COMP/ELEC 416: Computer Networks

Project#2

Transport Protocols and Secure Sockets Layer Analysis with Wireshark

Group 7

Emir Atışay, 53590

Ege Berk Süzgen, 53714

Onur Şahin, 64232

03.04.2020

Table of Contents

Introduction:..... 3

Project Guideline:..... 3

Part 1: SSL Implementation and Experiments:..... 3

Part 2: Wireshark Analysis..... 3

Analysis of Experiment:..... 4

Part 1 - SSL Implementation and Experiments:..... 4

Part 2.a - TCP Experiments:..... 7

Part 2.b - Comparison (TCP vs SSL) Experiments:..... 13

Part 3 - UDP Experiments:..... 14

Part 4 – Experiments for Two Competing Streams:..... 16

Conclusion:..... 17

Task Distrubtion:..... 18

Introduction:

This project is based on applying some of network protocols and structures and examining the packet flow in a detailed manner via Wireshark. The study was mainly on the transport layer of the network such as TCP, UDP and SSL. In addition, while using "nslookup" function the DNS protocol is used beside the UDP. So, the application layer is also referred in this project. In order to examine transport layers individually, three different programs are used: For examining SSL, a java code is constructed, for the UDP, the command prompt is used and for the TCP part, the web browser is used. Regardless of the packet type, all of them were examined in the Wireshark with the same operations.

Project Guideline:

Part 1: SSL Implementation and Experiments:

To have the experiment done, one should have following steps done:

1. Set up the Wireshark for sniffing.
2. From the source code, run the main class of the Server.
    a. When you first run the server, both TCP and SSL servers will be initialized.
    b. As they are initialized, both will begin to listen to corresponding ports.
3. From the source code, run the main class of the Client.
    a. Main class initiates a connection class with the client.
    b. Connection class declares a TCP connection with the server.
4. Enter one of the predefined user info from the users.txt file.
    a. Server checks the validation of the user profile.
    b. Certificate transfer occurs if the user is verificated.
    c. TCP connection closed and SSL connection observed.
    d. Parsed message sent to client from server via SSL connection.
5. Observe the console of the Client side to see the message and connection status.
6. Analyze the packet details via the Wireshark.

Part 2: Wireshark Analysis

The following steps will be a guide for the Wireshark Analysis part in the rest of the project. With this content any other projects can be also analyzed.

1. The type of the network connection should be chosen properly. In this project except the first part which TCP and SSL examined through the java code, all other parts are performed on the internet connection with the modem. So, if the connection is wireless, the wi-fi option should be preferred. If it is wireless, the appropriate local network connection should be preferred. For the Part 1, adapter for the loopback capture is chosen as it is suggested in the project manual.
2. The session in the Wireshark should be started before performing any kind of network operation and closed after performing the desired network operation.
3. The packets in the session are automatically ordered according to the time that they are transacted. However, it is possible to sort them according to the headers or apply a display filter to see some of them in a specific manner.
4. In order to examine a packet in a more detailed way, there is the captured segment which can be displayed in addition to general and it gives opportunity to see what is inside that package such as header including both IPs and port numbers, payload carrying the data, checksum of the packet and so on so forth.

Analysis of Experiment:

Part 1 - SSL Implementation and Experiments:

1- First, we tried to have the sum of our IDs, 171.536, as the desired port. However, as it could not work, we have searched about the maximum port number that can be assigned. In the light of that research, we see that the maximum number which a port can take is 65535, because ports are 16-bits numbers. So, the total possible number is 2 to the power 16 which is 65536. The reason for the difference (1) is that it starts from zero. Figure 1 shows SSL packages after applying the “ssl” display filter. It is observed that both address of the source and the destination are the same which is “127.0.0.1”. However, in order to find server’s and client’s port numbers, one of the SSL packets should be investigated.

ssl

No.	Time	Source	Destination	Protocol	Length	Info
192	13.277902	127.0.0.1	127.0.0.1	TLSv1.3	419	Client Hello
194	13.304303	127.0.0.1	127.0.0.1	TLSv1.3	171	Server Hello
196	13.315425	127.0.0.1	127.0.0.1	TLSv1.3	50	Change Cipher Spec
198	13.317011	127.0.0.1	127.0.0.1	TLSv1.3	50	Change Cipher Spec
200	13.325737	127.0.0.1	127.0.0.1	TLSv1.3	114	Application Data
202	13.328404	127.0.0.1	127.0.0.1	TLSv1.3	968	Application Data
204	13.359074	127.0.0.1	127.0.0.1	TLSv1.3	346	Application Data

Figure 1

After going into details with a SSL packet, both ports are determined. In Figure 1, client's port can be observed as "55162" and server's as "4444".

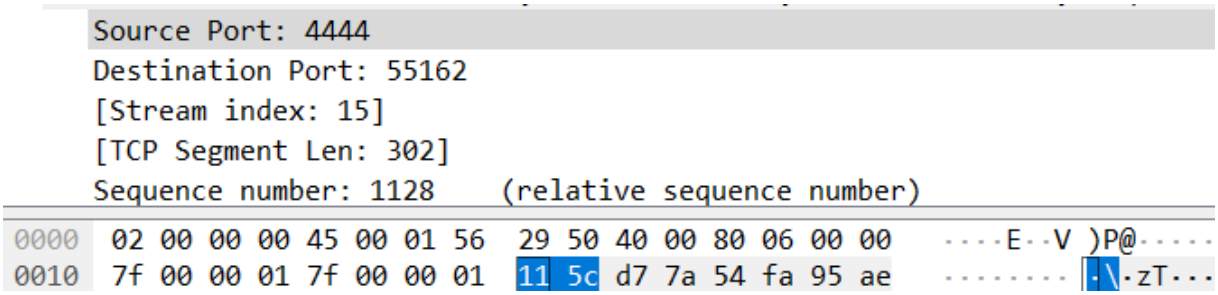


Figure 2

2- The approach is started by applying the display filter "tcp" in order to solely see TCP segments. Then some of the writings in the data field can be seen in Figure 3 which is a combination of several TCP segments (in order to save space).

