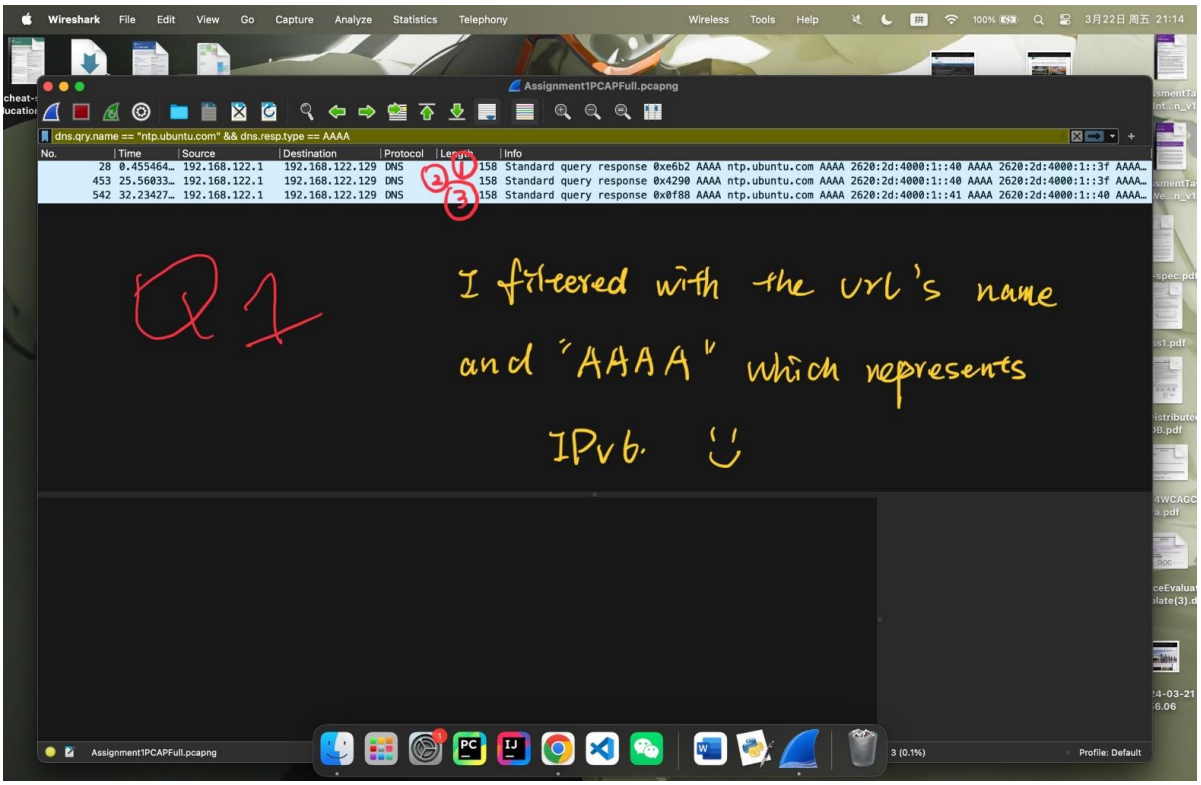
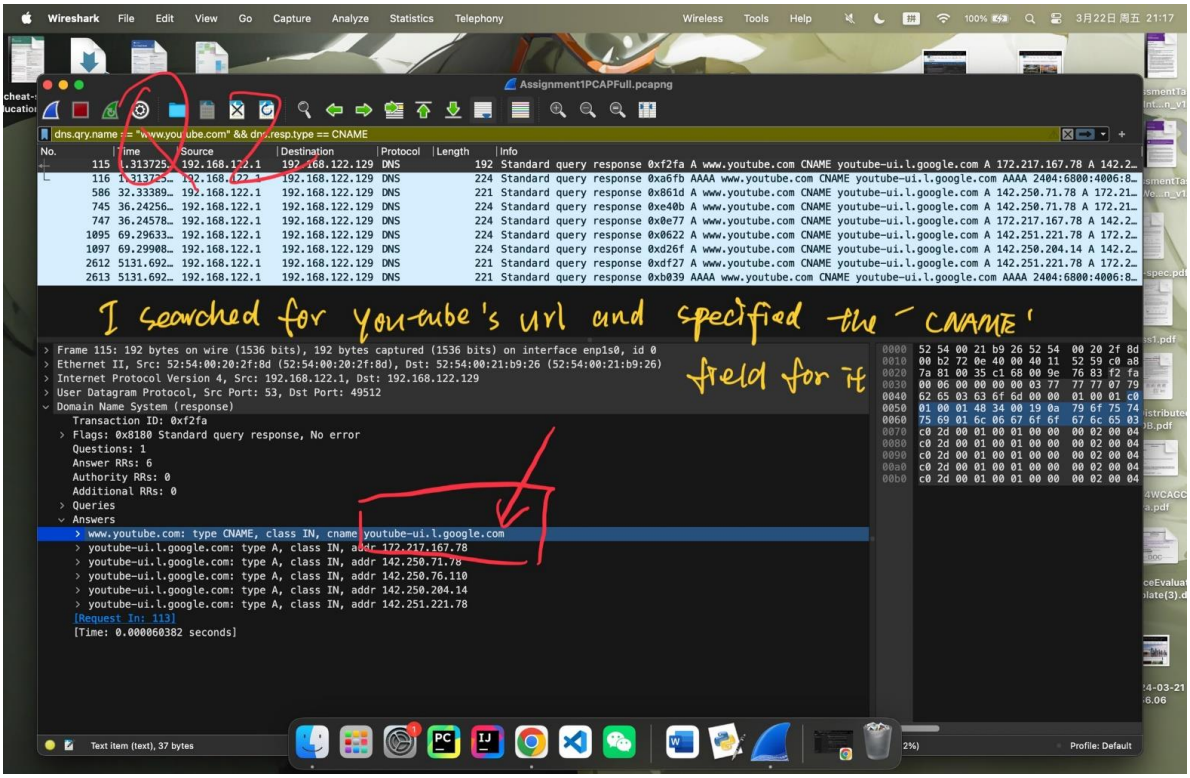
Question 1:

