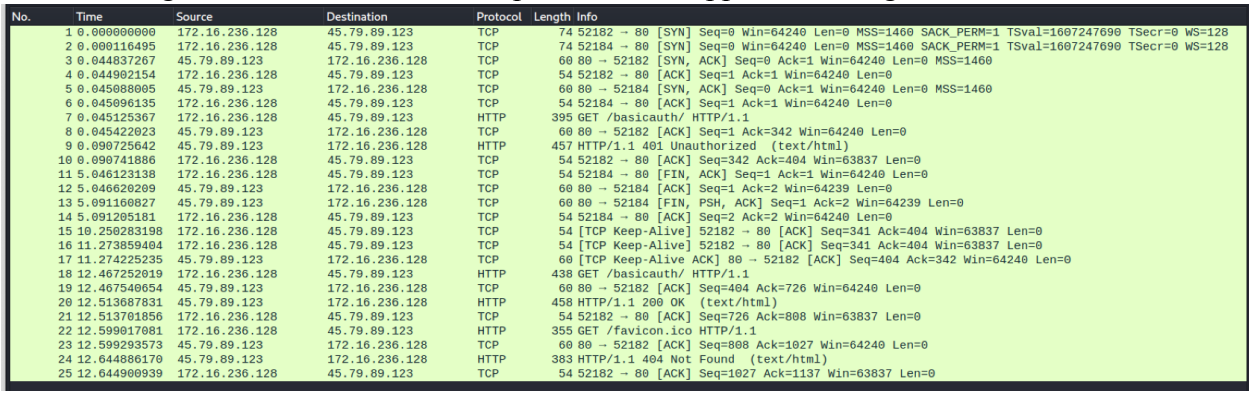
Lane Maitland

I first found the IP address for the site.

```
(base) lane-maitland@Lanes-MacBook-Air ~ % nslookup cs338.jeffondich.com
Server:         137.22.198.41
Address:        137.22.198.41#53

Non-authoritative answer:
Name:   cs338.jeffondich.com
Address: 45.79.89.123
```

In wireshark, I entered the capture filter `host 45.79.89.123` to observe interactions with the site. After entering the correct username and password, I stopped recording. This returned 25 frames.



