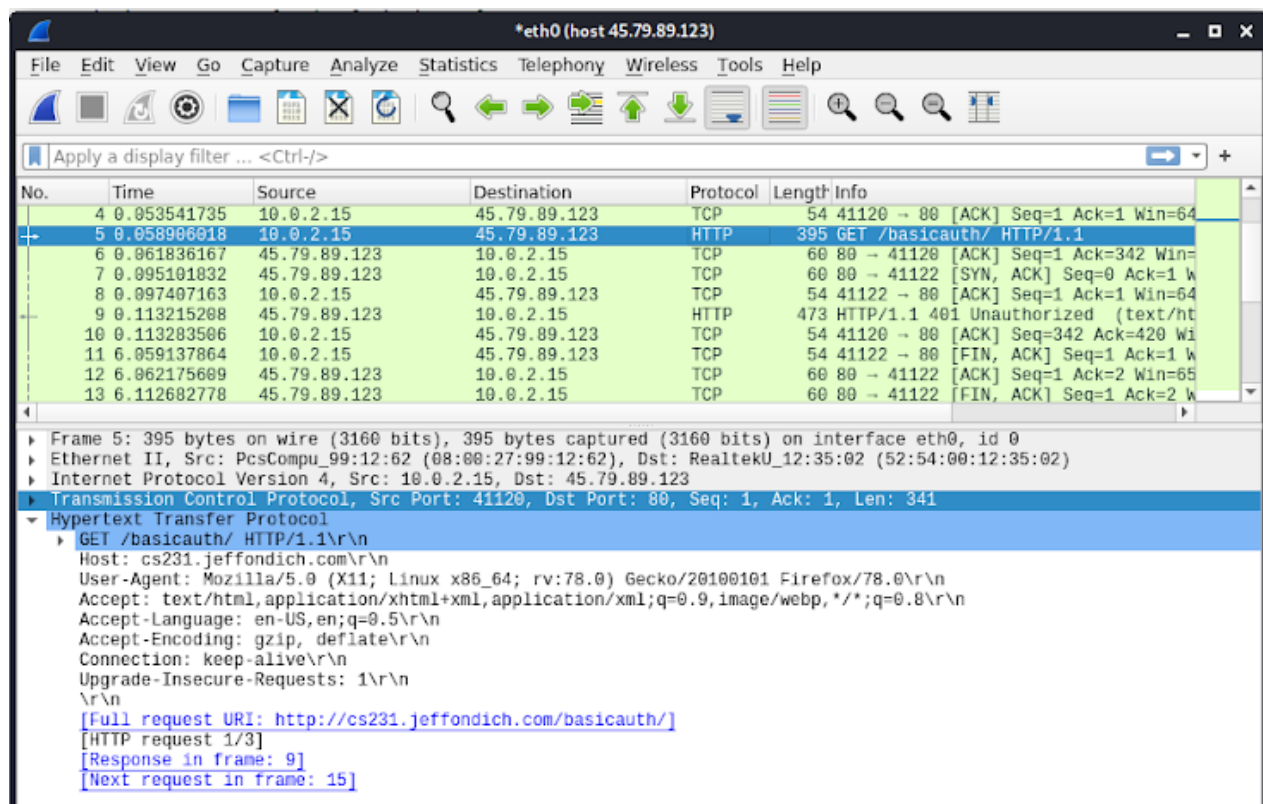


Ben Aoki-Sherwood, Avery Watts & Kenyon Nystrom

April 9, 2021

To begin our journey into <http://cs231.jeffondich.com/basicauth/>, we resolved to determine the IP address of the site. Doing so took a simple command into our terminal: “nslookup cs231.jeffondich.com”. Among other things, the response indicated that the address we were seeking lived at 45.79.89.123, as shown under “Non-authoritative answer: . . . Address:.” Next, we moved to the application Wireshark, where we navigated to the Capture menu and filtered using the command “host 45.79.89.123”.

At this point, Wireshark had begun monitoring our interactions with the server, i.e. the nginx server serving Jeff’s website. Jumping over to a web browser, we entered the url and let fire. Back in Wireshark, we were now confronted with a plethora of interaction frames. The first few were clearly a pair of standard TCP handshakes, as evidenced by the [SYN] and [SYN, ACK] tags sent between our client and the server. The first HTTP



packet was a GET request by our client for the resource /basicauth/. The information for this packet displayed in Wireshark is shown in the image above. This is a standard GET request as specified in the HTTP method specs

