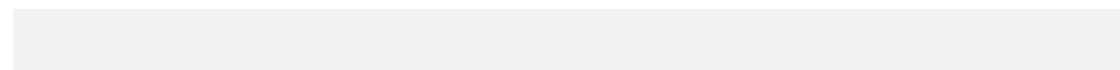


# COMPUTER NETWORK ASSSIGNMENT -2

## WIRESHARK

*Submitted By:*  
*Srinidhi Kadapanatham*  
*1002111809*



*Find a situation in which you may use Wireshark to observe or track protocols as they are being used while engaging with a network application of your choice. Write a brief report that details, step by step, how Wireshark is being used to examine or follow protocols.*

*2. Go to any unsecured website (Example: <http://vbsca.ca/login/login.asp>) and use Wireshark to check if the password is encrypted. Provide screenshots from Wireshark. HINT: Use contains filter*

*3. Using Wireshark demonstrate TCP three-way handshake. Explain  $tcp.completeness == 31$*

*4. For a request, capture the TCP flow using the Wireshark Flow chart that highlights the [SYN], [SYN,ACK],[ACK] and [FIN,ACK]. Explain what it does.*

*5. Using nslookup find your host's MAC address and compare it using WireShark.*

*6. What is the Internet address of the google.com? What is the Internet address of your computer?*

*7. Which version of HTTP—1.0 or 1.1—is your browser using?*

*Click the following URL <http://gaia.cs.umass.edu/ethereal-labs/INTRO-ethereal-file1.html> To*

*load the above page what version of HTTP is the server using?*

### **Solution for 2:**

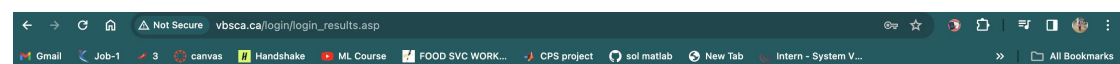
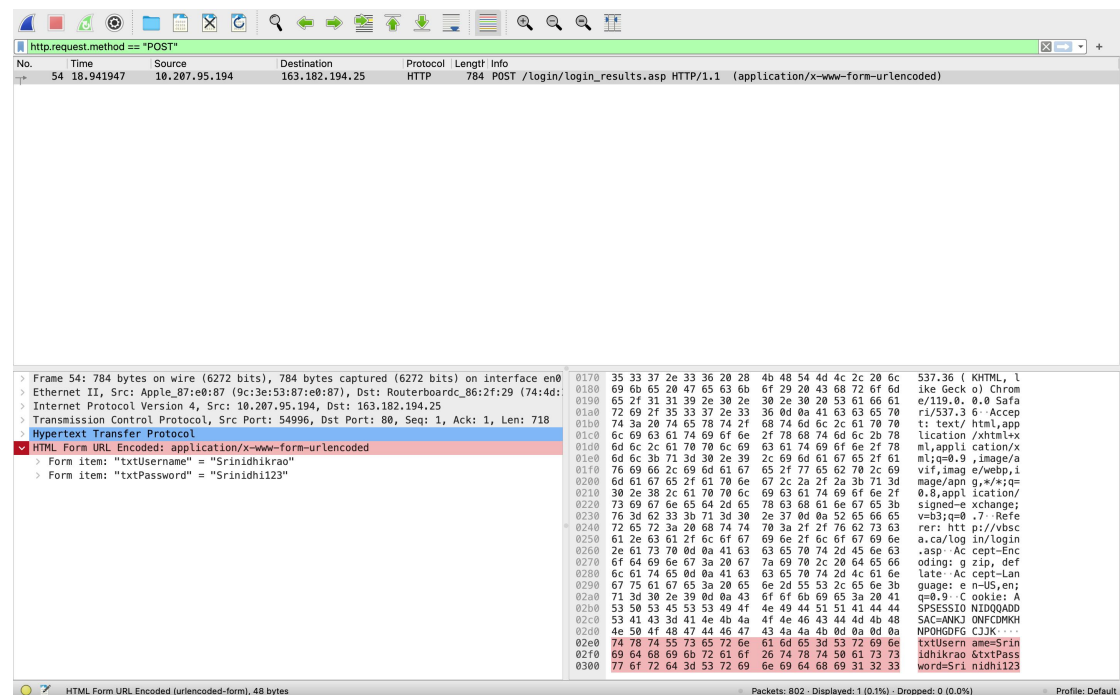
#### **Steps for this:**

