

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.

10	2.752028126	45.79.89.123	192.168.249.128	TCP	54 46742 → 443 [ACK] Seq=1 Ack=1 Win=64240 Len=0
19	2.756318663	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	2.799190630	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	2.885450697	192.168.249.128	45.79.89.123	TLSv1.2	212 Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
31	2.885505295	45.79.89.123	192.168.249.128	TCP	60 443 → 46742 [ACK] Seq=4577 Ack=676 Win=64240 Len=0
32	2.885717192	192.168.249.128	45.79.89.123	TLSv1.2	85 Encrypted Alert
33	2.885795590	45.79.89.123	192.168.249.128	TCP	60 443 → 46742 [ACK] Seq=4577 Ack=707 Win=64240 Len=0
34	2.885888088	192.168.249.128	45.79.89.123	TCP	54 46742 → 443 [FIN, ACK] Seq=707 Ack=4577 Win=63756 Len=0
35	2.886171282	45.79.89.123	192.168.249.128	TCP	60 443 → 46742 [ACK] Seq=4577 Ack=708 Win=64239 Len=0
36	2.888564231	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.

```

GET /basicauth/ HTTP/1.1\r\n
  [Expert Info (Chat/Sequence): GET /basicauth/ HTTP/1.1\r\n]
  [GET /basicauth/ HTTP/1.1\r\n]
  [Severity level: Chat]
  [Group: Sequence]
  Request Method: GET
  Request URI: /basicauth/
  Request Version: HTTP/1.1
  Host: cs338.jeffondich.com\r\n
  User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:91.0) Gecko/20100101 Firefox/91.0\r\n
  Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8\r\n
  Accept-Language: en-US,en;q=0.5\r\n
  Accept-Encoding: gzip, deflate\r\n
  DNT: 1\r\n
  Connection: keep-alive\r\n
  Upgrade-Insecure-Requests: 1\r\n
  \r\n
[Full request URI: http://cs338.jeffondich.com/basicauth/]
[HTTP request 1/5]
[Response in frame: 43]
[Next request in frame: 55]

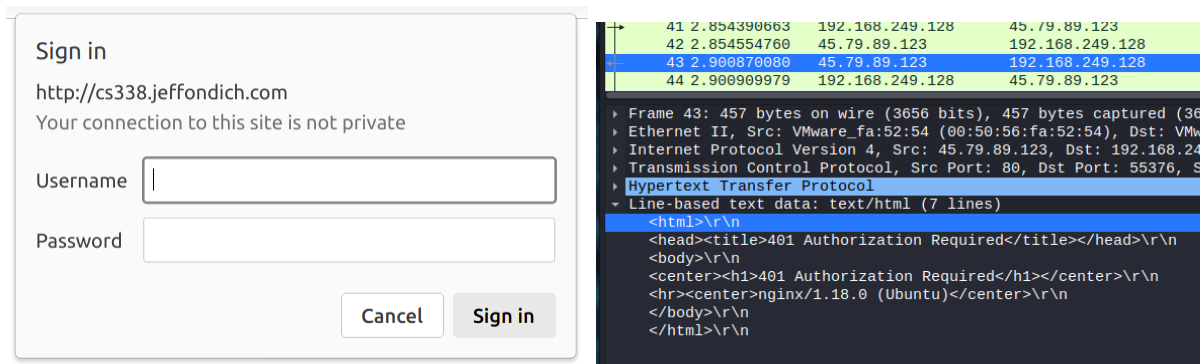
```

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.

No.	Time	Source	Destination	Protocol
43	2.900870080	45.79.89.123	192.168.249.128	HTTP
44	2.900909979	192.168.249.128	45.79.89.123	TCP
45	3.005471869	192.168.249.1	224.77.77.77	UDP
46	3.034223461	192.168.249.1	224.77.77.77	UDP
Internet Protocol Version 4, Src: 45.79.89.123, Dst: 192.168.249.128 Transmission Control Protocol, Src Port: 80, Dst Port: 55376, Seq: 1 Hypertext Transfer Protocol				
HTTP/1.1 401 Unauthorized\r\n [Expert Info (Chat/Sequence): HTTP/1.1 401 Unauthorized\r\n]				
Response Version: HTTP/1.1 Status Code: 401 [Status Code Description: Unauthorized]				

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).