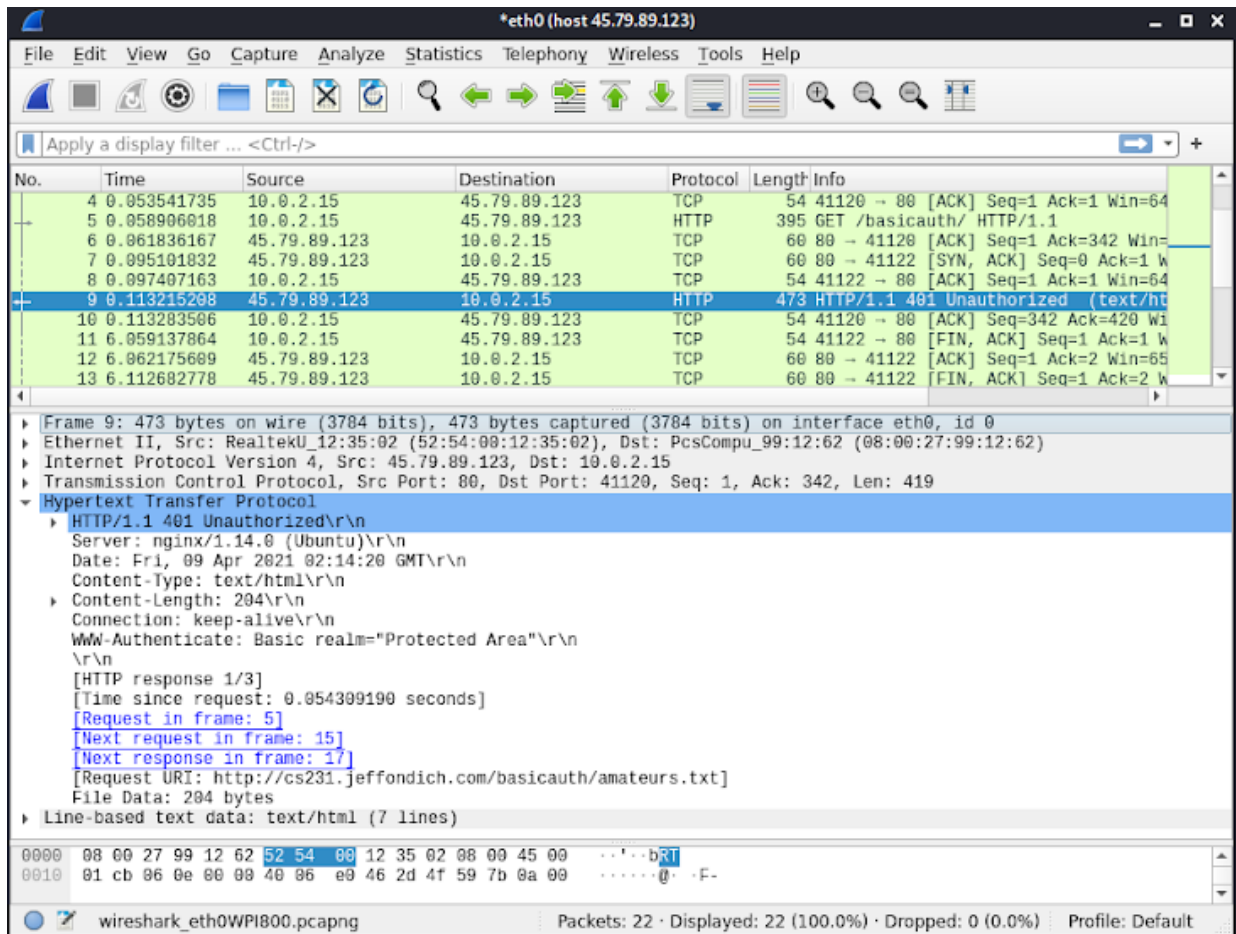
(<https://tools.ietf.org/html/rfc7231#section-4.3>); the client is asking the server for the /basicauth/ resource. As can be seen in the request headers, we were using Firefox as our browser, so this is the user-agent associated with the request. Also, note that there is no “Authorization” header field in this request. This is important, because the resource we are trying to access is a protected space; we will see later that this first request does not result in the delivery of the resource, because we are not currently authenticated to access the resource.

The server detects that we are trying to access a protected resource and responds with a challenge with the error code 401. The response includes the “WWW-Authenticate: Basic realm=’Protected Area’\r\n” header field, indicating that the authentication scheme is Basic Authentication and that the name of this protection space (realm) is “Protected Area”. This response format follows the Basic Authentication specs (<https://tools.ietf.org/html/rfc7617>). The information displayed in Wireshark for the 401 response packet is shown in the image on the next page.

As a result of this challenge, an alert popped up in our browser telling us that “http://cs231.jeffondich.com is requesting your username and password. The site says: ‘Protected Area’”, with text boxes for us to input the username and password.