As this is a DNS query, firstly I need to search for the protocol name “dns”. And then as the host name is “ntp.ubuntu.com”, I useded the filter “qry.name”. For the question asked me to find the “return query” with IPv6 Addresses, I used the filter of “resp” signaling respond and with “AAAA” representing IPv6. And as shown in the picture, there are three responses satisfying the above filtering.



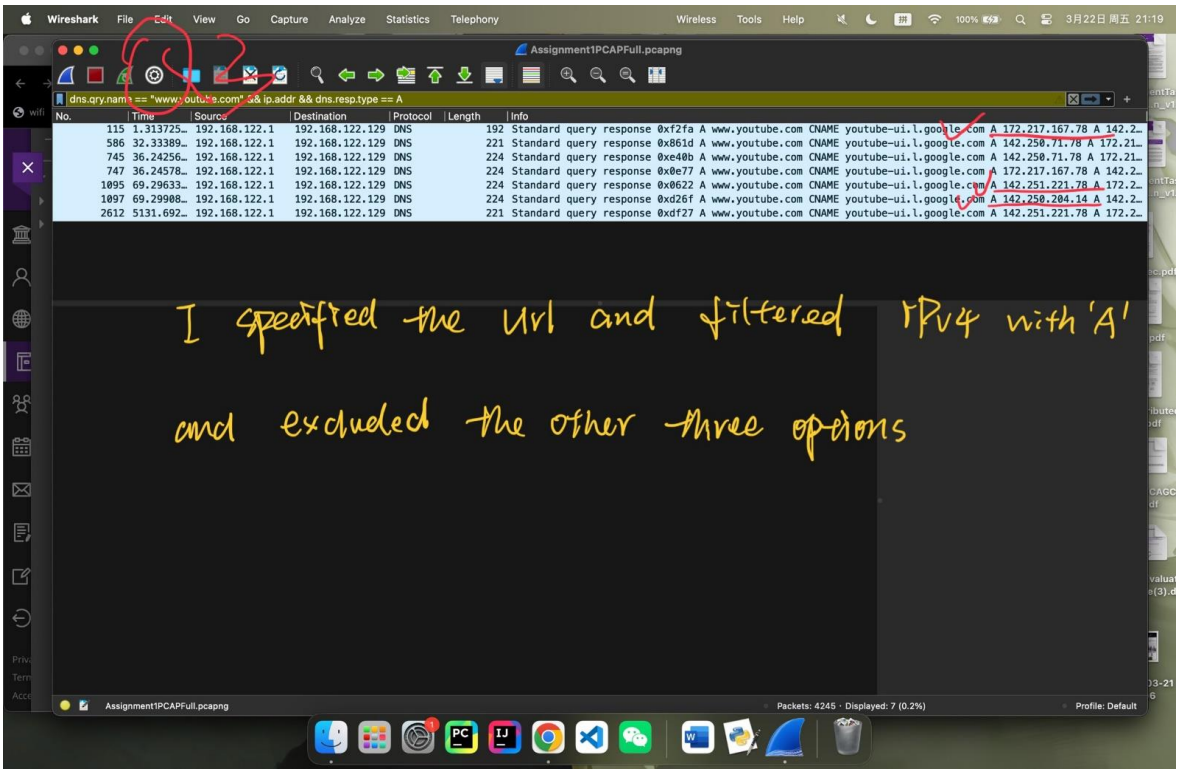
Question 2:

For the qustion ask me to find the canonical domain name, I need to check response type of “CNAME” in this DNS query. So firstly I filtered the host name with dns.query.name == www.youtube.com and then search for the response type of CNAME, and find the canonial name in the response of the domain name system.



Question 3:

