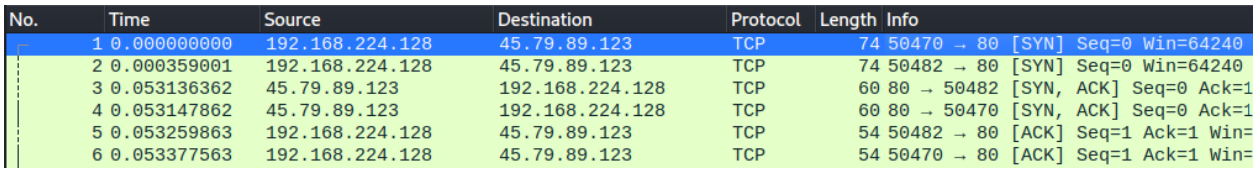
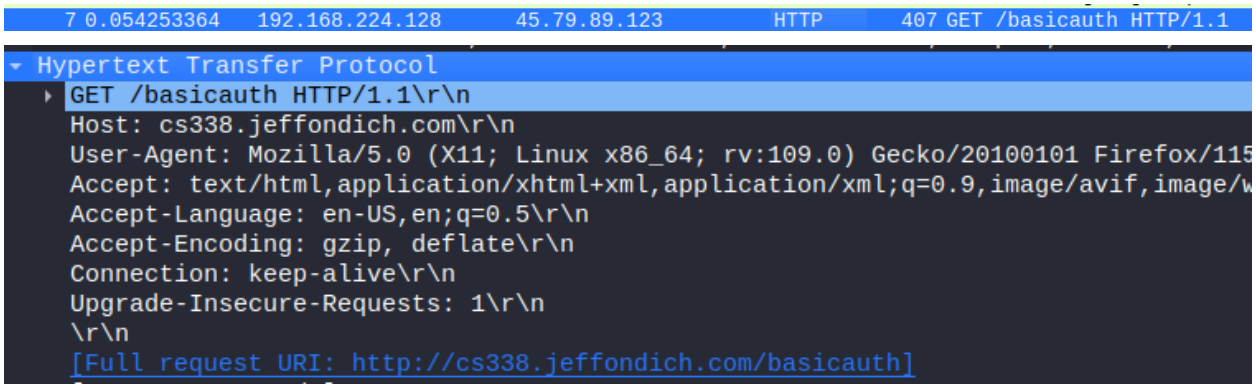
Kiri Salij
20 September 2023
Steps of HTTP Basic Authentication

1) Set up TCP connections
    a) Client sends two [SYN] packets (in my example, from ports 50470 and 50482). Server responds to both with [SYN, ACK] packets, and the client acknowledges both of those. Now two TCP connections are established between the client and server.



No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	192.168.224.128	45.79.89.123	TCP	74	50470 → 80 [SYN] Seq=0 Win=64240
2	0.000359001	192.168.224.128	45.79.89.123	TCP	74	50482 → 80 [SYN] Seq=0 Win=64240
3	0.053136362	45.79.89.123	192.168.224.128	TCP	60	80 → 50482 [SYN, ACK] Seq=0 Ack=1
4	0.053147862	45.79.89.123	192.168.224.128	TCP	60	80 → 50470 [SYN, ACK] Seq=0 Ack=1
5	0.053259863	192.168.224.128	45.79.89.123	TCP	54	50482 → 80 [ACK] Seq=1 Ack=1 Win=
6	0.053377563	192.168.224.128	45.79.89.123	TCP	54	50470 → 80 [ACK] Seq=1 Ack=1 Win=

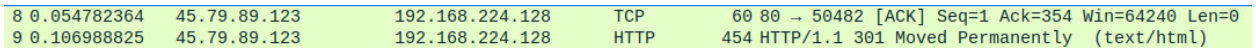
2) Client asks for http://cs338.jeffondich.com/basicauth (take 1).



