

HTTP Basic Authentication Investigation

Investigating with Curl

Observations

```
[kali㉿kali: ~]
$ curl -v http://cs338.jeffondich.com/basicauth/
* Host cs338.jeffondich.com:80 was resolved.
* IPv6: (none)
* IPv4: 172.233.221.124
* Trying 172.233.221.124:80...
* Connected to cs338.jeffondich.com (172.233.221.124) port 80
> GET /basicauth/ HTTP/1.1
> Host: cs338.jeffondich.com
> User-Agent: curl/8.8.0
> Accept: */*
>
* Request completely sent off
< HTTP/1.1 401 Unauthorized
< Server: nginx/1.18.0 (Ubuntu)
< Date: Tue, 24 Sep 2024 13:40:08 GMT
< Content-Type: text/html
< Content-Length: 188
< Connection: keep-alive
< WWW-Authenticate: Basic realm="Protected Area"
<
<html>
<head><title>401 Authorization Required</title></head>
<body>
<center><h1>401 Authorization Required</h1></center>
<hr><center>nginx/1.18.0 (Ubuntu)</center>
</body>
</html>
* Connection #0 to host cs338.jeffondich.com left intact
```

Because curl simply does a GET request for the page at /basicauth, it is met with a “401 Unauthorized” response. I wonder what kind of request has to be made to enable authentication to happen. Could one authenticate through netcat for instance? Or is it only possible through a browser window?

There were two things that stood out to me:

- * Request completely sent off
 - I was unsure what that meant but trying to google it wasn't much help. From what I can gather it doesn't seem to be anything of note
- WWW-Authenticate: Basic realm="Protected Area"
 - I wanted to take note of this header before I accessed the webpage as I have a feeling that this might be the name on the dialog window that pops up asking for authentication. Or it could just be a setting that means authentication is required. (Update: it's not)

Investigating with Wireshark and Burpsuite

Main Observations

I decided to combine Wireshark with Burpsuite at first to see what that would reveal. The first thing I noticed is that there seemed to be an issue switching from https to http. Perhaps I clicked the buttons in the wrong order, but it took several forwards and Continue to sites before I was finally able to access the web page. However, because humans are slow and because I had Wireshark loaded as well, I can kind of tell which of the requests was actually sent using http.

```
1 | GET /basicauth HTTP/1.1
2 | Host: cs338.jeffondich.com
3 | Cache-Control: max-age=0
4 | Accept-Language: en-US
5 | Upgrade-Insecure-Requests: 1
6 | User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
   |   AppleWebKit/537.36 (KHTML, like Gecko) Chrome/127.0.6533.100
   |   Safari/537.36
7 | Accept:
   |   text/html,application/xhtml+xml,application/xml;q=0.9,image/avif
   |   ,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b
   |   3;q=0.7
8 | Accept-Encoding: gzip, deflate, br
9 | Connection: keep-alive
10|
11|
```

I noticed that several GET requests were made for the web page. I wonder perhaps if this initial GET request was met with a 401 Error which instructed the browser to send a new GET request with the appropriate Authorization header.

Update: It was! The initial GET request was met with a 401 Unauthorized response. This prompted the browser to send the request again but with the Authorization header included.

