Ethan Li & Adela Dujsikova

HTTP's Basic Authentication

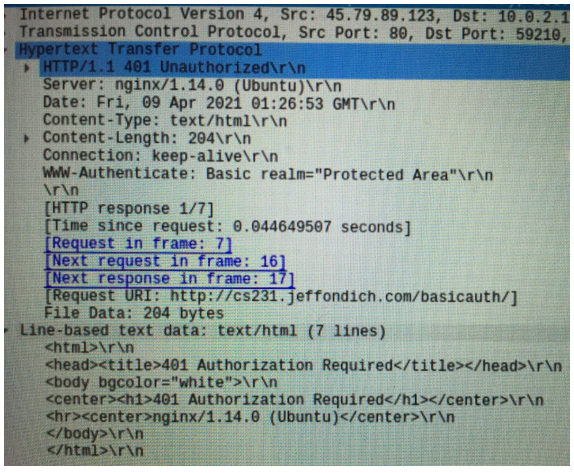
The interaction between the client and the server begins as the URL link is submitted on the web browser. Two sets of TCP three-way handshakes are initiated by the client from two different client ports. The server uses port 80 to open both of the connections. The contents of the three-way handshakes that set up the two connections are very similar, the only noticeable change was the client port number.

Interesting observation: Throughout the entire captured conversation between the client and the server, all of the HTTP data packets between the client and the server went through the second connection opened initially (port 49228 in some picture examples and port 59210 in others). Before almost every new GET request, a new connection is opened through a new client port, but the HTTP packets are always sent from the connection going through the same port. The new connection, without being directly involved in any data transfer, is then closed while the old connection remains open.

No.	Time	Source	Destination	Protocol	Length	Info
1	0	10.0.2.15	45.79.89.123	TCP	74	49226 > 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=1190009100 TSecr=0 WS=128
2	0.000046083	10.0.2.15	45.79.89.123	TCP	74	49228 > 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=1190009100 TSecr=0 WS=128
3	0.044896013	45.79.89.123	10.0.2.15	TCP	60	80 > 49226 [SYN, ACK] Seq=0 Ack=1 Win=32768 Len=0 MSS=1460
4	0.044942295	10.0.2.15	45.79.89.123	TCP	54	49226 > 80 [ACK] Seq=1 Ack=1 Win=64240 Len=0
5	0.044896058	45.79.89.123	10.0.2.15	TCP	60	80 > 49228 [SYN, ACK] Seq=0 Ack=1 Win=32768 Len=0 MSS=1460
6	0.04495711	10.0.2.15	45.79.89.123	TCP	54	49228 > 80 [ACK] Seq=1 Ack=1 Win=64240 Len=0

A GET /basicauth/ HTTP command requesting the website is then sent through the second TCP connection, and it is met with a 401 Unauthorized response from the server. The 401 response also includes a "WWW-Authenticate" header with the content 'Basic realm="Protected Area"', which defines the authentication method that should be used to gain access. This makes sense because the basic authorization is set up on the website and no credentials were sent to the server yet. The client then acknowledges the server's response.

7	0.045128612	10.0.2.15	45.79.89.123	HTTP	403	GET /basicauth/ HTTP/1.1
8	0.089857179	45.79.89.123	10.0.2.15	HTTP	473	HTTP/1.1 401 Unauthorized (text/html)
9	0.089879407	10.0.2.15	45.79.89.123	TCP	54	49228 > 80 [ACK] Seq=350 Ack=420 Win=63821 Len=0



A few seconds later, one of the two TCP connections (the first one opened which did **not** send the HTTP command) is terminated by the client, with no apparent packages sent or received during the short time of connection. In other words, the connection only existed during the unsuccessful request of the website, but didn't serve as a channel between the server and the client for exchange of any data. We are not sure what is the purpose of this connection, as the only packets relevant to it are the 3-way handshake TCP packets and then the [FIN] packets.

