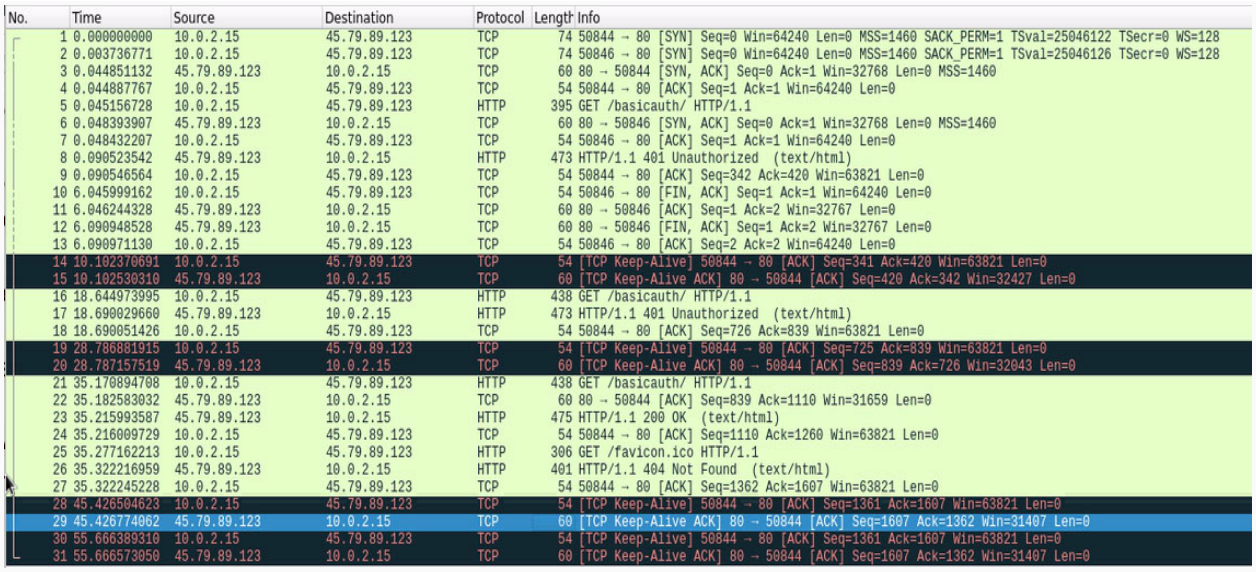
HTTP's Basic Authentication: A Story

Rebecca Fox and Yemi Shin

One day, two adventurous young women were determined to dig up the secrets of the network, and decided to go “web-diving.” The following is the log they wrote up in an effort to document their experience so that the acquired knowledge may persevere through posterity.

Mission:
Try to log in to the mystery website, and observe what happens!

1. **What queries are sent from the browser, and what responses does it receive?**



