**CSE3052 - INFORMATION SECURITY MANAGEMENT**

**DIGITAL ASSIGNMENT-5**

**ALOKAM NIKHITHA**

**19BCE2555**

**Wireshark**

**TITLE:**

**Wireshark Captures**

**AIM:**

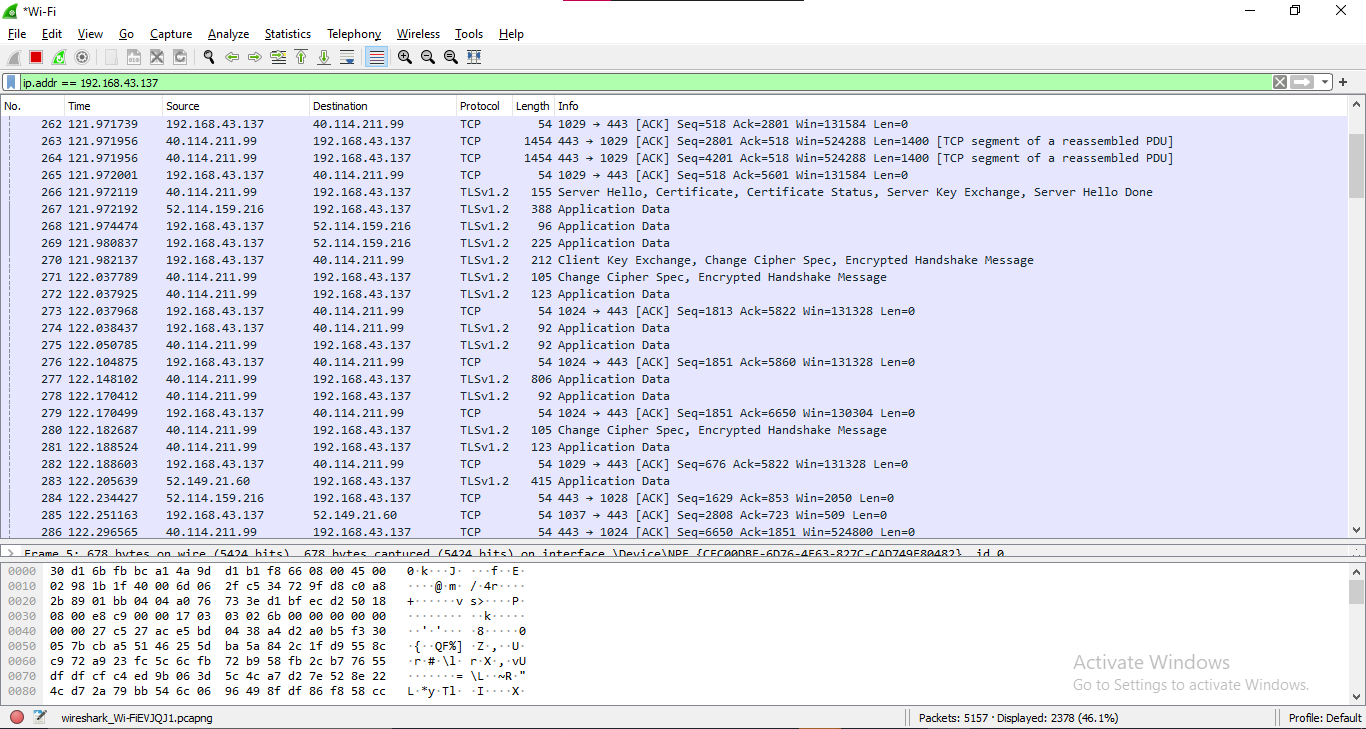
**To capture packets with various filters in Wireshark**

