Ben Hafner
Jeff Ondich
Computer Security
4/8/22

I opened a Firefox browser in my VMWare virtual machine, started Wireshark recording, went to jeffondich.com/basicauth, typed in the password and username, and then stopped Wireshark recording. Here's what I deduced from looking at the many packets Wireshark saw during that short interaction.

First, Firefox opened two TCP connections on two different randomized ports (38744 and 38746) to jeffondich.com using the standard 3-step TCP handshake. Then Firefox asked to see the page jeffondich.com/basicauth/ by sending this HTTP request over the TCP connection:

```
GET /basicauth/ HTTP/1.1
```

(plus some other header fields too of course). The server acknowledged the packet with a TCP [ACK] packet and then sent an HTTP packet that said

```
HTTP/1.1 401 Unauthorized
```

and one of the header fields that followed was

```
WWW-Authenticate: Basic realm="Protected Area"
```

which is the server's was of saying "you can't go here! To see this page, you need to be authorized for the 'Protected Area' realm." Firefox was stymied, so it turned to me for help. It gave me a pop-up box to enter a username and password. Once I typed in the magic words "cs338" and "password," Firefox turned back to the server and asked to see the page again, still using

```
GET /basicauth/ HTTP/1.1
```

But this time adding on an extra header field:

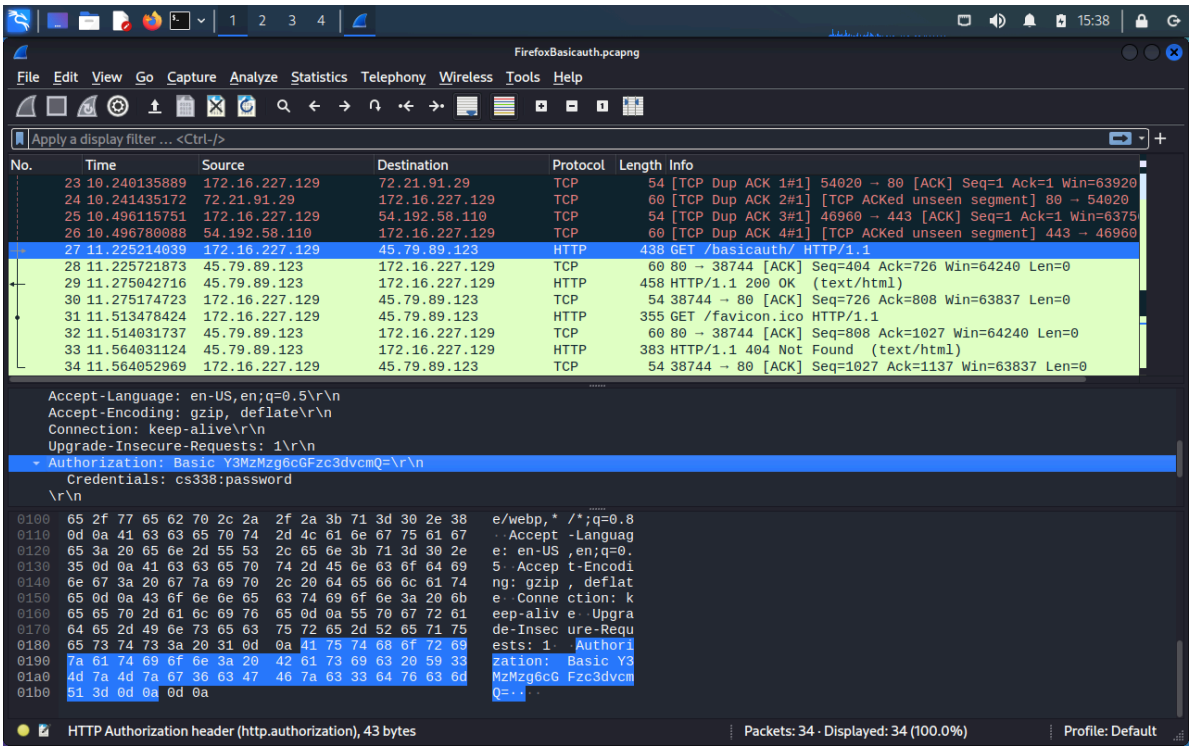
```
Authorization: Basic Y3MzMzg6cGFzc3dvcmQ=
```

Where did this come from? It's the base64 encoding of the string "cs338:password." Just to double check, I plugged "Y3MzMzg6cGFzc3dvcmQ=" into an online base64 decoder and it really does just mean "cs338:password."