Request	Response
<pre> 1 GET /basicauth/ HTTP/1.1 2 Host: cs338.jeffondich.com 3 Cache-Control: max-age=0 4 Upgrade-Insecure-Requests: 1 5 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/127.0.6533.100 Safari/537.36 6 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7 7 Accept-Language: en-US 8 Accept-Encoding: gzip, deflate, br 9 Connection: keep-alive 10 11 </pre>	<pre> 1 HTTP/1.1 401 Unauthorized 2 Server: nginx/1.18.0 (Ubuntu) 3 Date: Tue, 24 Sep 2024 13:57:46 GMT 4 Content-Type: text/html 5 Content-Length: 590 6 Connection: keep-alive 7 WWW-Authenticate: Basic realm="Protected Area" 8 9 <html> 10 <head><title>401 Authorization Required</title></head> 11 <body> 12 <center><h1>401 Authorization Required</h1></center> 13 <hr><center>nginx/1.18.0 (Ubuntu)</center> 14 </body> 15 </html> 16 <!!-- a padding to disable MSIE and Chrome friendly 17 <!!-- a padding to disable MSIE and Chrome friendly 18 <!!-- a padding to disable MSIE and Chrome friendly 19 <!!-- a padding to disable MSIE and Chrome friendly 20 <!!-- a padding to disable MSIE and Chrome friendly 21 <!!-- a padding to disable MSIE and Chrome friendly 22 <!!-- a padding to disable MSIE and Chrome friendly </pre>
<input type="button" value="①"/> <input type="button" value="⚙"/> <input type="button" value="↶"/> <input type="button" value="↷"/> <input type="text" value="Search"/> <input type="button" value="⌕"/> 0 highlights	<input type="button" value="②"/> <input type="button" value="⚙"/> <input type="button" value="↶"/> <input type="button" value="↷"/> <input type="text" value="Search"/> <input type="button" value="⌕"/> 0 highlights

Request to http://cs338.jeffondich.com:80 [172.23.221.124]

Pretty	Raw	Hex
<pre> 1 GET /basicauth/ HTTP/1.1 2 Host: cs338.jeffondich.com 3 Cache-Control: max-age=0 4 Authorization: Basic Y3MzMzg6cGFzc3dvcmQ= 5 Accept-Language: en-US 6 Upgrade-Insecure-Requests: 1 7 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/127.0.6533.100 Safari/537.36 8 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif ,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b 3;q=0.7 9 Accept-Encoding: gzip, deflate, br 10 Connection: keep-alive 11 12 </pre>		

This is the second GET request made by the browser. Most of it is the same except for the added “Authorization: Basic Y3MzMzg6cGFzc3dvcmQ=”

I’m assuming this prompted the server to ask for authentication. However I’m not sure what “Basic Y3MzMzg6cGFzc3dvcmQ=” means in this context.

Another update: Knowing how the basic HTTP authentication protocol works, I decided to decode "Y3MzMzg6cGFzc3dvcmQ=" through base64. It is in fact the username and password!

This reveals that the server does the authenticating because it sends a 200 OK response after that.

Request		Response	
Pretty	Raw	Pretty	Raw
1 GET /basicauth/ HTTP/1.1		1 HTTP/1.1 200 OK	
2 Host: cs338.jeffondich.com		2 Server: nginx/1.18.0 (Ubuntu)	
3 Cache-Control: max-age=0		3 Date: Tue, 24 Sep 2024 13:58:24 GMT	
4 Authorization: Basic Y3MzMzg6cGFzc3dvcmQ=		4 Content-Type: text/html	
5 Accept-Language: en-US		5 Connection: keep-alive	
6 Upgrade-Insecure-Requests: 1		6 Content-Length: 509	
7 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/127.0.6533.100 Safari/537.36		7	
8 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7		8 <html>	
9 Accept-Encoding: gzip, deflate, br		9 <head><title>Index of /basicauth/</title></head>	
10 Connection: keep-alive		10 <body>	
11		11 <h1>Index of /basicauth/</h1><hr><pre>	
12		12 amateurs.txt	
		13 armed-guards.txt	04-Apr-2022 14:10
		14 dancing.txt	04-Apr-2022
		15 </pre><hr></body>	04-Apr-2022
		16 </html>	
		17	

I was able to see with Wireshark that after the 401 Unauthorized Error, the browser initiated another TCP handshake. It then made a GET request with the username and password I entered in the header.

