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.

Table

Description automatically generated with medium confidence

After 3 seconds, frame 25 to frame 31 appear.

A picture containing text

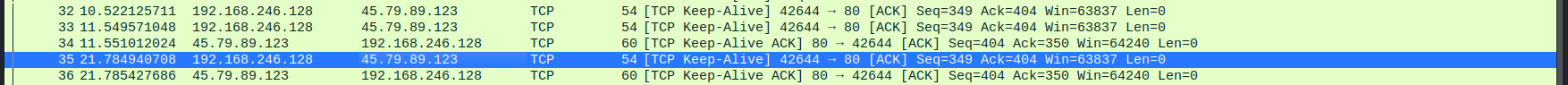
Description automatically generated

After another 3 seconds, frame 32 to 35 appear.

Calendar

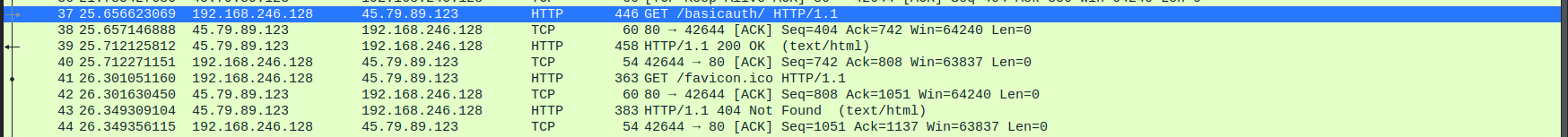
Description automatically generated

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

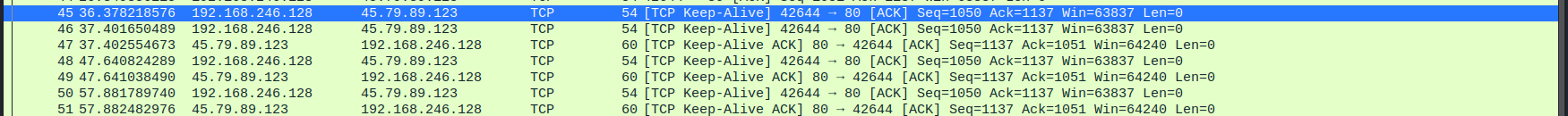


The browser and the server are communicating with each other here, asking one another to keep the page alive, until the authentication is successful.

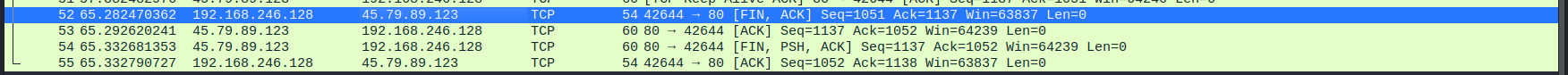
When that happens, these frames from 37 to 44 appear:



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.

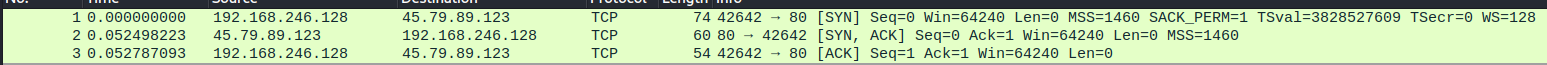


And when the website is closed, the process is finished, and frames 52 to 55 appear:



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)

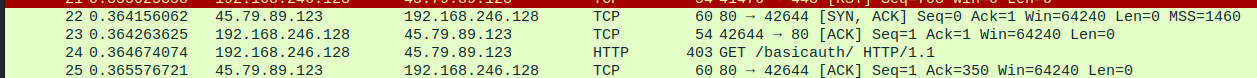


1. Using the TCP and the TSL protocol, the browser requests the facilitation of privacy and data security to encrypt the communication between the browser and the server (frame 4 to 21)