Now, since Firefox sent that authorization header across the internet completely unencrypted (see screenshot #1), anyone listening to my network traffic could figure out my username and password, as long as they knew how to decode base64. This could be kind of a big security problem. Obviously, once someone else knows my password and username, they could log in to jeffondich.com/basicauth/ and see the page, so that page is not really very protected. But potentially even worse, if I happen to use the same password for both jeffondich.com/basicauth/ and for my bank account, that person could now log into my bank account. That would not be good. The Basic HTTP Authentication RFC discusses this problem in detail https://datatracker.ietf.org/doc/html/rfc7617.

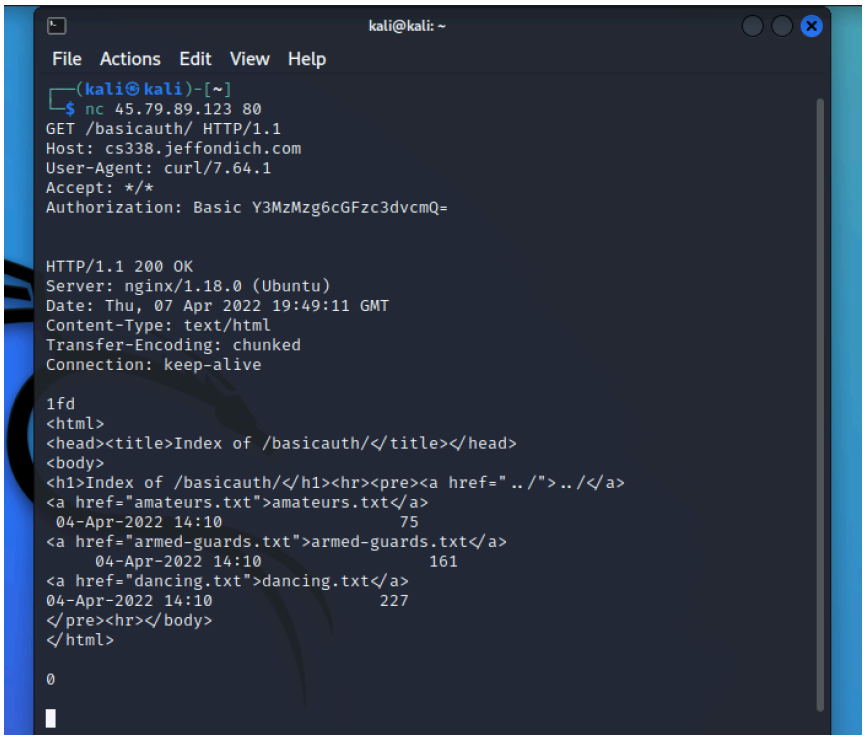
Anyway, adding that authorization header with the correct password and username satisfied the server and the server sent back a "200 OK" followed by the actual data of the webpage. Shortly afterword, I stopped Wireshark recording.

