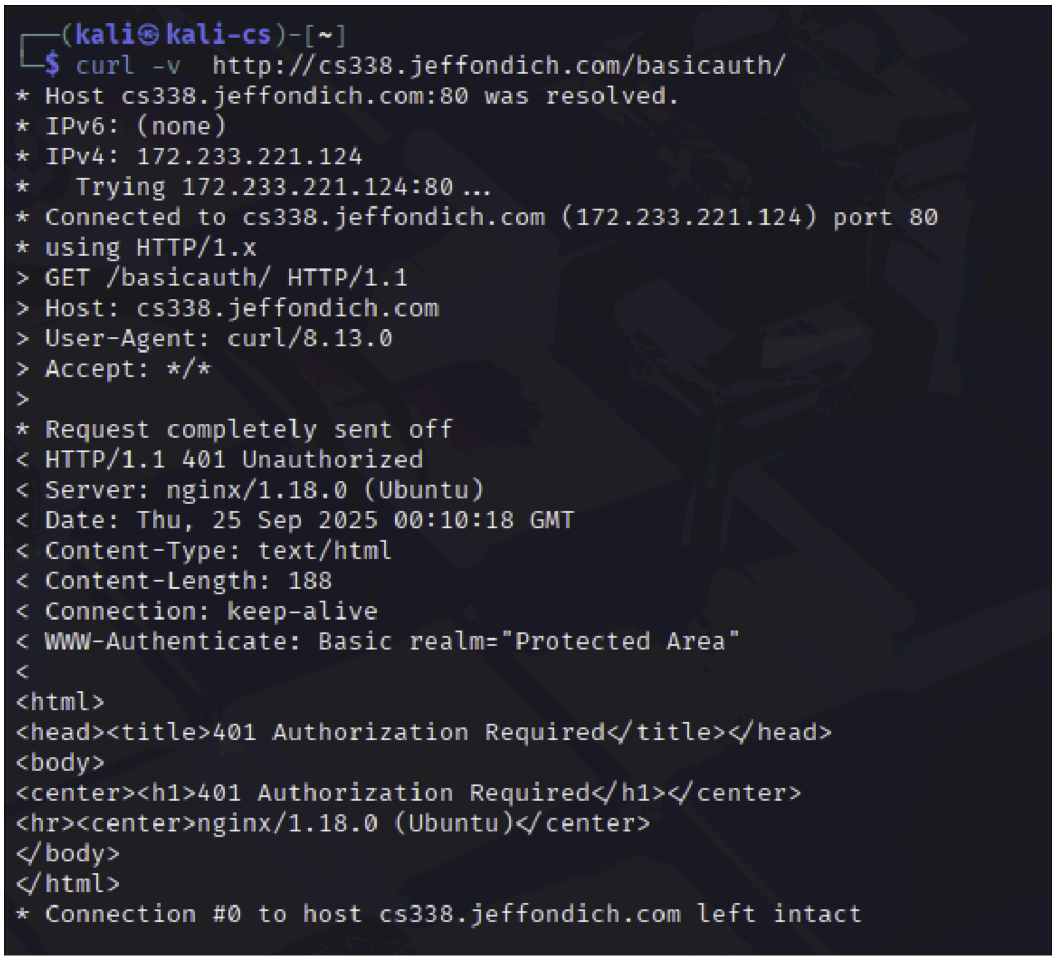
Alex Wcislo

A story about basic authentication

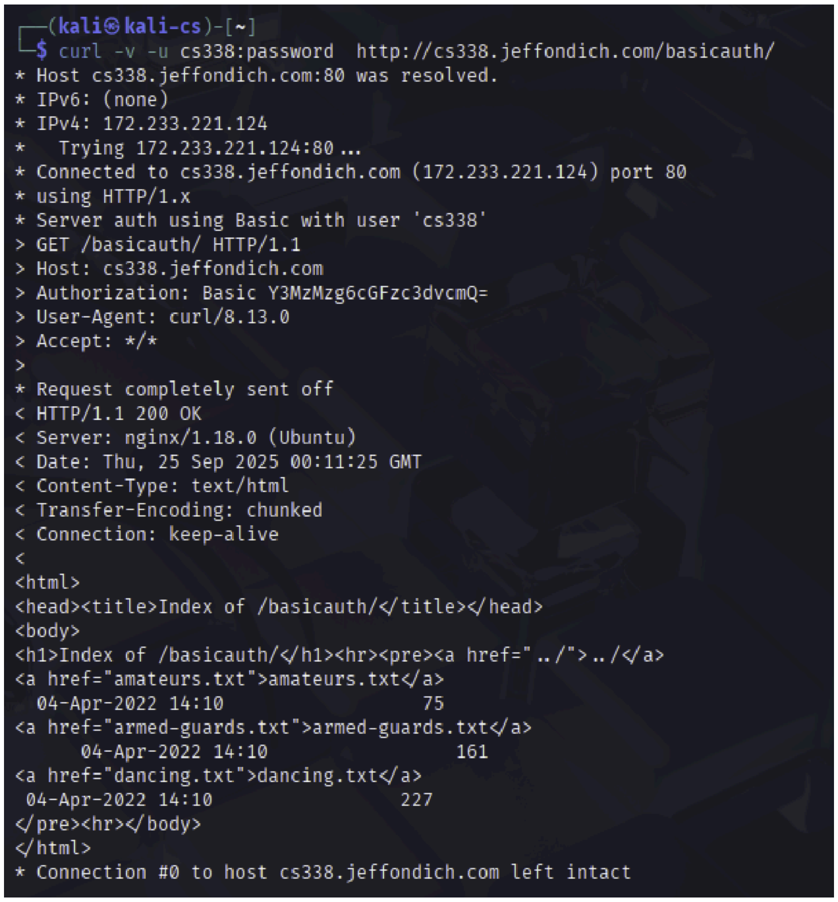
When trying to enter http://cs338.jeffondich.com/basicauth/ you get greeted with a quick authentication where a username and password (“cs338” and “password” respectively) is required to continue to the basicauth page. Given this page is part of a computer security course the next course of action was for me to begin figuring out what is going on behind the scenes (between the browser and the server). Before looking at the sequence of events that occurred, I decided to find out the relevant information within the terminal.



```
┌──(kali㉿kali-cs)-[~]
└─$ curl -v  http://cs338.jeffondich.com/basicauth/
* Host cs338.jeffondich.com:80 was resolved.
* IPv6: (none)
* IPv4: 172.233.221.124
*   Trying 172.233.221.124:80...
* Connected to cs338.jeffondich.com (172.233.221.124) port 80
* using HTTP/1.x
> GET /basicauth/ HTTP/1.1
> Host: cs338.jeffondich.com
> User-Agent: curl/8.13.0
> Accept: */*
>
* Request completely sent off
< HTTP/1.1 401 Unauthorized
< Server: nginx/1.18.0 (Ubuntu)
< Date: Thu, 25 Sep 2025 00:10:18 GMT
< Content-Type: text/html
< Content-Length: 188
< Connection: keep-alive
< WWW-Authenticate: Basic realm="Protected Area"
<
<html>
<head><title>401 Authorization Required</title></head>
<body>
<center><h1>401 Authorization Required</h1></center>
<hr><center>nginx/1.18.0 (Ubuntu)</center>
</body>
</html>
* Connection #0 to host cs338.jeffondich.com left intact
```

Right from the start we can see that the client (me) is trying to and succeeds in connecting to cs338.jeffondich.com at IP address 172.233.221.124. However when the client sends the GET request to access basicauth page, in return we get greeted with an error-401 Unauthorized. Among these response headers we see “WWW-Authenticate: Basic realm=”Protected Area” [I will dive more into this later].

