HTTP's Basic Authentication: A Story

Beshir Said

To explore the details and limitations of Basic Authentication, I used BurpSuite Community Edition and WireShark inside the Kali Linux. BurpSuite is a software tool used for analyzing security and discovering the ins and outs of HTTP requests and responses. Wireshark is a software tool that allows you to monitor the HTTP traffic.

I first used Wireshark to look at the initial exchange between the client and server. Figure 6 shows the frames of the initial TCP handshake that HTTP traffic uses. Then, I loaded Jeff's nginx server http://cs338.jeffondich.com/basicauth/ into BurpSuite without signing in. The initial request inside the HTTP history came back with 401 Authorization Required message with the Authorization header of WWW-Authenticate: Basic realm = "Protected Area" (Figures 1-2). The Authorization header is the part of the HTTP request that carries the clients credentials to the server, so the server can verify the clients identity.

Once I entered the username and password, the browser resent the request with a Base64-encoded string (Highlighted in Figure 3). I decoded the Base64 string inside the terminal and received this output: cs338:password (Figure 5). This tells us the password was sent from the browser to the server from the Authorization header. After this, the server responded with a HTTP/1.1 200 OK and gave me the contents of the secret directory (Figure 4). The curl command confirmed the same interaction, showing the earlier 401 message and later the successful 200 response (Figure 3).

This experiment shows how HTTP Basic Authentication works but at the same time how it is insecure. The flow from 401 Unauthorized to the Authorization Header to 200 OK shows how the browser and server interact. However, the credentials are Base64-encoded and anyone monitoring the plaintext HTTP traffic can recover them on a network like Wireshark. I think for this reason, one should try to use stronger authentication methods and also note that Basic Authentication should always be used over HTTPS. Unlike HTTP, HTTPS wraps the content inside TLS/SSL encryption, meaning intruders cannot read the requests, responses, or headers. When using HTTPS, the encryption key is exchanged in the TLS handshake and those keys are then used to encrypt the HTTPS headers and body.