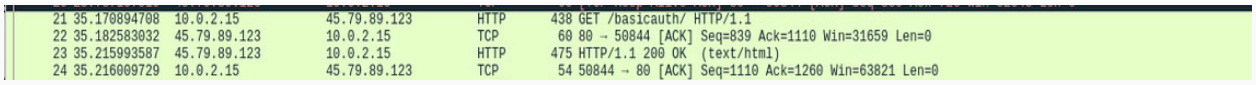
No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	10.0.2.15	45.79.89.123	TCP	74	50844 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=25046122 TSecr=0 WS=128
2	0.003736771	10.0.2.15	45.79.89.123	TCP	74	50846 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=25046126 TSecr=0 WS=128
3	0.044851132	45.79.89.123	10.0.2.15	TCP	60	80 → 50844 [SYN, ACK] Seq=0 Ack=1 Win=32768 Len=0 MSS=1460
4	0.044887767	10.0.2.15	45.79.89.123	TCP	54	50844 → 80 [ACK] Seq=1 Ack=1 Win=64240 Len=0
5	0.045156728	10.0.2.15	45.79.89.123	HTTP	395	GET /basicauth/ HTTP/1.1
6	0.048393907	45.79.89.123	10.0.2.15	TCP	60	80 → 50846 [SYN, ACK] Seq=0 Ack=1 Win=32768 Len=0 MSS=1460
7	0.048432207	10.0.2.15	45.79.89.123	TCP	54	50846 → 80 [ACK] Seq=1 Ack=1 Win=64240 Len=0
8	0.090523542	45.79.89.123	10.0.2.15	HTTP	473	HTTP/1.1 401 Unauthorized (text/html)
9	0.090546564	10.0.2.15	45.79.89.123	TCP	54	50844 → 80 [ACK] Seq=342 Ack=420 Win=63821 Len=0
10	6.045999162	10.0.2.15	45.79.89.123	TCP	54	50846 → 80 [FIN, ACK] Seq=1 Ack=1 Win=64240 Len=0
11	6.046244328	45.79.89.123	10.0.2.15	TCP	60	80 → 50846 [ACK] Seq=1 Ack=2 Win=32767 Len=0
12	6.090948528	45.79.89.123	10.0.2.15	TCP	60	80 → 50846 [FIN, ACK] Seq=1 Ack=2 Win=32767 Len=0
13	6.090971130	10.0.2.15	45.79.89.123	TCP	54	50846 → 80 [ACK] Seq=2 Ack=2 Win=64240 Len=0
14	10.102370691	10.0.2.15	45.79.89.123	TCP	54	[TCP Keep-Alive] 50844 → 80 [ACK] Seq=341 Ack=420 Win=63821 Len=0
15	10.102530310	45.79.89.123	10.0.2.15	TCP	60	[TCP Keep-Alive ACK] 80 → 50844 [ACK] Seq=420 Ack=342 Win=32427 Len=0
16	18.644973995	10.0.2.15	45.79.89.123	HTTP	438	GET /basicauth/ HTTP/1.1
17	18.690029660	45.79.89.123	10.0.2.15	HTTP	473	HTTP/1.1 401 Unauthorized (text/html)
18	18.690051426	10.0.2.15	45.79.89.123	TCP	54	50844 → 80 [ACK] Seq=726 Ack=839 Win=63821 Len=0
19	28.786881915	10.0.2.15	45.79.89.123	TCP	54	[TCP Keep-Alive] 50844 → 80 [ACK] Seq=725 Ack=839 Win=63821 Len=0
20	28.787157519	45.79.89.123	10.0.2.15	TCP	60	[TCP Keep-Alive ACK] 80 → 50844 [ACK] Seq=839 Ack=726 Win=32043 Len=0
21	35.170894708	10.0.2.15	45.79.89.123	HTTP	438	GET /basicauth/ HTTP/1.1
22	35.182583032	45.79.89.123	10.0.2.15	TCP	60	80 → 50844 [ACK] Seq=839 Ack=1110 Win=31659 Len=0
23	35.215993587	45.79.89.123	10.0.2.15	HTTP	475	HTTP/1.1 200 OK (text/html)
24	35.216009729	10.0.2.15	45.79.89.123	TCP	54	50844 → 80 [ACK] Seq=1110 Ack=1260 Win=63821 Len=0
25	35.277162213	10.0.2.15	45.79.89.123	HTTP	306	GET /favicon.ico HTTP/1.1
26	35.322216959	45.79.89.123	10.0.2.15	HTTP	401	HTTP/1.1 404 Not Found (text/html)
27	35.322245228	10.0.2.15	45.79.89.123	TCP	54	50844 → 80 [ACK] Seq=1362 Ack=1607 Win=63821 Len=0
28	45.426504623	10.0.2.15	45.79.89.123	TCP	54	[TCP Keep-Alive] 50844 → 80 [ACK] Seq=1361 Ack=1607 Win=63821 Len=0
29	45.426774062	45.79.89.123	10.0.2.15	TCP	60	[TCP Keep-Alive ACK] 80 → 50844 [ACK] Seq=1607 Ack=1362 Win=31407 Len=0
30	55.666389310	10.0.2.15	45.79.89.123	TCP	54	[TCP Keep-Alive] 50844 → 80 [ACK] Seq=1361 Ack=1607 Win=63821 Len=0
31	55.666573050	45.79.89.123	10.0.2.15	TCP	60	[TCP Keep-Alive ACK] 80 → 50844 [ACK] Seq=1607 Ack=1362 Win=31407 Len=0