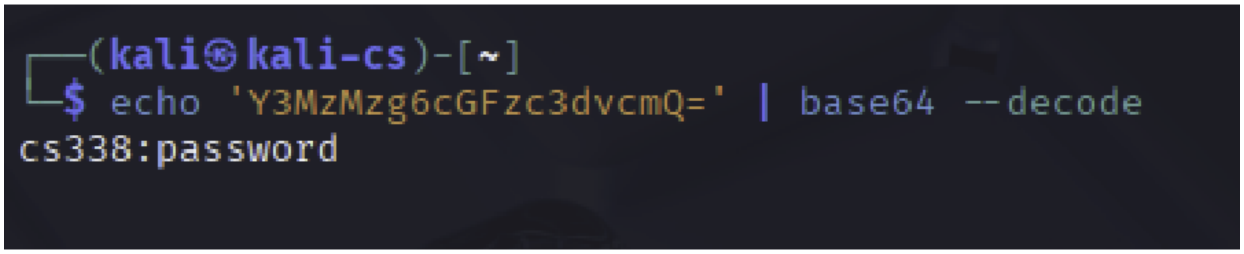
My next thought was to get the brief summary using the curl command with the proper username & password.



```
┌──(kali㉿kali-cs)-[~]
└─$ curl -v -u cs338:password  http://cs338.jeffondich.com/basicauth/
* Host cs338.jeffondich.com:80 was resolved.
* IPv6: (none)
* IPv4: 172.233.221.124
*   Trying 172.233.221.124:80 ...
* Connected to cs338.jeffondich.com (172.233.221.124) port 80
* using HTTP/1.x
* Server auth using Basic with user 'cs338'
> GET /basicauth/ HTTP/1.1
> Host: cs338.jeffondich.com
> Authorization: Basic Y3MzMzg6cGFzc3dvcmQ=
> User-Agent: curl/8.13.0
> Accept: */*
>
* Request completely sent off
< HTTP/1.1 200 OK
< Server: nginx/1.18.0 (Ubuntu)
< Date: Thu, 25 Sep 2025 00:11:25 GMT
< Content-Type: text/html
< Transfer-Encoding: chunked
< Connection: keep-alive
<
<html>
<head><title>Index of /basicauth/</title></head>
<body>
<h1>Index of /basicauth/</h1><hr><pre><a href="../">../</a>
<a href="amateurs.txt">amateurs.txt</a>
  04-Apr-2022 14:10                  75
<a href="armed-guards.txt">armed-guards.txt</a>
      04-Apr-2022 14:10                  161
<a href="dancing.txt">dancing.txt</a>
 04-Apr-2022 14:10                 227
</pre><hr></body>
</html>
* Connection #0 to host cs338.jeffondich.com left intact
```

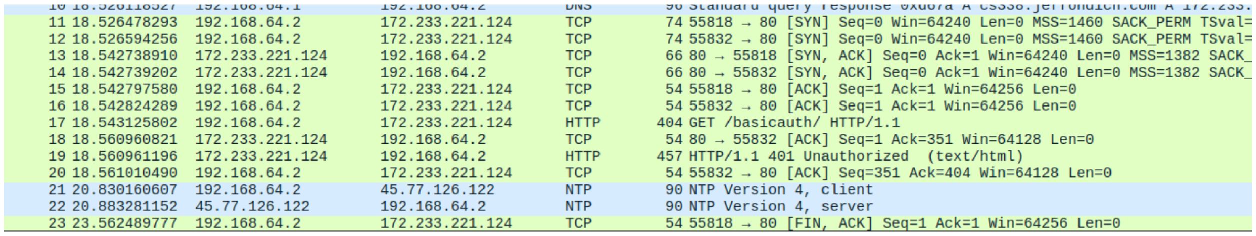
Similarly to above we see the initial connection to cs338.jeffondich.com. The difference this time, since we enter the username and password when the client sends GET /basicauth/ HTTP/1.1 we also see the header Authorization: Basic Y3MzMzg6cGFzc3dvcmQ= Which we know is the Base64 encoded version of the username:password that we entered. In response, we are greeted with HTTP/1.1 200 OK indicating that we passed the authentication process.

We can verify that Y3MzMzg6cGFzc3dvcmQ= is infact the username and password by running the following command which decodes it. Throughout the authentication process the password does not get encrypted in other ways using a key besides base64 encoding.