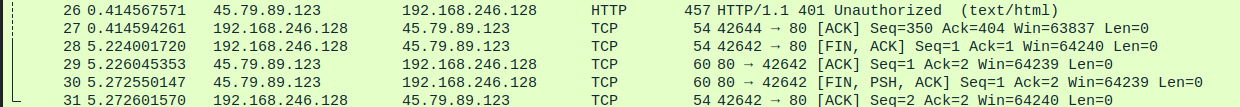
A picture containing calendar

Description automatically generated

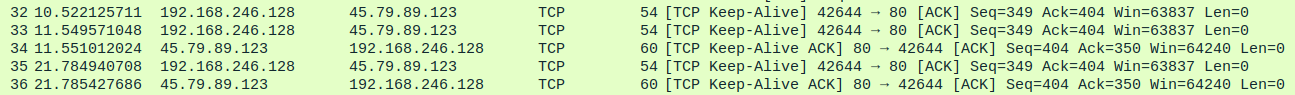
1. Using the TCP protocol and the HTTP protocol, the browser requests the server to fetch the HTML for the page (frame 22 to 25), specifically the prompt that asks users to enter username and password, as shown in frame 24, the browser requests the basic access authentication. The server acknowledges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



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



1. 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).

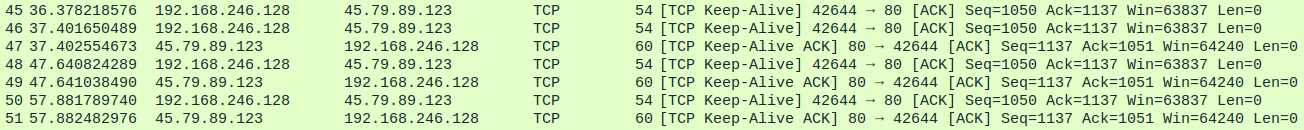


1. As the user enters in the credentials, the browser requests a GET to use HTTP basic authentication to see if users have correctly put in the right credentials (frame 37). As the server acknowledges so (frame 38), the server returns an OK using the HTTP protocol (frame 39). The browser then proceeds to request the rest of the HTML to be fetched from the server, using the TCP and HTTP protocol (frame 41 to 44)

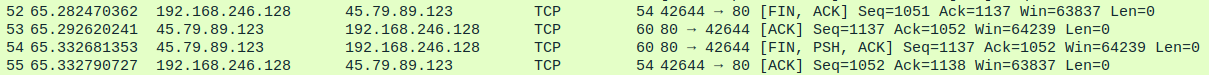
A picture containing chart

Description automatically generated

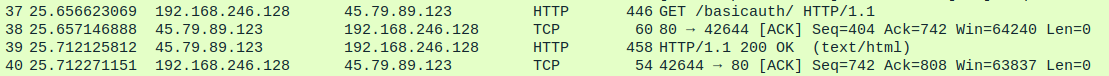
1. Similar to phase 5, the browser asks the server to keep the page alive with the TCP protocol and the TCP Keep-Alive flag (frame 45 to 51). Again, this can loop as long as the user stays on the page.



1. 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)



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.



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:

A picture containing calendar

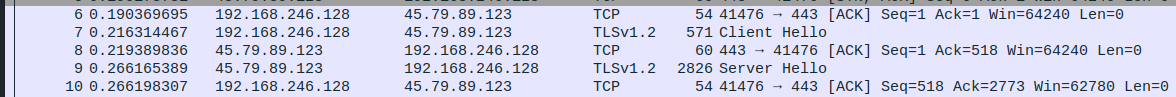
Description automatically generated

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:

Text

Description automatically generated

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:



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:

Text

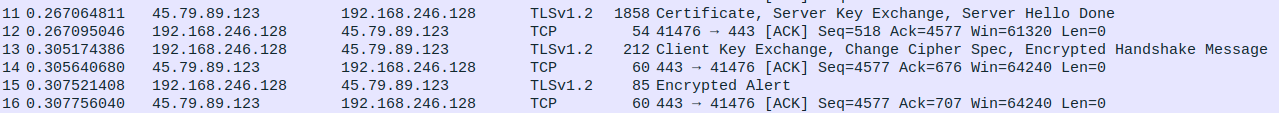
Description automatically generated

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.

Text

Description automatically generated

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



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.



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