



Wireshark: Traffic Analysis

Nmap Scans

- Industrial tool for mapping networks.
- It identifies hosts and discovering services

Types of Nmap Scans:

1. TCP Connect Scans
 - Relies on a three-way handshake (needs to finish the handshake process).
 - Usually conducted with `nmap -sT` <--- initiating a TCP Connect Scan; which is a default TCP scan type. Completing the full TCP three-way handshake.
 - Used by non-privileged users (only option for a non-root user).
 - Usually has a windows size larger than 1024 bytes as the request expects some data due to the nature of the protocol.

TCP flags in a nutshell:

Notes	Wireshark Filters
Global search.	<ul style="list-style-type: none">• <code>tcp</code>• <code>udp</code>
<ul style="list-style-type: none">• Only SYN flag.• SYN flag is set. The rest of the bits are not important.	<ul style="list-style-type: none">• <code>tcp.flags == 2</code>• <code>tcp.flags.syn == 1</code>
<ul style="list-style-type: none">• Only ACK flag.• ACK flag is set. The rest of the bits are not important.	<ul style="list-style-type: none">• <code>tcp.flags == 16</code>• <code>tcp.flags.ack == 1</code>
<ul style="list-style-type: none">• Only SYN, ACK flags.• SYN and ACK are set. The rest of the bits are not important.	<ul style="list-style-type: none">• <code>tcp.flags == 18</code>• <code>(tcp.flags.syn == 1) and (tcp.flags.ack == 1)</code>
<ul style="list-style-type: none">• Only RST flag.	<ul style="list-style-type: none">• <code>tcp.flags == 4</code>

<ul style="list-style-type: none">• RST flag is set. The rest of the bits are not important.	<ul style="list-style-type: none">• <code>tcp.flags.reset == 1</code>	
<ul style="list-style-type: none">• Only RST, ACK flags.• RST and ACK are set. The rest of the bits are not important.	<ul style="list-style-type: none">• <code>tcp.flags == 20</code>• <code>(tcp.flags.reset == 1) and (tcp.flags.ack == 1)</code>	
<ul style="list-style-type: none">• Only FIN flag• FIN flag is set. The rest of the bits are not important.	<ul style="list-style-type: none">• <code>tcp.flags == 1</code>• <code>tcp.flags.fin == 1</code>	
Open TCP Port	Open TCP Port	Closed TCP Port
<ul style="list-style-type: none">• SYN -->• <-- SYN, ACK• ACK -->	<ul style="list-style-type: none">• SYN -->• <-- SYN, ACK• ACK -->• RST, ACK -->	<ul style="list-style-type: none">• SYN -->• <-- RST, ACK

Open TCP port (Connect):

Wireshark packet capture for 'tcp-connect-open-port.pcapng'. The packet list shows four packets:

No.	Time	Source	Destination	Protocol	Info
1	0.000000000	10.10.60.7	10.10.47.123	TCP	36958 → 22 [SYN] Seq=0 Win=62727 Len=0 MSS=8961 SACK_PERM TSval=1438758498 TSecr=0
2	0.000012250	10.10.47.123	10.10.60.7	TCP	22 → 36958 [SYN, ACK] Seq=0 Ack=1 Win=62643 Len=0 MSS=8
3	0.000209974	10.10.60.7	10.10.47.123	TCP	36958 → 22 [ACK] Seq=1 Ack=1 Win=62848 Len=0 TSval=1438
4	0.000244154	10.10.60.7	10.10.47.123	TCP	36958 → 22 [RST, ACK] Seq=1 Ack=1 Win=62848 Len=0 TSval=1438

