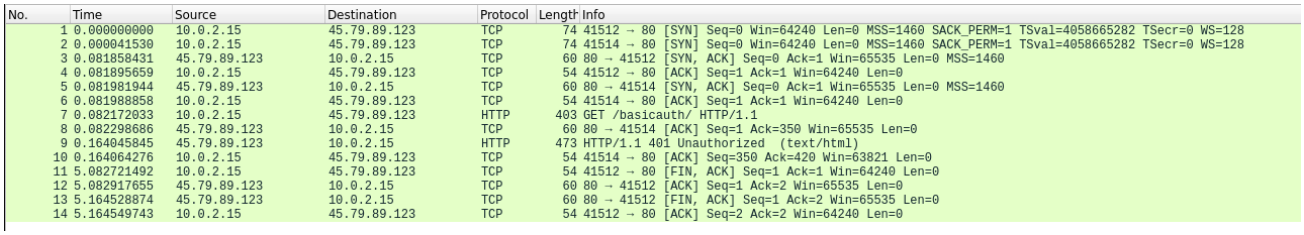
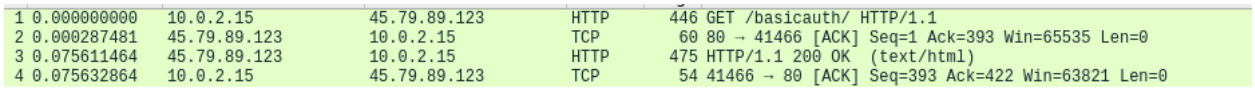
When we clicked on the link, the following frames showed up:



No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	10.0.2.15	45.79.89.123	TCP	74	41512 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=4058665282 TSecr=0 WS=128
2	0.000041530	10.0.2.15	45.79.89.123	TCP	74	41514 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=4058665282 TSecr=0 WS=128
3	0.081858431	45.79.89.123	10.0.2.15	TCP	60	80 → 41512 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460
4	0.081895659	10.0.2.15	45.79.89.123	TCP	54	41512 → 80 [ACK] Seq=1 Ack=1 Win=64240 Len=0
5	0.081981944	45.79.89.123	10.0.2.15	TCP	60	80 → 41514 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460
6	0.081988858	10.0.2.15	45.79.89.123	TCP	54	41514 → 80 [ACK] Seq=1 Ack=1 Win=64240 Len=0
7	0.082172033	10.0.2.15	45.79.89.123	HTTP	403	GET /basicauth/ HTTP/1.1
8	0.082298686	45.79.89.123	10.0.2.15	TCP	60	80 → 41514 [ACK] Seq=1 Ack=350 Win=65535 Len=0
9	0.164045845	45.79.89.123	10.0.2.15	HTTP	473	HTTP/1.1 401 Unauthorized (text/html)
10	0.164064276	10.0.2.15	45.79.89.123	TCP	54	41514 → 80 [ACK] Seq=350 Ack=420 Win=63821 Len=0
11	5.082721492	10.0.2.15	45.79.89.123	TCP	54	41512 → 80 [FIN, ACK] Seq=1 Ack=1 Win=64240 Len=0
12	5.082917655	45.79.89.123	10.0.2.15	TCP	60	80 → 41512 [ACK] Seq=1 Ack=2 Win=65535 Len=0
13	5.164528874	45.79.89.123	10.0.2.15	TCP	60	80 → 41512 [FIN, ACK] Seq=1 Ack=2 Win=65535 Len=0
14	5.164549743	10.0.2.15	45.79.89.123	TCP	54	41512 → 80 [ACK] Seq=2 Ack=2 Win=64240 Len=0

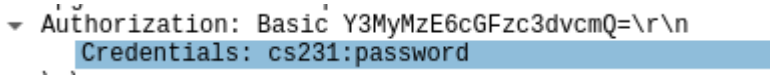
The browser sent a GET request to the server, but the server sent back a 401 unauthorized error, because we had not provided the username and password. “WWW-Authenticate: Basic realm="Protected Area" was in the content of this frame, asking for authentication.

When we gave the username and the right password, on Wireshark the following frames showed up.



1	0.000000000	10.0.2.15	45.79.89.123	HTTP	446	GET /basicauth/ HTTP/1.1
2	0.000287481	45.79.89.123	10.0.2.15	TCP	60	80 → 41466 [ACK] Seq=1 Ack=393 Win=65535 Len=0
3	0.075611464	45.79.89.123	10.0.2.15	HTTP	475	HTTP/1.1 200 OK (text/html)
4	0.075632864	10.0.2.15	45.79.89.123	TCP	54	41466 → 80 [ACK] Seq=393 Ack=422 Win=63821 Len=0

In the content of the first frame, the client sent a GET request along with the encoded but unencrypted string of the username and password as Y3MyMzE6cGFzc3dvcmQ= in clear text:



This authentication string was produced by first constructing the user-pass by concatenating the user-id, a single colon (":") character, and the password, encoding it into a octet sequence, and then encoding this octet sequence using Base64 ([RFC4648], Section 4) into a sequence of US-ASCII characters ([RFC0020]) ('Basic' HTTP Authentication Scheme, Section 2.1).