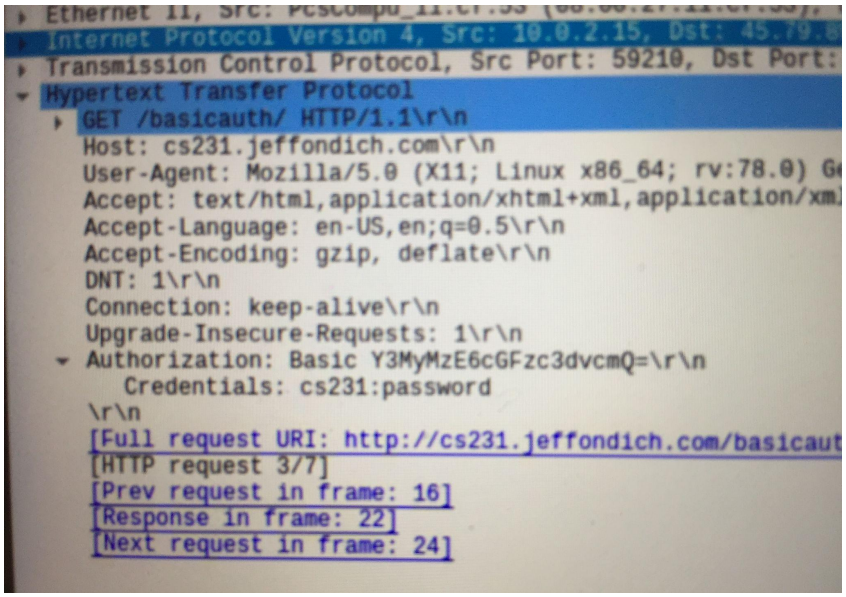
10	5.045632237	10.0.2.15	45.79.89.123	TCP	54	49226 > 80 [FIN, ACK] Seq=1 Ack=1 Win=64240 Len=0
11	5.045866436	45.79.89.123	10.0.2.15	TCP	60	80 > 49226 [ACK] Seq=1 Ack=2 Win=32767 Len=0
12	5.090555465	45.79.89.123	10.0.2.15	TCP	60	80 > 49226 [FIN, ACK] Seq=1 Ack=2 Win=32767 Len=0
13	5.090576558	10.0.2.15	45.79.89.123	TCP	54	49226 > 80 [ACK] Seq=2 Ack=2 Win=64240 Len=0

If nothing is done on the browser, several TCP Keep-Alive messages are sent from the client which is responded to by a TCP Keep-Alive ACK by the server. This interaction occurs every few seconds while the user is idle or in the process of entering their username and password. The purpose of this is to keep the connection up instead of closing it due to inactivity.

14	10.1379573	10.0.2.15	45.79.89.123	TCP	54	[TCP Keep-Alive] 49228 > 80 [ACK] Seq=349 Ack=420 Win=63821 Len=0
15	10.13812717	45.79.89.123	10.0.2.15	TCP	60	[TCP Keep-Alive ACK] 80 > 49228 [ACK] Seq=420 Ack=350 Win=32419 Len=0

Once the username and password is entered, another GET /basicauth/ HTTP command is sent to the server. Compared to the previous GET command, this packet contains a line that says "Authorization: Basic Y3MyMzE6cGFzc3dvcmQ=\r\n", which represents the Base64-encoded user-entered username and password. The details of the Base64 encoding method can be viewed here. The values of the username and password entered are simply concatenated together with a single colon character in between, before being encoded and sent to the server.

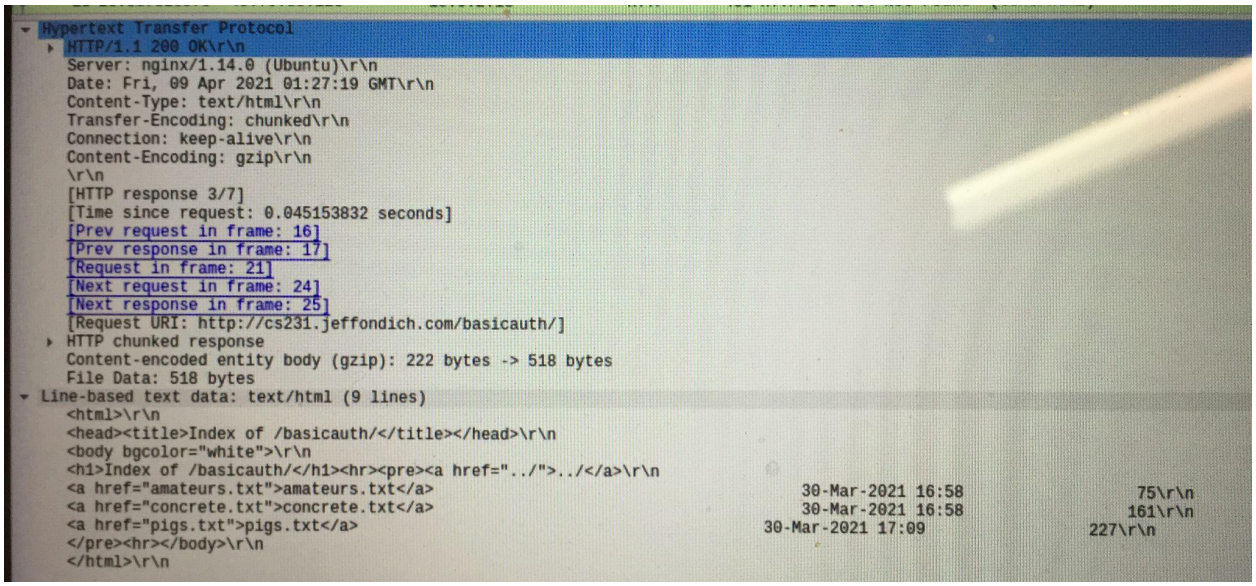
16	17.96073531	10.0.2.15	45.79.89.123	HTTP	446	GET /basicauth/ HTTP/1.1