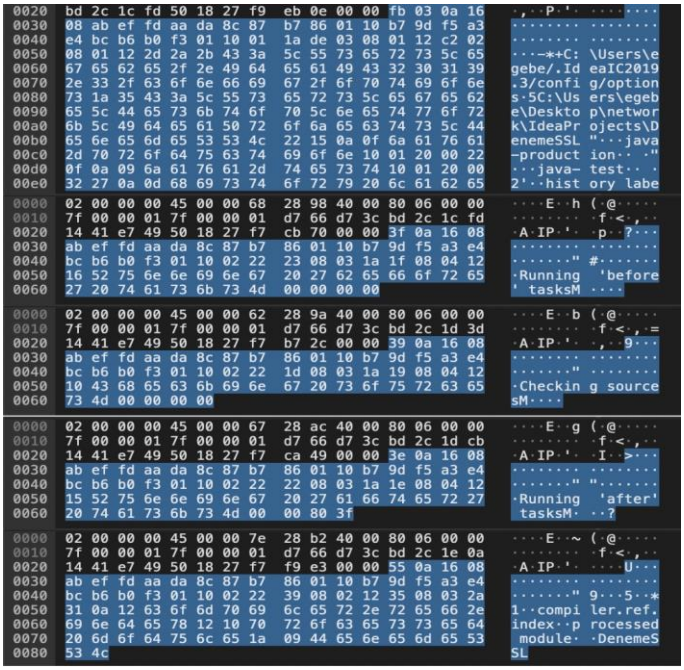


Figure 3

When compared with the data which is exchanged between the client and server, those messages do not exist in the program. The only common messages are the username and the password can be observed in both of them (in question 4, it will be demonstrated). According to the knowledge, it is because of each protocol's own library. For instance, there are connections in handshake which are already implemented in the library.

3- There are 3 students in the project group. So, each message holds 3 characters of each student's email. First of all, the packet number should be greater than 181 because the

email address exchange starts after logging in and password is exchanged in the 181st packet. Then, SSL packets are examined to count the number of packets sent which includes 3 characters. Figure 4 demonstrates that there are several consecutive packets which have the same length (85). So, they should be the packets which exchange 3 characters. The number of packets is 25. So, there should be 25 corresponding TCP segments to these SSL packets.

212	13.374519	127.0.0.1	127.0.0.1	TLSv1.3	85 Application Data
214	13.374826	127.0.0.1	127.0.0.1	TLSv1.3	85 Application Data
216	13.375064	127.0.0.1	127.0.0.1	TLSv1.3	85 Application Data
218	13.375328	127.0.0.1	127.0.0.1	TLSv1.3	85 Application Data
220	13.375587	127.0.0.1	127.0.0.1	TLSv1.3	85 Application Data
222	13.375833	127.0.0.1	127.0.0.1	TLSv1.3	85 Application Data
224	13.376065	127.0.0.1	127.0.0.1	TLSv1.3	85 Application Data
226	13.376346	127.0.0.1	127.0.0.1	TLSv1.3	85 Application Data
228	13.376527	127.0.0.1	127.0.0.1	TLSv1.3	85 Application Data
230	13.376722	127.0.0.1	127.0.0.1	TLSv1.3	85 Application Data
232	13.376958	127.0.0.1	127.0.0.1	TLSv1.3	85 Application Data
234	13.377225	127.0.0.1	127.0.0.1	TLSv1.3	85 Application Data
236	13.377407	127.0.0.1	127.0.0.1	TLSv1.3	85 Application Data
238	13.377686	127.0.0.1	127.0.0.1	TLSv1.3	85 Application Data
240	13.377925	127.0.0.1	127.0.0.1	TLSv1.3	85 Application Data
242	13.378189	127.0.0.1	127.0.0.1	TLSv1.3	85 Application Data
244	13.378419	127.0.0.1	127.0.0.1	TLSv1.3	85 Application Data
246	13.378609	127.0.0.1	127.0.0.1	TLSv1.3	85 Application Data
248	13.378795	127.0.0.1	127.0.0.1	TLSv1.3	85 Application Data
250	13.379481	127.0.0.1	127.0.0.1	TLSv1.3	85 Application Data
252	13.379630	127.0.0.1	127.0.0.1	TLSv1.3	85 Application Data
254	13.379768	127.0.0.1	127.0.0.1	TLSv1.3	85 Application Data
256	13.380483	127.0.0.1	127.0.0.1	TLSv1.3	84 Application Data
258	13.380602	127.0.0.1	127.0.0.1	TLSv1.3	84 Application Data
262	13.386282	127.0.0.1	127.0.0.1	TLSv1.3	84 Application Data

