

# WIRESHARK

## Computer Networks Lab Assignment

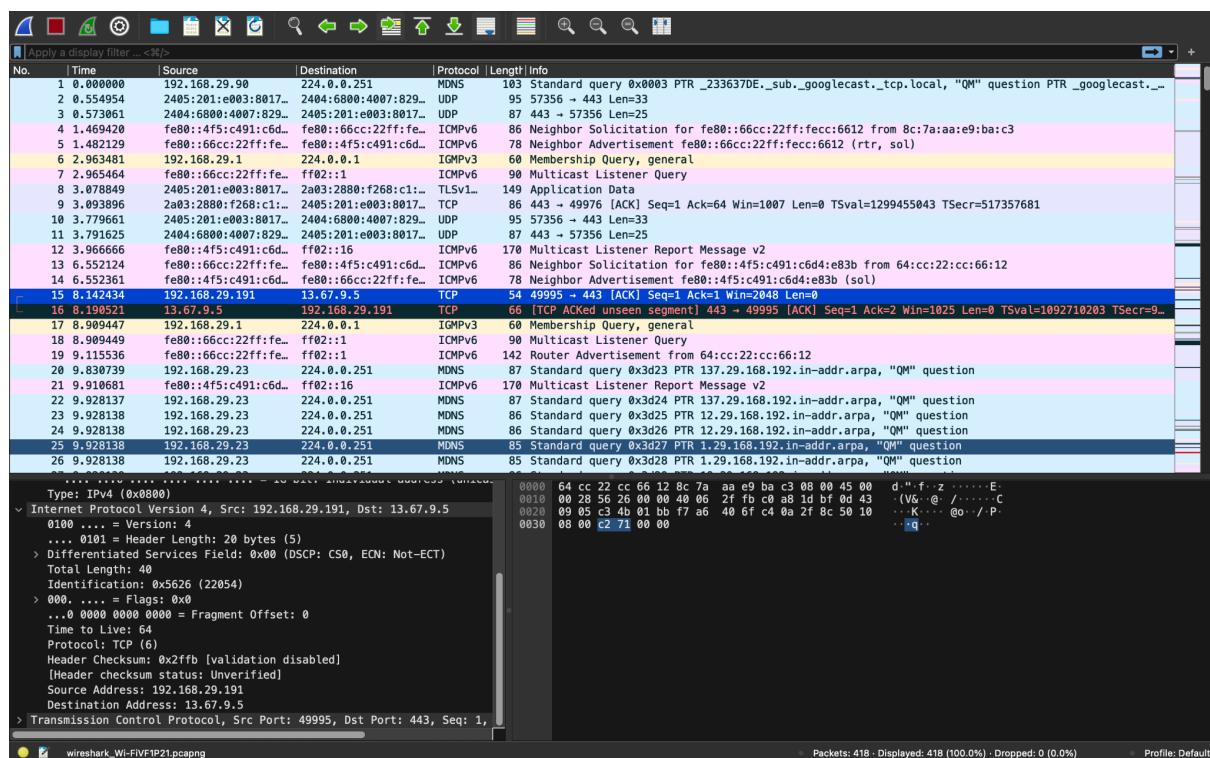
21BCE1796 - B SHAKTHI

Step 1:

Open Wireshark and select the interface

Step 2:

Start capturing by clicking the start in Wireshark



## Step 3:

## Exploring the IP Protocol in a UDP or TCP Packet

The image shows a Wireshark network traffic capture. The packet list on the left displays 26 packets, including DNS queries, ICMPv6 messages, and TCP segments. The packet details pane on the right shows the structure of a selected packet, including Ethernet II, Internet Protocol Version 4, and Transmission Control Protocol.

