Doug Pham
CS338

We will be looking in depth at what HTTP Basic Authentication is and learning more about the interactions between the browser and Jeff's nginx server.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	192.168.133.128	45.79.89.123	TCP	74	40102 → 80 [SYN] Seq=0
2	0.000056886	192.168.133.128	45.79.89.123	TCP	74	40110 → 80 [SYN] Seq=0
3	0.049351121	45.79.89.123	192.168.133.128	TCP	60	80 → 40110 [SYN, ACK]
4	0.049351482	45.79.89.123	192.168.133.128	TCP	60	80 → 40102 [SYN, ACK]
5	0.049401345	192.168.133.128	45.79.89.123	TCP	54	40110 → 80 [ACK] Seq=1
6	0.049432753	192.168.133.128	45.79.89.123	TCP	54	40102 → 80 [ACK] Seq=1

In the image above we see in these first six frames that there are two TCP 3-way handshakes being made between the client and the web server and vice versa.

7	0.049651897	192.168.133.128	45.79.89.123	HTTP	408	GET /basicauth/ HTTP/1.1
8	0.049990765	45.79.89.123	192.168.133.128	TCP	60	80 → 40110 [ACK] Seq=1 A
9	0.101651217	45.79.89.123	192.168.133.128	HTTP	457	HTTP/1.1 401 Unauthorize
10	0.101687976	192.168.133.128	45.79.89.123	TCP	54	40110 → 80 [ACK] Seq=355
11	5.157362884	192.168.133.128	45.79.89.123	TCP	54	40102 → 80 [FIN, ACK] Se
12	5.157712746	45.79.89.123	192.168.133.128	TCP	60	80 → 40102 [ACK] Seq=1 A
13	5.211329497	45.79.89.123	192.168.133.128	TCP	60	80 → 40102 [FIN, PSH, AC
14	5.211368450	192.168.133.128	45.79.89.123	TCP	54	40102 → 80 [ACK] Seq=2 A
15	10.155070098	192.168.133.128	45.79.89.123	TCP	54	[TCP Keep-Alive] 40110 →
16	11.179249666	192.168.133.128	45.79.89.123	TCP	54	[TCP Keep-Alive] 40110 →
17	11.179485586	45.79.89.123	192.168.133.128	TCP	60	[TCP Keep-Alive ACK] 80

```
Ethernet II, Src: VMware_4b:27:27 (00:0c:29:4b:27:27), Dst: VMware_f9:21:75 (00:50:56:f9:21:75)
Internet Protocol Version 4, Src: 192.168.133.128, Dst: 45.79.89.123
Transmission Control Protocol, Src Port: 40110, Dst Port: 80, Seq: 1, Ack: 1, Len: 354
Hypertext Transfer Protocol
  GET /basicauth/ HTTP/1.1\r\n
  Host: cs338.jeffondich.com\r\n
  User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0\r\n
  Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8\r\n
  Accept-Language: en-US,en;q=0.5\r\n
  Accept-Encoding: gzip, deflate\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 1/2]
```

Now in frame 7 we see the GET request for the HTML of the basicauth page from the client to the server. In frame 8 we see the acknowledgement from the server that the request has been received and in frame 9 we see the server's response which says it is unauthorized so this is where we see the 401 error being sent because we haven't entered the username and password and we are denied access to receiving the HTML. After some acknowledgements being sent back and forth and keeping the connection alive whilst there is nothing in the username and password field being entered, we now reach the point where once we enter the correct username and password, we see how the server responds.

```
18 14.391204703  192.168.133.128  45.79.89.123      HTTP  451 GET /basicauth/ HTTP/1.1
19 14.391624881  45.79.89.123     192.168.133.128   TCP    60 80 → 40110 [ACK] Seq=404 Ack=752
20 *REF*         45.79.89.123     192.168.133.128   HTTP  458 HTTP/1.1 200 OK  (text/html)
21 0.000037249   192.168.133.128  45.79.89.123      TCP    54 40110 → 80 [ACK] Seq=752 Ack=808

▸ Frame 18: 451 bytes on wire (3608 bits), 451 bytes captured (3608 bits) on interface eth0, id 0
▸ Ethernet II, Src: VMware_4b:27:27 (00:0c:29:4b:27:27), Dst: VMware_f9:21:75 (00:50:56:f9:21:75)
▸ Internet Protocol Version 4, Src: 192.168.133.128, Dst: 45.79.89.123
▸ Transmission Control Protocol, Src Port: 40110, Dst Port: 80, Seq: 355, Ack: 404, Len: 397
▾ Hypertext Transfer Protocol
  ▸ GET /basicauth/ HTTP/1.1\r\n
    Host: cs338.jeffondich.com\r\n
    User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0\r\n
    Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8\r\n
    Accept-Language: en-US,en;q=0.5\r\n
    Accept-Encoding: gzip, deflate\r\n
    Connection: keep-alive\r\n
    Upgrade-Insecure-Requests: 1\r\n
  ▸ Authorization: Basic Y3MzMzg6cGFzc3dvcmQ=\r\n
    \r\n
    [Full request URI: http://cs338.jeffondich.com/basicauth/]
    [HTTP request 2/2]
```

