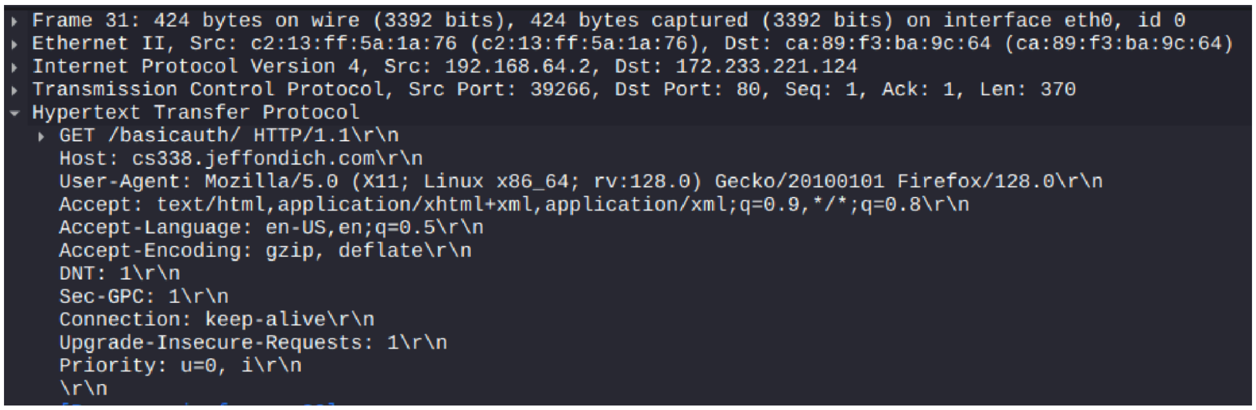
Kezia Sharnoff
CS 338 - Computer Security
Jeff Ondich
September 25, 2025

HTTP’s Basic Authentication

Once upon a time, a user wanted to access a website, cs338.jeffondich.com/basicauth/. Alas, the poor user is unable to view the webpage, because they are not authorized to! At the authentication prompt, they remember the username (“cs338”) and the password (“password”) and then, at last, they can rejoice. The user is now at peace, with many possible files to explore. This story is retold, with a few more details, in the rest of this file.

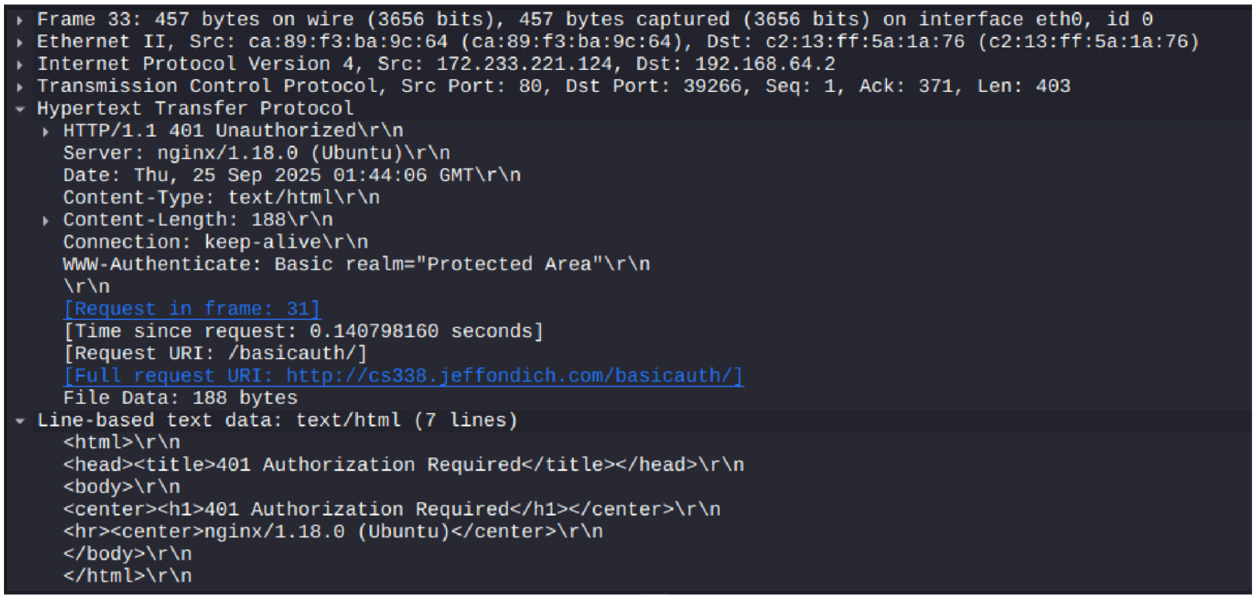
When a user on Firefox first visits http://cs338.jeffondich.com/basicauth/ (from now on called the Webpage) several steps happen. Two TCP handshakes take place on two different source ports, with both ports synchronizing with cs338.jeffondich.com (now referred to as the Server). Then, an HTTP GET request is sent to the Server to request the Webpage.



```
▸ Frame 31: 424 bytes on wire (3392 bits), 424 bytes captured (3392 bits) on interface eth0, id 0
▸ Ethernet II, Src: c2:13:ff:5a:1a:76 (c2:13:ff:5a:1a:76), Dst: ca:89:f3:ba:9c:64 (ca:89:f3:ba:9c:64)
▸ Internet Protocol Version 4, Src: 192.168.64.2, Dst: 172.233.221.124
▸ Transmission Control Protocol, Src Port: 39266, Dst Port: 80, Seq: 1, Ack: 1, Len: 370
▾ Hypertext Transfer Protocol
  ▸ GET /basicauth/ HTTP/1.1\r\n
    Host: cs338.jeffondich.com\r\n
    User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0\r\n
    Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8\r\n
    Accept-Language: en-US,en;q=0.5\r\n
    Accept-Encoding: gzip, deflate\r\n
    DNT: 1\r\n
    Sec-GPC: 1\r\n
    Connection: keep-alive\r\n
    Upgrade-Insecure-Requests: 1\r\n
    Priority: u=0, i\r\n
    \r\n
```