**Packet List:**

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.29.90	224.0.0.251	MDNS	103	Standard query 0x0003 PTR _233637DE._sub._googlecast._tcp.local, "QM" question PTR _googlecast._
2	0.554954	2405:201:e003:8017...	2404:6800:4007:829...	UDP	95	57356 → 443 Len=33
3	0.573061	2404:6800:4007:829...	2405:201:e003:8017...	UDP	87	443 → 57356 Len=25
4	1.469420	fe80::4f5:c491:c6d...	fe80::66cc:22ff:fe...	ICMPv6	86	Neighbor Solicitation for fe80::66cc:22ff:fecc:6612 from 8c:7a:aa:e9:ba:c3
5	1.482129	fe80::66cc:22ff:fe...	fe80::4f5:c491:c6d...	ICMPv6	78	Neighbor Advertisement fe80::66cc:22ff:fecc:6612 (rtr, sol)
6	2.963481	192.168.29.1	224.0.0.1	IGMPv3	60	Membership Query, general
7	2.965464	fe80::66cc:22ff:fe...	ff02::1	ICMPv6	90	Multicast Listener Query
8	3.078849	2405:201:e003:8017...	2a03:2880:f268:c1...	TLsv1...	149	Application Data
9	3.093896	2a03:2880:f268:c1...	2405:201:e003:8017...	TCP	86	443 → 49976 [ACK] Seq=1 Ack=64 Win=1007 Len=0 TSval=1299455043 TSecr=517357681
10	3.779661	2405:201:e003:8017...	2404:6800:4007:829...	UDP	95	57356 → 443 Len=33
11	3.791625	2404:6800:4007:829...	2405:201:e003:8017...	UDP	87	443 → 57356 Len=25
12	3.966666	fe80::4f5:c491:c6d...	ff02::16	ICMPv6	170	Multicast Listener Report Message v2
13	6.552124	fe80::66cc:22ff:fe...	fe80::4f5:c491:c6d...	ICMPv6	86	Neighbor Solicitation for fe80::4f5:c491:c6d4:e83b from 64:cc:22:cc:66:12
14	6.552361	fe80::4f5:c491:c6d...	fe80::66cc:22ff:fe...	ICMPv6	78	Neighbor Advertisement fe80::4f5:c491:c6d4:e83b (sol)
15	8.142434	192.168.29.191	13.67.9.5	TCP	54	49995 → 443 [ACK] Seq=1 Ack=1 Win=2048 Len=0
16	8.190521	13.67.9.5	192.168.29.191	TCP	86	[TCP ACKED unseen segment] 443 → 49995 [ACK] Seq=1 Ack=2 Win=1025 Len=0 TSval=1092710203 TSecr=9...
17	8.989447	192.168.29.1	224.0.0.1	IGMPv3	60	Membership Query, general
18	8.989449	fe80::66cc:22ff:fe...	ff02::1	ICMPv6	90	Multicast Listener Query
19	9.115536	fe80::66cc:22ff:fe...	ff02::1	ICMPv6	142	Router Advertisement from 64:cc:22:cc:66:12
20	9.830739	192.168.29.23	224.0.0.251	MDNS	87	Standard query 0x3d23 PTR 137.29.168.192.in-addr.arpa, "QM" question
21	9.910681	fe80::4f5:c491:c6d...	ff02::16	ICMPv6	170	Multicast Listener Report Message v2
22	9.928137	192.168.29.23	224.0.0.251	MDNS	87	Standard query 0x3d24 PTR 137.29.168.192.in-addr.arpa, "QM" question
23	9.928138	192.168.29.23	224.0.0.251	MDNS	86	Standard query 0x3d25 PTR 12.29.168.192.in-addr.arpa, "QM" question
24	9.928138	192.168.29.23	224.0.0.251	MDNS	86	Standard query 0x3d26 PTR 12.29.168.192.in-addr.arpa, "QM" question
25	9.928138	192.168.29.23	224.0.0.251	MDNS	85	Standard query 0x3d27 PTR 1.29.168.192.in-addr.arpa, "QM" question
26	9.928138	192.168.29.23	224.0.0.251	MDNS	85	Standard query 0x3d28 PTR 1.29.168.192.in-addr.arpa, "QM" question

**Packet Details:**

Type: IPv4 (0x0800)

Internet Protocol Version 4, Src: 192.168.29.191, Dst: 13.67.9.5

0100 .... = Version: 4

.... 0101 = Header Length: 20 bytes (5)

> Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)

Total Length: 40

Identification: 0x5626 (22054)

> 000. .... = Flags: 0x0

...0 0000 0000 0000 = Fragment Offset: 0

Time to Live: 64

Protocol: TCP (6)

