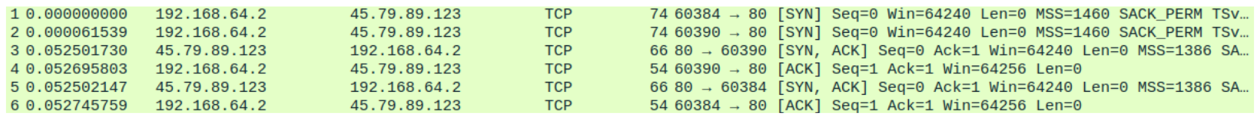
Miles Hoene-Langdon

September 20, 2023

CS338

HTTP's Basic Authentication: A story

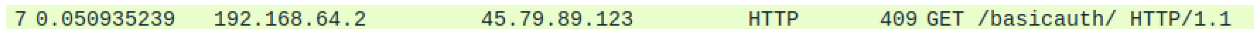
When the link http://cs338.jeffondich.com/basicauth/ is typed into the search bar of the browser, the browser initiates a DNS lookup to find the IP address associated with the domain name cs338.jeffondich.com and finds the IP address 45.79.89.123 associated with the domain. The browser, which is the client, then sends two [SYN] packets to the server associated with the IP 45.79.89.123. With these packets the client is requesting that a TCP connection be established so that the client can send data from the client ports 60384 and 60390 to port 80 of the server. This is the beginning of two TCP handshakes. The packets involved in this handshake are provided below.



```
1 0.000000000   192.168.64.2         45.79.89.123         TCP        74 60384 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSv…
2 0.000061539   192.168.64.2         45.79.89.123         TCP        74 60390 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSv…
3 0.052501730   45.79.89.123         192.168.64.2         TCP        66 80 → 60390 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1386 SA…
4 0.052695803   192.168.64.2         45.79.89.123         TCP        54 60390 → 80 [ACK] Seq=1 Ack=1 Win=64256 Len=0
5 0.052502147   45.79.89.123         192.168.64.2         TCP        66 80 → 60384 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1386 SA…
6 0.052745759   192.168.64.2         45.79.89.123         TCP        54 60384 → 80 [ACK] Seq=1 Ack=1 Win=64256 Len=0
```

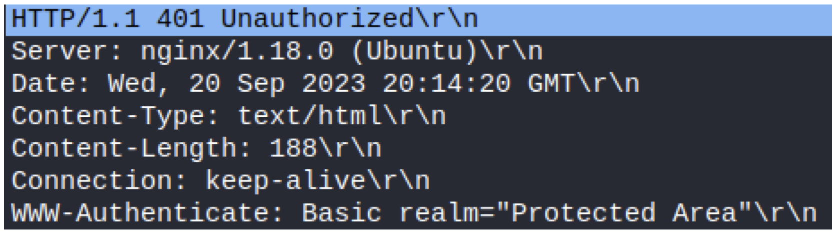
When the server receives the first packet it sends a [SYN, ACK] packet back to the client that acknowledges the request and sends a request of its own to be able to send data to the client. Immediately after receiving this, the client sends a [ACK] packet back to the server acknowledging the connection request. These two packets are then repeated for the second TCP connection on a different port. After this the TCP handshake is complete and a TCP connection is fully established.

Following the successful connection, the client sends a packet with a GET query to the server asking for the HTML file associated with the /basicauth/ page. This packet's info is shown below:



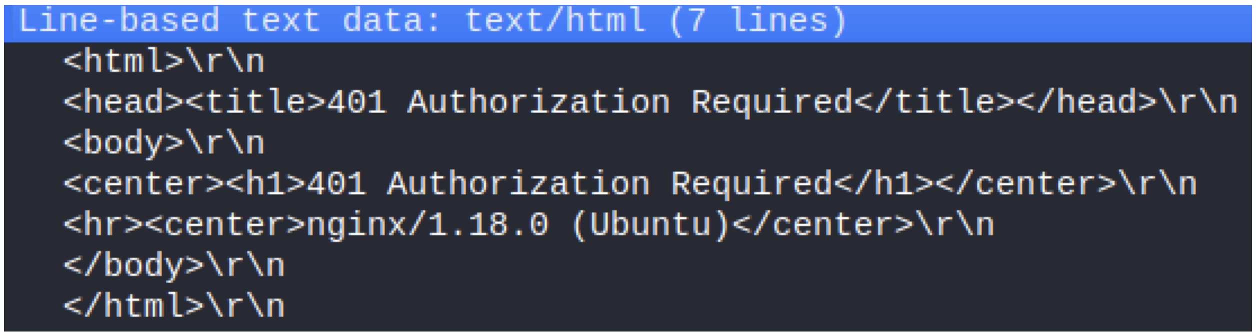
```
7 0.050935239   192.168.64.2         45.79.89.123         HTTP       409 GET /basicauth/ HTTP/1.1
```

