Packet Sniffing with Wireshark Report

Introduction

This report outlines the steps and findings of a packet sniffing exercise using Wireshark. The primary objective of this exercise was to utilize Wireshark to capture, filter, and analyze network traffic. The specific tasks performed included creating both capture and display filters, visiting various websites, and eliminating packets associated with a specific website. This report provides a detailed account of each step and the corresponding results.

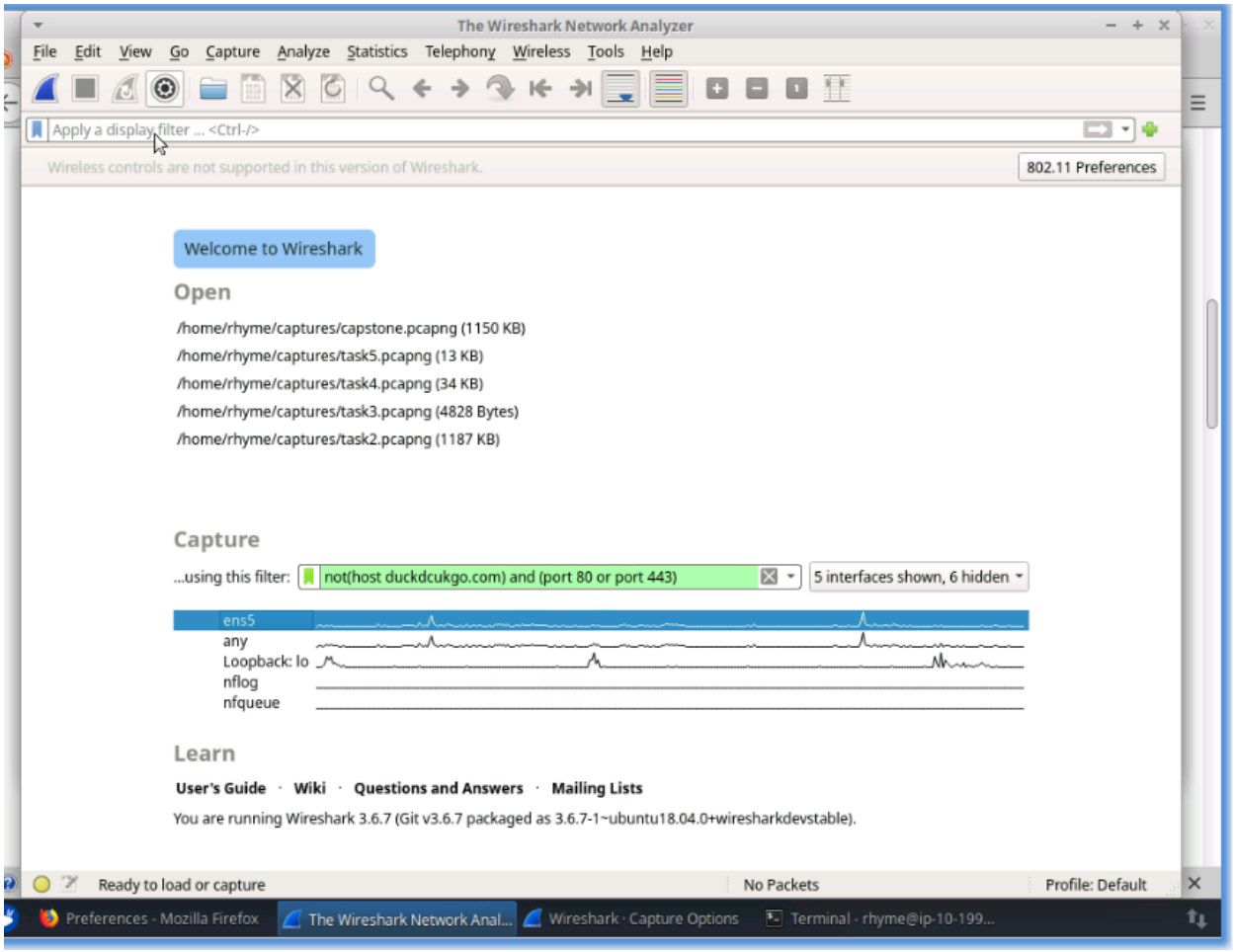
Methodology

Clearing Firefox Cache

Before initiating the packet capture process, the Firefox web browser's cache was cleared to ensure that the browser wouldn't retrieve cached content during subsequent website visits. This step was essential to obtain accurate results.