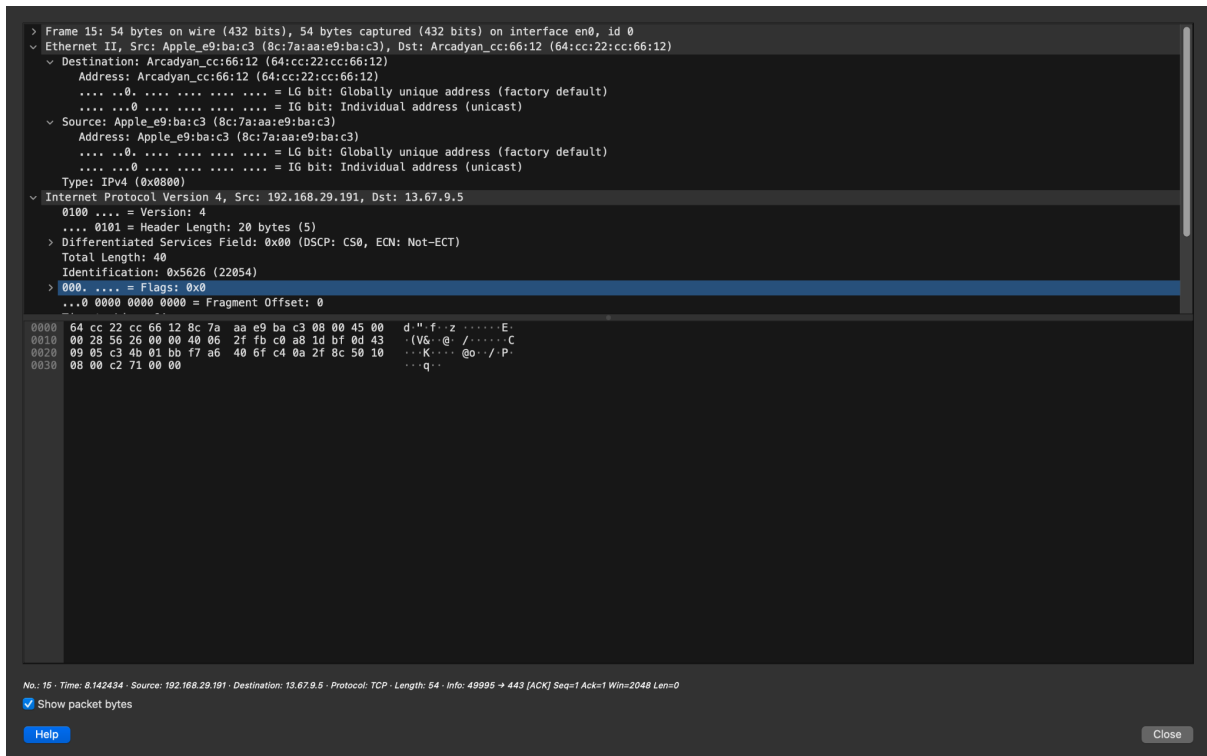
Header Checksum: 0x2ffb [validation disabled]

[Header checksum status: Unverified]