Following this request the server sends back an [ACK] packet acknowledging that it received this request. This is immediately followed by the server replying with a packet, that is a challenge to the client, saying 401 unauthorized. This is the beginning of the 'Basic' HTTP authentication scheme. The exact HTTP response is provided below.



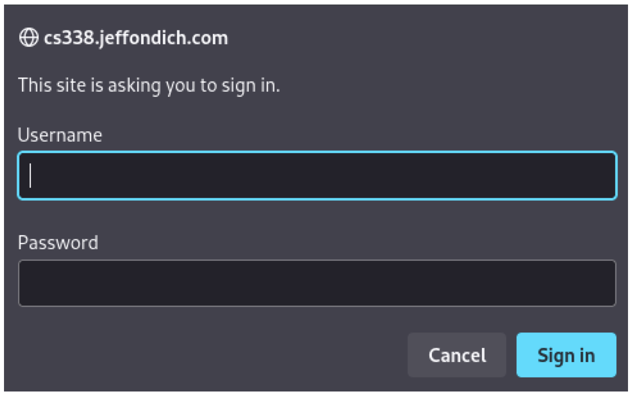
```
HTTP/1.1 401 Unauthorized\r\n
Server: nginx/1.18.0 (Ubuntu)\r\n
Date: Wed, 20 Sep 2023 20:14:20 GMT\r\n
Content-Type: text/html\r\n
Content-Length: 188\r\n
Connection: keep-alive\r\n
WWW-Authenticate: Basic realm="Protected Area"\r\n
```

The client was unable to access the realm “Protected Area” because the client has yet to be authenticated. The client needs to be authenticated in order to demonstrate its authorization to view the page. The server in this same packet (7) sends the user a simple HTML file that is shown below. The client then sends a packet acknowledgement that it received the message.



```
Line-based text data: text/html (7 lines)
    <html>\r\n
    <head><title>401 Authorization Required</title></head>\r\n
    <body>\r\n
    <center><h1>401 Authorization Required</h1></center>\r\n
    <hr><center>nginx/1.18.0 (Ubuntu)</center>\r\n
    </body>\r\n
    </html>\r\n
```

From here the exchange of packets between the client browser and the server stops as the browser requests a username and password from the user. This request is done through a pop up window with text fields for the Username and Password, as shown below.



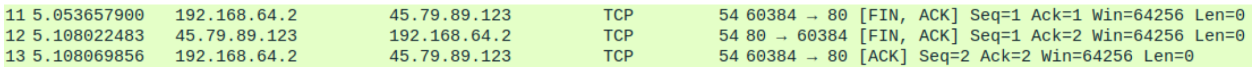
cs338.jeffondich.com

This site is asking you to sign in.