A. TCP Handshake is initiated. (Lines 1-4)
B. The browser asks the server to access the web page - without any use of an ID or password. (Line 5)
C. The server responds by telling the browser that it needs authorization. (Line 8)
D. The browser/user asks for access to the page again, but puts in the incorrect ID and PW. (Line 16)
E. The server responds with a 401 Unauthorized error message, saying that the inputted ID and PW is incorrect. (Line 17)
F. The browser asks again for access and types in the correct password (Line 21)
G. The server acknowledges the correct password (Line 22)
H. The server gives access to the page (Line 23)

I. The browser asks for access to favicon.ico (line 25)
J. The server responds that favicon.ico does not exist in the server with a 404 not found error (line 26)

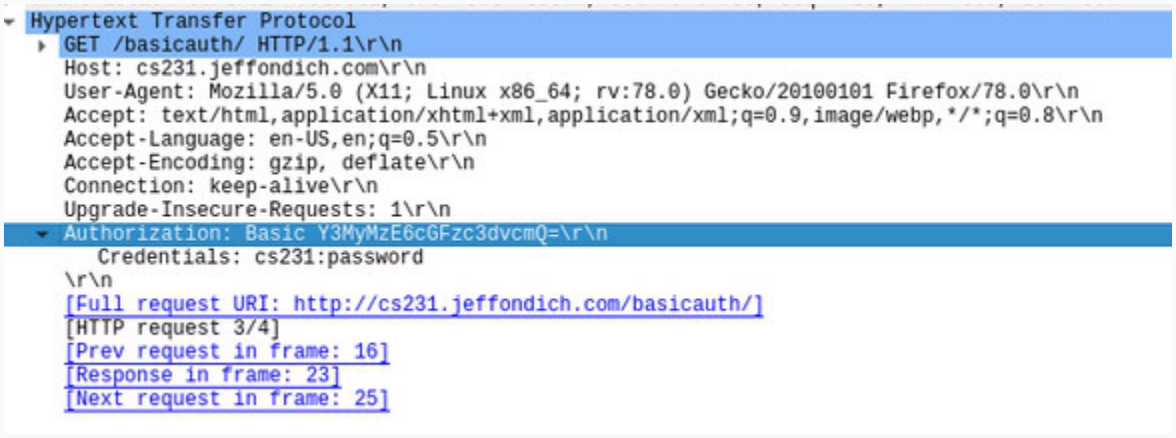
2. **Is the password sent by the browser to the server, or does the browser somehow do the password checking itself?**
   It seems that the browser (the client) does the password checking itself. According to the specification, this is behavior consistent with the “basic” HTTP authentication scheme.

(Bird’s Eye View)



We attempted to log in two times. First time we put in the wrong credentials, but the second time, we succeeded in breaking in.

(Line 21) Second Try (One that passed)

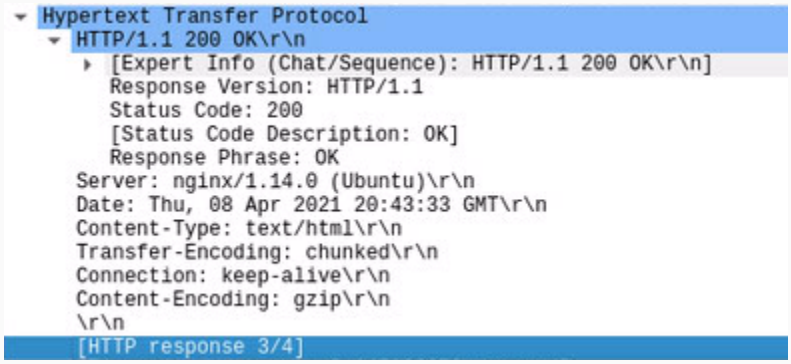


We noticed that in the GET request, there was an Authorization header, and under that a Credentials header. The format of the ID:Password was consistent with what was indicated in the specification.

