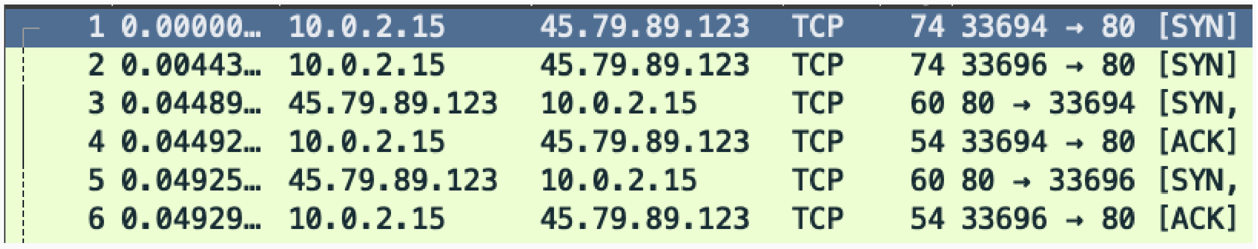
[ASSIGNMENT 4/9] HTTP's Basic Authentication: A Story

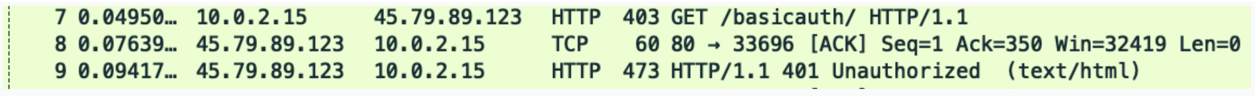
Sean Lee and Peter McCrea
4/8/2021

1. **What queries are sent from the browser, and what responses does it receive?**



```
1 0.00000…  10.0.2.15     45.79.89.123  TCP   74 33694 → 80 [SYN]
2 0.00443…  10.0.2.15     45.79.89.123  TCP   74 33696 → 80 [SYN]
3 0.04489…  45.79.89.123  10.0.2.15     TCP   60 80 → 33694 [SYN,
4 0.04492…  10.0.2.15     45.79.89.123  TCP   54 33694 → 80 [ACK]
5 0.04925…  45.79.89.123  10.0.2.15     TCP   60 80 → 33696 [SYN,
6 0.04929…  10.0.2.15     45.79.89.123  TCP   54 33696 → 80 [ACK]
```

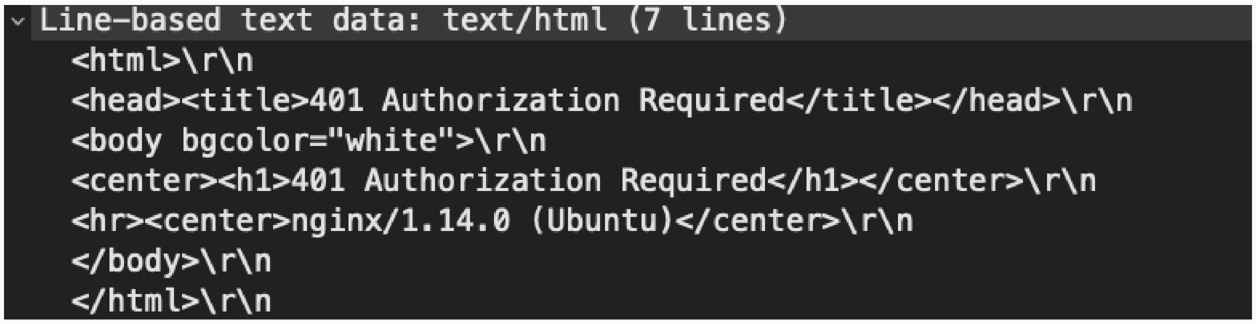
As expected, the first information sent between the two systems are TCP packets. Before anything interesting happens TCP [SYN] (synchronization) packets are sent from our browser to the server, followed by the server sending back its own TCP [SYN,ACK] (synchronization and acknowledgement) packets. Our client then proceeds to acknowledge the server's acknowledgment. Now onto the good stuff ...



```
7 0.04950…  10.0.2.15     45.79.89.123  HTTP  403 GET /basicauth/ HTTP/1.1
8 0.07639…  45.79.89.123  10.0.2.15     TCP    60 80 → 33696 [ACK] Seq=1 Ack=350 Win=32419 Len=0
9 0.09417…  45.79.89.123  10.0.2.15     HTTP  473 HTTP/1.1 401 Unauthorized  (text/html)
```