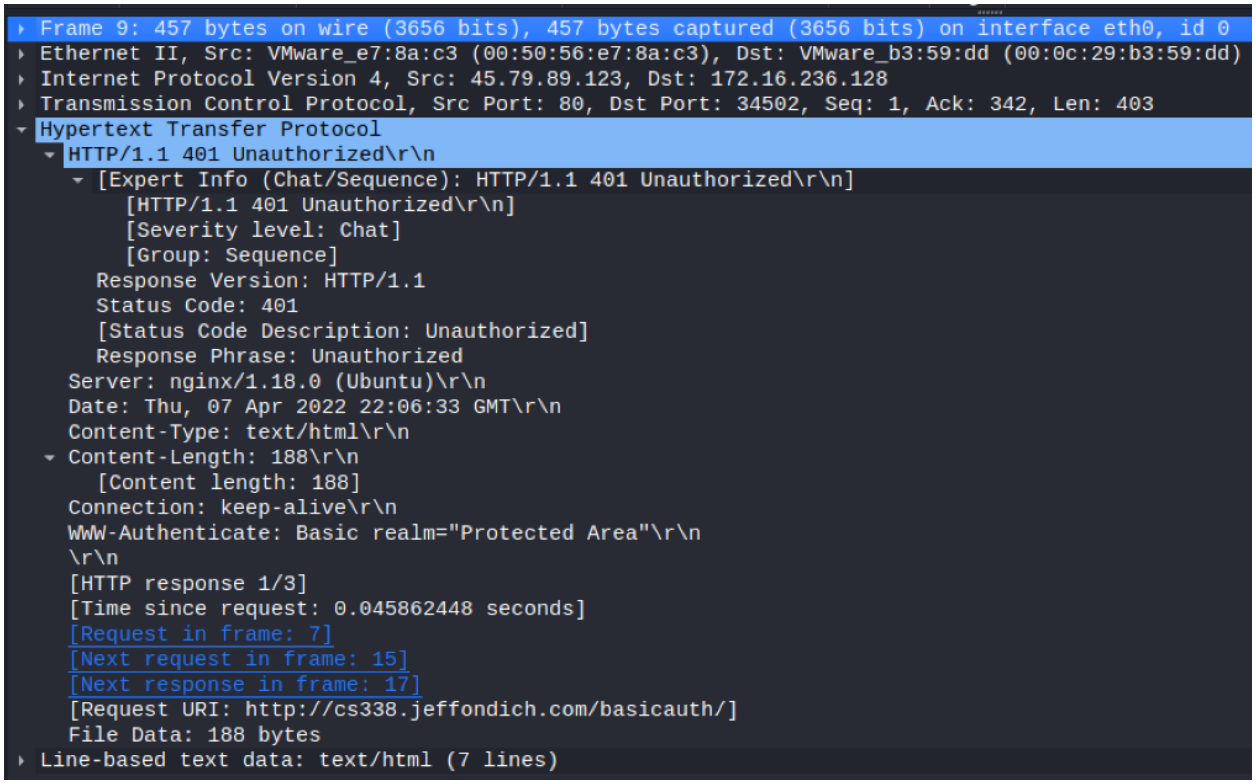
No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	172.16.236.128	45.79.89.123	TCP	74	52182 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=1607247690 TSecr=0 WS=128
2	0.000116495	172.16.236.128	45.79.89.123	TCP	74	52184 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=1607247690 TSecr=0 WS=128
3	0.044837267	45.79.89.123	172.16.236.128	TCP	60	80 → 52182 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
4	0.044902154	172.16.236.128	45.79.89.123	TCP	54	52182 → 80 [ACK] Seq=1 Ack=1 Win=64240 Len=0
5	0.045088005	45.79.89.123	172.16.236.128	TCP	60	80 → 52184 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
6	0.045096135	172.16.236.128	45.79.89.123	TCP	54	52184 → 80 [ACK] Seq=1 Ack=1 Win=64240 Len=0
7	0.045125367	172.16.236.128	45.79.89.123	HTTP	395	GET /basicauth/ HTTP/1.1
8	0.045422023	45.79.89.123	172.16.236.128	TCP	60	80 → 52182 [ACK] Seq=1 Ack=342 Win=64240 Len=0
9	0.090725642	45.79.89.123	172.16.236.128	HTTP	457	HTTP/1.1 401 Unauthorized (text/html)
10	0.090741886	172.16.236.128	45.79.89.123	TCP	54	52182 → 80 [ACK] Seq=342 Ack=404 Win=63837 Len=0
11	5.046123138	172.16.236.128	45.79.89.123	TCP	54	52184 → 80 [FIN, ACK] Seq=1 Ack=1 Win=64240 Len=0
12	5.046620209	45.79.89.123	172.16.236.128	TCP	60	80 → 52184 [ACK] Seq=1 Ack=2 Win=64239 Len=0
13	5.091160827	45.79.89.123	172.16.236.128	TCP	60	80 → 52184 [FIN, PSH, ACK] Seq=1 Ack=2 Win=64239 Len=0
14	5.091205181	172.16.236.128	45.79.89.123	TCP	54	52184 → 80 [ACK] Seq=2 Ack=2 Win=64240 Len=0
15	10.250283198	172.16.236.128	45.79.89.123	TCP	54	[TCP Keep-Alive] 52182 → 80 [ACK] Seq=341 Ack=404 Win=63837 Len=0
16	11.273859404	172.16.236.128	45.79.89.123	TCP	54	[TCP Keep-Alive] 52182 → 80 [ACK] Seq=341 Ack=404 Win=63837 Len=0
17	11.274225235	45.79.89.123	172.16.236.128	TCP	60	[TCP Keep-Alive ACK] 80 → 52182 [ACK] Seq=404 Ack=342 Win=64240 Len=0
18	12.467252019	172.16.236.128	45.79.89.123	HTTP	438	GET /basicauth/ HTTP/1.1
19	12.467540654	45.79.89.123	172.16.236.128	TCP	60	80 → 52182 [ACK] Seq=404 Ack=726 Win=64240 Len=0
20	12.513687831	45.79.89.123	172.16.236.128	HTTP	458	HTTP/1.1 200 OK (text/html)
21	12.513701856	172.16.236.128	45.79.89.123	TCP	54	52182 → 80 [ACK] Seq=726 Ack=808 Win=63837 Len=0
22	12.599017081	172.16.236.128	45.79.89.123	HTTP	355	GET /favicon.ico HTTP/1.1
23	12.599293573	45.79.89.123	172.16.236.128	TCP	60	80 → 52182 [ACK] Seq=808 Ack=1027 Win=64240 Len=0
24	12.644886170	45.79.89.123	172.16.236.128	HTTP	383	HTTP/1.1 404 Not Found (text/html)
25	12.644900939	172.16.236.128	45.79.89.123	TCP	54	52182 → 80 [ACK] Seq=1027 Ack=1137 Win=63837 Len=0

There are two TCP handshakes. For both, the source is Kali on my machine and the destination is the site. This means that Kali on my machine is initiating communication. The IP addresses for both of these handshakes are the same. Mine is 172.16.236.128 and, as seen previously in the terminal, the IP address for the site is 45.79.89.123. However, my machine is using two different ports (52182, 52184), while the site is only using one port (80).