Step 1: open wireshark run the packets

Step 2: open the unsecured website <http://vbsca.ca/login/login.asp>

Step 3: Then stop the packets using filter search for the login packet by using `http.request.method=POST`

Step 4 : Under HTML form url we can see in wireshark that the details used to login the webpage is unencrypted and same can be seen in the webpage when we inspect the webpage as well the user id and password is unencrypted which can be seen in the screenshot.



Sorry, but the username that you entered does not exist.



### Solution for 3:

3 way Handshake: Establishing a dependable communication channel between two devices over a TCP/IP network is done through the TCP three-way handshake. It is an essential component of the TCP protocol, guaranteeing that data is prepared for transmission and reception on both ends. The handshake consists of these three steps:

1. SYN: Client sends a TCP packet with SYN flag to tell that it wants to establish connection.

**SYN-ACK:** The server responds with a TCP packet that has both the SYN and ACK flags set, acknowledging the client's request and showing its readiness to establish a connection.

**ACK:** The client sends a TCP packet with the ACK flag set, acknowledging the server's response. At this point, the connection is established, and data can start flowing in both directions.

Step 1: Once the packets are captured in Wireshark.

Step 2: Open a website [www.wikipedia.org](http://www.wikipedia.org).

Step 3: Then apply filter for TCP traffic by using TCP in filter box. (tcp.flags.reset==31)

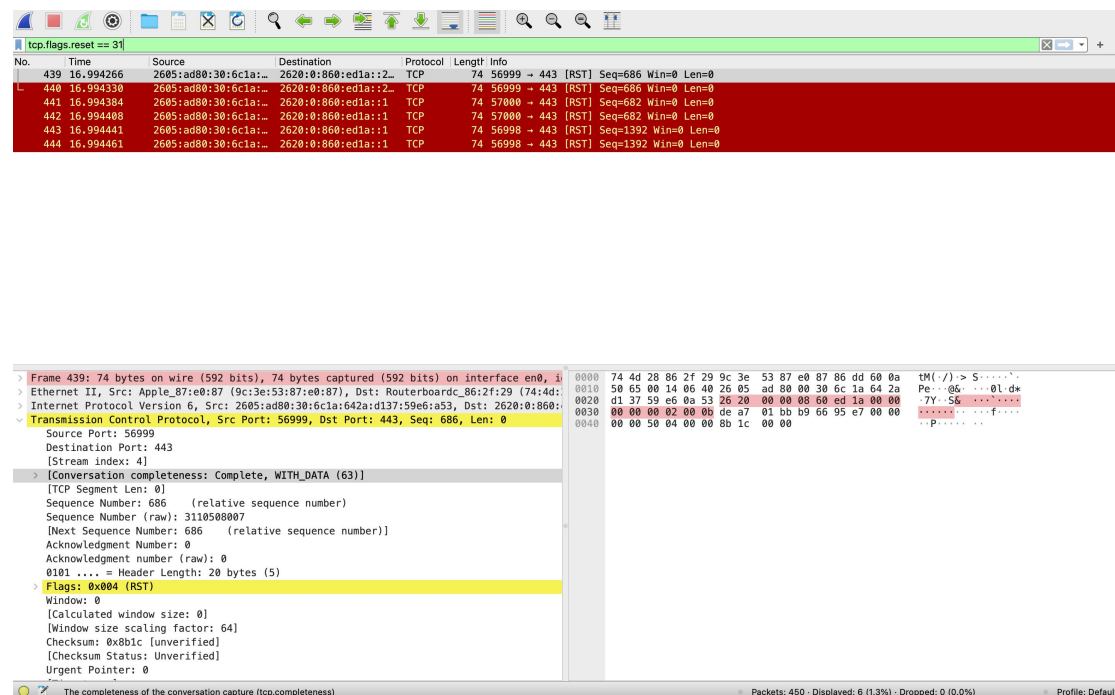
Step 4: We can see here SYN, SYN-ACK and ACK.