Frame 168: 508 bytes on wire (4544 bits), 508 bytes captured (4544 bits) on interface eth0, id 0

Ethernet II, Src: VMware_da:be:26 (00:0c:29:da:be:26), Dst: VMware_e3:b7:bb (00:50:56:e3:b7:bb)

Internet Protocol Version 4, Src: 192.168.204.129, Dst: 172.233.221.124

0100 = Header Length: 20 bytes (5)

Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)

Total Length: 554

010. = Flags: 0x2, Don't fragment

... 0 0000 0000 0000 = Fragment Offset: 0

Time to Live: 64

Protocol: TCP (6)

Header Checksum: 0xf458 [validation disabled]

[Header checksums stripped/unverified]

Source Address: 192.168.204.129

Destination Address: 172.233.221.124

Transmission Control Protocol, Src Port: 48644, Dst Port: 80, Seq: 1, Ack: 1, Len: 514

Hypertext Transfer Protocol

GET /basicauth/ HTTP/1.1\r\n

Host: cs338.jeffondich.com\r\n

Cache-Control: max-age=0\r\n

Authorization: Basic Y3MzMzg6cGFzc3dvcmQ=\r\n

Accept-Language: en-US\r\n

Upgrade-Insecure-Requests: 1\r\n

Frame 000: 67 65 3d 30 00 0a 41 75 74 68 61 0000 69 5f 6e 3a 20 42 61 73 69 63 20

0001 7a 67 36 63 47 46 78 63 33 64 74

0002 64 40 63 60 65 70 2d 4c 66 61

0003 3d 20 43 68 55 53 60 68 55 71

0004 49 60 73 65 53 72 45 2d 51

0005 74 73 3a 20 31 00 0a 55 73 65 72

0006 74 3a 20 4d 6f 74 69 6c 66 61 21

0100 57 69 66 64 6f 77 73 20 4e 54 20

0110 20 57 69 36 3b 20 78 36 34

0120 6c 65 57 65 62 4b 69 74 2f 35 3:

0130 28 4b 48 54 4d 4c 2c 20 66 69 61

0140 66 6f 29 20 43 68 72 6f 60 65 21

0150 2e 35 35 33 28 37 30 30 20 5:

0160 2d 25 37 33 23 36 60 61 60 65

0170 74 65 78 74 2f 68 74 2d 66 21

0180 63 61 74 69 6f 6e 2f 78 69 74 60

0190 2c 61 70 78 6c 69 63 61 74 69 61

01a0 3b 71 3d 39 29 36 29 69 6d 61 61

01b0 66 2c 69 6d 61 67 65 2f 77 65 6:

01c0 67 65 2f 61 70 66 67 2c 2a 2f 2:

01d0 38 2c 61 70 70 6c 69 63 61 74 6:

Frame (568 bytes) Basic Credentials (14 bytes)

What I'm unsure about is who initiated the pop up. At this point it seems like it was the browser. But it seems like it should be a feature of the website so I'm not sure.

What is clear is that the server does the authenticating. I'm just not sure if the pop-up is a feature of the web page or the browser.

Additional Observations

I decided to look at all the traffic rather than just traffic to cs338.jeffondich.com, I learned that even just between your browser and your machine, there is a lot of communication. There is a TCP connection set up and they agree upon a cipher for TLS.

I opened up an incognito window and I found a lot of requests between 172.217.0.170, which after running that through ARIN, I learned was google.

Usually whenever I'm looking at Wireshark, I filter out my results but it was interesting to see what other communication is happening between the browser and Kali. Especially when you are presented with one of those "This website is insecure pages"

I also saw mentions of *safebrowsing.googleapis.com* which drew my attention. I was curious about whether that applied to just the incognito windows or if it was always present.

