Part C - HTTP Analysis

tcpdump

Note: I ran tcpdump commands on an Ubuntu machine to capture packets from Mozilla Firefox as the browser.

Commands to capture the packets:

- sudo tcpdump -i any -nn "port 1080" -w http_1080.pcap
- sudo tcpdump -i any -nn "port 1081" -w http_1081.pcap
- sudo tcpdump -i any -nn "port 1082" -w http_1082.pcap

High level analysis

I use the same information from the `analysis_pcap_tcp.py` file and enhance it to parse HTTP data. HTTP data (headers and payload) is just the TCP payload. Therefore, using the offsets described in part A, I am able to get the offset of the TCP payload and by extracting the first 4 letters of the TCP payload, I am able to identify whether the transaction is either a request or a response.

Reassemble HTTP Packets

The first 3 letters of a request is `GET`. Using this information, I identified GET requests. The first 4 letters of a response were `HTTP`. Using this information, I identified HTTP responses.

The sender sends requests with a sequence number and the corresponding response is received with ack number same as the sequence number of the sent message. Using this information, I was able to segregate the request and response pairs. I maintained a mapping of the sent sequence numbers and I mapped every response having the ack number the same as the sequence number to form the request response pair.

Here's the output for `http_1080.pcap`:

```
Request: SrcIP: 172.24.21.13, DestIP: 34.193.77.105, SeqNo: 151958699, Ack: 447271783
Response: SrcIP: 34.193.77.105, DestIP: 172.24.21.13, SeqNo: 447271783, Ack: 151959055
Request: SrcIP: 172.24.21.13, DestIP: 34.193.77.105, SeqNo: 3438422312, Ack:
```

```
1802131887
Response: SrcIP: 34.193.77.105, DestIP: 172.24.21.13, SeqNo: 1802131887, Ack:
3438422637
Request: SrcIP: 172.24.21.13, DestIP: 34.193.77.105, SeqNo: 473546070, Ack:
1817067414
Response: SrcIP: 34.193.77.105, DestIP: 172.24.21.13, SeqNo: 1817067414, Ack:
473546370
Request: SrcIP: 172.24.21.13, DestIP: 34.193.77.105, SeqNo: 2123919522, Ack:
2484582999
Response: SrcIP: 34.193.77.105, DestIP: 172.24.21.13, SeqNo: 2484582999, Ack:
2123919848
Request: SrcIP: 172.24.21.13, DestIP: 34.193.77.105, SeqNo: 1297494830, Ack:
23553853
Response: SrcIP: 34.193.77.105, DestIP: 172.24.21.13, SeqNo: 23553853, Ack:
1297495156
Request: SrcIP: 172.24.21.13, DestIP: 34.193.77.105, SeqNo: 2244422714, Ack:
3913508654
Response: SrcIP: 34.193.77.105, DestIP: 172.24.21.13, SeqNo: 3913508654, Ack:
2244423044
Request: SrcIP: 172.24.21.13, DestIP: 34.193.77.105, SeqNo: 789573452, Ack:
1619839084
Response: SrcIP: 34.193.77.105, DestIP: 172.24.21.13, SeqNo: 1619839084, Ack:
789573788
Request: SrcIP: 172.24.21.13, DestIP: 34.193.77.105, SeqNo: 2744447402, Ack:
3084454584
Response: SrcIP: 34.193.77.105, DestIP: 172.24.21.13, SeqNo: 3084454584, Ack:
2744447736
Request: SrcIP: 172.24.21.13, DestIP: 34.193.77.105, SeqNo: 4049386016, Ack:
1758933874
Response: SrcIP: 34.193.77.105, DestIP: 172.24.21.13, SeqNo: 1758933874, Ack:
4049386347
Request: SrcIP: 172.24.21.13, DestIP: 34.193.77.105, SeqNo: 1571885932, Ack:
2410021199
Response: SrcIP: 34.193.77.105, DestIP: 172.24.21.13, SeqNo: 2410021199, Ack:
1571886261
Request: SrcIP: 172.24.21.13, DestIP: 34.193.77.105, SeqNo: 2167213727, Ack:
1034848060
Response: SrcIP: 34.193.77.105, DestIP: 172.24.21.13, SeqNo: 1034848060, Ack:
2167214058
Request: SrcIP: 172.24.21.13, DestIP: 34.193.77.105, SeqNo: 3301583612, Ack:
```

```
3478500650
Response: SrcIP: 34.193.77.105, DestIP: 172.24.21.13, SeqNo: 3478500650, Ack:
3301583941
Request: SrcIP: 172.24.21.13, DestIP: 34.193.77.105, SeqNo: 1132742101, Ack:
1963441166
Response: SrcIP: 34.193.77.105, DestIP: 172.24.21.13, SeqNo: 1963441166, Ack:
1132742431
Request: SrcIP: 172.24.21.13, DestIP: 34.193.77.105, SeqNo: 770301417, Ack:
651841260
Response: SrcIP: 34.193.77.105, DestIP: 172.24.21.13, SeqNo: 651841260, Ack:
770301748
Request: SrcIP: 172.24.21.13, DestIP: 34.193.77.105, SeqNo: 3307960407, Ack:
817140690
Response: SrcIP: 34.193.77.105, DestIP: 172.24.21.13, SeqNo: 817140690, Ack:
3307960736
Request: SrcIP: 172.24.21.13, DestIP: 34.193.77.105, SeqNo: 934353919, Ack:
1912246562
Response: SrcIP: 34.193.77.105, DestIP: 172.24.21.13, SeqNo: 1912246562, Ack:
934354245
Request: SrcIP: 172.24.21.13, DestIP: 34.193.77.105, SeqNo: 3668535727, Ack:
2258022175
Response: SrcIP: 34.193.77.105, DestIP: 172.24.21.13, SeqNo: 2258022175, Ack:
3668536053
Request: SrcIP: 172.24.21.13, DestIP: 34.193.77.105, SeqNo: 2898022901, Ack:
4246593919
Response: SrcIP: 34.193.77.105, DestIP: 172.24.21.13, SeqNo: 4246593919, Ack:
2898023223
```