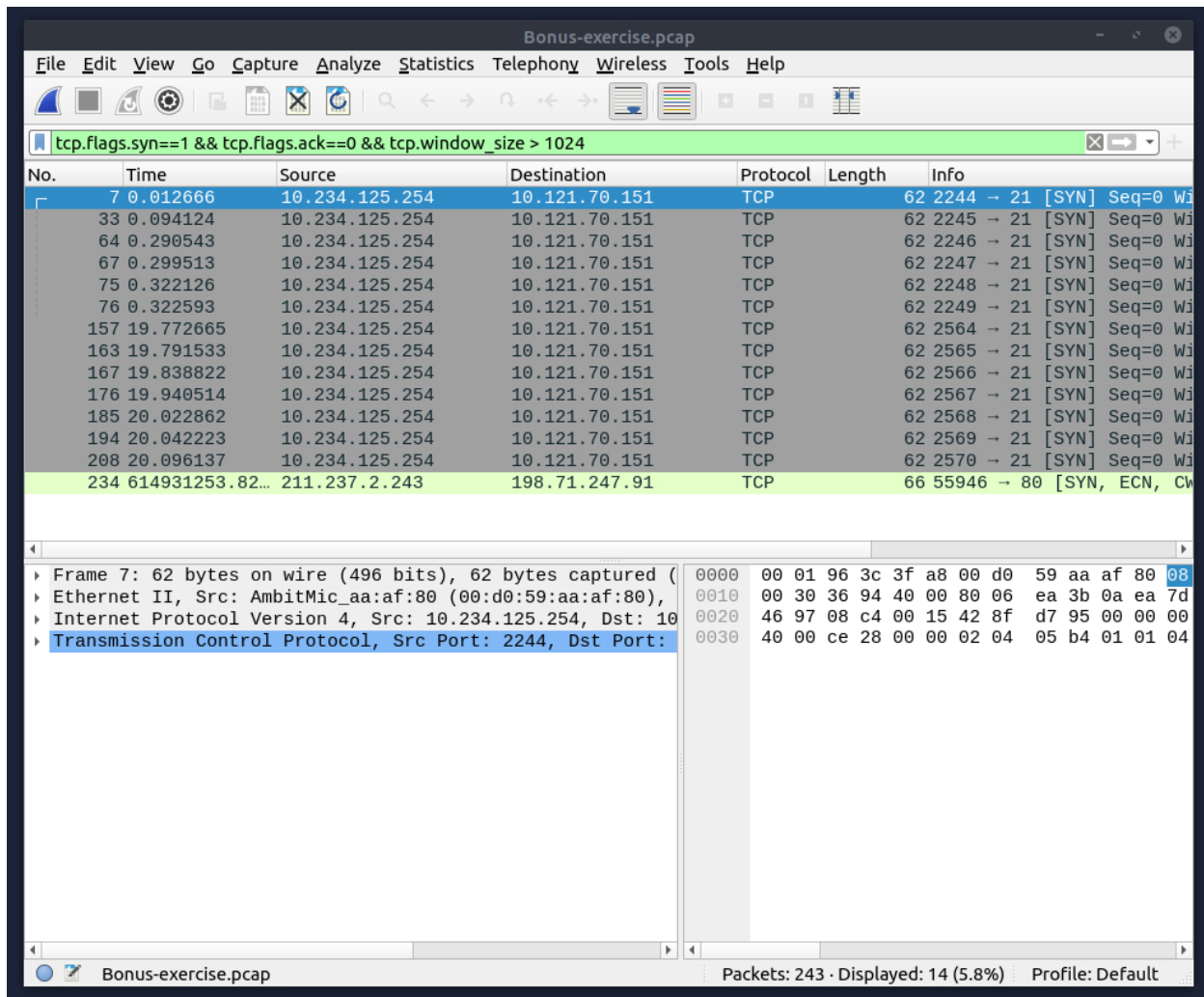
Closed TCP port (Connect):

Wireshark packet capture for 'tcp-connect-close-port.pcapng'. The packet list shows two packets:

No.	Time	Source	Destination	Protocol	Info
1	0.000000000	10.10.60.7	10.10.47.123	TCP	59934 → 21 [SYN] Seq=0 Win=62727 Len=0 MSS=8961 SACK_PERM TSval=1438758498 TSecr=0
2	0.000005840	10.10.47.123	10.10.60.7	TCP	21 → 59934 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0

- the image above shows a pattern is an isolated traffic. It is not always easy to spot these patterns in a big capture file.

- Therefore, an analyst needs to use filters to view the initial anomaly patterns.
- `tcp.flags.syn==1 && tcp.flags.ack==0 && tcp.window_size > 1024`



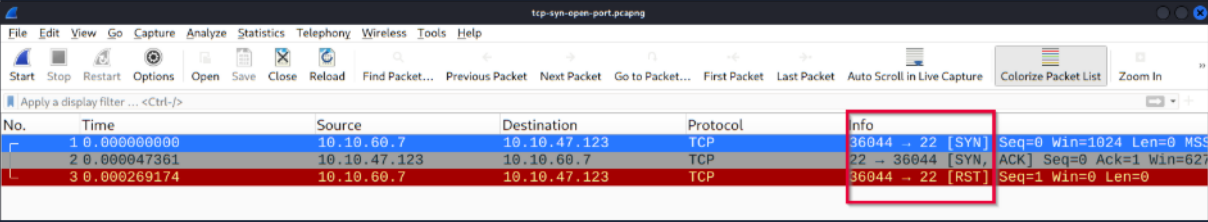
- `tcp.flags.syn==1` <--- This filter for packets where the SYN (synchronize) flag is set.
- `tcp.flags.ack==0` <--- This filter packets where the ACK (Acknowledgment) flag is set.
 - a. in the TCP three-way handshake, the FIRST packet sent is a “pure” SYN packet (SYN=1, ACK=0). The SECOND (the response) has both SYN and ACK set (SYN=1, ACK=1). By specifying `ack==0`, we are filtering specifically for the initial connection request from the client to the server.
- `tcp.window_size > 1024` <--- This filter for packets where the TCP Windows Size is greater than 1024 bytes.
 - b. The window size indicates how many bytes the sender is willing to receive before requiring an acknowledgment. This part of the filter excludes packets with very small buffers, which might be used to identify specific types of traffic, operating systems, or to filter out certain types of automated scanning tools that use minimal window sizes.