Figure 4

4- The difference between payloads of SSL and TCP is obvious, the payload of the SSL is encrypted which is not the actual transferred message. However, TCP payload is in plain text. Following screenshots demonstrate that TCP payload can be readable, and the login information is transferred by TCP (username is “ege berk”, password is “1234”).

```
Data (10 bytes)
    Data: 656765206265726b0d0a
    [Length: 10]
000  02 00 00 00 45 00 00 32  29 35 40 00 80 06 00 00   ····E··2 )5@·····
010  7f 00 00 01 7f 00 00 01  d7 79 d1 56 24 0f 6e 94   ········ ·y·V$·n·
020  b4 36 e4 e5 50 18 27 f9  08 d4 00 00 65 67 65 20   ·6··P·'· ····ege
030  62 65 72 6b 0d 0a                                 berk··
```

Figure 5

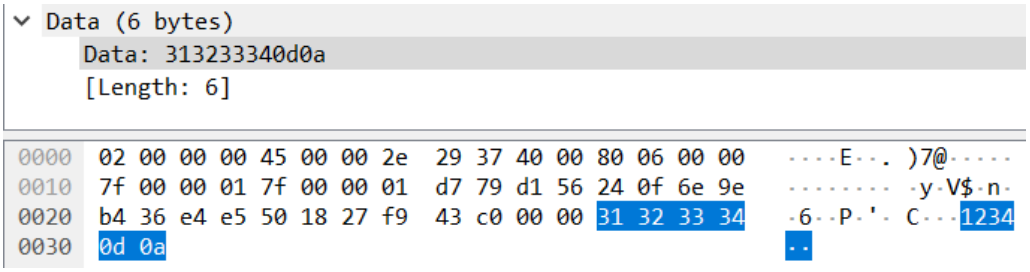


Figure 6

However, reading e-mail addresses is not possible because messages about e-mail are transferred via SSL which is encrypted.

Part 2.a - TCP Experiments:

5- At first, the proper TCP packet should be selected from the Wireshark pane. In order to make sure, all other browsers and programs which use Internet connection are closed. Then the experiment in the manual is performed and observed carefully to determine which is the TCP packet after pressing to the upload button. In the upper pane, the destination IP is written as 128.119.245.12. Figure 7 illustrates the captured segment and the highlighted part is the hexadecimal presentation of the destination's IP address. To demonstrate, "80 77 f5 0c" is converted to decimal presentation to confirm whether it is correct or not. Also, before the destination IP address there is the source's IP address, which is "c0 a8 01 29".

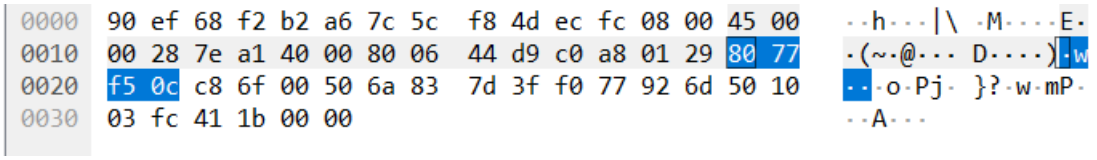


Figure 7

Figure 8 shows the destination port number. It is understood by going forward bytewise. As it is shown in Figure 7, the place of the destination's IP is definite. The next two bytes represent the source's port number. The source port number is "c8 6f" which corresponds to "51311" in decimal. After that the port number of the destination is located. It is shown as "00 50" in hexadecimal form which corresponds to "80" in the decimal representation.

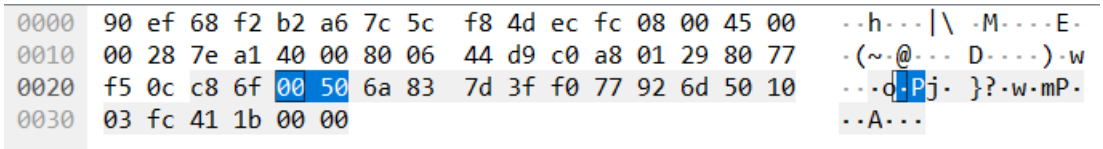


Figure 8

6- The payload can be found at the end of the TCP segment. A TCP segment with the payload is shown in Figure 9.

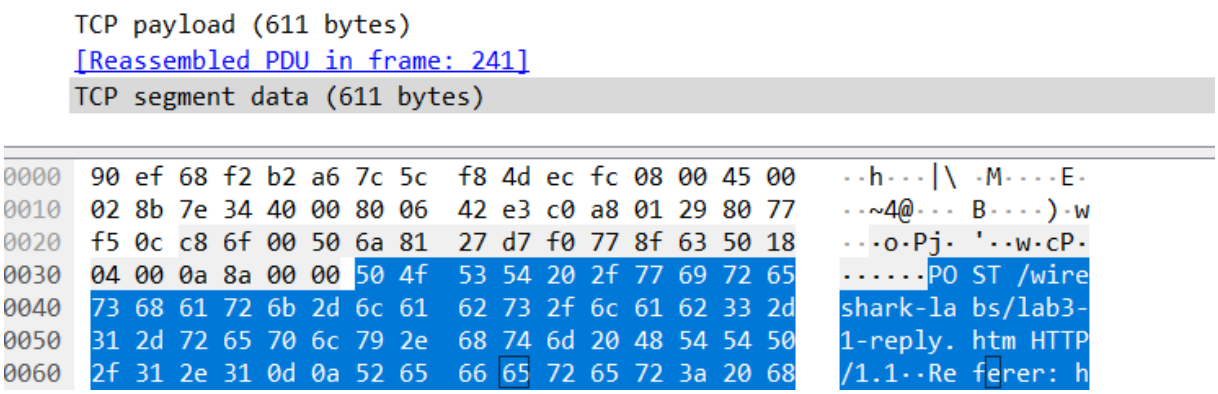


Figure 9

7- In order to understand more efficiently, the general structure of the initial sequence number is searched through the web. When the TCP section is started, it randomly assigns a number between 0 and 4,294,967,295. However, Wireshark displays a relative sequence number instead of the actual one. This relative sequence number is relative to the randomly initiated sequence number. Figure 10 shows the location of the initial sequence number in the captured segment which is 1.787.002.175 in decimal. However, even if the same operation is performed and the same TCP packet is selected, this number will be different, because of the random initiation.