Step 5: At SYN check for source and destination IP addresses, it indicates it's a SYN packet.

Step 6: Second packet is SYN-ACK, it should show the acknowledgment of SYN packet with both SYN and ACK.

Step 7: In the third selected packet ACK, it acknowledges the SYN-ACK that it wants to establish connection and wants to acknowledge the same and thus completes 3-way handshake.

Regarding the `tcp.completeness == 31`, this is likely a display filter in Wireshark that filters for packets where the TCP segment is complete, meaning all the expected data has been received. The completeness value of 31 may indicate a fully captured TCP segment.



#### Solution for 4:

Step 1: Start capturing packets in Wireshark.

Step 2: Open any web browser and navigate to [www.wikipedia.org](http://www.wikipedia.org).

Step 3: Stop the capturing of packets and analyse the captured data.

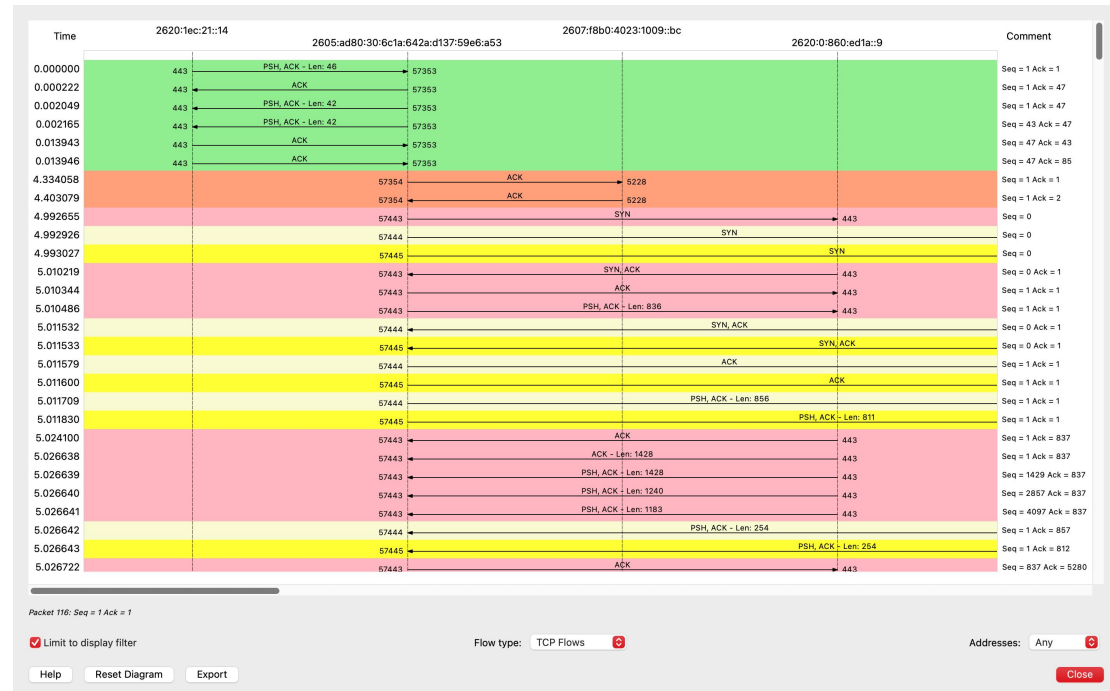
Step 4: From menu bar select statistics -> Flow graph

Step 5: Observe the graph generated by filtering out some additional information.

TCP flow information is as follows:

1. The initial Connection is established with a 3 way handshake SYN, SYN-ACK, ACK.
2. Data is transmitted with a Push(PSH) flag and acknowledged.
3. The connection is terminated with a FIN-ACK and a final ACK.
4. A reset (RST) packet is sent to terminate the connection abruptly.

This includes information about different TCP communication events such as SYN (synchronization), ACK (acknowledgment), PSH (push), and data lengths the data includes timestamps and IP addresses associated with these communication events. This type of data is commonly used for network analysis and troubleshooting to understand the flow of communication and identify any potential issues or anomalies.



