

# How Basic Auth Works

When we first try to connect to a website protected by HTML Basic Auth, the browser does not know that the page is password protected. After exchanging SYN and ACK packets with the server, the browser does a normal GET request, attempting to view the page. (In this case, I entered in the URL <http://cs338.jeffondich.com/basicauth/>, and the server sent me to /basicauth/, which is what happened with #5 in the screenshot.)

The screenshot displays the Burp Suite interface. The top menu bar includes options like Burp, Project, Intruder, Repeater, View, and Help. Below the menu is a toolbar with various icons. The main window is divided into several panes. The top pane shows a table of HTTP history with columns for #, Host, Method, URL, Params, Status code, Length, MIME type, Extension, Title, Notes, TLS, IP, Cookies, and Time. The table contains several entries, with the 5th entry highlighted: a GET request to http://cs338.jeffondich.com/basicauth/ with a status code of 401. The bottom pane is split into two sections: 'Request' and 'Response'. The 'Request' section shows a GET request to /basicauth/ with various headers. The 'Response' section shows a 401 Unauthorized response with headers including Server: nginx/1.18.0 (Ubuntu), Date: Fri, 26 Sep 2025 21:15:45 GMT, Content-Type: text/html, Content-Length: 590, Connection: keep-alive, and WWW-Authenticate: Basic realm="Protected Area". The 'Inspector' pane on the right shows the response headers in a table format.

#	Host	Method	URL	Params	Status code	Length	MIME type	Extension	Title	Notes	TLS	IP	Cookies	Time
1	https://www.google.com	GET	/warmup.html		200	2408	HTML	html	Warmup Page		✓	142.250.190.36		16:15:32.26..
5	http://cs338.jeffondich.com	GET	/basicauth		301	400	HTML		301 Moved Permanently			172.233.221.124		16:15:45.26..
6	http://cs338.jeffondich.com	GET	/basicauth/		401	805	HTML		401 Authorization Req...			172.233.221.124		16:15:45.26..
7	http://cs338.jeffondich.com	GET	/basicauth/		200	666	HTML		Index of /basicauth/			172.233.221.124		16:15:56.26..
8	http://cs338.jeffondich.com	GET	/favicon.ico		404	728	HTML	ico	404 Not Found			172.233.221.124		16:15:56.26..

**Request**

```
1 GET /basicauth/ HTTP/1.1
2 Host: cs338.jeffondich.com
3 Accept-Language: en-US,en;q=0.9
4 Upgrade-Insecure-Requests: 1
5 User-Agent: Mozilla/5.0 (X11; Linux x86_64)
  AppleWebKit/537.36 (KHTML, like Gecko) Chrome/140.0.0.0
  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-exch
  ange;v=b3;q=0.7
7 Accept-Encoding: gzip, deflate, br
8 Connection: keep-alive
9
10
```

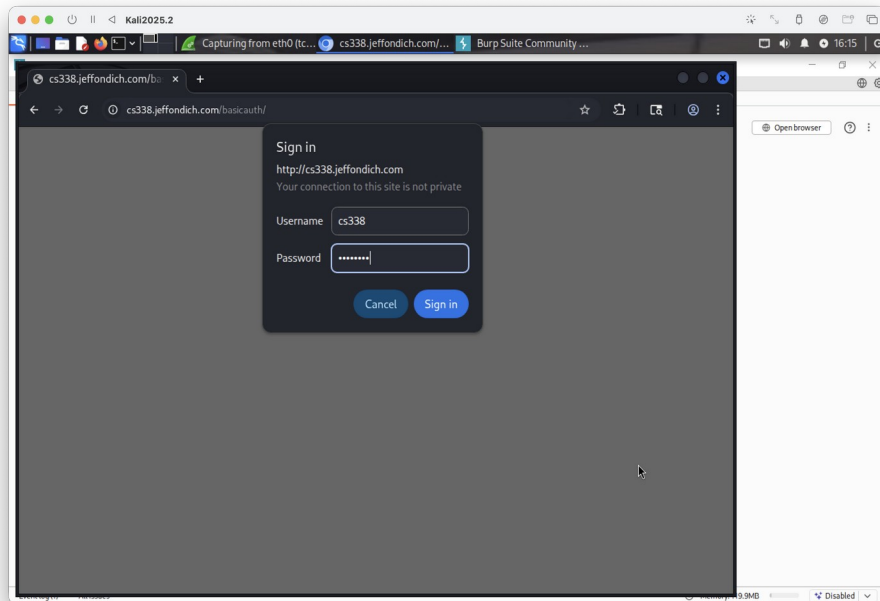
**Response**

```
1 HTTP/1.1 401 Unauthorized
2 Server: nginx/1.18.0 (Ubuntu)
3 Date: Fri, 26 Sep 2025 21:15:45 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>
11 <title>
12 401 Authorization Required
13 </title>
14 </head>
15 <body>
16 <center>
17 401 Authorization Required
18 </center>
19 </body>
20 </html>
21
```