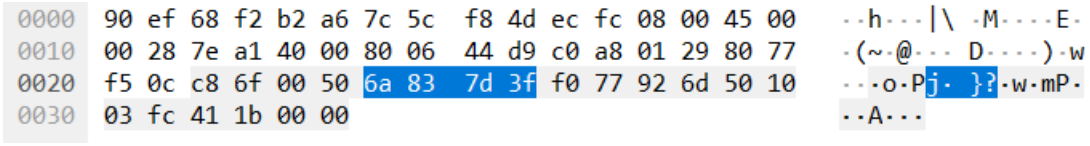


Figure 10

Components of 3-way handshake protocol are SYN (from the client to the server, establishment of a connection), SYNACK (server's response to the client) and ACK (from the client to the server, confirmation that it has received the packet). The sequence numbers of SYN, SYNACK, and ACK are random numbers between 0 and 4,294,967,295 as discussed in the first part of the answer. However, the relative sequence numbers of SYN, SYN-ACK, and ACK are 0, 0, 1, respectively (as shown in Figure 11). The port number used on the client side is "51311" (c8 6f) and the port number used on the server side is "80" (00 50) as mentioned in the 5th answer.

8- Starting from this question, the next three questions will be answered from the below table. It is a Wireshark utility. After choosing the TCP packet, we are navigated to

Statistics tab, then flow graph to construct this TCP Flow in order to answer three questions.

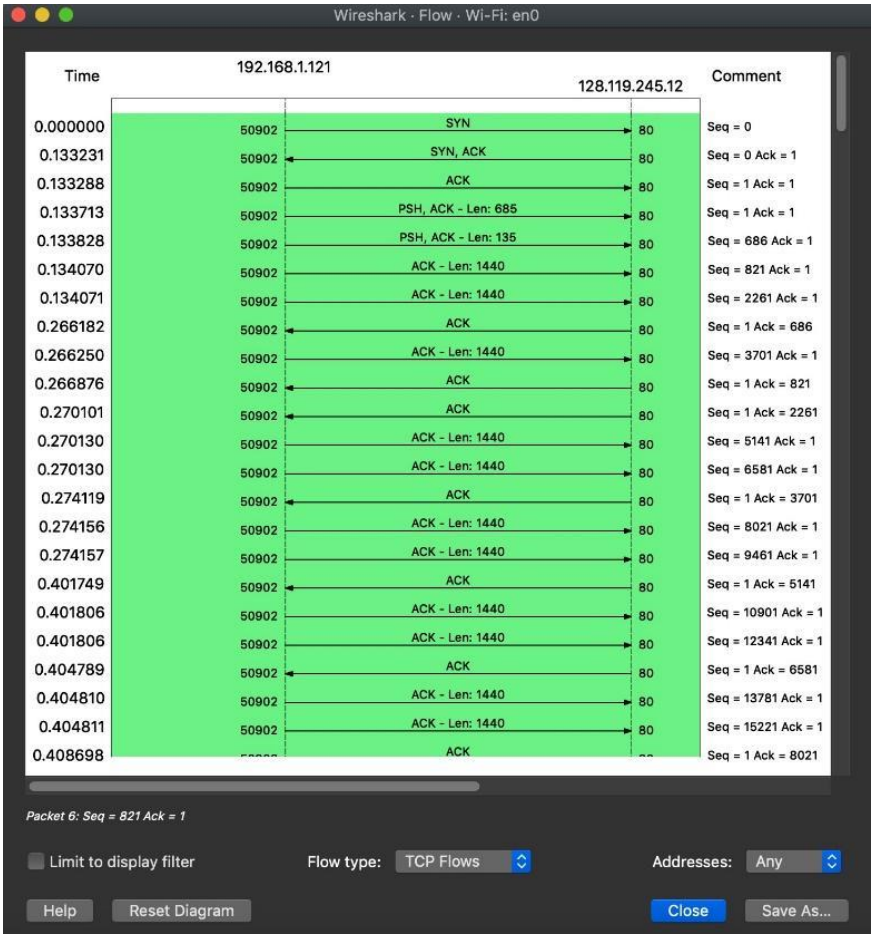


Figure 11

From the above table, it can be observed that at time 0 there is the SYN segment and the sequence number is 0. A segment is identified as the SYN segment if the SYN flag is set to 1.

9- After t=0, the SYNACK segment by gaia.cs.umass.edu to the client computer in reply to the SYN is observed. The sequence number of the SYNACK segment is 0.

10- As it is explained in the previous question, the SYNACK segment is determined as the next one after the initial SYN segment. It can be observed that the value of the Acknowledgement field in the SYNACK is 1. The value of the Acknowledgement field in the SYNACK segment is evaluated by adding 1 to the sequence number of the SYN segment. Since the sequence number of the SYN segment is 0 (as mentioned in the 8th answer), the value of ACK in the SYNACK segment is 1. If the ACK and

SYN flags are both set to 1 in a segment, this segment can be identified as the SYNACK segment (as shown in Figure 12).