TCP SYN Scans

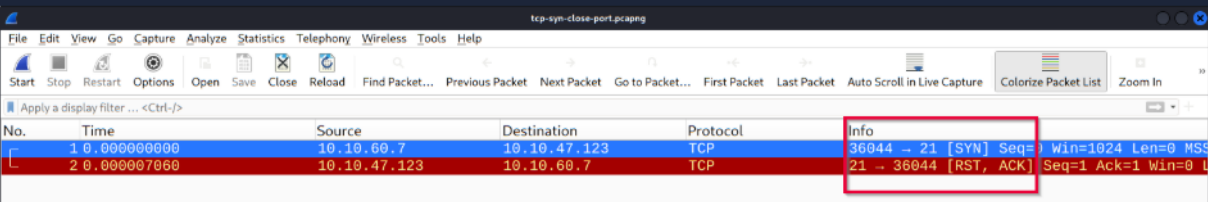
- Doesn't rely on the three-way handshake
- Usually conducted with **nmap -sS** <--- this performs a TCP SYN Scan, often called "stealth" or "half-open" scan. This Never completes the handshake.
- Used by privileged users.
- Usually, this have a size less than 1024 bytes as the request is not finished, and it doesn't expect to receive data.

Open TCP Port	Close TCP Port
<ul style="list-style-type: none">• SYN -->• <-- SYN,ACK• RST-->	<ul style="list-style-type: none">• SYN -->• <-- RST,ACK

Open TCP port (SYN):



Closed TCP port (SYN):



This filter below shows TCP SYN patterns in a capture file.

tcp.flags.syn==1 && tcp.flags.ack==0 && tcp.window_size <=1024 <--- filters out suspicious or automated TCP connection attempts.

Exercise.pcapng

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

tcp.flags.syn==1 && tcp.flags.ack==0 and tcp.window_size <= 1024

No.	Time	Source	Destination	Protocol	Length	Info
2005	153.750287125	10.10.60.7	10.10.47.123	TCP	58	36044 → 445 [SYN]
2006	153.750287215	10.10.60.7	10.10.47.123	TCP	58	36044 → 1723 [SYN]
2007	153.750287365	10.10.60.7	10.10.47.123	TCP	58	36044 → 22 [SYN] S
2008	153.750287375	10.10.60.7	10.10.47.123	TCP	58	36044 → 3306 [SYN]
2009	153.750287415	10.10.60.7	10.10.47.123	TCP	58	36044 → 995 [SYN]
2015	153.750366886	10.10.60.7	10.10.47.123	TCP	58	36044 → 111 [SYN]
2017	153.750378427	10.10.60.7	10.10.47.123	TCP	58	36044 → 135 [SYN]
2018	153.750378497	10.10.60.7	10.10.47.123	TCP	58	36044 → 8080 [SYN]
2019	153.750378537	10.10.60.7	10.10.47.123	TCP	58	36044 → 25 [SYN] S
2023	153.750394477	10.10.60.7	10.10.47.123	TCP	58	36044 → 8888 [SYN]
2026	153.750676811	10.10.60.7	10.10.47.123	TCP	58	36044 → 143 [SYN]
2027	153.750685211	10.10.60.7	10.10.47.123	TCP	58	36044 → 5900 [SYN]
2028	153.750685281	10.10.60.7	10.10.47.123	TCP	58	36044 → 21 [SYN] S
2032	153.750750092	10.10.60.7	10.10.47.123	TCP	58	36044 → 113 [SYN]
2033	153.750750222	10.10.60.7	10.10.47.123	TCP	58	36044 → 1720 [SYN]
2034	153.750750262	10.10.60.7	10.10.47.123	TCP	58	36044 → 53 [SYN] S

Frame 2005: 58 bytes on wire (464 bits), 58 bytes capture

Ethernet II, Src: 02:6d:30:b1:b9:69 (02:6d:30:b1:b9:69),

Internet Protocol Version 4, Src: 10.10.60.7, Dst: 10.10.