As RFC7617 states, the 'Basic' HTTP Authentication Scheme “is not considered to be a secure method of user authentication unless used in conjunction with some external secure system”. When looking at the data packet on Wireshark, the message of the encoded correct username-password pair is automatically decoded as “Credentials: cs231:password”. The credentials in their true form are not directly present in the packet that is sent to the server, but the easily decoded Base64 version is. We know this because when we clicked on the line with the encoded credentials, the corresponding section of the packet bytes were highlighted, but when we clicked on the decoded version, there was no corresponding section in the packet.

When the username-password pair is correct, the HTTP response code from the server is 200 OK. This response also includes the contents of the webpage in html. If the login credentials are incorrect, the response from the server would be 401 Unauthorized, just like the initial response before the credentials were given to the server.

17	18.00623467	45.79.89.123	10.0.2.15	HTTP	475	HTTP/1.1 200 OK (text/html)



Another HTTP GET command is then sent to the server, asking for /favicon.ico. This request should stand for requesting the small icon that shows up on the browser on the left side of either the URL or the tab (depending on the browser). This GET command is an automated request, since modern browsers are able to display these icons. The response from the server was 404 Not Found, because there is no favicon.ico set up for the website. The GET /favicon.ico is also sent in the same way if the user clicks “cancel” when initially logging in the website.

19	18.03420339	10.0.2.15	45.79.89.123	HTTP	314	GET /favicon.ico HTTP/1.1
20	18.07928168	45.79.89.123	10.0.2.15	HTTP	401	HTTP/1.1 404 Not Found (text/html)
21	18.07930479	10.0.2.15	45.79.89.123	TCP	54	49228 > 80 [ACK] Seq=1002 Ack=1188 Win=63821 Len=0

Finally, when browsing the files listed on http://cs231.jeffondich.com/basicauth/, the following same pattern is observed when each of the linked files are clicked on the web browser:

1. A new 3-way TCP handshake is initiated by the client using a new port.
2. The file is requested by the client using the GET HTTP command, using the original connection opened at the very beginning (and not the new TCP connection opened in step 1 above); This HTTP packet also contains the Base64-encoded credentials.
3. The server sends back a 200 OK package through the same connection, acknowledging the GET command and sending along the contents of the webpage in html.
4. The client acknowledges (through the same connection) that the data was received.
5. The client initiates the closing of the new connection, which did not appear to transmit any packages during the time it was opened

22	27.38732948	10.0.2.15	45.79.89.123	TCP	74	49230 > 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=1190036487 TSecr=0 WS=128
23	27.43239863	45.79.89.123	10.0.2.15	TCP	60	80 > 49230 [SYN, ACK] Seq=0 Ack=1 Win=32768 Len=0 MSS=1460
24	27.43245123	10.0.2.15	45.79.89.123	TCP	54	49230 > 80 [ACK] Seq=1 Ack=1 Win=64240 Len=0
25	27.53917631	10.0.2.15	45.79.89.123	HTTP	507	GET /basicauth/amateurs.txt HTTP/1.1
26	27.58445487	45.79.89.123	10.0.2.15	HTTP	375	HTTP/1.1 200 OK (text/plain)
27	27.58447595	10.0.2.15	45.79.89.123	TCP	54	49228 > 80 [ACK] Seq=1455 Ack=1509 Win=63821 Len=0
28	32.43366676	10.0.2.15	45.79.89.123	TCP	54	49230 > 80 [FIN, ACK] Seq=1 Ack=1 Win=64240 Len=0
29	32.4339421	45.79.89.123	10.0.2.15	TCP	60	80 > 49230 [ACK] Seq=1 Ack=2 Win=32767 Len=0
30	32.47847494	45.79.89.123	10.0.2.15	TCP	60	80 > 49230 [FIN, ACK] Seq=1 Ack=2 Win=32767 Len=0
31	32.47849208	10.0.2.15	45.79.89.123	TCP	54	49230 > 80 [ACK] Seq=2 Ack=2 Win=64240 Len=0

Sources:

https://www.whitelist1.com/2018/04/capturing-http-basic-authentication.html
https://stackoverflow.com/questions/61302812/why-do-i-see-plaintext-credentials-in-wireshark-using-basic-auth-over-http
https://www.cisco.com/c/en/us/support/docs/security/web-security-appliance/117995-qna-wsa-00.html
https://tools.ietf.org/html/rfc7617
https://tools.ietf.org/html/rfc4648#section-4
https://developer.mozilla.org/en-US/docs/Web/HTTP/Authentication
https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/WWW-Authenticate