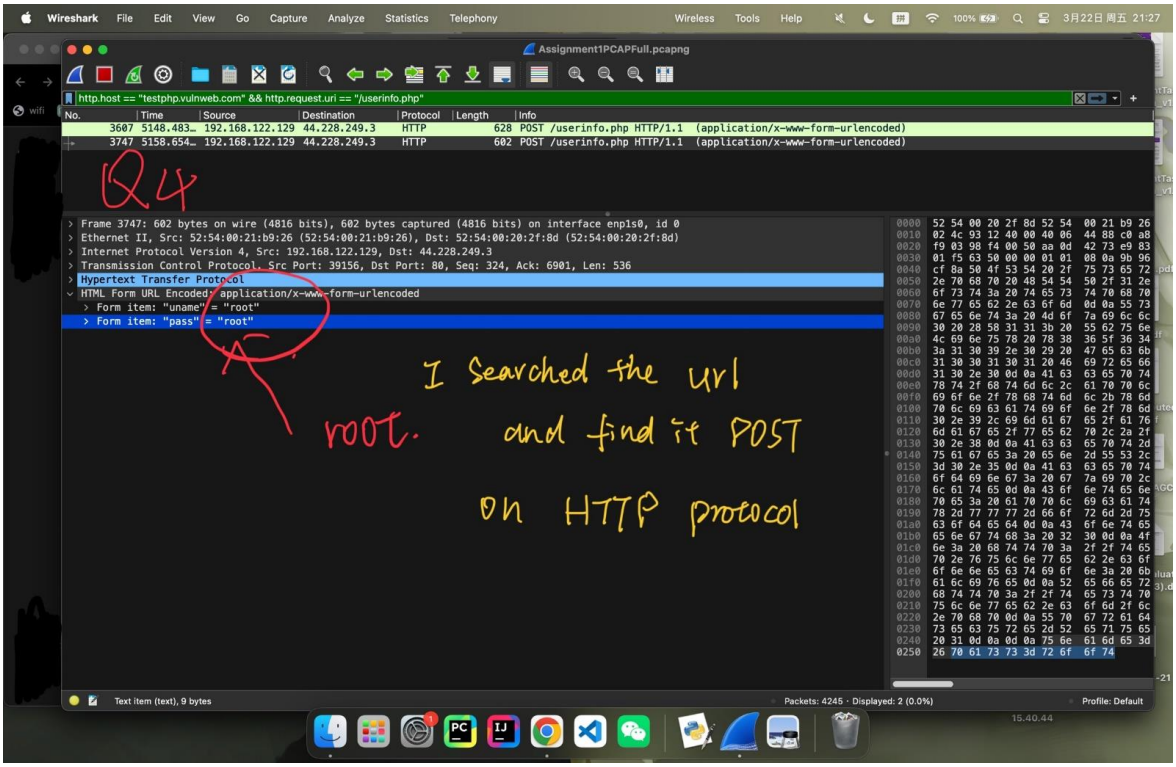
As the question asks me to find the IPv4 address to find the DNS response query, so I used the filter “dns.resp.type == A”. So after searching with the domain name and the ip type, we checked each option one by one to exclude the wrong option.



Question 4:

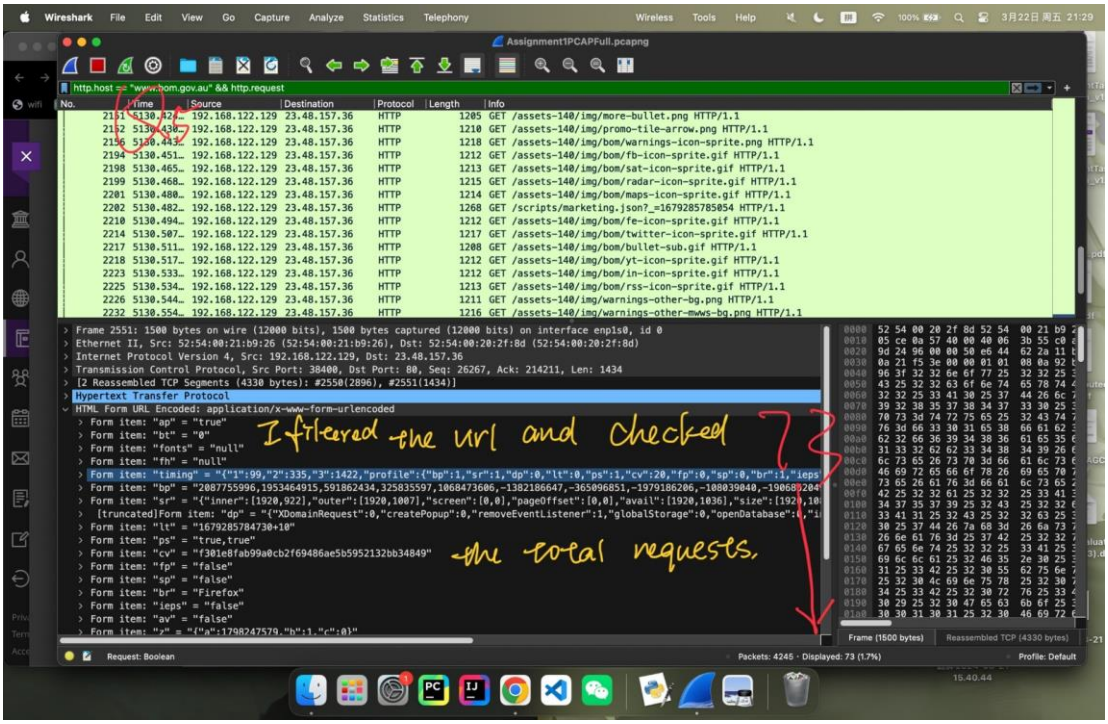
As we need to check for the password, it is evident that we need to post something to the webpage. So we can use the filter of the http protocal with the post method. Also we need to add the hostname and the subsequting url to filter with the url specified in the question.

After filtering, we only need to see what is post from the url under the HTML Form URL Encoded. It is evident that both the uname and password are filled with “root”.