Both ports send [SYN] packets before either receives a response. Then, the site acknowledges the request for initialization by port 52182 ([SYN, ACK]), and that port acknowledges the response ([ACK]). After both of those packets, the same type of communication occurs with port 52184.

The TCP handshakes are followed by an HTTP protocol to “GET” the page “basicauth” from the site. The source of this packet is my machine, and the destination is the site. This makes sense because I am asking my computer to search for a page and return it to me. (No one is asking me to return a page to them.)

After the HTTP protocol, there is another TCP protocol. This is a message from the site to my machine, specifically port 52182. It communicates that my machine is not authorized to display the page by sending a “401 (Unauthorized)” error. The same port (52182) acknowledges receiving that message.



In the header for this frame, we can see the status code, description, and response phrase, which all communicate that the client cannot access what they are trying to. We can also see that the content which they are attempting to access is HTML at the URI of the site. A keep-alive connection begins, and we see that the response in this frame is one of three. Before the "GET" reattempt in frame 18, there are indeed three responses from the site. These are for executing the authentication process. The frames listed (15,17) do indeed match the next request and next response for the keep-alive frames, and frame 7 does indeed correspond to the frame with the original "GET" request.

Back in frame 11, the other port (52184) requests to terminate communication with the site. At the same time, it acknowledges the previous packet it received, even though it already did. Between frames 6 and 11, nothing was sent to this port, and the sequence numbers in these frames are the same (1). This confirms that the acknowledgement is for the same packet sent in frame 5, where the acknowledgement number is 1. The site acknowledges the request to terminate, and in the next frame terminates the communication, pushes the data entered, and acknowledges the last packet it received again (the acknowledgement number is again the same, and no packets were sent to the site between frames 12 and 13).

The [PSH] flag means that the username and password should be sent to be checked. This must have happened after I clicked/entered "sign in." The [PSH] flag means that the site is not waiting for me to type more characters into the username or password fields. In other words, it signifies that I am all done with my input and it is ready to be sent. This is important because I would not want the site to check credentials before I am done typing the entire username or password. {https://www.howtouselinux.com/post/psh-ack-tcp-flags}

Now, the only active port on my machine is 52182. Interaction with the other port was likely to communicate about the encrypted version of the username and password. I entered the username and password on the site, so the [PSH] packet is sent by the site to my machine. From the HTTP Authentication document, I know that the password is encoded. My machine "encodes the user-pass into an octet sequence, … and obtains the basic-credentials by encoding this octet sequence using Base64 into a sequence of US-ASCII characters."
{https://datatracker.ietf.org/doc/html/rfc7617}

However, the password does not seem to be encrypted, as there is no encryption key, and the site does not send another packet after the username and password are pushed and before it acknowledges that my machine can try to request the page again. If the password was encrypted, I would expect more acknowledgement packets to communicate about keys and encrypted versions of the password.

There is a [TCP Keep-Alive] packet sent by my machine to the site (to be specific, it is sent 2 times). This happens because my machine wants to know when the site is done checking my credentials.
{https://knowledgebase.progress.com/articles/Article/P98399}

After the site acknowledges the keep-alive packet, my machine then tries again to "GET" the page "basicauth" from the site. The site acknowledges that request, and this time, the next HTTP protocol is "OK." My machine is able to display the site.

After that success, my machine tries to "GET" the "favicon.ico" from the site. This is met with a "404 Not Found" error.

Wireshark relates to the HTTP Authorization document because it displays the HTTP method of requiring credentials after receiving a "401 (Unauthorized)" response.
{https://datatracker.ietf.org/doc/html/rfc7235#section-4.2}

Wireshark relates to the HTTP specification document because it displays the expected syntax for the "GET" method and the headers.
{https://datatracker.ietf.org/doc/html/rfc7230#section-1}