Transmission Control Protocol, Src Port: 36044, Dst Port:

- **tcp.flags.syn==1** and **tcp.flags.ack==0** <--- This is the initial handshake: Together, these two flags isolate “pure” SYN packets. The first packet sent by the client has the SYN flag set “**SYN==1**” and the ACK flag unset “**ACK==0**”
- By filtering for **ack == 0**, we are excluding the “SYN/ACK” response from the server, focusing only on the initial connection request.
- **tcp.window_size <= 1024** <--- We are indicating the amount of data (in bytes) “**window_size**” that the sender willing to receive
- Most operating systems (Windows, Linux macOS) typically have larger initial window size, often ranging from 8,192 to 65,535 bytes.
- A window size of 1024 or less is unusual for a standard user’s computer

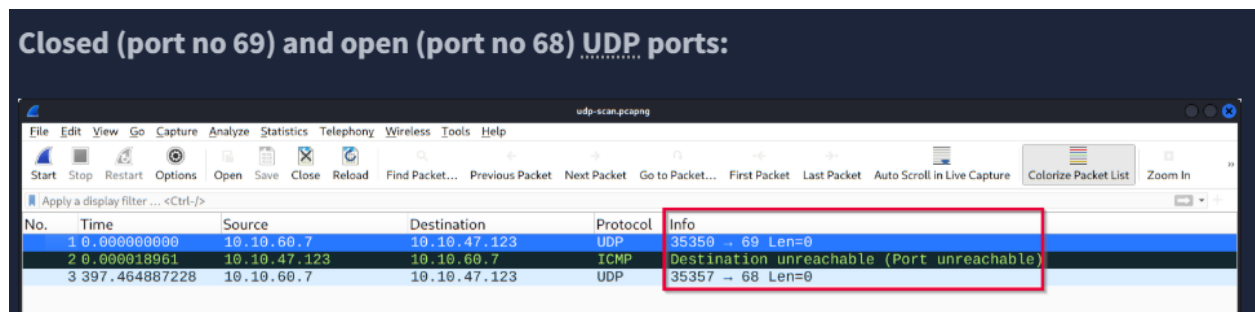
WHY THIS FILTER IS USEFUL

- This is used by network security professionals to detect Reconnaissance and Port Scanning
 - a. Nmap Detection
 - b. Botnet
 - c. Filtering Noise

UDP Scans

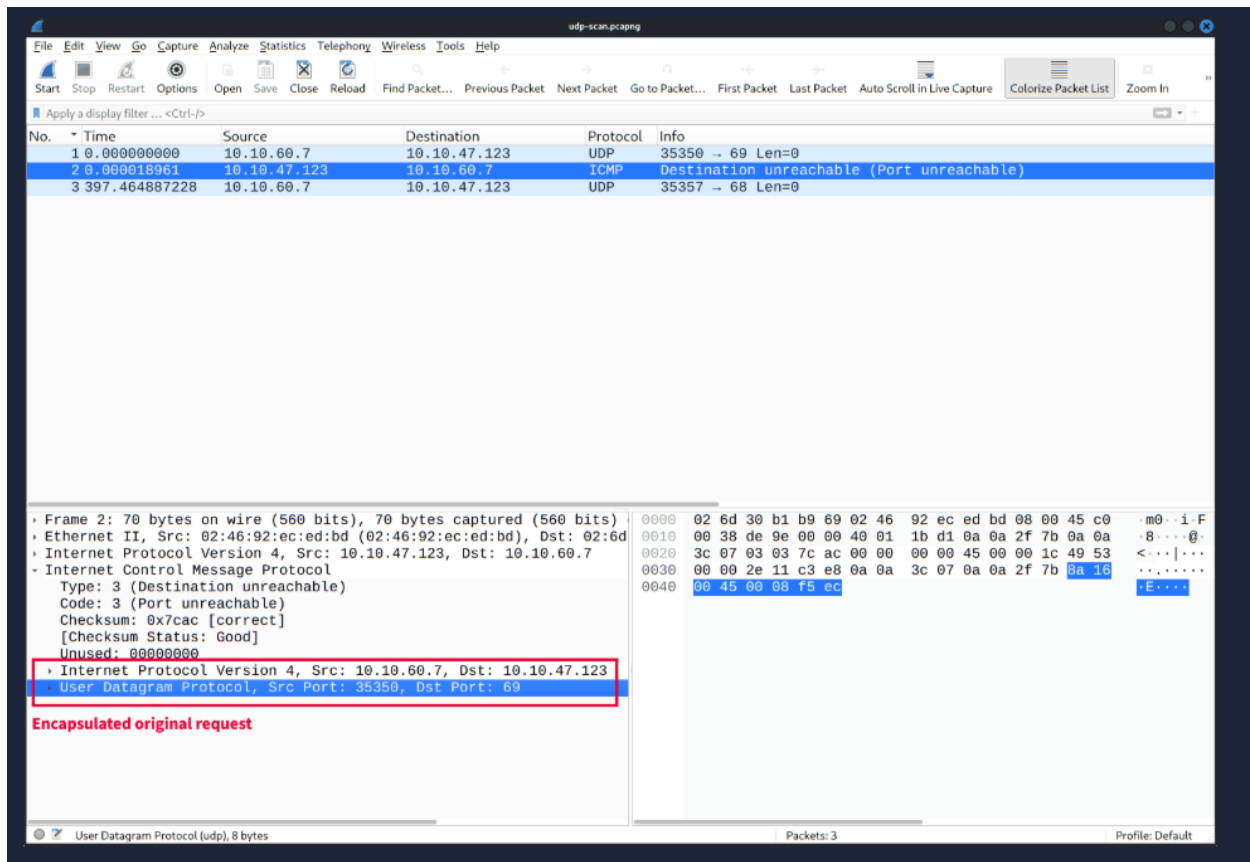
- Doesn't require a handshake process
- No prompt for open ports
- ICMP error message for close ports
- Usually conducted with a **nmap -sU** <--- this is used to prompt UDP scan. Services like DNS (port 53), SNMP (port 161/162), and DHCP (port 67/68) rely on UDP

