Jane Dee Horwitz Parson
September 2025
Computer Security

How does HTTP Basic Authentication Work?

Very simply, actually. This is an HTTP request, *not* an HTTPS request, and so there's an inherent level of vulnerability that stems from the fact every request and response is visible to an outside viewer, unencrypted and utterly standard

```
26 11.735013907  192.168.61.128    172.233.221.124    TCP        74 59598 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=3085431201 TSecr=0 WS=128
27 11.750459398  172.233.221.124    192.168.61.128     TCP        60 80 → 59598 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
28 11.750484888  192.168.61.128    172.233.221.124    TCP        54 59598 → 80 [ACK] Seq=1 Ack=1 Win=64240 Len=0
29 11.750677224  192.168.61.128    172.233.221.124    HTTP      493 GET /basicauth/ HTTP/1.1
30 11.750788755  172.233.221.124    192.168.61.128     TCP        60 80 → 59598 [ACK] Seq=1 Ack=440 Win=64240 Len=0
31 11.769253056  172.233.221.124    192.168.61.128     HTTP      859 HTTP/1.1 401 Unauthorized  (text/html)
32 11.769275352  192.168.61.128    172.233.221.124    TCP        54 59598 → 80 [ACK] Seq=440 Ack=806 Win=63435 Len=0
33 21.328159320  192.168.61.128    172.233.221.124    TCP        74 59604 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=3085440794 TSecr=0 WS=128
34 21.344646023  172.233.221.124    192.168.61.128     TCP        60 80 → 59604 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
35 21.344686008  192.168.61.128    172.233.221.124    TCP        54 59604 → 80 [ACK] Seq=1 Ack=1 Win=64240 Len=0
36 21.345025961  192.168.61.128    172.233.221.124    HTTP      562 GET /basicauth/ HTTP/1.1
37 21.345287961  172.233.221.124    192.168.61.128     TCP        60 80 → 59604 [ACK] Seq=1 Ack=509 Win=64240 Len=0
38 21.362629857  172.233.221.124    192.168.61.128     HTTP      458 HTTP/1.1 200 OK  (text/html)
39 21.362647605  192.168.61.128    172.233.221.124    TCP        54 59604 → 80 [ACK] Seq=509 Ack=405 Win=63836 Len=0
```

That's a lot of text, though, so let's break it down. First comes a standard SYN - ACK call and response as the two servers communicate. Client requests sync, Host requests sync and acknowledges, and Client acknowledges, and two now know that they can successfully send messages to one another. Host (172.223.221.124) is now going to await a message from Client.

26 - 192.168.61.128 -> 172.223.221.124 TCP 59598 -> 80 [SYN]
27 - 172.223.221.124 -> 192.168.61.128 TCP 80 -> 59598 [SYN, ACK]
28 - 192.168.61.128 -> 172.223.221.124 TCP 59598 -> 80 [ACK]

At line 29, the client makes the actual HTTP request, looking for http://cs.jeffondich.com/basicauth. The full summary of the text sent is below the shorthand line, but the basics are "I'm Mozilla Firefox, I speak english, I'm looking for HTTP 1.1, give me this site, please!"

29 - 192.168.61.128 -> 172.223.221.124 HTTP GET /basicauth/ HTTP/1.1

```
GET /basicauth/ HTTP/1.1
Host: cs338.jeffondich.com
Accept-Language: en-US,en;q=0.9
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/139.0.0.0 Safari/537.36
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
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
```

However, after the client makes the request to the server, it receives back a 401 error, Authorization required. This prompts your browser to open the sign-in dropdown from the top of

the screen. Just in case, it also sends the HTTP for an empty page that says “Authorization required.