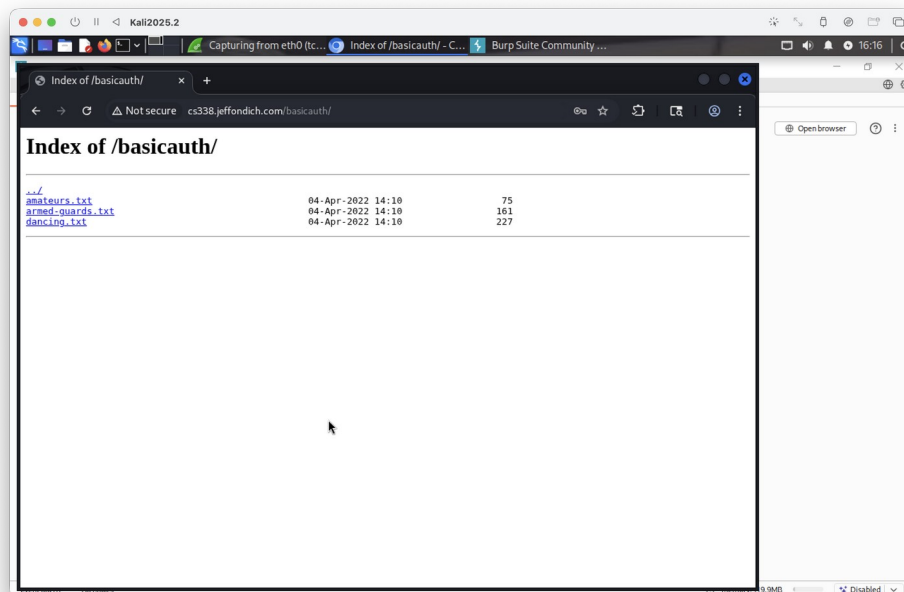
**Inspector**

Name	Value
Server	nginx/1.18.0 (Ubuntu)
Date	Fri, 26 Sep 2025 21:15:4...
Content-Type	text/html
Content-Length	590
Connection	keep-alive
WWW-Authenticate	Basic realm="Protected...

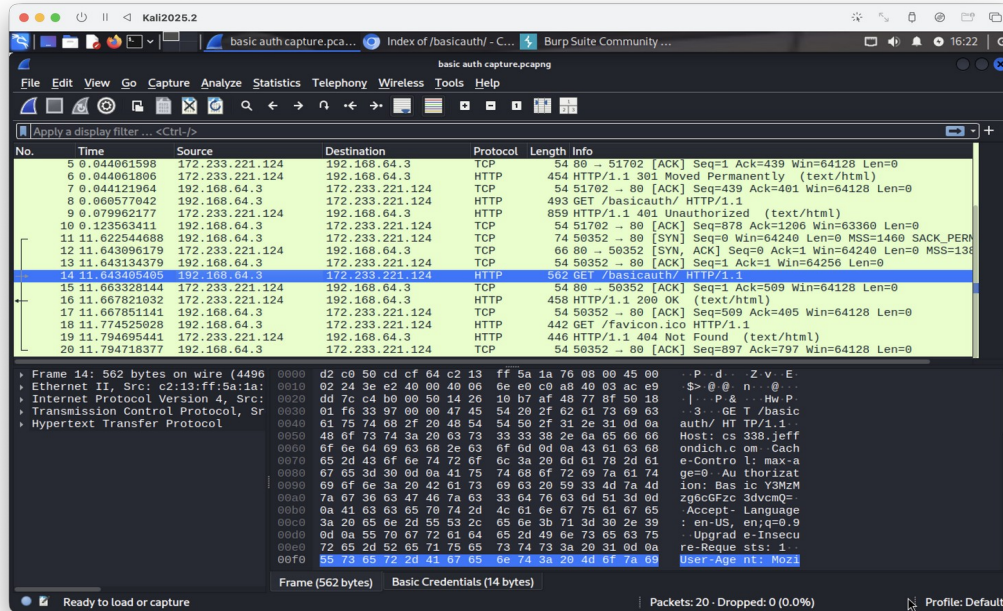
After the browser makes the GET request, the server then responds with the status code "401 Authorization Required." Because of this, the browser knows that this is a password-protected page, so it asks for a username and password.



