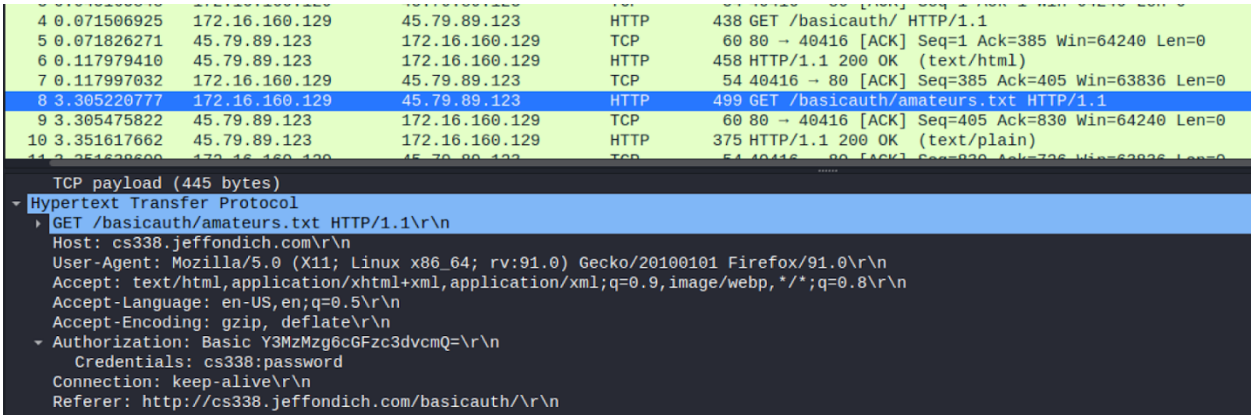
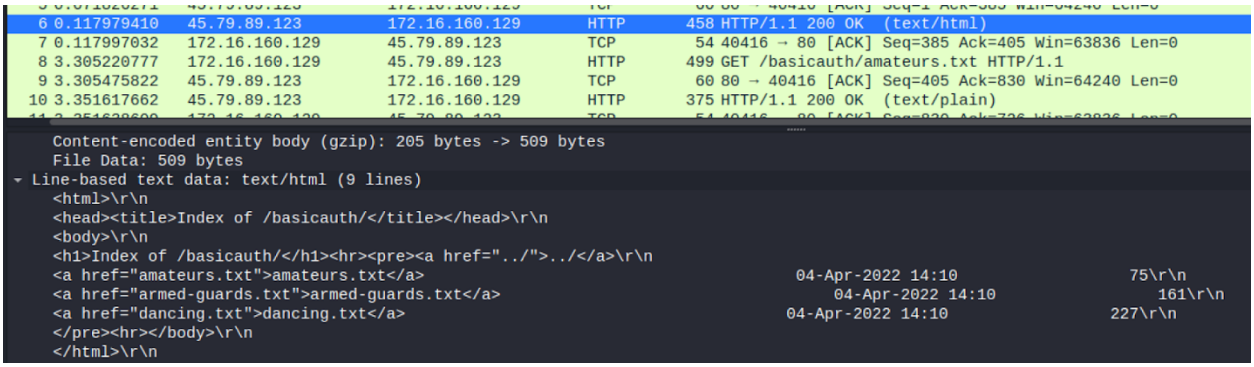
Lily Haas
Collaborated with Emma Roskopf and Ashok Khare (doing our own writeups but talking through the process together) (also we only wanted to run Kali on one machine)

Getting started:
We want to see the interactions between our browser and a password protected set of webpages by sniffing the packets with Wireshark. To make sure we were not catching irrelevant traffic, we used the filter “host 45.79.89.123”. First, we started capturing the packets and observed the TCP handshakes establishing whether the connection was using http or https.

Entering the Credentials:
Once we have established the connection, our browser sends a GET request for the webpage. This request includes the credentials we have provided to the site. These credentials are located under the HTTP header “Authorization”. It shows “Authorization: Basic” and a string that, based off my poking around in the Mozilla documentation, is a base64 encoding of the string “cs338:password” that appears below with “Credentials”. The GET request for any of the webpages under this one (amateurs.txt, armed-guards.txt, and dancing.txt) also contains this information. It looks like the browser encoded the credentials to send but also sent a plain text version of the username and password.



Gaining Access to the Password Protected Pages:
Every time we enter a password, we receive first an acknowledgement then a response either giving us access to the page or telling us the credentials were incorrect. When we have provided the correct username and password to the site, the server sends back a 200 OK HTTP response. This contains the content of the page we are trying to access. In addition to being able to access the credentials for the site, we are also able to find the HTML and text being sent back from the website to our browser. This is also unencrypted, and Wireshark puts it under the Line-based text data dropdown.



Trying to Fake Security:

Just for fun, we decided to see what would happen if we input a bad password or username. Seems this site is a little more secure than we were giving it credit for at this point, because it did not give us access with the wrong credentials. We wondered if a user who knew the site was not secure could use this to hide their password in a series of fake password attempts. No, no they could not. Entering the wrong password causes the website to send back a 401 Authorization Required HTTP response. Entering the correct password (or the browser using the cached correct password) gets a 200 response (pictured above with the HTML payload). Wireshark even does a nice way of showing which previous frames are being referenced, when we selected the 200 OK frame a little checkmark appeared next to the frame that sent the proper credentials.

A story of how a site like this can be exploited:

So, this site seems like it isn't very secure. What can you do with this? Well, depending on what the site stores, a lot. For sake of argument, let's say this site belongs to someone just tech savvy enough to know how to make and host their own password protected site to use as a sort of online storage, but not far enough in their computer science education to have learned about all the dangers of not setting up better encryption or how to do so. Let's call him Bob. Say Bob's computer breaks but he still needs to access some data from the site (maybe he uploaded a paper there that he needs to turn in tomorrow). He goes to the local public library to use a public computer configured by well-meaning but somewhat tech illiterate volunteers. Unfortunately for Bob, you know these computers do not reset and wipe their data after each use and just have a single profile accessible with username and password "locallibrary".

You have decided you want to cause problems on purpose and have set up Wireshark on this computer. Bob logs into his site with his credentials and retrieves and prints his paper. Say he decides to be a bit careful and decides to clear the browser's history and cache when he's done. At the end of the day, you access the computer and put all the information from Wireshark onto a flashdrive and take it home to inspect. You find the credentials to Bob's site and see the content of his paper. You start to think maybe this was a waste of time, but you start up your VPN (wouldn't want Bob tracking you down if he figured out how) and log in with his username and password anyways.

It's better than you could have imagined. Bob doesn't just use this site as storage for his schoolwork, he has a document called "passwords.txt" containing every password he uses. You almost feel sorry for Bob so instead of stealing his money or online shopping on his dime (you'd

have to use your address for that anyways) so you decide to just mess with him a bit. Using the email login he so graciously left at the top of the document, you change all his passwords. Honestly, he's lucky you're the one who figured this out, most hackers would have stolen his identity. You simply decided to teach him a lesson about securing your websites.

But what if this weren't just a story:
Clearly this kind of password protection on a site is not good enough. Imagine if instead of this being some personal website belonging to some computer science student, this was a website that belonged to a bank. Any number of people who do not own a computer could walk into a public library to check their bank balance online. By capturing the packets associated with them logging in, a thief could gain access to multiple bank accounts and steal large amounts of money with only a small amount of effort. If it was a file storage site (similar to Google Drive) you could gain access to someone's password file if it was not encrypted. Or a site that handles medical records, storing medical data unencrypted has to be a HIPAA violation. Clearly the password protection on this site is not good enough to be used for most real websites.