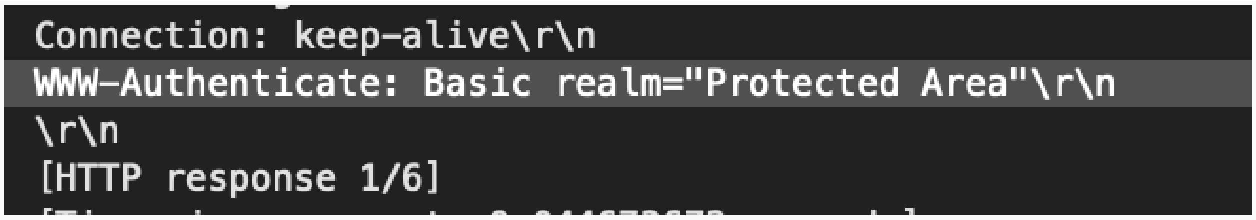
After that is done, our browser sends a HTTP GET request to the server, letting it know we want to "get" the resources located at /basicauth/ (see WireShark line 7).

On WireShark line 9, the server sends us a "401 Unauthorized" response, containing this information:



```
Line-based text data: text/html (7 lines)
  <html>\r\n
  <head><title>401 Authorization Required</title></head>\r\n
  <body bgcolor="white">\r\n
  <center><h1>401 Authorization Required</h1></center>\r\n
  <hr><center>nginx/1.14.0 (Ubuntu)</center>\r\n
  </body>\r\n
  </html>\r\n
```

Additionally, and very importantly, this response also includes this information:

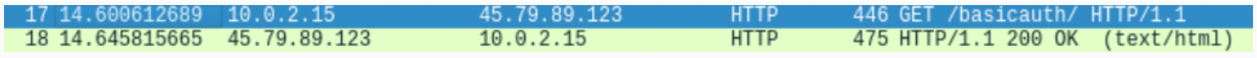


```
Connection: keep-alive\r\n
WWW-Authenticate: Basic realm="Protected Area"\r\n
\r\n
[HTTP response 1/6]
```

This “WWW-Authenticate: Basic” tells our browser that the server is using HTTP Basic authentication. The “realm=”Protected Area”” could be any string set by the server.

“The realm value (case-sensitive), in combination with the canonical root URL of the server being accessed, defines the protection space. These realms allow the protected resources on a server to be partitioned into a set of protection spaces, each with its own authentication scheme and/or authorization database. The realm value is a string, generally assigned by the origin server, which may have additional semantics specific to the authentication scheme.” - https://tools.ietf.org/html/rfc1945#section-11

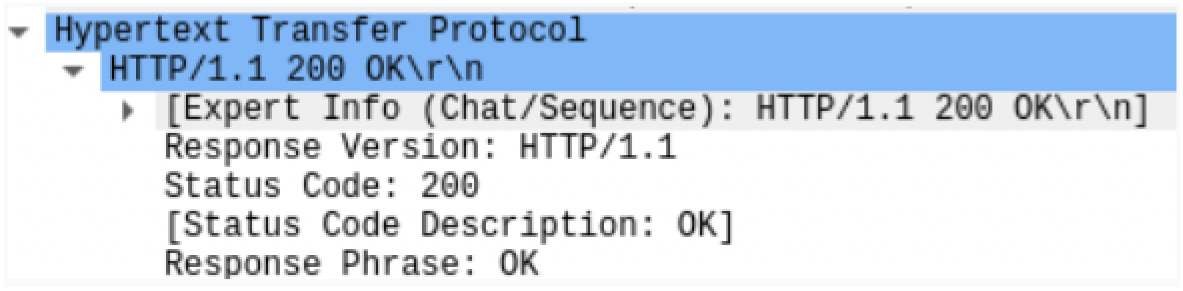
2. **After the password is typed by the user, what sequence of queries and responses do you see?**



```
17 14.600612689 10.0.2.15       45.79.89.123    HTTP    446 GET /basicauth/ HTTP/1.1
18 14.645815665 45.79.89.123    10.0.2.15       HTTP    475 HTTP/1.1 200 OK  (text/html)
```

