HTTP Basic Authentication

Goal: Investigate what happens when I log into the website http://cs338.jeffondich.com/basicauth/ and try to access the secrets folder.

Step 1: Log into Kali and setup my environment. I will be using Wireshark to intercept the packets between my client (Firefox) and the server.

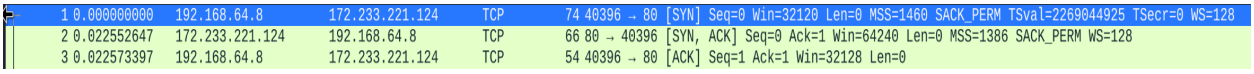
Step 2: In Wireshark, I am running the filter host 172.233.221.124 (which can be found using the command host cs338.jeffondich.com). This will only display web traffic between Kali and the website I care about.

Step 3: Log into Jeff's secrets folder. I provided the username cs338 and the password "password".

Step 4: Poke around and access some of the files to see what happens when I click on these links.

Here is what I observed in Wireshark:

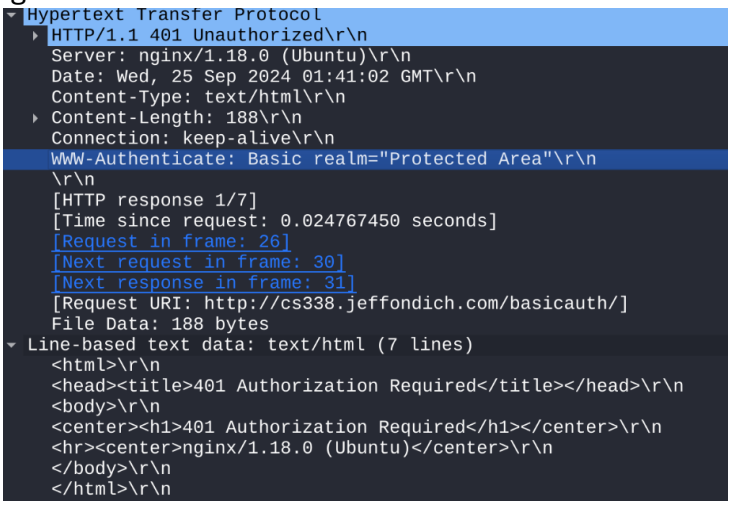
1. The first thing that happens is a TCP three-way handshake, just like any other TCP connection. The client initiates the communication on port 40396 in the case shown below. A SYN packet is sent to the server, the server sends SYN ACK, and the client sends an ACK.



2. Then, the client asks for access to the basicauth folder with the HTTP request GET /basicauth/ HTTP/1.1. At first, only the normal HTTP headers are included in the GET request (things like User-Agent, Host, Encoding, Connection). keep-alive is set to be true, and there are periodic TCP packets acknowledging that the connection should still be kept running (I am assuming to avoid a TCP timeout).

3. The server responds with a "401 Unauthorized" response, and includes the header WWW-Authenticate: Basic realm="Protected Area". Basically, the server realized that the user was trying to access a folder that is protected, and the client did not

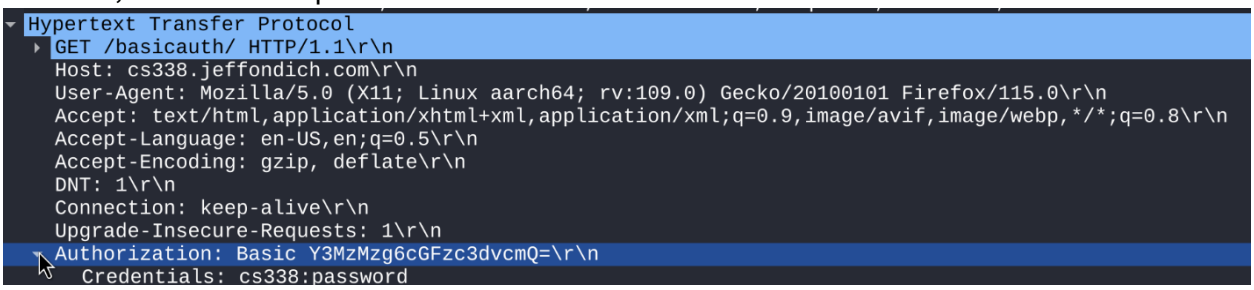
provide any authentication, so the server told the client “no, give me authentication.” As you can see below, the server was even nice enough to provide a little HTML saying that authentication is required, which by browser completely ignores.



