Cybersecurity Lab report – 03

14/11/2024

USN:1BM22IC044

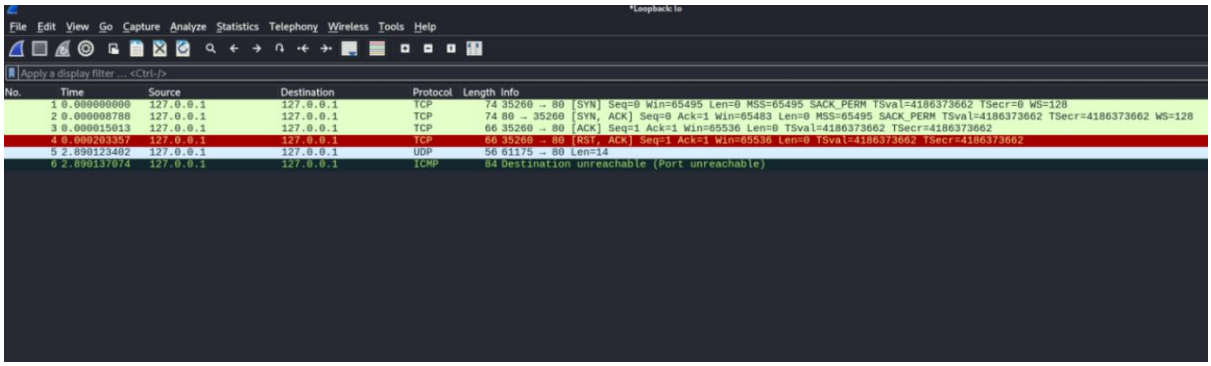
Network Analysis with Wireshark

Wireshark is a packet sniffing tool that allows us to capture network traffic, examine the contents of packets, view the protocols in use, and analyze the payloads. It can even reveal sensitive information, such as user credentials, if they are being transmitted. Additionally, Wireshark lets you filter packets based on specific protocols.

Step 1: Open the terminal and run Wireshark on the desired network interface.

wireshark -i lo

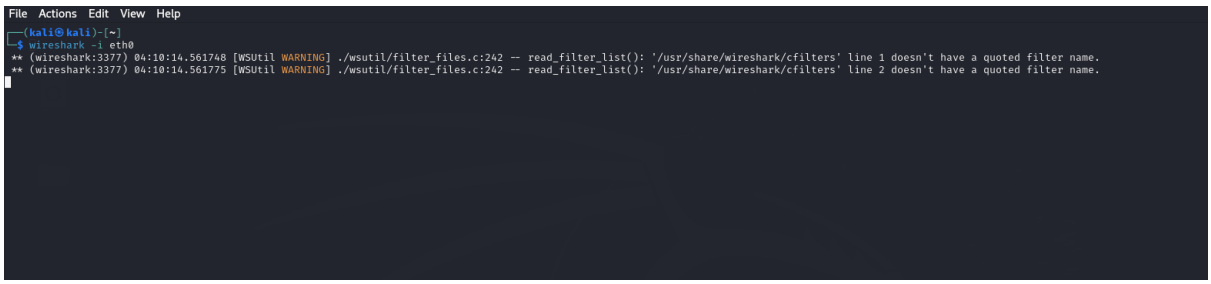
Here, I ran the command on the loopback interface to observe the color coding in Wireshark. Then, I executed Nmap commands with options like `-sU` and `-sT` to generate UDP and TCP packets, respectively. Additionally, I sent a ping to create ICMP traffic. The following output was generated as a result.