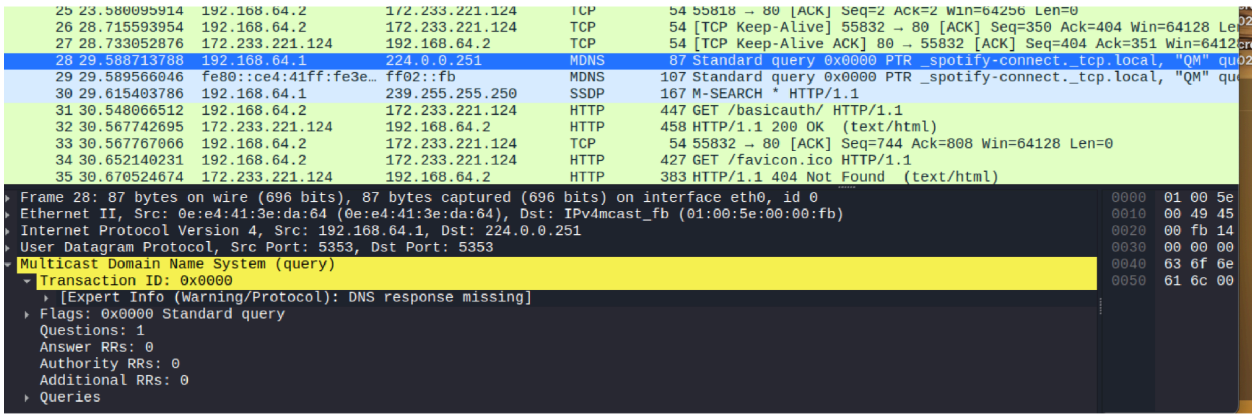
```
┌──(kali㉿kali-cs)-[~]
└─$ echo 'Y3MzMzg6cGFzc3dvcmQ=' | base64 --decode
cs338:password
```

While the sequence of events is simply shown above, diving into what is happening step by step allows us to see who is talking to who (browser, the cs338 nginx server, etc.)



```
   10 18.526118527  192.168.64.1      192.168.64.2      DNS     90 Standard query response 0xd67a A cs338.jeffondich.com A 172.233.
   11 18.526478293  192.168.64.2      172.233.221.124   TCP     74 55818 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=
   12 18.526594256  192.168.64.2      172.233.221.124   TCP     74 55832 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=
   13 18.542738910  172.233.221.124   192.168.64.2      TCP     66 80 → 55818 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1382 SACK_
   14 18.542739202  172.233.221.124   192.168.64.2      TCP     66 80 → 55832 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1382 SACK_
   15 18.542797580  192.168.64.2      172.233.221.124   TCP     54 55818 → 80 [ACK] Seq=1 Ack=1 Win=64256 Len=0
   16 18.542824289  192.168.64.2      172.233.221.124   TCP     54 55832 → 80 [ACK] Seq=1 Ack=1 Win=64256 Len=0
   17 18.543125802  192.168.64.2      172.233.221.124   HTTP   404 GET /basicauth/ HTTP/1.1
   18 18.560960821  172.233.221.124   192.168.64.2      TCP     54 80 → 55832 [ACK] Seq=1 Ack=351 Win=64128 Len=0
   19 18.560961196  172.233.221.124   192.168.64.2      HTTP   457 HTTP/1.1 401 Unauthorized  (text/html)
   20 18.561010490  192.168.64.2      172.233.221.124   TCP     54 55832 → 80 [ACK] Seq=351 Ack=404 Win=64128 Len=0
   21 20.830160607  192.168.64.2      45.77.126.122     NTP     90 NTP Version 4, client
   22 20.883281152  45.77.126.122     192.168.64.2      NTP     90 NTP Version 4, server
   23 23.562489777  192.168.64.2      172.233.221.124   TCP     54 55818 → 80 [FIN, ACK] Seq=1 Ack=1 Win=64256 Len=0
```