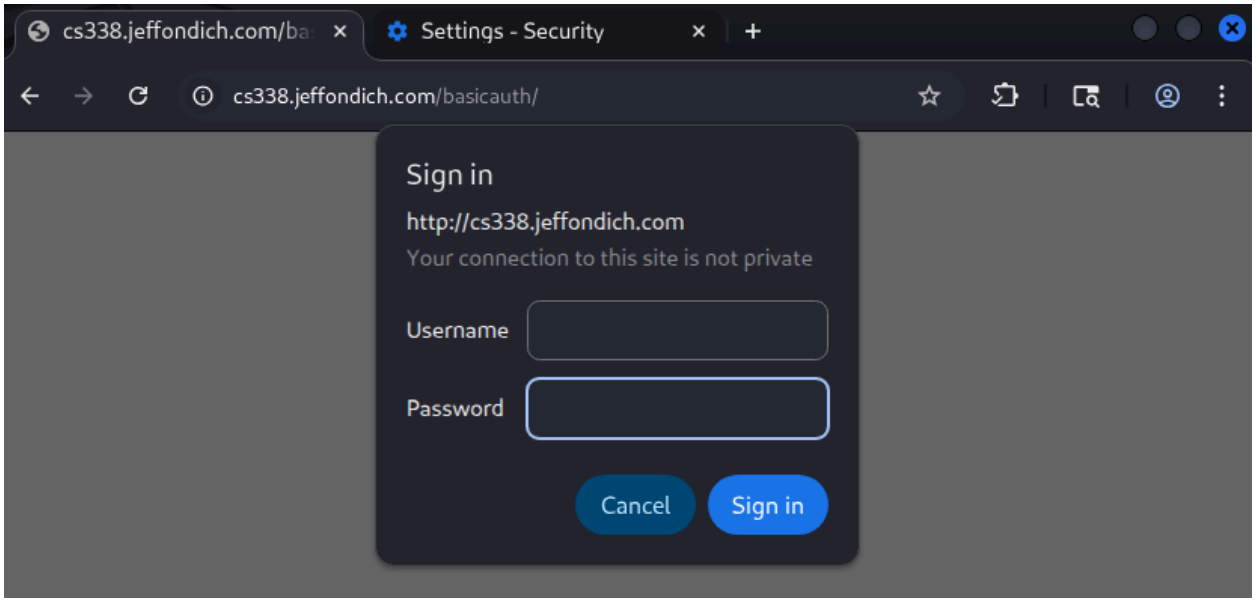
30 - 172.223.221.124 -> 192.168.61.128 80 -> 59598 TCP [ACK]
31 - 172.223.221.124 -> 192.168.61.128 HTTP HTTP/1.1 401 Unauthorized.

```
HTTP/1.1 401 Unauthorized
Server: nginx/1.18.0 (Ubuntu)
Date: Wed, 24 Sep 2025 01:51:25 GMT
Content-Type: text/html
Content-Length: 590
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>
```

The Client then confirms you got the denial, and your browser opens up a password entry field to send a response over. Note that you also get a small “Your connection to this site is not private” because yeah, it isn’t. All this messaging is in plain text HTTP signaling.

32 - 192.168.61.128 -> 172.223.221.124 TCP 59598 -> 80 [ACK]



At the same time, a new synchronization opens, on a different client port. This is the little bubble window that shows up, and all new communication goes through this port

33 - 192.168.61.128 -> 172.223.221.124 TCP 59604 -> 80 [SYN]
34 - 172.223.221.124 -> 192.168.61.128 TCP 80 -> 59804 [SYN, ACK]
35 - 192.168.61.128 -> 172.223.221.124 TCP 59604 -> 80 [ACK]

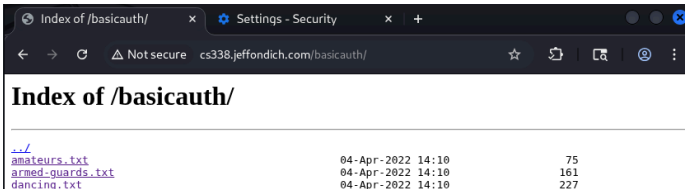
When you finish this, a second, nearly identical request is sent, this time with an additional field, “authorization,” within the HTTP request containing the password and username you sent formatted in base64. This is also unencrypted.

36 - 192.168.61.128 -> 172.223.221.124 HTTP GET /basicauth/ HTTP/1.1

```
GET /basicauth/ HTTP/1.1
Host: cs338.jeffondich.com
Cache-Control: max-age=0
Authorization: Basic Y3MzMzg6cGFzc3dvcmQ=
Accept-Language: en-US,en;q=0.9
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/139.0.0.0 Safari/537.36
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
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
```

The host server then acknowledges, and sends over the page you were asking for, should the password it received match what it’s looking for. But congratulations! You’ve done an authorization.

37 - 172.223.221.124 -> 192.168.61.128 TCP 80 -> 59604 [ACK]
38 - 172.223.221.124 -> 192.168.61.128 HTTP HTTP/1.1 200 OK [With Source HTTP Within]
39 - 192.168.61.128 -> 172.223.221.124 TCP 59604 -> 80 [ACK]




```
HTTP/1.1 200 OK
Server: nginx/1.18.0 (Ubuntu)
Date: Wed, 24 Sep 2025 02:03:15 GMT
Content-Type: text/html
Connection: keep-alive
Content-Length: 509

<html>
  <head>
    <title>
```

This style of authorization is incredibly insecure- not only is the password sent in unencrypted base64, the website you're looking for is as well. At the very least, though, the other server doesn't *tell you* what it's looking for, just whether the password you sent is it- but regardless anyone who's watching your internet communications can read this information and do as they please with it.

Fortunately, HTTPS fixes most of this.