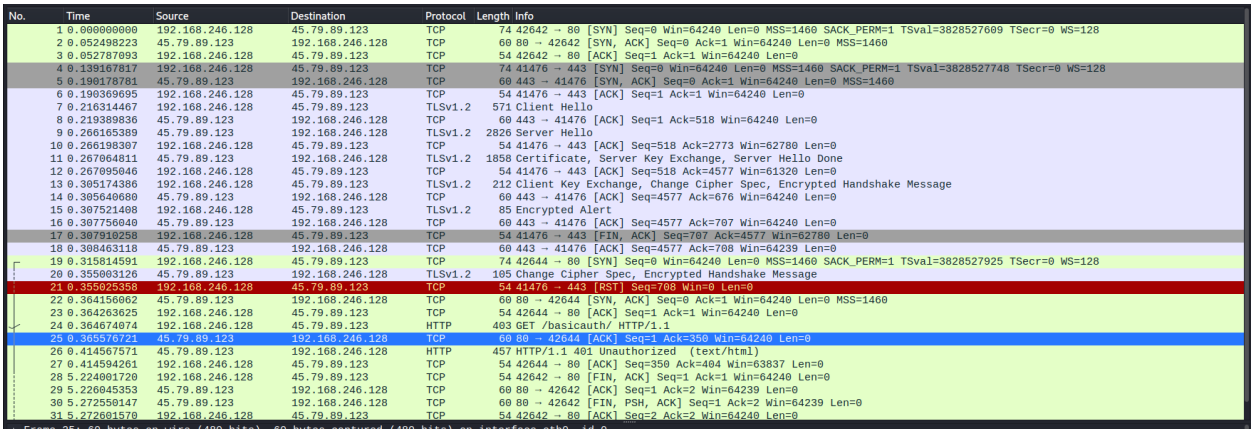
Minh Pham

Authorization is the process to determine whether one can do a certain thing. Authentication is the process to determine who someone's identity is.

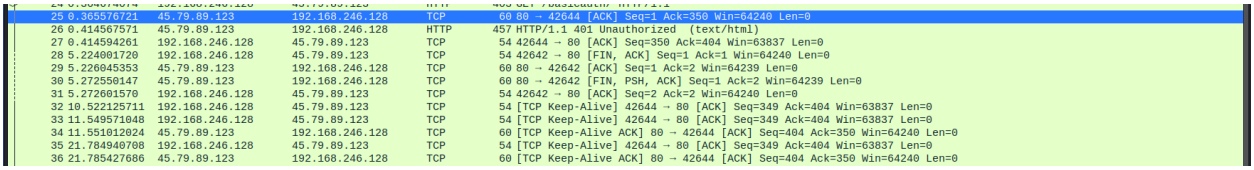
Firstly, let's go the timeline of the interaction between the browser and the server.

When the browser first accesses the website, the first 25 frames appear.



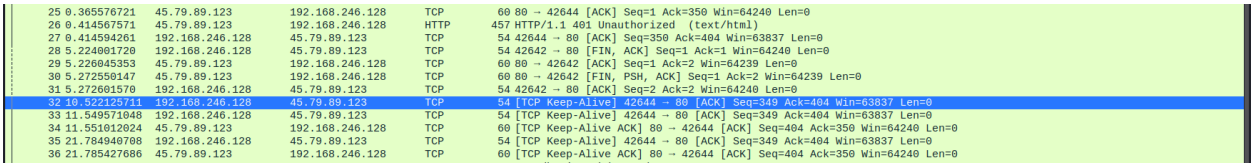
No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	192.168.246.128	45.79.89.123	TCP	74	42642 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=3828527609 TSecr=0 WS=128
2	0.052498223	45.79.89.123	192.168.246.128	TCP	60	80 → 42642 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
3	0.052787093	192.168.246.128	45.79.89.123	TCP	54	42642 → 80 [ACK] Seq=1 Ack=1 Win=64240 Len=0
4	0.139167817	192.168.246.128	45.79.89.123	TCP	74	41476 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=3828527748 TSecr=0 WS=128
5	0.190178781	45.79.89.123	192.168.246.128	TCP	60	443 → 41476 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
6	0.190369695	192.168.246.128	45.79.89.123	TCP	54	41476 → 443 [ACK] Seq=1 Ack=1 Win=64240 Len=0
7	0.216314467	192.168.246.128	45.79.89.123	TLSv1.2	571	Client Hello
8	0.219389836	45.79.89.123	192.168.246.128	TCP	60	443 → 41476 [ACK] Seq=1 Ack=518 Win=64240 Len=0
9	0.266165389	45.79.89.123	192.168.246.128	TLSv1.2	2826	Server Hello
10	0.266198307	192.168.246.128	45.79.89.123	TCP	54	41476 → 443 [ACK] Seq=518 Ack=2773 Win=62780 Len=0
11	0.267064811	45.79.89.123	192.168.246.128	TLSv1.2	1858	Certificate, Server Key Exchange, Server Hello Done
12	0.267095046	192.168.246.128	45.79.89.123	TCP	54	41476 → 443 [ACK] Seq=518 Ack=4577 Win=61320 Len=0
13	0.305174386	192.168.246.128	45.79.89.123	TLSv1.2	212	Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
14	0.305640680	45.79.89.123	192.168.246.128	TCP	60	443 → 41476 [ACK] Seq=4577 Ack=676 Win=64240 Len=0
15	0.307521408	192.168.246.128	45.79.89.123	TLSv1.2	85	Encrypted Alert
16	0.307756040	45.79.89.123	192.168.246.128	TCP	60	443 → 41476 [ACK] Seq=4577 Ack=707 Win=64240 Len=0
17	0.307910258	192.168.246.128	45.79.89.123	TCP	54	41476 → 443 [FIN, ACK] Seq=707 Ack=4577 Win=62780 Len=0
18	0.308463118	45.79.89.123	192.168.246.128	TCP	60	443 → 41476 [ACK] Seq=4577 Ack=708 Win=64239 Len=0
19	0.315814591	192.168.246.128	45.79.89.123	TCP	74	42644 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=3828527925 TSecr=0 WS=128
20	0.355003126	45.79.89.123	192.168.246.128	TLSv1.2	105	Change Cipher Spec, Encrypted Handshake Message
21	0.355025358	192.168.246.128	45.79.89.123	TCP	54	41476 → 443 [RST] Seq=708 Win=0 Len=0
22	0.364156062	45.79.89.123	192.168.246.128	TCP	60	80 → 42644 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
23	0.364263625	192.168.246.128	45.79.89.123	TCP	54	42644 → 80 [ACK] Seq=1 Ack=1 Win=64240 Len=0
24	0.364674074	192.168.246.128	45.79.89.123	HTTP	403	GET /basicauth/ HTTP/1.1
25	0.365576721	45.79.89.123	192.168.246.128	TCP	60	80 → 42644 [ACK] Seq=1 Ack=350 Win=64240 Len=0
26	0.414567571	45.79.89.123	192.168.246.128	HTTP	457	HTTP/1.1 401 Unauthorized (text/html)
27	0.414594261	192.168.246.128	45.79.89.123	TCP	54	42644 → 80 [ACK] Seq=350 Ack=404 Win=63837 Len=0
28	5.224001720	192.168.246.128	45.79.89.123	TCP	54	42642 → 80 [FIN, ACK] Seq=1 Ack=1 Win=64240 Len=0
29	5.226045353	45.79.89.123	192.168.246.128	TCP	60	80 → 42642 [ACK] Seq=1 Ack=2 Win=64239 Len=0
30	5.272550147	45.79.89.123	192.168.246.128	TCP	60	80 → 42642 [FIN, PSH, ACK] Seq=1 Ack=2 Win=64239 Len=0
31	5.272601570	192.168.246.128	45.79.89.123	TCP	54	42642 → 80 [ACK] Seq=2 Ack=2 Win=64240 Len=0