Username

Password

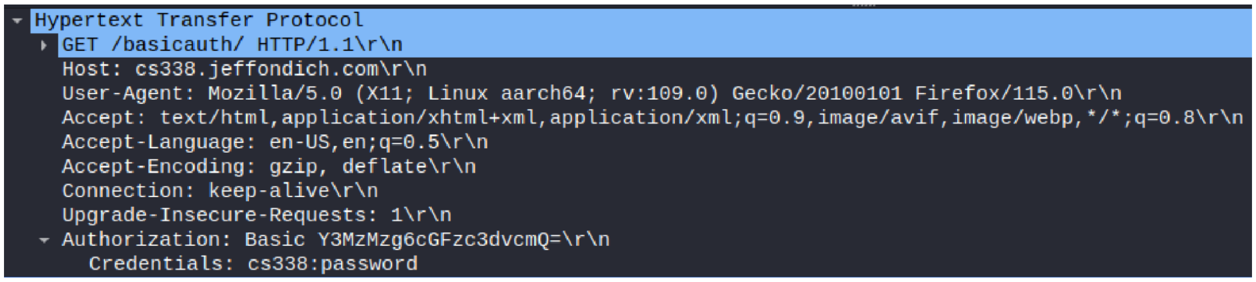
When the user types the username and password both the client and server send each other packets [FIN,ACK] packets to end the TCP connection between client port 60384 and server port 80. The client then sends a packet acknowledging that the TCP connection was closed. The information of this packet exchange is seen below.



```
11 5.053657900   192.168.64.2        45.79.89.123        TCP        54 60384 → 80 [FIN, ACK] Seq=1 Ack=1 Win=64256 Len=0
12 5.108022483   45.79.89.123        192.168.64.2        TCP        54 80 → 60384 [FIN, ACK] Seq=1 Ack=2 Win=64256 Len=0
13 5.108069856   192.168.64.2        45.79.89.123        TCP        54 60384 → 80 [ACK] Seq=2 Ack=2 Win=64256 Len=0
```

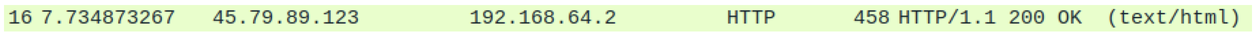
Next, with the received username and password from the user, the browser then creates the user-pass by concatenating the username, a single colon and the password. The browser then encodes the user pass into an octet sequence using the UTF-8 character encoding scheme. UTF-8 scheme encodes UCS characters as a varying number of octet values, where the number of octets, and the value of each, depend on the integer value assigned to the character. The browser finally creates the basic-credentials by encoding the octet sequence into ASCII characters using base64.

Now on the remaining TCP connection on port 68390, the client sends another packet with a GET query for the /basicauth/ page however this time it sends the basic credentials with the packet. The HTTP data is shown below.



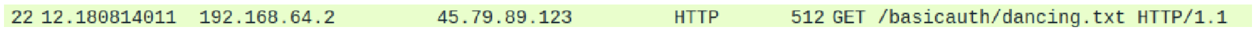
```
▾ Hypertext Transfer Protocol
  ▸ GET /basicauth/ HTTP/1.1\r\n
    Host: cs338.jeffondich.com\r\n
    User-Agent: Mozilla/5.0 (X11; Linux aarch64; rv:109.0) Gecko/20100101 Firefox/115.0\r\n
    Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8\r\n
    Accept-Language: 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
      Credentials: cs338:password
```

Here we can see the same GET request for the page /basicauth/ as earlier, but this time we have the field Authentication header. Under this header the word Basic is followed by a string of ASCII characters that encode the octet sequence that encodes the credentials. The specific credentials entered by the user here are cs338:password. The server responds to this packet from the client with an [ACK] packet to acknowledge that it has received the request. This is followed by another packet from the server with code 200 and the HTML file for the page /basicauth/. The server has verified that the username and password are correct by decoding the basic credentials. This indicates that the client has been successfully authenticated. Thus, the server now knows that the client is authorized to view the HTML file, so it sends it. Information on the packet where this happens is shown here.



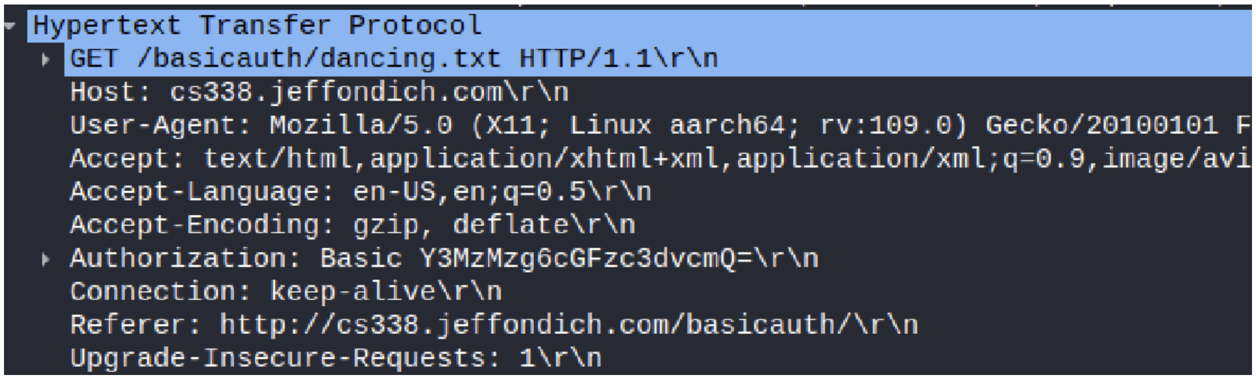
```
16 7.734873267    45.79.89.123          192.168.64.2          HTTP       458 HTTP/1.1 200 OK  (text/html)
```