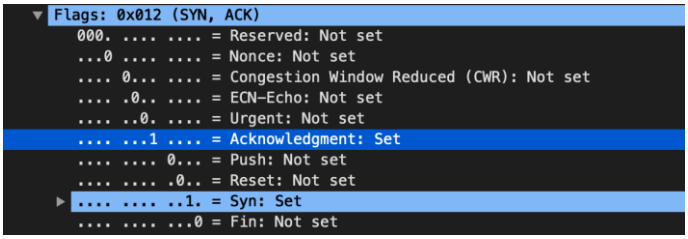


Figure 12

11- As shown in Figure 13, segment No. 4 contains the HTTP POST command in its data field. The sequence number of this segment is 1.

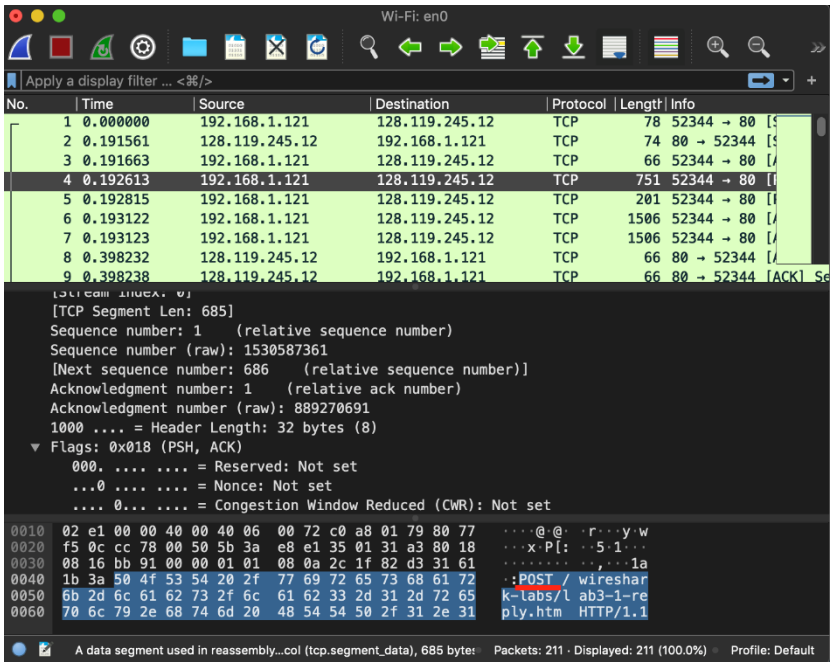
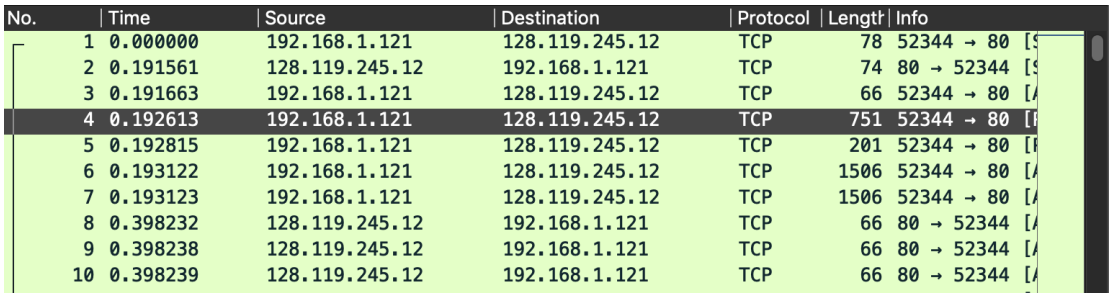


Figure 13

12- As mentioned before, the segment containing the HTTP POST is No. 4. If we consider the TCP segment containing the HTTP POST as the last element in the TCP connection; there are only three segments, not six, and these three segments would be No. 1, 3 and 4. The ACK of these segments are No. 2, 8, 9. Sequence numbers of segments No. 1, 3, 4 are 0, 1 and 1 respectively.



No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.1.121	128.119.245.12	TCP	78	52344 → 80 [S
2	0.191561	128.119.245.12	192.168.1.121	TCP	74	80 → 52344 [S
3	0.191663	192.168.1.121	128.119.245.12	TCP	66	52344 → 80 [A
4	0.192613	192.168.1.121	128.119.245.12	TCP	751	52344 → 80 [P
5	0.192815	192.168.1.121	128.119.245.12	TCP	201	52344 → 80 [P
6	0.193122	192.168.1.121	128.119.245.12	TCP	1506	52344 → 80 [A
7	0.193123	192.168.1.121	128.119.245.12	TCP	1506	52344 → 80 [A
8	0.398232	128.119.245.12	192.168.1.121	TCP	66	80 → 52344 [A
9	0.398238	128.119.245.12	192.168.1.121	TCP	66	80 → 52344 [A
10	0.398239	128.119.245.12	192.168.1.121	TCP	66	80 → 52344 [A

Figure 14

According to Figure 14, sent time and ACK received time obtained.