### Solution for 5:

Step 1: open wireshark run the packets

Step 2: open a website (www.sephora.com)

Step 3: pause the packets on wireshark, open the website and once that is completed, open terminal type ifconfig

Step 4: Once the ip address and mac address of the system is visible choose en0 and select ether and copy this address on notepad.

Step 5: Once this is completed compare the address with wireshark by using this filter `eth.addr==MAC address of the pc`.

Step 6: Expand the ethernet II src and check for source MAC address, since this will be same we can

In this way we can compare both the PC Mac address and compare it to wireshark.

```

stf0: flags=0<> mtu 1280
anp11: flags=8863<UP,BROADCAST,SMART,RUNNING,SIMPLEX,MULTICAST> mtu 1500
options=400<CHANNEL_IO>
ether 82:f7:fa:cc:12:22
media: none
status: inactive
anp10: flags=8863<UP,BROADCAST,SMART,RUNNING,SIMPLEX,MULTICAST> mtu 1500
options=400<CHANNEL_IO>
ether 82:f7:fa:cc:12:21
media: none
status: inactive
en3: flags=8863<UP,BROADCAST,SMART,RUNNING,SIMPLEX,MULTICAST> mtu 1500
options=400<CHANNEL_IO>
ether 82:f7:fa:cc:12:01
nd6 options=201<PERFORMNUD,DAD>
media: none
status: inactive
en4: flags=8863<UP,BROADCAST,SMART,RUNNING,SIMPLEX,MULTICAST> mtu 1500
options=400<CHANNEL_IO>
ether 82:f7:fa:cc:12:02
nd6 options=201<PERFORMNUD,DAD>
media: none
status: inactive
en1: flags=8963<UP,BROADCAST,SMART,RUNNING,PROMISC,SIMPLEX,MULTICAST> mtu 1500
options=400<TSO4,TSO6,CHANNEL_IO>
ether 36:ad:b1:59:2f:40
media: autoselect <full-duplex>
status: inactive
en2: flags=8963<UP,BROADCAST,SMART,RUNNING,PROMISC,SIMPLEX,MULTICAST> mtu 1500
options=400<TSO4,TSO6,CHANNEL_IO>
ether 36:ad:b1:59:2f:44
media: autoselect <full-duplex>
status: inactive
bridge0: flags=8863<UP,BROADCAST,SMART,RUNNING,SIMPLEX,MULTICAST> mtu 1500
options=63<RXCSUM,TXCUM,TSO4,TSO6>
ether 36:ad:b1:59:2f:40
Configuration:
  id 0:0:0:0:0:0 priority 0 hellotime 0 fdbdelay 0
  maxage 0 holdent 0 proto stp maxaddr 100 timeout 1200
  root id 0:0:0:0:0:0 priority 0 ifcost 0 port 0
  ipfilter disabled flags 0x0
  member: en1 flags=3<LEARNING,DISCOVER>
    ifmaxaddr 0 port 0 priority 0 path cost 0
  member: en2 flags=3<LEARNING,DISCOVER>
    ifmaxaddr 0 port 9 priority 0 path cost 0
  nd6 options=201<PERFORMNUD,DAD>
  media: <unknown type>
  status: inactive
spi: flags=8863<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu 1500
options=440<TSO4,TSO6,CHANNEL_IO,PARTIAL_CSUM,ZEROINVERT_CSUM>
ether be:3e:53:87:e0:87
inet6 fe80::dc3e:53ff:fe87:e087::1 prefixlen 64 scopeid 0xb
nd6 options=201<PERFORMNUD,DAD>
media: autoselect <unknown type>
status: inactive
en0: flags=8863<UP,BROADCAST,SMART,RUNNING,SIMPLEX,MULTICAST> mtu 1500
options=440<TSO4,TSO6,CHANNEL_IO,PARTIAL_CSUM,ZEROINVERT_CSUM>
ether 9c:3e:53:87:e0:87
inet6 fe80::8231:e427:58e2:9eb3::1 prefixlen 64 secured scopeid 0xc
inet6 2605:ad80:30:6c1a:14ff:844f:dsd4:f960 prefixlen 64 autoconf secured
inet6 2605:ad80:30:6c1a:14ff:844f:dsd4:f960 prefixlen 64 autoconf temporary
inet 10.207.95.194 netmask 0xfffff000 broadcast 10.207.95.255
nd6 options=201<PERFORMNUD,DAD>