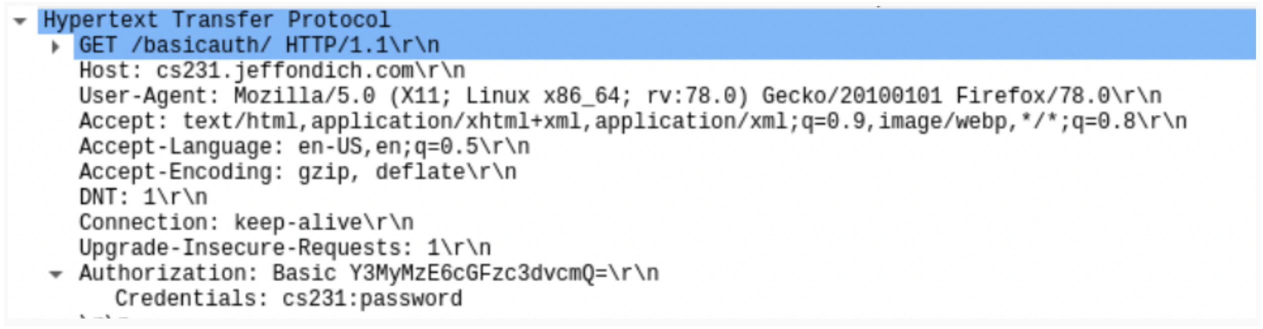
The server sends a 401 unauthorized response to our client, which prompts the username and password box to pop up. When we type in a username and password, our client sends a HTTP GET header with information including the plaintext username and password shown.

After this, the server locally checks this information and sends back a HTTP 200 response with the information shown in the next section if the correct password was entered. (403 forbidden if not)



```
Hypertext Transfer Protocol
  HTTP/1.1 200 OK\r\n
    [Expert Info (Chat/Sequence): HTTP/1.1 200 OK\r\n]
    Response Version: HTTP/1.1
    Status Code: 200
    [Status Code Description: OK]
    Response Phrase: OK
```

3. **Is the password sent by the browser to the server, or does the browser somehow do the password checking itself?**

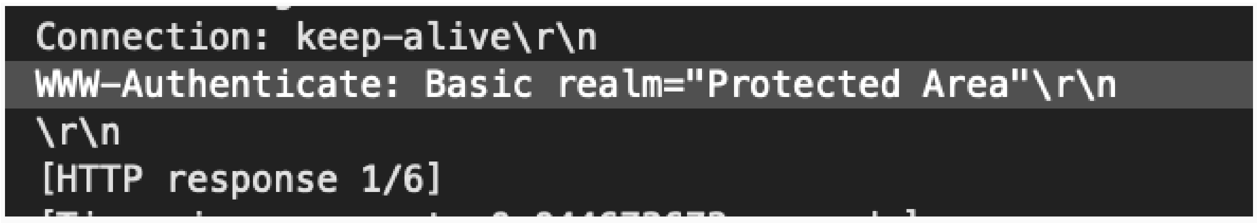


```
Hypertext Transfer Protocol
  GET /basicauth/ HTTP/1.1\r\n
  Host: cs231.jeffondich.com\r\n
  User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0\r\n
  Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8\r\n
  Accept-Language: en-US,en;q=0.5\r\n
  Accept-Encoding: gzip, deflate\r\n
  DNT: 1\r\n
  Connection: keep-alive\r\n
  Upgrade-Insecure-Requests: 1\r\n
  Authorization: Basic Y3MyMzE6cGFzc3dvcmQ=\r\n
    Credentials: cs231:password
```

