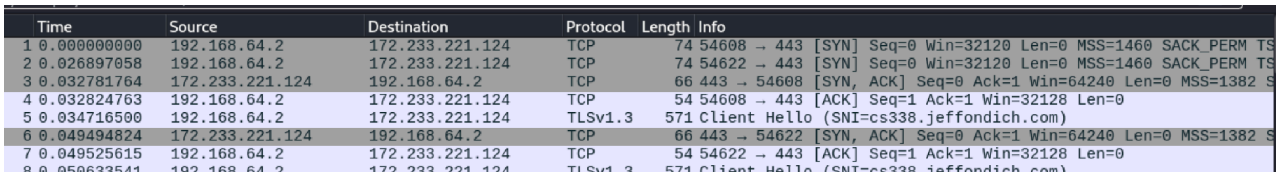
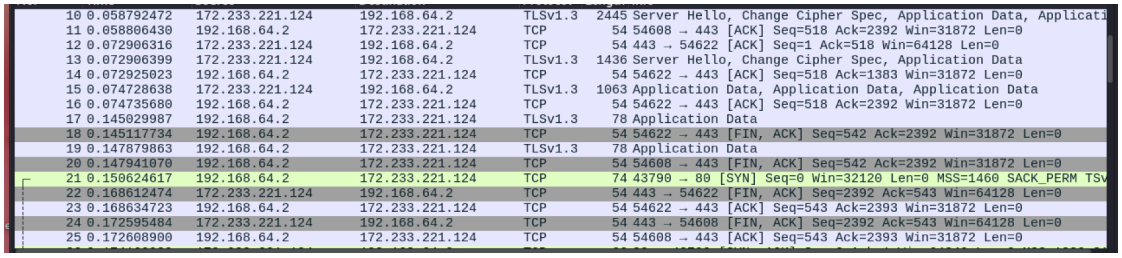
Partners: Anne Lehr & Jacky Guzman Nunez

The computer you are using, typically called client, looks up the IP address of the server (another computer) that has what you are looking for: the website you want to use is. In this instance, we will be accessing a website that requires authentication; thus, the interactions will shift slightly in order to demonstrate the interactions between the client and server when a password and username is needed. Regardless of the need for authentication a TCP[1] handshake takes place. A TCP handshake involves the client and server establishing a two-way connection between them in order to be able to send and receive information from each other. Essentially, the client tells the server "Hey can I talk to you?" and the server says "Sure! Can I talk to you too?" and the client says "Yeah!". This is seen through the sending of the packets that have the information [SYN][2], [SYN, ACK], and [ACK].



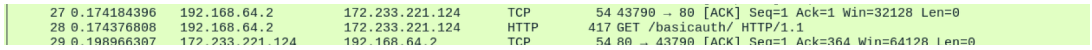
	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	192.168.64.2	172.233.221.124	TCP	74	54608 → 443 [SYN] Seq=0 Win=32120 Len=0 MSS=1460 SACK_PERM TS
2	0.026897058	192.168.64.2	172.233.221.124	TCP	74	54622 → 443 [SYN] Seq=0 Win=32120 Len=0 MSS=1460 SACK_PERM TS
3	0.032781764	172.233.221.124	192.168.64.2	TCP	66	443 → 54608 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1382 S
4	0.032824763	192.168.64.2	172.233.221.124	TCP	54	54608 → 443 [ACK] Seq=1 Ack=1 Win=32128 Len=0
5	0.034716500	192.168.64.2	172.233.221.124	TLSv1.3	571	Client Hello (SNI=cs338.jeffondich.com)
6	0.049494824	172.233.221.124	192.168.64.2	TCP	66	443 → 54622 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1382 S
7	0.049525615	192.168.64.2	172.233.221.124	TCP	54	54622 → 443 [ACK] Seq=1 Ack=1 Win=32128 Len=0
8	0.050633541	192.168.64.2	172.233.221.124	TLSv1.3	571	Client Hello (SNI=cs338.jeffondich.com)

Typically multiple TCP handshakes are made so that all the images and text files and whatnot is sent over in a timely manner as multiple 2-way connections are needed to ask and receive information from. (Only one "thing" can be asked for at a time.)