Packet Capture with Wireshark

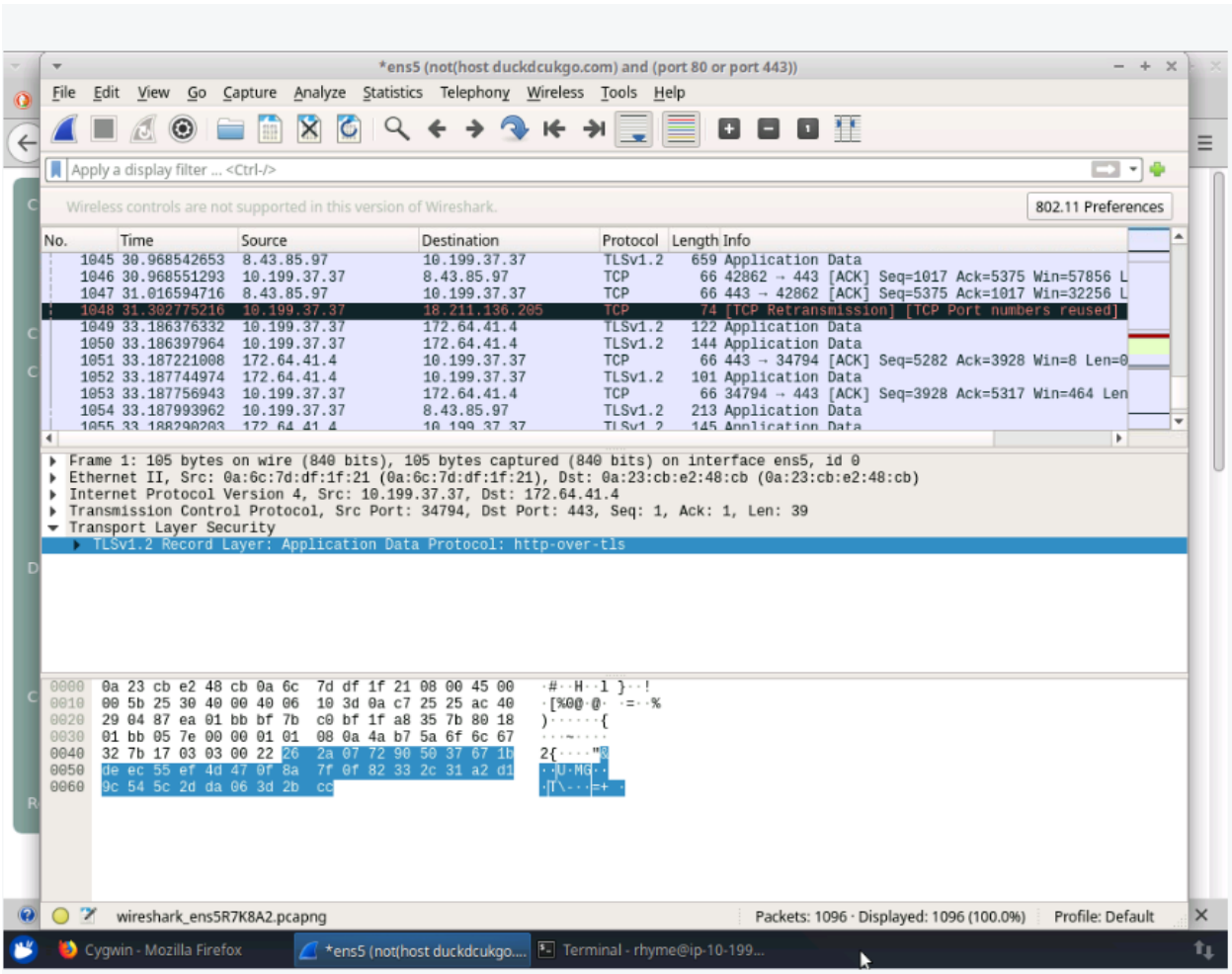
Wireshark was used to capture network traffic on the Ethernet interface. The packet capture session was initiated, capturing all packets passing through the specified interface.