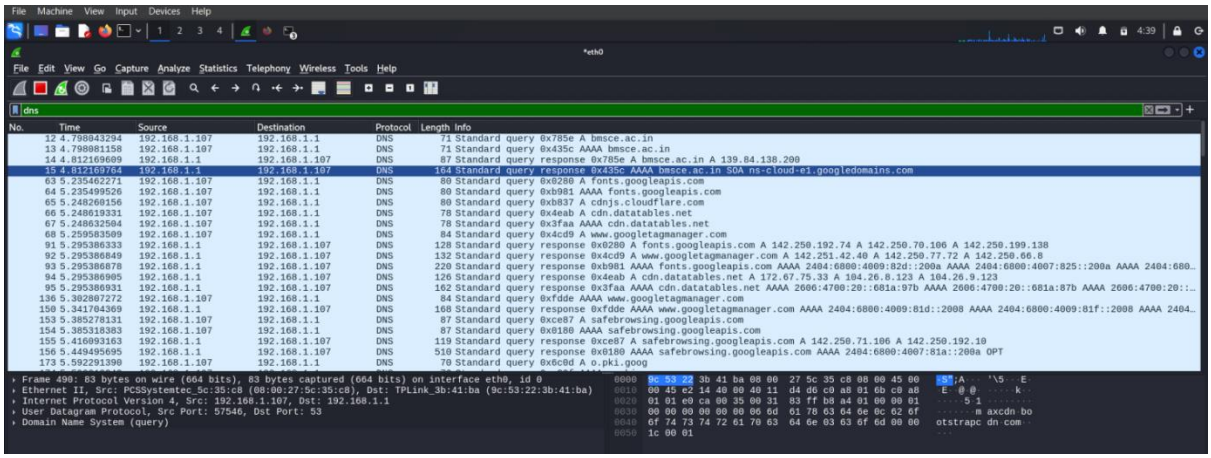
From this, I concluded that UDP packets were displayed in gray, ICMP packets were shown in green, and red packets indicated a reset (RST) or some kind of protocol violation. Additionally, I was able to observe the 3-way handshake that occurred during the communication.

Next, I ran Wireshark on the `eth0` interface using the following command:

`wireshark -i eth0`



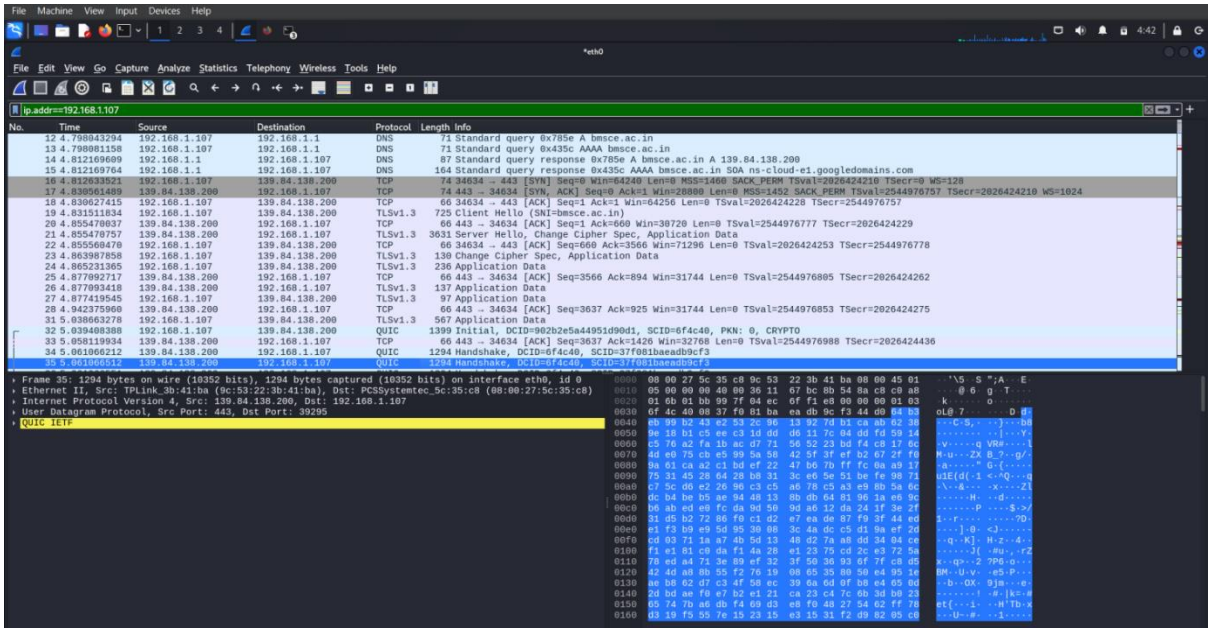
I opened the browser and searched for the website **bmsce.ac.in**. Since we know that DNS queries are sent during this process, I applied a filter in Wireshark to capture DNS traffic. As a result, I obtained the following output.