10	0.058792472	172.233.221.124	192.168.64.2	TLSv1.3	2445	Server Hello, Change Cipher Spec, Application Data, Applicati
11	0.058806430	192.168.64.2	172.233.221.124	TCP	54	54608 → 443 [ACK] Seq=518 Ack=2392 Win=31872 Len=0
12	0.072906316	172.233.221.124	192.168.64.2	TCP	54	443 → 54622 [ACK] Seq=1 Ack=518 Win=64128 Len=0
13	0.072906399	172.233.221.124	192.168.64.2	TLSv1.3	1436	Server Hello, Change Cipher Spec, Application Data
14	0.072925023	192.168.64.2	172.233.221.124	TCP	54	54622 → 443 [ACK] Seq=518 Ack=1383 Win=31872 Len=0
15	0.074728638	172.233.221.124	192.168.64.2	TLSv1.3	1063	Application Data, Application Data, Application Data
16	0.074735680	192.168.64.2	172.233.221.124	TCP	54	54622 → 443 [ACK] Seq=518 Ack=2392 Win=31872 Len=0
17	0.145029987	192.168.64.2	172.233.221.124	TLSv1.3	78	Application Data
18	0.145117734	192.168.64.2	172.233.221.124	TCP	54	54622 → 443 [FIN, ACK] Seq=542 Ack=2392 Win=31872 Len=0
19	0.147879863	192.168.64.2	172.233.221.124	TLSv1.3	78	Application Data
20	0.147941070	192.168.64.2	172.233.221.124	TCP	54	54608 → 443 [FIN, ACK] Seq=542 Ack=2392 Win=31872 Len=0
21	0.150624617	192.168.64.2	172.233.221.124	TCP	74	43790 → 80 [SYN] Seq=0 Win=32120 Len=0 MSS=1460 SACK_PERM TSv
22	0.168612474	172.233.221.124	192.168.64.2	TCP	54	443 → 54622 [FIN, ACK] Seq=2392 Ack=543 Win=64128 Len=0
23	0.168634723	192.168.64.2	172.233.221.124	TCP	54	54622 → 443 [ACK] Seq=543 Ack=2393 Win=31872 Len=0
24	0.172595484	172.233.221.124	192.168.64.2	TCP	54	443 → 54608 [FIN, ACK] Seq=2392 Ack=543 Win=64128 Len=0
25	0.172608900	192.168.64.2	172.233.221.124	TCP	54	54608 → 443 [ACK] Seq=543 Ack=2393 Win=31872 Len=0

After we have established the connection, the client is able to send an HTTP request[3], just like this (line 28):



27	0.174184396	192.168.64.2	172.233.221.124	TCP	54	43790 → 80 [ACK] Seq=1 Ack=1 Win=32128 Len=0
28	0.174376808	192.168.64.2	172.233.221.124	HTTP	417	GET /basicauth/ HTTP/1.1
29	0.198966307	172.233.221.124	192.168.64.2	TCP	54	80 → 43790 [ACK] Seq=1 Ack=364 Win=64128 Len=0

Line 28 is where the client tells the server to give us the information needed to be able to display on the screen the website we desire to see.

---

[1] TCP is a protocol (protocol is "a set of rules or procedures for transmitting data between electronic devices, such as computers" (Protocol | Definition, Examples, & Facts | Britannica)) that rearranges packets in order to make sure there are no duplicates and no missing data (reference: Jeff Ondich)

[2] "SYN is a TCP (Transmission Control Protocol) packet sent to another computer requesting that a connection be established between them." (What is SYN (Synchronize)? (computerhope.com)) ACK means acknowledgement.