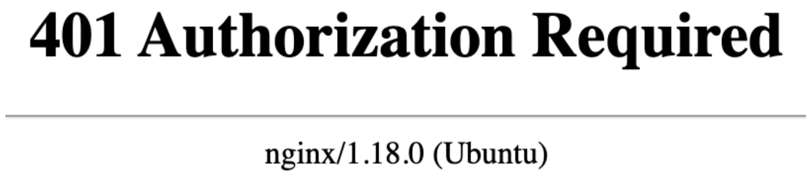
Figure 1: The packet of the initial GET request sent to the Server, captured using Wireshark.

The Server responded to the HTTP GET request to view the Webpage with a status code of 401: Unauthorized. This response included the header “WWW-Authenticate: Basic realm=“Protected Area”. According to the HTTP Authentication framework section 4.1 by the IETF, any server sending a 401: Unauthorized must send the “WWW-Authenticate” header with at least one challenge. From reading the nginx documentation on “Restricting Access with HTTP Basic Authentication", and more specifics about the auth_basic keyword used in the server definition, the string chosen for auth_basic is sent as the challenge realm. nginx servers can have multiple areas protected with HTTP basic authentication, so each has their own auth_basic string to distinguish them. The challenge was: ‘realm=“Protected Area”’ and therefore it is known that the auth_basic in the Server was “Protected Area.”



```
▸ Frame 33: 457 bytes on wire (3656 bits), 457 bytes captured (3656 bits) on interface eth0, id 0
▸ Ethernet II, Src: ca:89:f3:ba:9c:64 (ca:89:f3:ba:9c:64), Dst: c2:13:ff:5a:1a:76 (c2:13:ff:5a:1a:76)
▸ Internet Protocol Version 4, Src: 172.233.221.124, Dst: 192.168.64.2
▸ Transmission Control Protocol, Src Port: 80, Dst Port: 39266, Seq: 1, Ack: 371, Len: 403
▾ Hypertext Transfer Protocol
  ▸ HTTP/1.1 401 Unauthorized\r\n
    Server: nginx/1.18.0 (Ubuntu)\r\n
    Date: Thu, 25 Sep 2025 01:44:06 GMT\r\n
    Content-Type: text/html\r\n
  ▸ Content-Length: 188\r\n
    Connection: keep-alive\r\n
    WWW-Authenticate: Basic realm="Protected Area"\r\n
    \r\n
    [Request in frame: 31]
    [Time since request: 0.140798160 seconds]
    [Request URI: /basicauth/]
    [Full request URI: http://cs338.jeffondich.com/basicauth/]
    File Data: 188 bytes
▾ Line-based text data: text/html (7 lines)
    <html>\r\n
    <head><title>401 Authorization Required</title></head>\r\n
    <body>\r\n
    <center><h1>401 Authorization Required</h1></center>\r\n
    <hr><center>nginx/1.18.0 (Ubuntu)</center>\r\n
    </body>\r\n
    </html>\r\n
```

Figure 2: The packet sent from the Server with the 401: Unauthorized error. Notice the WWW-Authenticate header. HTML for an error page to be displayed is also sent. This packet was captured using Wireshark.