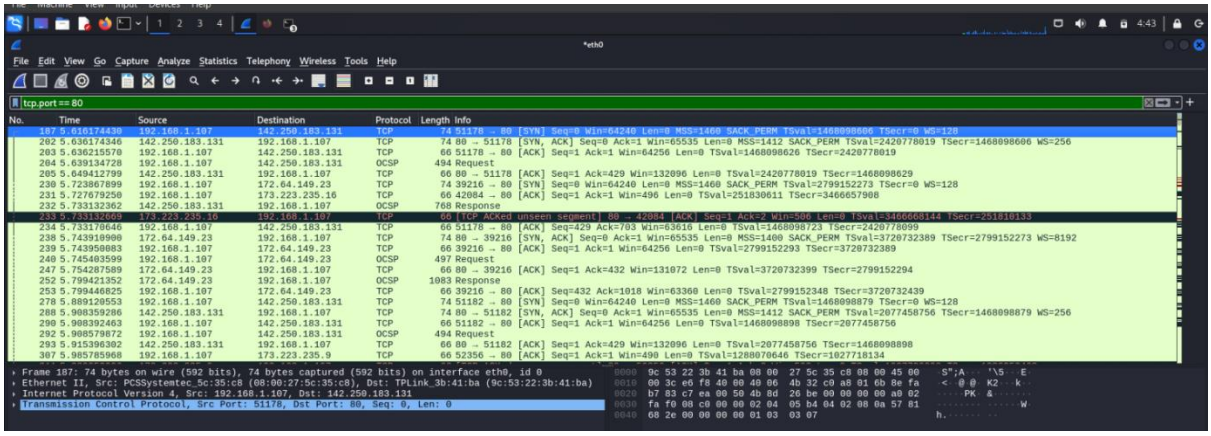
This showed that DNS packets were displayed in blue, which concluded the color coding part.

Packet Filtering:

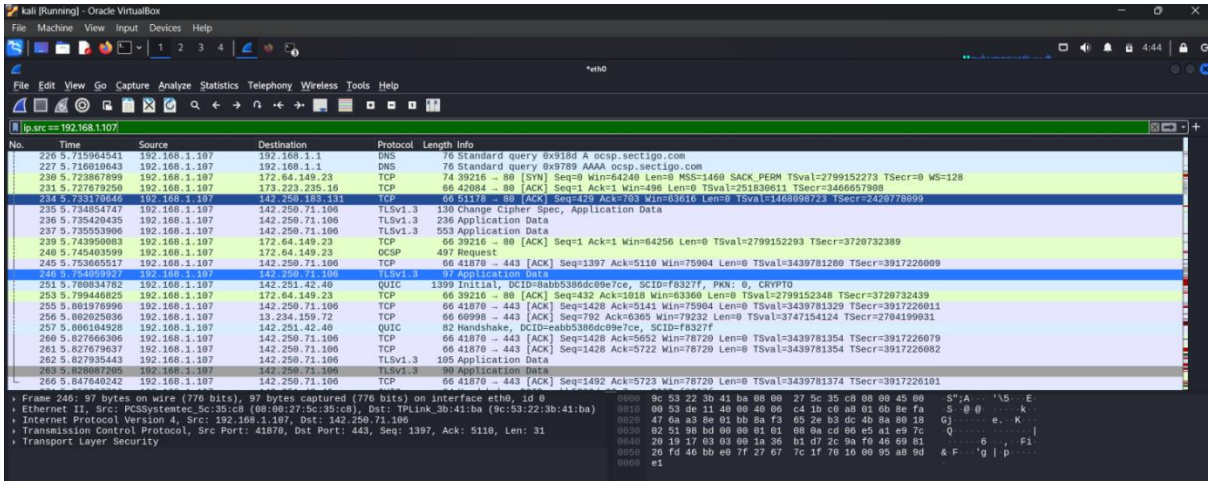
Next, I applied the filter `ip.addr==192.168.1.107` to capture packets related to the IP address 192.168.1.107. The following output was generated as a result.