After 3 seconds, frame 25 to frame 31 appear.



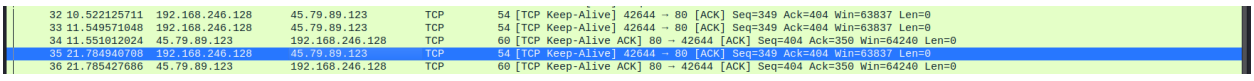
25	0.365576721	45.79.89.123	192.168.246.128	TCP	60	80 → 42644 [ACK] Seq=1 Ack=350 Win=64240 Len=0
26	0.414567571	45.79.89.123	192.168.246.128	HTTP	457	HTTP/1.1 401 Unauthorized (text/html)
27	0.414594261	192.168.246.128	45.79.89.123	TCP	54	42644 → 80 [ACK] Seq=350 Ack=404 Win=63837 Len=0
28	5.224001720	192.168.246.128	45.79.89.123	TCP	54	42642 → 80 [FIN, ACK] Seq=1 Ack=1 Win=64240 Len=0
29	5.226045353	45.79.89.123	192.168.246.128	TCP	60	80 → 42642 [ACK] Seq=1 Ack=2 Win=64239 Len=0
30	5.272550147	45.79.89.123	192.168.246.128	TCP	60	80 → 42642 [FIN, PSH, ACK] Seq=1 Ack=2 Win=64239 Len=0
31	5.272601570	192.168.246.128	45.79.89.123	TCP	54	42642 → 80 [ACK] Seq=2 Ack=2 Win=64240 Len=0
32	10.522125711	192.168.246.128	45.79.89.123	TCP	54	[TCP Keep-Alive] 42644 → 80 [ACK] Seq=349 Ack=404 Win=63837 Len=0
33	11.549571048	192.168.246.128	45.79.89.123	TCP	54	[TCP Keep-Alive] 42644 → 80 [ACK] Seq=349 Ack=404 Win=63837 Len=0
34	11.551012024	45.79.89.123	192.168.246.128	TCP	60	[TCP Keep-Alive ACK] 80 → 42644 [ACK] Seq=404 Ack=350 Win=64240 Len=0
35	21.784940708	192.168.246.128	45.79.89.123	TCP	54	[TCP Keep-Alive] 42644 → 80 [ACK] Seq=349 Ack=404 Win=63837 Len=0
36	21.785427686	45.79.89.123	192.168.246.128	TCP	60	[TCP Keep-Alive ACK] 80 → 42644 [ACK] Seq=404 Ack=350 Win=64240 Len=0

After another 3 seconds, frame 32 to 35 appear.



25	0.365576721	45.79.89.123	192.168.246.128	TCP	60	80 → 42644 [ACK] Seq=1 Ack=350 Win=64240 Len=0
26	0.414567571	45.79.89.123	192.168.246.128	HTTP	457	HTTP/1.1 401 Unauthorized (text/html)
27	0.414594261	192.168.246.128	45.79.89.123	TCP	54	42644 → 80 [ACK] Seq=350 Ack=404 Win=63837 Len=0
28	5.224001720	192.168.246.128	45.79.89.123	TCP	54	42642 → 80 [FIN, ACK] Seq=1 Ack=1 Win=64240 Len=0
29	5.226045353	45.79.89.123	192.168.246.128	TCP	60	80 → 42642 [ACK] Seq=1 Ack=2 Win=64239 Len=0
30	5.272550147	45.79.89.123	192.168.246.128	TCP	60	80 → 42642 [FIN, PSH, ACK] Seq=1 Ack=2 Win=64239 Len=0
31	5.272601570	192.168.246.128	45.79.89.123	TCP	54	42642 → 80 [ACK] Seq=2 Ack=2 Win=64240 Len=0
32	10.522125711	192.168.246.128	45.79.89.123	TCP	54	[TCP Keep-Alive] 42644 → 80 [ACK] Seq=349 Ack=404 Win=63837 Len=0
33	11.549571048	192.168.246.128	45.79.89.123	TCP	54	[TCP Keep-Alive] 42644 → 80 [ACK] Seq=349 Ack=404 Win=63837 Len=0
34	11.551012024	45.79.89.123	192.168.246.128	TCP	60	[TCP Keep-Alive ACK] 80 → 42644 [ACK] Seq=404 Ack=350 Win=64240 Len=0
35	21.784940708	192.168.246.128	45.79.89.123	TCP	54	[TCP Keep-Alive] 42644 → 80 [ACK] Seq=349 Ack=404 Win=63837 Len=0
36	21.785427686	45.79.89.123	192.168.246.128	TCP	60	[TCP Keep-Alive ACK] 80 → 42644 [ACK] Seq=404 Ack=350 Win=64240 Len=0

