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.

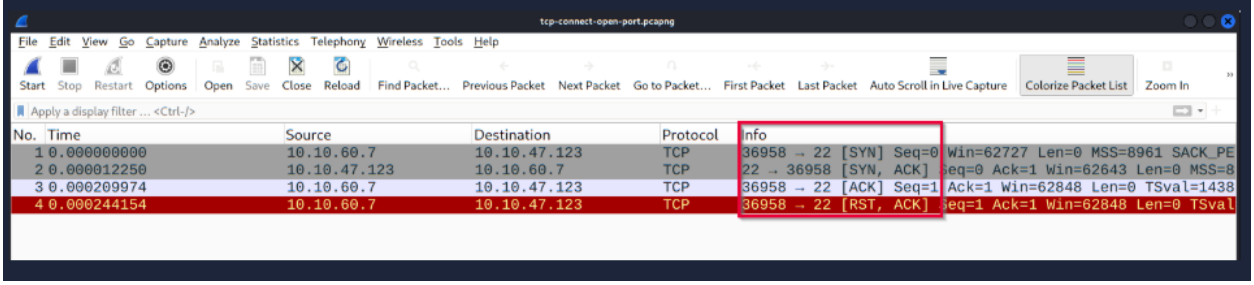
TCP flags in a nutshell:

Notes	Wireshark Filters
Global search.	• `tcp` • `udp`
• Only SYN flag. • SYN flag is set. The rest of the bits are not important.	• `tcp.flags == 2` • `tcp.flags.syn == 1`
• Only ACK flag. • ACK flag is set. The rest of the bits are not important.	• `tcp.flags == 16` • `tcp.flags.ack == 1`
• Only SYN, ACK flags. • SYN and ACK are set. The rest of the bits are not important.	• `tcp.flags == 18` • `(tcp.flags.syn == 1) and (tcp.flags.ack == 1)`
• Only RST flag.	• `tcp.flags == 4`

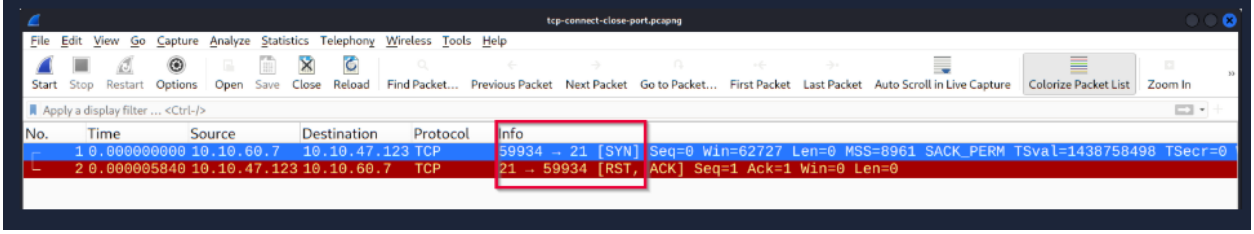
• RST flag is set. The rest of the bits are not important.	• `tcp.flags.reset == 1`
• Only RST, ACK flags. • RST and ACK are set. The rest of the bits are not important.	• `tcp.flags == 20` • `(tcp.flags.reset == 1) and (tcp.flags.ack == 1)`
• Only FIN flag • FIN flag is set. The rest of the bits are not important.	• `tcp.flags == 1` • `tcp.flags.fin == 1`

Open TCP Port	Open TCP Port	Closed TCP Port
• SYN --> • <-- SYN, ACK • ACK -->	• SYN --> • <-- SYN, ACK • ACK --> • RST, ACK -->	• SYN --> • <-- RST, ACK

Open TCP port (Connect):