(Line 5) Unidentified (This one doesn't have Authorization header)

Then we noticed, a few lines up, that we had another GET request, one that was certainly not our failed first GET request attempt. This GET request did not have an Authorization header. We were confused. (We explore this more in the last 'Additional Experimentation' question.)

(Line 23) Response from Server



This is the response we got from the server after our successful attempt. 200 OK. Just the way we like it.

3. **If the former, is the password sent in clear text or is it encrypted?**
   It seems like the password is encrypted in base 64, although we are able to view the un-encrypted versions in the credentials part of Wireshark. Our question would be, how are we able to view the bare-bones credentials??
4. **If it's encrypted, where did the encryption key come from?**
   Base 64 encryption. This is consistent with the specification's description of the client authorization scheme, which indicates that upon receipt of the credentials, the client initially encodes this user-pass into an octet sequence, which is subsequently converted to a sequence of US-ASCII characters using Base64.
5. **How does what you observe via Wireshark connect to the relevant sections of the HTTP and HTTP Basic Authentication specification documents?**

We see in the specification that the “basic” model for HTTP Authentication is “the client authenticating itself”, which is consistent with our observation of the client browser seemingly authenticating “itself” by including the credentials within its GET request.

The specification also notes that upon the initial GET /basicauth/ request, the server can respond with a “401 Unauthorized” error message, which also matches our observation of the server’s behavior (see our notes in additional experimentation).

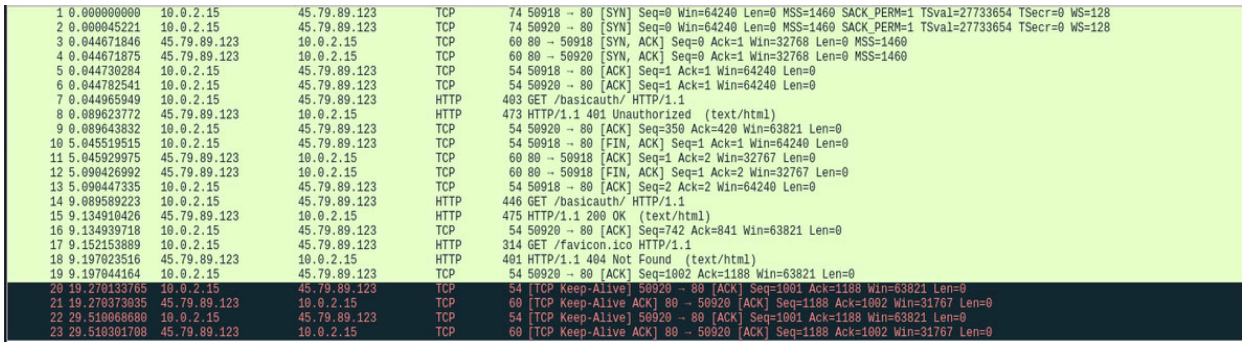
(Part of the 401 Unauthorized Response)

```
▸ Content-Length: 204\r\n
  Connection: keep-alive\r\n
  WWW-Authenticate: Basic realm="Protected Area"\r\n
  \r\n
```

6. Additional Experimentation

Because we were a little bit confused about the first GET /basicauth/ request from the browser (Line 5), we conducted another experiment to see if that was a result of typing in the wrong password or if it was part of the typical HTTP process.