At this point, it is useful to take a step back and look up what WWW-Authenticate actually means. According to Mozilla, this header is included with 401 responses, and it includes information on how to proceed with authentication. The first token is the method of authentication (auth-scheme). Here the scheme is Basic, which transmits user ID/password pairs encoded in base64. The scheme is codified in RFC 7617. Since the scheme just encodes the data in base64, on its own it is not secure and *should* be encrypted in the real world.

Source: https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/WWW-Authenticate.

4. Anyway, once the client sees that it is needs authentication, and armed with the authentication scheme, it prompts the user (me) for credentials. I typed them in and hit enter, and a new request was sent to the server.



The browser has now included the Authentication header. This header has two tokens (although it could have more depending on the scheme, encoding, etc.). The first token is Basic, which is just the authentication scheme being used. The second token is the credentials, encoded in base64 as you can see. Wireshark is nice

enough to decode this for you, but you can also run base64 -d and get the same output:

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

5. Finally, the server sends back a “200 OK” response, along with some HTML describing the file structure.

```
Hypertext Transfer Protocol, has 2 chunks (including last chunk)
  HTTP/1.1 200 OK\r\n
  Server: nginx/1.18.0 (Ubuntu)\r\n
  Date: Wed, 25 Sep 2024 01:41:18 GMT\r\n
  Content-Type: text/html\r\n
  Transfer-Encoding: chunked\r\n
  Connection: keep-alive\r\n
  Content-Encoding: gzip\r\n
  \r\n
  [HTTP response 3/7]
  [Time since request: 0.026294528 seconds]
  [Prev request in frame: 30]
  [Prev response in frame: 31]
  [Request in frame: 33]
  [Next request in frame: 36]
  [Next response in frame: 37]
  [Request URI: http://cs338.jeffondich.com/basicauth/]
  HTTP chunked response
  Content-encoded entity body (gzip): 205 bytes -> 509 bytes
  File Data: 509 bytes
Line-based text data: text/html (9 lines)
  <html>\r\n
  <head><title>Index of /basicauth/</title></head>\r\n
  <body>\r\n
```

6. One more important thing to note: the user sends its credentials every time it wants to access a file in the directory. If I click one of the links shown on the basicauth page, the browser sends the Authentication header with credentials that it remembered from the last time you typed it in. So, when I request the page amateurs.txt, this is the request the browser sends:

```
Hypertext Transfer Protocol
  GET /basicauth/amateurs.txt HTTP/1.1\r\n
  Host: cs338.jeffondich.com\r\n
  User-Agent: Mozilla/5.0 (X11; Linux aarch64; rv:109.0) Gecko/20100101 Firefox/115.0\r\n
  Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8\r\n
  Accept-Language: en-US,en;q=0.5\r\n
  Accept-Encoding: gzip, deflate\r\n
  Referer: http://cs338.jeffondich.com/basicauth/\r\n
  DNT: 1\r\n
  Authorization: Basic Y3MzMzg6cGFzc3dvcmQ=\r\n
    Credentials: cs338:password
  Connection: keep-alive\r\n
  Upgrade-Insecure-Requests: 1\r\n
  \r\n
```

Otherwise, the server would respond with another “401 Unauthorized” response.

Summary

The Basic authentication scheme is a way for a server to only allow access to certain files if credentials are known. If credentials are not known, the server will send a 401 response. The WWW-Authenticate header in this response tells the client why it can't access those resources and what authentication scheme should be used. In order for the client to gain access, it needs to send credentials with the Authenticate header in its GET request. In the example above, the authentication scheme was Basic, which just encodes the credentials in base64 before sending them off. Since this data is not inherently encrypted, if Basic authentication is used, it should be over HTTPS.