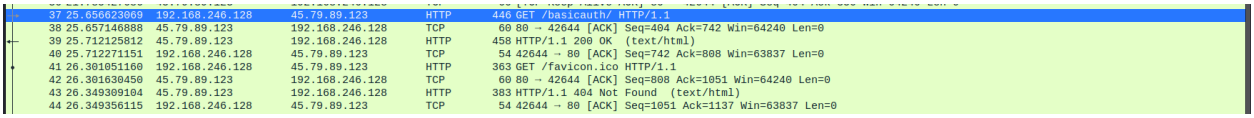
From then on, copies of the frames 35 and 36 appear as a pair after every few seconds.



32	10.522125711	192.168.246.128	45.79.89.123	TCP	54	[TCP Keep-Alive] 42644 → 80 [ACK] Seq=349 Ack=404 Win=63837 Len=0
33	11.549571048	192.168.246.128	45.79.89.123	TCP	54	[TCP Keep-Alive] 42644 → 80 [ACK] Seq=349 Ack=404 Win=63837 Len=0
34	11.551012024	45.79.89.123	192.168.246.128	TCP	60	[TCP Keep-Alive ACK] 80 → 42644 [ACK] Seq=404 Ack=350 Win=64240 Len=0
35	21.784940708	192.168.246.128	45.79.89.123	TCP	54	[TCP Keep-Alive] 42644 → 80 [ACK] Seq=349 Ack=404 Win=63837 Len=0
36	21.785427686	45.79.89.123	192.168.246.128	TCP	60	[TCP Keep-Alive ACK] 80 → 42644 [ACK] Seq=404 Ack=350 Win=64240 Len=0

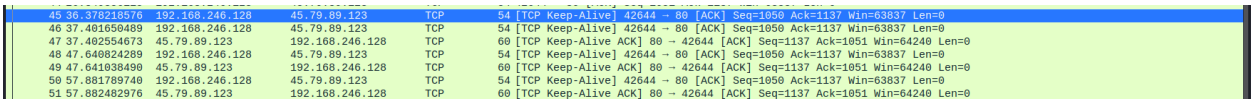
The browser and the server are communicating with each other here, asking one another to keep the page alive, until the authentication is successful.

When that happens, these frames from 37 to 44 appear:



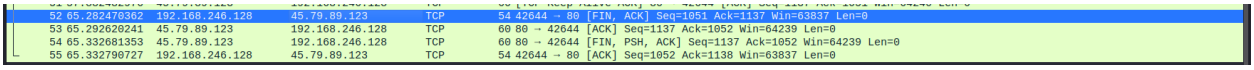
37	25.656623069	192.168.246.128	45.79.89.123	HTTP	446	GET /basicauth/ HTTP/1.1
38	25.657146888	45.79.89.123	192.168.246.128	TCP	60	80 → 42644 [ACK] Seq=404 Ack=742 Win=64240 Len=0
39	25.712125812	45.79.89.123	192.168.246.128	HTTP	458	HTTP/1.1 200 OK (text/html)
40	25.712271151	192.168.246.128	45.79.89.123	TCP	54	42644 → 80 [ACK] Seq=742 Ack=808 Win=63837 Len=0
41	26.301051160	192.168.246.128	45.79.89.123	HTTP	363	GET /favicon.ico HTTP/1.1
42	26.301630450	45.79.89.123	192.168.246.128	TCP	60	80 → 42644 [ACK] Seq=808 Ack=1051 Win=64240 Len=0
43	26.349309104	45.79.89.123	192.168.246.128	HTTP	383	HTTP/1.1 404 Not Found (text/html)
44	26.349356115	192.168.246.128	45.79.89.123	TCP	54	42644 → 80 [ACK] Seq=1051 Ack=1137 Win=63837 Len=0

After 3 seconds, the browser and the server once again are talking to each other to keep the page alive, as shown in frames 45 to 51 in this situation. If the page is held open longer, a pair such as the frames 50 and 51 will continually stack up.



45	36.378218576	192.168.246.128	45.79.89.123	TCP	54	[TCP Keep-Alive] 42644 → 80 [ACK] Seq=1050 Ack=1137 Win=63837 Len=0
46	37.401650489	192.168.246.128	45.79.89.123	TCP	54	[TCP Keep-Alive] 42644 → 80 [ACK] Seq=1050 Ack=1137 Win=63837 Len=0
47	37.402554673	45.79.89.123	192.168.246.128	TCP	60	[TCP Keep-Alive ACK] 80 → 42644 [ACK] Seq=1137 Ack=1051 Win=64240 Len=0
48	47.640824289	192.168.246.128	45.79.89.123	TCP	54	[TCP Keep-Alive] 42644 → 80 [ACK] Seq=1050 Ack=1137 Win=63837 Len=0
49	47.641038490	45.79.89.123	192.168.246.128	TCP	60	[TCP Keep-Alive ACK] 80 → 42644 [ACK] Seq=1137 Ack=1051 Win=64240 Len=0
50	57.881789740	192.168.246.128	45.79.89.123	TCP	54	[TCP Keep-Alive] 42644 → 80 [ACK] Seq=1050 Ack=1137 Win=63837 Len=0
51	57.882482976	45.79.89.123	192.168.246.128	TCP	60	[TCP Keep-Alive ACK] 80 → 42644 [ACK] Seq=1137 Ack=1051 Win=64240 Len=0