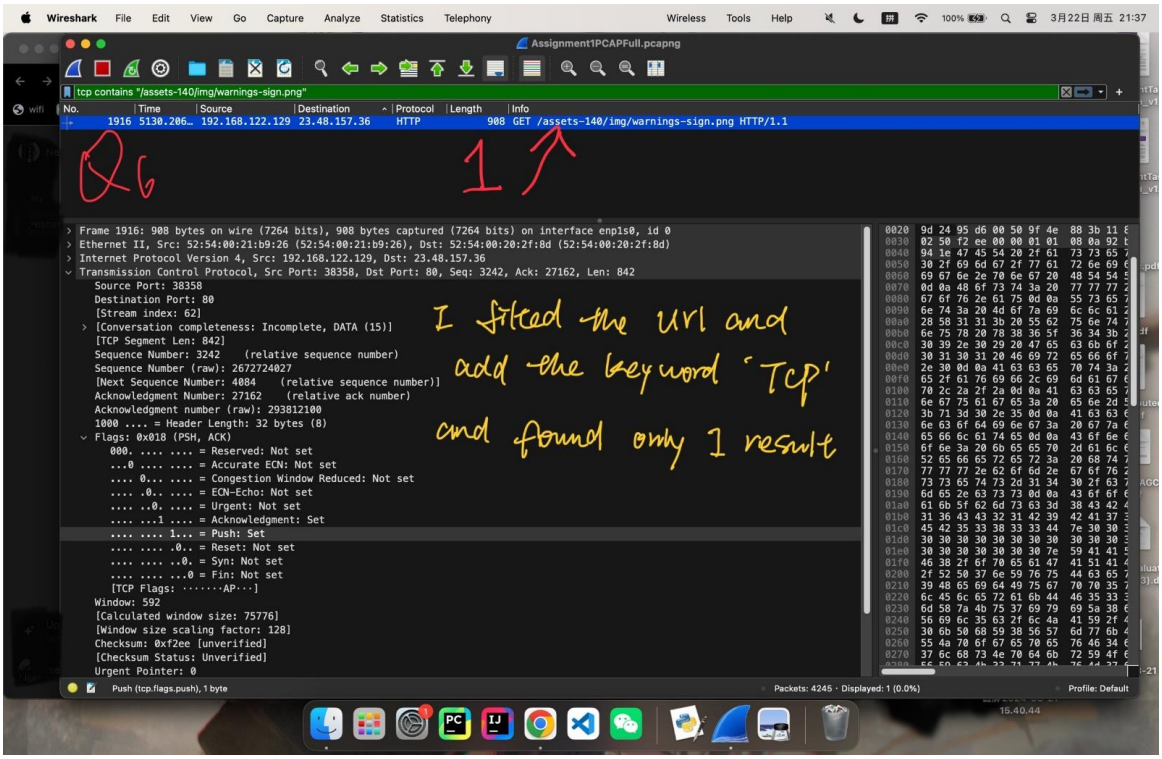
Question 5:

As the question only asked for the number of the requests, so we need to filter with the protocol of “HTTP” with the hostname. The total number of the requests can be found at the bottom of the wireshark showing the number of the displayed results.



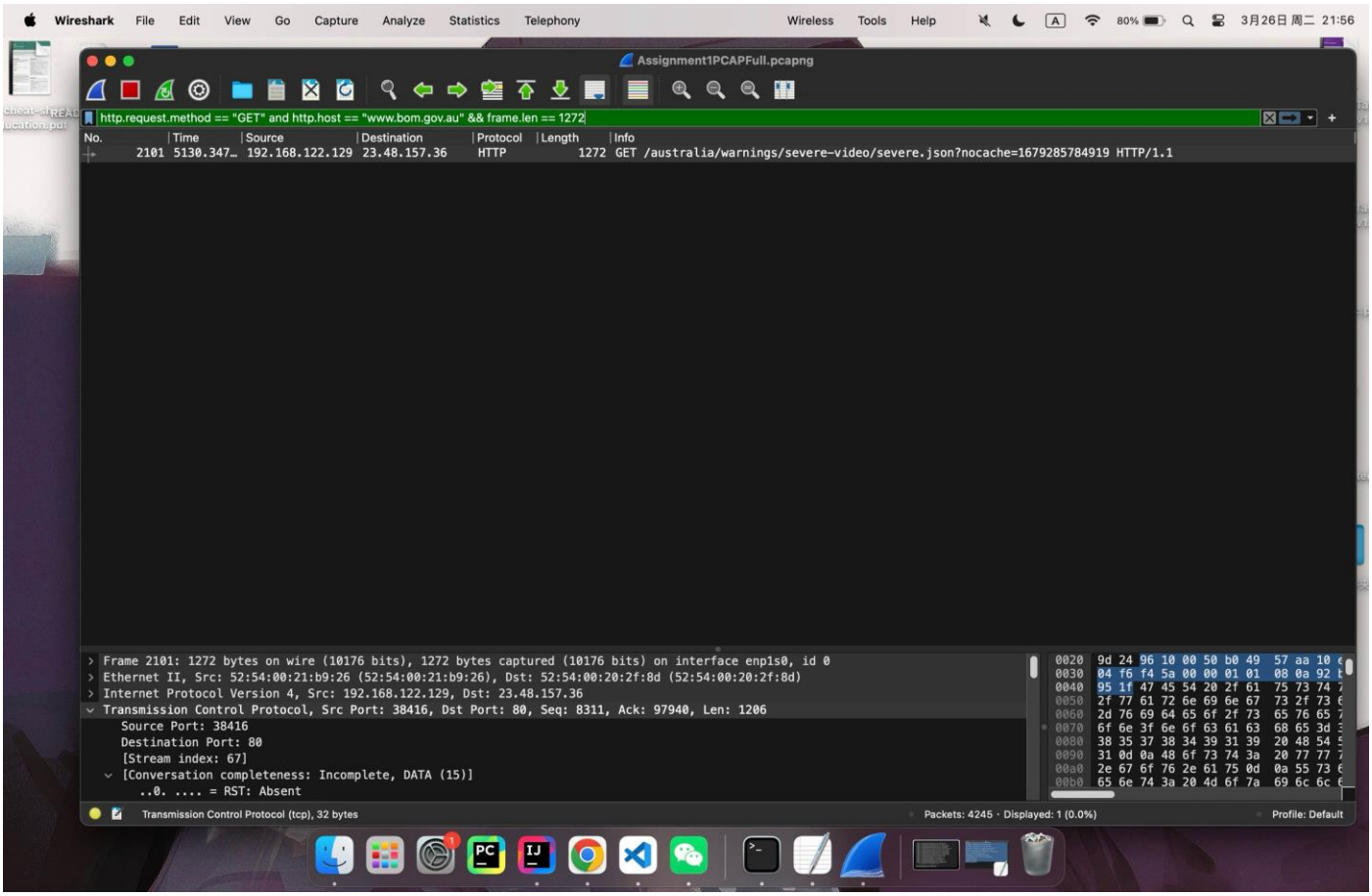
Question 6:

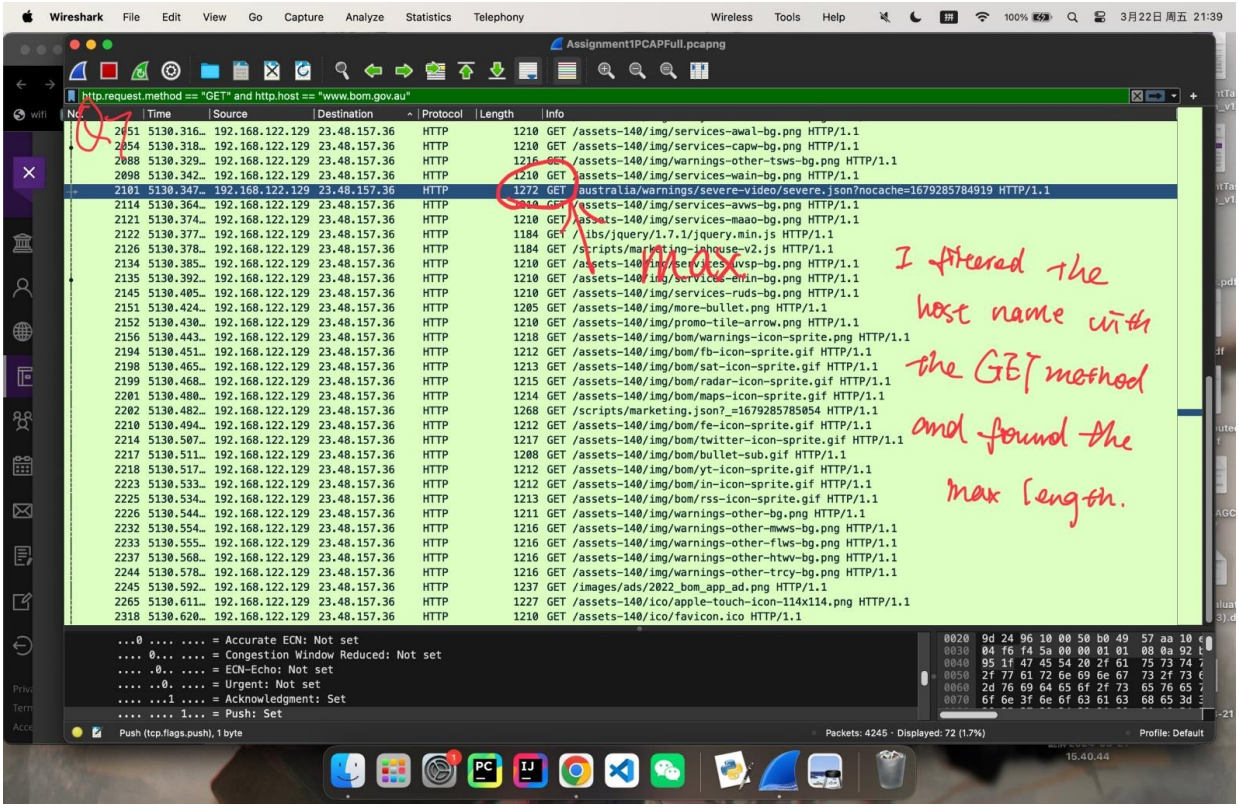
For this question, I just filter my request with the url (hostname should be added actually). And then I found that there is only one requests of the GET image requests. Then I opened the TCP part as the following picture shows, and it’s clear that the length of this TCP segment length is 842 bytes. However, I couldn’t find the MSS in the TCP part in this packet (Not shown in the Option part), so I just assume the MSS is 1460 bytes as mentioned in the lecture slides. Therefore, only one segment are needed for this image.



Question 7:

As the question shows, I filtered with the protocol of “HTTP” with the method of “GET” and the host name of the www.bov.gov.au. To find the largest size, I just select the maximum number in the options (1500). And it shows that there is NO request with the length of 1500, so then I filter it with the length of 1272 and found that there is a result. To double check, I also looked through all length of requests manually to find the largest number.