20 66.848571948	192.168.204.129	172.217.0.170	TCP	74 54062 - 443 [SYN] Seq=0 Win=32120 Len=0 MSS=1460 SACK_PERM TStamp=3218685740 TSecr=0 WS=1024
21 66.8677761318	172.217.0.170	192.168.204.129	TCP	60 443 - 54062 [ACK] Seq=1 Ack=1 Win=32120 Len=0
22 66.867881848	192.168.204.129	172.217.0.170	TCP	54 54062 - 443 [ACK] Seq=1 Ack=1 Win=32120 Len=0
23 66.871993869	192.168.204.129	172.217.0.170	TLSv1.3	726 Client Hello (SNI=safebrowsing.googleapis.com)
24 66.872423191	172.217.0.170	192.168.204.129	TCP	60 443 - 54062 [ACK] Seq=1 Ack=673 Win=64240 Len=0
25 66.873235044	192.168.204.129	172.217.0.170	TLSv1.3	60 Change Cipher Spec
26 66.873289330	192.168.204.129	172.217.0.170	TLSv1.3	813 Application Data
27 66.873428291	172.217.0.170	192.168.204.129	TCP	60 443 - 54062 [ACK] Seq=1 Ack=679 Win=64240 Len=0
28 66.873501201	172.217.0.170	192.168.204.129	TCP	60 443 - 54062 [ACK] Seq=1 Ack=1439 Win=64240 Len=0
29 66.899292550	172.217.0.170	192.168.204.129	TLSv1.3	884 Server Hello, Change Cipher Spec, Application Data, Application Data, Application Data
30 66.899292550	192.168.204.129	172.217.0.170	TCP	54 54062 - 443 [ACK] Seq=1438 Ack=831 Win=31540 Len=0
31 66.1002180824	192.168.204.129	172.217.0.170	TLSv1.3	138 Application Data, Application Data
32 66.100463812	172.217.0.170	192.168.204.129	TCP	60 443 - 54062 [ACK] Seq=831 Ack=1522 Win=64240 Len=0
33 66.101632865	192.168.204.129	172.217.0.170	TLSv1.3	85 Application Data
34 66.101879026	172.217.0.170	192.168.204.129	TCP	60 443 - 54062 [ACK] Seq=831 Ack=1553 Win=64240 Len=0
35 66.107725161	172.217.0.170	192.168.204.129	TLSv1.3	5227 Application Data, Application Data, Application Data, Application Data, Application Data, Application Data, Application Data
36 66.107743877	192.168.204.129	172.217.0.170	TCP	54 54062 - 443 [ACK] Seq=1553 Ack=6094 Win=31540 Len=0
37 66.110269224	192.168.204.129	172.217.0.170	TLSv1.3	93 Application Data
38 66.110496344	192.168.204.129	172.217.0.170	TLSv1.3	78 Application Data
39 66.110533933	192.168.204.129	172.217.0.170	TCP	54 54062 - 443 [FIN, ACK] Seq=1016 Ack=6094 Win=31540 Len=0
40 66.110606720	172.217.0.170	192.168.204.129	TCP	60 443 - 54062 [ACK] Seq=6094 Ack=1592 Win=64240 Len=0
41 66.110764883	172.217.0.170	192.168.204.129	TCP	60 443 - 54062 [ACK] Seq=6094 Ack=1616 Win=64240 Len=0
42 66.110868214	172.217.0.170	192.168.204.129	TCP	60 443 - 54062 [ACK] Seq=6094 Ack=1617 Win=64239 Len=0
43 66.126738852	172.217.0.170	192.168.204.129	TCP	60 443 - 54062 [FIN, PSH, ACK] Seq=6094 Ack=1617 Win=64239 Len=0
44 66.126786476	192.168.204.129	172.217.0.170	TCP	54 54062 - 443 [ACK] Seq=1617 Ack=6095 Win=31540 Len=0

Useful acronyms:

- SNI: Server Name Identification
 - Client indicates which host it is trying to connect with.

Investigating with Wireshark

Observations

Since my first experiments with both Wireshark and Burpsuite included a little noise and a few unanswered questions, I decided to look at each one on its own.

One thing I noticed this time that I didn't before was the presence of TCP Keep Alive packets.

I'm assuming the browser was prompting the server to not close the connection as I typed the username and password.

