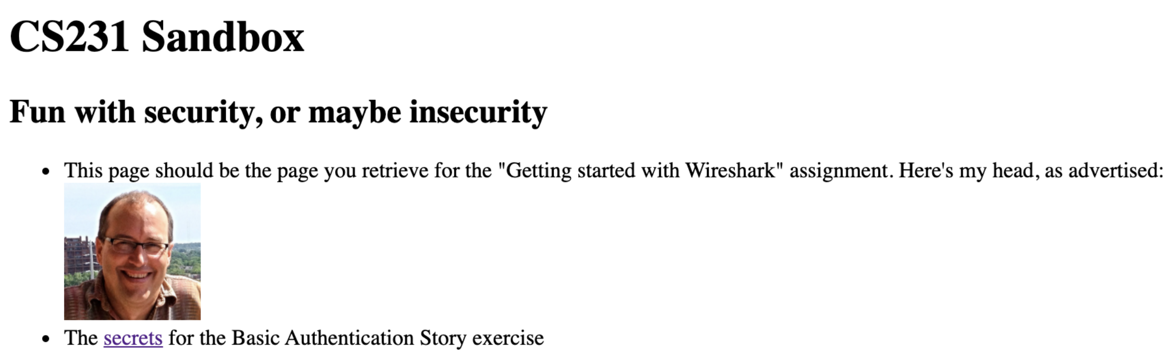
Ankit Sanghi
Ginnie White

Basic Authentication

As we navigated through the CS231 Sandbox, here are all the steps, in chronological order, that we took using Wireshark in order to force the website to divulge all its secrets. We started by going through the index page of the site, then to the secret authentication.

Step One: TCP 3 Way Handshake

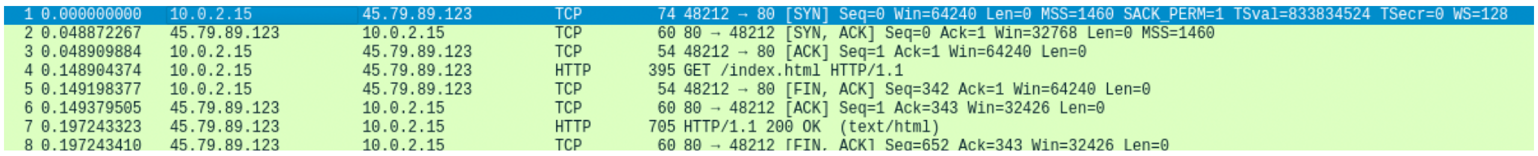


CS231 Sandbox

Fun with security, or maybe insecurity

- This page should be the page you retrieve for the "Getting started with Wireshark" assignment. Here's my head, as advertised:
- The secrets for the Basic Authentication Story exercise

1. We started on the CS231 Sandbox homepage. First, our host computer sends a SYN packet through TCP to connect to the server.
2. The server then sends a SYN packet as well as an ACK packet to tell the client that it acknowledges receiving the previous packet.
3. The client sends its own acknowledgement to the server's acknowledgement.
4. The client then sends an HTTP GET Request for the index.html page for Jeff's site. Then it sends a FIN and ACK packet to tell the server that it's done talking to the server.
5. The server sends an ACK packet to acknowledge receiving the GET request, and then sends the index.html page to the client. Then it sends a FIN and ACK packet to the client to say that it acknowledges the request and is finished talking to the client.