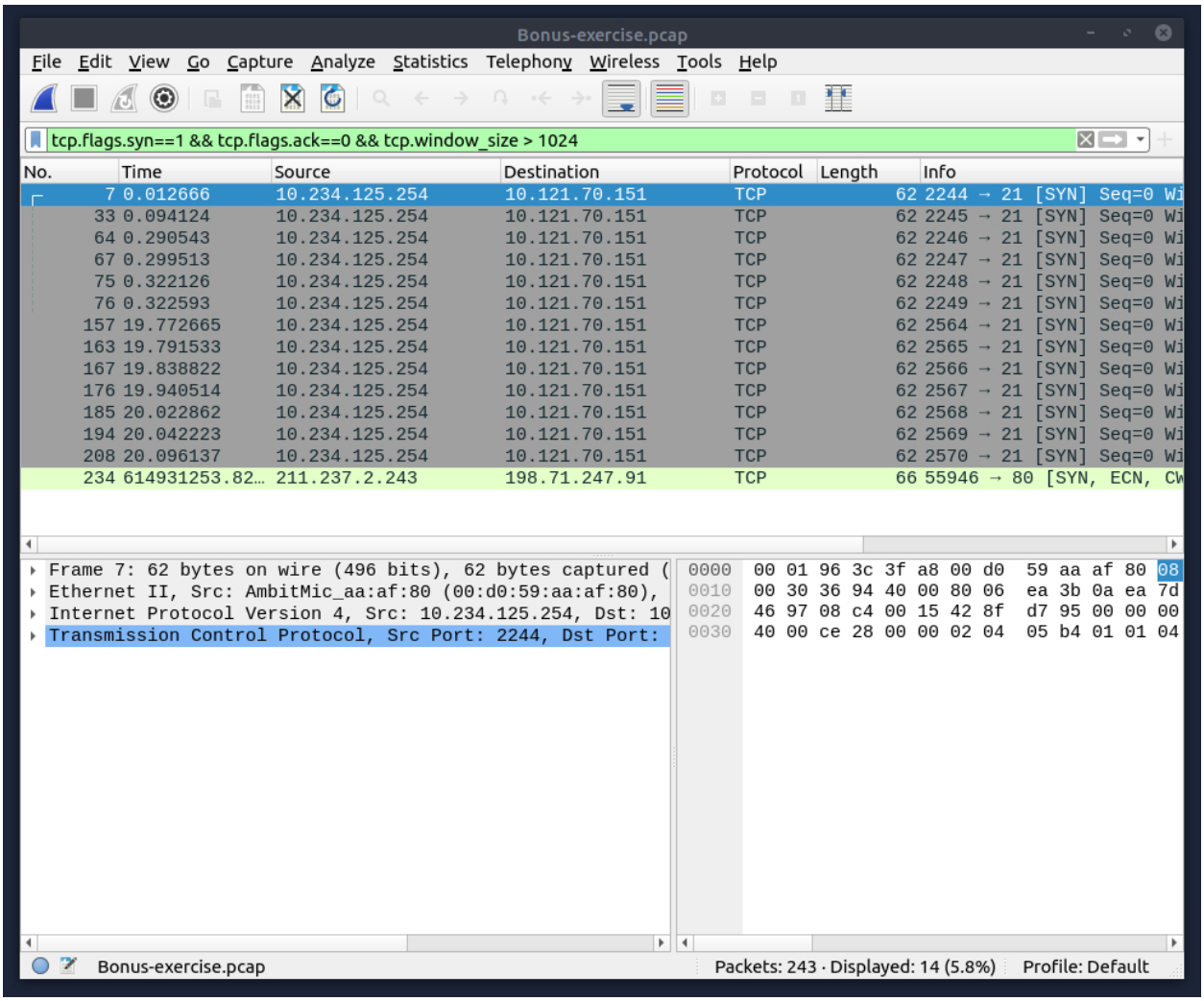


Closed TCP port (Connect):



- 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



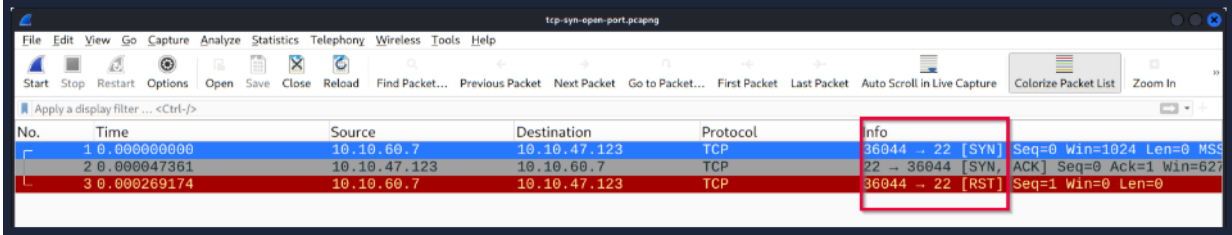
- tcp.flags.syn==1 <--- This filter for packets where the SYN (synchronize) flag is set.
- tcp.flags.ack==0 <--- This filter packets where the *ACK* (Acknowledgment) flag is set.
    a. in the TCP three-way handshake, the FIRST packet sent is a "pure" *SYN* packet (SYN=1, ACK=0). The SECOND (the response) 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