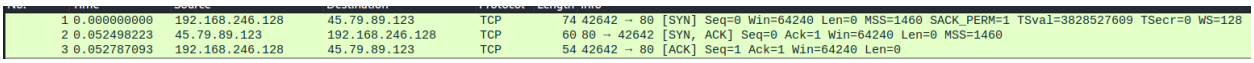
And when the website is closed, the process is finished, and frames 52 to 55 appear:



52	65.282470362	192.168.246.128	45.79.89.123	TCP	54	42644 → 80 [FIN, ACK] Seq=1051 Ack=1137 Win=63837 Len=0
53	65.292620241	45.79.89.123	192.168.246.128	TCP	60	80 → 42644 [ACK] Seq=1137 Ack=1052 Win=64239 Len=0
54	65.332681353	45.79.89.123	192.168.246.128	TCP	60	80 → 42644 [FIN, PSH, ACK] Seq=1137 Ack=1052 Win=64239 Len=0
55	65.332790727	192.168.246.128	45.79.89.123	TCP	54	42644 → 80 [ACK] Seq=1052 Ack=1138 Win=63837 Len=0

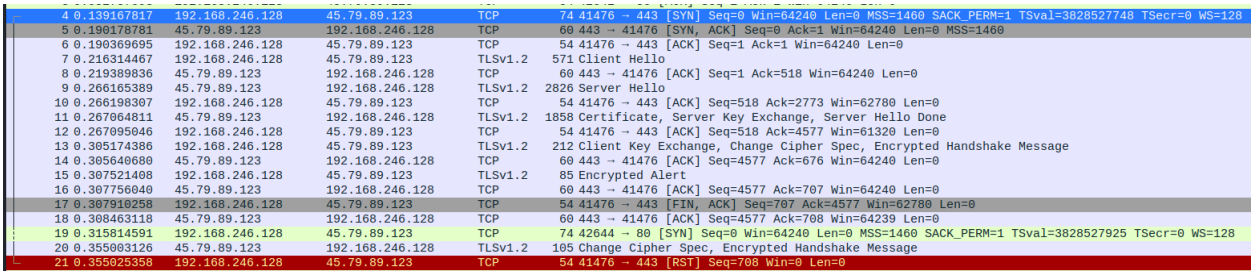
That's the timeline of how all the frames appear. In details, the whole process can be divided into these phases:

1. Using the TCP protocol, the browser requests connection from the server (frame 1 to 3)



1	0.000000000	192.168.246.128	45.79.89.123	TCP	74	42642 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=3828527609 TSecr=0 WS=128
2	0.052498223	45.79.89.123	192.168.246.128	TCP	60	80 → 42642 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
3	0.052787093	192.168.246.128	45.79.89.123	TCP	54	42642 → 80 [ACK] Seq=1 Ack=1 Win=64240 Len=0

2. Using the TCP and the TSL protocol, the browser requests the facilitation of privacy and data security to encrypt the communication between the browser and the server (frame 4 to 21)

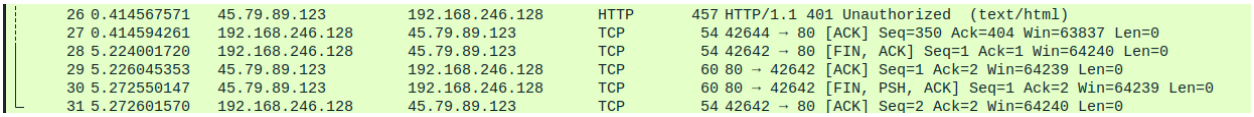


4	0.139167817	192.168.246.128	45.79.89.123	TCP	74	41476 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=3828527748 TSecr=0 WS=128
5	0.190178781	45.79.89.123	192.168.246.128	TCP	60	443 → 41476 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
6	0.190369695	192.168.246.128	45.79.89.123	TCP	54	41476 → 443 [ACK] Seq=1 Ack=1 Win=64240 Len=0
7	0.216314467	192.168.246.128	45.79.89.123	TLSv1.2	571	Client Hello
8	0.219389836	45.79.89.123	192.168.246.128	TCP	60	443 → 41476 [ACK] Seq=1 Ack=518 Win=64240 Len=0
9	0.266165389	45.79.89.123	192.168.246.128	TLSv1.2	2826	Server Hello
10	0.266198307	192.168.246.128	45.79.89.123	TCP	54	41476 → 443 [ACK] Seq=518 Ack=2773 Win=62780 Len=0
11	0.267064811	45.79.89.123	192.168.246.128	TLSv1.2	1858	Certificate, Server Key Exchange, Server Hello Done
12	0.267095046	192.168.246.128	45.79.89.123	TCP	54	41476 → 443 [ACK] Seq=518 Ack=4577 Win=61320 Len=0
13	0.305174386	192.168.246.128	45.79.89.123	TLSv1.2	212	Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
14	0.305640680	45.79.89.123	192.168.246.128	TCP	60	443 → 41476 [ACK] Seq=4577 Ack=676 Win=64240 Len=0
15	0.307521408	192.168.246.128	45.79.89.123	TLSv1.2	85	Encrypted Alert
16	0.307756040	45.79.89.123	192.168.246.128	TCP	60	443 → 41476 [ACK] Seq=4577 Ack=707 Win=64240 Len=0
17	0.307910258	192.168.246.128	45.79.89.123	TCP	54	41476 → 443 [FIN, ACK] Seq=707 Ack=4577 Win=62780 Len=0
18	0.308463118	45.79.89.123	192.168.246.128	TCP	60	443 → 41476 [ACK] Seq=4577 Ack=708 Win=64239 Len=0
19	0.315814591	192.168.246.128	45.79.89.123	TCP	74	42644 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=3828527925 TSecr=0 WS=128
20	0.355003126	45.79.89.123	192.168.246.128	TLSv1.2	105	Change Cipher Spec, Encrypted Handshake Message
21	0.355025358	192.168.246.128	45.79.89.123	TCP	54	41476 → 443 [RST] Seq=708 Win=0 Len=0

3. Using the TCP protocol and the HTTP protocol, the browser requests the server to fetch the HTML for the page (frame 22 to 25), specifically the prompt that asks users to enter username and password, as shown in frame 24, the browser requests the basic access authentication. The server acknowledges that in frame 25.