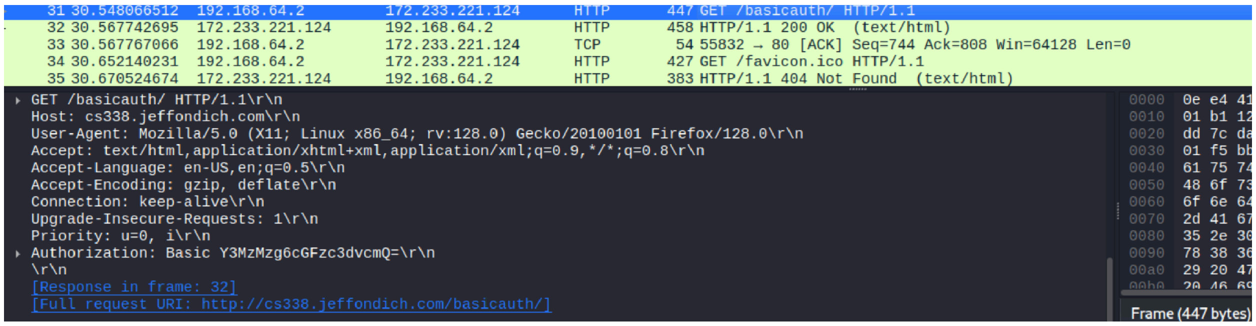
In the image above we can see after connecting to cs338.jeffondich.com at IP address 172.233.221.124 and the handshake finishes we (at 192.168.64.2) send a HTTP request (GET /basicauth/ HTTP/1.1) in packet 17. The nginx server responds with an acknowledgement of our request, but then follows up with an HTTP error 401 Unauthorized. The handshake is then closed with [FIN, ACK]s and then several [TCP Keep-Alive] which essentially just keep the page open giving time for the client to be authenticated.



```
   25 23.580095914  192.168.64.2      172.233.221.124   TCP     54 55818 → 80 [ACK] Seq=2 Ack=2 Win=64256 Len=0
   26 28.715593954  192.168.64.2      172.233.221.124   TCP     54 [TCP Keep-Alive] 55832 → 80 [ACK] Seq=350 Ack=404 Win=64128 Le
   27 28.733052876  172.233.221.124   192.168.64.2      TCP     54 [TCP Keep-Alive ACK] 80 → 55832 [ACK] Seq=404 Ack=351 Win=6412
   28 29.588713788  192.168.64.1      224.0.0.251       MDNS    87 Standard query 0x0000 PTR _spotify-connect._tcp.local, "QM" qu
   29 29.589566046  fe80::ce4:41ff:fe3e… ff02::fb        MDNS   107 Standard query 0x0000 PTR _spotify-connect._tcp.local, "QM" qu
   30 29.615403786  192.168.64.1      239.255.255.250   SSDP   167 M-SEARCH * HTTP/1.1
   31 30.548066512  192.168.64.2      172.233.221.124   HTTP   447 GET /basicauth/ HTTP/1.1
   32 30.567742695  172.233.221.124   192.168.64.2      HTTP   458 HTTP/1.1 200 OK  (text/html)
   33 30.567767066  192.168.64.2      172.233.221.124   TCP     54 55832 → 80 [ACK] Seq=744 Ack=808 Win=64128 Len=0
   34 30.652140231  192.168.64.2      172.233.221.124   HTTP   427 GET /favicon.ico HTTP/1.1
   35 30.670524674  172.233.221.124   192.168.64.2      HTTP   383 HTTP/1.1 404 Not Found  (text/html)

▸ Frame 28: 87 bytes on wire (696 bits), 87 bytes captured (696 bits) on interface eth0, id 0     0000  01 00 5e
▸ Ethernet II, Src: 0e:e4:41:3e:da:64 (0e:e4:41:3e:da:64), Dst: IPv4mcast_fb (01:00:5e:00:00:fb)   0010  00 49 45
▸ Internet Protocol Version 4, Src: 192.168.64.1, Dst: 224.0.0.251                                 0020  00 fb 14
▸ User Datagram Protocol, Src Port: 5353, Dst Port: 5353                                           0030  00 00 00
▾ Multicast Domain Name System (query)                                                             0040  63 6f 6e
  ▾ Transaction ID: 0x0000                                                                         0050  61 6c 00
    ▸ [Expert Info (Warning/Protocol): DNS response missing]
  ▸ Flags: 0x0000 Standard query
    Questions: 1
    Answer RRs: 0
    Authority RRs: 0
    Additional RRs: 0
  ▸ Queries
```

While unrelated to the basicauth page, this packet was unexpected as I see spotify-connect in the information section of packet 28. I was shocked that wireshark on the vm was picking up traffic from spotify on my local machine. This raised a question that after research I still have unanswered; How much traffic is being mixed up with a browser/clients requests at any given time?