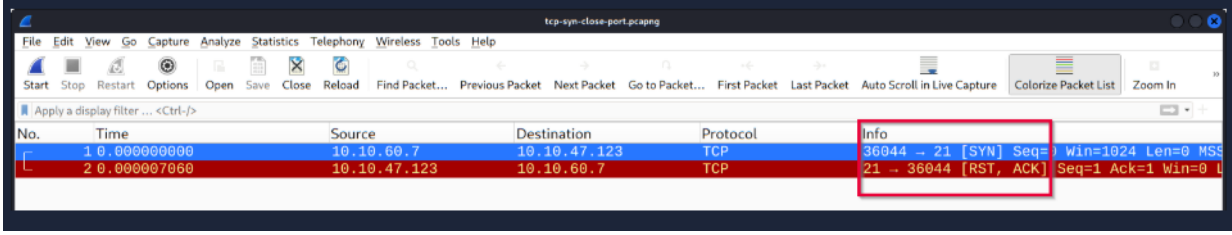
- Doesn't rely on the three-way handshake
- Usually conducted with nmap –sS <--- this performs a TCP SYN Scan, often called "stealth" or "half-open" scan. This Never completes the handshake.
- Used by privileged users.
- Usually, this have a size less than 1024 bytes as the request is not finished, and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

Open TCP Port	Close TCP Port
• SYN --> • <-- SYN,ACK • RST-->	• SYN --> • <-- RST,ACK

Open TCP port (SYN):

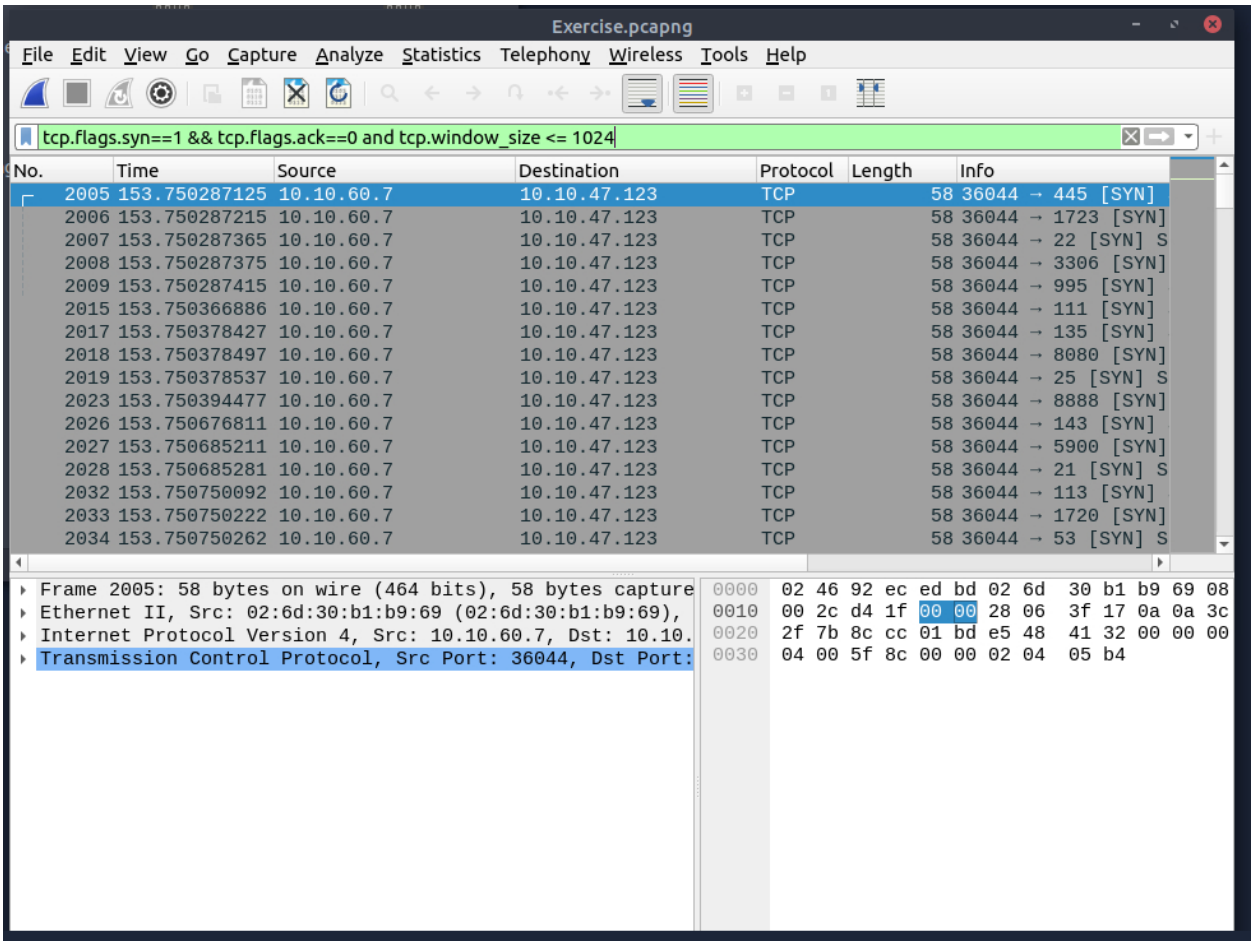


Closed TCP port (SYN):



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.