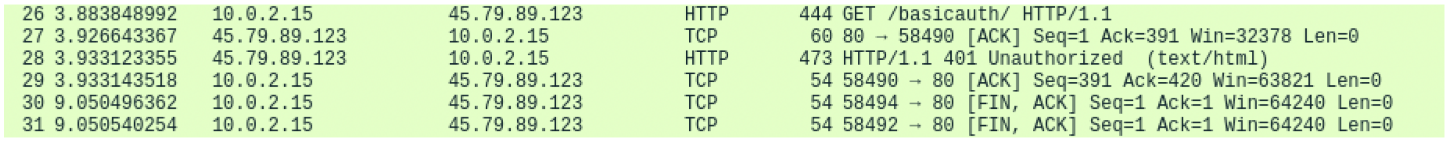
```
1 0.000000000   10.0.2.15       45.79.89.123    TCP      74 48212 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=833834524 TSecr=0 WS=128
2 0.048872267   45.79.89.123    10.0.2.15       TCP      60 80 → 48212 [SYN, ACK] Seq=0 Ack=1 Win=32768 Len=0 MSS=1460
3 0.048909884   10.0.2.15       45.79.89.123    TCP      54 48212 → 80 [ACK] Seq=1 Ack=1 Win=64240 Len=0
4 0.148904374   10.0.2.15       45.79.89.123    HTTP    395 GET /index.html HTTP/1.1
5 0.149198377   10.0.2.15       45.79.89.123    TCP      54 48212 → 80 [FIN, ACK] Seq=342 Ack=1 Win=64240 Len=0
6 0.149379505   45.79.89.123    10.0.2.15       TCP      60 80 → 48212 [ACK] Seq=1 Ack=343 Win=32426 Len=0
7 0.197243323   45.79.89.123    10.0.2.15       HTTP    705 HTTP/1.1 200 OK  (text/html)
8 0.197243410   45.79.89.123    10.0.2.15       TCP      60 80 → 48212 [FIN, ACK] Seq=652 Ack=343 Win=32426 Len=0
```

Step Two: Requesting Secrets and Authentication

1. When requesting the basicauth/ endpoint, the client and server do another handshake to re-establish the connection. First the client sends a SYN packet to the server, the server

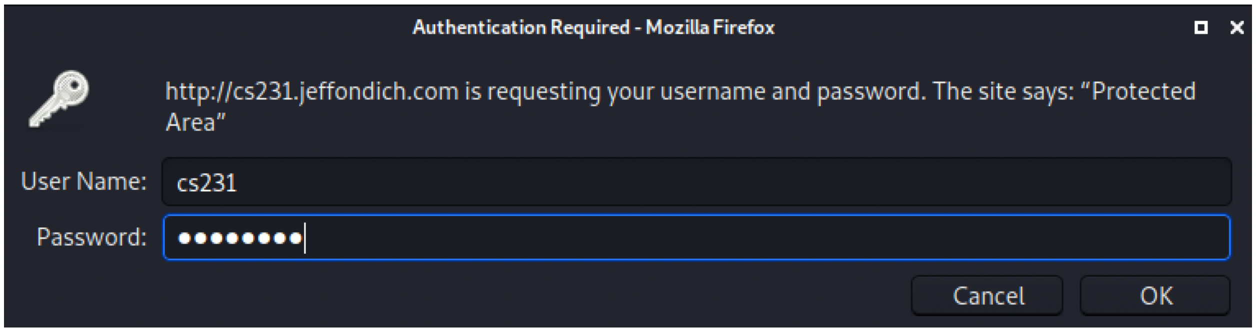
responds with a SYN and ACK packet to acknowledge receiving the client's packet, and then the client sends its own ACK packet to acknowledge receiving the server's packet.

2. Then the client sends a GET request to /basicauth/. GET is a type of request discussed in the HTTP specification documents, and it's basically just a request for information or a file. The server sends an ACK packet to acknowledge receiving the packet from the client. It then responds with a 401 Unauthorized error message.
3. The client then sends an ACK packet and a FIN packet to acknowledge getting the message and closing the connection between the client and the server.



```
26 3.883848992   10.0.2.15       45.79.89.123    HTTP     444 GET /basicauth/ HTTP/1.1
27 3.926643367   45.79.89.123    10.0.2.15       TCP       60 80 → 58490 [ACK] Seq=1 Ack=391 Win=32378 Len=0
28 3.933123355   45.79.89.123    10.0.2.15       HTTP     473 HTTP/1.1 401 Unauthorized  (text/html)
29 3.933143518   10.0.2.15       45.79.89.123    TCP       54 58490 → 80 [ACK] Seq=391 Ack=420 Win=63821 Len=0
30 9.050496362   10.0.2.15       45.79.89.123    TCP       54 58494 → 80 [FIN, ACK] Seq=1 Ack=1 Win=64240 Len=0
31 9.050540254   10.0.2.15       45.79.89.123    TCP       54 58492 → 80 [FIN, ACK] Seq=1 Ack=1 Win=64240 Len=0
```