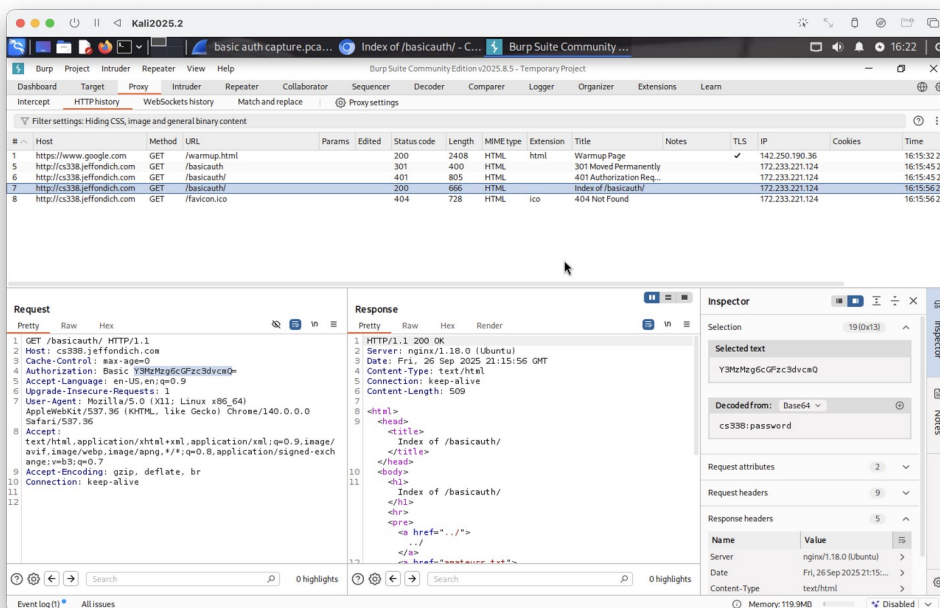
After we have entered our username and password, the page loads and we can view what was once password-protected. But how does this work?



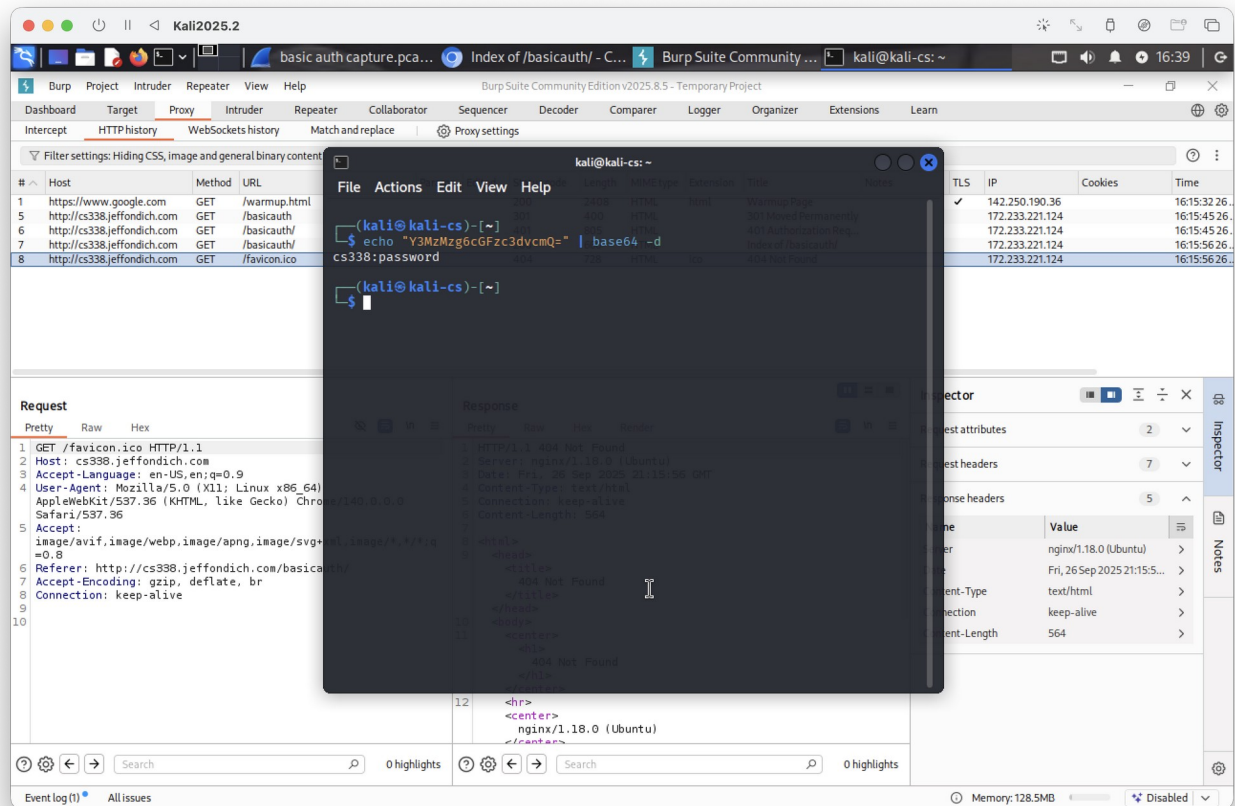
After we enter a username and password, the browser sends the server back the same GET request, but this time with an additional header, "Authorization: Basic Y3MzMzg6cGFzc3dvcmQ=". After the browser sees this, it lets us in and responds with the page. On the surface, this looks pretty secure. Surely, we are sending the browser some kind of an encrypted version of the username and password.



But wait! As it turns out, the username and password aren't encrypted at all! As Burp Suite conveniently tells us, this is just our username:our password encoded into base64!



We can verify this by using the base64 utility in the terminal.



So, HTTP Basic Auth is obviously pretty insecure and a great reason for why we should just be using HTTPS.