Screenshot #1



But I was a little annoyed by all the other random packets that I didn't understand. Why did it take so many packets to just access one website? I figured all the confusing packets were probably Firefox's fault, so I tried Netcat instead. This screenshot shows how I did it:

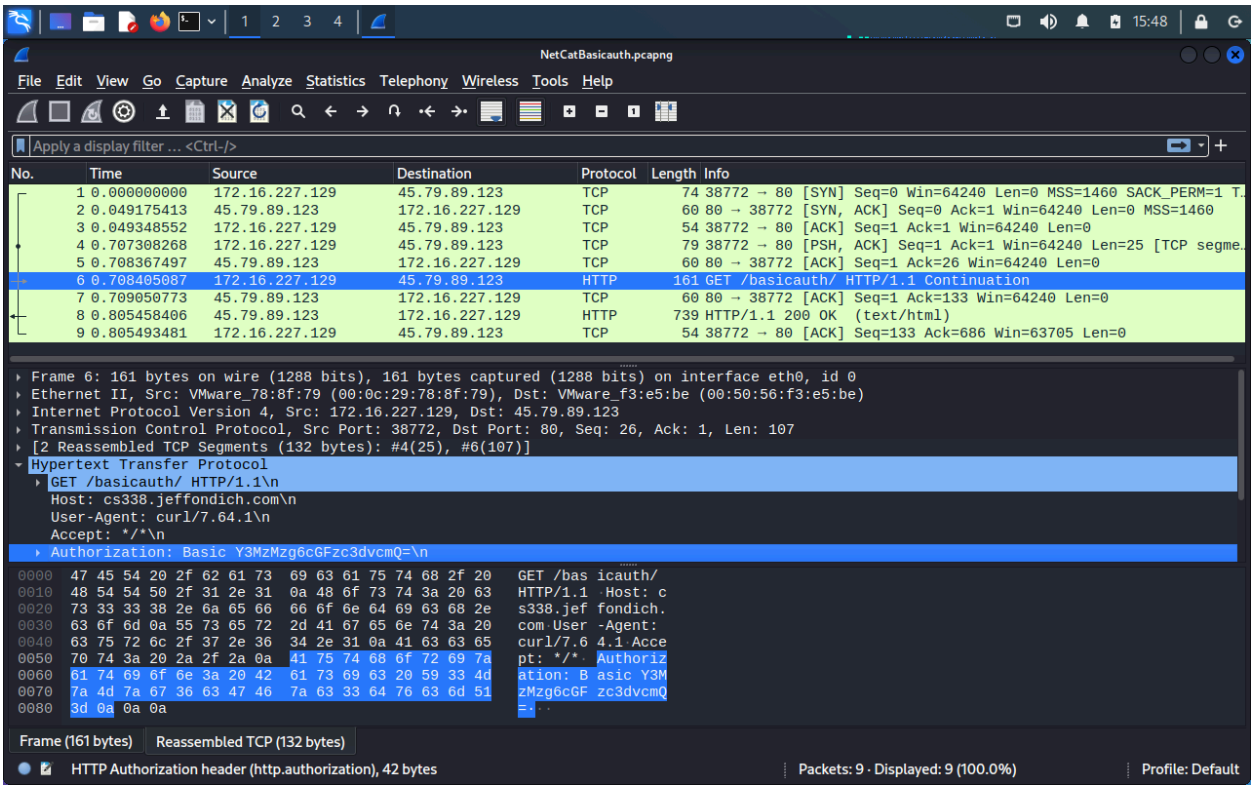
Screenshot #2



I just opened a connection with jeffondich.com and then manually typed in the GET command and the header fields. I wasn't sure what to put for the user agent field, so I pretended I was curl, even though I wasn't using curl. I used the same authentication field that Firefox did, which is just the base64 encoding of "cs338:password," as I mentioned before. This worked, and Netcat showed my the hypertext of the page I asked for.

I recorded this whole interaction on Wireshark and it was MUCH simpler than the Firefox one.

Screenshot #3



As you can see, there were only 9 packets. The first three are the TCP handshake. The next two are just empty TCP packets. I'm not sure exactly why they are necessary but I think they might have to do with the buffer. Then packet 6 is the HTTP GET request with the proper authentication field, 7 is the server's acknowledgment, 8 is the server actually showing me jeffondich.com/basicauth/, and 9 is Netcat acknowledging that it received the packet.