Open <u>UDP</u> Port	Closed <u>UDP</u> Portstrong>
<ul style="list-style-type: none">• <u>UDP</u> packet -->	<ul style="list-style-type: none">• <u>UDP</u> packet -->• ICMP Type 3, Code 3 message. (Destination unreachable, port unreachable)



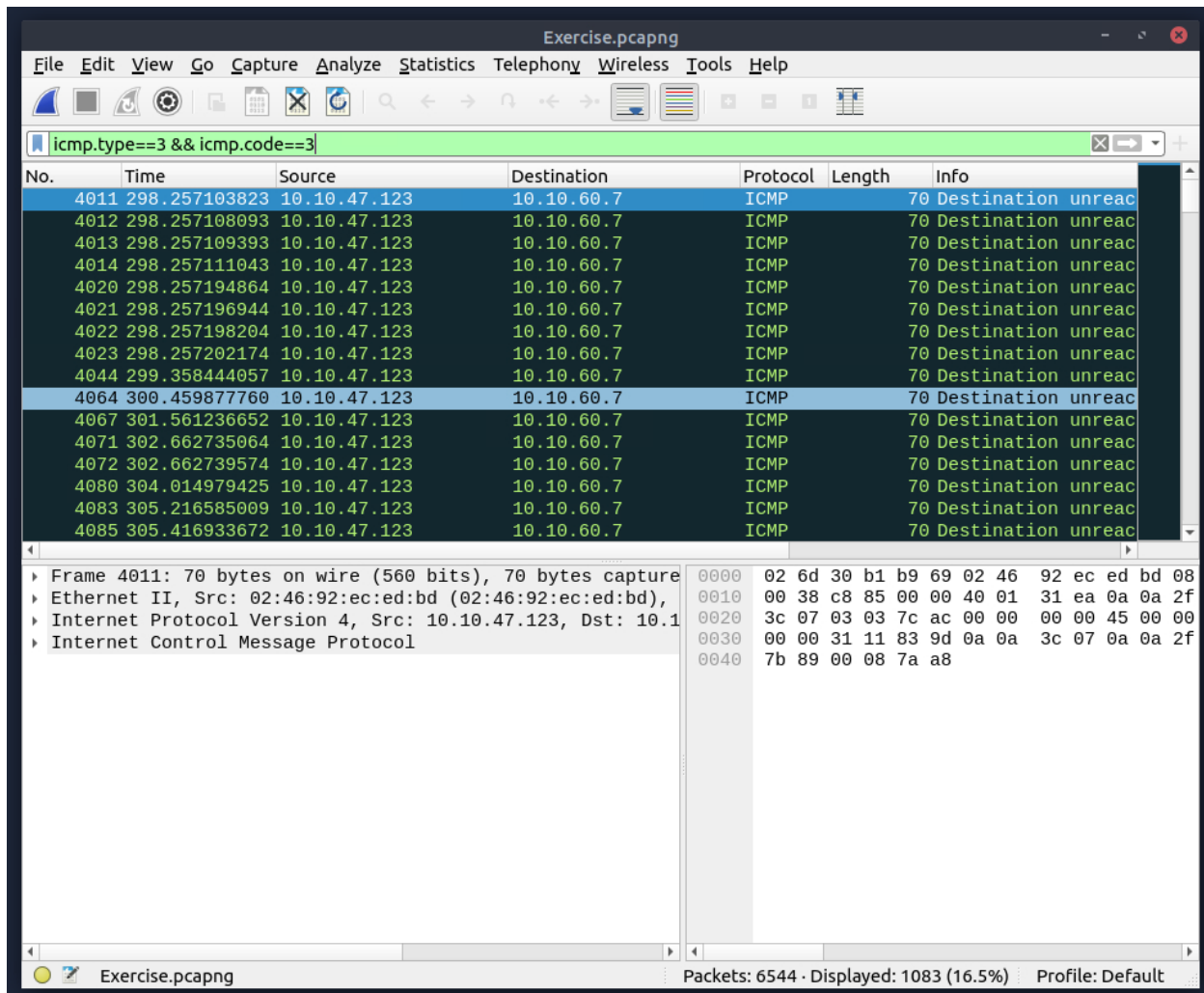
- this is showing a close port returns an ICMP error packet. The ICMP error message uses the original request as encapsulated data to show the source/reason of the packet.