```

Wi-Fi: en0

eth.addr == 9c:3e:53:87:e0:87

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	2620:1ec:21::14	2605:ad80:30:6c1a::...	TCP	74	443 → 49156 [ACK] Seq=1 Ack=1 Win=163
2	0.000001	2620:1ec:21::14	2605:ad80:30:6c1a::...	TCP	74	443 → 49156 [ACK] Seq=1 Ack=2352 Win=
3	0.141258	2620:1ec:21::14	2605:ad80:30:6c1a::...	TLSv1...	389	Application Data
4	0.141466	2605:ad80:30:6c1a::...	2620:1ec:21::14	TCP	74	49156 → 443 [ACK] Seq=2352 Ack=316 Wi
5	0.349436	2620:1ec:21::14	2605:ad80:30:6c1a::...	TLSv1...	120	Application Data
6	0.349632	2605:ad80:30:6c1a::...	2620:1ec:21::14	TCP	74	49156 → 443 [ACK] Seq=2352 Ack=362 Wi
7	0.351367	2605:ad80:30:6c1a::...	2620:1ec:21::14	TLSv1...	116	Application Data
8	0.351513	2605:ad80:30:6c1a::...	2620:1ec:21::14	TLSv1...	116	Application Data
9	0.359346	2620:1ec:21::14	2605:ad80:30:6c1a::...	TLSv1...	120	Application Data
10	0.359428	2605:ad80:30:6c1a::...	2620:1ec:21::14	TCP	74	49156 → 443 [ACK] Seq=2436 Ack=408 Wi
11	0.360958	2605:ad80:30:6c1a::...	2620:1ec:21::14	TLSv1...	116	Application Data
12	0.362523	2620:1ec:21::14	2605:ad80:30:6c1a::...	TCP	74	443 → 49156 [ACK] Seq=408 Ack=2436 Wi
13	0.373397	2620:1ec:21::14	2605:ad80:30:6c1a::...	TCP	74	443 → 49156 [ACK] Seq=408 Ack=2478 Wi
14	2.089291	2605:ad80:30:6c1a::...	2607:f8b0:4023:100...	QUIC	1292	Initial, DCID=3a6aed3b38b15d1d, PKN:
15	2.089451	2605:ad80:30:6c1a::...	2607:f8b0:4023:100...	QUIC	138	0-RTT, DCID=3a6aed3b38b15d1d
16	2.089787	2605:ad80:30:6c1a::...	2607:f8b0:4023:100...	QUIC	1288	0-RTT, DCID=3a6aed3b38b15d1d
17	2.089800	2605:ad80:30:6c1a::...	2607:f8b0:4023:100...	QUIC	666	0-RTT, DCID=3a6aed3b38b15d1d
18	2.105907	2607:f8b0:4023:100...	2605:ad80:30:6c1a::...	QUIC	1292	Protected Payload (KP0)
19	2.105908	2607:f8b0:4023:100...	2605:ad80:30:6c1a::...	QUIC	854	Protected Payload (KP0)
20	2.105909	2607:f8b0:4023:100...	2605:ad80:30:6c1a::...	QUIC	234	Protected Payload (KP0)
21	2.105910	2607:f8b0:4023:100...	2605:ad80:30:6c1a::...	QUIC	89	Protected Payload (KP0)

> Frame 1: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface en0, id 0000

> Ethernet II, Src: Routerboardc\_86:2f:29 (74:4d:28:86:2f:29), Dst: Apple\_87:e0:87 (9c:3e:53:87:e0:87),

> Internet Protocol Version 6, Src: 2620:1ec:21::14, Dst: 2605:ad80:30:6c1a:642a:d137:59e6

> Transmission Control Protocol, Src Port: 443, Dst Port: 49156, Seq: 1, Ack: 1, Len: 0