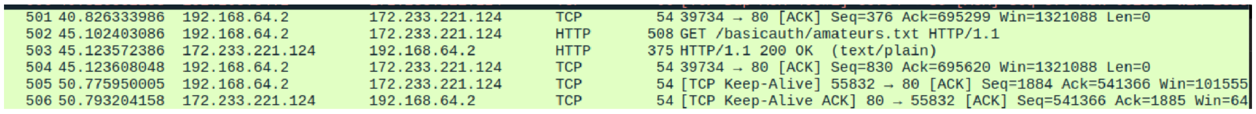


```
   31 30.548066512  192.168.64.2      172.233.221.124   HTTP   447 GET /basicauth/ HTTP/1.1
   32 30.567742695  172.233.221.124   192.168.64.2      HTTP   458 HTTP/1.1 200 OK  (text/html)
   33 30.567767066  192.168.64.2      172.233.221.124   TCP     54 55832 → 80 [ACK] Seq=744 Ack=808 Win=64128 Len=0
   34 30.652140231  192.168.64.2      172.233.221.124   HTTP   427 GET /favicon.ico HTTP/1.1
   35 30.670524674  172.233.221.124   192.168.64.2      HTTP   383 HTTP/1.1 404 Not Found  (text/html)

▸ GET /basicauth/ HTTP/1.1\r\n                                                                      0000  0e e4 41
  Host: cs338.jeffondich.com\r\n                                                                    0010  01 b1 12
  User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0\r\n            0020  dd 7c da
  Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8\r\n                       0030  01 f5 bb
  Accept-Language: en-US,en;q=0.5\r\n                                                               0040  61 75 74
  Accept-Encoding: gzip, deflate\r\n                                                                0050  48 6f 73
  Connection: keep-alive\r\n                                                                        0060  6f 6e 64
  Upgrade-Insecure-Requests: 1\r\n                                                                  0070  2d 41 67
  Priority: u=0, i\r\n                                                                              0080  35 2e 30
▸ Authorization: Basic Y3MzMzg6cGFzc3dvcmQ=\r\n                                                     0090  78 38 36
  \r\n                                                                                              00a0  29 20 47
  [Response in frame: 32]                                                                           00b0  20 46 69
  [Full request URI: http://cs338.jeffondich.com/basicauth/]
                                                                                                    Frame (447 bytes)
```

