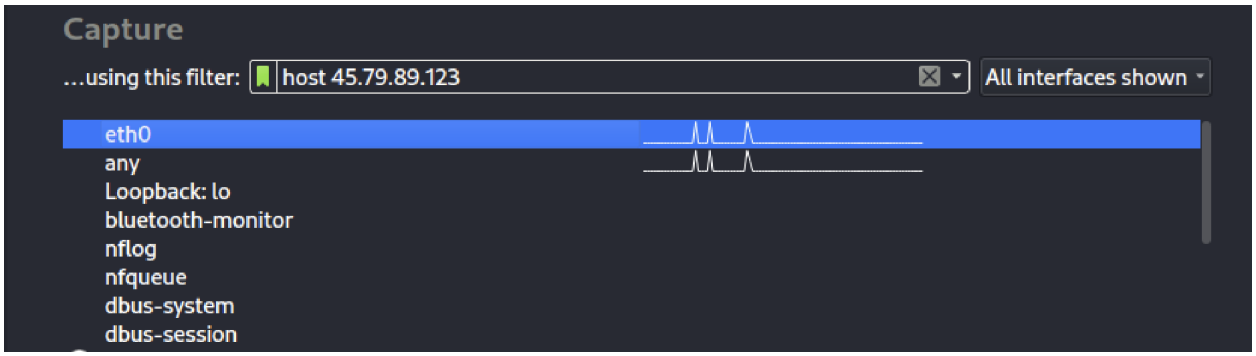
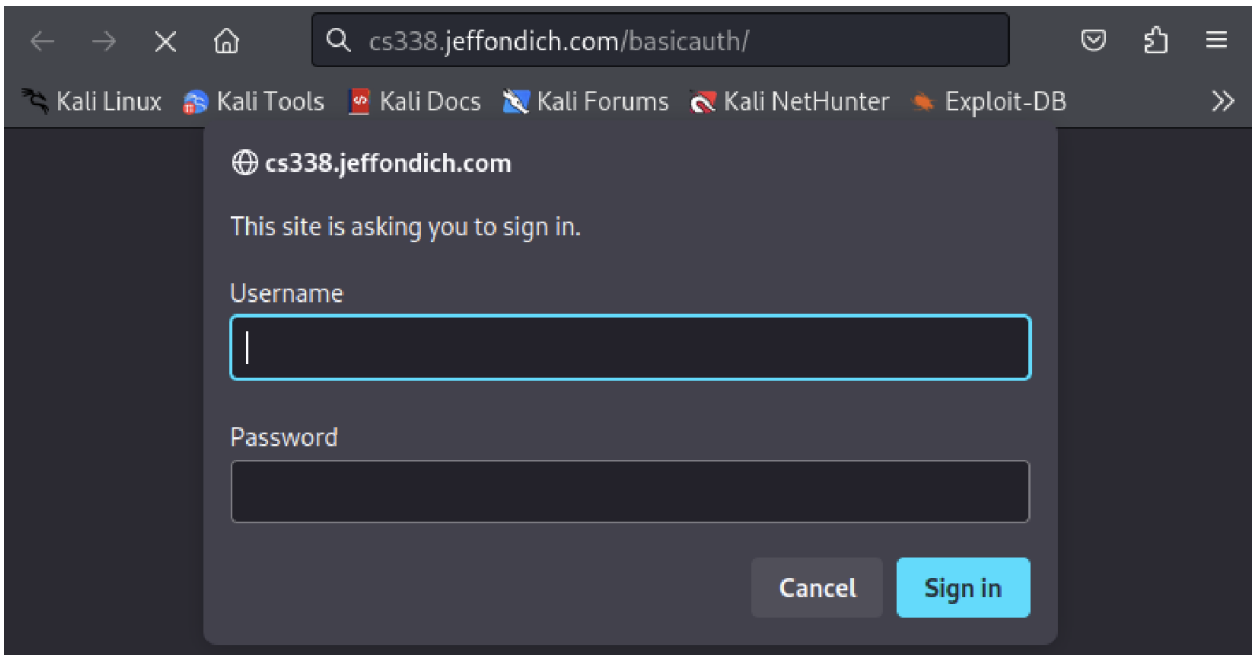
After some difficulty and professor consultation, I managed to open Wireshark on the ethernet interface (for me `eth0`) and filter to the target site's IP address.