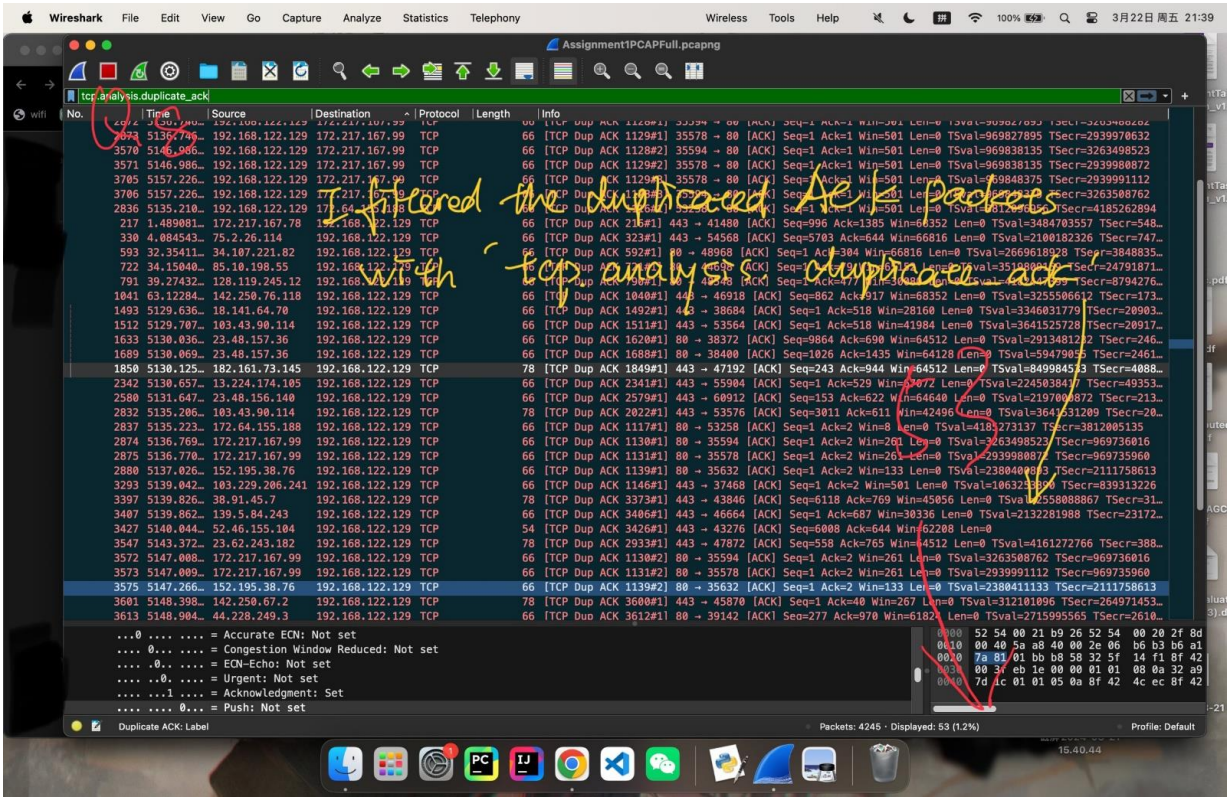




Question 8:

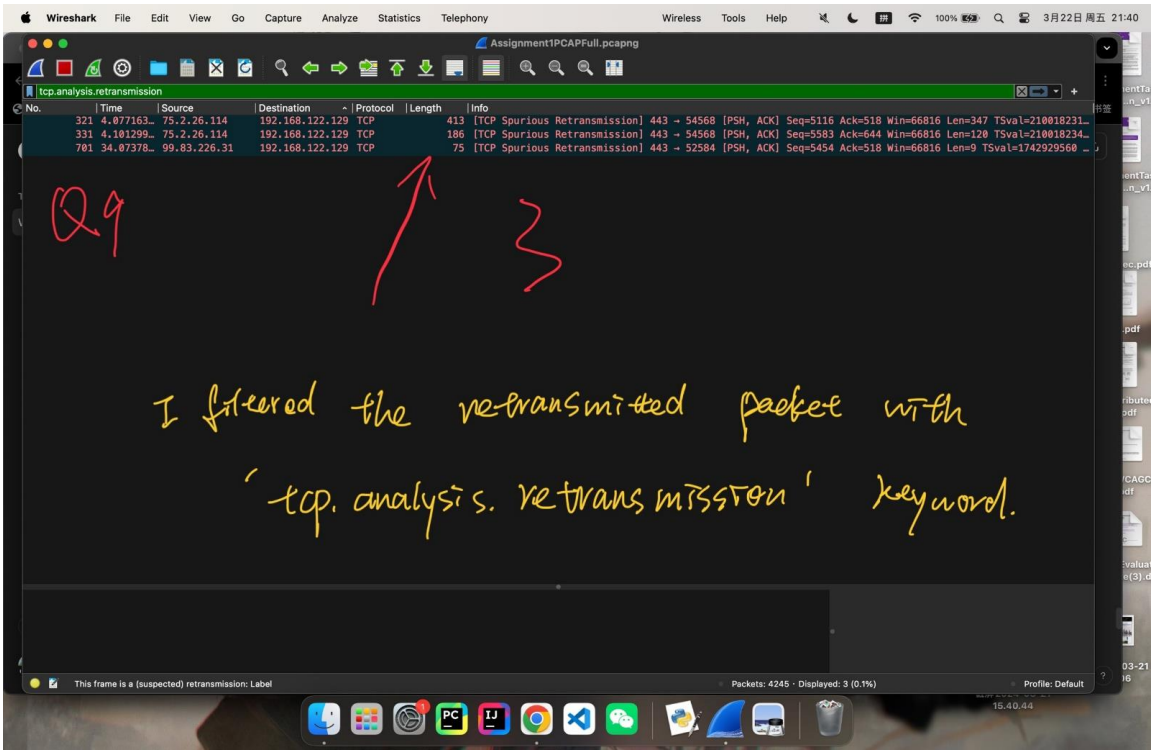
As the question asked for the number of duplicate ACK packets in TCP, I just use the filter query with

“tcp.analysis.duplicate_ack”. And the total number displayed can be found at the bottom of the wireshark.