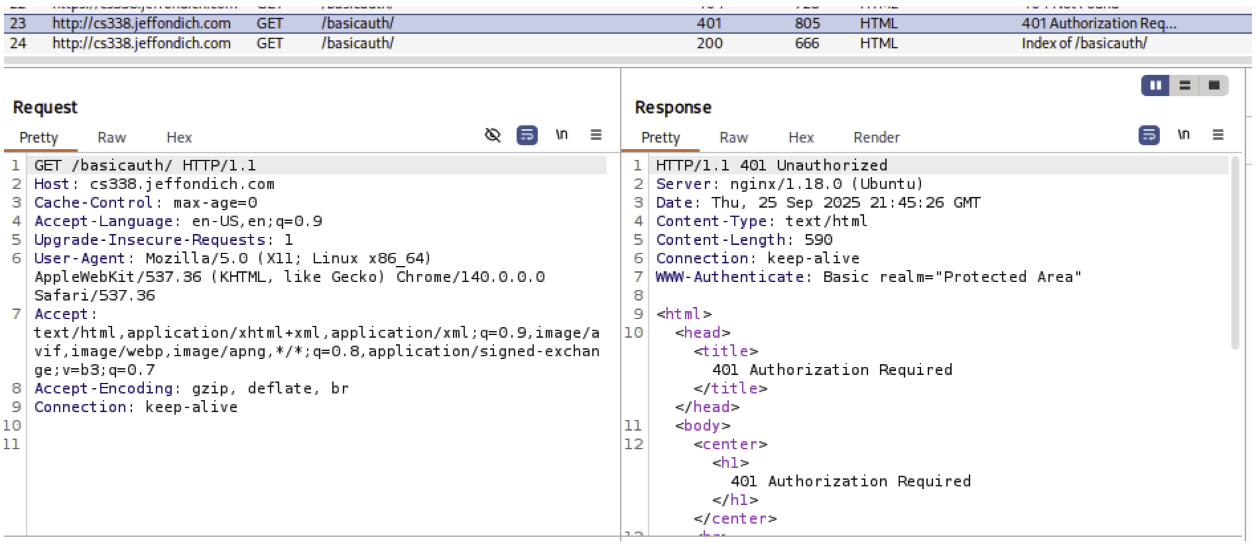
The next part in the sequence of events of the authentication process is when we click continue or press the enter button after we enter the username and password. We ask for the basicauth page but this time the nginx server tells the browser that the authentication has passed with the HTTP/1.1 200 ok response. Since we can also monitor where each request/response is coming from, we are able to identify that the password is sent from the browser to the server, which in turn checks that the password is appropriate.

Once we are on the page we see a lot of continuation packets as the elements load. This is followed be more TCP handshakes and [TCP Keep-Alive] as we access different pages such as when we send the query GET /basicauth/amateurs.txt HTTP/1.1.



```
501 40.826333986  192.168.64.2        172.233.221.124     TCP       54 39734 → 80 [ACK] Seq=376 Ack=695299 Win=1321088 Len=0
502 45.102403086  192.168.64.2        172.233.221.124     HTTP     508 GET /basicauth/amateurs.txt HTTP/1.1
503 45.123572386  172.233.221.124     192.168.64.2        HTTP     375 HTTP/1.1 200 OK  (text/plain)
504 45.123608048  192.168.64.2        172.233.221.124     TCP       54 39734 → 80 [ACK] Seq=830 Ack=695620 Win=1321088 Len=0
505 50.775950005  192.168.64.2        172.233.221.124     TCP       54 [TCP Keep-Alive] 55832 → 80 [ACK] Seq=1884 Ack=541366 Win=101555
506 50.793204158  172.233.221.124     192.168.64.2        TCP       54 [TCP Keep-Alive ACK] 80 → 55832 [ACK] Seq=541366 Ack=1885 Win=64
```

We can also look at this process through the Burp Suite browser, but it doesn't tell us much more than we already know. However, we do get a better look and understanding in regards to the authentication headers.



```
23  http://cs338.jeffondich.com  GET  /basicauth/  401  805  HTML  401 Authorization Req...
24  http://cs338.jeffondich.com  GET  /basicauth/  200  666  HTML  Index of /basicauth/
```