Segment Time	Sent	ACK received	RTT
Segment No. 1	0.000000	0.191561	0.191561
Segment No. 3	0.191663	0.398232	0.206569
Segment No. 4	0.192613	0.398238	0.205625

The EstimatedRTT equation (Section 3.5.3 in the textbook):

EstimatedRTT (current) = 0.875* EstimatedRTT (previous) + 0.125*SampleRTT

EstimatedRTT for segment No. 1 = 0.191561 seconds

EstimatedRTT for segment No. 3 = 0.875*0.191561 + 0.125*0.206569 = 0.193437 s

EstimatedRTT for segment No. 4 = 0.875*0.193437 + 0.125*0.205625 = 0.1949605 s

13- The below graph demonstrates that the number of IP packets is barely higher than the number of the TCP segments. The numbers are 116 to 114 which shows that there is no significant difference between them. The reason for the difference is there is another protocol such as HTTP to perform the POST request.

Figure 15

14- As the below graph demonstrates there are no retransmitted segments because all numbers are increasing monotonically.

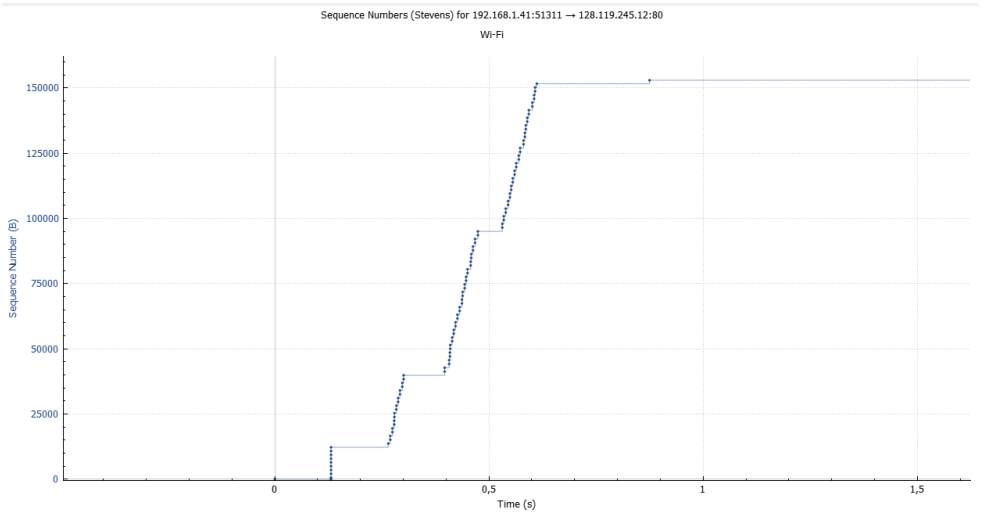


Figure 16

15- In order to identify lost packages, first the I/O graph is examined to visualize whether there is any packet loss occurred or not. However, there were no TCP errors. Then, some of the filters are used to find and detect any packet losses such as tcp.analysis.retransmission and tcp.analysis.lost_segment, but again those display filters did not return any packets. To conclude, there are not any lost packets which are found.

16- As it is explained, in question 7; when the TCP section is started, it randomly assigns a number between 0 and 4,294,967,295. However, Wireshark displays relative sequence number instead of the actual one. This relative sequence number is relative to the

randomly initiated sequence number. So, the answer is obvious now. The chance of getting the same initial sequence number is extremely low, which means that we will not face with the same number in our different tries usually. In addition, this situation is repeated 5 times and we did not see any similarities with the initial sequence number.

Part 2.b - Comparison (TCP vs SSL) Experiments:

17- For retrieving the exchanged application layer message from the captured segments, we converted the hexadecimal payload to ASCII format. First, the integers in the hexadecimal format separated, and then the characters which correspond to these integers appended. The “OK” message, which added the payload, is removed, then the actual message obtained.

18- In question 4, the difference is also explained and demonstrated with screenshots. There are IP addresses and port numbers of source and destination in both cases. The main difference is the visibility of the data in the payload. The data is visible when SSL is not used and not visible when SSL is used. In addition, SSL causes latency because of the secured connection.

19- As explained in the previous parts, Data in the payload is encrypted in the TCP messages with SSL, and using the proper key is the only way for description. In that regard, the content of the data and the payload is not retrievable from a third party without the proper description key. In addition, the change in certificate or key will result in data loss. Parties should be aware in the case of key changes.

20- As we noticed, SSL/TCP segment has two checksums in the checksum field. One of them is for the IP header, and the other one is for the TCP header.

For the IP header, we believed that checksum is necessary to the corruption detection in the IP packet’s header, not the payload. And according to our research, it is named as hop-to-hop checksum, meaning recomputed everytime Ethernet header fields change. For TCP header, checksum is used for the correctness and the integrity of the data. It is basically for the error specification and corruption detection of the TCP header and the data. Furthermore, it is end-to-end checksum, meaning computed by the sender and verified by the receiver.

Part 3 - UDP Experiments:

21- Figure 17 illustrates the filter that is used to see appropriate packets; “ip.addr ==195.175.39.49”. The address is retrieved from the nslookup function which is run in the command line. This command’s run phase will be shown in question 23.

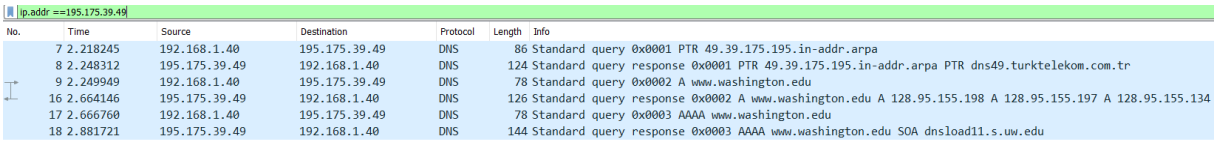


Figure 17

22- As Figure 17 shows, the Protocol which is used in the Application Layer is the DNS Protocol. So, the chosen Transport Layer protocol is UDP. The DNS needs fast and small segments which fits with UDP and in addition, DNS provides security in the Application Layer. So, security in the Transportation Layer becomes not necessary.