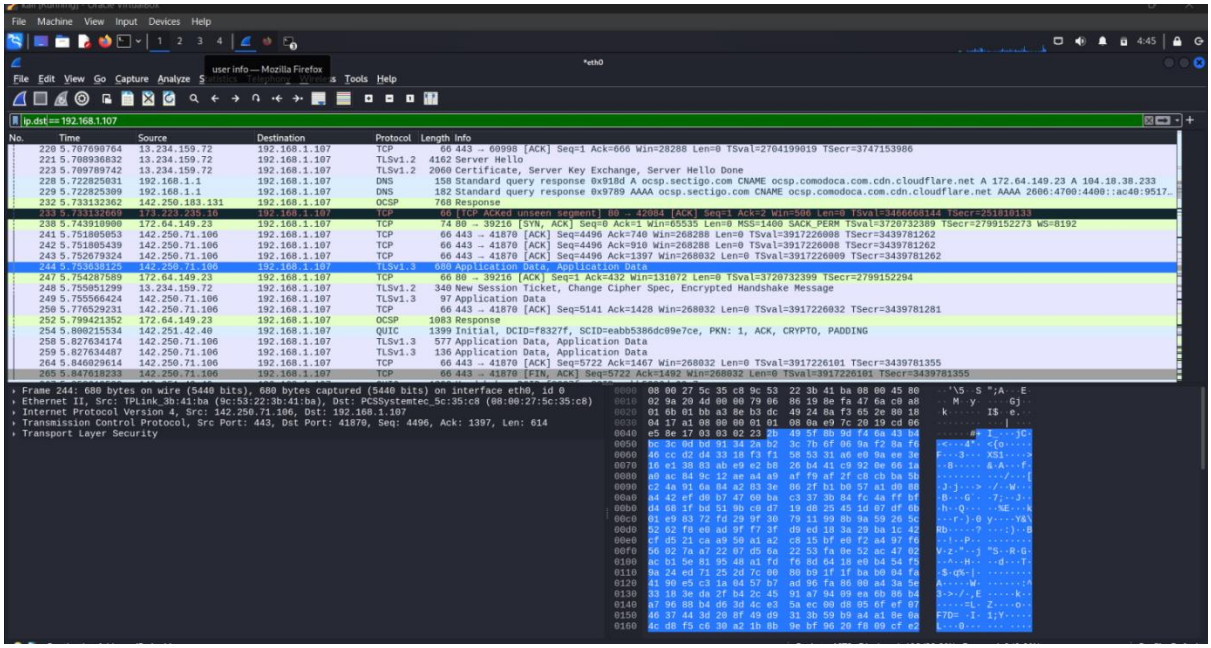
I entered the filter **tcp.port==80** to get the tcp packets consists of port 80 either in source or destination



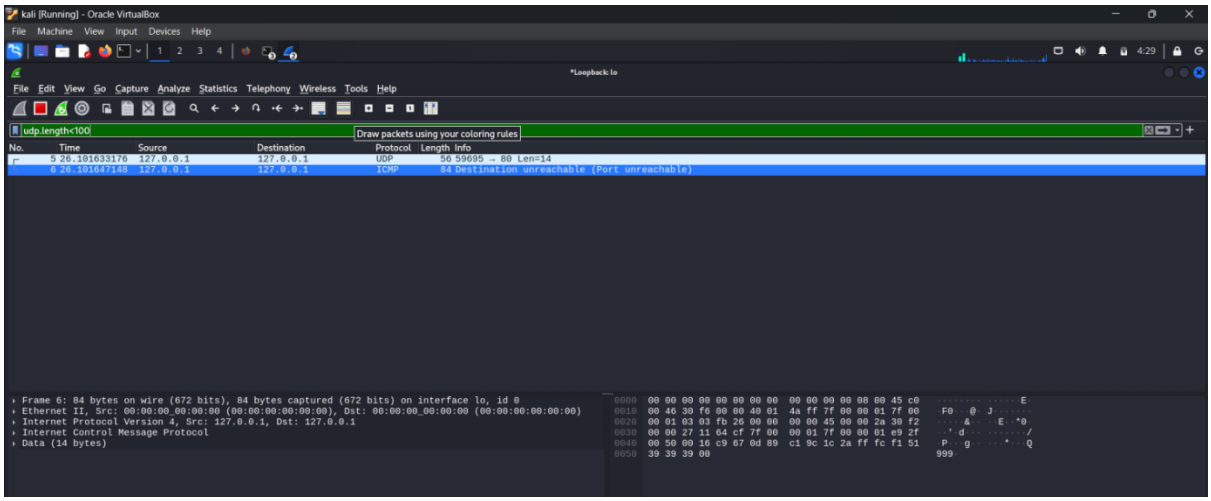
to filter the packets which has 192.168.1.107 has source I applied the filter **ip.src==192.168.1.107**