The client then sends back a packet acknowledging that it received this data. Then, the user is then brought to the /basicauth/ page by the browser. The user may now select one of the files on the page. The same sequence of packets will be exchanged regardless of what file is clicked on with a different file name. For a particular example, if the user clicks on the link to the file dancing.txt, then the client sends a packet to the server with the following info:



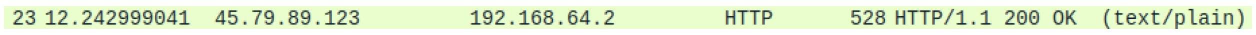
```
22 12.180814011  192.168.64.2          45.79.89.123          HTTP       512 GET /basicauth/dancing.txt HTTP/1.1
```

This is a GET request for the dancing.txt file on the basicauth page. If we look closer at this packet we will see that in the HTTP info, the same authorization code is delivered to the server, to ensure the client proves that it is authorized to receive the file. Thus, the browser stores the authorization code to be used later for later authorization. The HTTP info for all this is shown below.

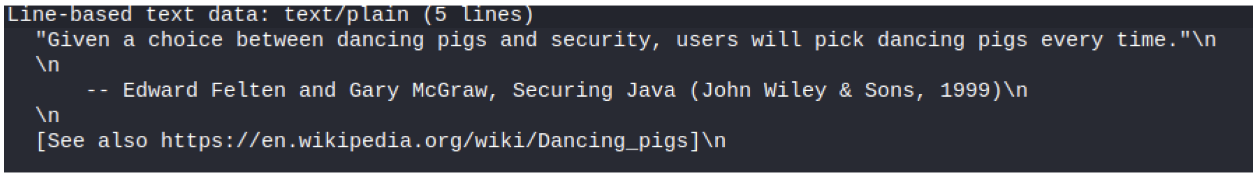


```
▾ Hypertext Transfer Protocol
  ▸ GET /basicauth/dancing.txt HTTP/1.1\r\n
    Host: cs338.jeffondich.com\r\n
    User-Agent: Mozilla/5.0 (X11; Linux aarch64; rv:109.0) Gecko/20100101 F
    Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avi
    Accept-Language: en-US,en;q=0.5\r\n
    Accept-Encoding: gzip, deflate\r\n
  ▸ Authorization: Basic Y3MzMzg6cGFzc3dvcmQ=\r\n
    Connection: keep-alive\r\n
    Referer: http://cs338.jeffondich.com/basicauth/\r\n
    Upgrade-Insecure-Requests: 1\r\n
```

After receiving the packet with the GET request for /basicauth/dancing.txt, the server sends back a packet with the line based text data for dancing.txt. The packet info and the text data are shown below.



```
23 12.242999041  45.79.89.123        192.168.64.2        HTTP      528 HTTP/1.1 200 OK  (text/plain)
```



```
Line-based text data: text/plain (5 lines)
  "Given a choice between dancing pigs and security, users will pick dancing pigs every time."\n
  \n
      -- Edward Felten and Gary McGraw, Securing Java (John Wiley & Sons, 1999)\n
  \n
  [See also https://en.wikipedia.org/wiki/Dancing_pigs]\n
```

For the final step in the process of accessing dancing.txt, the client returns a packet acknowledging that dancing.txt has been received. Finally, the user is presented with the contents of dance.txt in their browser. This concludes the story of what happens when secret files of the page /basicauth/ are accessed successfully through basic authentication.