

Lab Week 2

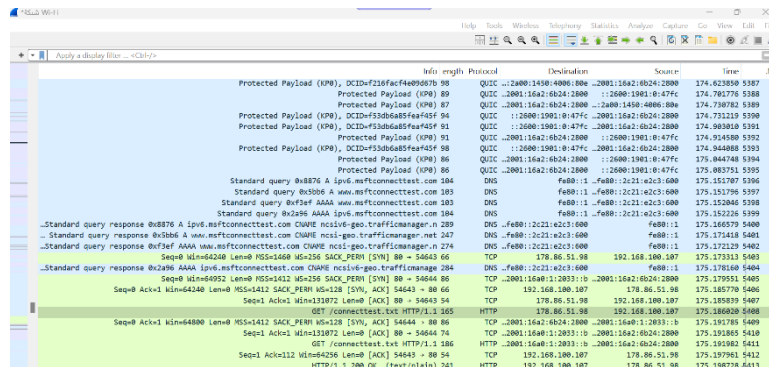
The Internet Protocols

Shahad Nasser Alwatban

Part 1: Capturing HTTP Traffic.

Task 1: Start Wireshark and capture packets.

"I opened Wireshark and Selected the network interface connected to the internet (Wi Fi) and then I Capturing Packets and then Open web browser and navigate to (<https://qu.edu.sa>) website after the website has fully loaded, stop capturing packets "

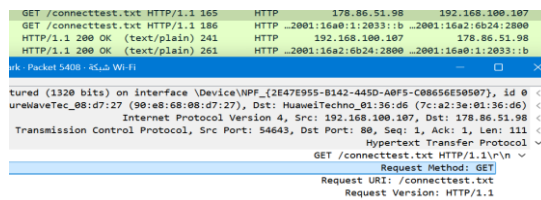


Task 2: Filter HTTP packets and analyze them.

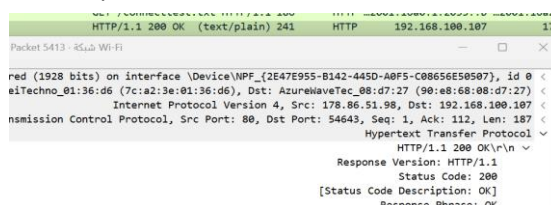
In filter bar, I filter only HTTP packets from the capture, and I found HTTP request messages (GET) also another one with response codes (200 OK):

Filter	Info	Length	Protocol	Destination	Source	Time	No.
http	GET /connecttest.txt HTTP/1.1 165	178	HTTP	178.86.51.98	192.168.100.107	175.186020	5408
	GET /connecttest.txt HTTP/1.1 186	178	HTTP	192.168.100.107	178.86.51.98	175.191982	5411
	HTTP/1.1 200 OK (text/plain) 241	192	HTTP	192.168.100.107	178.86.51.98	175.198728	5413
	HTTP/1.1 200 OK (text/plain) 261	192	HTTP	192.168.100.107	178.86.51.98	175.197961	5412

Here is HTTP request with GET method.



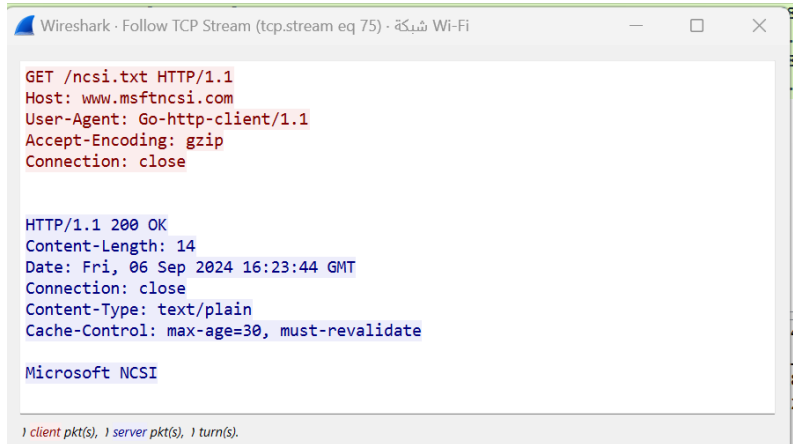
Here is HTTP response with response 200 ok.



Part 2: Analyzing TCP/IP Traffic.