- tcp.flags.syn==1 and tcp.flgs.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

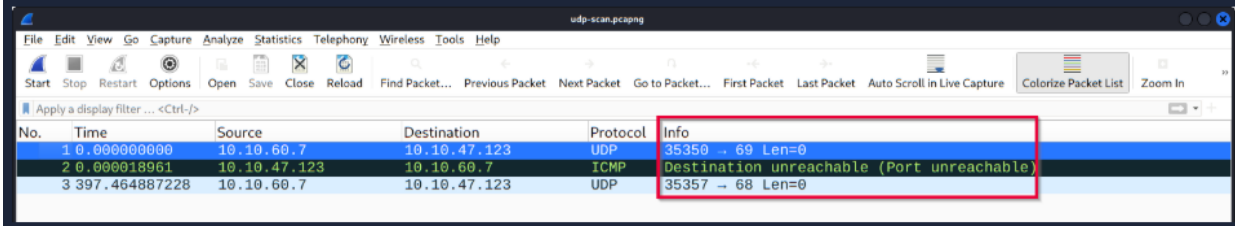
- This is used by network security professionals to detect Reconnaissance and Port Scanning
  a. Nmap Detection
  b. Botnet
  c. Filtering Noise

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 (port161/162), and DHCP (port 67/68) rely on UDP

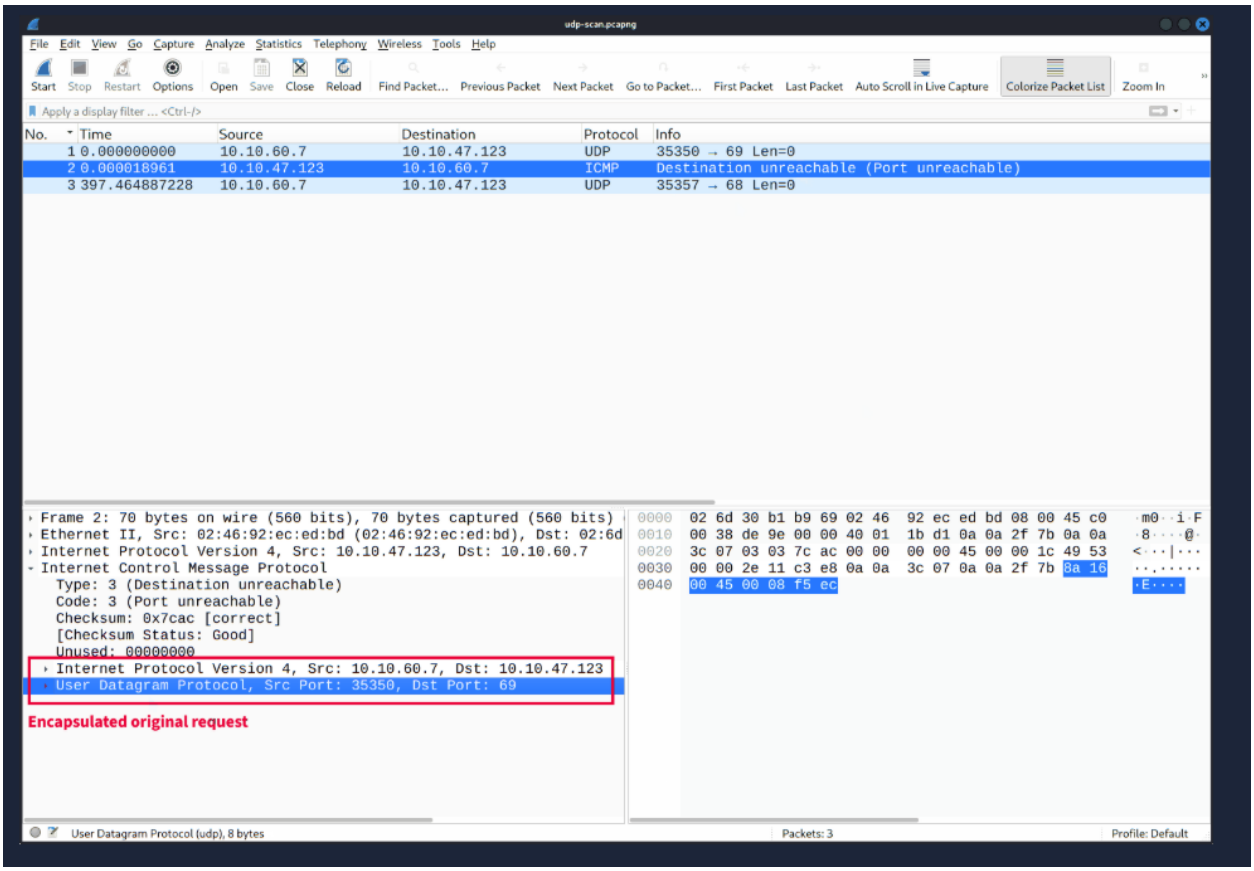
Open UDP Port	Closed UDP Portstrong>
• UDP packet -->	• UDP packet --> • ICMP Type 3, Code 3 message. (Destination unreachable, port unreachable)