1 0.000000000 192.168.204.129	172.233.221.124	TCP	74 52328 - 80 [SYN] Seq=0 Win=32120 Len=0 MSS=1460 SACK_PERM TSval=908855631 TSeср=0 WS=1024
2 0.033811361 172.233.221.124	192.168.204.129	TCP	68 80 - 52328 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
3 0.034112792 192.168.204.129	172.233.221.124	TCP	54 52328 - 80 [ACK] Seq=1 Ack=1 Win=32120 Len=0
4 0.034524889 192.168.204.129	172.233.221.124	HTTP	409 GET /basicauth/ HTTP/1.1
5 0.035196734 172.233.221.124	192.168.204.129	TCP	68 80 - 52328 [ACK] Seq=1 Ack=356 Win=64240 Len=0
6 0.065043624 172.233.221.124	192.168.204.129	HTTP	457 HTTP/1.1 401 Unauthorized (text/html)
7 0.065112029 192.168.204.129	172.233.221.124	TCP	54 52328 - 80 [ACK] Seq=356 Ack=404 Win=31717 Len=0
8 10.197725906 192.168.204.129	172.233.221.124	TCP	54 [TCP Keep-Alive] 52328 - 80 [ACK] Seq=355 Ack=404 Win=31717 Len=0
9 11.222333584 192.168.204.129	172.233.221.124	TCP	54 [TCP Keep-Alive] 52328 - 80 [ACK] Seq=355 Ack=404 Win=31717 Len=0
10 11.222836457 172.233.221.124	192.168.204.129	TCP	68 [TCP Keep-Alive ACK] 80 - 52328 [ACK] Seq=404 Ack=356 Win=64240 Len=0
11 11.369769913 192.168.204.129	172.233.221.124	HTTP	452 GET /basicauth/ HTTP/1.1
12 11.370167830 172.233.221.124	192.168.204.129	TCP	68 80 - 52328 [ACK] Seq=404 Ack=754 Win=64240 Len=0
13 11.399421229 172.233.221.124	192.168.204.129	HTTP	458 HTTP/1.1 200 OK (text/html)
14 11.399475511 192.168.204.129	172.233.221.124	TCP	54 52328 - 80 [ACK] Seq=754 Ack=808 Win=31717 Len=0
15 11.667399609 192.168.204.129	172.233.221.124	HTTP	369 GET /favicon.ico HTTP/1.1
16 11.667914147 172.233.221.124	192.168.204.129	TCP	68 80 - 52328 [ACK] Seq=808 Ack=1069 Win=64240 Len=0
17 11.712481164 172.233.221.124	192.168.204.129	HTTP	383 HTTP/1.1 404 Not Found (text/html)
18 11.712540196 192.168.204.129	172.233.221.124	TCP	54 52328 - 80 [ACK] Seq=1069 Ack=137 Win=31717 Len=0

I decided to also try using the wrong password and it once again just showed a series of GET requests each with the wrong encoding. What's interesting is that after you decide to cancel the sign in, your browser still has to send another GET request before returning with the 401 error. It has me wondering what other response is possible.