The request and response could not be deciphered for ports 1081 and 1082 due to encrypted TCP payload.

HTTP Protocol

The payload for the ports 1081 and 1082 was encrypted. Therefore, no analysis could be done on those packets to determine the protocol.

Instead as mentioned in the assignment, I used the knowledge of HTTP and TCP to determine the HTTP version. I used the number of TCP connections as a parameter to decide the version of the HTTP Protocol.

The following table shows the number of TCP connections for each of the pcap files. I counted the number of SYN packets (which matched with SYN ACK packets) to determine the number of TCP Connections.

PCAP File	Number of TCP Connections
1080	18
1081	6
1082	3

Port 1080

In this case, the number of TCP connections were 18. Since one connection is opened to transfer every object, I deduced the HTTP protocol version is **HTTP/1.0.**

My explanation for 18 connections

Going by intuition, the ideal number of connections would have been 17 (index + 16 images). However, one image was not loading. This got me thinking of that extra connection.

Therefore, I decided to dig deep into the browser (Firefox in my case) to analyze what was happening. In the network tab, I was able to see the **GET** requests fired to the server except for the **pear** image which was fired to another server **sbunetsooslabs.com** and the corresponding response was not obtained. Out of curiousity, I decided to check the IP of this server using the dig tool. Surprisingly, the dig tool didn't return any A records. I even used my own DNS Resolver (first assignment) to see if it was synonymous with the dig tool. My DNS Resolver returned a SOA response. Therefore, I was now convinced that the connection to this *different* server was not possible since its IP was unresolvable.

Therefore, now going by the valid number of images and index.html, the number of connections should have been = 15 + 1 = 16. There were 2 more connections that were being formed. On looking closely at the network calls, I found out that 2 more web objects were being fetched from the server. Those were a `hello.js` object and a `favicon.ico` object. This explained 18 connections.

Port 1081

The number of connections that were formed for this port = 6. The browser formed 6 TCP connections to fetch 18 web objects. This is the maximum number of parallel connections that browsers form in HTTP/1.1.

Hence, I deduced that this port was using protocol **HTTP/1.1**

Port 1082

In this case, the number of connections = 3. This is not ideally the number of connections in HTTP/2.0 but the number of connections is fairly reduced which is the motivation behind HTTP/2.0

Hence, I deduced that the version of HTTP protocol for this port is **HTTP/2.0**

Metrics analysis

Load time

I computed the page load time by subtracting the max timestamp of a packet from the min timestamp.

Number of packets

The total number of packets captured using `tcpdump` determined the number of packets exchanged. Hence, this was just the number of entries parsed in the pcap file.

Bytes transferred

I added up the packet sizes to calculate the sum total of the bytes transferred between sender and receiver.

The following table shows the computed values of load times, number of packets and total bytes sent:

HTTP Version	Load time	No of packets	Total bytes sent
HTTP/1.0	1.943958044052124 seconds	2745	2334841 bytes
HTTP/1.1	0.9973759651184082 seconds	2338	2323672 bytes
HTTP/2.0	1.0110599994659424 seconds	2358	2350170 bytes

Analysis:

Metric	Most	Least
Load time	HTTP/1.0	HTTP/1.1 and HTTP/2.0 are comparable
Number of packets	HTTP/1.0	HTTP/1.1 and HTTP/2.0 are comparable

Metric	Most	Least
Bytes transferred	HTTP/2.0	HTTP/1.1

Comments:

- The highest load time is for HTTP/1.0 which is on expected lines. Since it is opening a TCP connection for every object, it is expected to be the slowest.
- The number of packets is comparable and there is not much to distinguish between HTTP/1.1 and HTTP/2.0
- The total bytes transferred is largest for HTTP/2.0. This may be due to server push feature where unnecessary bytes are being transferred from the server to the client representing server's intention to cache at the client. As from the below table, the bytes from the server is the most in HTTP/2.0.
- Since the bytes transferred for HTTP/1.0 will also be expected to be large due to a large number of control packets (SYN, SYN ACK and FIN for every connection), hence it is observed that the bytes for HTTP/1.1 is the lowest.

HTTP Version	Bytes transferred from server
1080	2252000
1081	2260914
1082	2294581