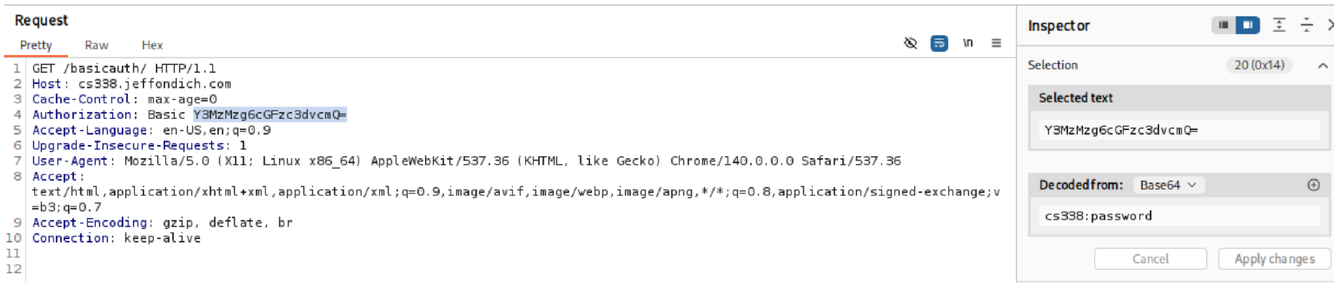
401 Authorization Required

nginx/1.18.0 (Ubuntu)

Figure 3: The rendered error page sent in the packet from Figure 2.

The user was next prompted with a box labeled “Sign in.” There were username and password fields, as well as a place to submit and to cancel. As the user typed in their details, the client computer sent keep alive TCP messages to the Server. If the user clicked cancel, they would have seen the error page that was sent in the original 401: Unauthorized message, Figure 3. After the user submitted a username and password, the browser Base64 encoded them with the format “username:password”, seen in Figure 4. There was no encryption of the password and username, only Base64 encoding which is easily recognizable and easily able to be converted to the plaintext. Given that this is being transported over HTTP, not HTTPS, it would be trivial to intercept and read the username and password.

The username and password were sent off in a new GET request for the Webpage that was the same as the first, except, there was now an Authorization header using the Base64 encoded username and password, as well as the word “basic” to identify the encoding type.



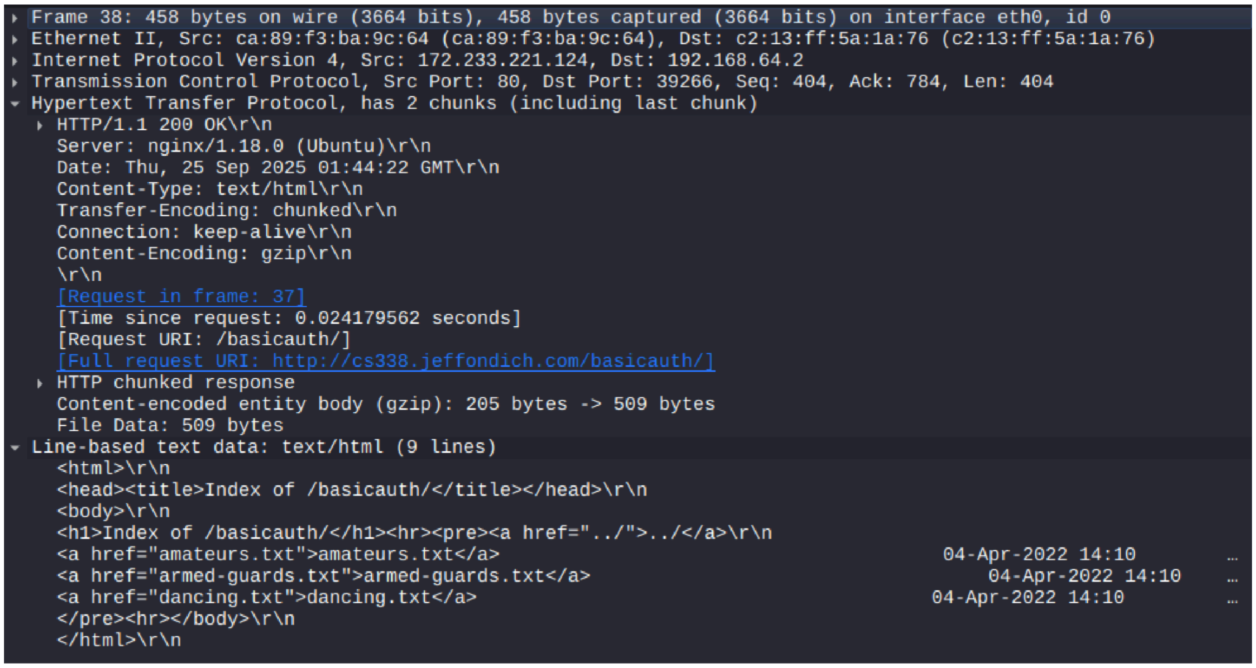
```
GET /basicauth/ HTTP/1.1
Host: cs338.jeffondich.com
Cache-Control: max-age=0
Authorization: Basic Y3MzMzg6cGFzc3dvcmQ=
Accept-Language: en-US,en;q=0.9
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/140.0.0.0 Safari/537.36
Accept:
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
```

```
Y3MzMzg6cGFzc3dvcmQ=
```

```
cs338:password
```

Figure 4: The second HTTP GET request after submitting the username and password. The username and password are easily decoded. This HTTP request was captured by Burp Suite to show the easy decoding.