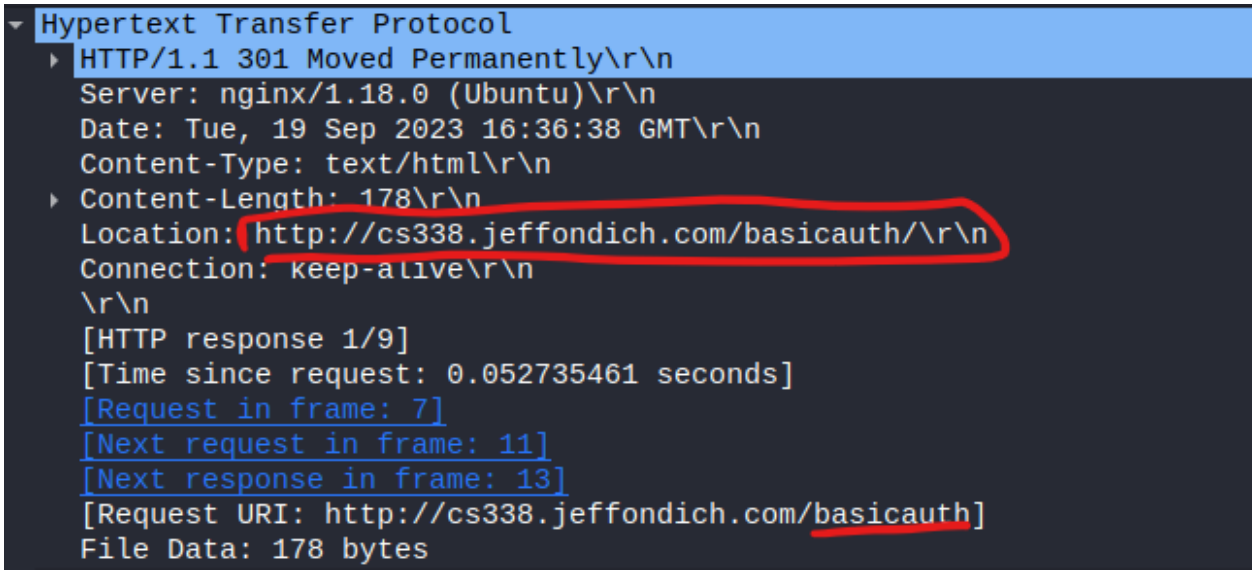
7	0.054253364	192.168.224.128	45.79.89.123	HTTP	407	GET /basicauth HTTP/1.1

```
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
  Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/w
  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]
```

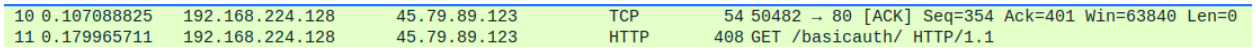
3) Server acknowledges the request and responds with another HTTP message saying that http://cs338.jeffondich.com/basicauth has moved permanently to http://cs338.jeffondich.com/basicauth/.
    a) At first I was very confused, but then I realized I typed in the url without the ending slash! However, it is interesting to note that the server is able to redirect the client to the correct url without the user even realizing. (I wonder if there would be ways to trick users into going to websites that they didn't want to go to by sending this "301 Moved Permanently" message and redirecting them to another website.)
    b) Marked in red is the difference between the requested URI and the location that it actually is at.



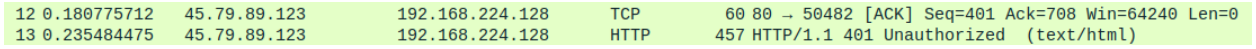
8	0.054782364	45.79.89.123	192.168.224.128	TCP	60	80 → 50482 [ACK] Seq=1 Ack=354 Win=64240 Len=0
9	0.106988825	45.79.89.123	192.168.224.128	HTTP	454	HTTP/1.1 301 Moved Permanently (text/html)

