McKenna Wirth and Antonia Ritter
CS338 Spring 2022

HTTP'S BASIC AUTHENTICATION: A STORY

1. Handshakes: Hello I Am Client

Before we can talk to cs338.jeffondich.com, we have to find cs338.jeffondich.com, a process which occupies frames 1-9 and 11-14. In frame 10 we first see destination 45.79.89.123, the IP address corresponding to cs338.jeffondich.com (confirmed using nslookup). Frame 10 is a TCP SYN from port 55374.

Frame 15 is another TCP SYN from port 46742, followed by a SYN, ACK to port 55374; that is, it's responding to frame 10. Then frame 17 is an ACK response to this connection. Meanwhile, in frame 18, the server responds to the message in frame 15. In summary, we have two TCP handshakes with different client source ports.

Later on, the connection with port 46742 is ended with FIN (frame 34), while the connection with 55374 persists. Browsers often open multiple connections to allow for multiple simultaneous downloads, therefore speeding up the process. Port numbers are randomized for security considerations, as discussed in class.

Throughout the session, the client opens a TLSv1.2 exchange. There's an interaction similar to the TCP handshake ("Client Hello," "Server Hello") and some packets labeled as key exchanges (see screenshot below). As there are keys exchanged and references to ciphers, this exchange seems to be encrypted.

18	2.750272204	45.79.89.123	192.168.249.128	TCP	60 443 → 46742 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
19	2.750318663	192.168.249.128	45.79.89.123	TCP	54 46742 → 443 [ACK] Seq=1 Ack=1 Win=64240 Len=0
20	2.752028126	192.168.249.128	45.79.89.123	TLSv1.2	571 Client Hello
21	2.752192723	45.79.89.123	192.168.249.128	TCP	60 443 → 46742 [ACK] Seq=1 Ack=518 Win=64240 Len=0
22	2.798643241	45.79.89.123	192.168.249.128	TLSv1.2	1440 Server Hello
23	2.798682940	192.168.249.128	45.79.89.123	TCP	54 46742 → 443 [ACK] Seq=518 Ack=1387 Win=63756 Len=0
24	2.799190030	45.79.89.123	192.168.249.128	TCP	1440 443 → 46742 [PSH, ACK] Seq=1387 Ack=518 Win=64240 Len=1386 [TCP segment of a reassembled PDU]
25	2.799209329	192.168.249.128	45.79.89.123	TCP	54 46742 → 443 [ACK] Seq=518 Ack=2773 Win=63756 Len=0
26	2.799635620	45.79.89.123	192.168.249.128	TCP	1378 443 → 46742 [PSH, ACK] Seq=2773 Ack=518 Win=64240 Len=1324 [TCP segment of a reassembled PDU]
27	2.799646020	192.168.249.128	45.79.89.123	TCP	54 46742 → 443 [ACK] Seq=518 Ack=4097 Win=63756 Len=0
28	2.799850116	45.79.89.123	192.168.249.128	TLSv1.2	534 Certificate, Server Key Exchange, Server Hello Done
29	2.799859815	192.168.249.128	45.79.89.123	TCP	54 46742 → 443 [ACK] Seq=518 Ack=4577 Win=63756 Len=0
30	2.805450697	192.168.249.128	45.79.89.123	TLSv1.2	212 Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
31	2.805565295	45.79.89.123	192.168.249.128	TCP	60 443 → 46742 [ACK] Seq=4577 Ack=676 Win=64240 Len=0
32	2.805717192	192.168.249.128	45.79.89.123	TLSv1.2	85 Encrypted Alert
33	2.805795590	45.79.89.123	192.168.249.128	TCP	60 443 → 46742 [ACK] Seq=4577 Ack=707 Win=64240 Len=0
34	2.805888088	192.168.249.128	45.79.89.123	TCP	54 46742 → 443 [FIN, ACK] Seq=707 Ack=4577 Win=63756 Len=0
35	2.806171282	45.79.89.123	192.168.249.128	TCP	60 443 → 46742 [ACK] Seq=4577 Ack=708 Win=64239 Len=0
36	2.808564231	192.168.249.128	45.79.89.123	TCP	74 55376 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=2866640111 TSecr=0 WS=128
37	2.851984014	45.79.89.123	192.168.249.128	TLSv1.2	105 Change Cipher Spec, Encrypted Handshake Message
38	2.852006214	192.168.249.128	45.79.89.123	TCP	54 46742 → 443 [RST] Seq=708 Win=0 Len=0