Closed (port no 69) and open (port no 68) UDP ports:



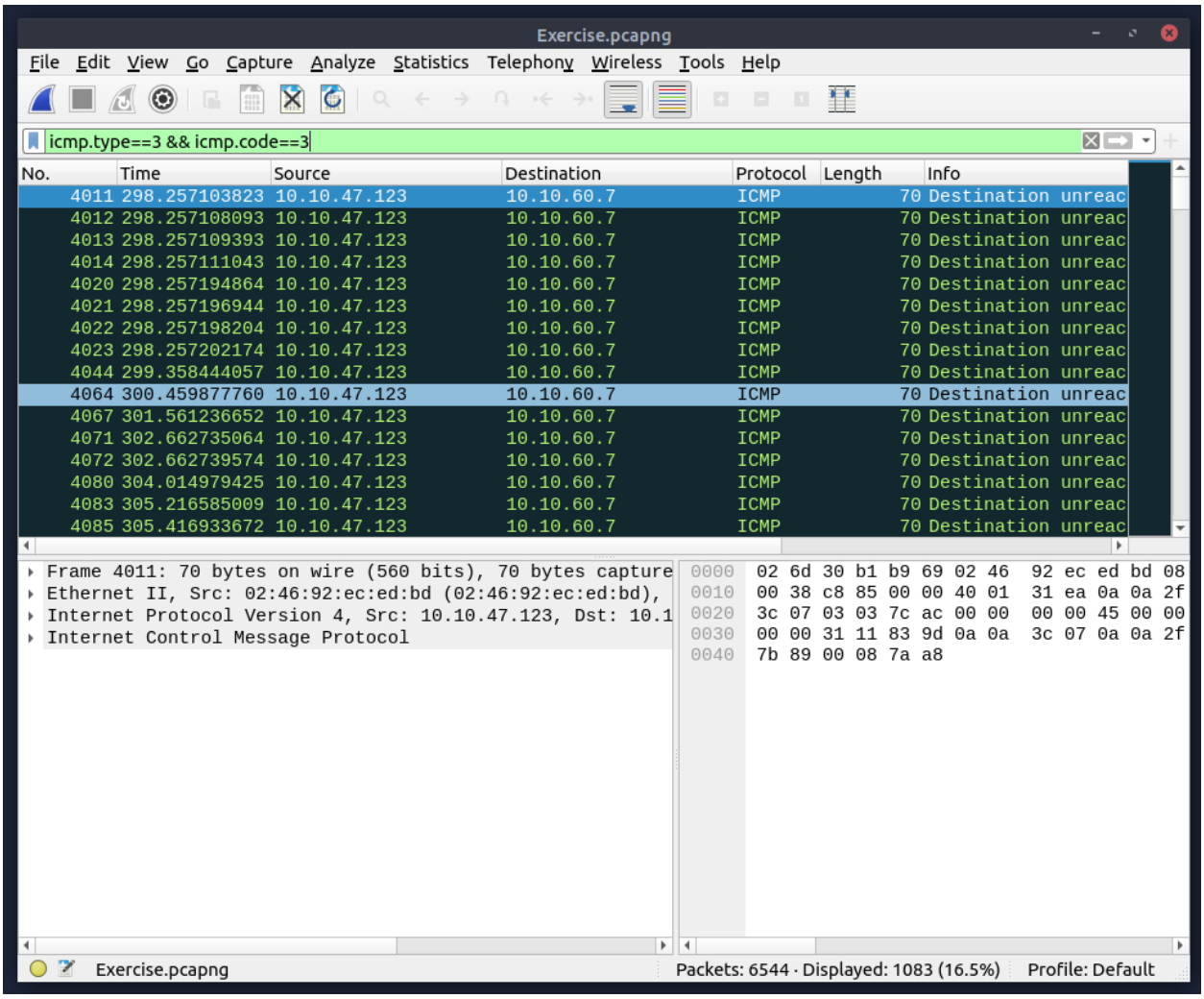
- this is showing a close port returns an ICMP error packet. The ICMP error message uses the original request as enacapulated data to show the source/reason of the packet.

