

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\nHost: cs338.jeffondich.com\r\nUser-Agent: Mozilla/5.0 (X11; Linux x86\_64; rv:109.0) Gecko/20100101 Firefox/115\r\nAccept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/w\r\nAccept-Language: en-US,en;q=0.5\r\nAccept-Encoding: gzip, deflate\r\nConnection: keep-alive\r\nUpgrade-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)

```
▼ Hypertext Transfer Protocol
  ▶ HTTP/1.1 301 Moved Permanently\r\n
    Server: nginx/1.18.0 (Ubuntu)\r\n
    Date: Tue, 19 Sep 2023 16:36:38 GMT\r\n
    Content-Type: text/html\r\n
    ▶ Content-Length: 178\r\n
    Location: http://cs338.jeffondich.com/basicauth/\r\n
    Connection: keep-alive\r\n
    \r\n
    [HTTP response 1/9]
    [Time since request: 0.052735461 seconds]
    [Request in frame: 7]
    [Next request in frame: 11]
    [Next response in frame: 13]
    [Request URI: http://cs338.jeffondich.com/basicauth]
    File Data: 178 bytes
```

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

10	0.107088825	192.168.224.128	45.79.89.123	TCP	54 50482 → 80 [ACK] Seq=354 Ack=401 Win=63840 Len=0
11	0.179965711	192.168.224.128	45.79.89.123	HTTP	408 GET /basicauth/ HTTP/1.1

- 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.

- 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.
- The specifications for this interaction are explained in section 2 of RFC 7617 (<https://datatracker.ietf.org/doc/html/rfc7617>).
- In our example, we can see that Basic is specified as the authentication scheme and the realm is set to “Protected Area.”

12	0.180775712	45.79.89.123	192.168.224.128	TCP	60 80 → 50482 [ACK] Seq=401 Ack=708 Win=64240 Len=0
13	0.235484475	45.79.89.123	192.168.224.128	HTTP	457 HTTP/1.1 401 Unauthorized (text/html)

```
▼ Hypertext Transfer Protocol
  ▶ HTTP/1.1 401 Unauthorized\r\n
    Server: nginx/1.18.0 (Ubuntu)\r\n
    Date: Tue, 19 Sep 2023 16:36:38 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
```

- 6) The next few packets are filler.
- The client acknowledges the 401 Unauthorized HTTP message.
  - 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.

14	0.235660876	192.168.224.128	45.79.89.123	TCP	54 50482 → 80 [ACK] Seq=708 Ack=804 Win=63840 Len=0
15	5.055821700	192.168.224.128	45.79.89.123	TCP	54 50470 → 80 [FIN, ACK] Seq=1 Ack=1 Win=64240 Len=0
16	5.056484301	45.79.89.123	192.168.224.128	TCP	60 80 → 50470 [ACK] Seq=1 Ack=2 Win=64239 Len=0
17	5.108354757	45.79.89.123	192.168.224.128	TCP	60 80 → 50470 [FIN, PSH, ACK] Seq=1 Ack=2 Win=64239 Len=0
18	5.108402257	192.168.224.128	45.79.89.123	TCP	54 50470 → 80 [ACK] Seq=2 Ack=2 Win=64240 Len=0
19	10.238280498	192.168.224.128	45.79.89.123	TCP	54 [TCP Keep-Alive] 50482 → 80 [ACK] Seq=707 Ack=804 Win=63840 Len=0
20	11.261722114	192.168.224.128	45.79.89.123	TCP	54 [TCP Keep-Alive] 50482 → 80 [ACK] Seq=707 Ack=804 Win=63840 Len=0
21	11.262048615	45.79.89.123	192.168.224.128	TCP	60 [TCP Keep-Alive ACK] 80 → 50482 [ACK] Seq=804 Ack=708 Win=64240 Len=0

- 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.
- This is all specified in RFC 7617.
  - We can check what Y3MzMzg6cGFzc3dvcmQ= is by converting the base64 back to the original characters and lo and behold! We get cs338:password.

22	12.828897742	192.168.224.128	45.79.89.123	HTTP	451 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/2010
  Accept: text/html,application/xhtml+xml,application/xml;q=0.9,im
  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
  \r\n
  [Full request URI: http://cs338.jeffondich.com/basicauth/]

```

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

23	12.830972344	45.79.89.123	192.168.224.128	TCP	60 80 → 50482 [ACK] Seq=804 Ack=1105 Win=64240 Len=0
24	12.883265894	45.79.89.123	192.168.224.128	HTTP	458 HTTP/1.1 200 OK (text/html)

```

Line-based text data: text/html (9 lines)
  <html>\r\n
  <head><title>Index of /basicauth/</title></head>\r\n
  <body>\r\n
  <h1>Index of /basicauth/</h1><hr><pre><a href="..">../</a>\r\n
  <a href="amateurs.txt">amateurs.txt</a>
  <a href="armed-guards.txt">armed-guards.txt</a>
  <a href="dancing.txt">dancing.txt</a>
  </pre><hr></body>\r\n
  </html>\r\n

```

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.