TLSv1.2 exchanges

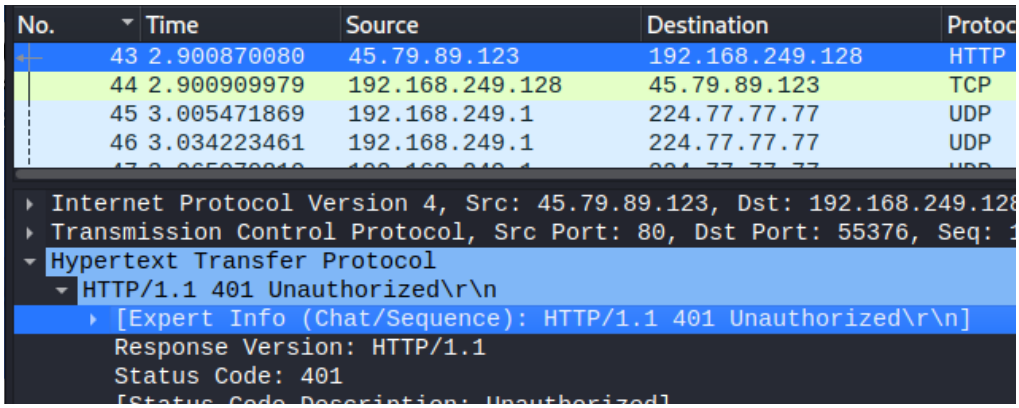
2. HTTP Requests: Will You Be My Friend?

The first HTTP request for /basicauth/ comes in frame 41, with the user-agent field indicating that it's Mozilla Firefox asking, as the HTTP documentation reflects in section 2.1 of RFC 7230

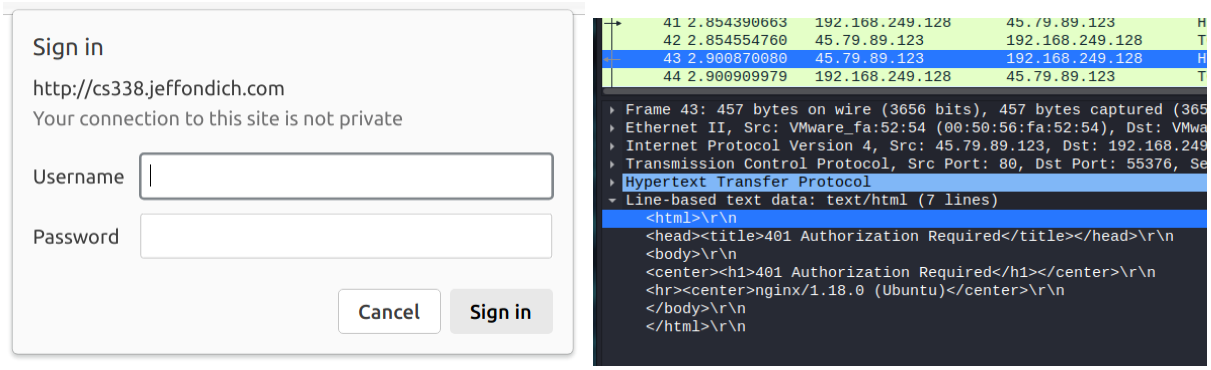
(https://datatracker.ietf.org/doc/html/rfc7230#section-2.1). The Source and Destination fields in Wireshark show the IP addresses for our client (192.168.249.128) and Jeff's server (45.79.89.123). All of the HTTP communications take place between these two IP addresses.