13 17.691102906 172.233.221.124	192.168.204.129	TCP	60 80 - 33886 [ACK] Seq=807 Ack=1180 Win=64240 Len=0
14 17.727373545 172.233.221.124	192.168.204.129	HTTP	457 HTTP/1.1 401 Unauthorized (text/html)
15 17.727410539 192.168.204.129	172.233.221.124	TCP	54 33886 - 80 [ACK] Seq=1180 Ack=1210 Win=31717 Len=0
16 27.772014777 192.168.204.129	172.233.221.124	HTTP	484 GET /basicauth/ HTTP/1.1
17 27.772404878 172.233.221.124	192.168.204.129	TCP	60 80 - 33886 [ACK] Seq=1210 Ack=64240 Len=0
18 27.801792280 172.233.221.124	192.168.204.129	HTTP	457 HTTP/1.1 401 Unauthorized (text/html)
19 27.801848402 192.168.204.129	172.233.221.124	TCP	54 33886 - 80 [ACK] Seq=1610 Ack=1613 Win=31717 Len=0
20 31.229317576 192.168.204.129	172.233.221.124	HTTP	369 GET /favicon.ico HTTP/1.1
21 31.229630899 172.233.221.124	192.168.204.129	TCP	60 80 - 33886 [ACK] Seq=1613 Ack=1925 Win=64240 Len=0
22 31.259939235 172.233.221.124	192.168.204.129	HTTP	383 HTTP/1.1 404 Not Found (text/html)
23 31.259971480 192.168.204.129	172.233.221.124	TCP	54 33886 - 80 [ACK] Seq=1925 Ack=1942 Win=31717 Len=0
24 41.384657610 192.168.204.129	172.233.221.124	TCP	54 [TCP Keep-Alive] 33886 - 80 [ACK] Seq=1924 Ack=1942 Ack=1942 Win=31717 Len=0
25 42.469216335 192.168.204.129	172.233.221.124	TCP	54 [TCP Keep-Alive] 33886 - 80 [ACK] Seq=1924 Ack=1942 Win=31717 Len=0
26 42.469934423 172.233.221.124	192.168.204.129	TCP	60 [TCP Keep-Alive ACK] 80 - 33886 [ACK] Seq=1942 Ack=1925 Win=64240 Len=0
27 52.648549628 192.168.204.129	172.233.221.124	TCP	54 [TCP Keep-Alive] 33886 - 80 [ACK] Seq=1924 Ack=1942 Win=31717 Len=0
28 52.648982747 172.233.221.124	192.168.204.129	TCP	60 [TCP Keep-Alive ACK] 80 - 33886 [ACK] Seq=1942 Ack=1925 Win=64240 Len=0
29 62.889759972 192.168.204.129	172.233.221.124	TCP	54 [TCP Keep-Alive] 33886 - 80 [ACK] Seq=1924 Ack=1942 Win=31717 Len=0
30 62.890752614 172.233.221.124	192.168.204.129	TCP	60 [TCP Keep-Alive ACK] 80 - 33886 [ACK] Seq=1942 Ack=1925 Win=64240 Len=0
31 73.129654237 192.168.204.129	172.233.221.124	TCP	54 [TCP Keep-Alive] 33886 - 80 [ACK] Seq=1924 Ack=1942 Win=31717 Len=0
32 73.130346248 172.233.221.124	192.168.204.129	TCP	60 [TCP Keep-Alive ACK] 80 - 33886 [ACK] Seq=1942 Ack=1925 Win=64240 Len=0
33 83.368740828 192.168.204.129	172.233.221.124	TCP	54 [TCP Keep-Alive] 33886 - 80 [ACK] Seq=1924 Ack=1942 Ack=1942 Win=31717 Len=0
34 83.369294986 172.233.221.124	192.168.204.129	TCP	60 [TCP Keep-Alive ACK] 80 - 33886 [ACK] Seq=1942 Ack=1925 Win=64240 Len=0
35 93.668702132 192.168.204.129	172.233.221.124	TCP	54 [TCP Keep-Alive] 33886 - 80 [ACK] Seq=1942 Ack=1942 Ack=1942 Win=31717 Len=0
36 93.669408514 172.233.221.124	192.168.204.129	TCP	60 [TCP Keep-Alive ACK] 80 - 33886 [ACK] Seq=1942 Ack=1925 Win=64240 Len=0

Investigating with Burpsuite

Observations

Initial observations running everything again. This time there were 2 GET requests. The first is just a normal GET request and the second is a GET request with the username and password “encoded” in base64.

Which leads me to believe that the browser prompts for the username and password, then uses that information to generate a second GET request to authenticate with the server.

Additional Observations

Motivating question #1: what would happen if I entered the password and username wrong?

The browser continues to prompt for your password indefinitely. Or at least until something on either end crashes. I was curious because if the sign in prompt is generated by the browser, how would it know when to stop. Especially since the response from the server is the same, the browser would understandably have the same response: to prompt for a sign-in.

