Andres Bracho Valles

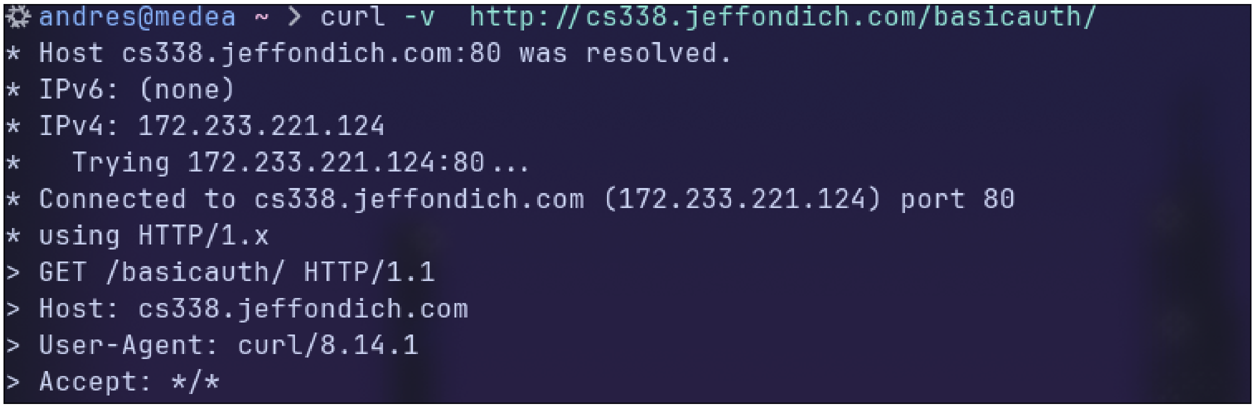
Jeff Ondich

Computer Security

25 September 2025

HTTP Basic Authentication - A story

The first step on this investigation regarding HTTP authorization was clear from the start, to figure out IPv4 address connected to the hostname http://cs338.jeffondich.com/basicauth/ The IPv4 address ended up being 172.233.221.124, as displayed at the header when running the command *curl -v http://cs338.jeffondich.com/basicauth/* As seen below:

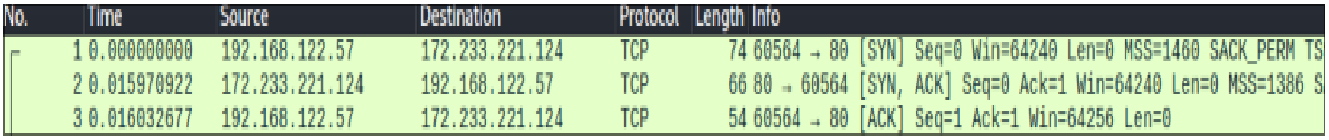


```
andres@medea ~ > curl -v  http://cs338.jeffondich.com/basicauth/
* Host cs338.jeffondich.com:80 was resolved.
* IPv6: (none)
* IPv4: 172.233.221.124
*   Trying 172.233.221.124:80...
* Connected to cs338.jeffondich.com (172.233.221.124) port 80
* using HTTP/1.x
> GET /basicauth/ HTTP/1.1
> Host: cs338.jeffondich.com
> User-Agent: curl/8.14.1
> Accept: */*
```

Knowing the host's IPv4 address the most obvious next step would be to tell Wireshark to capture all traffic going to and coming from the IPv4 address 172.233.221.124. Then the next was to set up burp to intercept requests.

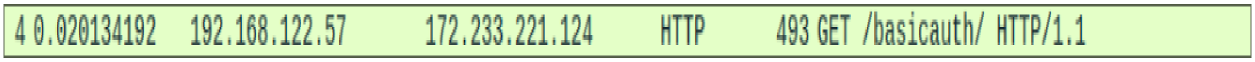
To put this whole setup process in simpler terms, imagine we're a super secret spy in the 60s trying to figure out if a politician, let's call him Mario, is entangled in a corruption scandal or not, and in order to do this we must get audio recordings on tape of every conversation Mario has a with a person of interest of codename Pumpkin. Now, even though we're a super secret spy, we don't have limitless resources, so we can't just plant a bug in his house and listen to absolutely every conversation Mario has (we'd run out of tape!!). What we need to do instead is figure out what this Pumpkin person looks like, and with watchful surveillance we'd know when they were about to enter Mario's house and when we'd need to start recording our audio. In overly simplistic terms, wireshark is a tool that lets us hire a street urchin (let's name him wireshark for simplicity) who will alert us when a person with the physical appearance of Pumpkin walks into Mario's house, and he will also tell us how greeted each other at the other before said door closed and will hold up the mic close to the windows so we can hear what goes on inside. Now, us super secret agents are listening to a live transmission of that tape recorder audio, and are armed with a high-end technological gadget called burp, that lets us slow down the audio that we're getting from inside as well as to pause it to truly digest the information (in real time!! Talk about a time bending artifact!). I'm sure that explanation wasn't convoluted at all.