Request for /basicauth/

The server responds with a TCP ACK and then an HTTP packet with an error code "401: Unauthorized." The HTTP documentation states that a server will always respond with a success or error code in Section 2.1 (https://datatracker.ietf.org/doc/html/rfc7230#section-2.1). This packet also contains the HTML for the sign-in box.



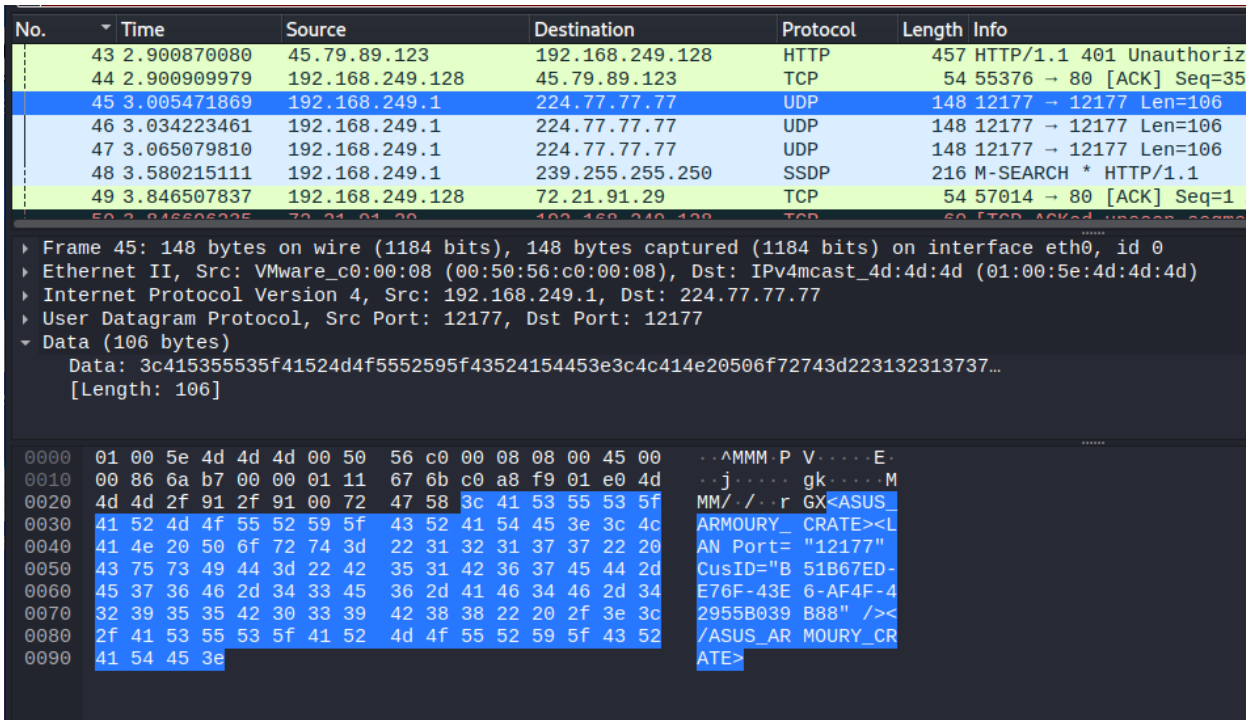
Error 401: Unauthorized



The sign-in box and corresponding HTML

Then TCP sends an ACK packet to the server indicating incomplete data; it received the sign-in box instead of what it had asked for.

Sidenote: At this point there is a series of frames which involve the UDP protocol and contain data about Armoury Crate, a hardware optimization program already on McKenna's computer, which she thinks is creepy (and she's right about that).