We now see that in frame 18 another query from the client to the server and it is similar to frame 7 from earlier which a GET request for the HTML of basicauth page (More on client/server messaging). Now that we have entered the username and password in, we now see under the Hypertext Transfer Protocol, a header that says Authorization: Basic Y3MzMzg6cGFzc3dvcmQ=\r\n. Essentially, what it means is the login credentials are being sent to the destination, which is the web browser in this case, and encoded with Base64. So if we decode this message we actually get that it is converted to cs338:password, which are the username and password we entered, conjoined by a colon. This is what the server does, which is decode this Base64 to get what was entered into the login and it will parse this at the colon to get which part is the username and password login. For more on the Basic Authentication click here.

```
20 46 69 72 65 66 6f 78  2f 31 31 35 2e 30 0d 0a   Firefox /115.0··
41 63 63 65 70 74 3a 20  74 65 78 74 2f 68 74 6d  Accept:  text/htm
6c 2c 61 70 70 6c 69 63  61 74 69 6f 6e 2f 78 68  l,applic ation/xh
74 6d 6c 2b 78 6d 6c 2c  61 70 70 6c 69 63 61 74  tml+xml, applicat
69 6f 6e 2f 78 6d 6c 3b  71 3d 30 2e 39 2c 69 6d  ion/xml; q=0.9,im
61 67 65 2f 61 76 69 66  2c 69 6d 61 67 65 2f 77  age/avif ,image/w
65 62 70 2c 2a 2f 2a 3b  71 3d 30 2e 38 0d 0a 41  ebp,*/*; q=0.8··A
63 63 65 70 74 2d 4c 61  6e 67 75 61 67 65 3a 20  ccept-La nguage:
65 6e 2d 55 53 2c 65 6e  3b 71 3d 30 2e 35 0d 0a  en-US,en ;q=0.5··
41 63 63 65 70 74 2d 45  6e 63 6f 64 69 6e 67 3a  Accept-E ncoding:
20 67 7a 69 70 2c 20 64  65 66 6c 61 74 65 0d 0a   gzip, d eflate··
43 6f 6e 6e 65 63 74 69  6f 6e 3a 20 6b 65 65 70  Connecti on: keep
2d 61 6c 69 76 65 0d 0a  55 70 67 72 61 64 65 2d  -alive·· Upgrade-
49 6e 73 65 63 75 72 65  2d 52 65 71 75 65 73 74  Insecure -Request
73 3a 20 31 0d 0a 41 75  74 68 6f 72 69 7a 61 74  s: 1··Au thorizat
69 6f 6e 3a 20 42 61 73  69 63 20 59 33 4d 7a 4d  ion: Bas ic Y3MzM
7a 67 36 63 47 46 7a 63  33 64 76 63 6d 51 3d 0d  zg6cGFzc 3dvcmQ=·
0a 0d 0a                                          ···
```

Above we see where in the hexadecimals, lies the data for the login credentials (highlighted in blue) and so now we see how these get sent over to the web browser. It is important to note though that this message is not encrypted. It is only encoded in

Base64 so though it is not plain text, it can easily be converted into the actual message. So you can imagine that if someone were able to get this, it would be easy to figure out the login credentials which are not secure at all. In order for the message to be encrypted, we would have to use HTTPS which makes sure that this communication is encrypted which happens during the TLS handshake process. This would make the communication between the client and server more secure as they would need the encryption key in order to get the real login credentials.

Going back to the previous image, we then see in frame 19 the acknowledgement being sent that it received the GET request and then in frame 20, we get the 200 status code with an OK and we see that the request got approved and the server sent over the HTML. How the HTML gets sent over is they use HTTP chunked response which you can see in the image below. What this does is it begins writing the contents of the HTML to the output stream before knowing how large the output is, which is advantageous because you won't need to buffer the whole page before you can start transmitting. (See [transfer codings](#) or [stackoverflow](#)) We also see that content-encoding gzip is used which helps in sending the HTTP content before it arrives to the client by compressing its contents, reducing its file size and making for a more optimal transmission (more on [gzip](#)). Once all this content is sent over, then we see the secret files that we now have access to on the basicauth page.