(Bird’s Eye View)



1	0.000000000	10.0.2.15	45.79.89.123	TCP	74 50918 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=27733654 TSecr=0 WS=128
2	0.000045221	10.0.2.15	45.79.89.123	TCP	74 50920 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=27733654 TSecr=0 WS=128
3	0.044671846	45.79.89.123	10.0.2.15	TCP	60 80 → 50918 [SYN, ACK] Seq=0 Ack=1 Win=32768 Len=0 MSS=1460
4	0.044671875	45.79.89.123	10.0.2.15	TCP	60 80 → 50920 [SYN, ACK] Seq=0 Ack=1 Win=32768 Len=0 MSS=1460
5	0.044730284	10.0.2.15	45.79.89.123	TCP	54 50918 → 80 [ACK] Seq=1 Ack=1 Win=64240 Len=0
6	0.044782541	10.0.2.15	45.79.89.123	TCP	54 50920 → 80 [ACK] Seq=1 Ack=1 Win=64240 Len=0
7	0.044965949	10.0.2.15	45.79.89.123	HTTP	403 GET /basicauth/ HTTP/1.1
8	0.089623772	45.79.89.123	10.0.2.15	HTTP	473 HTTP/1.1 401 Unauthorized (text/html)
9	0.089643832	10.0.2.15	45.79.89.123	TCP	54 50920 → 80 [ACK] Seq=350 Ack=420 Win=63821 Len=0
10	5.045519515	10.0.2.15	45.79.89.123	TCP	54 50918 → 80 [FIN, ACK] Seq=1 Ack=1 Win=64240 Len=0
11	5.045929975	45.79.89.123	10.0.2.15	TCP	60 80 → 50918 [ACK] Seq=1 Ack=2 Win=32767 Len=0
12	5.090426992	45.79.89.123	10.0.2.15	TCP	60 80 → 50918 [FIN, ACK] Seq=1 Ack=2 Win=32767 Len=0
13	5.090447335	10.0.2.15	45.79.89.123	TCP	54 50918 → 80 [ACK] Seq=2 Ack=2 Win=64240 Len=0
14	9.089589223	10.0.2.15	45.79.89.123	HTTP	446 GET /basicauth/ HTTP/1.1
15	9.134910426	45.79.89.123	10.0.2.15	HTTP	475 HTTP/1.1 200 OK (text/html)
16	9.134939718	10.0.2.15	45.79.89.123	TCP	54 50920 → 80 [ACK] Seq=742 Ack=841 Win=63821 Len=0
17	9.152153889	10.0.2.15	45.79.89.123	HTTP	314 GET /favicon.ico HTTP/1.1
18	9.197023516	45.79.89.123	10.0.2.15	HTTP	401 HTTP/1.1 404 Not Found (text/html)
19	9.197044164	10.0.2.15	45.79.89.123	TCP	54 50920 → 80 [ACK] Seq=1002 Ack=1188 Win=63821 Len=0
20	19.270133765	10.0.2.15	45.79.89.123	TCP	54 [TCP Keep-Alive] 50920 → 80 [ACK] Seq=1001 Ack=1188 Win=63821 Len=0
21	19.270373035	45.79.89.123	10.0.2.15	TCP	60 [TCP Keep-Alive ACK] 80 → 50920 [ACK] Seq=1188 Ack=1002 Win=31767 Len=0
22	29.510068680	10.0.2.15	45.79.89.123	TCP	54 [TCP Keep-Alive] 50920 → 80 [ACK] Seq=1001 Ack=1188 Win=63821 Len=0
23	29.510301708	45.79.89.123	10.0.2.15	TCP	60 [TCP Keep-Alive ACK] 80 → 50920 [ACK] Seq=1188 Ack=1002 Win=31767 Len=0

(Line 7)

```
▾ 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: 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
    DNT: 1\r\n
    Connection: keep-alive\r\n
    Upgrade-Insecure-Requests: 1\r\n
    \r\n
    [Full request URI: http://cs231.jeffondich.com/basicauth/]
    [HTTP request 1/3]
    [Response in frame: 8]
    [Next request in frame: 14]
```

We did another capture where we were sure to type the password correctly the first time and saw that the original GET /basicauth/ request without a password was still

sent (Line 7 of the following screenshot). We interpreted this as the browser originally asking for access to the page on the server without knowing that there needs to be an id or password. Then, the 401 unauthorized error (Line 8) tells the browser that they need a password which is then sent in the second GET /basicauth/ request.