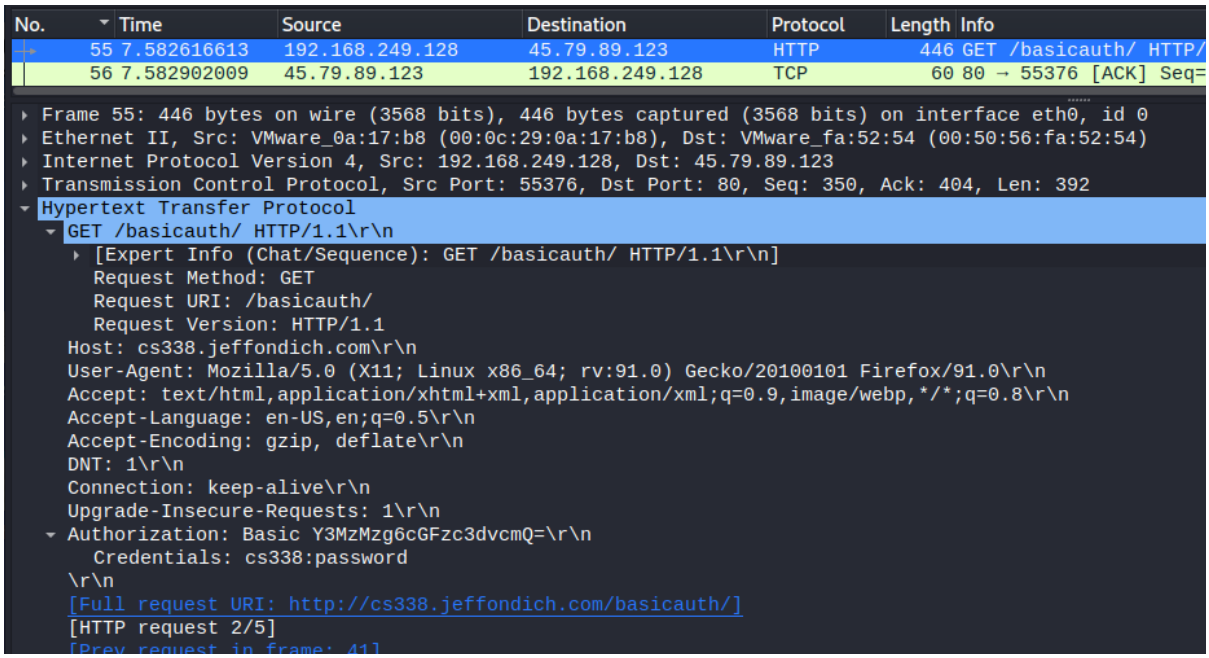


Mysterious UDP Armoury Crate-related data

3. Authorization: Okay, Friends

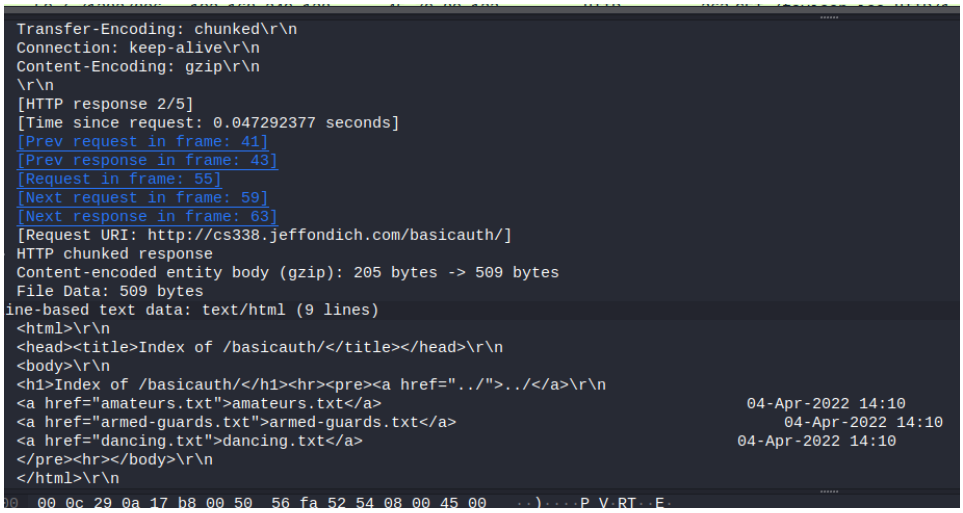
Finally, frame 55 is a second HTTP GET request from the browser to the server for the index file of the basicauth site. This request also includes the username and password for the site, in the "Authorization" section of the HTTP header. It is encoded in base64 as "Y3MzMzg6cGFzc3dvcmQ=", which Wireshark helpfully decoded into "cs338:password" for us. Note that both the password and username are encoded but not encrypted.