Visiting Websites

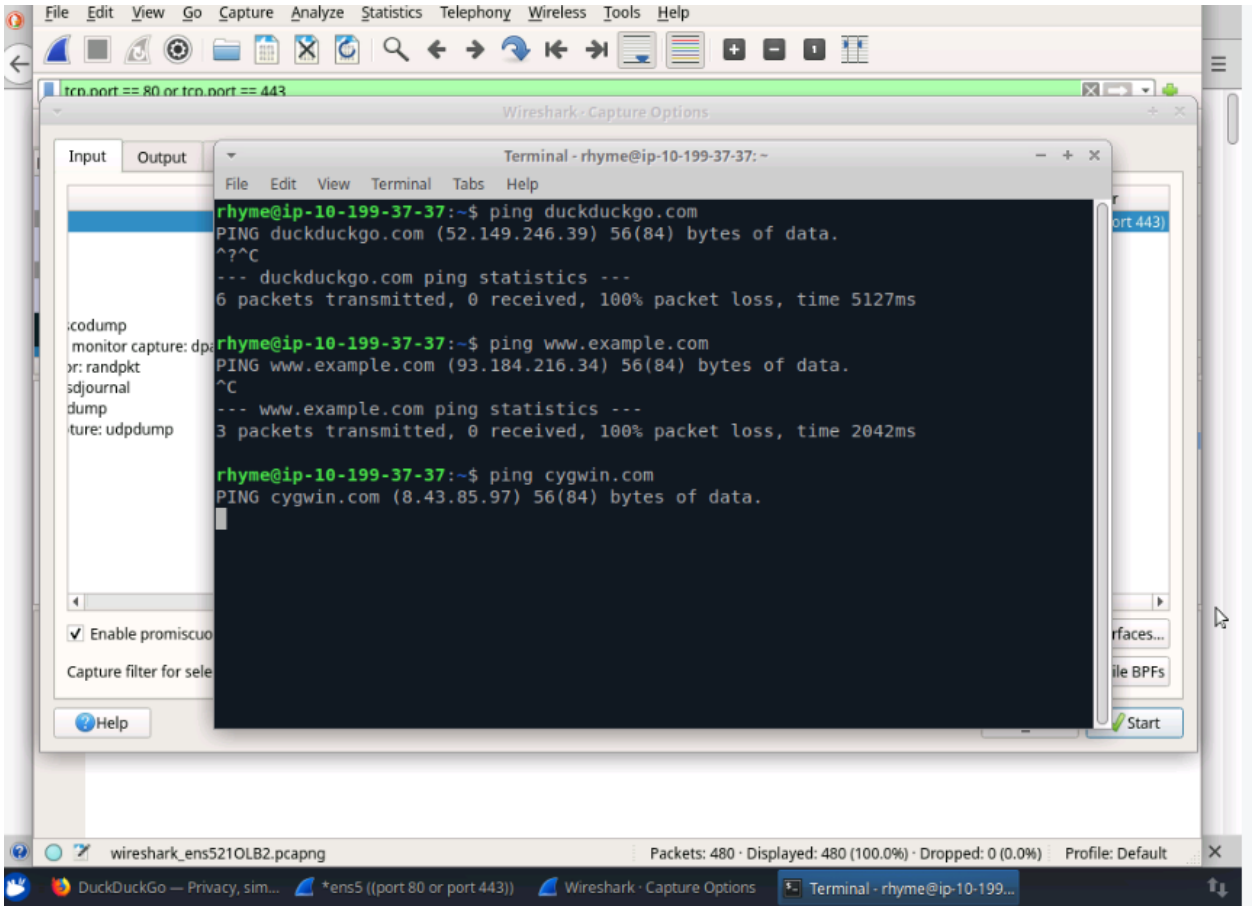
Three websites were visited during the capture session:

- **duckduckgo.com**
- https://example.com
- http://cygwin.com


*ens5 (not(host duckdcukgo.com) and (port 80 or port 443))
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
Wireless controls are not supported in this version of Wireshark.
802.11 Preferences
No. Time Source Destination Protocol Length Info
1045 30.968542653 8.43.85.97 10.199.37.37 TLSv1.2 659 Application Data
1046 30.968551293 10.199.37.37 8.43.85.97 TCP 66 42862 → 443 [ACK] Seq=1017 Ack=5375 Win=57856 L
1047 31.016594716 8.43.85.97 10.199.37.37 TCP 66 443 → 42862 [ACK] Seq=5375 Ack=1017 Win=32256 L
1048 31.302775216 10.199.37.37 18.211.136.205 TCP 74 [TCP Retransmission] [TCP Port numbers reused]
1049 33.186376332 10.199.37.37 172.64.41.4 TLSv1.2 122 Application Data
1050 33.186397964 10.199.37.37 172.64.41.4 TLSv1.2 144 Application Data
1051 33.187221008 172.64.41.4 10.199.37.37 TCP 66 443 → 34794 [ACK] Seq=5282 Ack=3928 Win=8 Len=0
1052 33.187744974 172.64.41.4 10.199.37.37 TLSv1.2 101 Application Data
1053 33.187756943 10.199.37.37 172.64.41.4 TCP 66 34794 → 443 [ACK] Seq=3928 Ack=5317 Win=464 Len
1054 33.187993962 10.199.37.37 8.43.85.97 TLSv1.2 213 Application Data
Frame 1: 105 bytes on wire (840 bits), 105 bytes captured (840 bits) on interface ens5, id 0
Ethernet II, Src: 0a:6c:7d:df:1f:21 (0a:6c:7d:df:1f:21), Dst: 0a:23:cb:e2:48:cb (0a:23:cb:e2:48:cb)
Internet Protocol Version 4, Src: 10.199.37.37, Dst: 172.64.41.4
Transmission Control Protocol, Src Port: 34794, Dst Port: 443, Seq: 1, Ack: 1, Len: 39
Transport Layer Security
TLSv1.2 Record Layer: Application Data Protocol: http-over-tls
0000 0a 23 cb e2 48 cb 0a 6c 7d df 1f 21 08 00 45 00
0010 00 5b 25 30 40 00 40 06 10 3d 0a c7 25 25 ac 40
0020 29 04 87 ea 01 bb bf 7b c0 bf 1f a8 35 7b 80 18
0030 01 bb 05 7e 00 00 01 01 08 0a 4a b7 5a 6f 6c 67
0040 32 7b 17 03 03 00 22 26 2a 07 72 90 50 37 67 1b
0050 de ec 55 ef 4d 47 0f 8a 7f 0f 82 33 2c 31 a2 d1
0060 9c 54 5c 2d da 06 3d 2b cc
wireshark_ens5R7K8A2.pcapng
Packets: 1096 · Displayed: 1096 (100.0%)
Profile: Default
Cygwin - Mozilla Firefox
*ens5 (not(host duckdcukgo....
Terminal - rhyme@ip-10-199...

