To begin their communication, the two computers perform a TCP handshake:

```
 1 0.000000000   192.168.161.128   172.233.221.124   TCP      74 38580 → 80 [SYN] Seq=0 Win=3
 2 0.039656853   172.233.221.124   192.168.161.128   TCP      60 80 → 38580 [SYN, ACK] Seq=0
 3 0.039800652   192.168.161.128   172.233.221.124   TCP      54 38580 → 80 [ACK] Seq=1 Ack=1
```

Then, my client sends a GET request to the server requesting the page /basicauth. In response, the server sends an ACK message to the client to indicate that it received its message:

```
 4 0.129256246   192.168.161.128   172.233.221.124   HTTP     408 GET /basicauth/ HTTP/1.1
 5 0.129759546   172.233.221.124   192.168.161.128   TCP       60 80 → 38580 [ACK] Seq=1 Ac
```

The server then lets my client know that it is unauthorized to access the page that it has requested by returning “401 Unauthorized”, and my client acknowledges:

```
 6 0.276906970   172.233.221.124   192.168.161.128   HTTP     457 HTTP/1.1 401 Unauthorized  (text/html)
 7 0.277002970   192.168.161.128   172.233.221.124   TCP       54 38580 → 80 [ACK] Seq=355 Ack=404 Win=31717 Len=0
```

Looking at the [HTTP Basic Authentication documentation](), we can see this behavior is as expected:

> Upon receipt of a request for a URI within the protection space that lacks credentials, the server can reply with a challenge using the 401 (Unauthorized) status code ([RFC7235], Section 3.1) and the WWW-Authenticate header field ([RFC7235], Section 4.1).
>
> For instance:
>
> ```
> HTTP/1.1 401 Unauthorized
> Date: Mon, 04 Feb 2014 16:50:53 GMT
> WWW-Authenticate: Basic realm="WallyWorld"
> ```

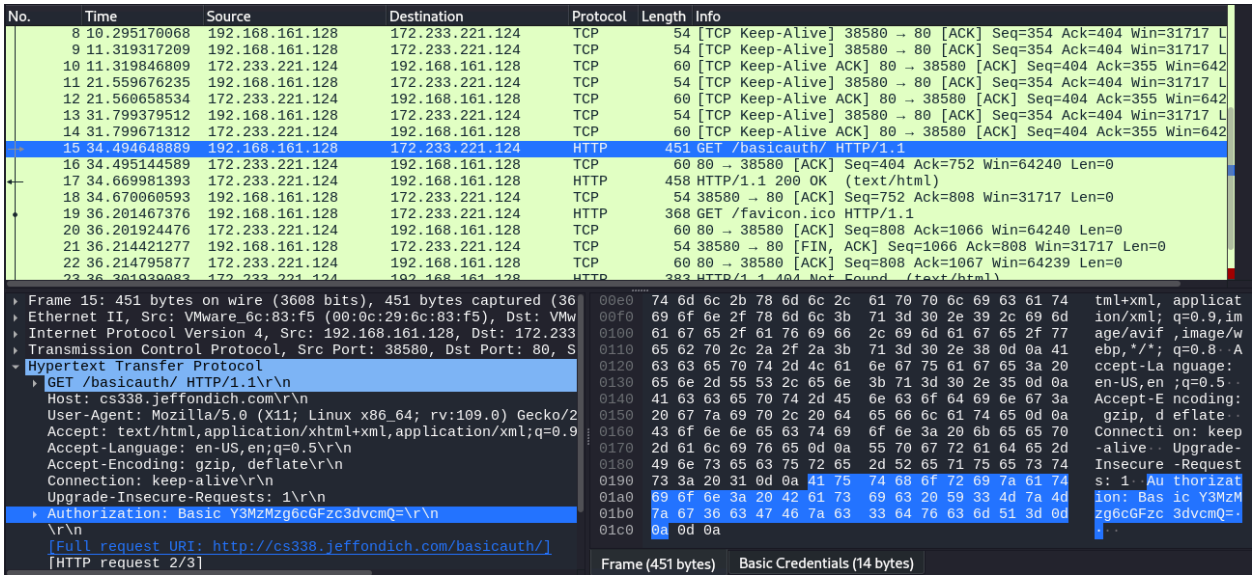
The client and server then enter a “TCP Keep-Alive” loop, where the server waits for the client to either provide valid credentials or to sever the connection:

```
 8 10.295170068  192.168.161.128   172.233.221.124   TCP      54 [TCP Keep-Alive] 38580 → 80 [ACK] Seq=354 Ack=4
 9 11.319317209  192.168.161.128   172.233.221.124   TCP      54 [TCP Keep-Alive] 38580 → 80 [ACK] Seq=354 Ack=4
10 11.319846809  172.233.221.124   192.168.161.128   TCP      60 [TCP Keep-Alive ACK] 80 → 38580 [ACK] Seq=404 A
11 21.559676235  192.168.161.128   172.233.221.124   TCP      54 [TCP Keep-Alive] 38580 → 80 [ACK] Seq=354 Ack=4
12 21.560658534  172.233.221.124   192.168.161.128   TCP      60 [TCP Keep-Alive ACK] 80 → 38580 [ACK] Seq=404 A
13 31.799379512  192.168.161.128   172.233.221.124   TCP      54 [TCP Keep-Alive] 38580 → 80 [ACK] Seq=354 Ack=4
14 31.799671312  172.233.221.124   192.168.161.128   TCP      60 [TCP Keep-Alive ACK] 80 → 38580 [ACK] Seq=404 A
```