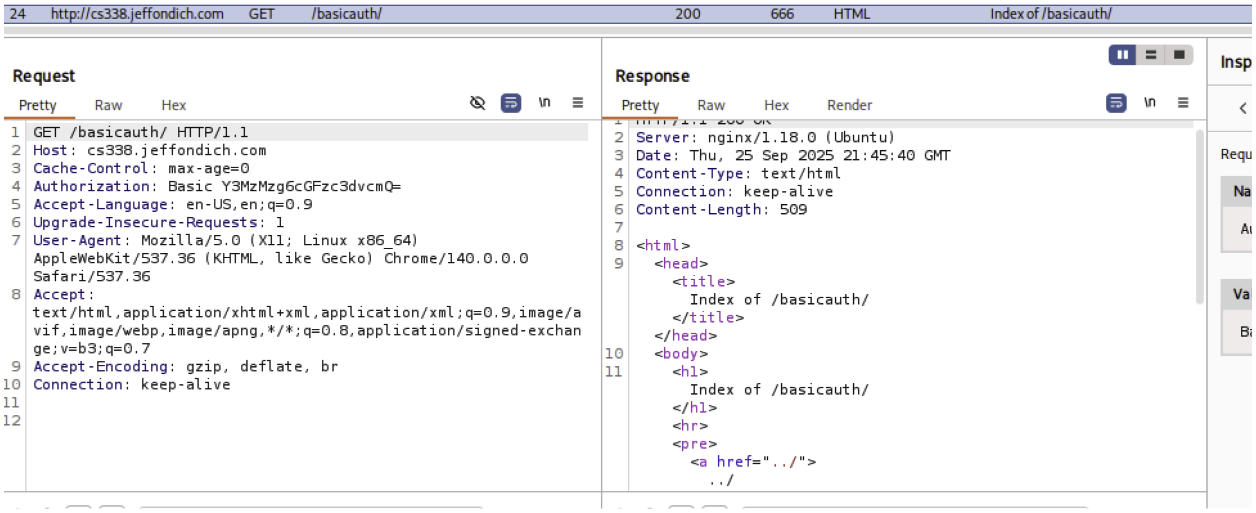
Request

```
1  GET /basicauth/ HTTP/1.1
2  Host: cs338.jeffondich.com
3  Cache-Control: max-age=0
4  Accept-Language: en-US,en;q=0.9
5  Upgrade-Insecure-Requests: 1
6  User-Agent: Mozilla/5.0 (X11; Linux x86_64)
   AppleWebKit/537.36 (KHTML, like Gecko) Chrome/140.0.0.0
   Safari/537.36
7  Accept:
   text/html,application/xhtml+xml,application/xml;q=0.9,image/a
   vif,image/webp,image/apng,*/*;q=0.8,application/signed-exchan
   ge;v=b3;q=0.7
8  Accept-Encoding: gzip, deflate, br
9  Connection: keep-alive
10
11
```

Response

```
1  HTTP/1.1 401 Unauthorized
2  Server: nginx/1.18.0 (Ubuntu)
3  Date: Thu, 25 Sep 2025 21:45:26 GMT
4  Content-Type: text/html
5  Content-Length: 590
6  Connection: keep-alive
7  WWW-Authenticate: Basic realm="Protected Area"
8
9  <html>
10   <head>
       <title>
         401 Authorization Required
       </title>
     </head>
11   <body>
12     <center>
         <h1>
           401 Authorization Required
         </h1>
       </center>
```

When we first send the request to access the basicauth page the response we get is 401 Unauthorized and the header in line 7 reads www-Authenticate: Basic realm="Protected Area"



```
24   http://cs338.jeffondich.com   GET   /basicauth/   200   666   HTML   Index of /basicauth/
```

```
GET /basicauth/ HTTP/1.1
Host: cs338.jeffondich.com
Cache-Control: max-age=0
Authorization: Basic Y3MzMzg6cGFzc3dvcmQ=
Accept-Language: en-US,en;q=0.9
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (X11; Linux x86_64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/140.0.0.0
Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/a
vif,image/webp,image/apng,*/*;q=0.8,application/signed-exchan
ge;v=b3;q=0.7
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
```

```
HTTP/1.1 200 OK
Server: nginx/1.18.0 (Ubuntu)
Date: Thu, 25 Sep 2025 21:45:40 GMT
Content-Type: text/html
Connection: keep-alive
Content-Length: 509

<html>
  <head>
    <title>
      Index of /basicauth/
    </title>
  </head>
  <body>
    <h1>
      Index of /basicauth/
    </h1>
    <hr>
    <pre>
      <a href="../">
        ../
```

In response the servers unauthorized message the browser/client responds with another authorization header - ‘Authorization: Basic Y3MzMzg6cGFzc3dvcmQ=’

These 2 authorization headers is the authentication process. When we first access the page and the server respons with 401 and www-Authenticate: Basic realm=”Protected Area” it is essentially asking the client to provide a username and password. The client responds by sending the authorization header with the Base64 encoded password and as we know from earlier the encoded part of the header translates to cs338:password. This means that while the password is sent as an encoded message, it is not encrypted. Base64 is reversible and this password could be captured by someone monitoring the given network.