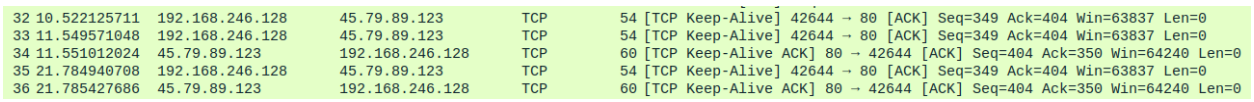
22	0.364156062	45.79.89.123	192.168.246.128	TCP	60	80 → 42644 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
23	0.364263625	192.168.246.128	45.79.89.123	TCP	54	42644 → 80 [ACK] Seq=1 Ack=1 Win=64240 Len=0
24	0.364674074	192.168.246.128	45.79.89.123	HTTP	403	GET /basicauth/ HTTP/1.1
25	0.365576721	45.79.89.123	192.168.246.128	TCP	60	80 → 42644 [ACK] Seq=1 Ack=350 Win=64240 Len=0

4. Using the TCP protocol, the browser ends the request for the server to fetch the HTML (frame 26 to 29)



26	0.414567571	45.79.89.123	192.168.246.128	HTTP	457	HTTP/1.1 401 Unauthorized (text/html)
27	0.414594261	192.168.246.128	45.79.89.123	TCP	54	42644 → 80 [ACK] Seq=350 Ack=404 Win=63837 Len=0
28	5.224001720	192.168.246.128	45.79.89.123	TCP	54	42642 → 80 [FIN, ACK] Seq=1 Ack=1 Win=64240 Len=0
29	5.226045353	45.79.89.123	192.168.246.128	TCP	60	80 → 42642 [ACK] Seq=1 Ack=2 Win=64239 Len=0
30	5.272550147	45.79.89.123	192.168.246.128	TCP	60	80 → 42642 [FIN, PSH, ACK] Seq=1 Ack=2 Win=64239 Len=0
31	5.272601570	192.168.246.128	45.79.89.123	TCP	54	42642 → 80 [ACK] Seq=2 Ack=2 Win=64240 Len=0

5. Using the TCP protocol with the TCP Keep-Alive flag, the browser requests the server to keep the page alive for as long as needed (frame 32 to 36, can go longer depending on how long the user stays on the page).



32	10.522125711	192.168.246.128	45.79.89.123	TCP	54	[TCP Keep-Alive] 42644 → 80 [ACK] Seq=349 Ack=404 Win=63837 Len=0
33	11.549571048	192.168.246.128	45.79.89.123	TCP	54	[TCP Keep-Alive] 42644 → 80 [ACK] Seq=349 Ack=404 Win=63837 Len=0
34	11.551012024	45.79.89.123	192.168.246.128	TCP	60	[TCP Keep-Alive ACK] 80 → 42644 [ACK] Seq=404 Ack=350 Win=64240 Len=0
35	21.784940708	192.168.246.128	45.79.89.123	TCP	54	[TCP Keep-Alive] 42644 → 80 [ACK] Seq=349 Ack=404 Win=63837 Len=0
36	21.785427686	45.79.89.123	192.168.246.128	TCP	60	[TCP Keep-Alive ACK] 80 → 42644 [ACK] Seq=404 Ack=350 Win=64240 Len=0

6. As the user enters in the credentials, the browser requests a GET to use HTTP basic authentication to see if users have correctly put in the right credentials (frame 37). As the server acknowledges so (frame 38), the server returns an OK using the HTTP protocol

(frame 39). The browser then proceeds to request the rest of the HTML to be fetched from the server, using the TCP and HTTP protocol (frame 41 to 44)

```
37 25.656623069  192.168.246.128  45.79.89.123     HTTP  446 GET /basicauth/ HTTP/1.1
38 25.657146888  45.79.89.123     192.168.246.128  TCP    60 80 → 42644 [ACK] Seq=404 Ack=742 Win=64240 Len=0
39 25.712125812  45.79.89.123     192.168.246.128  HTTP  458 HTTP/1.1 200 OK  (text/html)
40 25.712271151  192.168.246.128  45.79.89.123     TCP    54 42644 → 80 [ACK] Seq=742 Ack=808 Win=63837 Len=0
41 26.301051160  192.168.246.128  45.79.89.123     HTTP  363 GET /favicon.ico HTTP/1.1
42 26.301630450  45.79.89.123     192.168.246.128  TCP    60 80 → 42644 [ACK] Seq=808 Ack=1051 Win=64240 Len=0
43 26.349309104  45.79.89.123     192.168.246.128  HTTP  383 HTTP/1.1 404 Not Found  (text/html)
44 26.349356115  192.168.246.128  45.79.89.123     TCP    54 42644 → 80 [ACK] Seq=1051 Ack=1137 Win=63837 Len=0
```