23- Figure 18 also illustrates the applied filter in the 21’st question. Alternatively, IP addresses of the washington.edu are shown below which are “128.95.155.134”, “128.95.155.197” or “128.95.155.198”.

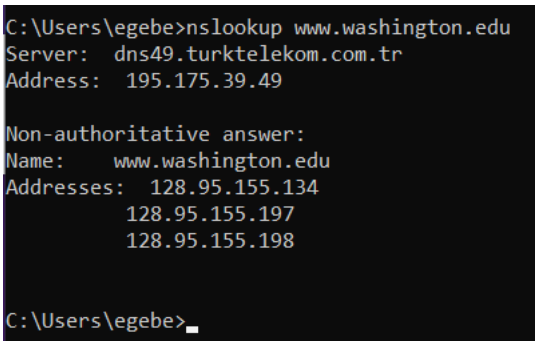


Figure 18

24- As the research says, the previous nslookup call was a recursive call which helped us to derive the local DNS Server that we have connected. So, the command which performs an iterative DNS query call is the following one “nslookup -recurse www.washington.edu”. Figure 18 illustrates the result of this DNS query.

```
C:\Users\egebe> nslookup -norecurse www.washington.edu
Server:  dns49.turktelekom.com.tr
Address:  195.175.39.49

Name:    www.washington.edu
Served by:
- hanna.cac.washington.edu
          140.142.5.5
          2607:4000:200:42::5
          washington.edu
- marge.cac.washington.edu
          140.142.5.13
          2607:4000:200:43::13
          washington.edu
- holly.s.uw.edu
          173.250.227.69
          2607:4000:301:1::69
          washington.edu
```

Figure 19

As it can be seen, the local DNS Server can be also derived from the non-recursive call. What has changed is the IP addresses of the Washington University's URL. Actually, it did not change. The difference is being non-recursive, because the name server did not go and fetch the final answer to the query. What it gave is a referral to other DNS' servers which can be helpful to reach the final answer. So, the advantage of the recursive call is reaching to the final answer faster and simpler. Also, if the queried address already exists in the cache it directly returns the answer to the client without going into DNS servers. The cached answers are stored in a limited amount of time which is called Time To Live. However, the disadvantage of the recursive calls when compared to the iterative is the security vulnerabilities. According to the research, recursive configuration makes the system vulnerable to DNS cache poisoning and DNS amplification attacks.

25- For both protocols which are DNS and UDP the header length is 20 bytes as Figure 20 shows (the screenshot is downsized due to the enormous size of the screenshot when DNS and UDP captions both appear in the screenshot).

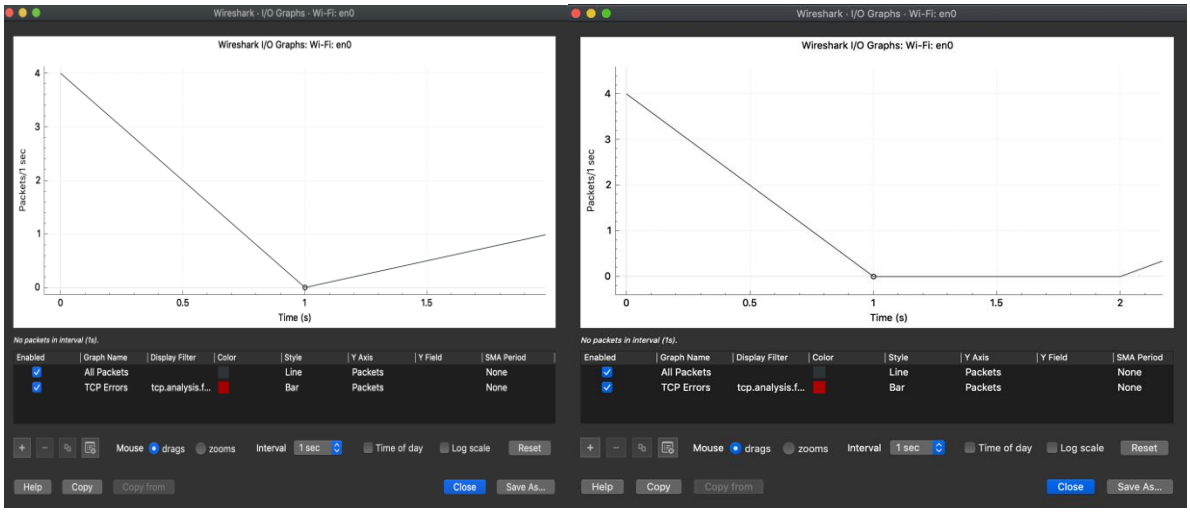
```
.... 0101 = Header Length: 20 bytes (5)
```