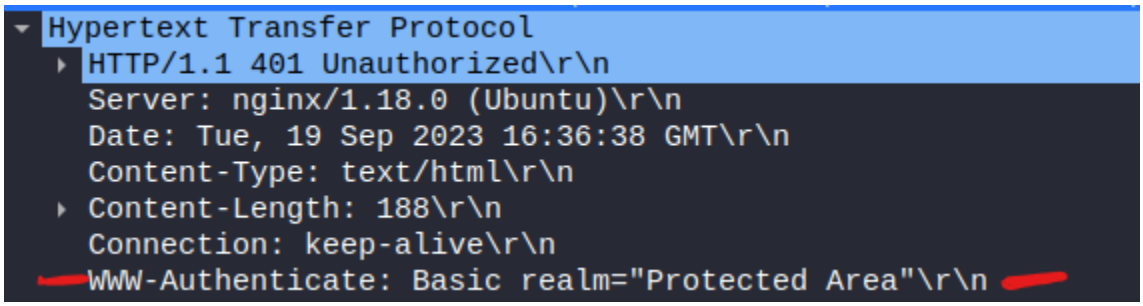


4) Client acknowledges the moved permanently packet, and requests the correct url (take 2).



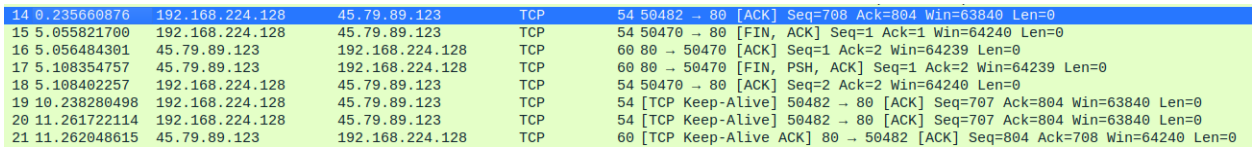
5) Server acknowledges the request and responds “oh sorry actually you are not authorized to view this content” in the form of a 401 Unauthorized packet.
    a) The packet contains the WWW-Authenticate header field which specifies the type of authentication scheme and the name of the protected space. The basic authentication scheme is a scheme where the user needs to authenticate themselves with a user-id and password. The credentials needed could vary depending on which protected space/realm the user was trying to access.
    b) The specifications for this interaction are explained in section 2 of RFC 7617 (https://datatracker.ietf.org/doc/html/rfc7617).
    c) In our example, we can see that Basic is specified as the authentication scheme and the realm is set to “Protected Area.”