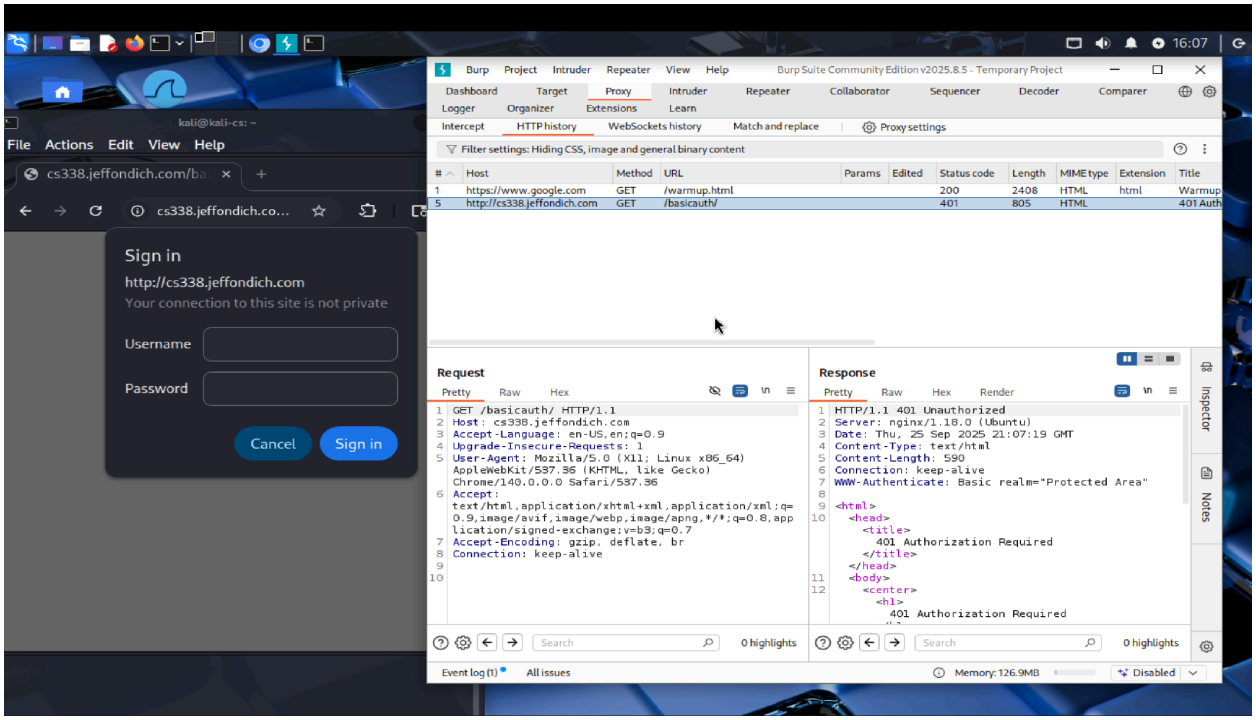

Figure 1: 401 Authorization Required before signing in

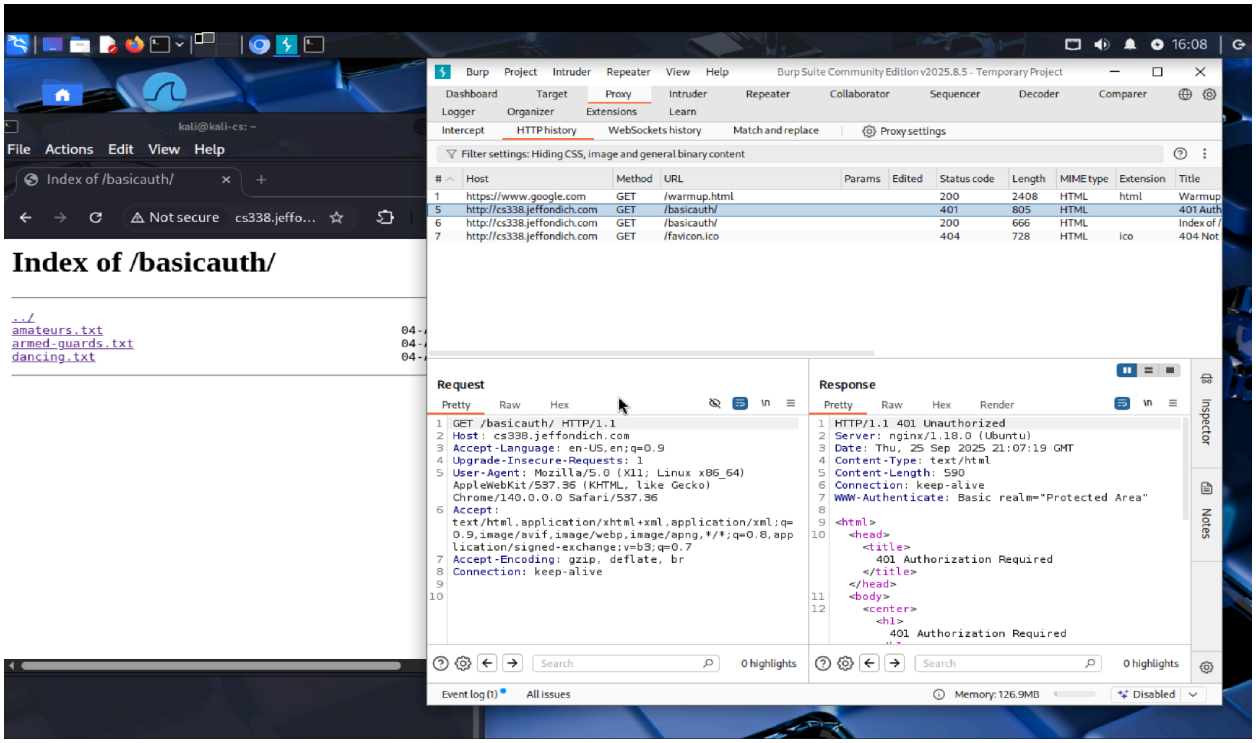

Figure 2: 401 Unauthorized with WWW-Authenticate Basic realm = “Protected Area”