0000 9c 3e 53 87 e0 87 74 4c

0010 37 01 00 14 06 3b 26 20

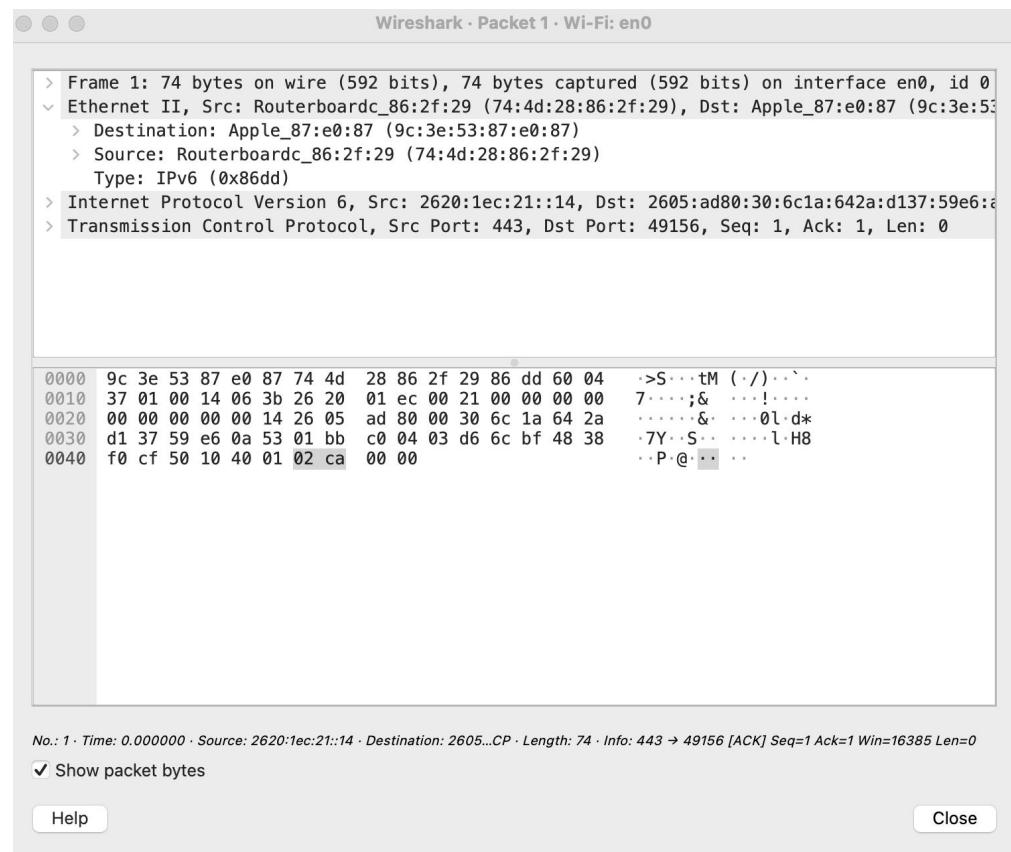
0020 00 00 00 00 00 14 26 05

0030 d1 37 59 e6 0a 53 01 b1

0040 f0 cf 50 10 40 01 02 ca

wireshark\_Wi-Fi1COGF2.pcapng

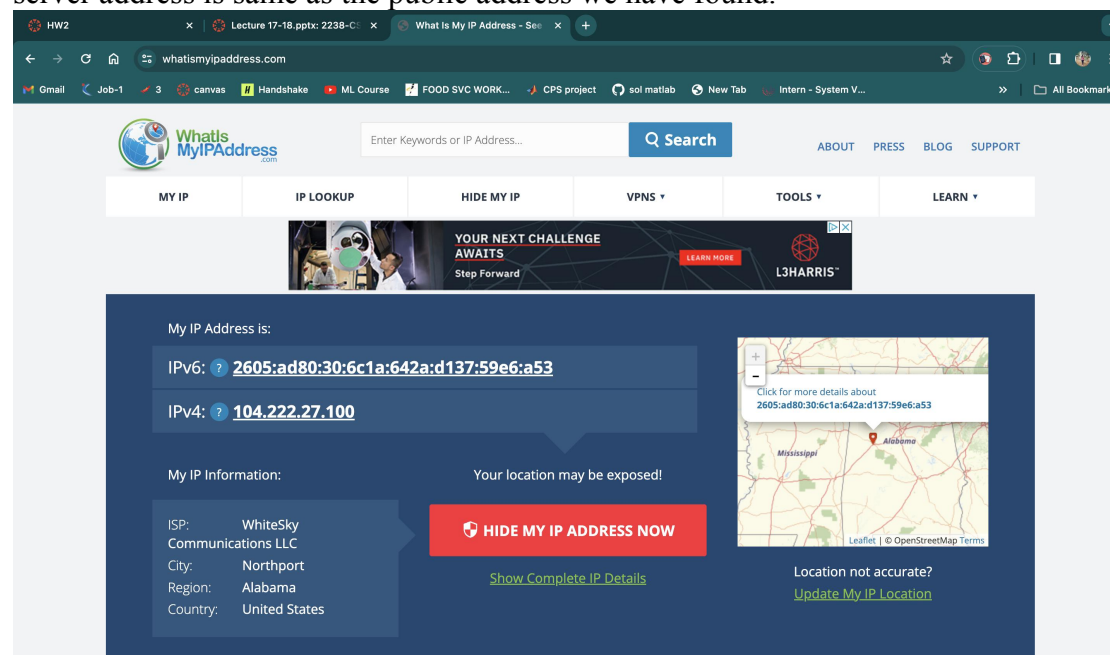
Packets: 13393 · Displayed: 13393 (100.0%) · Dropped: 0 (0.0%) · Profile: Default



### Solution for 6:

Step 1: in a web browser check for pc's public ip address using what's my ip? and hence you get the below ip address.

Step 2: Now go to terminal type the command `nslookup google.com` we can see the server address is same as the public address we have found.



IP of google.com



```
(base) srinidhi@Srinidhis-MacBook-Pro ~ % lookup google.com
zsh: command not found: lookup
(base) srinidhi@Srinidhis-MacBook-Pro ~ % nslookup google.com
Server:      2605:ad00:1f:c000::6
Address:     2605:ad00:1f:c000::6#53

Non-authoritative answer:
Name:   google.com
Address: 142.250.115.102
Name:   google.com
Address: 142.250.115.139
Name:   google.com
Address: 142.250.115.100
Name:   google.com
Address: 142.250.115.138
Name:   google.com
Address: 142.250.115.101
Name:   google.com
Address: 142.250.115.113

(base) srinidhi@Srinidhis-MacBook-Pro ~ %
```

## Solution for 7:

Step 1: Here packets are filters using http and thus there are 4 packets for a particular website given in the question I have found 4 packets of http and its using 1.1