- By expanding the the ICMP packet in the pane, we will see the encapsulated data and the original requests.



- Let view the UDP scan patterns in the capture file using the filter below:

icmp.type==3 && icmp.code==3 <--- this is identifying ICMP Port Unreachable Message (one of the most important ICMP filter to monitor)



- `icmp.type == 3` <--- Identifying the destination is unreachable
- This is indicating that packets could not be delivered to its final destination
- `icmp.code == 3` <--- Identifying the port is unreachable
- This indicates that the destination was reached; however, there was no application or service listening on the specific port the sender tried to connect to.

1. What is the total number of the “TCP Connect” scans?

- To identify the total TCP connected, the following filter is needed to determine the number.
 - `tcp.flags.syn == 1 && tcp.flags.ack == 0 && tcp.window_size > 1024` <--- this filter is identifying the initial TCP connection request that appears to come from standard operating systems.

- When a port scanner is used such as **nmap** command, it automatically uses a windows default window size of exactly 1024 bytes when sending a SYN packet.
- By filtering out and looking for values greater than 1024, we are focusing on traffic that looks like it is coming from a real user or application

Exercise.pcapng

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

tcp.flags.syn == 1 && tcp.flags.ack == 0 and tcp.window_size > 1024

No.	Time	Source	Destination	Protocol	Length	Info
1973	0.019705354	10.10.60.7	10.10.47.123	TCP	74	50004 → 8652 [SYN]
1974	0.019705364	10.10.60.7	10.10.47.123	TCP	74	53158 → 1099 [SYN]
1975	0.019705404	10.10.60.7	10.10.47.123	TCP	74	57444 → 44443 [SYN]
1976	0.019705414	10.10.60.7	10.10.47.123	TCP	74	51074 → 1048 [SYN]
1977	0.019705444	10.10.60.7	10.10.47.123	TCP	74	50360 → 1124 [SYN]
1978	0.019705454	10.10.60.7	10.10.47.123	TCP	74	57828 → 548 [SYN]
1987	0.019750994	10.10.60.7	10.10.47.123	TCP	74	48290 → 5859 [SYN]
1988	0.019751534	10.10.60.7	10.10.47.123	TCP	74	42572 → 1152 [SYN]
1989	0.019751064	10.10.60.7	10.10.47.123	TCP	74	48200 → 524 [SYN]
1993	0.019772005	10.10.60.7	10.10.47.123	TCP	74	36770 → 1052 [SYN]
1995	0.019780935	10.10.60.7	10.10.47.123	TCP	74	56394 → 3527 [SYN]
1997	0.019804565	10.10.60.7	10.10.47.123	TCP	74	39726 → 1666 [SYN]
1999	0.019820025	10.10.60.7	10.10.47.123	TCP	74	49406 → 687 [SYN]
2001	0.019837646	10.10.60.7	10.10.47.123	TCP	74	35908 → 9917 [SYN]
2003	0.019869016	10.10.60.7	10.10.47.123	TCP	74	35260 → 6901 [SYN]

Frame 2003: 74 bytes on wire (592 bits), 74 bytes capture
 Ethernet II, Src: 02:6d:30:b1:b9:69 (02:6d:30:b1:b9:69),
 Internet Protocol Version 4, Src: 10.10.60.7, Dst: 10.10.
 Transmission Control Protocol, Src Port: 35260, Dst Port:

```

0000  02 46 92 ec ed bd 02 6d 30 b1 b9 69 08
0010  00 3c 1d 5c 40 00 40 06 9d ca 0a 0a 3c
0020  2f 7b 89 bc 1a f5 a9 d8 31 16 00 00 00
0030  f5 07 2a 3e 00 00 02 04 23 01 04 02 08
0040  b6 75 00 00 00 00 01 03 03 07
  
