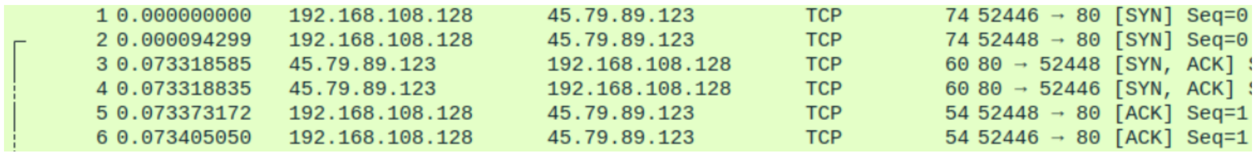
Jared Chen
4/7/2022
CS338 Jeff Onidch

HTTP'S BASIC AUTHENTICATION: A STORY

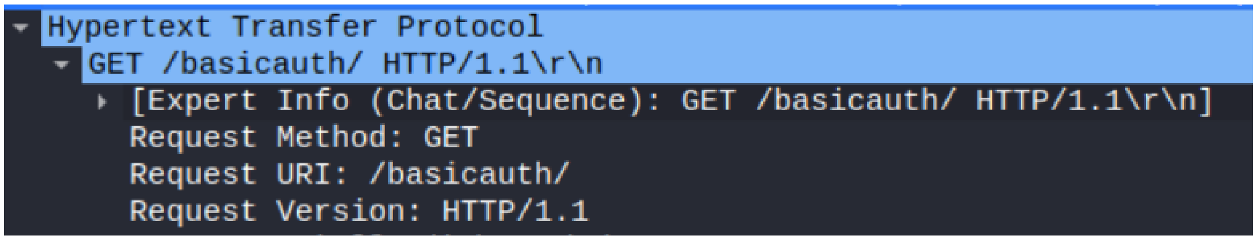
The packets here are recorded from an entry into the password-protected web page on firefox without incognito mode. I did try it with incognito mode and the main difference I saw was a chain of TLS1.2 protocols. The TLS1.2 seems to be used for end-to-end encryption, as keys were being passed back and forth, though I'm not sure if it makes the entire interaction more secure.

The first thing that occurs is the three-way TCP handshake, in this case, two of them seem to occur. Great, now everyone is friends! (TCP protocols observed in Wireshark)



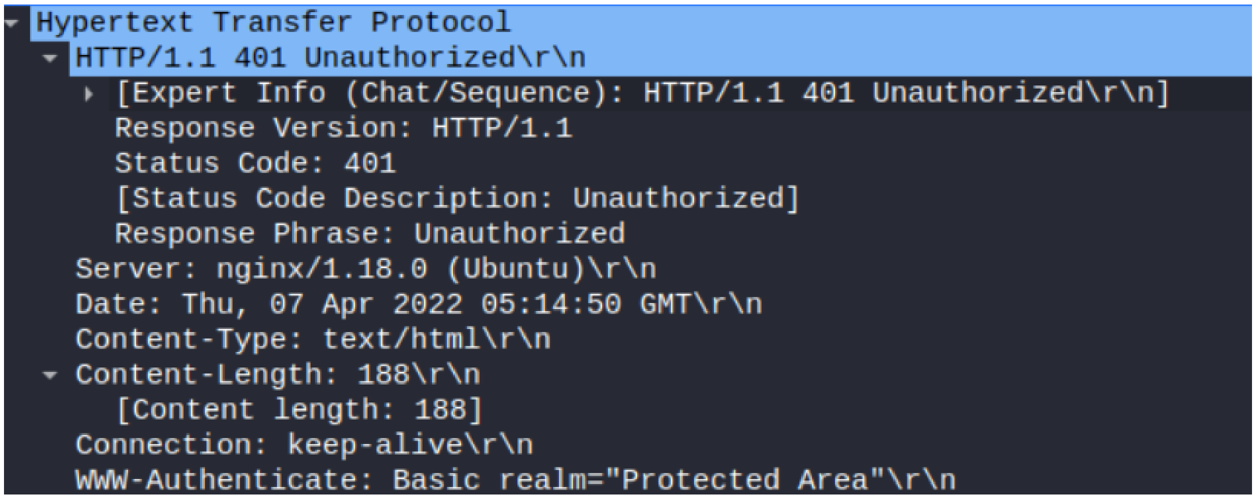
```
1 0.000000000   192.168.108.128      45.79.89.123         TCP        74 52446 → 80 [SYN] Seq=0
2 0.000094299   192.168.108.128      45.79.89.123         TCP        74 52448 → 80 [SYN] Seq=0
3 0.073318585   45.79.89.123         192.168.108.128      TCP        60 80 → 52448 [SYN, ACK]
4 0.073318835   45.79.89.123         192.168.108.128      TCP        60 80 → 52446 [SYN, ACK]
5 0.073373172   192.168.108.128      45.79.89.123         TCP        54 52448 → 80 [ACK] Seq=1
6 0.073405050   192.168.108.128      45.79.89.123         TCP        54 52446 → 80 [ACK] Seq=1
```