Step Three: Getting the Password:



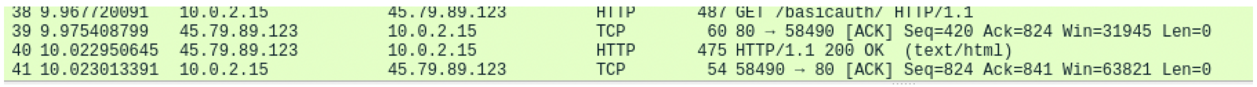
Authentication Required - Mozilla Firefox

http://cs231.jeffondich.com is requesting your username and password. The site says: "Protected Area"

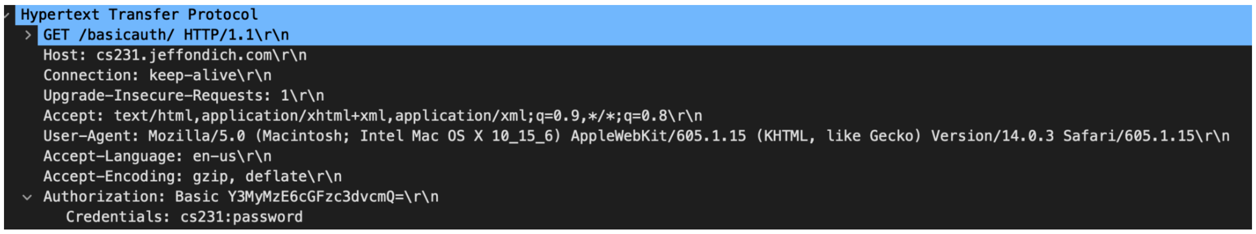
User Name: cs231

Password: ●●●●●●●●

1. When we input the username and password into the website and hit enter, we send another GET request to the server. This time, we have additional data in our headers, specifically the Authorization header. This includes a field called Credentials which contains the username and password entered by the user. This can be seen in the second screenshot below. Originally, this information was encrypted in Base64, but Wireshark converts it to plaintext for us so we can see that our credentials were cs231:password. This is not secure at all, if Wireshark can decrypt it so easily.
2. The server sends an ACK packet to acknowledge the GET request sent by the client.
3. We get an HTTP 200 OK response, which means we've been authenticated.
4. The client sends an ACK request to the server as we've acknowledged that we've been authenticated.
5. The server sends a FIN ACK request to the client, closing that connection (not in the below screenshot)

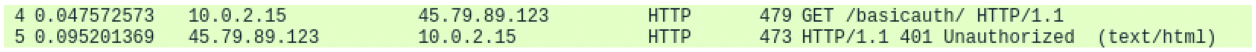


```
38 9.967720091   10.0.2.15          45.79.89.123       HTTP      487 GET /basicauth/ HTTP/1.1
39 9.975408799   45.79.89.123       10.0.2.15          TCP        60 80 → 58490 [ACK] Seq=420 Ack=824 Win=31945 Len=0
40 10.022950645  45.79.89.123       10.0.2.15          HTTP      475 HTTP/1.1 200 OK  (text/html)
41 10.023013391  10.0.2.15          45.79.89.123       TCP        54 58490 → 80 [ACK] Seq=824 Ack=841 Win=63821 Len=0
```