No.	Time	Source	Destination	Protocol	Length	Info
43	2.900870080	45.79.89.123	192.168.249.128	HTTP	457	HTTP/1.1 401 Unauthorized
44	2.900909979	192.168.249.128	45.79.89.123	TCP	54	55376 → 80 [ACK] Seq=35
45	3.005471869	192.168.249.1	224.77.77.77	UDP	148	12177 → 12177 Len=106
46	3.034223461	192.168.249.1	224.77.77.77	UDP	148	12177 → 12177 Len=106
47	3.065079810	192.168.249.1	224.77.77.77	UDP	148	12177 → 12177 Len=106
48	3.580215111	192.168.249.1	239.255.255.250	SSDP	216	M-SEARCH * HTTP/1.1
49	3.846507837	192.168.249.128	72.21.91.29	TCP	54	57014 → 80 [ACK] Seq=1
50	3.846507837	72.21.91.29	192.168.249.128	TCP	54	80 → 57014 [ACK] Seq=1

Frame 45: 148 bytes on wire (1184 bits), 148 bytes captured (1184 bits) on interface eth0, id 0	
Ethernet II, Src: VMware_c0:00:08 (00:50:56:c0:00:08), Dst: IPv4mcast_4d:4d:4d (01:00:5e:4d:4d:4d)	
Internet Protocol Version 4, Src: 192.168.249.1, Dst: 224.77.77.77	
User Datagram Protocol, Src Port: 12177, Dst Port: 12177	
Data (106 bytes)	
Data: 3c415355535f41524d4f5552595f43524154453e3c4c414e20506f72743d223132313737...	
[Length: 106]	

0000	01 00 5e 4d 4d 00 50	56 c0 00 08 08 00 45 00	..^MMM.P V....E.
0010	00 86 6a b7 00 00 11	67 6b c0 a8 f9 01 e0 4d	..j.....gk.....M
0020	4d 4d 2f 91 2f 91 00 72	47 58 3c 41 53 55 53 5f	MM/./.r GX<ASUS
0030	41 52 4d 4f 55 52 59 5f	43 52 41 54 45 3e 3c 4c	ARMOURY_CRATE><L
0040	41 4e 20 50 6f 72 74 3d	22 31 32 31 37 37 22 20	AN Port= "12177"
0050	43 75 73 49 44 3d 22 42	35 31 42 36 37 45 44 2d	CusID="B 51B67ED-
0060	45 37 36 46 2d 34 33 45	36 2d 41 46 34 46 2d 34	E76F-43E 6-AF4F-4
0070	32 39 35 35 42 30 33 39	42 38 38 22 20 2f 3e 3c	2955B039 B88" /><
0080	2f 41 53 55 53 5f 41 52	4d 4f 55 52 59 5f 43 52	/ASUS_AR MOURY_CR
0090	41 54 45 3e		ATE>

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>).

No.	Time	Source	Destination	Protocol	Length	Info
55	7.582616613	192.168.249.128	45.79.89.123	HTTP	446	GET /basicauth/ HTTP/1.1
56	7.582902009	45.79.89.123	192.168.249.128	TCP	60	80 → 55376 [ACK] Seq=

▶ Frame 55: 446 bytes on wire (3568 bits), 446 bytes captured (3568 bits) on interface eth0, id 0

▶ Ethernet II, Src: VMware_0a:17:b8 (00:0c:29:0a:17:b8), Dst: VMware_fa:52:54 (00:50:56:fa:52:54)

▶ Internet Protocol Version 4, Src: 192.168.249.128, Dst: 45.79.89.123

▶ Transmission Control Protocol, Src Port: 55376, Dst Port: 80, Seq: 350, Ack: 404, Len: 392

▶ Hypertext Transfer Protocol

 ▶ GET /basicauth/ HTTP/1.1\r\n

 ▶ [Expert Info (Chat/Sequence): GET /basicauth/ HTTP/1.1\r\n]

 Request Method: GET

 Request URI: /basicauth/

 Request Version: HTTP/1.1

Host: cs338.jeffondich.com\r\n

User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:91.0) Gecko/20100101 Firefox/91.0\r\n

Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8\r\n

Accept-Language: en-US,en;q=0.5\r\n

Accept-Encoding: gzip, deflate\r\n

DNT: 1\r\n

Connection: keep-alive\r\n

Upgrade-Insecure-Requests: 1\r\n

▶ Authorization: Basic Y3MzMzg6cGFzc3dvcmQ=\r\n

 Credentials: cs338:password

 \r\n

 [Full request URI: http://cs338.jeffondich.com/basicauth/]

 [HTTP request 2/5]

 [Prev request in frame: 41]

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/”.

```

Transfer-Encoding: chunked\r\n
Connection: keep-alive\r\n
Content-Encoding: gzip\r\n
\r\n
[HTTP response 2/5]
[Time since request: 0.047292377 seconds]
[Prev request in frame: 41]
[Prev response in frame: 43]
[Request in frame: 55]
[Next request in frame: 59]
[Next response in frame: 63]
[Request URI: http://cs338.jeffondich.com/basicauth/]
HTTP chunked response
Content-encoded entity body (gzip): 205 bytes -> 509 bytes
File Data: 509 bytes
line-based text data: text/html (9 lines)
<html>\r\n
<head><title>Index of /basicauth/</title></head>\r\n
<body>\r\n
<h1>Index of /basicauth/</h1><hr><pre><a href="..">../</a>\r\n
<a href="amateurs.txt">amateurs.txt</a>
04-Apr-2022 14:10
<a href="armed-guards.txt">armed-guards.txt</a>
04-Apr-2022 14:10
<a href="dancing.txt">dancing.txt</a>
04-Apr-2022 14:10
</pre><hr></body>\r\n
</html>\r\n
00 0c 29 0a 17 b8 00 50 56 fa 52 54 08 00 45 00 ... P V RT E

```

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.”

