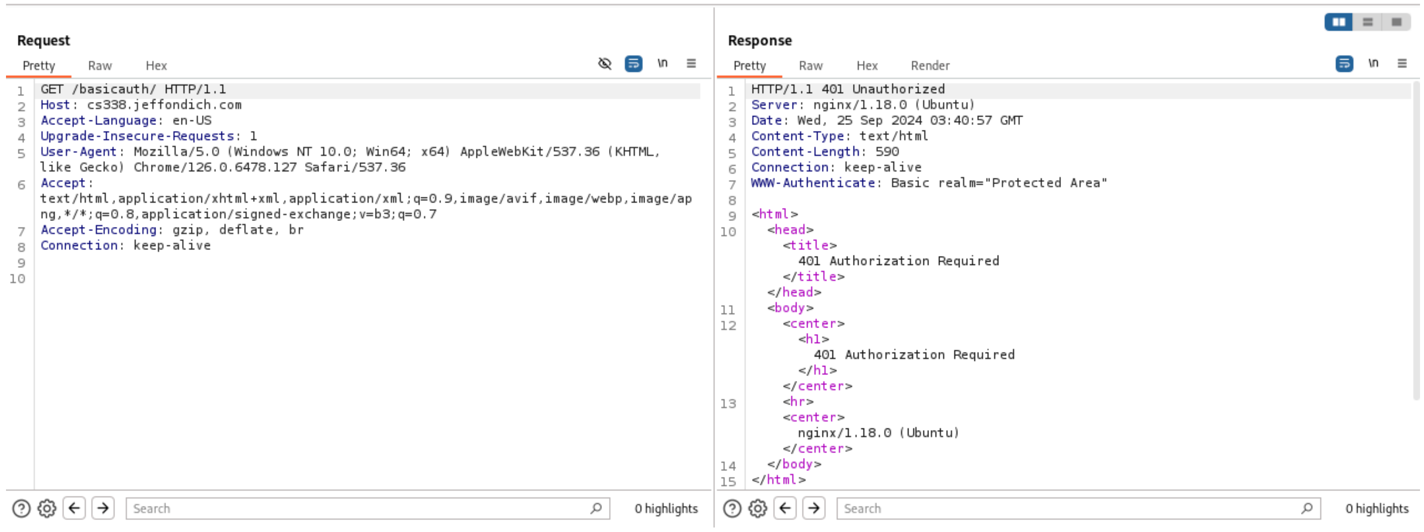
Basic Authentication

Burp Suite Observation:

When attempting to load the webpage the client sends a GET request to the server to request the contents of the http://cs338.jeffondich.com/basicauth/ page. However, this page is secured, requiring users to authenticate themselves. Therefore, it returns a 401 error message to the client, signifying unauthorized access, and the WWW-Authenticate: Basic realm="Protected Area" header. When breaking apart the WWW-Authenticate: Basic realm="Protected Area" header lots of information about the site's security can be retrieved. The WWW-Authenticate response header defines the HTTP authentication methods used to gain access to specific resources. The Basic value refers to HTTP Basic Authentication, where credentials are constructed by combining the username and password with a colon (css338:password) and then encrypted with Base64 (Y3MzMzg6cGFzc3dvcmQ=). Finally, the realm = Protected Area segment of the header value allows the server to inform the client of the credentials necessary to access its content.



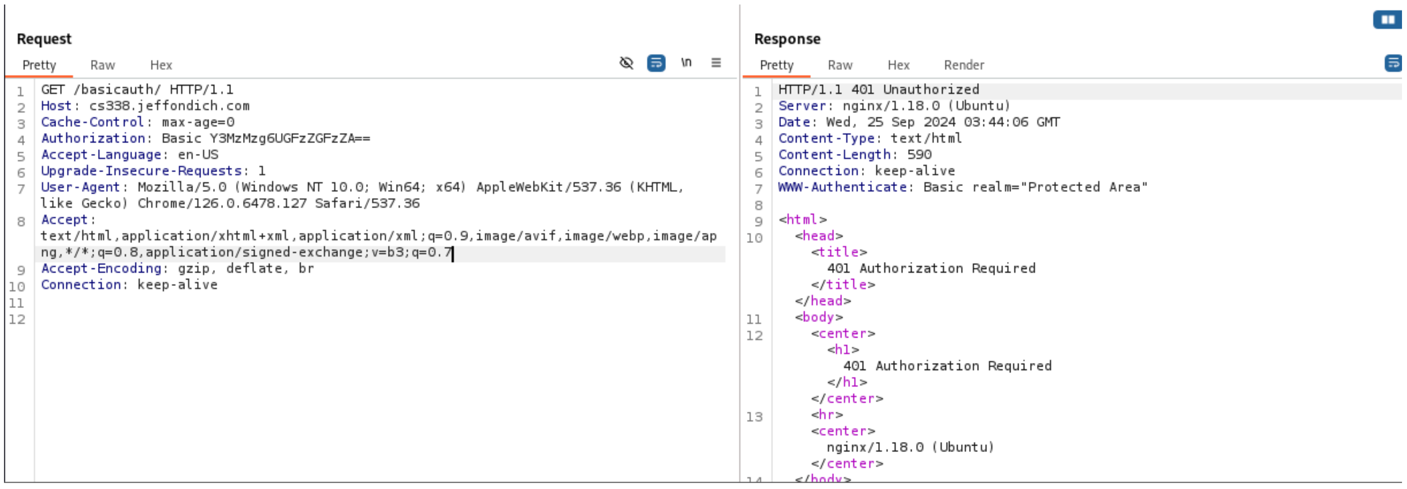
```
GET /basicauth/ HTTP/1.1
Host: cs338.jeffondich.com
Accept-Language: en-US
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/126.0.6478.127 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/ap
ng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
```

```
HTTP/1.1 401 Unauthorized
Server: nginx/1.18.0 (Ubuntu)
Date: Wed, 25 Sep 2024 03:40:57 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>
```