This authorization process is exactly as described in Section 4.2 of RFC 7235 (https://datatracker.ietf.org/doc/html/rfc7235#section-4.2).



Second request for /basicauth/, featuring Authorization field

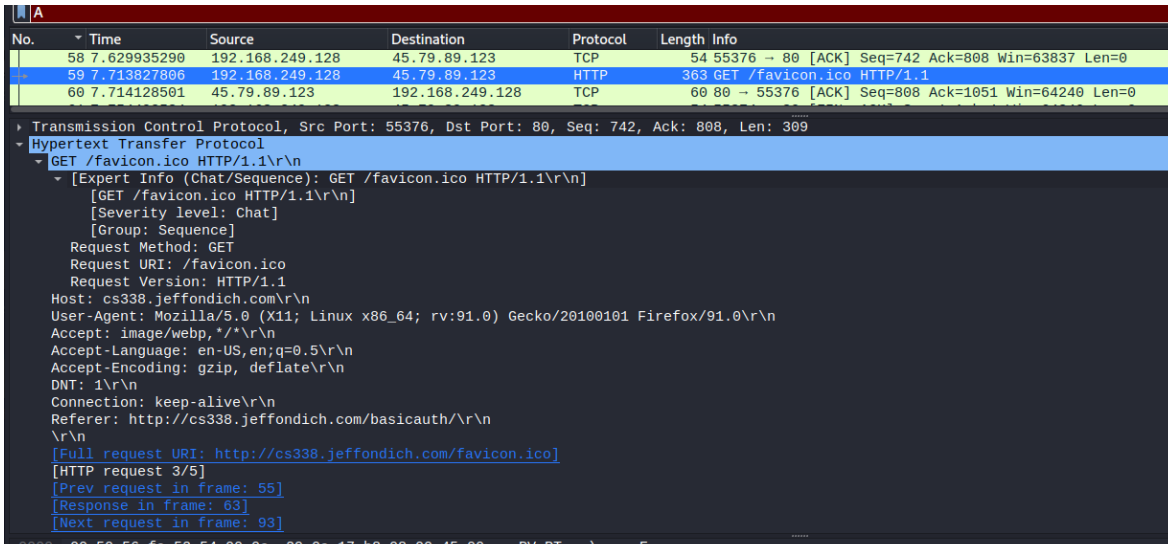
Frame 57 is the HTTP OK - the server decoded the credentials and deemed them sufficient, sending back the code 200 - and contains the HTML for "Index of /basicauth/".