Source Address: 192.168.29.191

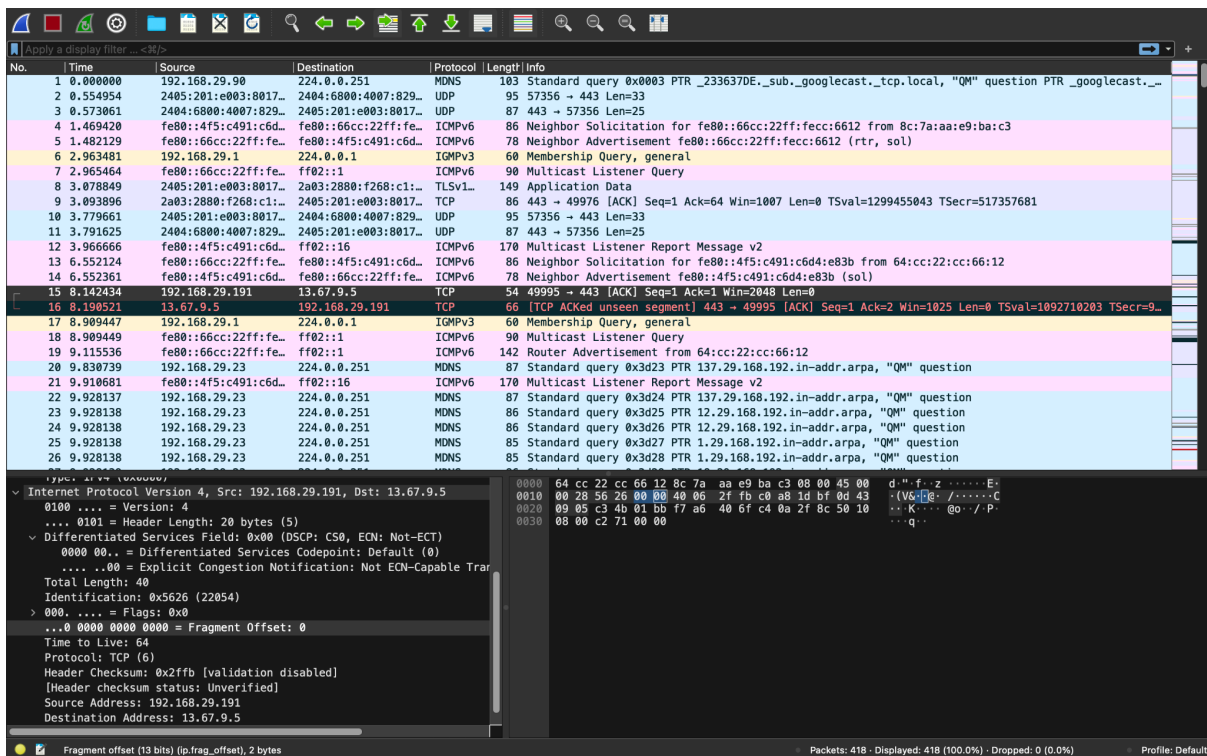
Destination Address: 13.67.9.5

> Transmission Control Protocol, Src Port: 49995, Dst Port: 443, Seq: 1,



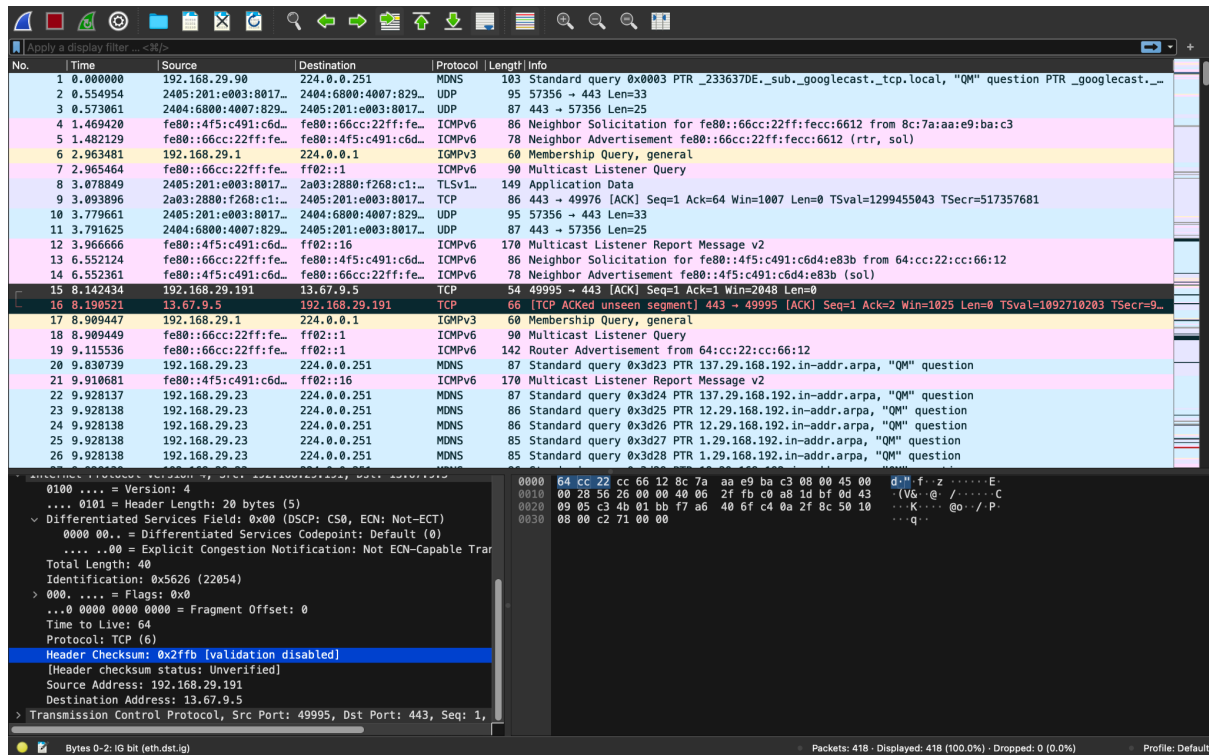
## STEP 4:

## Exploring the binary bytes of IP Header



## Step 5:

## Catching the header checksum from the binary byte



## Step 6:

## Then verify the checksum as follows

4500 0028 5626 0000 4006 **2ffb** c0a8 1dbf 0d43 0905

**2ffb** is the checksum

4500-> 0100010100000000

0028-> 00000000000101000

**1<sup>st</sup> result ->0100010100101000**

5626-> 101011000100110 (first result + next hex number)

**2<sup>nd</sup> result->01001101101001110**

0000-> 0000000000000000(second result + next hex number)

**3<sup>rd</sup> result->01001101101001110**

4006->0100000000000110(third + next hex)

**4<sup>th</sup> result->01101101101010100**

c0a8-> 1100000010101000(fourth + next hex)

5<sup>th</sup> result-> 0011010011101001 (1 carry)

So (0011010011101001+0000000000000001)

5<sup>th</sup> result->0011010011101010

1dbf->0001110110111111 (fifth + next hex)

6<sup>th</sup> result->01010010101001

0d43->0000110101000011(sixth + next hex)

7<sup>th</sup> result->0000110101000100 (carry 1)

So 7<sup>th</sup> result->0000110101000100

0905->0000100100000101

Final result->01011001001001

1's complement of final result:

1110100110110110->2ffb (checksum)

Therefore,

the packet is verified.