Back to business though. The next step was to enter the domain name http://cs338.jeffondich.com/basicauth/ into the burp browser, upon hitting enter client went ahead and initiated a TCP handshake with the server to start a connection, we can see the TCP handshake below:



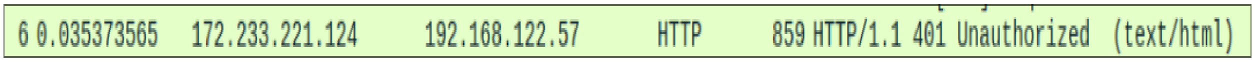
No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	192.168.122.57	172.233.221.124	TCP	74	60564 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TS
2	0.015970922	172.233.221.124	192.168.122.57	TCP	66	80 → 60564 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1386 S
3	0.016032677	192.168.122.57	172.233.221.124	TCP	54	60564 → 80 [ACK] Seq=1 Ack=1 Win=64256 Len=0

The first step is the connection request, the second is the acknowledgement from the server, and the third is the acknowledgement from the client. At this point the Handshake is complete. After the handshake the client makes the usual http request:



```
4 0.020134192   192.168.122.57      172.233.221.124     HTTP     493 GET /basicauth/ HTTP/1.1
```

However the reply from the server is not what we'd expect, since instead of getting the page's files from the server, it sends us this:

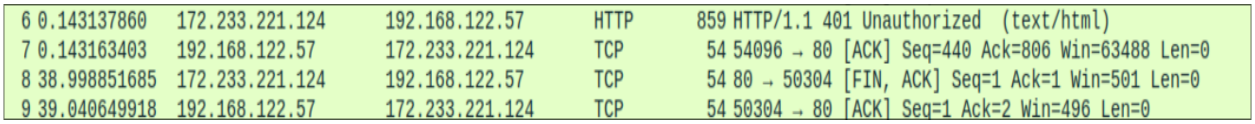


```
6 0.035373565   172.233.221.124     192.168.122.57      HTTP     859 HTTP/1.1 401 Unauthorized  (text/html)
```

What happened? Well to find out what actually happened we can look up the HTTP status code 401 that was handed to us alongside "Unauthorized" (HTTP codes page: https://en.wikipedia.org/wiki/List_of_HTTP_status_codes). As is made pretty obvious from the "Unauthorized" message, the 401 status code means that we do not have authorization to access the page so we will have to authenticate (through a username and password) that we're a user who is authorized to access the page.

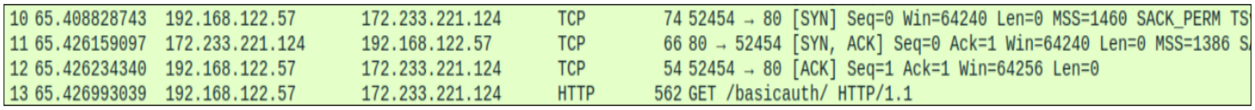
Let's go back to our super secret spy example for the folks who might need some clarification. Remember our street urchin "wireshark"? The kid we paid to keep an eye out for our person of interest Pumpkin and record their conversations with the politician Mario. Turns out that Mario got suspicious and he hired a guy to keep guard right outside his house so that no one could listen in on any of his conversations. However, Mario does hire some staff to keep his residence in order, and they are authorized to come into residence and it turns out that the only instructions the guard has from Mario is that each staff member should identify themselves by name and give a secret password to be allowed in. Luckily for us as super secret agents, each staff member is very much within earshot of wireshark when they give the guard their information to authenticate themselves as real staff members with the authorization of entering the premises. Next time our street urchin goes up to Mario's residence, he simply gives the guard a name and password that he's previously heard, and is allowed through to keep spying for us.