Obtaining IP Addresses

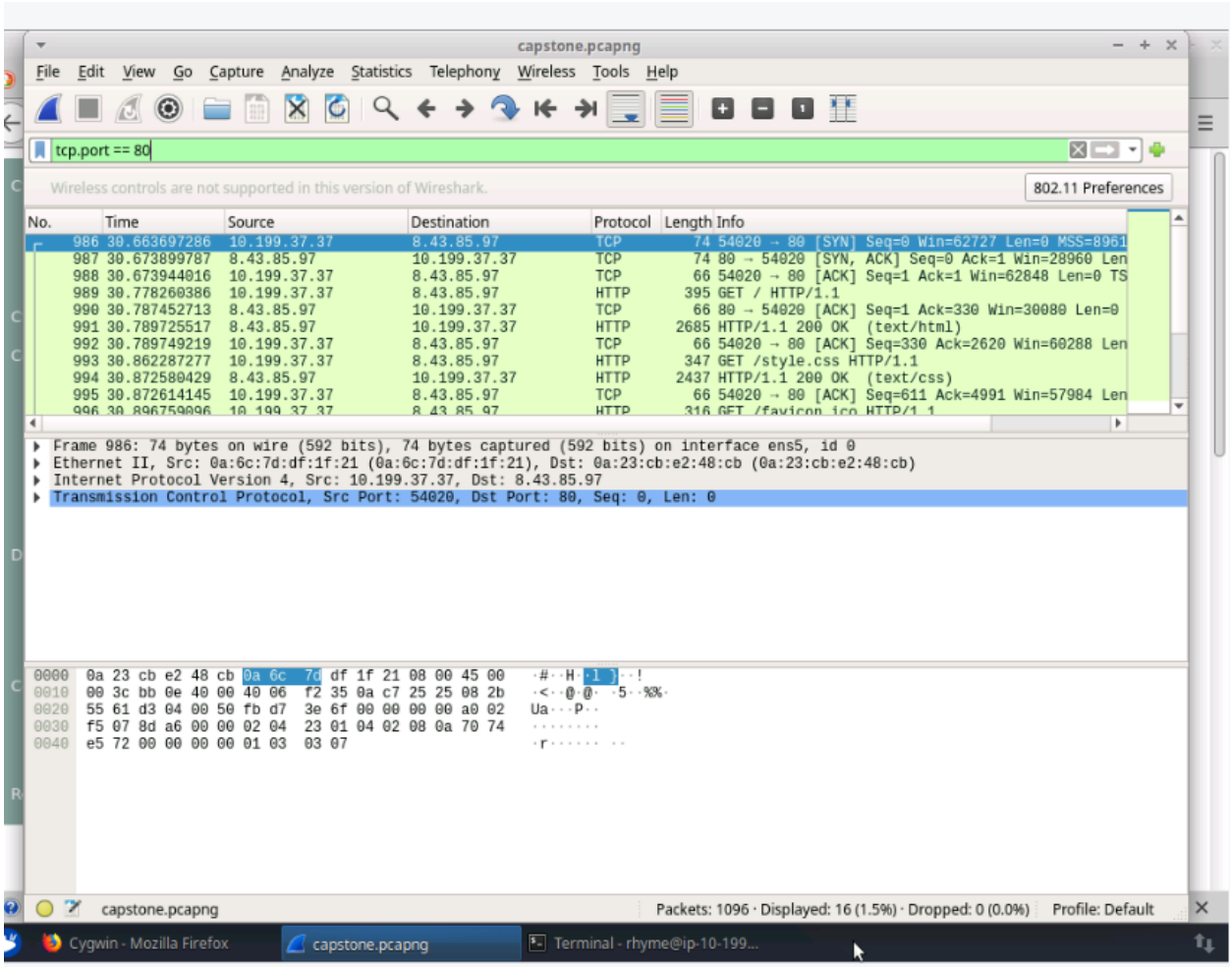
The **ping** command was used to obtain the IP addresses of the three websites, ensuring that we had the accurate IP addresses for subsequent filtering.



Creating Display Filters

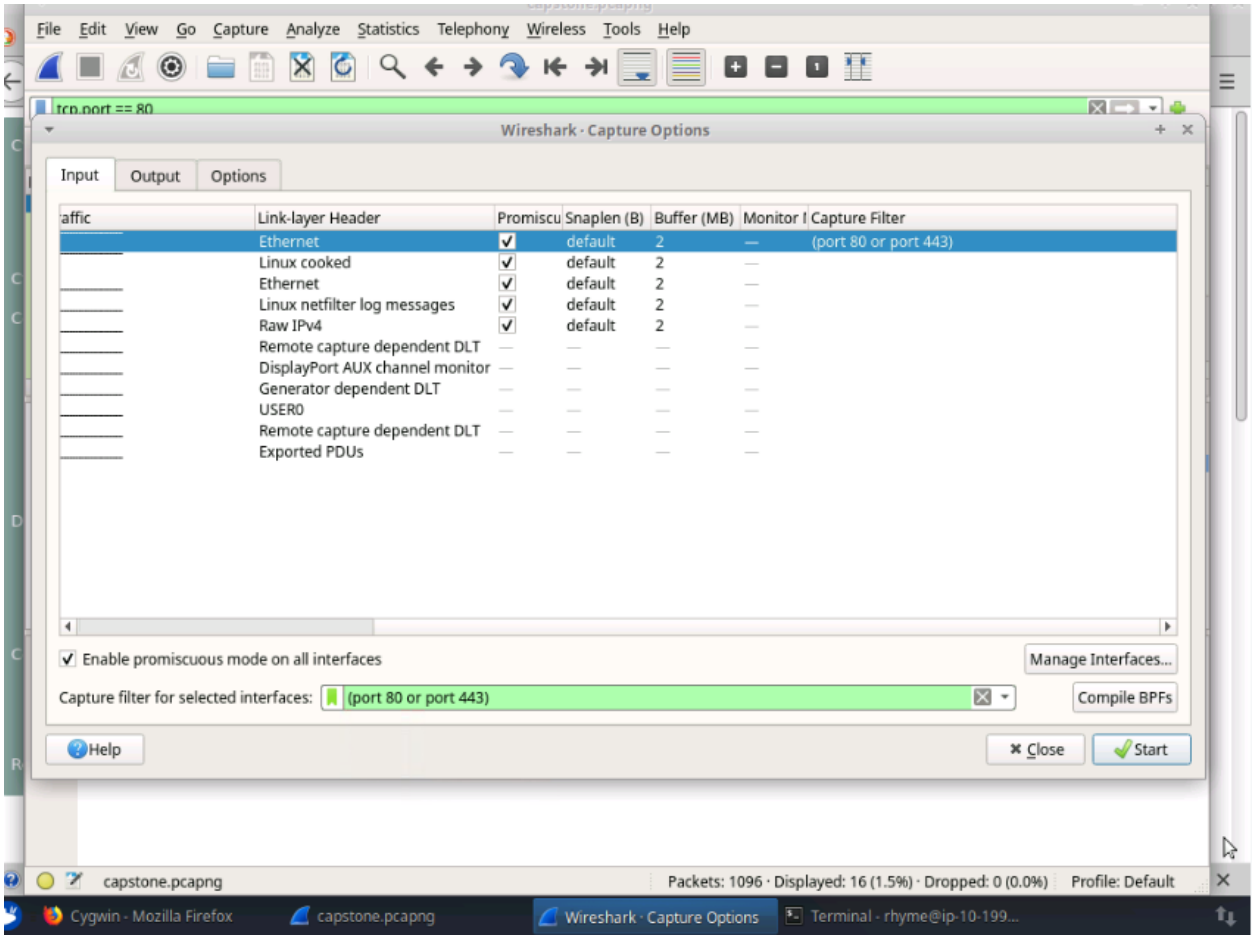
Two display filters were created:

- A filter to display only port 80 TCP data, which is commonly used for HTTP traffic.
- A filter to display only HTTP and HTTPS packets.


capstone.pcapng
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
tcp.port == 80
Wireless controls are not supported in this version of Wireshark.
802.11 Preferences
No. Time Source Destination Protocol Length Info
986 30.663697286 10.199.37.37 8.43.85.97 TCP 74 54020 → 80 [SYN] Seq=0 Win=62727 Len=0 MSS=8961
987 30.673899787 8.43.85.97 10.199.37.37 TCP 74 80 → 54020 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len
988 30.673944016 10.199.37.37 8.43.85.97 TCP 66 54020 → 80 [ACK] Seq=1 Ack=1 Win=62848 Len=0 TS
989 30.778260386 10.199.37.37 8.43.85.97 HTTP 395 GET / HTTP/1.1
990 30.787452713 8.43.85.97 10.199.37.37 TCP 66 80 → 54020 [ACK] Seq=1 Ack=330 Win=30080 Len=0
991 30.789725517 8.43.85.97 10.199.37.37 HTTP 2685 HTTP/1.1 200 OK (text/html)
992 30.789749219 10.199.37.37 8.43.85.97 TCP 66 54020 → 80 [ACK] Seq=330 Ack=2620 Win=60288 Len
993 30.862287277 10.199.37.37 8.43.85.97 HTTP 347 GET /style.css HTTP/1.1
994 30.872580429 8.43.85.97 10.199.37.37 HTTP 2437 HTTP/1.1 200 OK (text/css)
995 30.872614145 10.199.37.37 8.43.85.97 TCP 66 54020 → 80 [ACK] Seq=611 Ack=4991 Win=57984 Len
Frame 986: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface ens5, id 0
Ethernet II, Src: 0a:6c:7d:df:1f:21 (0a:6c:7d:df:1f:21), Dst: 0a:23:cb:e2:48:cb (0a:23:cb:e2:48:cb)
Internet Protocol Version 4, Src: 10.199.37.37, Dst: 8.43.85.97
Transmission Control Protocol, Src Port: 54020, Dst Port: 80, Seq: 0, Len: 0
0000 0a 23 cb e2 48 cb 0a 6c 7d df 1f 21 08 00 45 00
0010 00 3c bb 0e 40 00 40 06 f2 35 0a c7 25 25 08 2b
0020 55 61 d3 04 00 50 fb d7 3e 6f 00 00 00 00 a0 02
0030 f5 07 8d a6 00 00 02 04 23 01 04 02 08 0a 70 74
0040 e5 72 00 00 00 00 01 03 03 07
capstone.pcapng
Packets: 1096 · Displayed: 16 (1.5%) · Dropped: 0 (0.0%)
Profile: Default
Cygwin - Mozilla Firefox
capstone.pcapng
Terminal - rhyme@ip-10-199...