**PROCEDURE and Related Screen shots:**

**1. Filter traffic on specific IP address**

**It filters those traffic that have the specified IP address**

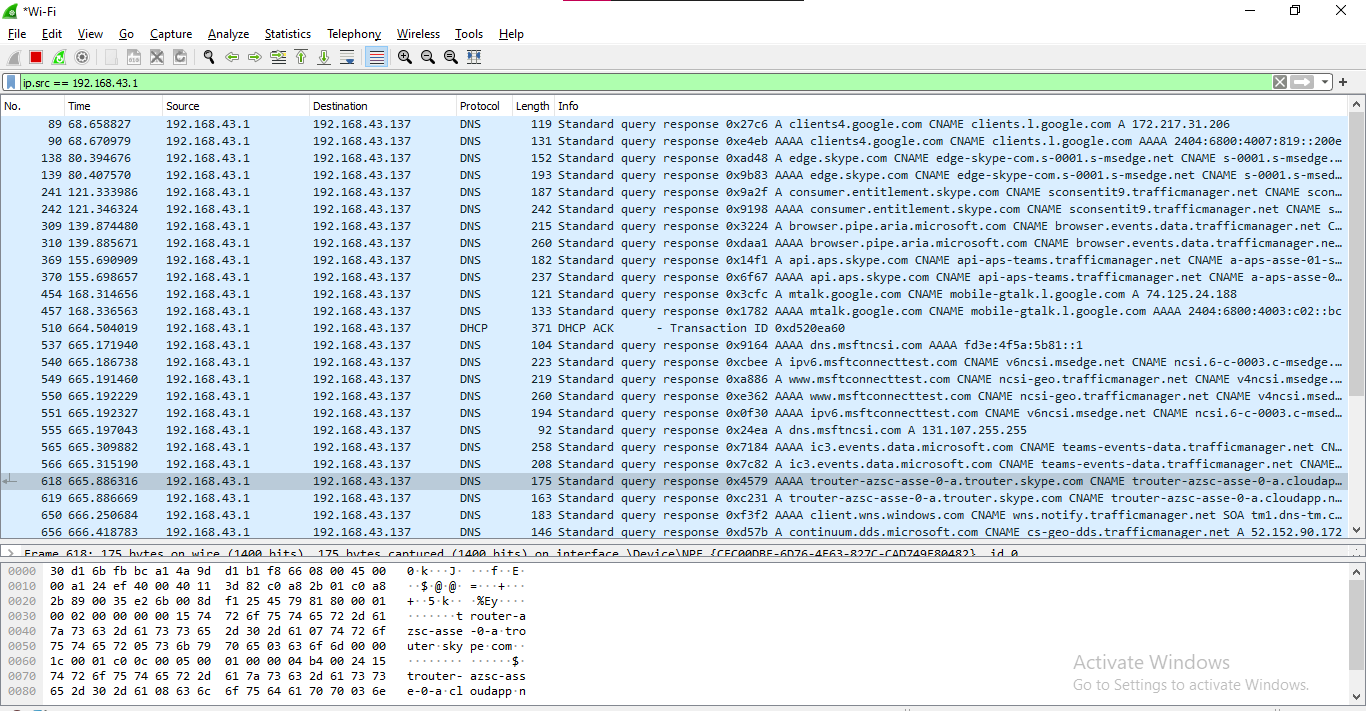
ip.addr == 192.168.43.137

****

**2. Filter by source address**

**Displays all the packets that have the specified source address**

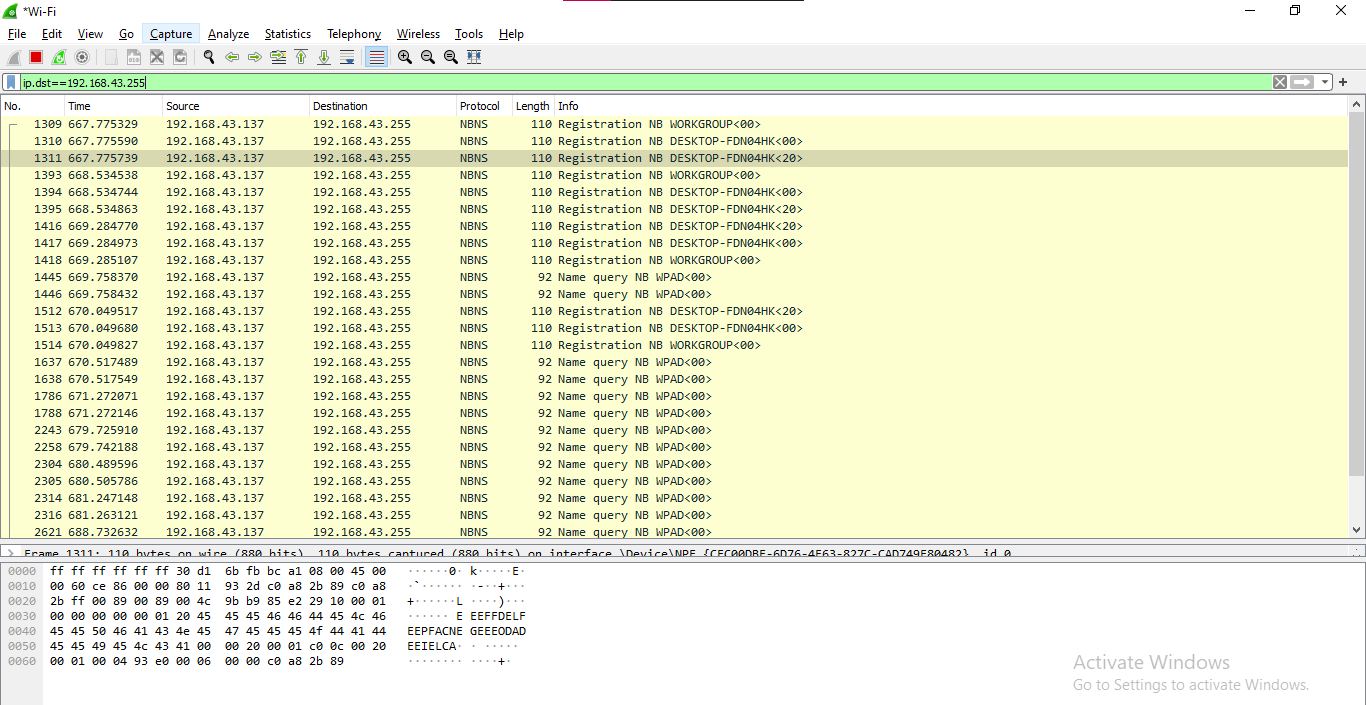
ip.src == 192.168.43.1



**3. Filter by destination address**

**Displays all the packets that have the specified destination address**

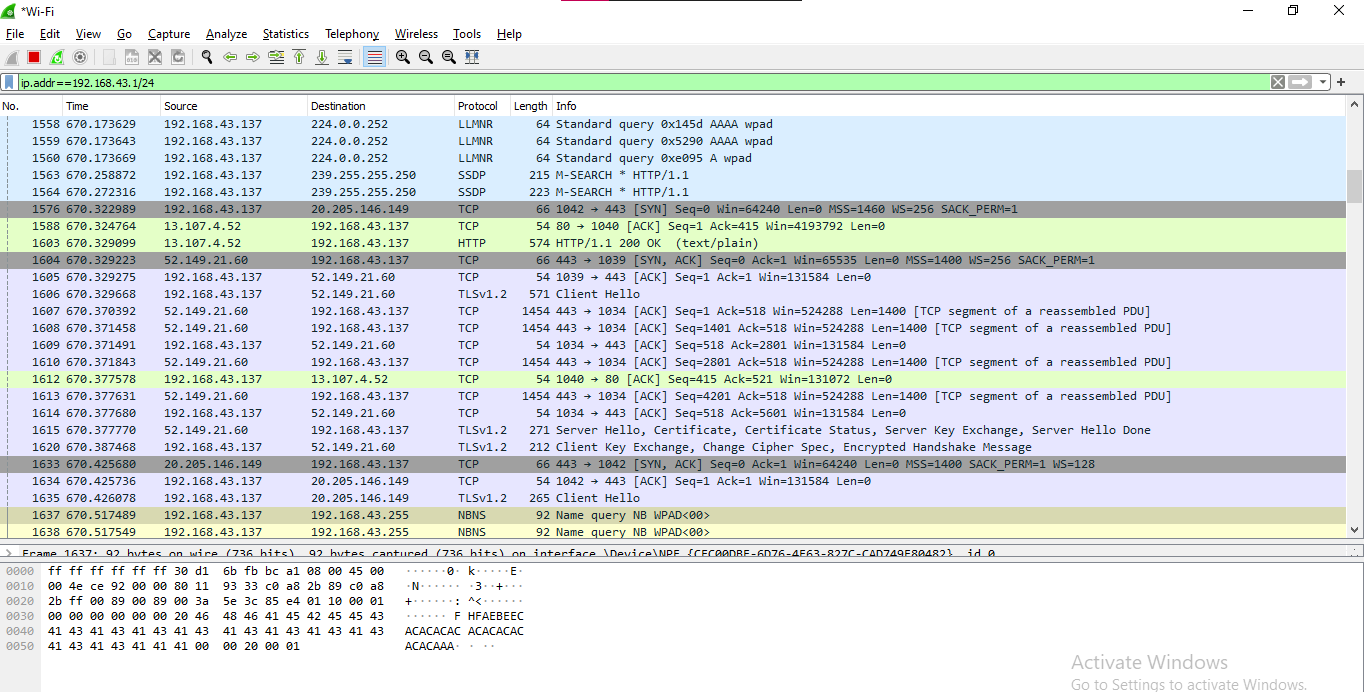
ip.dst==192.168.43.255



**4. Filter by IP subnet**

The mask does not need to match the local subnet mask since it is used to define the range. In order to display all the packet from 192.168.43.1, my display filter would be:

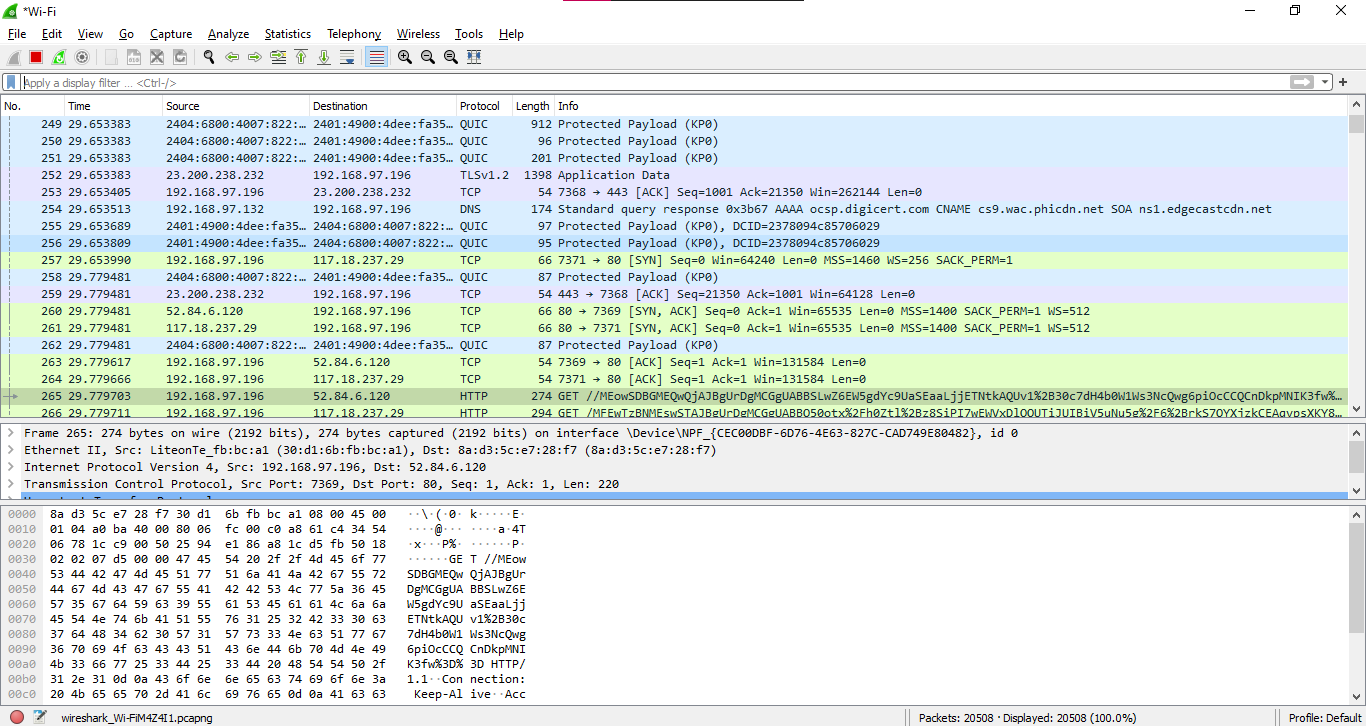
ip.addr==192.168.43.1/24

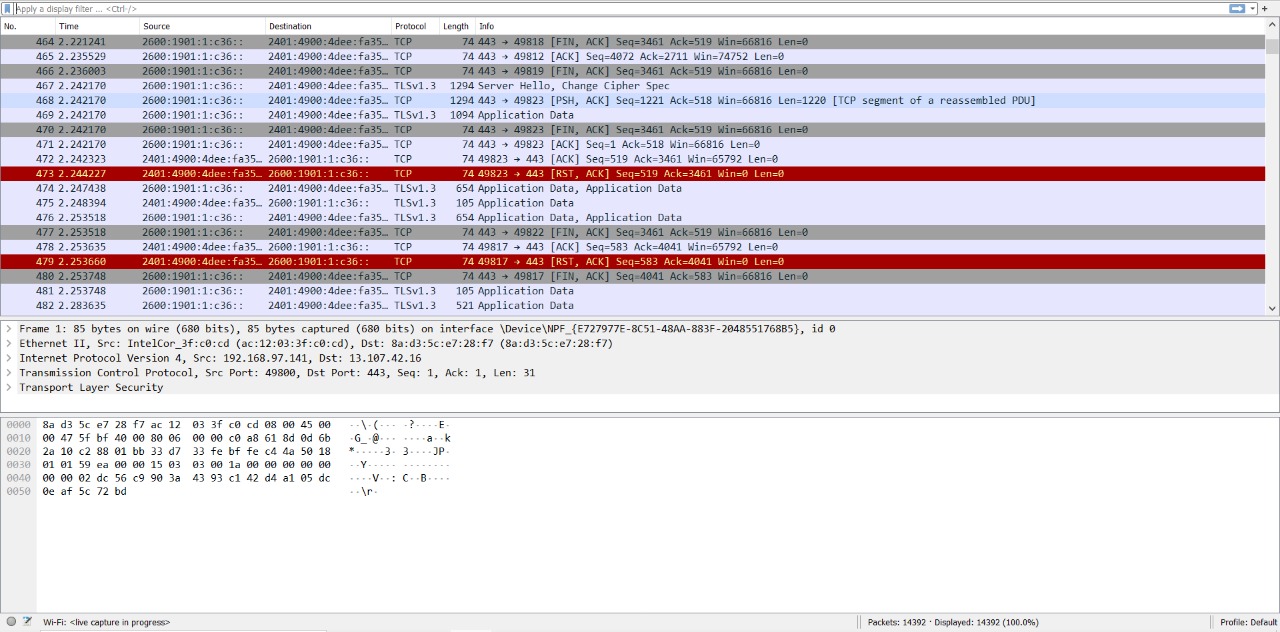


**5. Show all the packets**

It shows all the packets that are being captured by Wireshark while net surfing.

**ALL PACKETS:**

****

****

Here we are collecting all the packets on starting the capture Here we haven’t applied any filter. We can see that there are packets of different protocols which includes TCP (Transmission Control Protocol), HTTP (Hyper Text Transfer Protocol), OCSP (Online Certificate Status Protocol).

Here we are getting all are frames. We don’t apply any filters and get this default screen as soon as we start capturing them.

We can see frames of different protocols which includes TCP (Transmission Control sProtocol), HTTP (Hyper Text Transfer Protocol), OCSP (Online Certificate Status Protocol).

Further we can see that all our TCP packets are acknowledged owing to the fact that TCP is a reliable protocol and deploys three-way handshaking rule to ensure that the packets are delivered at required destination.

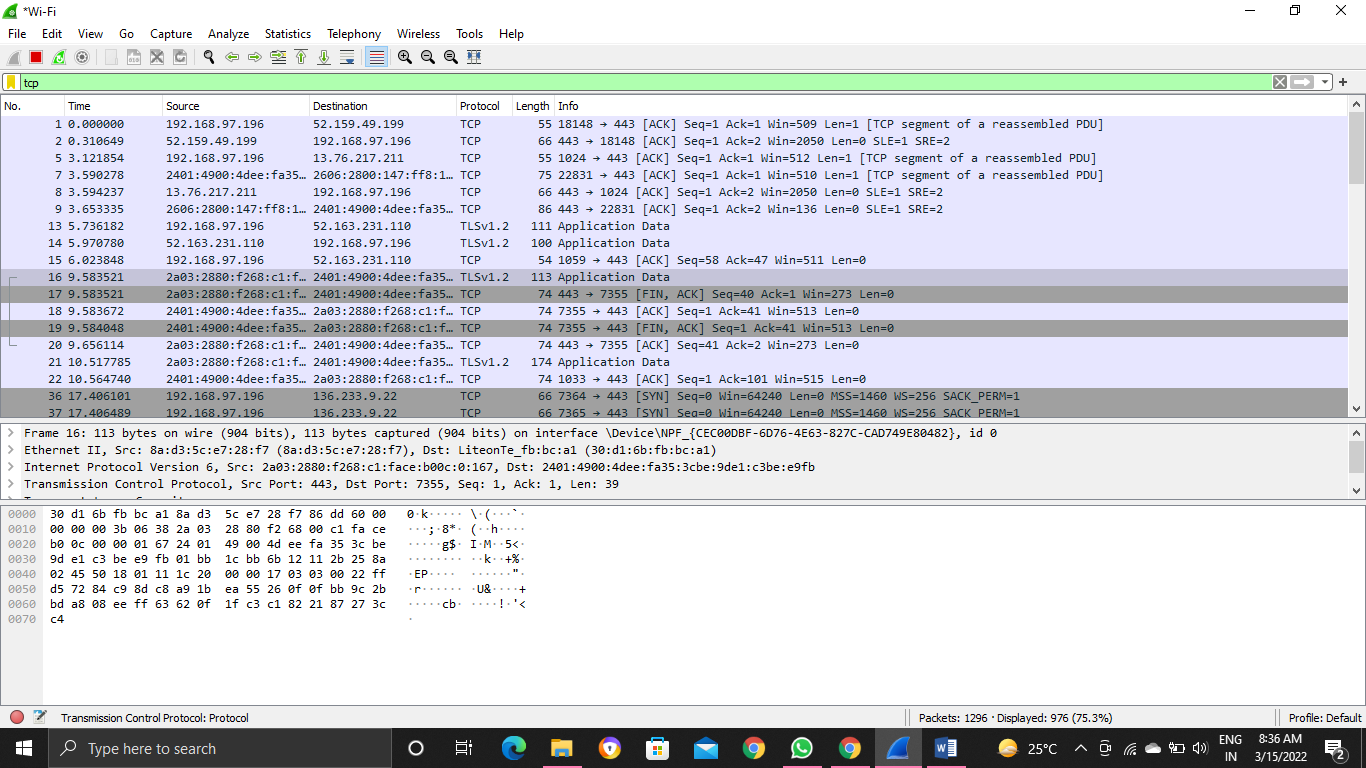
In the snippet we can see that there are few TCP packets colored in black. These are incorrect transmissions. In the first transmission we can see the error message of incorrect acknowledgement. This implies that the TCP packet didn’t satisfy the error check and control scheme. They seem to be a dirty read or incorrect checksum value while acknowledging the sequence number of that packet. Soon after we can see another black colored TCP packet. This packet is a retransmission packet as we need to re-send the packet which was not accepted by the receiver.