```

Exercise.pcapng Packets: 6544 - Displayed: 1000 (15.3%) Profile: Default

Answer: 1000

2. What scan type is used to scan TCP port 80?

- We will simply put in the filter **tcp.port == 80** and observe the results

Exercise.pcapng

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

tcp.port == 80

No.	Time	Source	Destination	Protocol	Length	Info
39	0.000480268	10.10.60.7	10.10.47.123	TCP	74	42026 → 80 [SYN] Seq=0 Win=62727 Len=0 MSS=896
40	0.000486458	10.10.47.123	10.10.60.7	TCP	74	80 → 42026 [SYN, ACK] Seq=0 Ack=1 Win=62643 Le
60	0.000706851	10.10.60.7	10.10.47.123	TCP	66	42026 → 80 [ACK] Seq=1 Ack=1 Win=62848 Len=0 T
61	0.000706901	10.10.60.7	10.10.47.123	TCP	66	42026 → 80 [RST, ACK] Seq=1 Ack=1 Win=62848 Le
2042	153.750818423	10.10.60.7	10.10.47.123	TCP	58	36044 → 80 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
2045	153.750829173	10.10.47.123	10.10.60.7	TCP	58	80 → 36044 [SYN, ACK] Seq=0 Ack=1 Win=62727 Le
2065	153.751019846	10.10.60.7	10.10.47.123	TCP	54	36044 → 80 [RST] Seq=1 Win=0 Len=0

▶ Frame 61: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on 0
 ▶ Ethernet II, Src: 02:6d:30:b1:b9:69 (02:6d:30:b1:b9:69), Dst: 02:46:9e:00:10:00
 ▶ Internet Protocol Version 4, Src: 10.10.60.7, Dst: 10.10.47.123
 ▶ Transmission Control Protocol, Src Port: 42026, Dst Port: 80, Seq: 1,

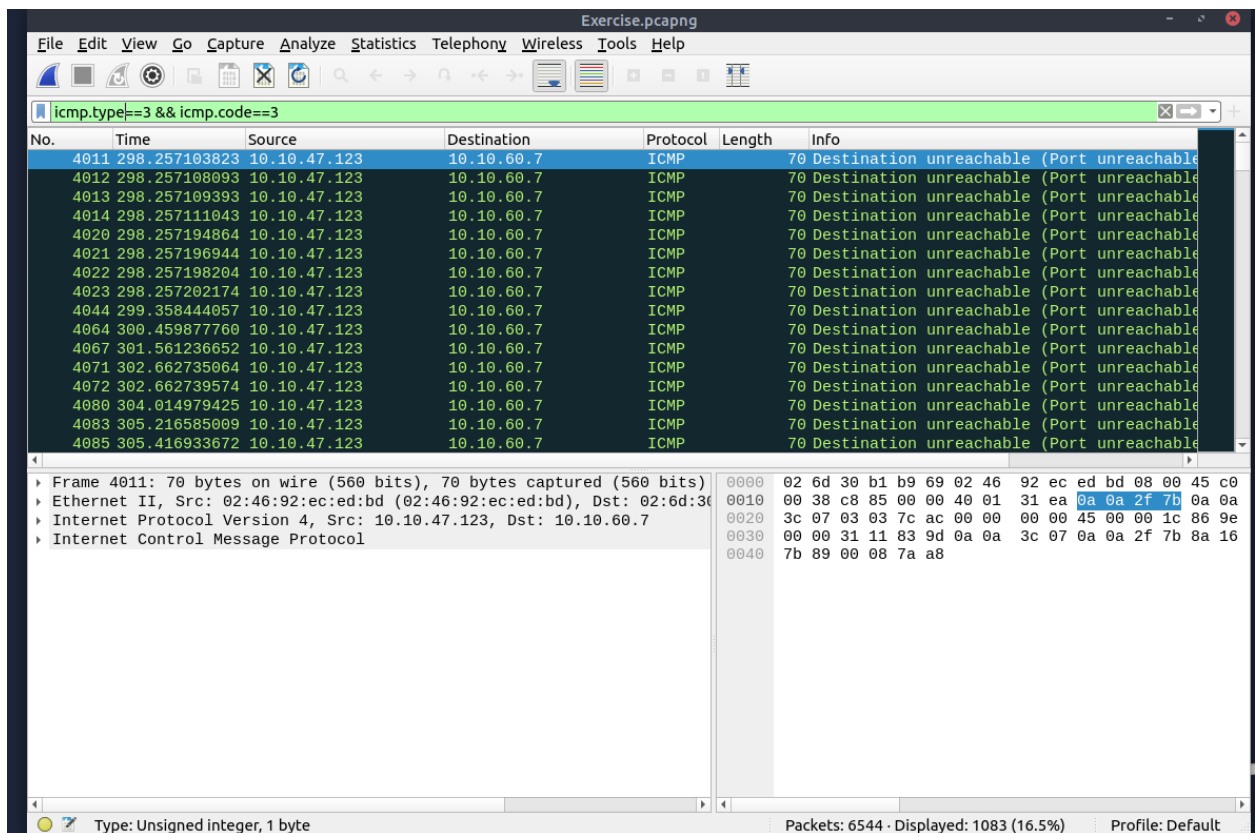
0000 02 46 92 ec ed bd 02 6d 30 b1 b9 69 08 00 45 00
 0010 00 34 70 8f 40 00 40 06 4a 9f 0a 0a 3c 07 0a 0a
 0020 2f 7b a4 2a 00 50 37 31 22 86 7f 8d 11 23 80 14
 0030 01 eb d1 76 00 00 01 01 08 0a 55 c1 b6 63 7b d6
 0040 0c e4