7. Similar to phase 5, the browser asks the server to keep the page alive with the TCP protocol and the TCP Keep-Alive flag (frame 45 to 51). Again, this can loop as long as the user stays on the page.

```
45 36.378218576  192.168.246.128  45.79.89.123     TCP  54 [TCP Keep-Alive] 42644 → 80 [ACK] Seq=1050 Ack=1137 Win=63837 Len=0
46 37.401650489  192.168.246.128  45.79.89.123     TCP  54 [TCP Keep-Alive] 42644 → 80 [ACK] Seq=1050 Ack=1137 Win=63837 Len=0
47 37.402554673  45.79.89.123     192.168.246.128  TCP  60 [TCP Keep-Alive ACK] 80 → 42644 [ACK] Seq=1137 Ack=1051 Win=64240 Len=0
48 47.640824289  192.168.246.128  45.79.89.123     TCP  54 [TCP Keep-Alive] 42644 → 80 [ACK] Seq=1050 Ack=1137 Win=63837 Len=0
49 47.641038490  45.79.89.123     192.168.246.128  TCP  60 [TCP Keep-Alive ACK] 80 → 42644 [ACK] Seq=1137 Ack=1051 Win=64240 Len=0
50 57.881789740  192.168.246.128  45.79.89.123     TCP  54 [TCP Keep-Alive] 42644 → 80 [ACK] Seq=1050 Ack=1137 Win=63837 Len=0
51 57.882482976  45.79.89.123     192.168.246.128  TCP  60 [TCP Keep-Alive ACK] 80 → 42644 [ACK] Seq=1137 Ack=1051 Win=64240 Len=0
```

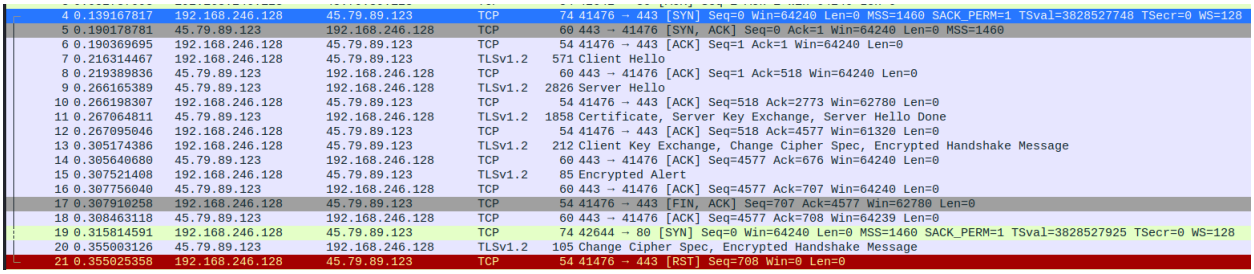
8. When the user closes the page, the TCP protocol is used once again as the browser requests the server to terminate the process, the usual way using the FIN flag (frame 52 to 55)

```
52 65.282470362  192.168.246.128  45.79.89.123     TCP  54 42644 → 80 [FIN, ACK] Seq=1051 Ack=1137 Win=63837 Len=0
53 65.292620241  45.79.89.123     192.168.246.128  TCP  60 80 → 42644 [ACK] Seq=1137 Ack=1052 Win=64239 Len=0
54 65.332681353  45.79.89.123     192.168.246.128  TCP  60 80 → 42644 [FIN, PSH, ACK] Seq=1137 Ack=1052 Win=64239 Len=0
55 65.332790727  192.168.246.128  45.79.89.123     TCP  54 42644 → 80 [ACK] Seq=1052 Ack=1138 Win=63837 Len=0
```

Out of the 8 phases above, we are familiar with all but phase 2, 3, and 6, as they are TCP and HTTP protocols that we have seen from the previous assignment. In phase 3 and 6, there is the HTTP's Basic Access Authentication model for the nginx server. An important detail is that the browser sent the input to the server to check, which they confirm to be true or not in frame 39 where they say OK through HTTP.

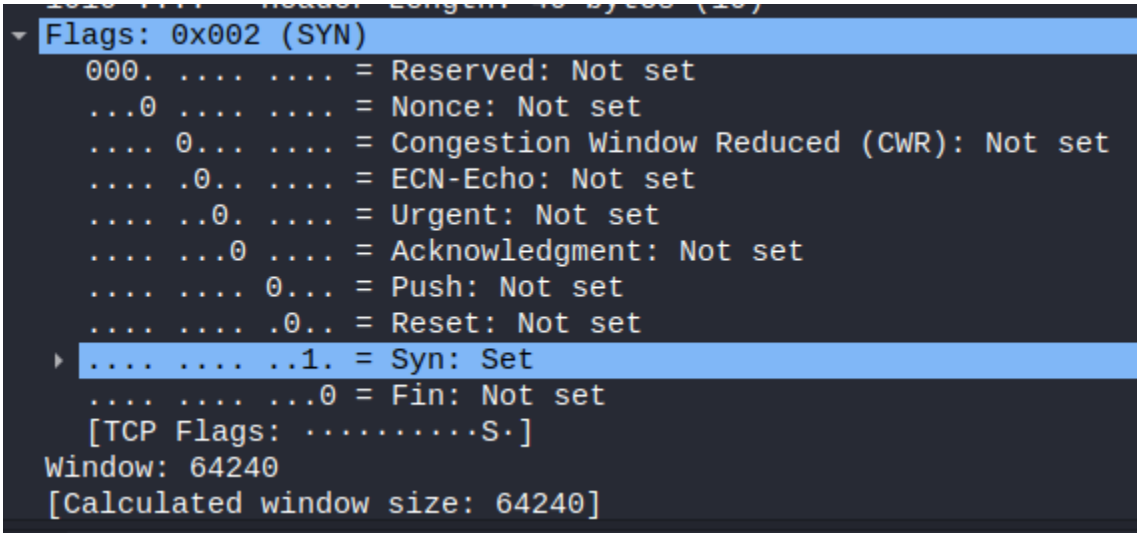
```
37 25.656623069  192.168.246.128  45.79.89.123     HTTP  446 GET /basicauth/ HTTP/1.1
38 25.657146888  45.79.89.123     192.168.246.128  TCP    60 80 → 42644 [ACK] Seq=404 Ack=742 Win=64240 Len=0
39 25.712125812  45.79.89.123     192.168.246.128  HTTP  458 HTTP/1.1 200 OK  (text/html)
40 25.712271151  192.168.246.128  45.79.89.123     TCP    54 42644 → 80 [ACK] Seq=742 Ack=808 Win=63837 Len=0
```