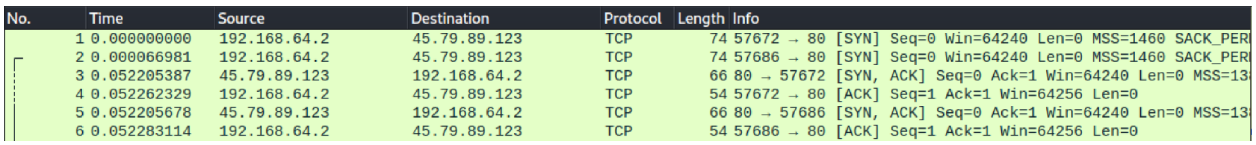


From there I opened Firefox on Kali and submitted the URL for the target page, and was greeted with the authentication pop-up.



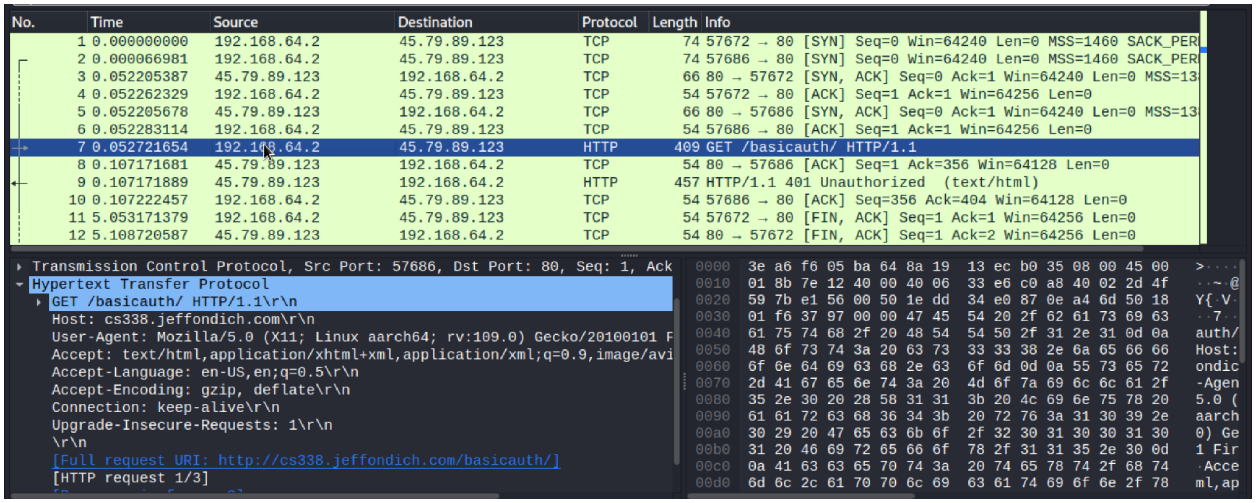
I input our credentials to gain access to the page, and then hopped over to Wireshark to stop sniffing and analyze the packets.

The first thing that was interesting was that my browser had apparently opened up two ports, 57672 and 57686, to the server's port 80. Jeff said this was likely to handle different kinds of incoming packets more quickly.