Task 1: Filter TCP packets

Here I Select a TCP packet related to HTTP and select "Follow" -> "TCP Stream", This shows the entire conversation between the client and server.



```
Wireshark · Follow TCP Stream (tcp.stream eq 75) · شبكة Wi-Fi

GET /ncsi.txt HTTP/1.1
Host: www.msftncsi.com
User-Agent: Go-http-client/1.1
Accept-Encoding: gzip
Connection: close

HTTP/1.1 200 OK
Content-Length: 14
Date: Fri, 06 Sep 2024 16:23:44 GMT
Connection: close
Content-Type: text/plain
Cache-Control: max-age=30, must-revalidate

Microsoft NCSI

1 client pkt(s), 1 server pkt(s), 1 turn(s).
```

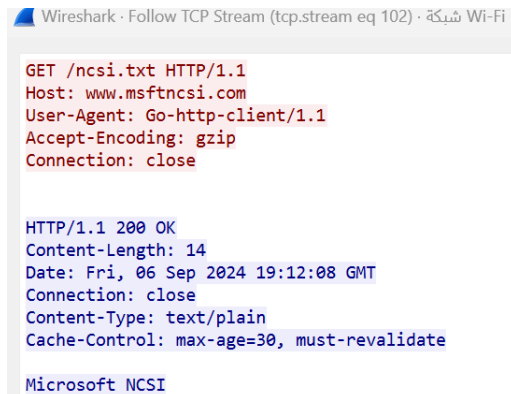
Task 2: Analyze TCP handshake and investigate Data Transfer and Termination

I Find packets related to the TCP three-way handshake (SYN: Initiates a connection, SYN-ACK: Acknowledges and responds to the SYN, ACK: Acknowledges the SYN-ACK and establishes the connection.)

Here sequence and acknowledgment numbers.

Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM [SYN] 80 → 55622 66	TCP	2.18.31.89	192.168.100.107	112.066103 3716
Seq=0 Ack=1 Win=64240 Len=0 MSS=1412 SACK_PERM WS=128 [SYN, ACK] 55622 → 80 66	TCP	192.168.100.107	2.18.31.89	112.141839 3718
Seq=1 Ack=1 Win=131072 Len=0 [ACK] 80 → 55622 54	TCP	2.18.31.89	192.168.100.107	112.141971 3719

I use the "Follow TCP Stream" option to view the entire conversation between the client and server. This feature will show the data exchanged in a more readable format.



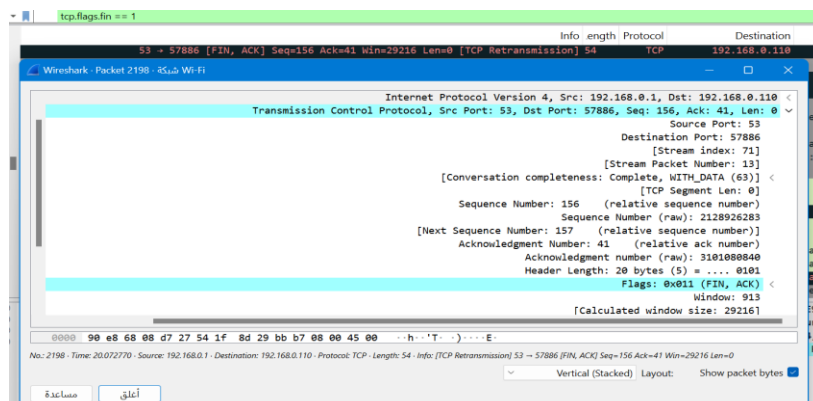
```
Wireshark · Follow TCP Stream (tcp.stream eq 102) · شبكة Wi-Fi

GET /ncsi.txt HTTP/1.1
Host: www.msftncsi.com
User-Agent: Go-http-client/1.1
Accept-Encoding: gzip
Connection: close

HTTP/1.1 200 OK
Content-Length: 14
Date: Fri, 06 Sep 2024 19:12:08 GMT
Connection: close
Content-Type: text/plain
Cache-Control: max-age=30, must-revalidate

Microsoft NCSI
```