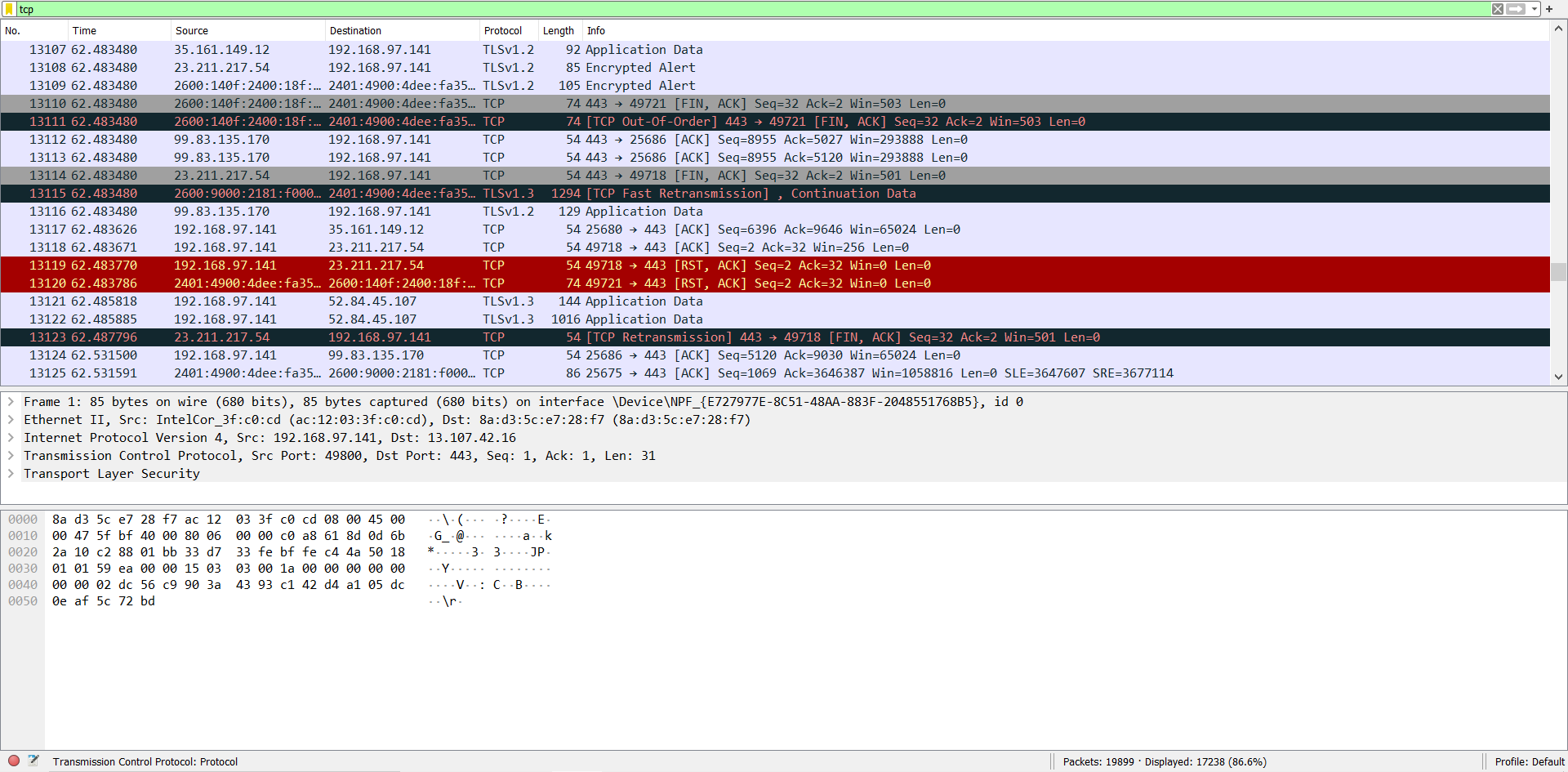
Next, we can see yet another TCP packet colored in black. This is a packet again being retransmitted because the sender didn’t receive an acknowledgment of it while transmitting it.

In our second screenshot we can see that two of the TCP packets are colored in red. The message here is [RST, ACK]. This implies that the connection is being reset and being halted. It ends a TCP session. The packet is acknowledging the previous packet stream and RST is used to reset the connection.

**6. TCP**

TCP stands for Transmission Control Protocol. It is a transport layer protocol that facilitates the transmission of packets from source to destination. It is a connection-oriented protocol that means it establishes the connection prior to the communication that occurs between the computing devices in a network. This protocol is used with an IP protocol, so together, they are referred to as a TCP/IP.