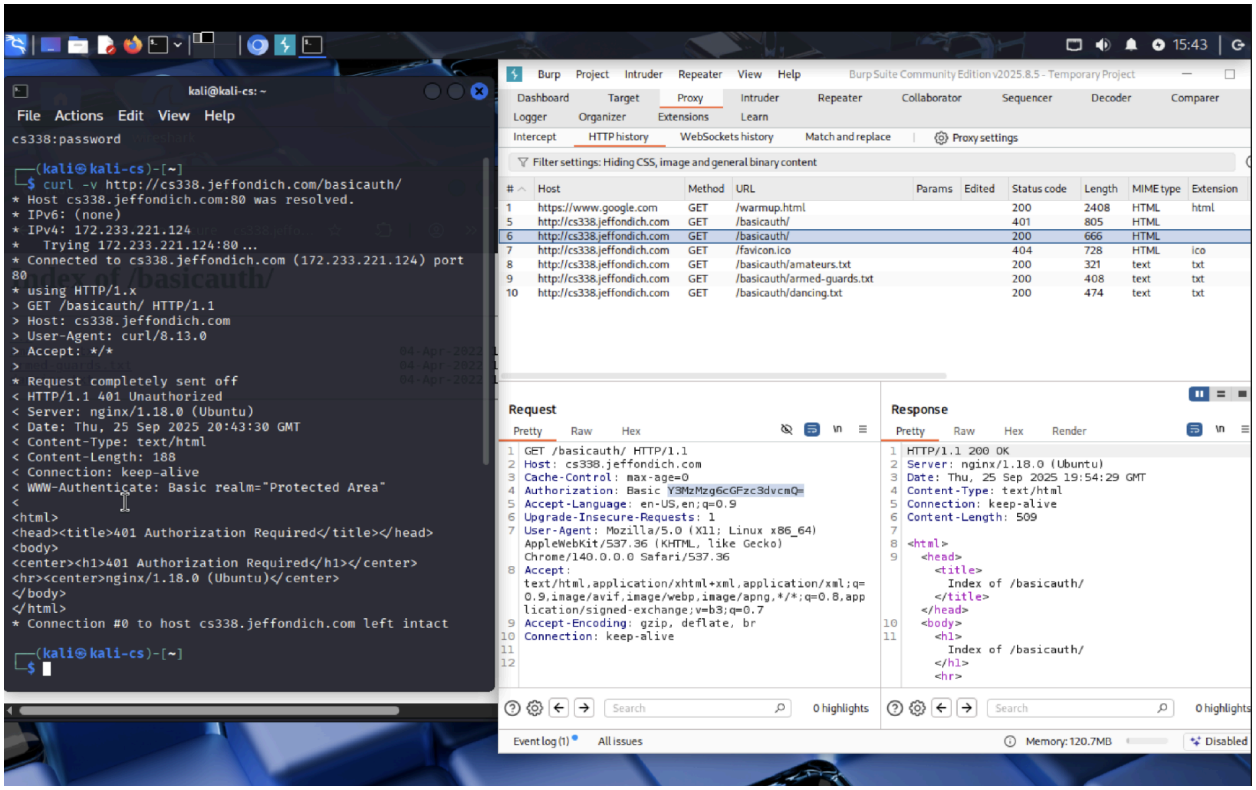

Figure 3: curl output confirming the 401

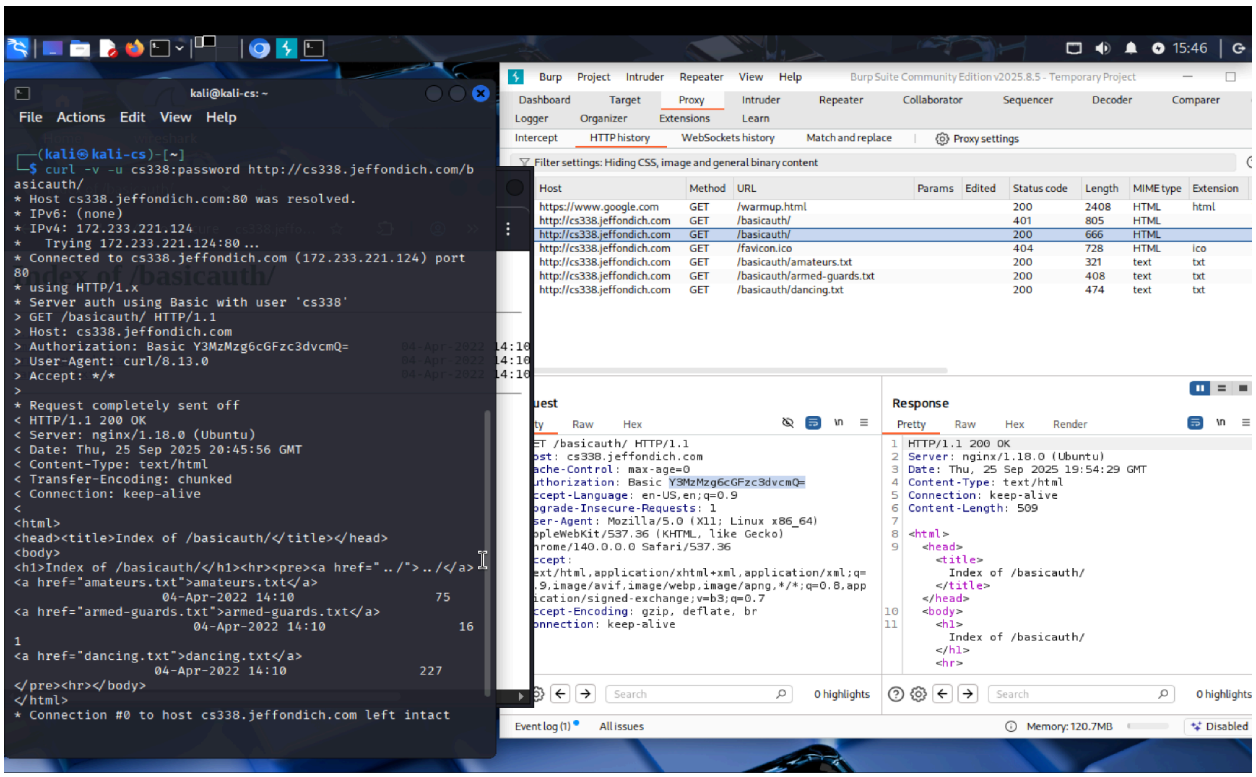