- 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:

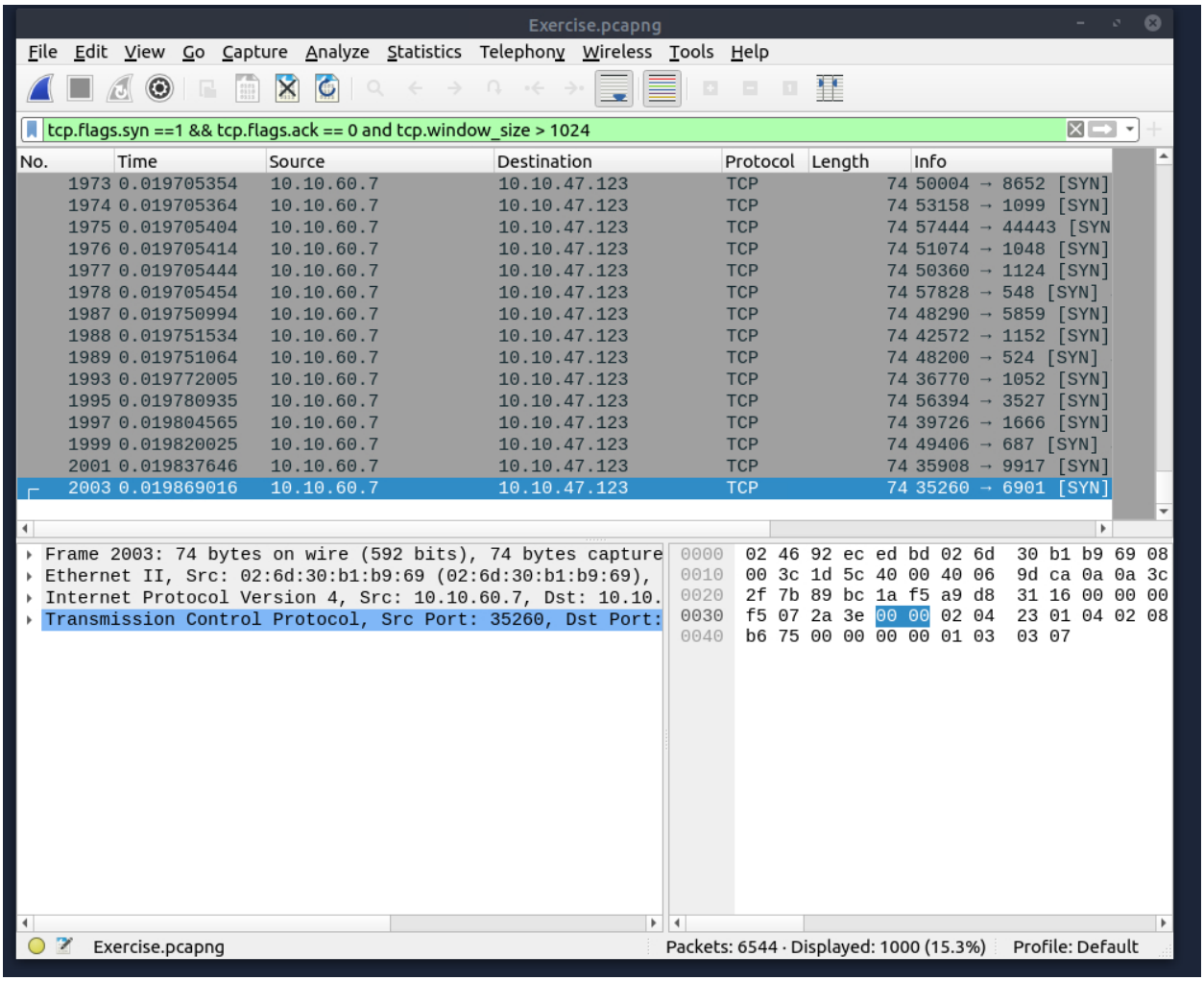
icmp.type==3 && icmp.code==3 <--- this is identifying ICMP Port Unreachable Message (one of the most important ICMP filter to monitor)



- 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.