Looking at the info column, we can see that this is attempting a connection request

Answer: TCP Connect

3. How many “UDP close port” messages are there?

- Here we will use the icmp filter to determine the closed ports
- `icmp.type == 3 && icmp.code == 3` <---- this filter is searching for destination and port that are unreachable



Answer: 1083

4. Which UDP port in the 55-70 port range is open?

- Identify which port is open between the range above, we can use an input filter below:
- **udp.dstport >= 50 and && udp.port <= 75** --- this is asking Wireshark to look for specific ports that are open with the range. We will see the results below:

Exercise.pcapng

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

udp.dstport >= 50 && udp.port <= 75

No.	Time	Source	Destination	Protocol	Length	Info
4202	333.257660399	10.10.60.7	10.10.47.123	UDP	42	35350 → 67 Len=0
4203	333.257693469	10.10.47.123	10.10.60.7	ICMP	70	Destination unreachable (Port unreachable)
4470	452.597038565	10.10.60.7	10.10.47.123	DNS	54	Server status request 0x0000
4471	452.597075365	10.10.47.123	10.10.60.7	ICMP	82	Destination unreachable (Port unreachable)
5414	872.284187999	10.10.60.7	10.10.47.123	UDP	42	35350 → 69 Len=0
5415	872.284206960	10.10.47.123	10.10.60.7	ICMP	70	Destination unreachable (Port unreachable)
6290	1263.3415093...	10.10.60.7	10.10.47.123	UDP	42	35350 → 68 Len=0
6291	1264.1424257...	10.10.60.7	10.10.47.123	UDP	42	35351 → 68 Len=0
6294	1265.7443556...	10.10.60.7	10.10.47.123	UDP	42	35352 → 68 Len=0
6295	1266.5452626...	10.10.60.7	10.10.47.123	UDP	42	35353 → 68 Len=0
6296	1267.3464438...	10.10.60.7	10.10.47.123	UDP	42	35354 → 68 Len=0
6297	1268.1471569...	10.10.60.7	10.10.47.123	UDP	42	35355 → 68 Len=0
6298	1268.9481544...	10.10.60.7	10.10.47.123	UDP	42	35356 → 68 Len=0
6299	1269.7490752...	10.10.60.7	10.10.47.123	UDP	42	35357 → 68 Len=0

Total Length: 28
 Identification: 0xcea7 (52903)
 Flags: 0x0000
 Fragment offset: 0
 Time to live: 53
 Protocol: UDP (17)
 Header checksum: 0x3794 [validation disabled]
 [Header checksum status: Unverified]
 Source: 10.10.60.7
 Destination: 10.10.47.123
 User Datagram Protocol, Src Port: 35357, Dst Port: 68
 Source Port: 35357
 Destination Port: 68
 Length: 8
 Checksum: 0xf5e6 [unverified]
 [Checksum Status: Unverified]
 [Stream index: 1321]

0000 02 46 92 ec ed bd 02 6d 30 b1 b9 69 08 00 45 00
 0010 00 1c ce a7 00 00 35 11 37 94 0a 0a 3c 07 0a 0a
 0020 2f 7b 8a 1d 00 44 00 08 f5 e6

User Datagram Protocol (udp), 8 byte(s)

Packets: 6544 · Displayed: 14 (0.2%) Profile: Default

Answer: 68

- We notice we are getting multiple returns from port 68.