Figure 20

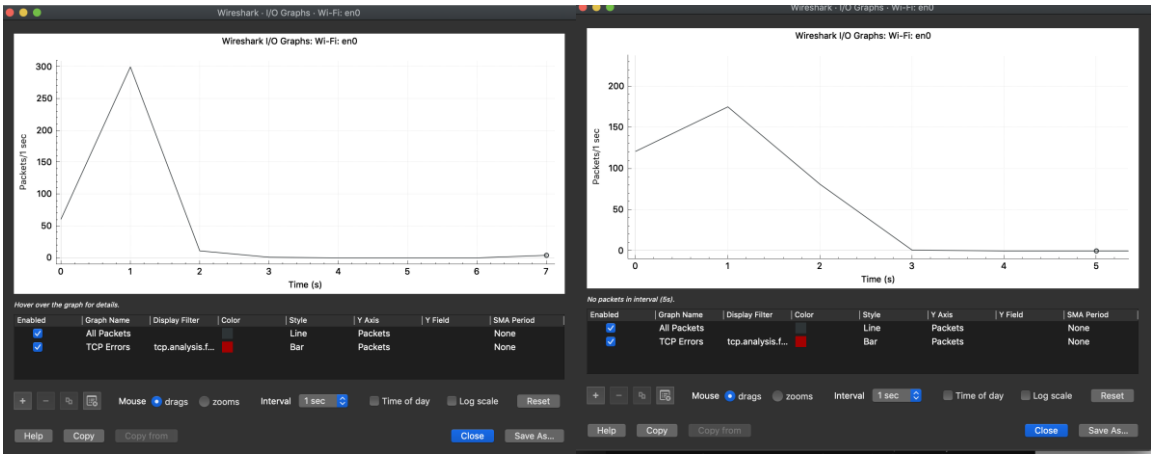
26- As far as we are concerned, UDP segment's checksum field is indicated as unverified. However, as we analyze it again and do some research, it's seen that the UDP segment has a checksum. Basically, that checksum is for the detection of corruption and error about the data. Unfortunately, even though there is a checksum, UDP is still unreliable since it does not guarantee the concession and avoid the unawared expiration.

Part 4 – Experiments for Two Competing Streams:

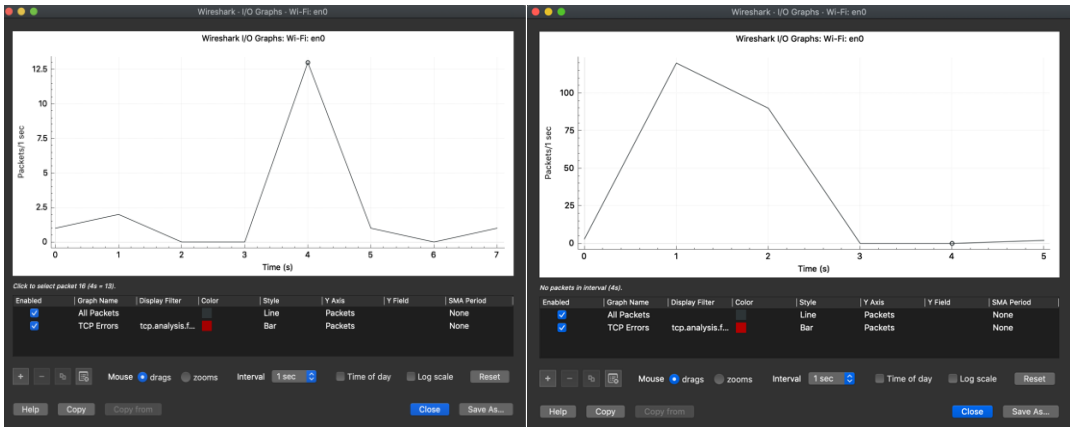
1- In this part of the experiment, two simultaneous lookup requests are sent. In that sense, UDP versus UDP case is observed. As UDP has no congestion control system, both of the requests reply with the maximum number of packages at the start. In addition, from our observations, we see that UDP also has no flow control. To be more specific, as no mechanism is there to control the data from the sender, as the package per second decreases, traffic decreases dramatically. The figure below illustrates the examples of the graphs that we obtained.



2- In that part, two simultaneous URL query sent, and TCP versus TCP case is analyzed. As one can see, since there is a congestion window, there is always a maximum amount of package delivery in total. To be more specific, congestion control mechanism optimizes the traffic in general. And as a result of that mechanism, rather than beginning with maximum package amount, deliveries begin from a particular point and then an increase is observed in that traffic. In addition, with the help of the flow control, delivery amount decreases after observing a peak, to control the data from the sender. The figure below shows the examples of the graphs that we obtained.



3- In the last part, DNS lookup and URL query both sent and UDP versus TCP is analyzed. From our observations, we see that UDP has less package delivery. In the first graph, first peak is the UDP, the second one is the TCP (There is a one-second interval between the DNS lookup and URL query, the difference is obvious). In the second graph below, the DNS lookup and URL query sent almost at the same time. The traffic increases slightly in TCP because of the number of packets sent at the same time.



Conclusion:

Throughout the project, the transport layer of the network protocol stack was the main issue. To have better focus on the transport protocols, some experiments have been done. In those experiments, we had a chance to observe the TCP, UDP and SSL protocols and their behaviors. With implementation and research experiments over TCP and SSL, the security issues and how one can handle those issues are analyzed. In addition, via Wireshark, TCP and

UDP aimed experiments have been done. In those experiments, packet ad delivery operations are taken as a primary focus and those protocols fundamentals are observed. Last and the foremost, two competing streams and the responses in that competition on resources are also another matter of fact. In that sense, we had a chance to see the flow control and congestion mechanism of TCP and their lack on UDP.

Task Distrubtion:

- Ege Berk Süzgen; 1, 2a, 3
- Emir Atışay; 1,2b, 4
- Onur Şahin; 2a, 2b, 4