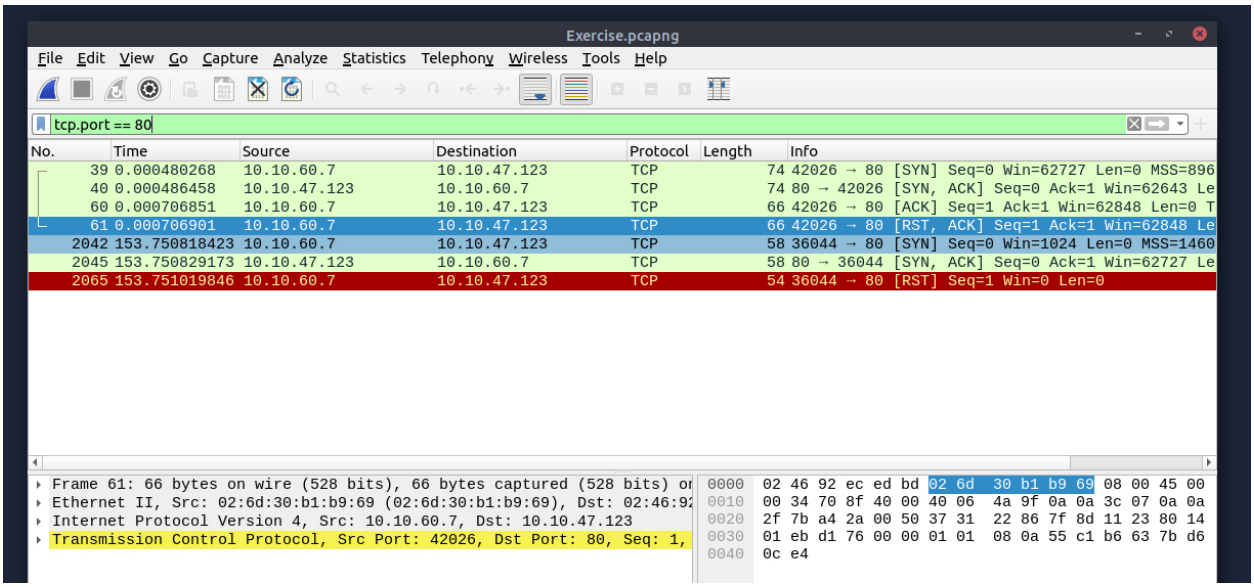
1. **What is the total number of the "TCP Connect" scans?**
   - To identify the total TCP connected, the following filter is needed to determine the number.
     - tcp.flags.syn == 1 && tcp.flags.ack ==0 && tcp.window_size > 1024 <--- this filter is identifying the initial TCP connection request that 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