Now this process is very clearly outlined by the packages we can see in wireshark (we're talking about the real wireshark now, not the street urchin). After the client is given the "Unauthorized" status code, that connection is closed by the server:



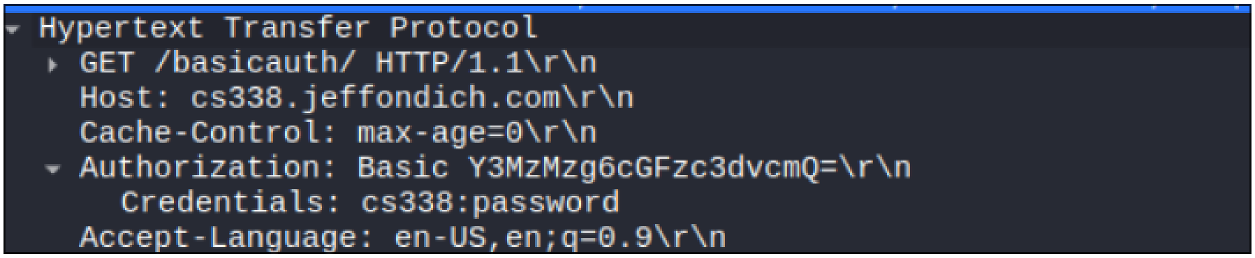
```
 6 0.143137860   172.233.221.124    192.168.122.57     HTTP     859 HTTP/1.1 401 Unauthorized  (text/html)
 7 0.143163403   192.168.122.57     172.233.221.124    TCP       54 54096 → 80 [ACK] Seq=440 Ack=806 Win=63488 Len=0
 8 38.998851685  172.233.221.124    192.168.122.57     TCP       54 80 → 50304 [FIN, ACK] Seq=1 Ack=1 Win=501 Len=0
 9 39.040649918  192.168.122.57     172.233.221.124    TCP       54 50304 → 80 [ACK] Seq=1 Ack=2 Win=496 Len=0
```

After which we're prompted for a username and password by the browser, which when input correctly prompts the client to send another TCP request as well as another GET http request:



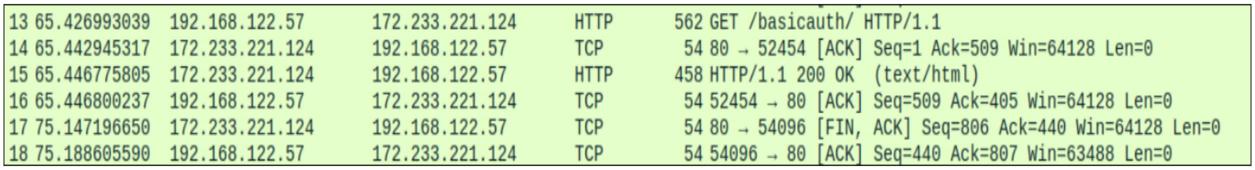
```
10 65.408828743  192.168.122.57     172.233.221.124    TCP       74 52454 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TS
11 65.426159097  172.233.221.124    192.168.122.57     TCP       66 80 → 52454 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1386 S
12 65.426234340  192.168.122.57     172.233.221.124    TCP       54 52454 → 80 [ACK] Seq=1 Ack=1 Win=64256 Len=0
13 65.426993039  192.168.122.57     172.233.221.124    HTTP     562 GET /basicauth/ HTTP/1.1
```

Now, remember when in the (not convoluted at all) explanation about user authentication with the super secret spy and corrupt politician? Well, it wasn't an exaggeration that the password and staff name was said out in the open and easy to listen, let's take a look at that get request sent by the client in Wireshark:



```
▾ Hypertext Transfer Protocol
  ▸ GET /basicauth/ HTTP/1.1\r\n
    Host: cs338.jeffondich.com\r\n
    Cache-Control: max-age=0\r\n
  ▾ Authorization: Basic Y3MzMzg6cGFzc3dvcmQ=\r\n
      Credentials: cs338:password
    Accept-Language: en-US,en;q=0.9\r\n
```

As you can see, the credentials are out in the open and in the plaintext for the world to see (Or hear, if you're a street urchin working for a super secret spy). After being authenticated as an authorized user, the server acknowledges the request, and sends us the requested data on another http package:



```
13 65.426993039  192.168.122.57     172.233.221.124    HTTP    562 GET /basicauth/ HTTP/1.1
14 65.442945317  172.233.221.124    192.168.122.57     TCP      54 80 → 52454 [ACK] Seq=1 Ack=509 Win=64128 Len=0
15 65.446775805  172.233.221.124    192.168.122.57     HTTP    458 HTTP/1.1 200 OK  (text/html)
16 65.446800237  192.168.122.57     172.233.221.124    TCP      54 52454 → 80 [ACK] Seq=509 Ack=405 Win=64128 Len=0
17 75.147196650  172.233.221.124    192.168.122.57     TCP      54 80 → 54096 [FIN, ACK] Seq=806 Ack=440 Win=64128 Len=0
18 75.188605590  192.168.122.57     172.233.221.124    TCP      54 54096 → 80 [ACK] Seq=440 Ack=807 Win=63488 Len=0
```

And so we're able to learn all the secrets that the street urchin wireshark captures for us on tape, and find the ultimate truth behind Mario's machinations… He runs a terribly protected website and should probably use https instead. The end.