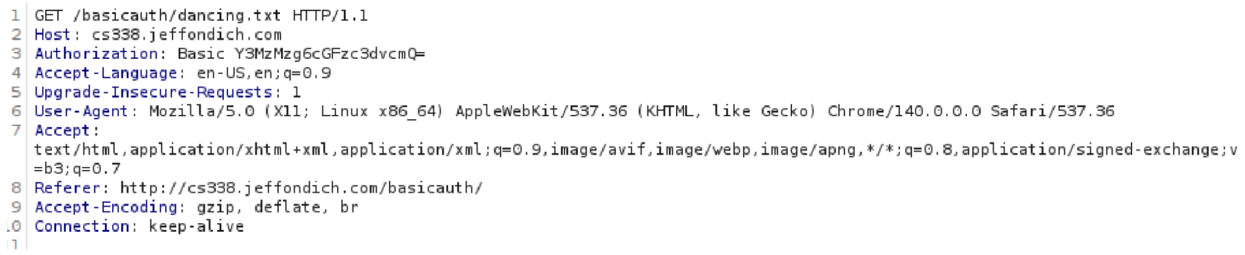
The Server returned to the GET request with a 200 status code, meaning OK. It sent the requested HTML, it was the index page for the directory /basicauth. There was no HTTP header related to Authentication sent in this response.



```
▸ Frame 38: 458 bytes on wire (3664 bits), 458 bytes captured (3664 bits) on interface eth0, id 0
▸ Ethernet II, Src: ca:89:f3:ba:9c:64 (ca:89:f3:ba:9c:64), Dst: c2:13:ff:5a:1a:76 (c2:13:ff:5a:1a:76)
▸ Internet Protocol Version 4, Src: 172.233.221.124, Dst: 192.168.64.2
▸ Transmission Control Protocol, Src Port: 80, Dst Port: 39266, Seq: 404, Ack: 784, Len: 404
▾ Hypertext Transfer Protocol, has 2 chunks (including last chunk)
  ▸ HTTP/1.1 200 OK\r\n
    Server: nginx/1.18.0 (Ubuntu)\r\n
    Date: Thu, 25 Sep 2025 01:44:22 GMT\r\n
    Content-Type: text/html\r\n
    Transfer-Encoding: chunked\r\n
    Connection: keep-alive\r\n
    Content-Encoding: gzip\r\n
    \r\n
    [Request in frame: 37]
    [Time since request: 0.024179562 seconds]
    [Request URI: /basicauth/]
    [Full request URI: http://cs338.jeffondich.com/basicauth/]
  ▸ HTTP chunked response
    Content-encoded entity body (gzip): 205 bytes -> 509 bytes
    File Data: 509 bytes
▾ Line-based text data: text/html (9 lines)
    <html>\r\n
    <head><title>Index of /basicauth/</title></head>\r\n
    <body>\r\n
    <h1>Index of /basicauth/</h1><hr><pre><a href="../">../</a>\r\n
    <a href="amateurs.txt">amateurs.txt</a>                                         04-Apr-2022 14:10    …
    <a href="armed-guards.txt">armed-guards.txt</a>                                     04-Apr-2022 14:10    …
    <a href="dancing.txt">dancing.txt</a>                                          04-Apr-2022 14:10    …
    </pre><hr></body>\r\n
    </html>\r\n
```

Figure 5: The HTTP response from the Server with the requested Webpage and no Authentication related headers. Captured using Wireshark.

Some other experiments tried were putting the wrong password or username and in exploring the other files. If the wrong password or username is submitted, they are encoded in Base64 in the GET request, but the response from the server is the same 401: Unauthorized error that is gotten from initially opening the page, seen in Figure 2. If the user starts their navigation from a page within the protected directory, like Webpage/dancing.txt, they will receive the same steps of being prompted for the authentication and then being authorized to the page. If the user is simply continuing their navigation to other pages after getting into one webpage successfully, the Authorization header is re-sent with every GET request, seen in Figure 6. According to the HTTP Authentication framework section 4.2 by the IETF, after successfully passing a challenge the same credentials are assumed to be valid within that realm.



```
GET /basicauth/dancing.txt HTTP/1.1
Host: cs338.jeffondich.com
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/140.0.0.0 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
Referer: http://cs338.jeffondich.com/basicauth/
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
```

Figure 6: An HTTP GET request for Webpage/dancing.txt which occurred after authenticating the client on the Webpage. The Authorization header is included in this GET request automatically. This HTTP request was captured with Burp Suite.