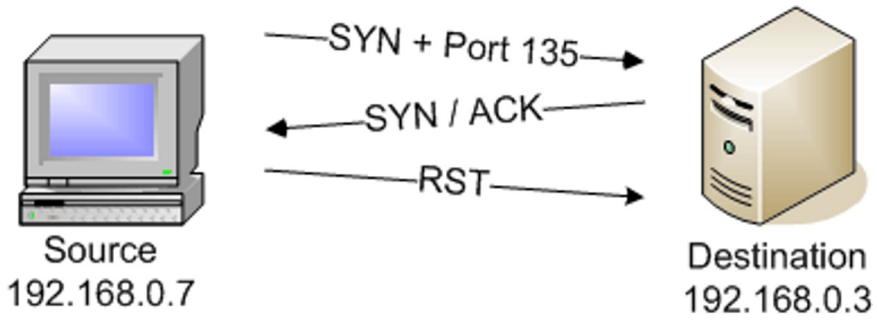


OS Fingerprinting Process

Nmap then performs six TCP SYN scans to the open port

The resulting SYN/ACK responses are used to compare

- TCP initial sequence numbers
- IP identification values
- TCP timestamp option sequences



Advantages of OS Fingerprinting

The operating system fingerprinting process provides detailed information about the operating system running on a device

In some cases

- the exact version number of the operating system and detailed hardware information can be determined with the OS fingerprinting option

The OS fingerprinting option can assist with locating systems that are out of compliance, and can also provide information about the operating system running on the "rogue" station

When this option is combined with the version scan (-sV)

- Specific services can also be checked for compliancy.

Disadvantages of OS Fingerprinting

The OS fingerprinting process requires privileged user access

This scan will not run if a non-privileged user attempts to use the –O option.

Although there are only about thirty frames that traverse the network during an OS fingerprinting process

- Some of the frames used to query the remote device are frame types that would never occur on a normal network

For example, it's unusual to see a frame with the SYN, FIN, PSH, and URG flags that would also include numerous TCP options

A trained eye will quickly identify these unusual frames

- Assuming that someone is watching the network during that timeframe

When to Use OS Fingerprinting

The operating system fingerprinting option is often integrated into many organization's compliance checks

- If an outdated or unexpected operating system is seen on the network, the security group can follow their policies to identify and remove the noncompliant station from the network.

In some cases, a particular operating system may have known vulnerabilities that need to be patched

- The OS fingerprinting process can assist with locating all of the specific operating system versions on the network, ensuring that organization's vulnerable holes will be patched

Limit Operating System Scanning

The operating system fingerprinting process is most accurate when both open ports and closed ports are available for testing

- If only one type of port is available, the fingerprinting process will not be as precise

In this situation, nmap provides a warning message:

```
Warning: OS detection will be MUCH less reliable because we did not find at least 1 open and 1 closed TCP port
```

The fingerprinting process will still function, but the results will not be optimal