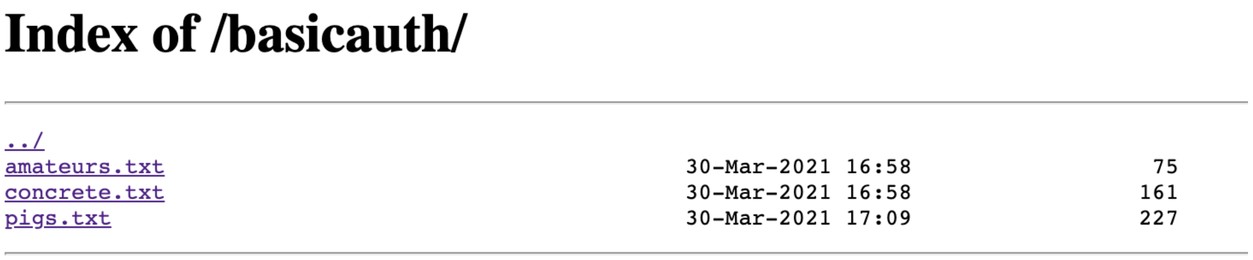
```
Hypertext Transfer Protocol
 > GET /basicauth/ HTTP/1.1\r\n
   Host: cs231.jeffondich.com\r\n
   Connection: keep-alive\r\n
   Upgrade-Insecure-Requests: 1\r\n
   Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8\r\n
   User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_6) AppleWebKit/605.1.15 (KHTML, like Gecko) Version/14.0.3 Safari/605.1.15\r\n
   Accept-Language: en-us\r\n
   Accept-Encoding: gzip, deflate\r\n
 ∨ Authorization: Basic Y3MyMzE6cGFzc3dvcmQ=\r\n
     Credentials: cs231:password
```

If for some reason you input the wrong username and password, you will keep getting stuck in a loop of packets. You'll get the GET /basicauth/ request again, and then another HTTP request saying Unauthorized after that (as seen below).



```
4 0.047572573    10.0.2.15         45.79.89.123        HTTP      479 GET /basicauth/ HTTP/1.1
5 0.095201369    45.79.89.123      10.0.2.15           HTTP      473 HTTP/1.1 401 Unauthorized  (text/html)
```

Step Four: Accessing the Files:



Index of /basicauth/

```
../
amateurs.txt                                    30-Mar-2021 16:58                  75
concrete.txt                                    30-Mar-2021 16:58                 161
pigs.txt                                        30-Mar-2021 17:09                 227
```

1. We're now at the website page shown above. After authentication, any request sent to the server to get any webpages inside the basicauth folder has the Authorization header containing the (poorly) encrypted credentials. If we click on a file in the folder, the client and server do another TCP handshake by exchanging SYN and ACK packets.
2. After the handshake, the client sends a GET request to /basicauth/pigs.txt. As mentioned previously, the GET request contains the Authorization header with the credentials inputted earlier.
3. If we try to directly go to this endpoint without going to basicauth/ first, it will still prompt us for the username and password since it's protected by HTTP Basic Authentication. When we sign in, it will add the Authorization header to the browser so that when the browser visits the page again, we won't have to authenticate again. The header will already be there.
4. We receive an HTTP 200 OK response from the server, because we are authenticated still and can now access the pigs.txt. After the response the client and server both send FIN and ACK packets to close the connection.
5. We can repeat the same process with the other two text files. The only real differences are in the filepath and in how large the packet size is (the packet size will depend on how big the text file in question is).

We’ve now made it through the whole sandbox, and found the three secret quotes!

<u>Main Takeaways:</u>

Our main takeaway from this is that HTTP Basic Authentication, while convenient, is highly insecure, as it uses base64 encoding. This is extremely easy to break as proven by the fact that Wireshark decrypts it automatically. Anyone with a packet sniffing tool (as in several enterprising CS231 students) could easily see all the passwords entered into a website. If you were using data a bit more vital than a couple of quotes, this could probably lead to a data breach rather quickly.