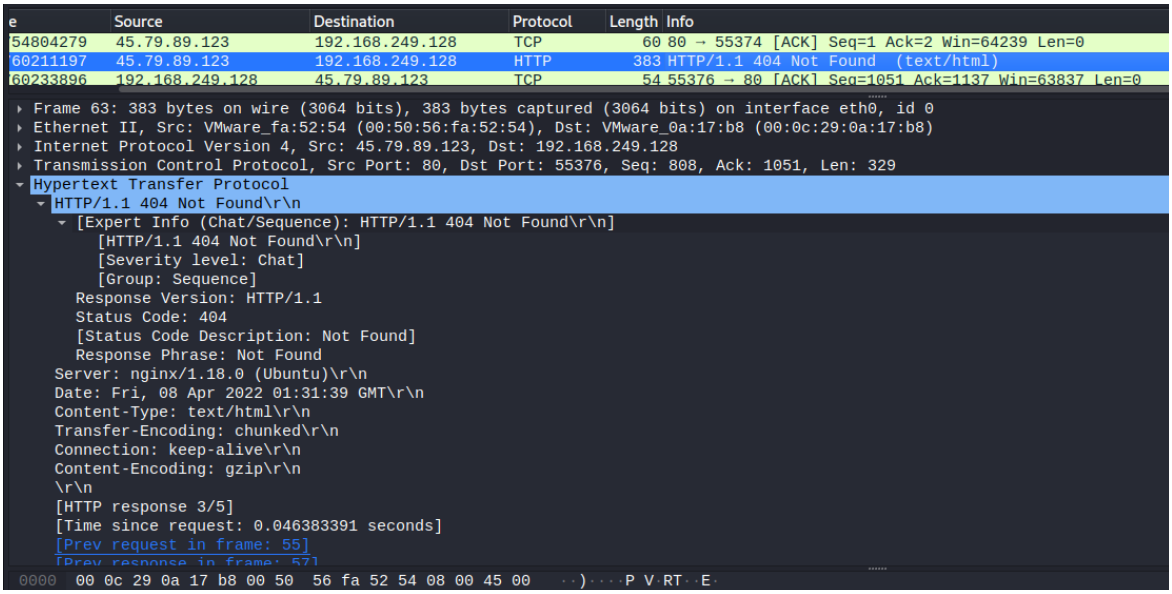
HTML for Index of /basicauth/

4. Hacker Voice: We're In

Once we receive the website HTML, the browser immediately makes another HTTP GET request: favicon.ico please? Unfortunately, says the server, "404 Not Found."



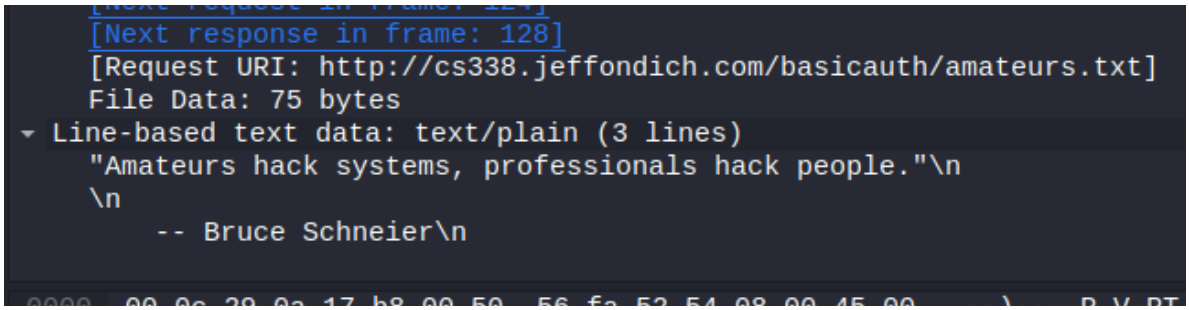
Request for favicon



404: No favicon found

Sidenote 2: In frame 82 the browser makes a request using NTP containing timestamp data, and gets a response in frame 95, containing the sent timestamp as well as a few other, slightly different ones. We feel slightly offended that our browser is checking the time while we're trying to have a conversation with jeffondich.com.

We also clicked on amateurs.txt, shown in frame 93 which contains the HTTP request for amateurs.txt, and the contents are returned in frame 98.



HTML for amateurs.txt

This marks the end of our browsing session. We opened this file, read the quote, and stopped Wireshark, leaving our web browser tab open, so as to better contemplate the wisdom of Bruce Schneier.