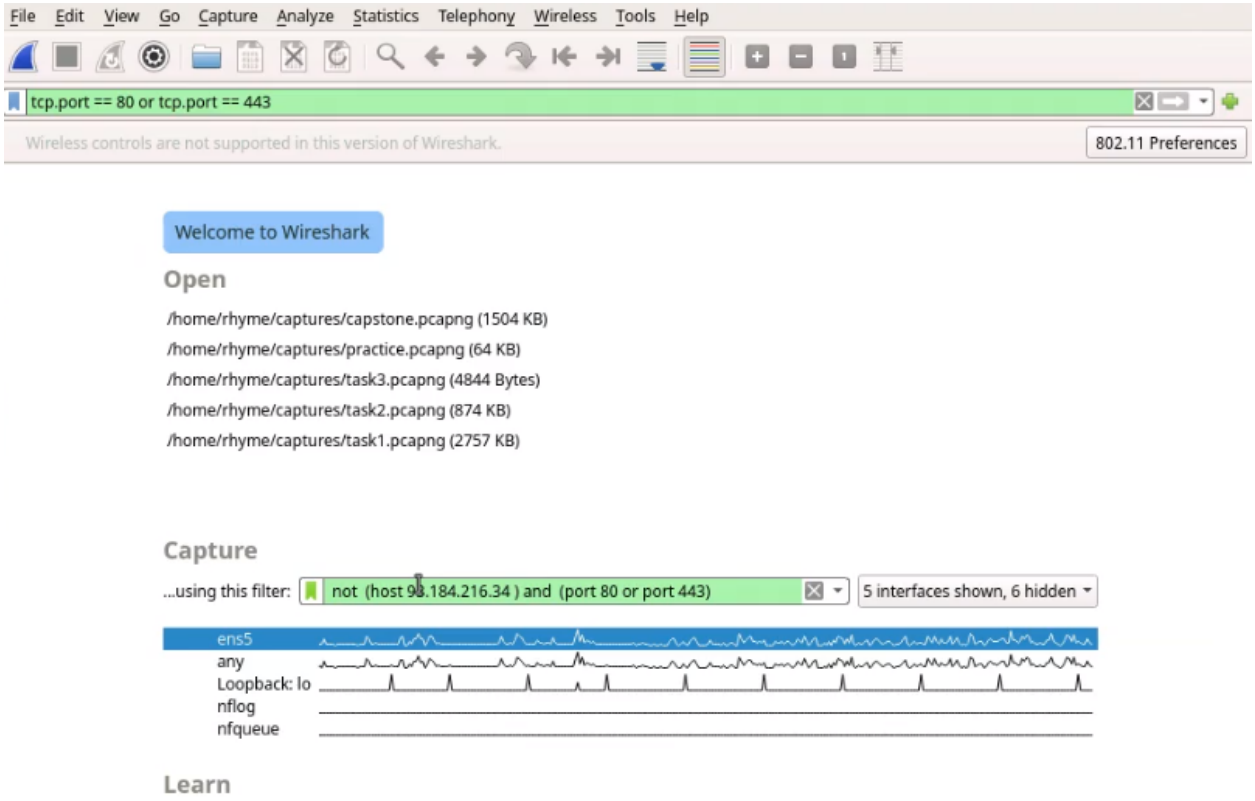
Creating Capture Filter

A capture filter was created to capture only HTTP and HTTPS traffic. This filter was designed to focus on the specific types of packets relevant to the exercise.