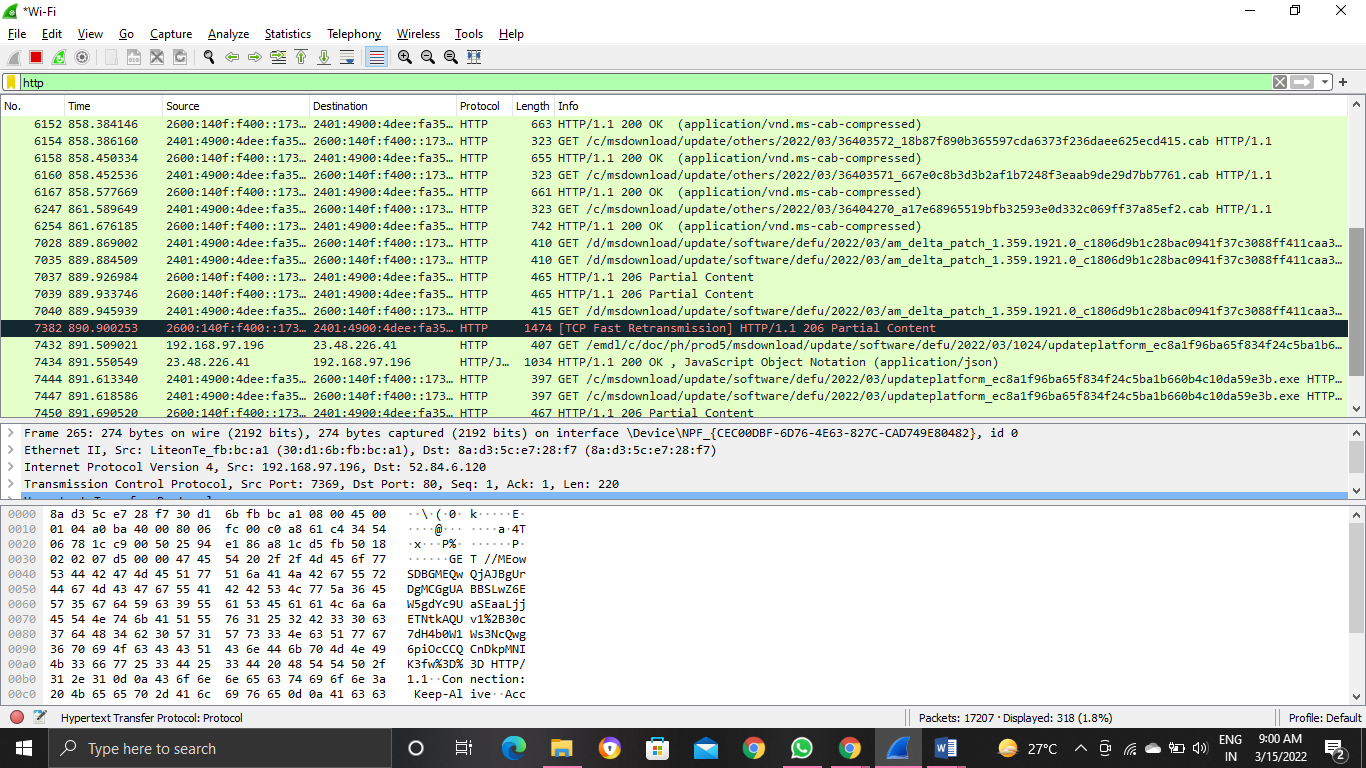
Only TCP packets have been blocked in this case. We can see that [ACK] is used to acknowledge these packets once again. We also retrieve the length of each packet, which varies from 54 bytes to 1474 bytes in our case.

Many coloured TCP packets may be seen in our second screenshot. TCP out-of-order, TCP Fast Retransmission, and TCP Retransmission are some of the error messages we see.

1) TCP Out-of-Order: This refers to the fact that a frame was received in a different order than it was sent, i.e., it was acknowledged after a later packet was received first in sequence. It is not usually a problem; it indicates that there are numerous paths between the source and the destination, and one of them takes a longer route.

Here we applied tcp filter and we are able to collect all the tcp packets .

**7. http**



Here we applied http filter and we are able to collect all the http packets .

We've filtered out all of the packets that are part of the HTT Protocol. Packet details such as source IP address, destination IP address, length, and information are visible.

The HTTP Status of Transmission can be found under the Information about Packets section. The status in our case is either 200 or 204.

1) HTTP 200 OK success code: The HTTP 200 OK success code indicates that the request was successful. By default, a 200 response can be cached.

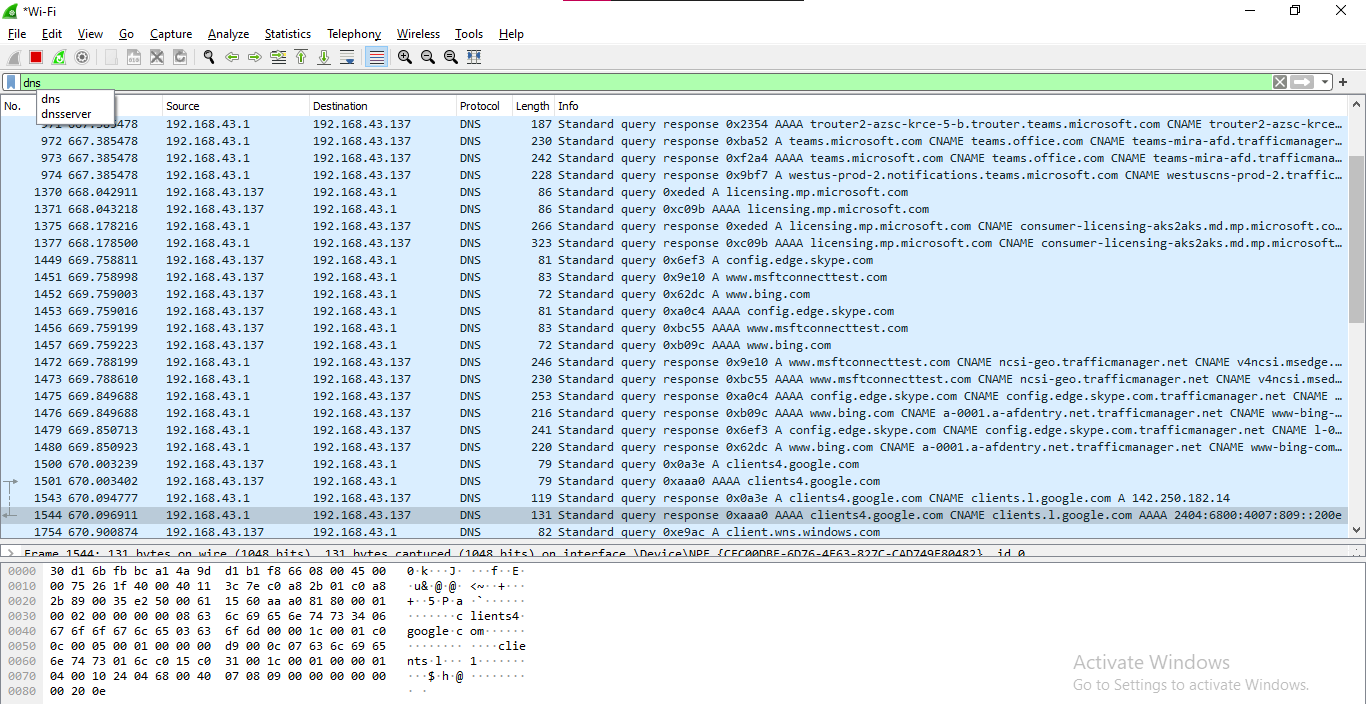
2) HTTP 204 No content success: The HTTP 204 No content success status response code indicates that a request has been completed, but the client does not need to leave its current page.This could be used, for example, when a wiki site's "save and continue editing" feature is implemented. The page would be saved using a PUT request, and the 204 No content response would be sent to signal that the editor should not be replaced by another page.