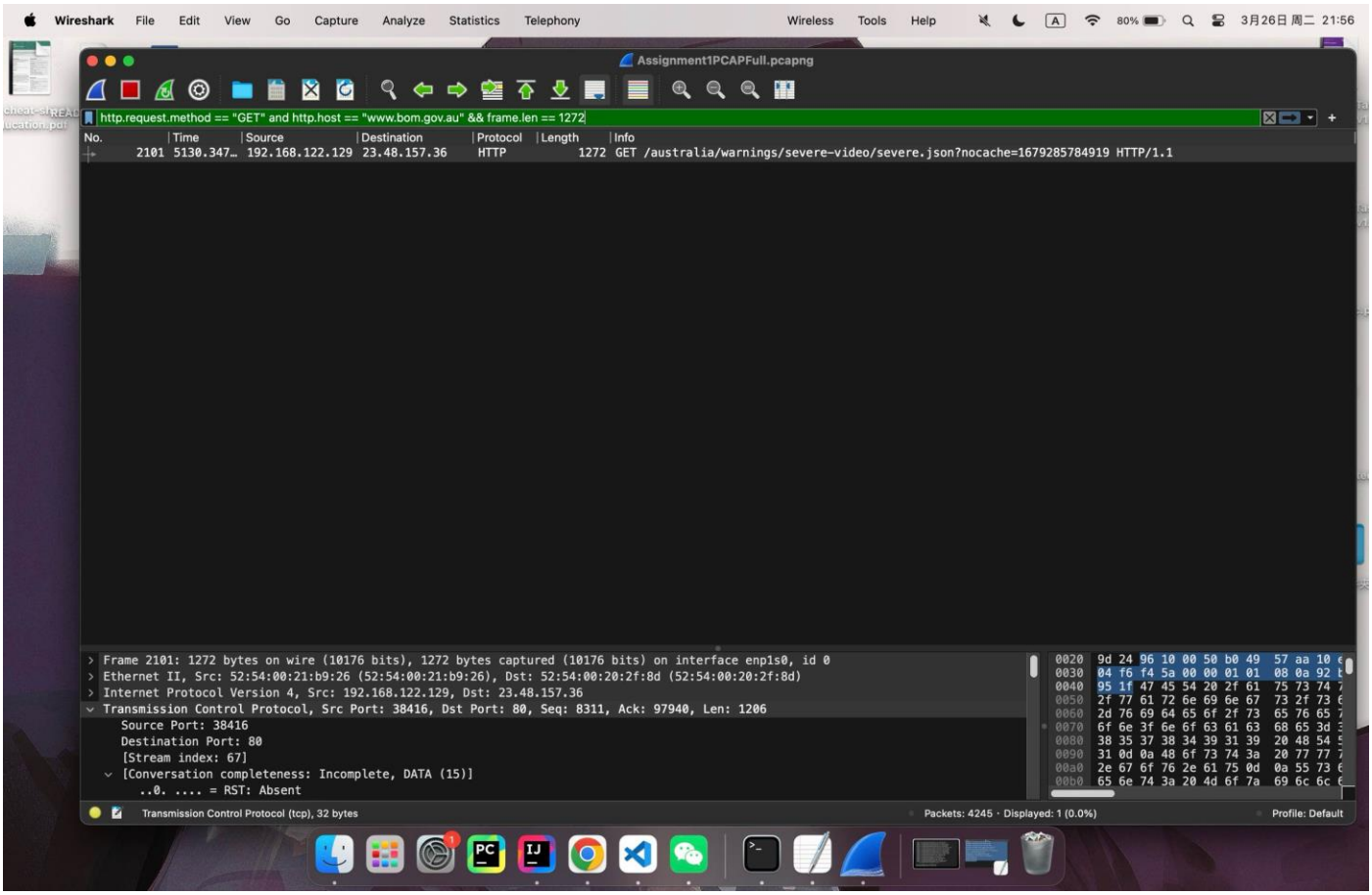
Question 9:

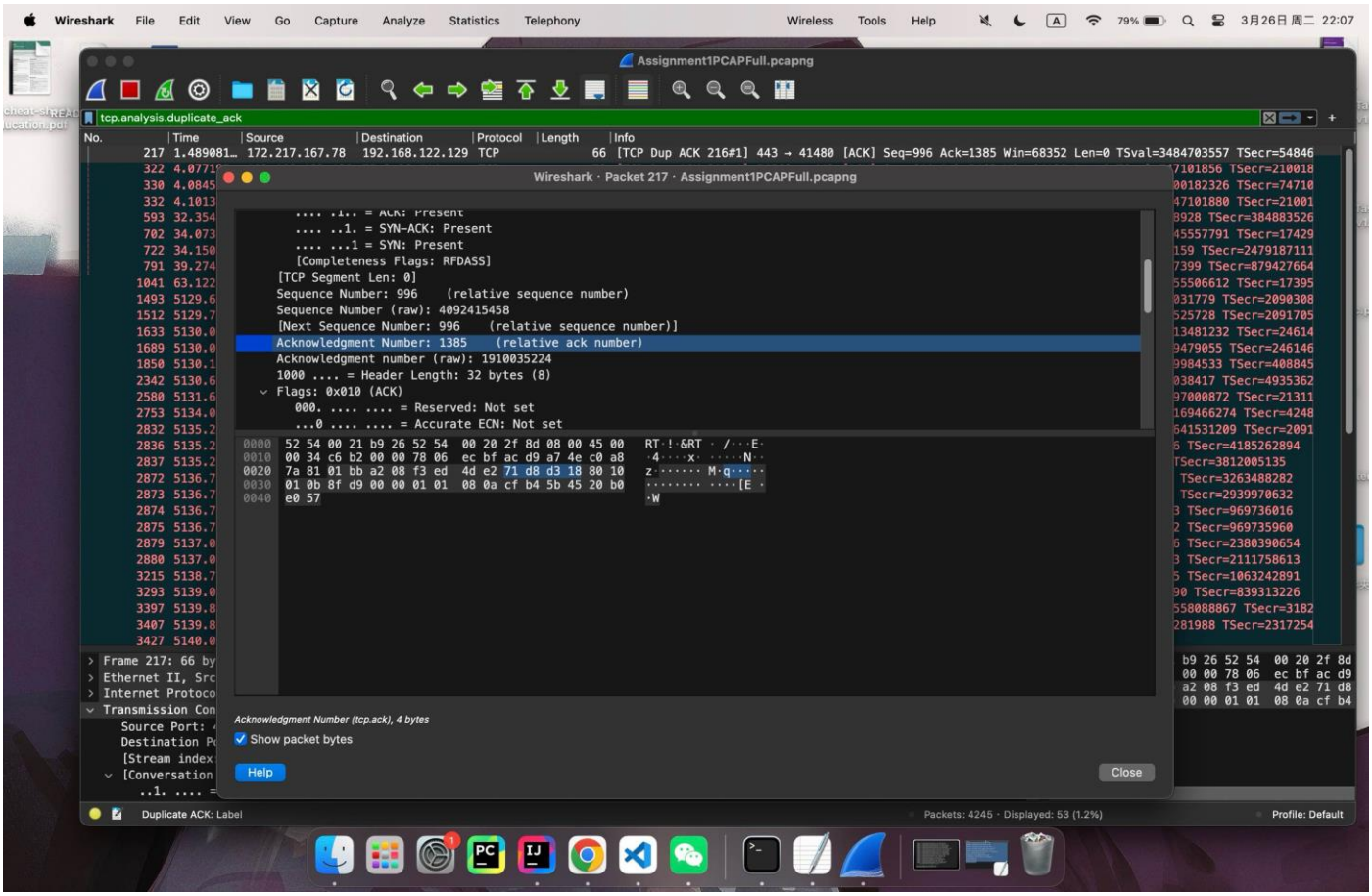
For the retransmitted packets, I just use the filter query of “tcp.analysis.retransmission”. And wireshark shows there are three retransmitted packets.



Question 10:

With the filtering query of “tcp.analysis.duplicate_ack”, I just choose the minimum time request to check the relative acknowledgement number in its TCP header. And as the following picture shows that, the relative acknowledgement number is 1385.


Wireshark File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Assignment1PCAPFull.pcapng
http.request.method == "GET" and http.host == "www.bom.gov.au" && frame.len == 1272
No. Time Source Destination Protocol Length Info
2101 5130.347… 192.168.122.129 23.48.157.36 HTTP 1272 GET /australia/warnings/severe-video/severe.json?nocache=1679285784919 HTTP/1.1
Frame 2101: 1272 bytes on wire (10176 bits), 1272 bytes captured (10176 bits) on interface enp1s0, id 0
Ethernet II, Src: 52:54:00:21:b9:26 (52:54:00:21:b9:26), Dst: 52:54:00:20:2f:8d (52:54:00:20:2f:8d)
Internet Protocol Version 4, Src: 192.168.122.129, Dst: 23.48.157.36
Transmission Control Protocol, Src Port: 38416, Dst Port: 80, Seq: 8311, Ack: 97940, Len: 1206
Source Port: 38416
Destination Port: 80
[Stream index: 67]
[Conversation completeness: Incomplete, DATA (15)]
..0. .... = RST: Absent
Transmission Control Protocol (tcp), 32 bytes
Packets: 4245 · Displayed: 1 (0.0%)
Profile: Default


Wireshark File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Assignment1PCAPFull.pcapng
tcp.analysis.duplicate_ack
No. Time Source Destination Protocol Length Info
217 1.489081… 172.217.167.78 192.168.122.129 TCP 66 [TCP Dup ACK 216#1] 443 → 41480 [ACK] Seq=996 Ack=1385 Win=68352 Len=0 TSval=3484703557 TSecr=54846
Wireshark · Packet 217 · Assignment1PCAPFull.pcapng
.... ..1. = SYN-ACK: Present
....1 = SYN: Present
[Completeness Flags: RFDASS]
[TCP Segment Len: 0]
Sequence Number: 996 (relative sequence number)
Sequence Number (raw): 4092415458
[Next Sequence Number: 996 (relative sequence number)]
Acknowledgment Number: 1385 (relative ack number)
Acknowledgment number (raw): 1910035224
1000 .... = Header Length: 32 bytes (8)
Flags: 0x010 (ACK)
000. .... = Reserved: Not set
...0 .... = Accurate ECN: Not set
0000 52 54 00 21 b9 26 52 54 00 20 2f 8d 08 00 45 00 RT·!·&RT · /···E·
0010 00 34 c6 b2 00 00 78 06 ec bf ac d9 a7 4e c0 a8 ·4····x· ·····N··
0020 7a 81 01 bb a2 08 f3 ed 4d e2 71 d8 d3 18 80 10 z····· M·q·····
0030 01 0b 8f d9 00 00 01 01 08 0a cf b4 5b 45 20 b0 ·····[E ·
0040 e0 57 ·W
Acknowledgment Number (tcp.ack), 4 bytes
Show packet bytes
Help
Close
Duplicate ACK: Label
Packets: 4245 · Displayed: 53 (1.2%)
Profile: Default