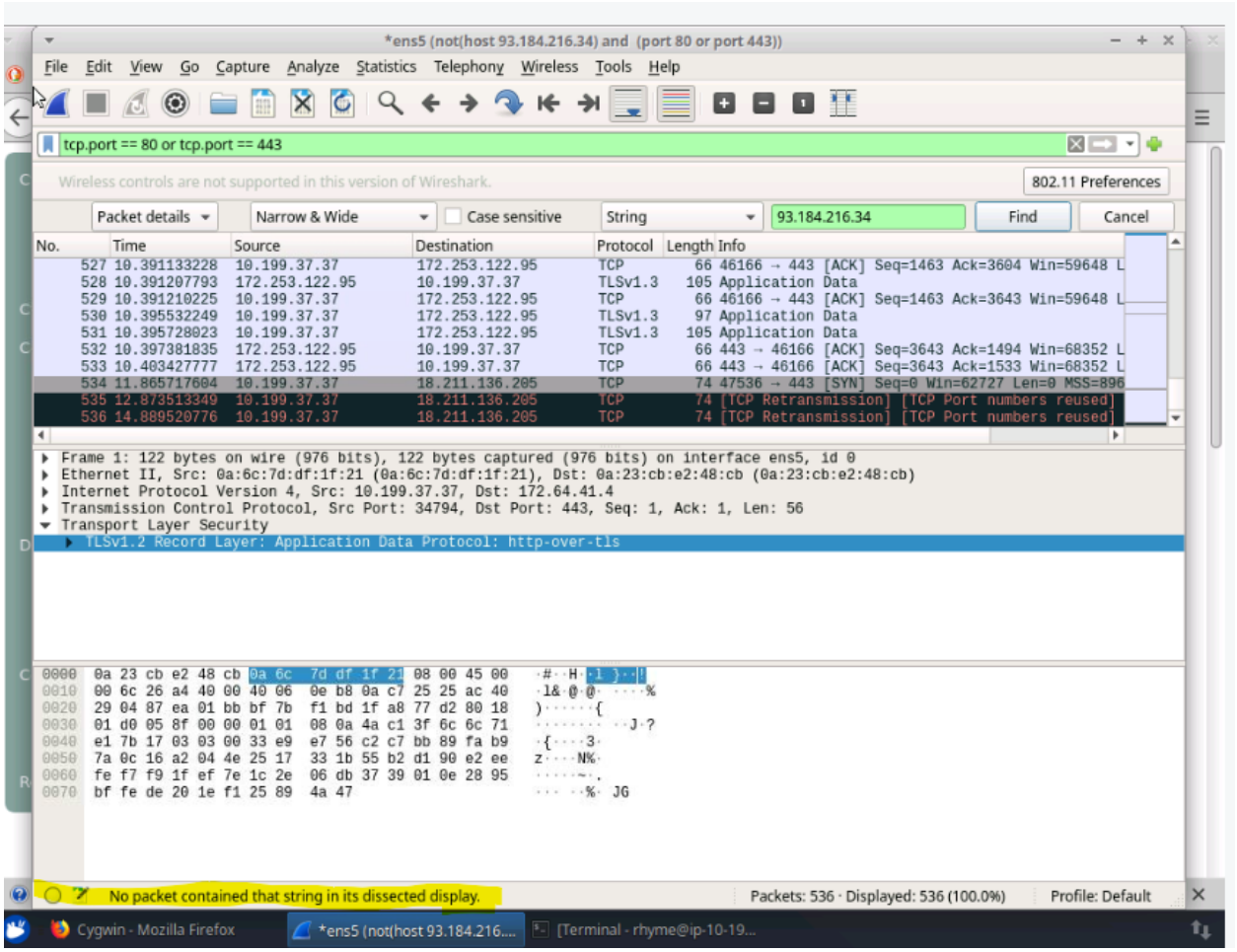
Eliminating example.com Traffic

The packets associated with visits to **example.com** were eliminated from the capture using the capture filter. This step involved filtering out packets with the IP address of **example.com**.



Verification

The final step involved verifying that **example.com**'s IP address was no longer present in the captured packets. This was done using Wireshark's search tool to confirm the successful elimination of the specified website's traffic.



Results

The following are the key findings and observations from the packet sniffing exercise:

1. After creating the display filters for port 80 TCP data and HTTP/HTTPS packets, it was evident that the capture contained a wide range of network traffic, including various protocols and services.
2. The capture filter successfully focused on capturing only HTTP and HTTPS traffic, effectively narrowing down the packets to the relevant web traffic.
3. After eliminating packets associated with visits to **example.com**, the search tool confirmed that **example.com**'s IP address was no longer present in the captured packets, demonstrating the successful implementation of the capture filter.

Conclusion

This packet sniffing exercise using Wireshark allowed us to gain valuable insights into network traffic filtering and analysis. By creating and applying capture and display filters, we were able to selectively capture and examine HTTP and HTTPS traffic while eliminating specific website traffic.

These skills are crucial for network administrators and security professionals when monitoring and troubleshooting network activity, ensuring that only relevant data is captured and analyzed.

Overall, this exercise provided a practical and hands-on experience in using Wireshark for network packet analysis and filtering, enhancing our understanding of network traffic management and analysis techniques.