The screenshot shows a Wireshark packet capture of HTTP traffic. The packet list at the top shows four packets:

No.	Time	Source	Destination	Protocol	Length	Info
57	5.385965	10.207.197.112	128.119.245.12	HTTP	459	GET /ethereal-labs/INTRO-ethereal-file1.html HTTP/1.1
59	5.388811	128.119.245.12	10.207.197.112	HTTP	491	HTTP/1.1 200 OK (text/html)
61	5.810577	10.207.197.112	128.119.245.12	HTTP	416	GET /favicon.ico HTTP/1.1
62	5.866238	128.119.245.12	10.207.197.112	HTTP	538	HTTP/1.1 404 Not Found (text/html)

The packet details pane for the selected packet (Frame 59) shows the following structure:

- Frame 59: 491 bytes on wire (3928 bits), 491 bytes captured (3928 bits) on interface en0, id 0
- Ethernet II, Src: Routerbo\_86:2f:29 (74:4d:28:86:2f:29), Dst: Apple\_5b:b8:e4 (1c:57:dc:5b:b8:e4)
  - Destination: Apple\_5b:b8:e4 (1c:57:dc:5b:b8:e4)
  - Source: Routerbo\_86:2f:29 (74:4d:28:86:2f:29)
  - Type: IPv4 (0x0800)
- Internet Protocol Version 4, Src: 128.119.245.12, Dst: 10.207.197.112
- Transmission Control Protocol, Src Port: 80, Dst Port: 52023, Seq: 1, Ack: 406, Len: 437
- Hypertext Transfer Protocol**
  - Line-based text data: text/html (3 lines)

The packet bytes pane on the right shows the raw data in hexadecimal and ASCII.