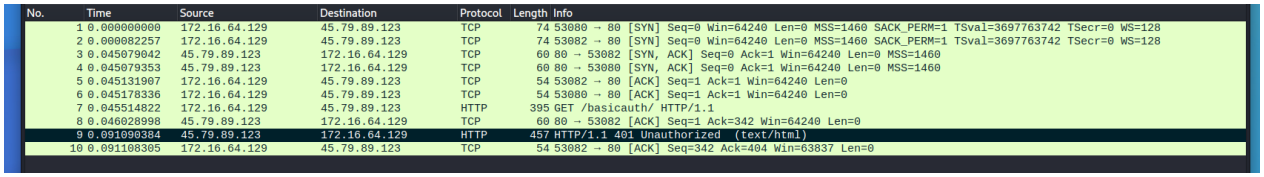
Exploring Jeff's Authentication Example: A Journey

Here are the initial packets upon loading of the page from a normal browser tab, and prior to interacting with the authentication message:



No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	172.16.64.129	45.79.89.123	TCP	74	53080 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=3697763742 TSecr=0 WS=128
2	0.000082257	172.16.64.129	45.79.89.123	TCP	74	53082 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=3697763742 TSecr=0 WS=128
3	0.045079042	45.79.89.123	172.16.64.129	TCP	60	80 → 53082 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
4	0.045079353	45.79.89.123	172.16.64.129	TCP	60	80 → 53080 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
5	0.045131907	172.16.64.129	45.79.89.123	TCP	54	53082 → 80 [ACK] Seq=1 Ack=1 Win=64240 Len=0
6	0.045178336	172.16.64.129	45.79.89.123	TCP	54	53080 → 80 [ACK] Seq=1 Ack=1 Win=64240 Len=0
7	0.045514822	172.16.64.129	45.79.89.123	HTTP	395	GET /basicauth/ HTTP/1.1
8	0.046028998	45.79.89.123	172.16.64.129	TCP	60	80 → 53082 [ACK] Seq=1 Ack=342 Win=64240 Len=0
9	0.091090384	45.79.89.123	172.16.64.129	HTTP	457	HTTP/1.1 401 Unauthorized (text/html)
10	0.091108305	172.16.64.129	45.79.89.123	TCP	54	53082 → 80 [ACK] Seq=342 Ack=404 Win=63837 Len=0

Figure 1: Initial packets

One thing to note about the initial interactions between our Kali client and the server is that our client sends out two TCP synchronization requests and the server ends up accepting only one. We can see this in Figure one on frames 1 and 2. Both frames show our client sending the same synchronization request to the server, but from two different ports (53080 and 53082). Initially, we believed that this was due to our client creating two ports in order to test for HTTP and HTTPS server cases, but the destination port for both client port requests is port 80. Because port 80 is an HTTP port, we were wondering whether the reason behind our two synchronization requests was something else.

Normal tab vs Private tab

Along a similar vein, we noticed stark differences in our wireshark captures when we accessed the website from a "normal" Firefox browser tab and from a private browsing tab. Here are the packets detected after loading the page and successfully authenticating from a normal tab:

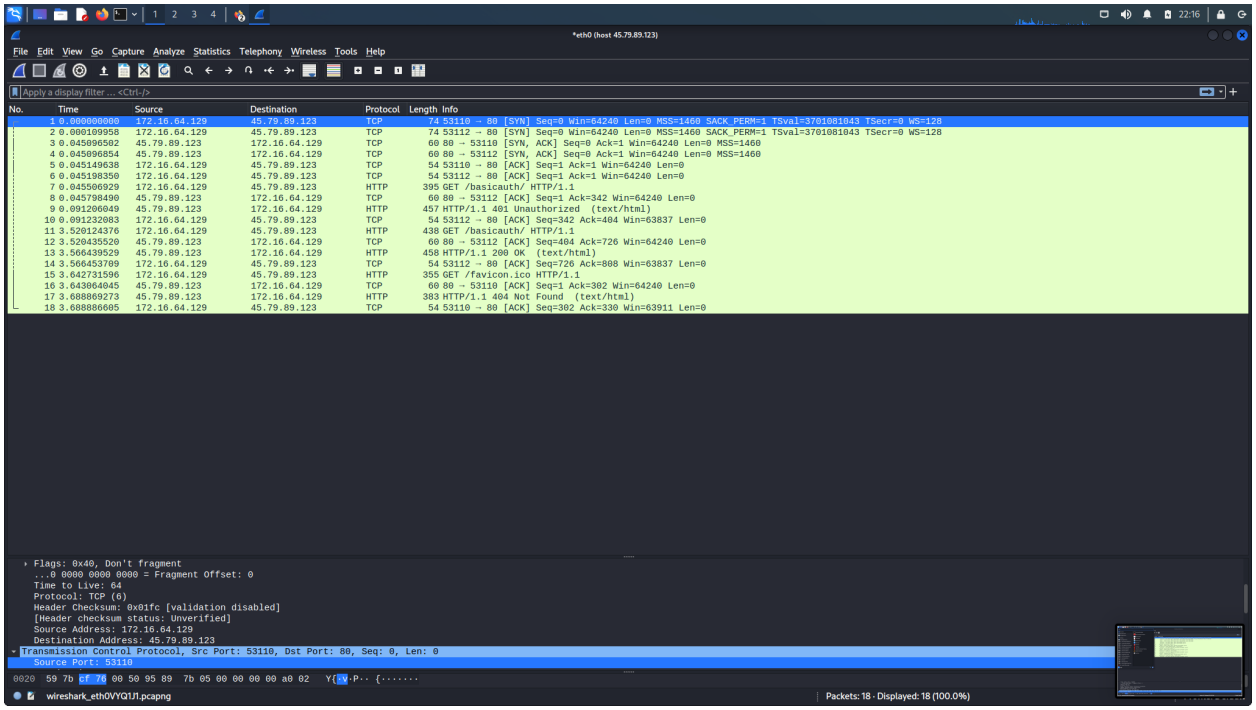


Figure 2: Accessing page from normal tab

On the other hand, we can see that accessing the page from a private browsing tab generates a different wireshark capture:

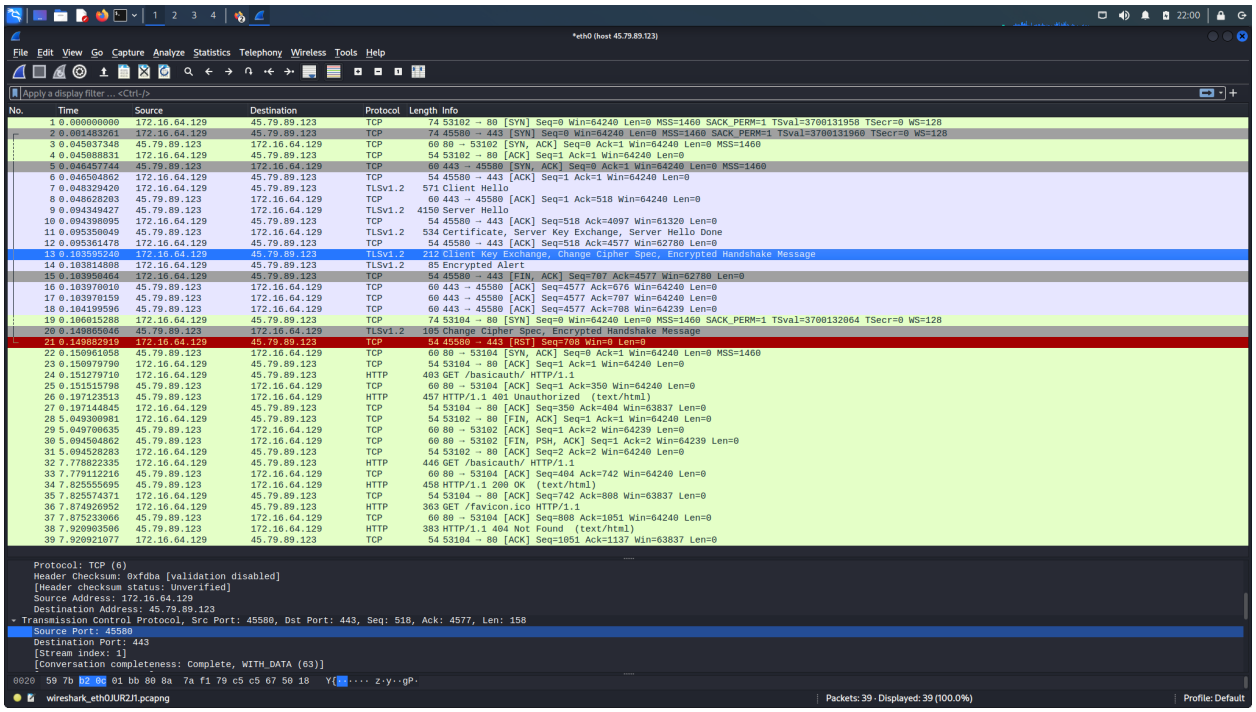


Figure 3: Accessing page from private tab

We discovered that accessing the page from a private browsing tab prompts our client to attempt synchronization from 3 distinct ports: 2 attempting to connect to an HTTP port and 1 attempting to connect to an HTTPS port. In Figure 3, the following frames indicate Kali's client attempts to establish an HTTPS encrypted connection with the server: 2, 5-18, 20-21. Wireshark shows us that the attempt to establish an HTTPS encrypted connection with the server fails on frame 21, the frame highlighted in red. Besides this difference of an attempt to establish an HTTPS connection with the server, Figures 2 and 3 relay the same information. Because of this, we will be primarily referring to frame information gathered from Wireshark when we access the page from a normal tab.

Loading page Pre-Authentication

After the TCP handshake, the browser sends a GET request for the resource /basicauth/ HTTP/1.1. This represents the browser trying to load the homepage. However, since it is password-protected, the server sends back an HTTP message that reads 401 Unauthorized. We're unable to access the resource as of yet. As seen in Figure 4 below, the HTTP header WWW-Authenticate announces that this resource is a "Protected Area."

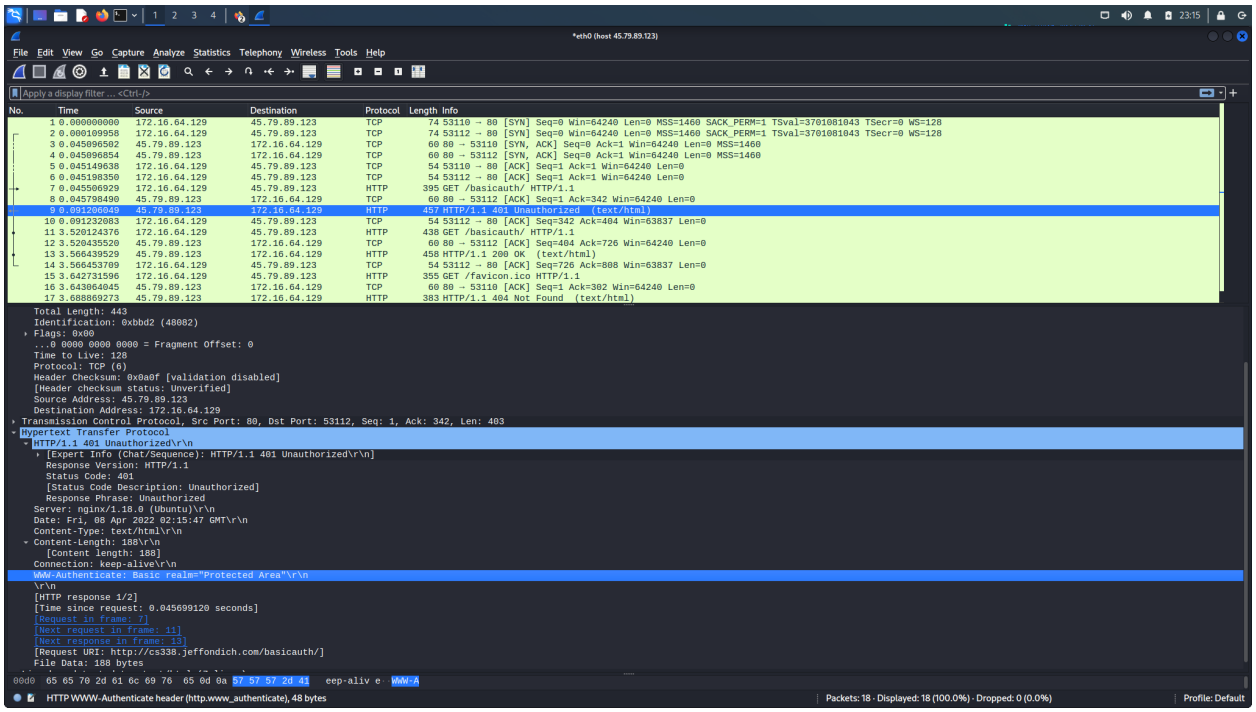


Figure 4

But by the next time the browser attempts the same GET request, we have filled the username and password fields with the correct credentials. Thus, the server sends back a 200 OK message.