The passwords are sent encrypted, and the encryption key is provided through TLS handshake in phase 2, which we mentioned that we haven't discussed in the last assignment. Recall these frames constitute phase 2:



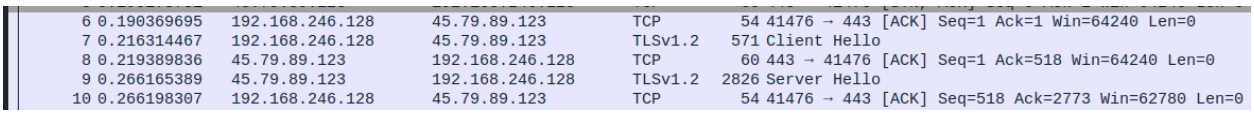
4	0.139167817	192.168.246.128	45.79.89.123	TCP	74 41476 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=3828527748 TSecr=0 WS=128
5	0.190178781	45.79.89.123	192.168.246.128	TCP	60 443 → 41476 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
6	0.190369695	192.168.246.128	45.79.89.123	TCP	54 41476 → 443 [ACK] Seq=1 Ack=1 Win=64240 Len=0
7	0.216314467	192.168.246.128	45.79.89.123	TLSv1.2	571 Client Hello
8	0.219389836	45.79.89.123	192.168.246.128	TCP	60 443 → 41476 [ACK] Seq=1 Ack=518 Win=64240 Len=0
9	0.266165389	45.79.89.123	192.168.246.128	TLSv1.2	2826 Server Hello
10	0.266198307	192.168.246.128	45.79.89.123	TCP	54 41476 → 443 [ACK] Seq=518 Ack=2773 Win=62780 Len=0
11	0.267064811	45.79.89.123	192.168.246.128	TLSv1.2	1858 Certificate, Server Key Exchange, Server Hello Done
12	0.267095046	192.168.246.128	45.79.89.123	TCP	54 41476 → 443 [ACK] Seq=518 Ack=4577 Win=61320 Len=0
13	0.305174386	192.168.246.128	45.79.89.123	TLSv1.2	212 Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
14	0.305640680	45.79.89.123	192.168.246.128	TCP	60 443 → 41476 [ACK] Seq=4577 Ack=676 Win=64240 Len=0
15	0.307521408	192.168.246.128	45.79.89.123	TLSv1.2	85 Encrypted Alert
16	0.307756040	45.79.89.123	192.168.246.128	TCP	60 443 → 41476 [ACK] Seq=4577 Ack=707 Win=64240 Len=0
17	0.307910258	192.168.246.128	45.79.89.123	TCP	54 41476 → 443 [FIN, ACK] Seq=707 Ack=4577 Win=62780 Len=0
18	0.308463118	45.79.89.123	192.168.246.128	TCP	60 443 → 41476 [ACK] Seq=4577 Ack=708 Win=64239 Len=0
19	0.315814591	192.168.246.128	45.79.89.123	TCP	74 42644 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=3828527925 TSecr=0 WS=128
20	0.355003126	45.79.89.123	192.168.246.128	TLSv1.2	105 Change Cipher Spec, Encrypted Handshake Message
21	0.355025358	192.168.246.128	45.79.89.123	TCP	54 41476 → 443 [RST] Seq=708 Win=0 Len=0

Upon further inspection of frame 4, we see that in this frame, the client is requesting a new TCP connection with the server based on the flags:



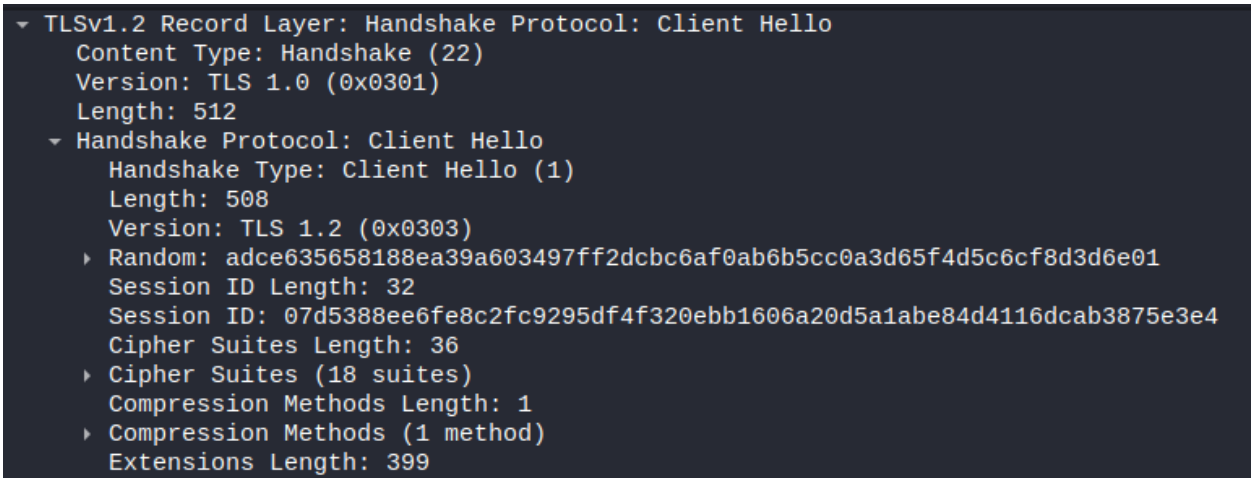
```
Flags: 0x002 (SYN)
    000. .... .... = Reserved: Not set
    ...0 .... .... = Nonce: Not set
    .... 0... .... = Congestion Window Reduced (CWR): Not set
    .... .0.. .... = ECN-Echo: Not set
    .... ..0. .... = Urgent: Not set
    .... ...0 .... = Acknowledgment: Not set
    .... .... 0... = Push: Not set
    .... .... .0.. = Reset: Not set
  ▸ .... .... ..1. = Syn: Set
    .... .... ...0 = Fin: Not set
    [TCP Flags: ··········S·]
  Window: 64240
  [Calculated window size: 64240]
```