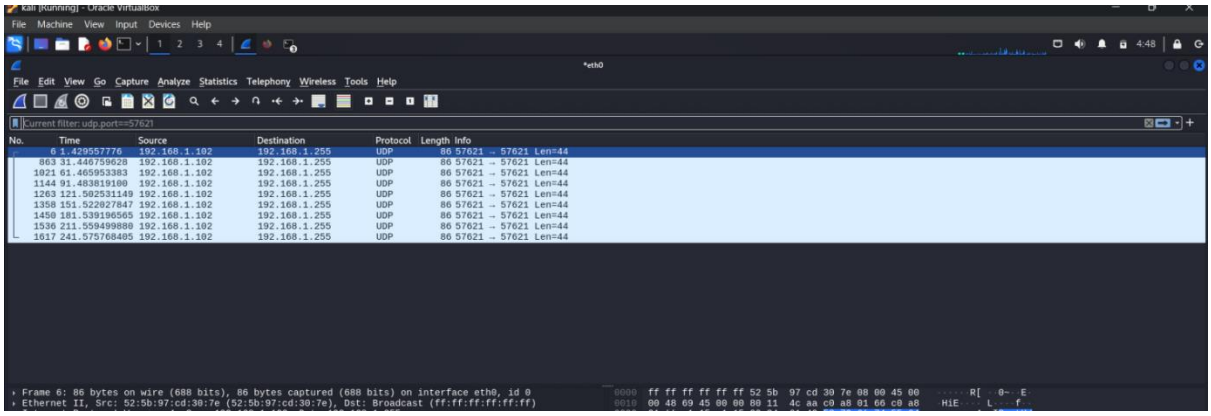
To identify the packets which has 192.168.1.107 has destination I applied the filter **ip.dst==192.168.1.107**



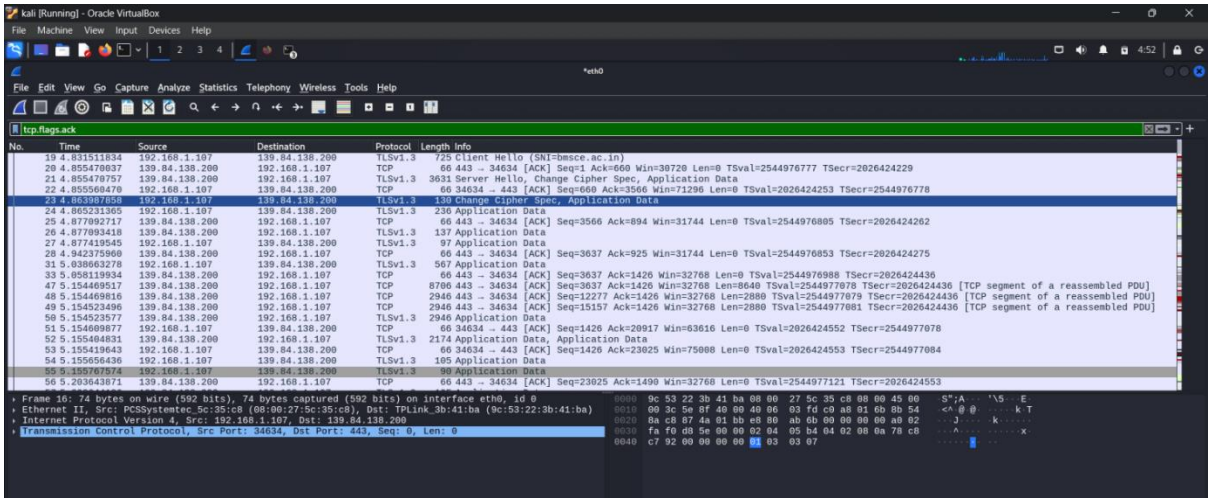
To filter the packets which has udp length less than particular value say 100 i ran the following filter **udp.length<100**