I looked at the TCP termination process (FIN, ACK packets) by using " tcp.flags.fin == 1" in the filter.



```
tcp.flags.fin == 1

53 → 57886 [FIN, ACK] Seq=156 Ack=41 Win=29216 Len=0 [TCP Retransmission] 54
TCP 192.168.0.110 → 192.168.0.110
Source Port: 53
Destination Port: 57886
[Stream index: 71]
[Stream Packet Number: 13]
[Conversation completeness: Complete, WITH DATA (63)]
[TCP Segment Len: 0]
Sequence Number: 156 (relative sequence number)
Sequence Number (raw): 2128926283
[Next Sequence Number: 157 (relative sequence number)]
Acknowledgment Number: 41 (relative ack number)
Acknowledgment number (raw): 3181080840
Header Length: 20 bytes (5) = .... 0101
Flags: 0x011 (FIN, ACK)
Window: 913
[Calculated window size: 292161]

0000 90 e8 68 08 d7 27 54 1f 8d 29 bb b7 08 00 45 00 ...h..T..}...E-
No: 2198 - Time: 20.072770 - Source: 192.168.0.1 - Destination: 192.168.0.110 - Protocol: TCP - Length: 54 - Info: [TCP Retransmission] 53 → 57886 [FIN, ACK] Seq=156 Ack=41 Win=29216 Len=0
Vertical (Stacked) Layout: Show packet bytes
```

Part 3: Capturing and Analyzing UDP Traffic

Task 1: Generate UDP traffic and capture packets

In the filter bar, I type UDP

Len=1250	53817 → 443	1292	UDP	192.168.0.110	86.51.76.204	2.966486	191
Len=1250	53817 → 443	1292	UDP	192.168.0.110	86.51.76.204	2.966652	192
Len=33	443 → 53817	75	UDP	86.51.76.204	192.168.0.110	2.966726	193
Len=1250	53817 → 443	1292	UDP	192.168.0.110	86.51.76.204	2.966840	194
Len=1250	53817 → 443	1292	UDP	192.168.0.110	86.51.76.204	2.967044	195
Len=1250	53817 → 443	1292	UDP	192.168.0.110	86.51.76.204	2.967304	196
Len=1250	53817 → 443	1292	UDP	192.168.0.110	86.51.76.204	2.967488	197
Len=1250	53817 → 443	1292	UDP	192.168.0.110	86.51.76.204	2.967661	198
Len=1250	53817 → 443	1292	UDP	192.168.0.110	86.51.76.204	2.967857	199
Len=1250	53817 → 443	1292	UDP	192.168.0.110	86.51.76.204	2.968072	200

Observe the source and destination ports, length, and data.

User Datagram Protocol, Src Port: 443, Dst Port: 53817	
Source Port: 443	
Destination Port: 53817	
Length: 1258	
Checksum: 0xfcd8 [unverified]	
[Checksum Status: Unverified]	
[Stream index: 1]	
[Stream Packet Number: 183]	
[Timestamps]	<
UDP payload (1250 bytes)	
Data (1250 bytes)	
4df2c857d8c2b348616f48ae17bccc8e1d4606194cc45625e411f36f4037463e04a19b0ac06e34c382a2ebcb396fe0209b5c4d754bf9beefd103a70b46b11f9011e1	
[Length: 1250]	

Compare the simplicity of UDP headers with TCP headers:

The UDP header is much simpler, consisting of only 4 fields while TCP header is significantly more complex because TCP offers features like connection establishment, reliability, flow control, and congestion control. The TCP header consists of 10 mandatory fields

Part 4: Comparing TCP and UDP by filling in the following tables. Save your work (e.g., in an MS Word document), and upload it to your online git repo.

Task 1: Fill in the following table and provide reasons.

	TCP or UDP	Reasons
Reliability and Connection Establishment	TCP	TCP uses acknowledgments (ACKs), Error Detection and Correction and Sequencing, unlike UDP
Data Integrity and Ordering	TCP	Each TCP segment includes a checksum for error detection, Guaranteed Ordering, Acknowledgments and Retransmissions, Error Detection and Correction unlike UDP

Task 2: Identify the use Cases and Performance of TCP and UDP.

	TCP	UDP
Use case	<ul style="list-style-type: none">Web browsingEmailFile Transfer	<ul style="list-style-type: none">Live videoOnline gamingAudio streaming
Performance	Reliability, Throughput, Latency, Overhead, Connection-Oriented Nature, Data Ordering and Performance Optimization Techniques	Optional checksum enables error detection, while minimal overhead, lack of connection setup, and small headers ensure high throughput, low latency, and reduced overhead.