This connection will be used for TLS handshake. The server responds with SYN ACK in frame 5 per usual. They will proceed with these frames:

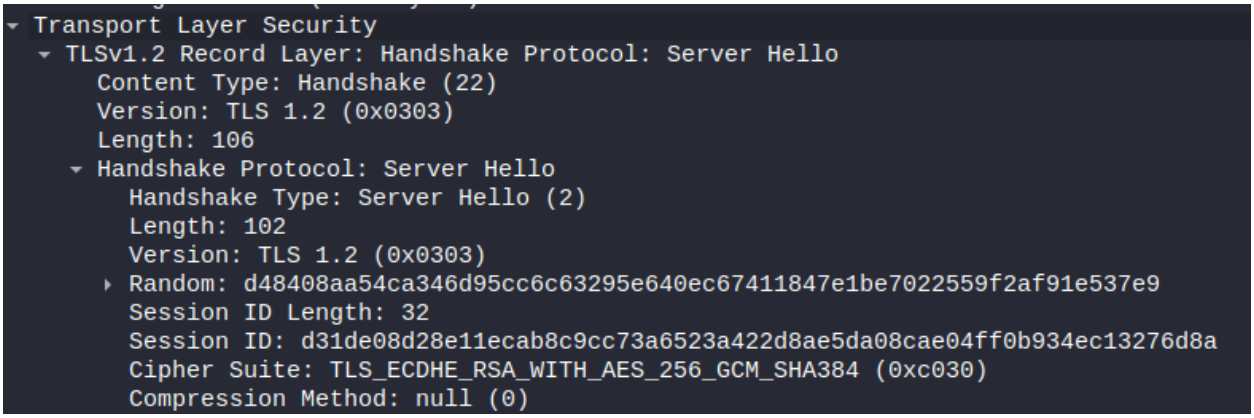


6	0.190369695	192.168.246.128	45.79.89.123	TCP	54 41476 → 443 [ACK] Seq=1 Ack=1 Win=64240 Len=0
7	0.216314467	192.168.246.128	45.79.89.123	TLSv1.2	571 Client Hello
8	0.219389836	45.79.89.123	192.168.246.128	TCP	60 443 → 41476 [ACK] Seq=1 Ack=518 Win=64240 Len=0
9	0.266165389	45.79.89.123	192.168.246.128	TLSv1.2	2826 Server Hello
10	0.266198307	192.168.246.128	45.79.89.123	TCP	54 41476 → 443 [ACK] Seq=518 Ack=2773 Win=62780 Len=0

In frame 6, the browser responds to the SYN ACK of frame 5 with an ACK per usual. In frame 7, the browser sends to the server through TLS, saying “Client Hello”. The response looks like this in detail:



We have the random**:** which is used to calculate the Master secret for the creation of the encryption key. We have the session ID for this specific handshake session. And in frame 8, the browser ACK, then in frame 9, the server sends back through TLS, saying “Server Hello”, which looks like this in detail. We also have the random here and the session ID.



In frame 10, the browser acknowledges that. The TLS Handshake is established. They proceed with these frames:

11	0.267064811	45.79.89.123	192.168.246.128	TLSv1.2	1858	Certificate, Server Key Exchange, Server Hello Done
12	0.267095046	192.168.246.128	45.79.89.123	TCP	54	41476 → 443 [ACK] Seq=518 Ack=4577 Win=61320 Len=0
13	0.305174386	192.168.246.128	45.79.89.123	TLSv1.2	212	Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
14	0.305640680	45.79.89.123	192.168.246.128	TCP	60	443 → 41476 [ACK] Seq=4577 Ack=676 Win=64240 Len=0
15	0.307521408	192.168.246.128	45.79.89.123	TLSv1.2	85	Encrypted Alert
16	0.307756040	45.79.89.123	192.168.246.128	TCP	60	443 → 41476 [ACK] Seq=4577 Ack=707 Win=64240 Len=0

In frame 11, the server generates session keys for encrypting messages between them after the handshake is complete. The server uses a public key to prove its identity to the client. Note that anyone with the public key can unscramble the data to ensure its authenticity, but not only the original sender, which is the server in this case, can encrypt such data with their secret private key. In frame 13, the browser takes their turn in the exchange of the key and unscrambles the encrypted handshake message and defines changes in ciphering strategies. In frame 15, the server responds with the encrypted alert, which is for if there is any error that has occurred in the exchange process.

16	0.307756040	45.79.89.123	192.168.246.128	TCP	60	443 → 41476 [ACK] Seq=4577 Ack=707 Win=64240 Len=0
17	0.307910258	192.168.246.128	45.79.89.123	TCP	54	41476 → 443 [FIN, ACK] Seq=707 Ack=4577 Win=62780 Len=0
18	0.308463118	45.79.89.123	192.168.246.128	TCP	60	443 → 41476 [ACK] Seq=4577 Ack=708 Win=64239 Len=0

Frame 16 to 18 finishes the connection established earlier in frame 4 and 5, the usual TCP way.

We now have discussed thoroughly every of the 8 phases that we outlined. This is the story of the interaction between the browser and the client, with basic access authentication.

Citations

https://docs.nginx.com/nginx/admin-guide/security-controls/configuring-http-basic-authentication/

https://www.cloudflare.com/learning/ssl/transport-layer-security-tls/

https://www.catchpoint.com/blog/wireshark-tls-handshake