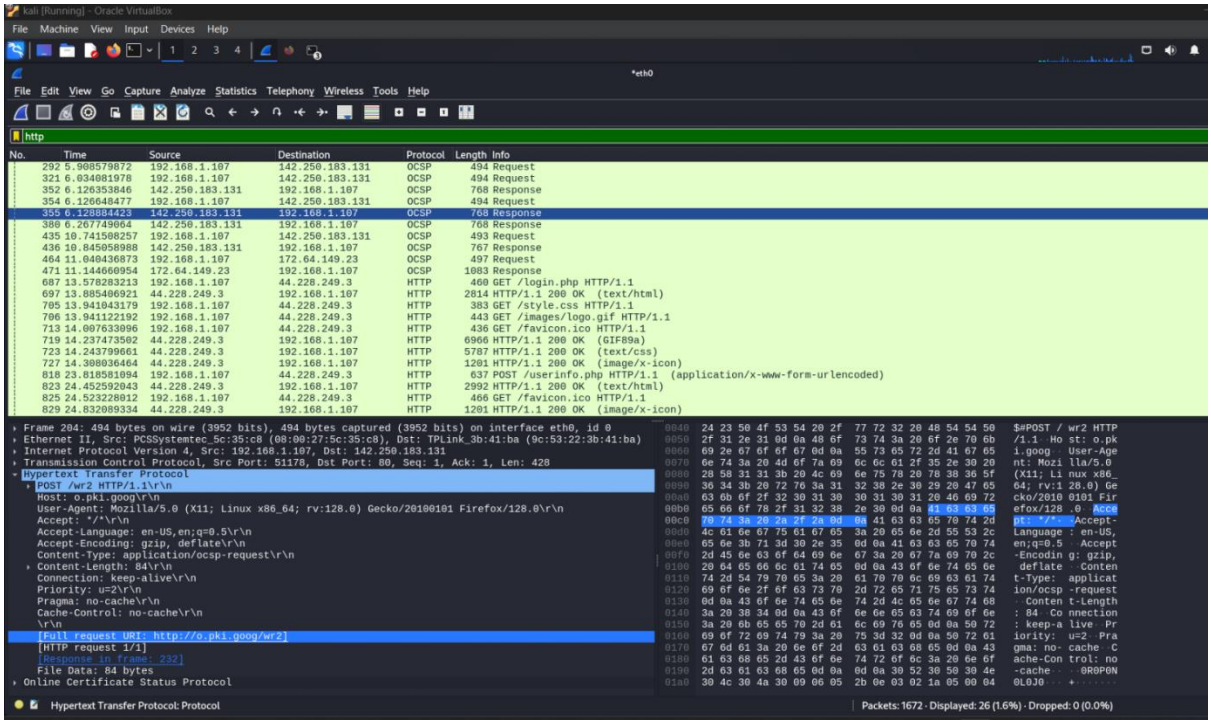
To get udp packet associated with a particular port i ran the following filter **udp.port==<port>**



To find packets with the ack flag set i ran the following filter **tcp.flags.ack**