After inputting the credentials in the alert bar in the web browser, we entered the protected space. Back in Wireshark, we clicked through the packets sent and noticed a second GET request for the /basicauth/ resource. Unlike the first request, this one had

an Authorization header indicating that the authentication scheme was “Basic” and containing our credentials encoded in base64. Wireshark provided a “Credentials”



sub-field showing us the plain text. The format of request Authorization header and the user-password format *username:password* follows the Basic Authentication spec. This information was not encrypted. The packet information for the second GET request is shown in the image below. Because this second GET request contained the appropriate authentication information, the server responded with a final HTTP 200 OK response to

The image shows a Wireshark packet capture analysis of an HTTP 401 Unauthorized response. The packet list on the left shows a GET request (frame 15) and its corresponding 401 response (frame 16). The packet details pane on the right shows the structure of the response, including the status bar (401 Unauthorized), the response body (HTML), and the authentication header (Basic). The packet bytes pane at the bottom shows the raw data of the response, including the status bar and the authentication header.

No.	Time	Source	Destination	Protocol	Length	Info
7	0.095101832	45.79.89.123	10.0.2.15	TCP	60	80 → 41122 [SYN, ACK] Seq=0 Ack=1 W
8	0.097407163	10.0.2.15	45.79.89.123	TCP	54	41122 → 80 [ACK] Seq=1 Ack=1 Win=64
9	0.113215288	45.79.89.123	10.0.2.15	HTTP	473	HTTP/1.1 401 Unauthorized (text/ht
10	0.113283506	10.0.2.15	45.79.89.123	TCP	54	41120 → 80 [ACK] Seq=342 Ack=420 Wi
11	6.059137864	10.0.2.15	45.79.89.123	TCP	54	41122 → 80 [FIN, ACK] Seq=1 Ack=1 W
12	6.062175689	45.79.89.123	10.0.2.15	TCP	60	80 → 41122 [ACK] Seq=1 Ack=2 Win=65
13	6.112682778	45.79.89.123	10.0.2.15	TCP	60	80 → 41122 [FIN, ACK] Seq=1 Ack=2 W
14	6.112745978	10.0.2.15	45.79.89.123	TCP	54	41122 → 80 [ACK] Seq=2 Ack=2 Win=64
15	9.104791029	10.0.2.15	45.79.89.123	HTTP	438	GET /basicauth/ HTTP/1.1
16	9.105680976	45.79.89.123	10.0.2.15	TCP	60	80 → 41120 [ACK] Seq=420 Ack=726 Wi

Frame 15: 438 bytes on wire (3504 bits), 438 bytes captured (3504 bits) on interface eth0, id 0  
 Ethernet II, Src: PcsCompu 99:12:62 (08:00:27:99:12:62), Dst: RealtekU 12:35:02 (52:54:00:12:35:02)  
 Internet Protocol Version 4, Src: 10.0.2.15, Dst: 45.79.89.123  
 Transmission Control Protocol, Src Port: 41120, Dst Port: 80, Seq: 342, Ack: 420, Len: 384  
 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  
 Connection: keep-alive\r\n  
 Upgrade-Insecure-Requests: 1\r\n  
 Authorization: Basic Y3MyZzE6cGFzc3dvcmQ=\r\n  
 Credentials: cs231:password  
 \r\n  
 [Full request URI: http://cs231.jeffondich.com/basicauth/]  
 [HTTP request 2/3]  
 [Prev request in frame: 5]  
 [Response in frame: 17]  
 [Next request in frame: 19]

0180 65 73 74 73 3a 20 31 0d 0a 41 75 74 68 6f 72 69 ests: 1:  
 0190 ya 61 74 69 6f 6e 3a 20 42 61 73 69 63 20 59 33 zation: Basic

HTTP Authorization header (application/javascript), 43 bytes Packets: 22 · Displayed: 22 (100.0%) · Dropped: 0 (0.0%) Profile: Default

Throughout our communications with the nginx web server, we noticed TCP Keep-Alive messages, which we understand to be normal network procedures which

ensure connection between the client and the server. Related to the Keep-Alive messages, RST (reset) messages occasionally occurred. These messages reset the TCP handshake with the browser. To find out more about these messages, we viewed [https://en.wikipedia.org/wiki/Keepalive#TCP\\_keepalive](https://en.wikipedia.org/wiki/Keepalive#TCP_keepalive).

Another interesting point we noticed was the fact that the HTTP communication included a correction for the error of entering the /basicauth site without a closing backslash. Wireshark showed a 301 error, essentially redirecting the initial request to instead travel to the correct /basicauth/ url. This is described in the HTTP snippets below:

Client to server -- GET /basicauth HTTP/1.1

Server to client -- HTTP/1.1 301 Moved Permanently

Client to server -- GET /basicauth/ HTTP/1.1

This is a feature that we had certainly taken for granted prior to understanding interactions using Wireshark.

In summary, to navigate through Jeff's website, we first inputted our credentials, then pursued through the files in the secret folder. Through Wireshark, we followed the packets being sent back and forth between us and the server hosting Jeff's website. Because the passwords were sent through basic authentication, it was easy to snoop and find the plaintext password, even though it was encoded in base64. Base64 is not actually encryption, so it is clear that basic authentication is simply not secure in the current context of computer privacy. All in all, Wireshark provided us with many useful insights in this exploration of web server-client communication and Basic Authentication security.