Then the client sends an HTTP protocol to the server. In the packet, we can see it's sending a GET request for basicauth, which is the page we are trying to access. This is a simple GET, like accessing any other web page.



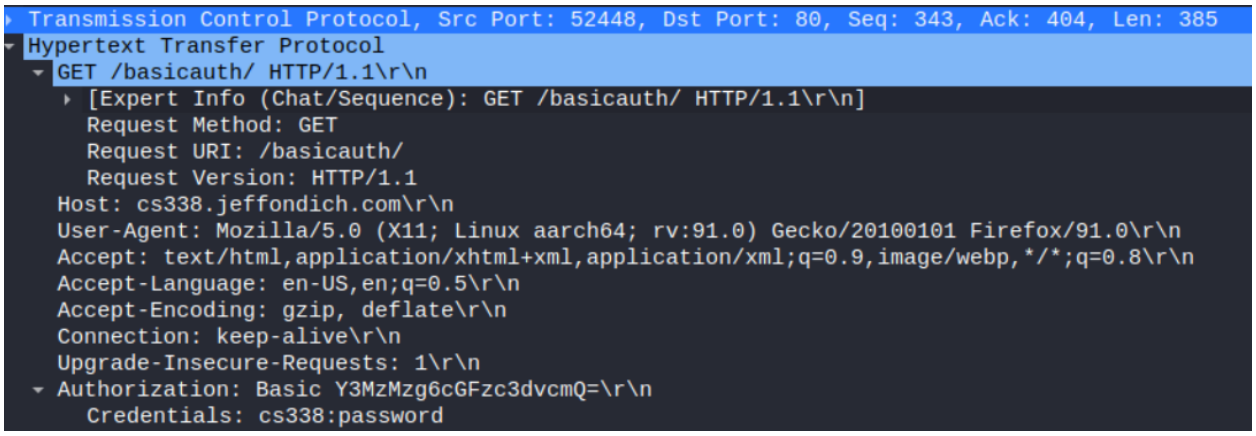
```
Hypertext Transfer Protocol
  GET /basicauth/ HTTP/1.1\r\n
    [Expert Info (Chat/Sequence): GET /basicauth/ HTTP/1.1\r\n]
    Request Method: GET
    Request URI: /basicauth/
    Request Version: HTTP/1.1
```

The server acknowledges this request and then responds with a Status Code 401. Status code 401 indicates that the client is trying to access a restricted area, and the packets containing the 401 Unauthorized code contain a www-Authenticate (authentication header) that indicates the type of "challenge" that must be completed in order to access the restricted area.



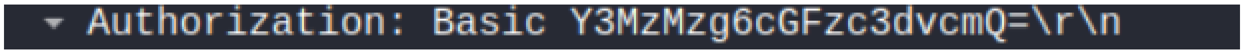
```
▾ Hypertext Transfer Protocol
  ▾ HTTP/1.1 401 Unauthorized\r\n
    ▸ [Expert Info (Chat/Sequence): HTTP/1.1 401 Unauthorized\r\n]
      Response Version: HTTP/1.1
      Status Code: 401
      [Status Code Description: Unauthorized]
      Response Phrase: Unauthorized
    Server: nginx/1.18.0 (Ubuntu)\r\n
    Date: Thu, 07 Apr 2022 05:14:50 GMT\r\n
    Content-Type: text/html\r\n
  ▾ Content-Length: 188\r\n
      [Content length: 188]
    Connection: keep-alive\r\n
    WWW-Authenticate: Basic realm="Protected Area"\r\n
```

So now the server wants information from the client to authenticate that they should be able to access the information. The client sends back a packet containing an HTTP header with another GET request. This time it also has an authorization header.