I then submitted the credentials. What followed was another GET request sent by my client for the /basicauth page, but this time with the credentials included in the submission.

```
15 34.494648889  192.168.161.128   172.233.221.124   HTTP     451 GET /basicauth/ HTTP/1.1
16 34.495144589  172.233.221.124   192.168.161.128   TCP       60 80 → 38580 [ACK] Seq=404 Ack=752 Win=64240 Len=0
```

I learned that this submission included the credentials by looking inside the submission's data:



As we can see, an "Authorization" string was included within the GET request, and based on the fact that it ends with an equal sign it seems to be encoded in base64.

If we then run this string of characters through a base64 decoder, we find that the string is actually just "cs338:password":

```
┌──(kali㉿kali)-[~]
└─$ echo Y3MzMzg6cGFzc3dvcmQ= | base64 -d -
cs338:password
```

Looking at the [HTTP Basic Authentication documentation] once more, we can see that my hunch was correct:

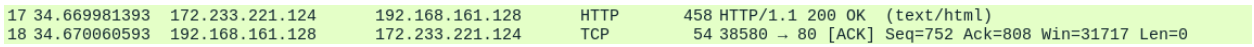
> To receive authorization, the client
>
> 1. obtains the user-id and password from the user,
>
> 2. constructs the user-pass by concatenating the user-id, a single colon (":") character, and the password,
>
> 3. encodes the user-pass into an octet sequence (see below for a discussion of character encoding schemes),
>
> 4. and obtains the basic-credentials by encoding this octet sequence using Base64 ([RFC4648], Section 4) into a sequence of US-ASCII characters ([RFC0020]).

This indicates that the credentials are just sent as plain text over the internet, as this string is leaving the computer within the GET request unencrypted. This also leads me to believe that the credential validation is done on the website's server and not on the local web browser, as sending the credentials would be redundant in that case.

The Basic Authentication documentation confirms my suspicions, saying that passwords are (extremely insecurely!) being sent over the internet as plain text:

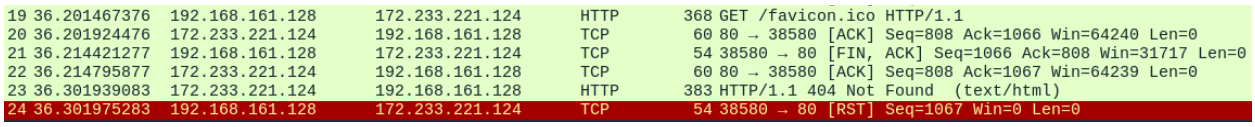
> **4. Security Considerations**
>
> The Basic authentication scheme is not a secure method of user authentication, nor does it in any way protect the entity, which is transmitted in cleartext across the physical network used as the carrier. HTTP does not prevent the addition of enhancements (such as schemes to use one-time passwords) to Basic authentication.
>
> The most serious flaw of Basic authentication is that it results in the cleartext transmission of the user's password over the physical network. Many other authentication schemes address this problem.

The server then sends the requested page and a "200 OK" message, telling my client that it has accepted the credentials and everything is all good:



17	34.669981393	172.233.221.124	192.168.161.128	HTTP	458	HTTP/1.1 200 OK (text/html)
18	34.670060593	192.168.161.128	172.233.221.124	TCP	54	38580 → 80 [ACK] Seq=752 Ack=808 Win=31717 Len=0

Finally, the browser requests favicon from the website (which gets the "404 Not Found" response) and wraps up the connection with a FIN:



19	36.201467376	192.168.161.128	172.233.221.124	HTTP	368	GET /favicon.ico HTTP/1.1
20	36.201924476	172.233.221.124	192.168.161.128	TCP	60	80 → 38580 [ACK] Seq=808 Ack=1066 Win=64240 Len=0
21	36.214421277	192.168.161.128	172.233.221.124	TCP	54	38580 → 80 [FIN, ACK] Seq=1066 Ack=808 Win=31717 Len=0
22	36.214795877	172.233.221.124	192.168.161.128	TCP	60	80 → 38580 [ACK] Seq=808 Ack=1067 Win=64239 Len=0
23	36.301939083	172.233.221.124	192.168.161.128	HTTP	383	HTTP/1.1 404 Not Found (text/html)
24	36.301975283	192.168.161.128	172.233.221.124	TCP	54	38580 → 80 [RST] Seq=1067 Win=0 Len=0