Pretty	Raw	Hex	Render
1 GET /basicauth/ HTTP/1.1 2 Host: cs338.jeffondich.com 3 Accept-Language: en-US 4 Upgrade-Insecure-Requests: 1 5 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/127.0.6533.100 Safari/537.36 6 Accept: 7 text/html,application/xhtml+xml,application/xml ;q=0.9,image/avif,image/webp,image/apng,*/*;q=0 .8,application/signed-exchange;v=b3;q=0.7 8 Accept-Encoding: gzip, deflate, br 9 Connection: keep-alive 10	1 HTTP/1.1 401 Unauthorized 2 Server: nginx/1.18.0 (Ubuntu) 3 Date: Tue, 24 Sep 2024 21:24:29 GMT 4 Content-Type: text/html 5 Content-Length: 590 6 Connection: keep-alive 7 WWW-Authenticate: Basic realm="Protected Area" 8 9 <html> 10 <head><title>401 Authorization Required</title></head> 11 <body> 12 <center><h1>401 Authorization Required</h1></center> 13 <hr><center>nginx/1.18.0 (Ubuntu)</center> 14 </body> 15 </html> 16 <!-- a padding to disable MSIE and Chrome friendly error page --> 17 <!-- a padding to disable MSIE and Chrome friendly error page --> 18 <!-- a padding to disable MSIE and Chrome friendly error page --> 19 <!-- a padding to disable MSIE and Chrome friendly error page --> 20 <!-- a padding to disable MSIE and Chrome friendly error page --> 21 <!-- a padding to disable MSIE and Chrome friendly error page -->	1 GET /basicauth/ HTTP/1.1 2 Host: cs338.jeffondich.com 3 Cache-Control: max-age=0 4 Authorization: Basic Y3MzMzg6GvshG90aGlzaXN0aGV3cm9uZ3Bhc3Nsb3Jk 5 Accept-Language: en-US 6 Upgrade-Insecure-Requests: 1 7 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/127.0.6533.100 Safari/537.36 8 Accept: 9 text/html,application/xhtml+xml,application/xml ;q=0.9,image/avif,image/webp,image/apng,*/*;q=0 .8,application/signed-exchange;v=b3;q=0.7 10 Accept-Encoding: gzip, deflate, br 11 Connection: keep-alive 12 13 <html> 14 <head><title>401 Authorization Required</title></head> 15 <body> 16 <center><h1>401 Authorization Required</h1></center> 17 <hr><center>nginx/1.18.0 (Ubuntu)</center> 18 </body> 19 </html> 20 <!-- a padding to disable MSIE and Chrome friendly error page --> 21 <!-- a padding to disable MSIE and Chrome friendly error page --> 22	

It seems that a stopping procedure would have to be implemented on the server side. For instance only allowing 2 sign-in attempts before sending a 403 error or something. Unless of course you press cancel and the browser recognizes you no longer wish to attempt to sign in.

Motivating question #1: what would happen if I edited the initial GET request to include the Authorization header?

Knowing that the initial GET request and 401 Unauthorized response simply prompts the browser to ask for the username and password to reformat its GET request, I want to see what happens when you just add it in.

I'm assuming it'll bypass the ask for the username and password and just send you straight through. Which is likely what happens when the browser caches that information in the first place.

That is exactly what happened.

The screenshot shows the Network tab of a browser developer tools interface. On the left, the request details are shown in 'Pretty' format:

```
1 GET /basicauth/ HTTP/1.1
2 Host: cs338.jeffondich.com
3 Cache-Control: max-age=0
4 Authorization: Basic Y3MzMzg6cGFzc3dvcmQ=
5 Accept-Language: en-US
6 Upgrade-Insecure-Requests: 1
7 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
  AppleWebKit/537.36 (KHTML, like Gecko) Chrome/127.0.6533.100
  Safari/537.36
8 Accept:
  text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,
  ,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b
  3;q=0.7
9 Accept-Encoding: gzip, deflate, br
10 Connection: keep-alive
11
12
```

On the right, the response details are shown in 'Pretty' format:

```
1 HTTP/1.1 200 OK
2 Server: nginx/1.18.0 (Ubuntu)
3 Date: Tue, 24 Sep 2024 21:43:15 GMT
4 Content-Type: text/html
5 Connection: keep-alive
6 Content-Length: 509
7
8 <html>
9 <head><title>Index of /basicauth/</title></head>
10 <body>
11 <h1>Index of /basicauth/<h1><hr><pre><a href="..">
  ..</a>
12 <a href="amateurs.txt">amateurs.txt</a> 04-Apr-2022
13 <a href="armed-guards.txt">armed-guards.txt</a> 04-Apr-2022 14:10
14 <a href="dancing.txt">dancing.txt</a> 04-Apr-2022
15 </pre><hr></body>
16 </html>
17
```