The password is sent by the browser to the server with the authorization request headers containing the credentials to authenticate a user agent. The username and password is sent in cleartext base64, which anyone can easily reverse into the “regular” string. This contains the credentials typed in by the user, which the server will check to see if the info entered in was correct

4. **If it's encrypted, where did the encryption key come from?**

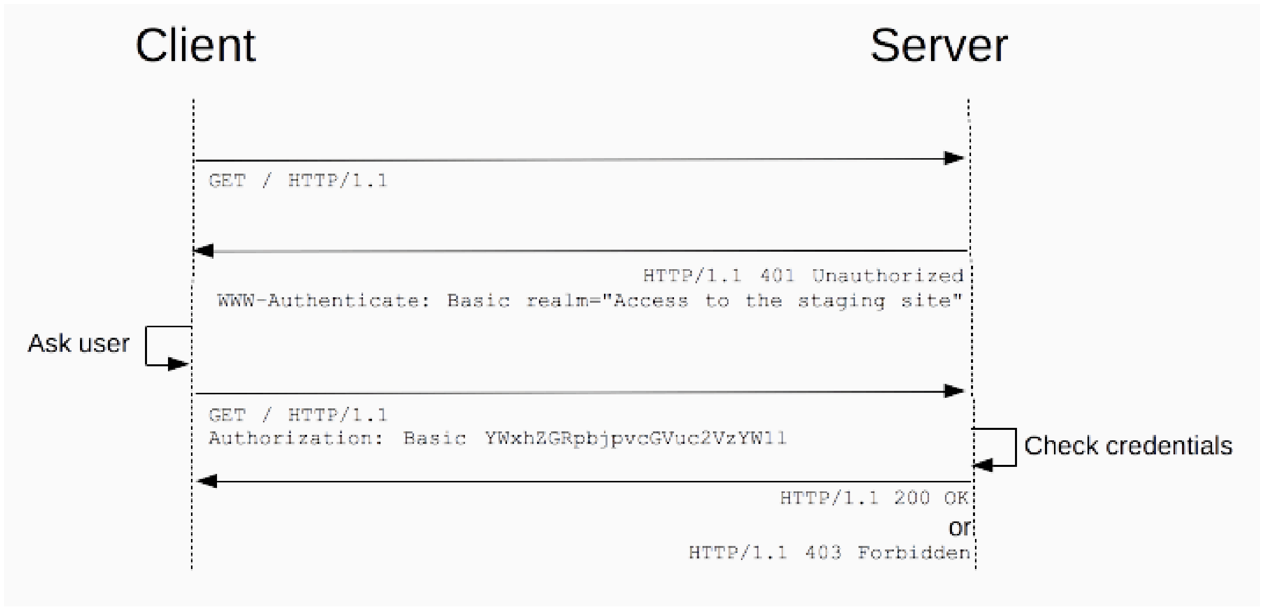
Not encrypted, we know that this will not be encrypted because very early on the server told our browser that the authentication method used was the HTTP Basic authentication, which is not secure / encrypted:



```
Connection: keep-alive\r\n
WWW-Authenticate: Basic realm="Protected Area"\r\n
\r\n
[HTTP response 1/6]
```

(picture from the original “401 unauthorized” response the server sent our browser after initially trying to access /basicauth/)

5. **How does what you observe via Wireshark connect to the relevant sections of the HTTP and HTTP Basic Authentication specification documents?**



```
GET / HTTP/1.1

HTTP/1.1 401 Unauthorized
WWW-Authenticate: Basic realm="Access to the staging site"

GET / HTTP/1.1
Authorization: Basic YWxhZGRpbjpvcGVuc2VzYW1l

HTTP/1.1 200 OK
HTTP/1.1 403 Forbidden
```

- https://developer.mozilla.org/en-US/docs/Web/HTTP/Authentication

What we observed (i.e. what the WireShark logs showed us) was exactly concept for concept with what the HTTP and HTTP Basic Authentication specification documents prescribe for proper implementation of HTTP and HTTP Basic Authentication.