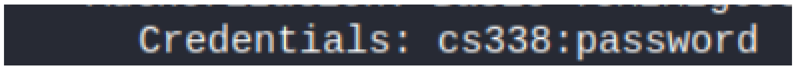
```
▸ Transmission Control Protocol, Src Port: 52448, Dst Port: 80, Seq: 343, Ack: 404, Len: 385
▾ Hypertext Transfer Protocol
  ▾ GET /basicauth/ HTTP/1.1\r\n
    ▸ [Expert Info (Chat/Sequence): GET /basicauth/ HTTP/1.1\r\n]
      Request Method: GET
      Request URI: /basicauth/
      Request Version: HTTP/1.1
    Host: cs338.jeffondich.com\r\n
    User-Agent: Mozilla/5.0 (X11; Linux aarch64; 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
    Connection: keep-alive\r\n
    Upgrade-Insecure-Requests: 1\r\n
  ▾ Authorization: Basic Y3MzMzg6cGFzc3dvcmQ=\r\n
      Credentials: cs338:password
```

The authorization header contains a representation of the username and password, formatted, mashed together, and then encoded in base64. In this case, our authentication "key" is: "Basic Y3MzMzg6cGFzc3dvcmQ=". The basic indicates that this is used for basic authentication.



```
▾ Authorization: Basic Y3MzMzg6cGFzc3dvcmQ=\r\n
```

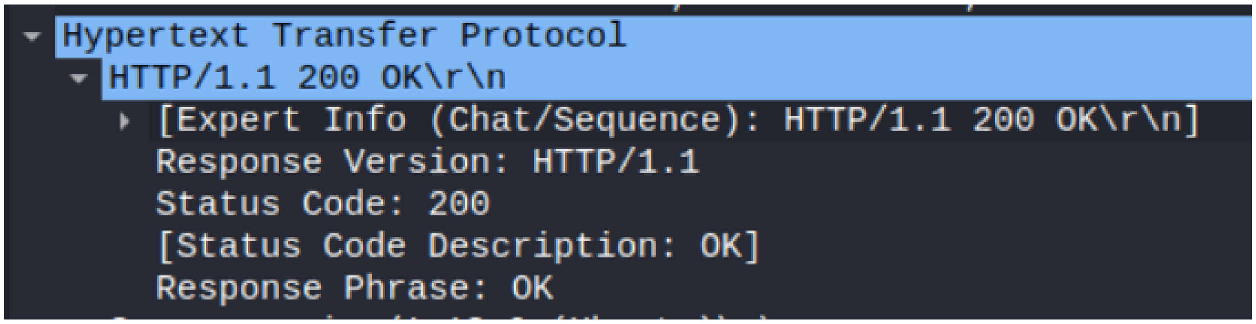
However it is important to note that this information is not encrypted, in fact, Wireshark is kind enough to interpret this line of characters into human-readable text, yielding our credentials line:



```
Credentials: cs338:password
```

Therefore, anyone that can see our packets can easily take out our username and password and get past the cs338 pages defenses.

Finally, the server sends another packet back with an HTTP header containing status code 200: OK. This means we've successfully gained access to the knowledge on the password-protected cs338 site!



```
▾ Hypertext Transfer Protocol
  ▾ HTTP/1.1 200 OK\r\n
    ▸ [Expert Info (Chat/Sequence): HTTP/1.1 200 OK\r\n]
      Response Version: HTTP/1.1
      Status Code: 200
      [Status Code Description: OK]
      Response Phrase: OK
```

Sources:

- https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/WWW-Authenticate
- https://blog.gigamon.com/2021/07/14/what-is-tls-1-2-and-why-should-you-still-care/
- https://www.youtube.com/watch?v=EeNzWUcPaFY