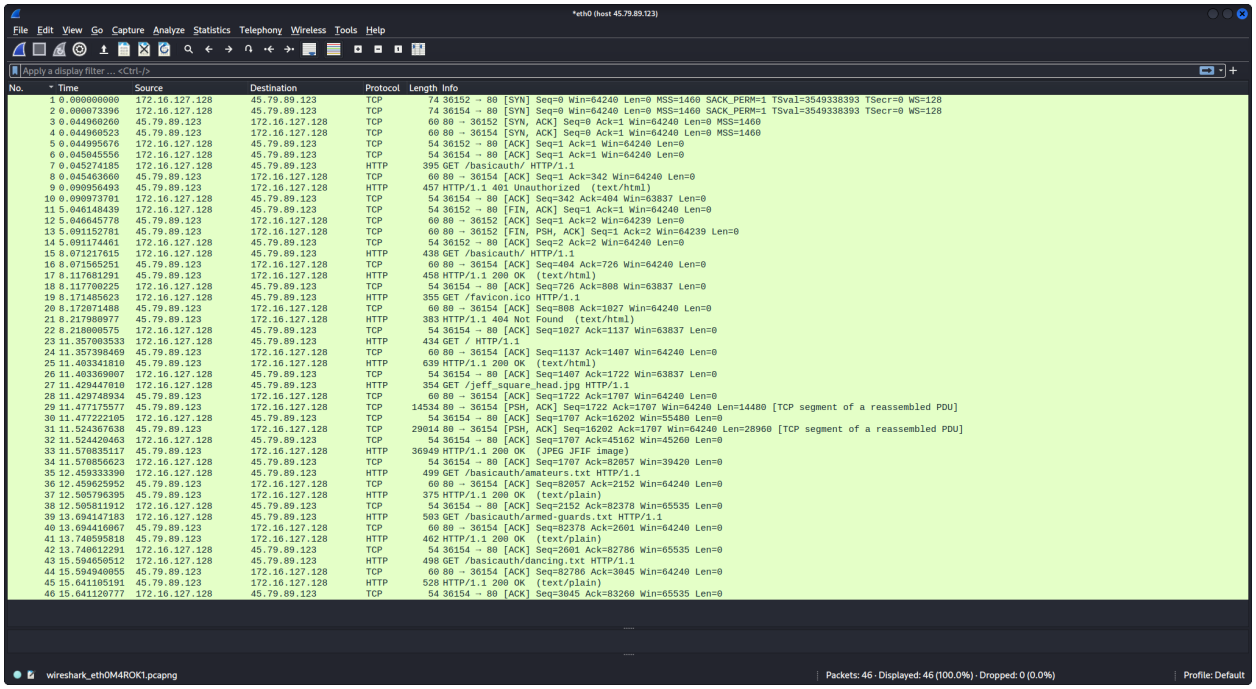
Basic-Authentication:

Oliver Clay
4/8/22



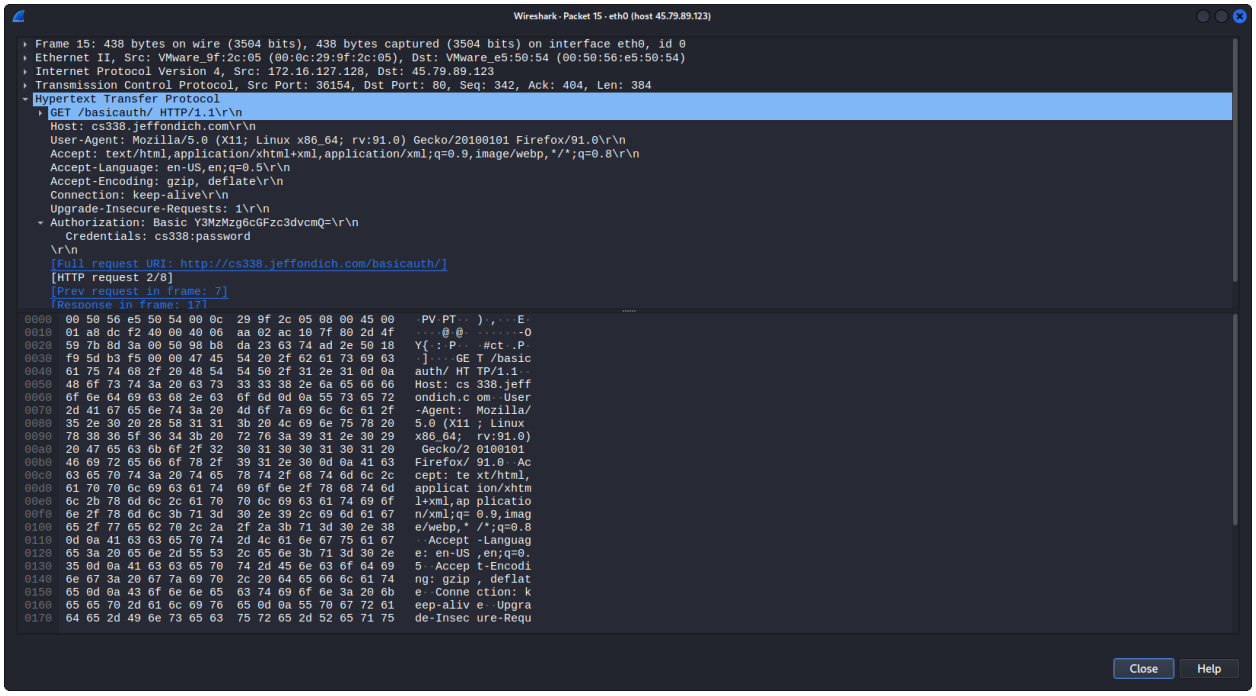
Above is an overview screenshot of the full client-server interaction when visiting the website http://cs338.jeffondich.com/basicauth/.

The first 6 frames detail the TCP handshake between the client and server. For whatever reason (perhaps multi-threading) the client sent two [SYN] requests, which, in turn, were answered by two [SYN, ACK] requests, and responded to by the client's own two [ACK] requests. Hence, the TCP handshake occurred twice, and this was consistent between multiple packet collection sessions.

In frame 7, however, things get interesting, the client is requesting "GET /basicauth/ HTTP/1.1", this seems like the client requesting the mechanism for authentication with the server. The server acknowledges this in frame 8, and then sends "Unauthorized (text/html)" in frame 9. Following this, communications are a combination of [ACK] and [FIN] between the client and

server, indicating that communications could very well end here should proper authentication not be sent by the client.

Yet, low and behold, the client sends its credentials in frame 15.



These credentials are found in the payload of the packet, under the HTTP headers, are encoded in base64, (not encrypted, no keys were visibly sent, and this observation matches my expectations from the Authentication's documentation) and are delivered to the server as plain text. I was expecting to have to dig around a little deeper to find these credentials, but apparently Basic HTTP Authentication is so readable that wireshark saw the characters Y3MzMzg6cGFzc3dvcmQ=\r\n and felt enlightened enough to decode them for me then and there, putting the username:password under its own little tab.

After this, the server sends an [ACK] followed by an OK (text/html) in frame 17 signaling that it got the client's message and found it acceptable. And... we're in.

Following this authentication sequence, things seem to happen in the manner of standard TCP communication and HTTP GET requests. I opened

each of the subsections of the website, looking at the server's sensitive information, such as jeff_square_head.jpg.

Imagine, if I (a malicious third-party) had intercepted these packets, I would have unauthorized access to this highly privy information.