[3] HTTP request: formal language used for asking and receiving information between client and server (reference: Ondich)

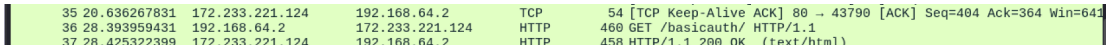
If the server is able to find what the client asked for, it sends it back, as seen on line 30:

However, we are told that we are unauthorized (401 Unauthorized) so we are not allowed to see the contents of the website. This website requires authentication as a pop up appears that asks us for a username and password. Authentication simply means that the client needs to prove the credentials needed in order to access what the server is protecting. Once we realize that we can't have access, we shrug our shoulders and say "ok :(" . [4]

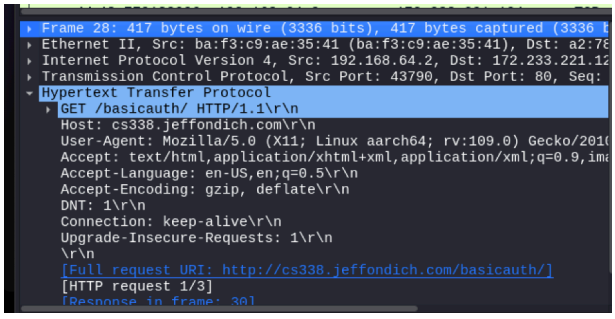
Whenever a request is made and the server sends over information, being the website or denial of it, there are always acknowledgements made to tell the other side, "thanks, I got it". Specifically in this example, we know that communication has been established because of the message [ACK] on line 31.

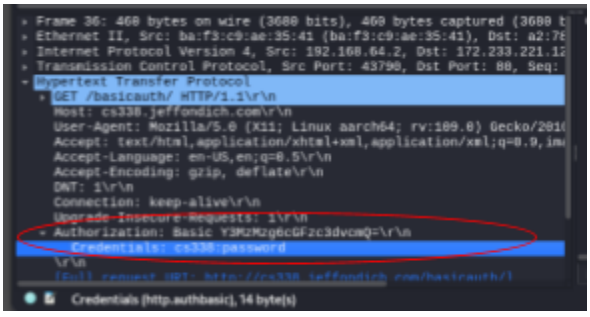
After we type in the username and password and hit Enter, the server checks if the information sent by the client was true, either saying "Who are you trying to fool? Not me. You are not who you say you are, so I cannot let you in/see the information I am keeping safe" or "Hey! Glad to see you back. Come on in, let me show you around…"[5].

The second time we request for the website (sending the password and username too) can be seen on line 36:



We know we sent the password and username the second time around as there is a clear difference between the HTTP requests:





[4]Source used as refresher: https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Authorization
[5]Reference for the whole paragraph: What is an Authentication Server (auth0.com)

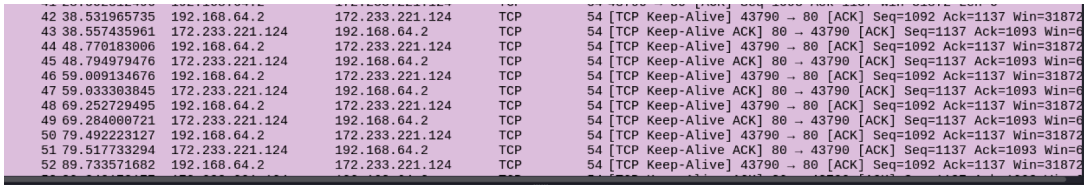
The presence of the Authorization header itself is confirmation of the system's authentication of the user. As you can see in the instance of line 28, when we aren't even given the ability to provide our credentials the authorization header is completely missing; however, for line 36 when we are accessing the page with our username and password, the authorization header is present. The password is sent by the browser to the server, that is why the username and password were converted from normal text to Base64: Basic Y3MzMzg6cGFzc3dvcmQ=\r\n .

They were not encrypted, but encoded. The format Base64 is used to encode the information, not encrypt it.[6] This means that if anyone were to have been snooping into the interaction between our computer and the server http://cs338.jeffondich.com/basicauth/, they could get our valuable information.

The transformation of plain text to Base64 is explicitly stated in the HTTP Basic Authentication specification document. The client followed the protocol!

As seen on line 37, the server let us in the website as we gave the right credentials. The 200 OK confirms that our request has been processed correctly. Yippee.

After that there are multiple lines where the client tells the server to keep the connection alive in case we need anything more from them.[7] The server then sends an acknowledgement and keeps the connection alive until the client no longer sends another request to keep it alive and terminates the connection.



[6] Used this as reference: RFC 7617 - The 'Basic' HTTP Authentication Scheme (ietf.org)
[7] Source used: https://orhanergun.net/understanding-tcp-keep-alive-mechanism