No.	Time	Source	Destination	Protocol	Length	Info
58	7.629935290	192.168.249.128	45.79.89.123	TCP	54	55376 → 80 [ACK] Seq=742 Ack=808 Win=63837 Len=0
59	7.713827806	192.168.249.128	45.79.89.123	HTTP	363	GET /favicon.ico HTTP/1.1
60	7.714128501	45.79.89.123	192.168.249.128	TCP	60	80 → 55376 [ACK] Seq=808 Ack=1051 Win=64240 Len=0

```

Transmission Control Protocol, Src Port: 55376, Dst Port: 80, Seq: 742, Ack: 808, Len: 309
Hypertext Transfer Protocol
  GET /favicon.ico HTTP/1.1\r\n
    [Expert Info (Chat/Sequence): GET /favicon.ico HTTP/1.1\r\n]
    [GET /favicon.ico HTTP/1.1\r\n]
    [Severity level: Chat]
    [Group: Sequence]
    Request Method: GET
    Request URI: /favicon.ico
    Request Version: HTTP/1.1
    Host: cs338.jeffondich.com\r\n
    User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:91.0) Gecko/20100101 Firefox/91.0\r\n
    Accept: image/webp,*/*\r\n
    Accept-Language: en-US,en;q=0.5\r\n
    Accept-Encoding: gzip, deflate\r\n
    DNT: 1\r\n
    Connection: keep-alive\r\n
    Referer: http://cs338.jeffondich.com/basicauth/\r\n
    \r\n
    [Full request URI: http://cs338.jeffondich.com/favicon.ico]
    [HTTP request 3/5]
    [Prev request in frame: 55]
    [Response in frame: 63]
    [Next request in frame: 93]

```

Request for favicon

```

e      Source      Destination      Protocol      Length Info
54804279 45.79.89.123    192.168.249.128 TCP          60 80 → 55374 [ACK] Seq=1 Ack=2 Win=64239 Len=0
60211197 45.79.89.123    192.168.249.128 HTTP         383 HTTP/1.1 404 Not Found (text/html)
60233896 192.168.249.128 45.79.89.123    TCP          54 55376 → 80 [ACK] Seq=1051 Ack=1137 Win=63837 Len=0

> Frame 63: 383 bytes on wire (3064 bits), 383 bytes captured (3064 bits) on interface eth0, id 0
> Ethernet II, Src: Vmware_fa:52:54 (00:50:56:fa:52:54), Dst: Vmware_0a:17:b8 (00:0c:29:0a:17:b8)
> Internet Protocol Version 4, Src: 45.79.89.123, Dst: 192.168.249.128
> Transmission Control Protocol, Src Port: 80, Dst Port: 55376, Seq: 808, Ack: 1051, Len: 329
< Hypertext Transfer Protocol
  < HTTP/1.1 404 Not Found\r\n
    < [Expert Info (Chat/Sequence): HTTP/1.1 404 Not Found\r\n]
      [HTTP/1.1 404 Not Found\r\n]
      [Severity level: Chat]
      [Group: Sequence]
      Response Version: HTTP/1.1
      Status Code: 404
      [Status Code Description: Not Found]
      Response Phrase: Not Found
      Server: nginx/1.18.0 (Ubuntu)\r\n
      Date: Fri, 08 Apr 2022 01:31:39 GMT\r\n
      Content-Type: text/html\r\n
      Transfer-Encoding: chunked\r\n
      Connection: keep-alive\r\n
      Content-Encoding: gzip\r\n
      \r\n
      [HTTP response 3/5]
      [Time since request: 0.046383391 seconds]
      [Prev request in frame: 55]
      [Prev response in frame: 57]
0000  00 0c 29 0a 17 b8 00 50 56 fa 52 54 08 00 45 00  ... P V R T E

```

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.

```

[Next response in frame: 128]
[Request URI: http://cs338.jeffondich.com/basicauth/amateurs.txt]
File Data: 75 bytes
< Line-based text data: text/plain (3 lines)
  "Amateurs hack systems, professionals hack people."\n
  \n
  -- Bruce Schneier\n
0000  00 0c 29 0a 17 b8 00 50 56 fa 52 54 08 00 45 00  ... P V R T E

```

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.