**8. Filter trаffic bаsed on protocol**

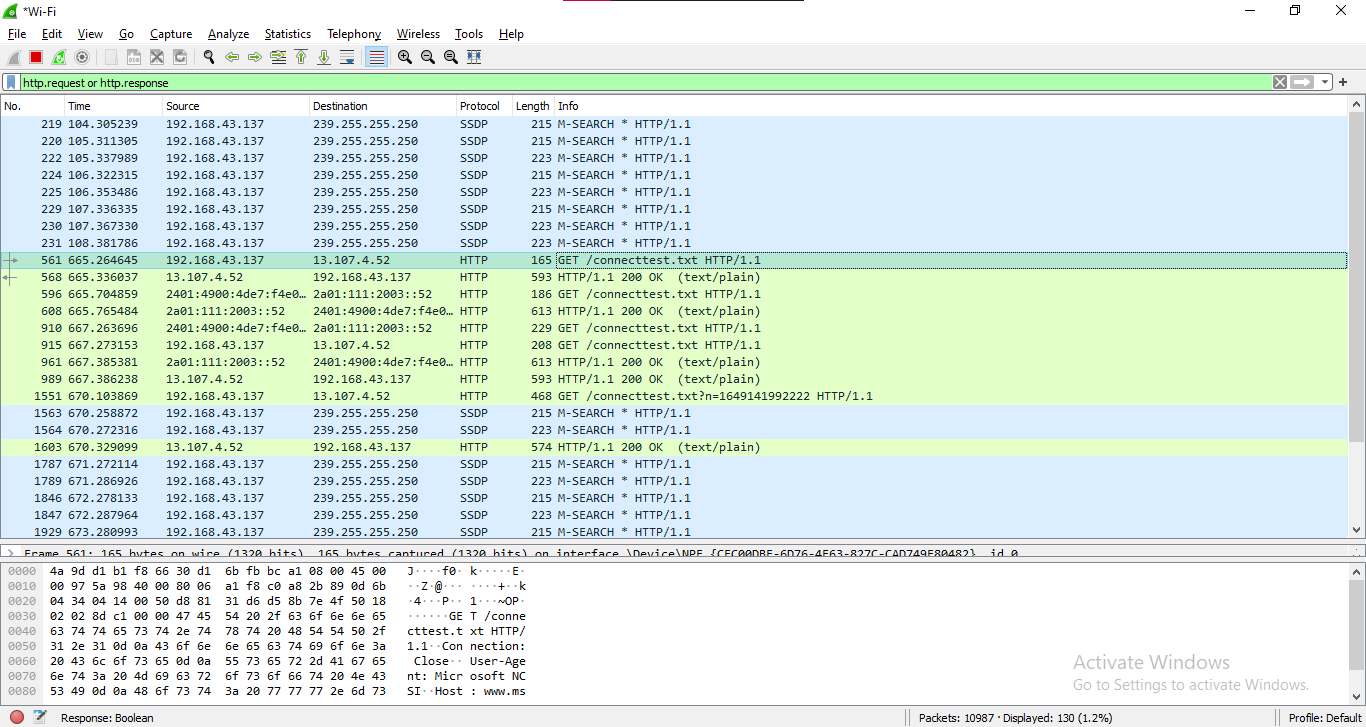
**DNS**

The Domain Name System (DNS) is the phonebook of the Internet. Humans access information online through domain names, like nytimes.com or espn.com. Web browsers interact through Internet Protocol(IP) addresses. DNS translates domain names to IP addresses so browsers can load Internet resources.

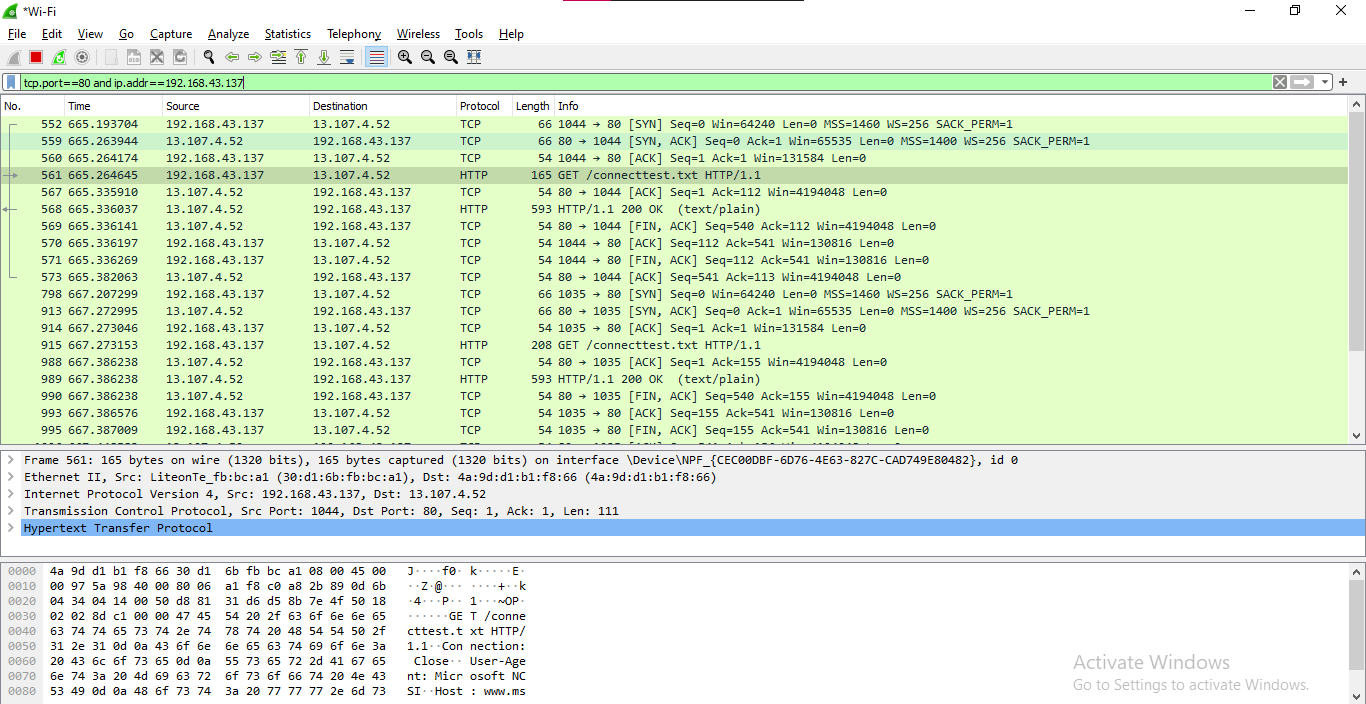
Each device connected to the Internet has a unique IP address which other machines use to find the device. DNS servers eliminate the need for humans to memorize IP addresses such as 192.168.1.1 (in IPv4), or more complex newer alphanumeric IP addresses such as 2400:cb00:2048:1::c629:d7a2 (in IPv6).