The last couple of interactions are requests and responses for the site's favicon, and a mysterious 404 Not Found whose reason is not apparent.

Page authentication/authorization checking

We believe that the server authenticates and authorizes the input credentials. It is the server's HTTP response back to the user that delivers either an Unauthorized or an OK. After some research, this seems to be confirmed:
"The server will service the request only if it can validate the user-id and password for the protection space applying to the requested resource."
(https://datatracker.ietf.org/doc/html/rfc7617#section-2)

As seen in Figure 5 below, the Authorization header stores a Basic encoded version of the credentials, which are shown in the Credential headers directly below:

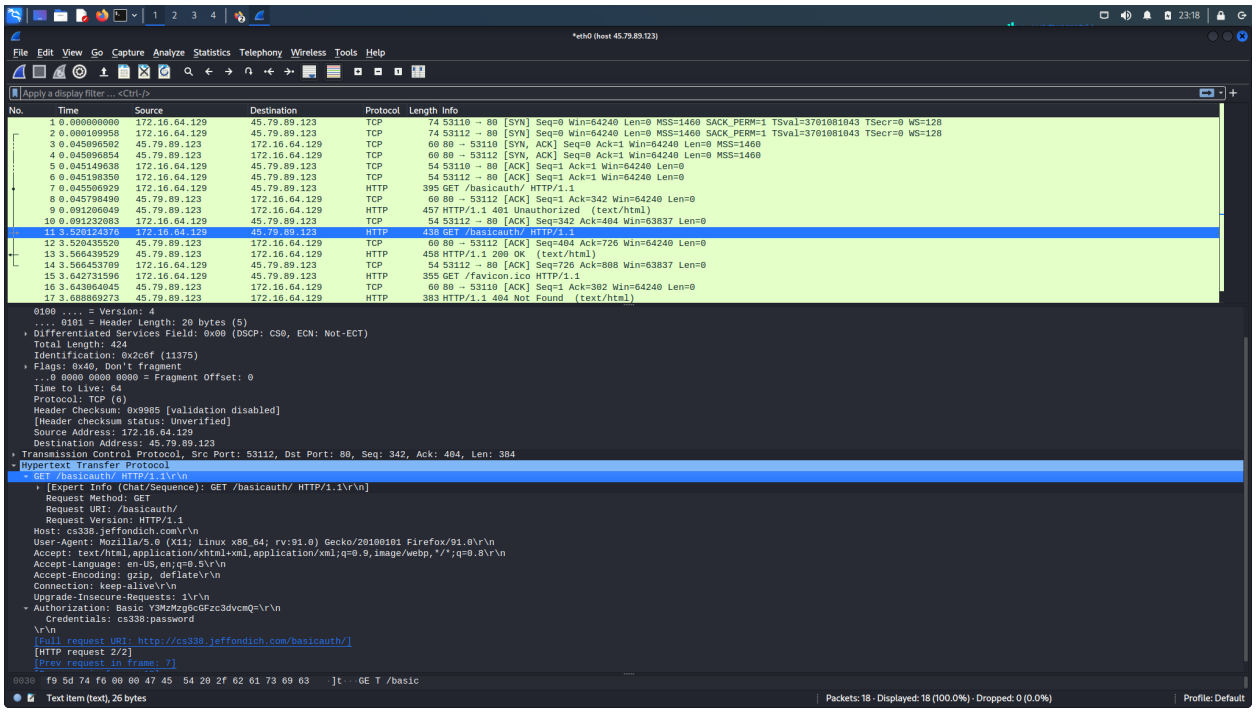


Figure 5

How do we get from the credentials to the encoded version? First, the user ID and password are concatenated using a colon. Then, that string is encoded into an octet sequence. The octet sequence used for this encoding is likely either ISO-8859-1 or UTF-8 (https://tools.ietf.org/id/draft-reschke-basicauth-enc-00.html Section A.1). The octet sequence is then encoded once more using Base64, the result of which is sent to the server. This final encoded message can be seen under "Authorization" in Figure 5 as the string beginning with "Y3MzMz…".