A weakness of HTTP’s basic authentication is that this clear text can be seen by other people on the same ethernet. This sequence can also be easily decoded, making the credentials vulnerable to leaking.

The server then acknowledged the access (the second frame) and sent back the website content (the third frame):

```
Line-based text data: text/html (9 lines)
    <html>\r\n
    <head><title>Index of /basicauth/</title></head>\r\n
    <body bgcolor="white">\r\n
    <h1>Index of /basicauth/</h1><hr><pre><a href="../">../</a>\r\n
    <a href="amateurs.txt">amateurs.txt</a>                                    30-Mar-2021 16:58                  75\r\n
    <a href="concrete.txt">concrete.txt</a>                                    30-Mar-2021 16:58                 161\r\n
    <a href="pigs.txt">pigs.txt</a>                                          30-Mar-2021 17:09                 227\r\n
    </pre><hr></body>\r\n
    </html>\r\n
```

Then the client acknowledge the reception (the fourth frame).

Incorrect Credentials

9	0.153421278	45.79.89.123	10.0.2.15	HTTP	473 HTTP/1.1 401 Unauthorized (text/html)
10	0.153463893	10.0.2.15	45.79.89.123	TCP	54 38500 → 80 [ACK] Seq=342 Ack=420 Win=63821 Len=0
11	6.078864365	10.0.2.15	45.79.89.123	TCP	54 38502 → 80 [FIN, ACK] Seq=1 Ack=1 Win=64240 Len=0
12	6.080487334	45.79.89.123	10.0.2.15	TCP	60 80 → 38502 [ACK] Seq=1 Ack=2 Win=65535 Len=0
13	6.142023003	45.79.89.123	10.0.2.15	TCP	60 80 → 38502 [FIN, ACK] Seq=1 Ack=2 Win=65535 Len=0
14	6.142050460	10.0.2.15	45.79.89.123	TCP	54 38502 → 80 [ACK] Seq=2 Ack=2 Win=64240 Len=0
15	9.333877579	10.0.2.15	45.79.89.123	HTTP	434 GET /basicauth/ HTTP/1.1
16	9.335495693	45.79.89.123	10.0.2.15	TCP	60 80 → 38500 [ACK] Seq=420 Ack=722 Win=65535 Len=0
17	9.398555267	45.79.89.123	10.0.2.15	HTTP	473 HTTP/1.1 401 Unauthorized (text/html)

When we supplied an incorrect set of login credentials, we received a response with the “401 Unauthorized” header, which was nearly identical to the initial response from the server that we received before even entering any credentials. In the context of the screenshot above, those frames were 9 and 17. A sample of the relevant portion of those frame is below:

```
HTTP/1.1 401 Unauthorized\r\n
Server: nginx/1.14.0 (Ubuntu)\r\n
Date: Wed, 07 Apr 2021 22:12:32 GMT\r\n
Content-Type: text/html\r\n
Content-Length: 204\r\n
Connection: keep-alive\r\n
WWW-Authenticate: Basic realm="Protected Area"\r\n
```

Additionally, we can see that even when the incorrect credentials are provided, a “GET” request is sent to the server with the invalid credentials. This provides strong evidence that it is the server, and not the browser, that is checking the password.

```
GET /basicauth/ HTTP/1.1\r\n
Host: cs231.jeffondich.com\r\n
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0\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 cG90YXRvOmhlbHA=\r\n
   Credentials: potato:help
```

It is also worth noting that when accessing http://cs231.jeffondich.com/basicauth/concrete.txt or other files located within http://cs231.jeffondich.com/basicauth/, there is no additional authentication required. In our wireshark frames, we find no additional “401 Unauthorized” or “GET” requests about authentication.This matches the behavior detailed in section 2.2 in RFC 7617. Namely, so long as we are authenticated for http://cs231.jeffondich.com/basicauth/, any further extension of that URI will not require further authentication. This possibly mundane fact is of great convenience: we usually don’t have to re-enter our credentials every time we go to a new page on a website. However, this sacrifice of security for convenience can be a problem since it does not ask for authentication every time a new subpage is accessed. Another problem is that, each time we clicked on one of the links, the credentials were sent again though we did not need to type anything. This may be even more unsafe since it provides others with more chance of stealing your credentials. The fact that the credentials are being sent without having to enter them again suggests the browser is storing the credentials for some period of time, which may be a security issue as well.

Per www.infosecmatter.com/capture-passwords-using-wireshark/#plain_text_network_protocols, it certainly appears to be the case that, given access to the packets moving on a given network, we could easily get login credentials for sites only using basic authentication.