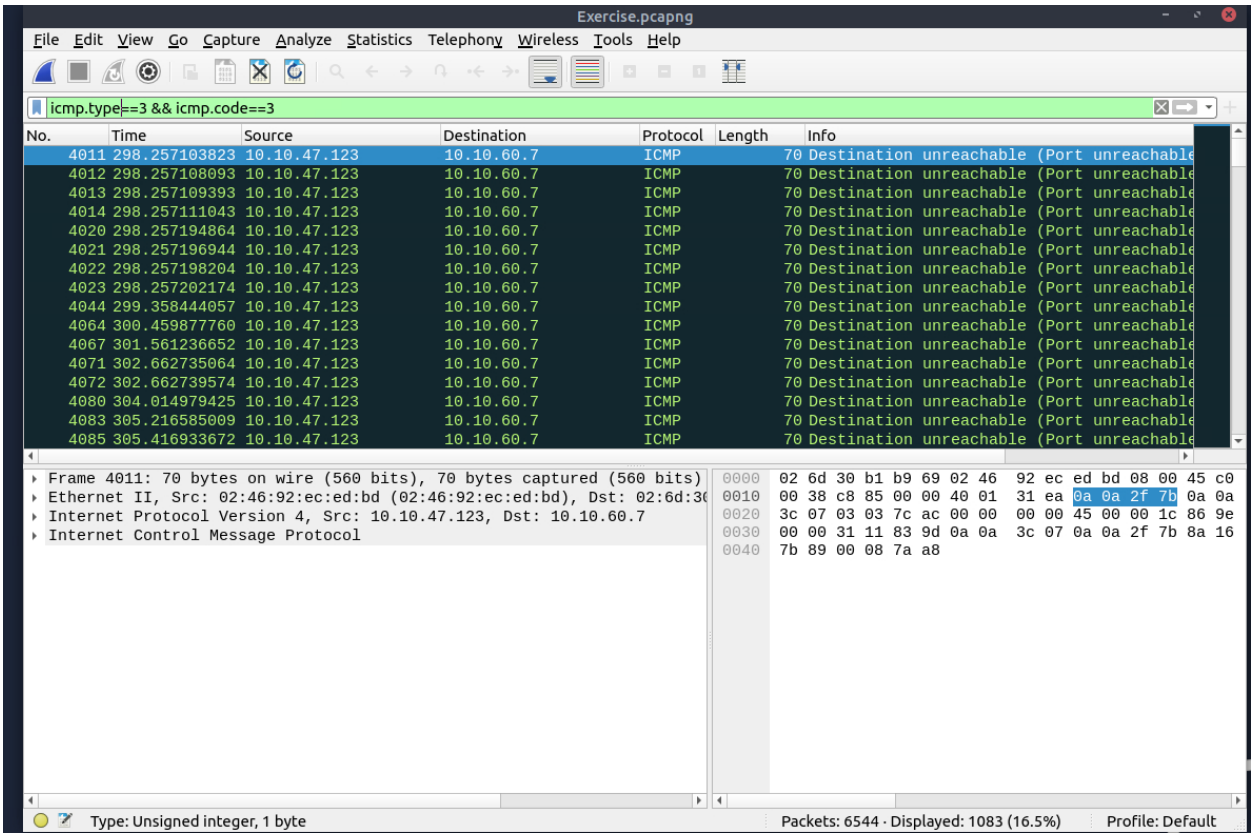
Answer: 1000

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



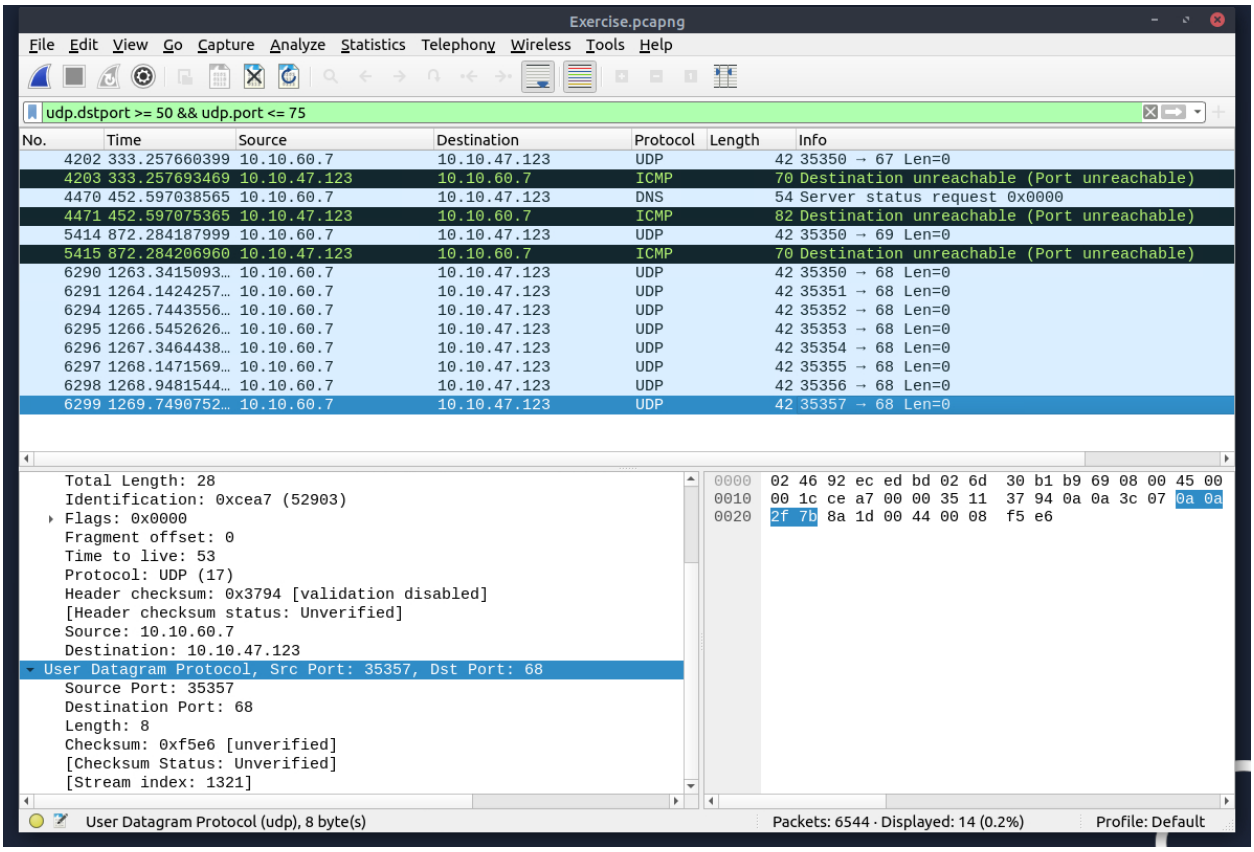
Looking at the info column, we can see that this is attempting a connection request
Answer: TCP Connect

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 iis asking Wireshark to look for specific ports that are open with the range. We will see the results below:



Answer: 68

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