Figure 4: Successful Authorized request with 200 OK with content listing

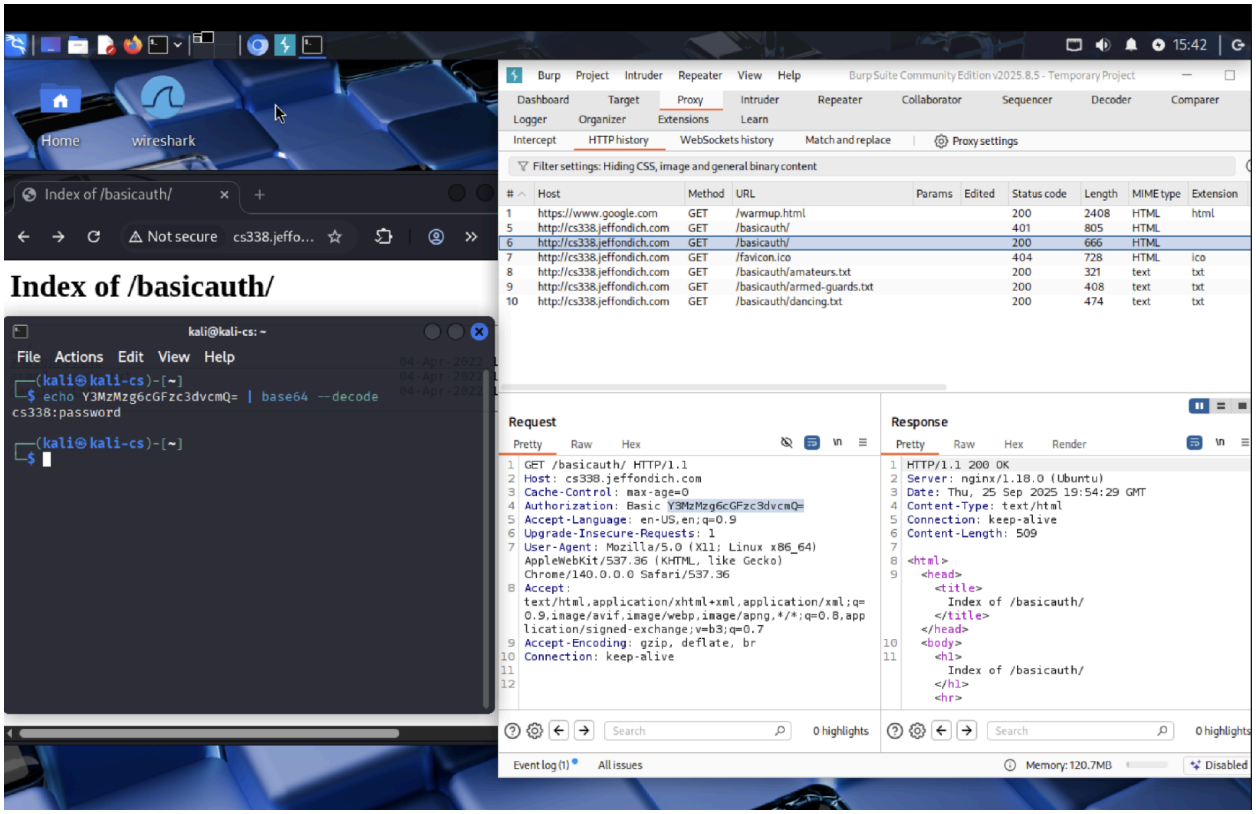

Figure 5: Request