**9. http.request or http.response**



**10. tcp.port==80 and ip.addr==192.168.43.137**



**Conclusion:**

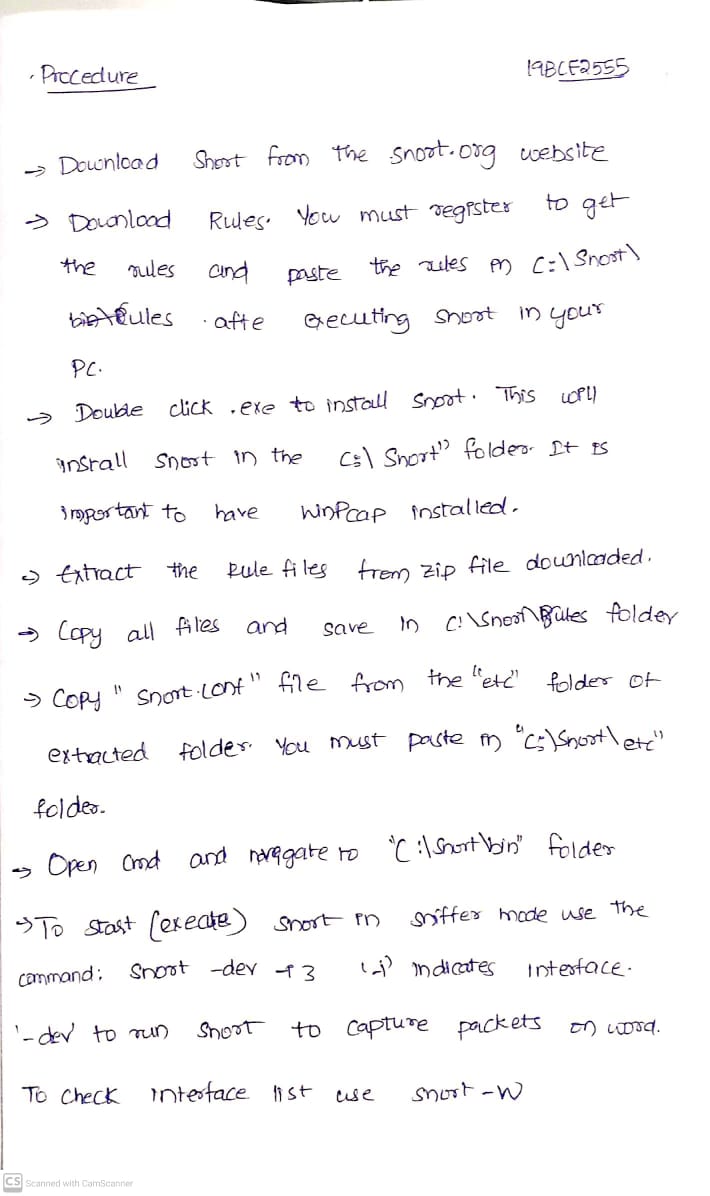
Here we have filtered all the packets and analysed different packets. We have summarized all the different errors we got during TCP transmission and also analysed various status states of HTTP.

**SNORT**

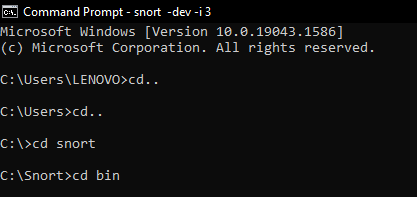
**Aim:**

**To Install and work with the SNORT Software**

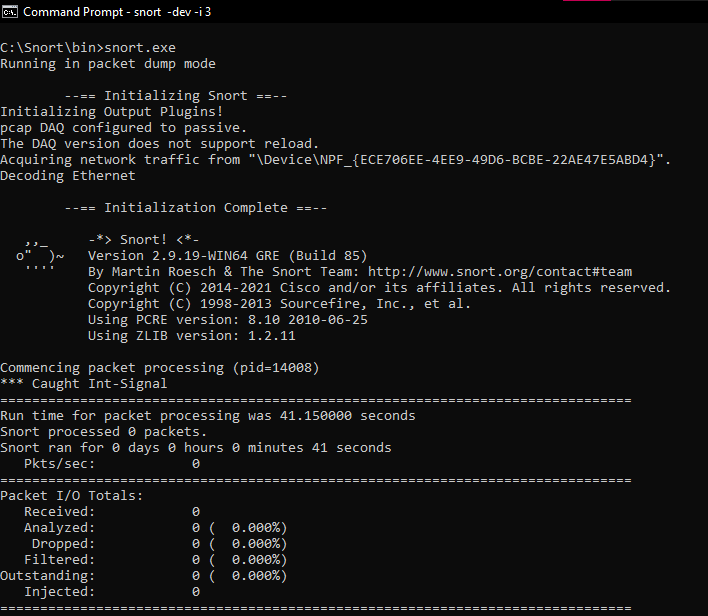
**Procedure:**

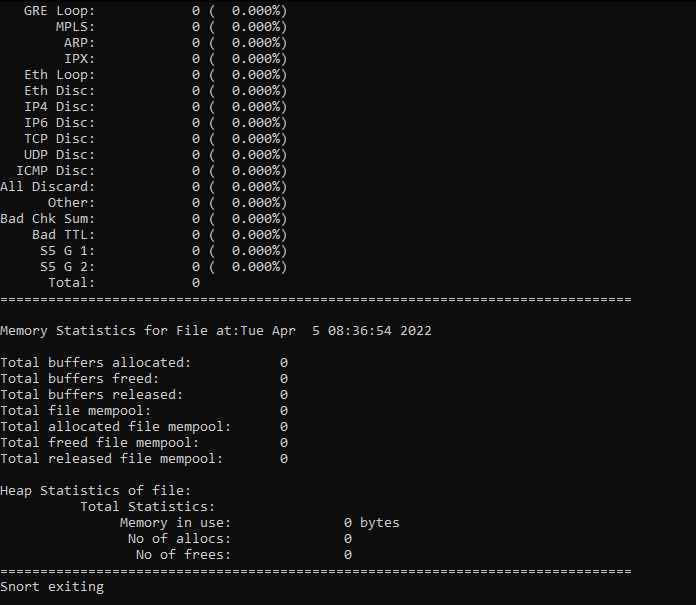
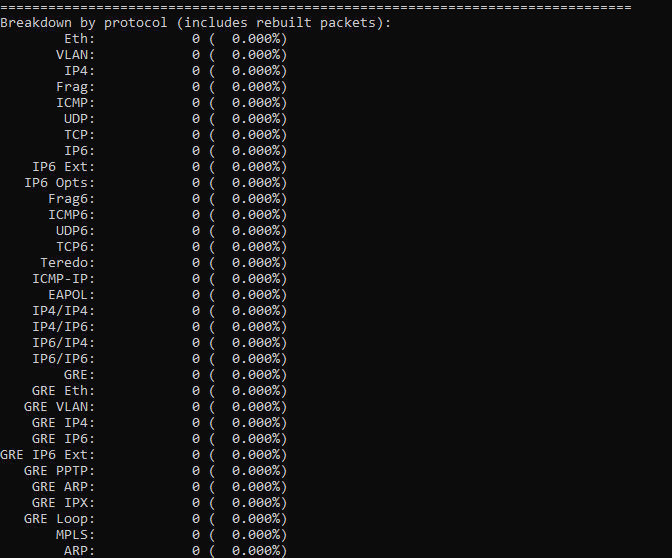
****

Moving to C:\Snort\bin directory in command prompt



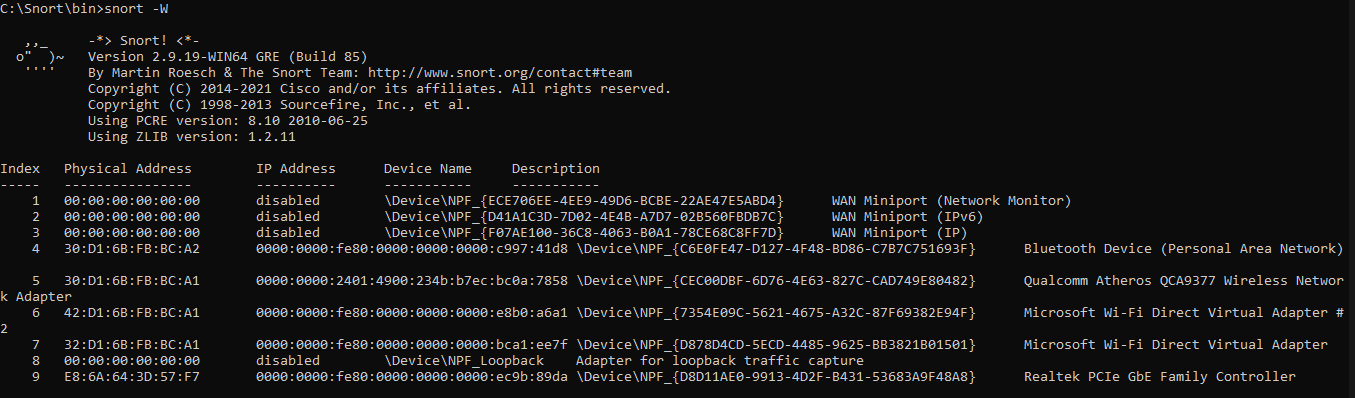
snort.exe for testing snort

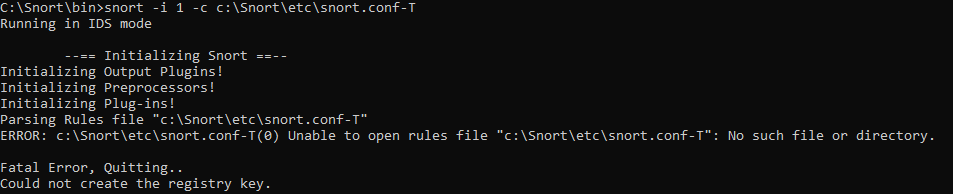




To check the interface list, use following command:

snort -W

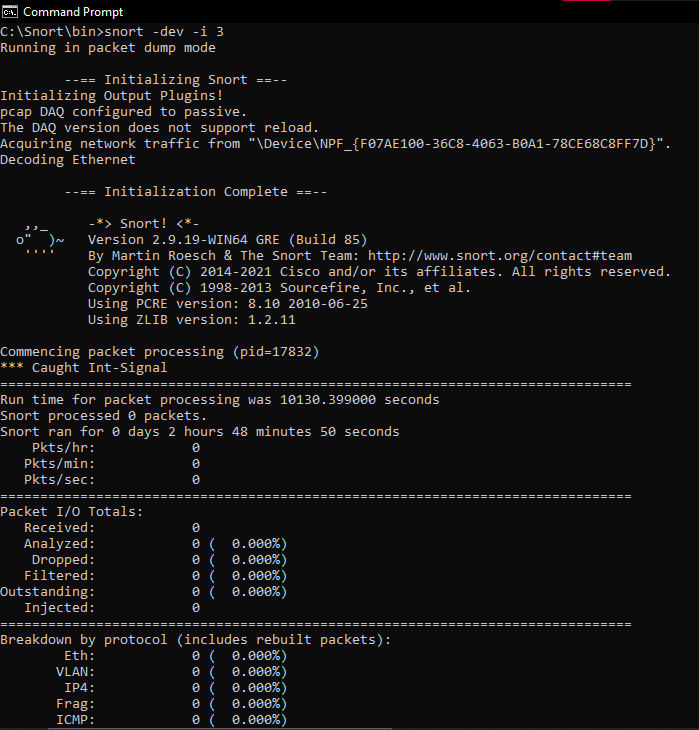


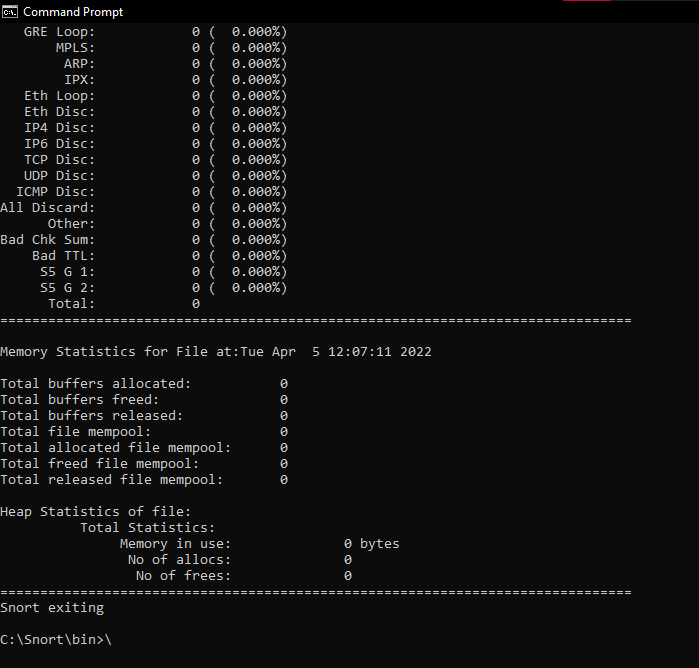
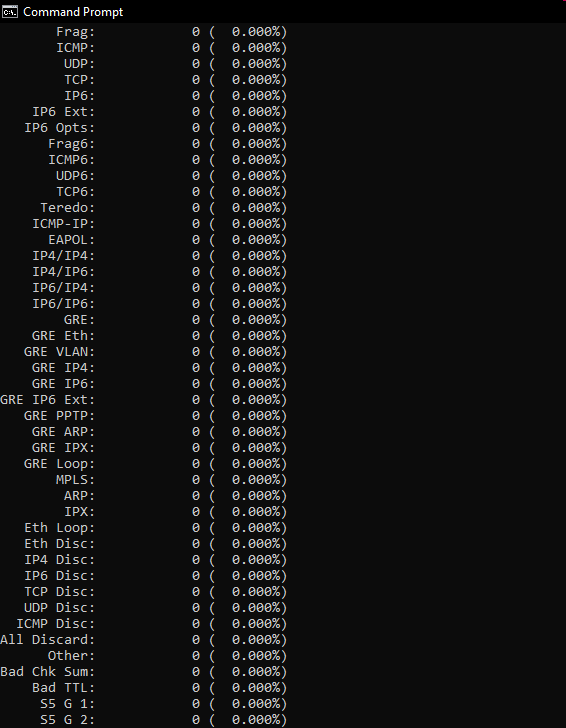


To start (execute) snort in sniffer mode use following command:

snort -dev -i 3

-i indicates the interface number. You must pick the correct interface number. In my case, it is 3. -dev is used to run snort to capture packets on your network.



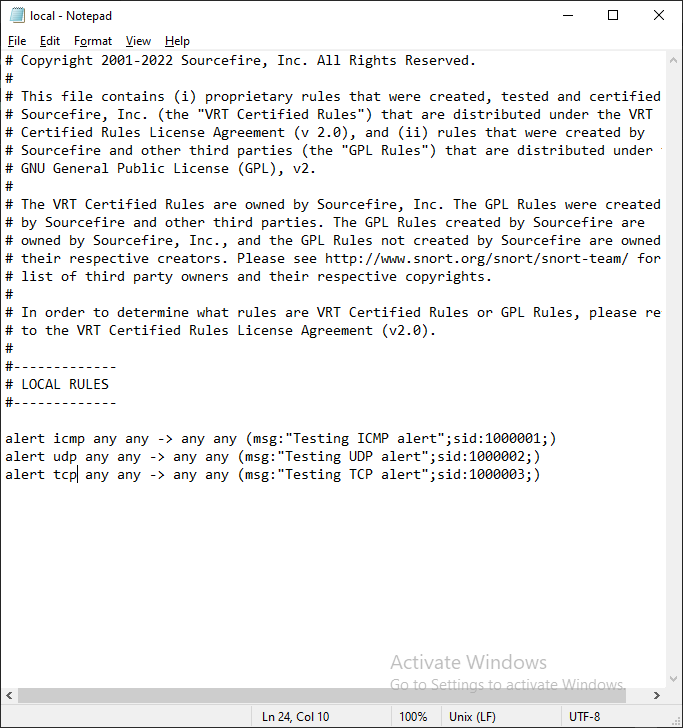


To run:

Text

Description automatically generated

Local Rules File:



We can add our own rules in the local file that is there in the c:\snort\rules\ directory.

On starting SNORT:

Text

Description automatically generated

On exiting (Ctrl+C)

–