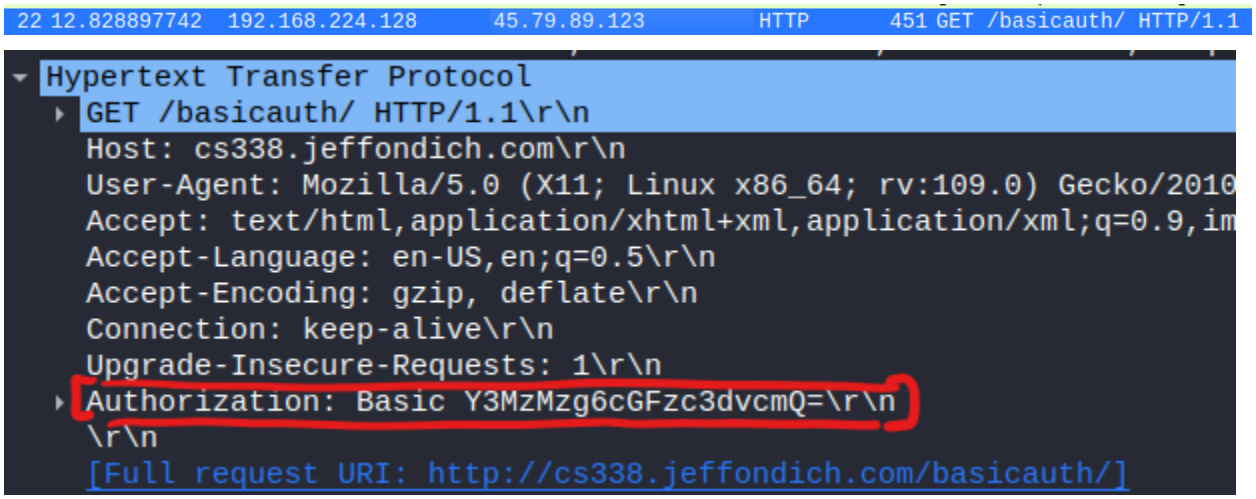


6) The next few packets are filler.
    a) The client acknowledges the 401 Unauthorized HTTP message.
    b) The client decides that it can close one of the TCP connections, so the next few packets relate to closing the connection from port 50470.

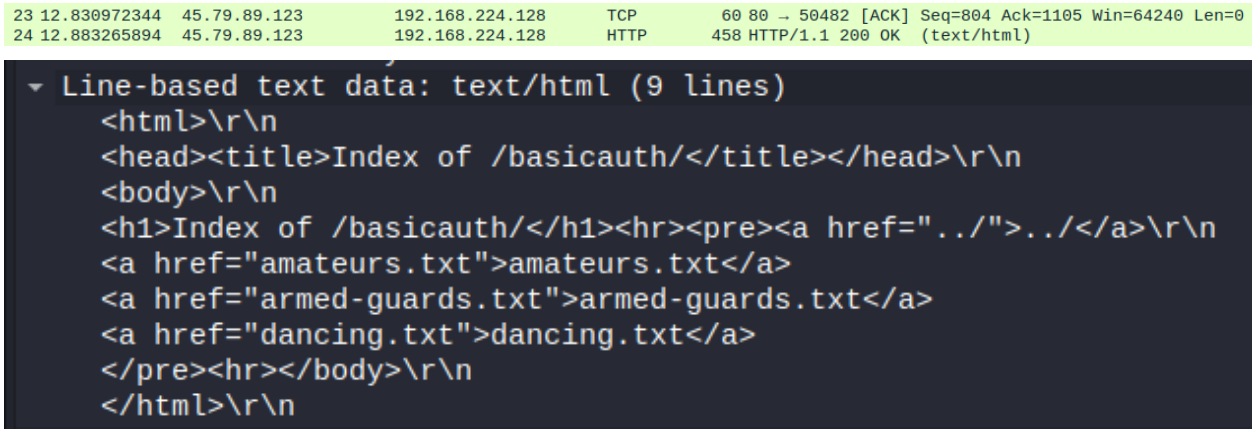
c) However, the client wants to keep the other TCP connection from port 50482 alive, so sends some packets to do so. While waiting for authentication by the user.



7) The client asks for /basicauth/ again (take 3). This time however, the user has inputted authentication and it is included in the packet (marked in red). The word Basic still signifies basic authentication scheme and the characters after are an encoded user-pass. The client created the user-pass by concatenating the user-id, a ':' colon character, and the password. This string is converted to an octet sequence and then into base64.
   a) This is all specified in RFC 7617.
   b) We can check what Y3MzMzg6cGFzc3dvcmQ= is by converting the base64 back to the original characters and lo and behold! We get cs338:password.



8) Server acknowledges the request, and finally says OK you can have this URI.
   a) We can even see that within the HTTP 200 OK packet is the actual content of the HTML file.



NOTES: Since wireshark only intercepts the communications between the client and the server, we could not see the internal mechanisms that the server was doing to check whether the user was authorized to view the content. However, someone looking at these communications for nefarious purposes would be able to see the 200 OK response and know that the authentication was authorized to view the content. Therefore, they could use the user-id and password that was sent, unencrypted, to view the restricted content. I wonder how often the basic authentication scheme is used in practice, especially with many more secure websites asking for 2-factor authentication.