To filter http packets i ran the following filter **http**

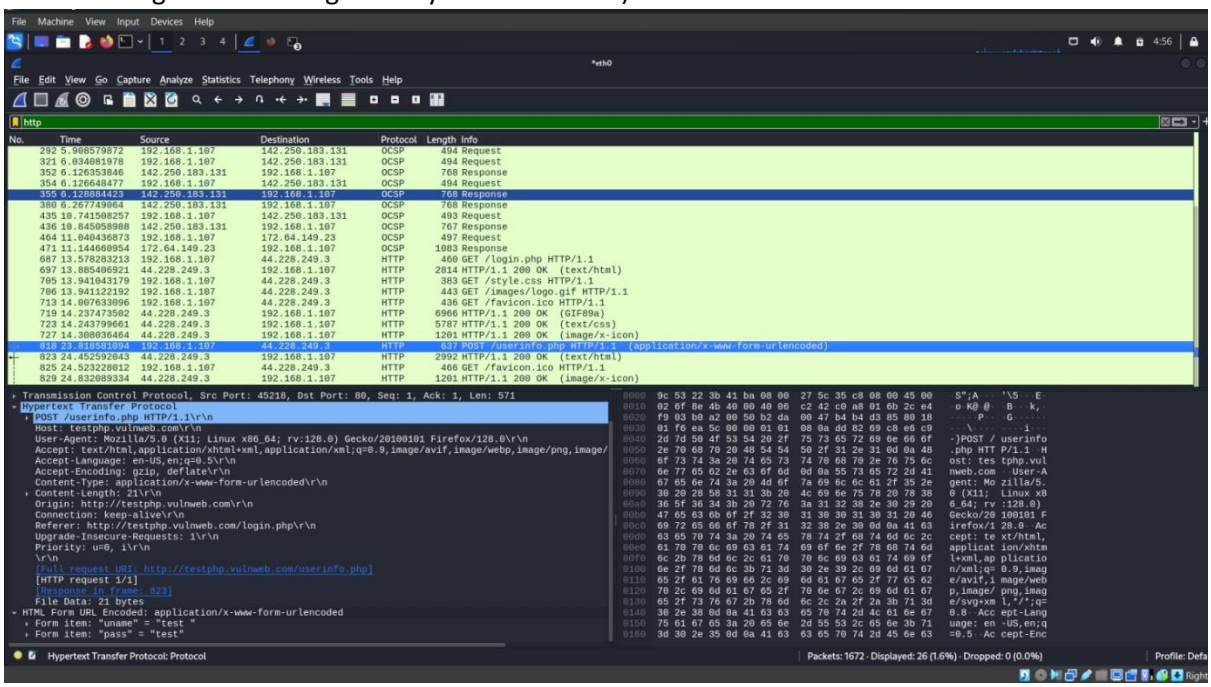


This concludes the filtering section

Packet analysis

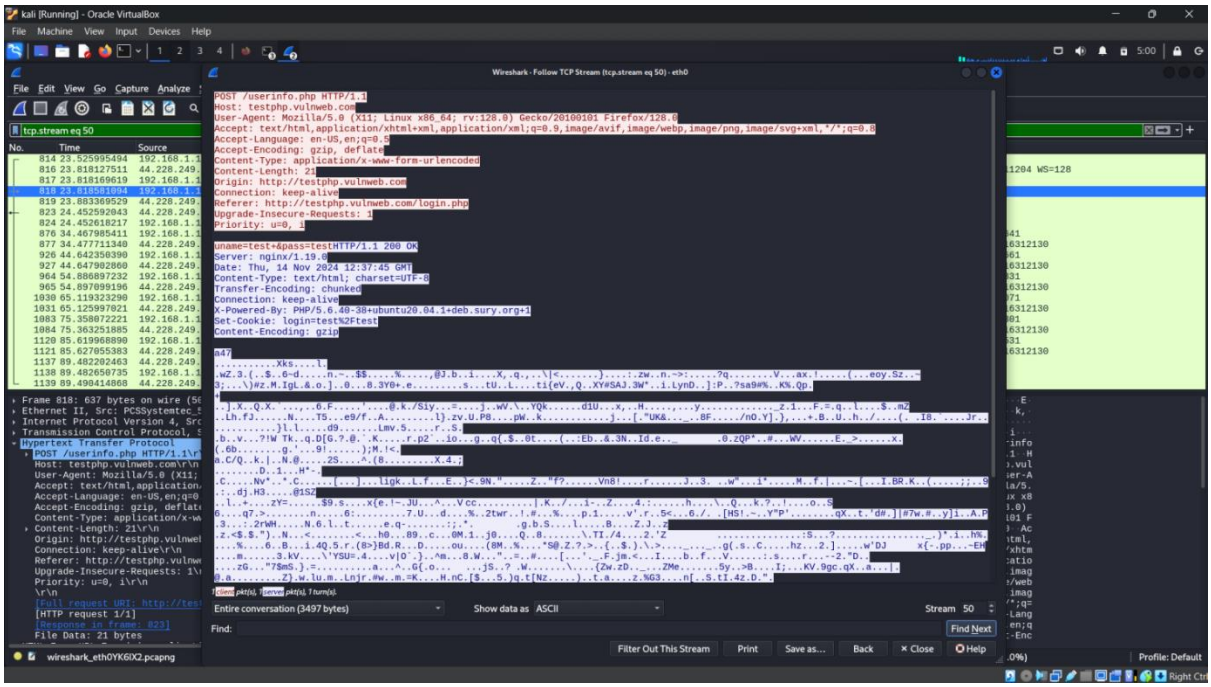
In the final image, I noticed a packet containing a POST request (the 4th one from the end), so I decided to investigate it further. I examined the application layer of the protocol and found that the user was attempting to submit data to the /userinfo.php page. Upon inspecting the details, I discovered that the login parameters were listed as `uname="test"` and `pass="test"`.

(Note: This test was performed on http://testphp.vulnweb.com/login.php, a purposely vulnerable website designed for testing security vulnerabilities.)



I right-clicked on that specific packet and selected the “Follow” option. This revealed two additional choices: Follow TCP Stream and Follow HTTP Stream. I analyzed both of these streams, and here are the outputs I obtained from each.

TCPSTREAM



HTTPSTREAM