containing Authorization Basic Header and base64 decoding

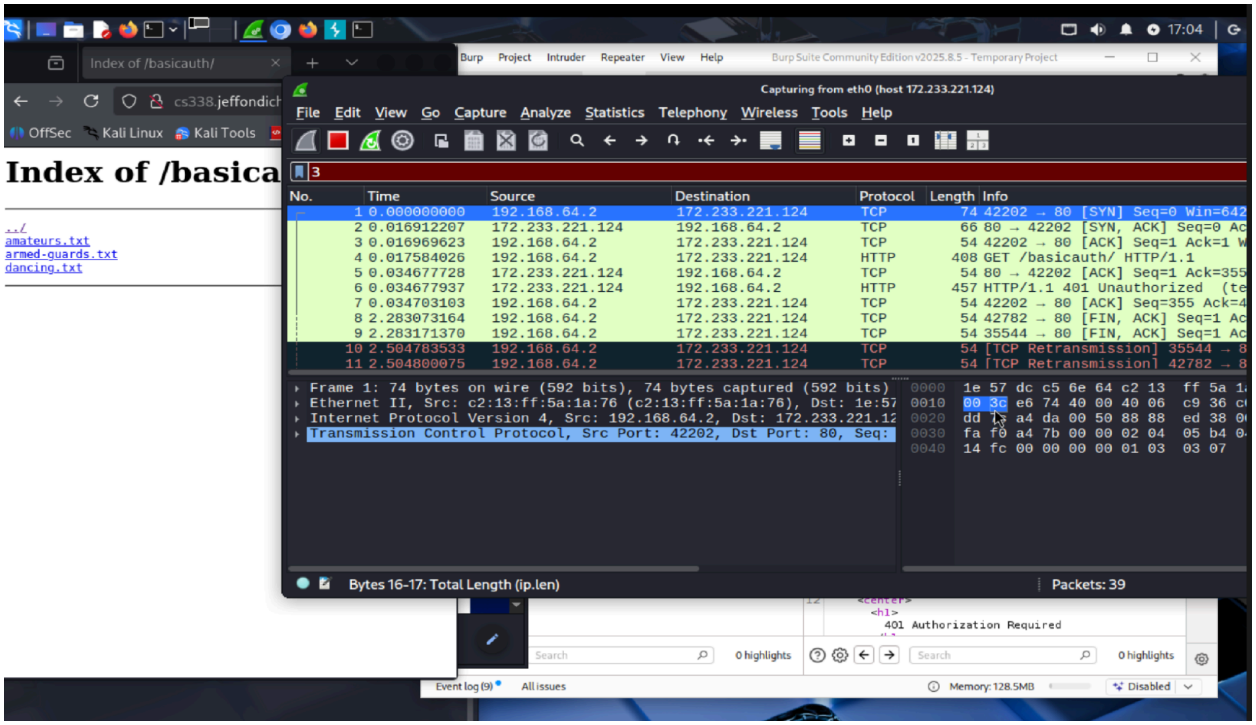

Figure 6: Wireshark TCP handshake