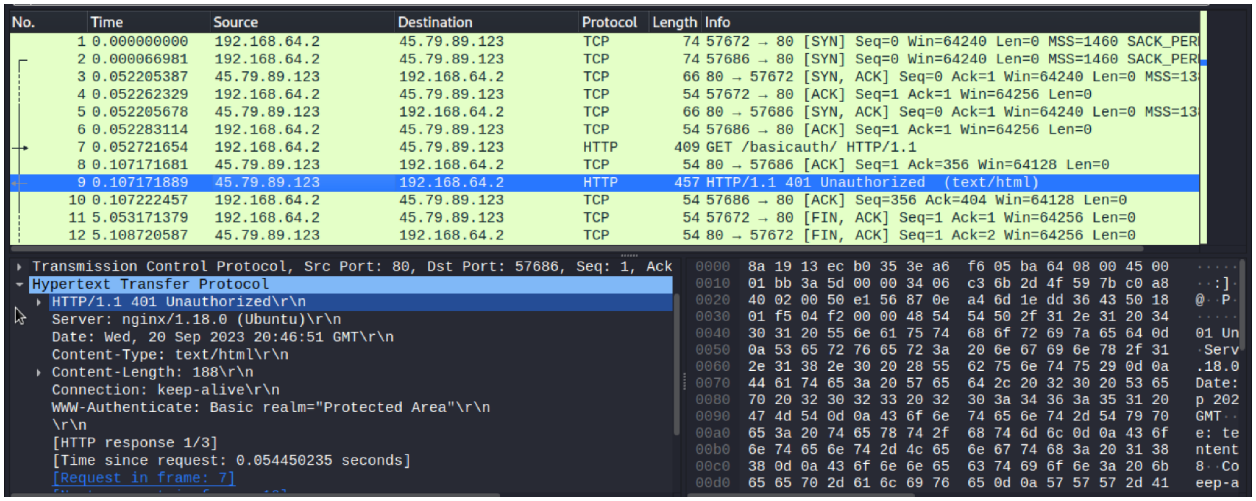


No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	192.168.64.2	45.79.89.123	TCP	74	57672 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PER
2	0.000066981	192.168.64.2	45.79.89.123	TCP	74	57686 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PER
3	0.052205387	45.79.89.123	192.168.64.2	TCP	66	80 → 57672 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=13
4	0.052262329	192.168.64.2	45.79.89.123	TCP	54	57672 → 80 [ACK] Seq=1 Ack=1 Win=64256 Len=0
5	0.052205678	45.79.89.123	192.168.64.2	TCP	66	80 → 57686 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=13
6	0.052283114	192.168.64.2	45.79.89.123	TCP	54	57686 → 80 [ACK] Seq=1 Ack=1 Win=64256 Len=0

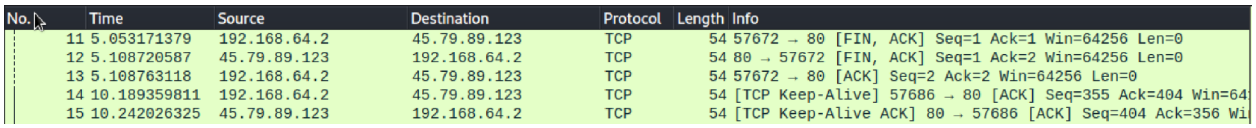
After the ports were opened, my browser requested the page at the URL,



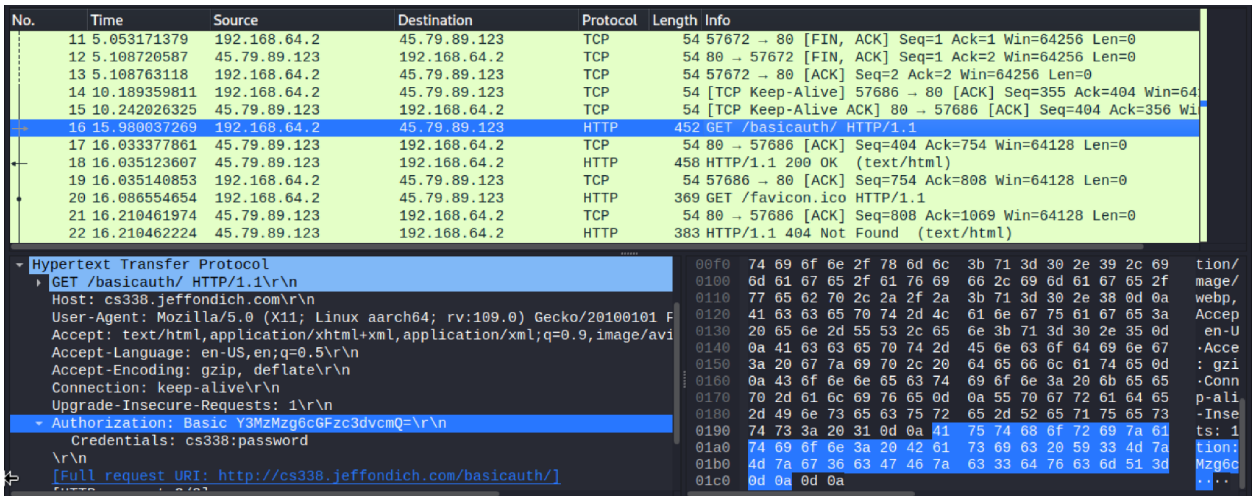
and the server denied access, claiming it was a protected area and offering the credential input pop-up.



Between the time the client received the denial HTTP and I sent the credentials back, my client asked to shut down one port, 57672, and the server agreed. The client requested, however, that the connection on port 57686 persist and the server also agreed.



Once I had input the credentials, the client sent another HTTP packet, this time including the username and password in base64.



The server accepted the credentials, and sent back the HTTP for the actual page.