Fig 1. Initial GET request to the webpage

When attempting to authenticate, by entering a username and password, the client sends an encoding authentication message to the server in base64 that contains the username and password the client inputted. This base64 message can be found by observing the Authorization header sent by the client. If the client is unauthorized, the server returns the authentication page and a 401 error message to the user. However, if the client is authorized the server will redirect the client to the page with the secret files and return a 200 error message, signifying a successful request.



Request

Pretty Raw Hex

```
GET /basicauth/ HTTP/1.1
Host: cs338.jeffondich.com
Cache-Control: max-age=0
Authorization: Basic Y3MzMzg6UGFzZGFzZA==
Accept-Language: en-US
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/126.0.6478.127 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/ap
ng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
```

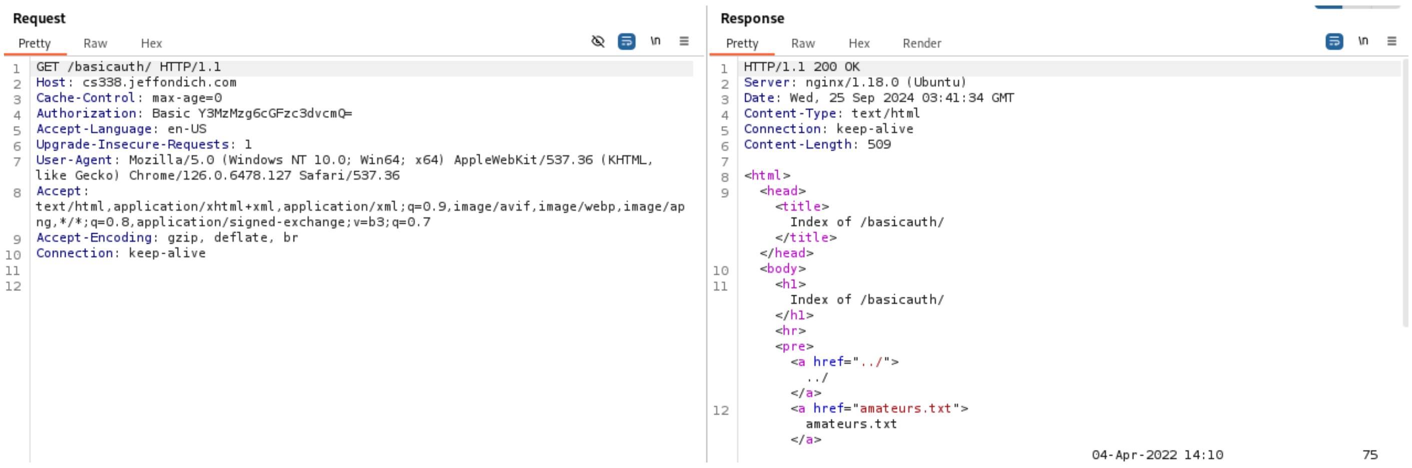
Response

Pretty Raw Hex Render

```
HTTP/1.1 401 Unauthorized
Server: nginx/1.18.0 (Ubuntu)
Date: Wed, 25 Sep 2024 03:44:06 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>
```

Fig 2. Failed authentication request to the server.Y3MzMzg6UGFzZGFzZA== (Base64) > cs338:Pasdads (ASCII)



Request

Pretty Raw Hex

```
GET /basicauth/ HTTP/1.1
Host: cs338.jeffondich.com
Cache-Control: max-age=0
Authorization: Basic Y3MzMzg6cGFzc3dvcmQ=
Accept-Language: en-US
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/126.0.6478.127 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/ap
ng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
```

Response

Pretty Raw Hex Render

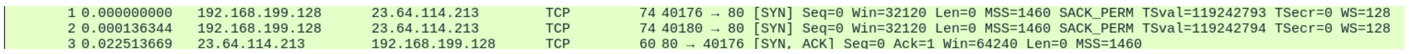
```
HTTP/1.1 200 OK
Server: nginx/1.18.0 (Ubuntu)
Date: Wed, 25 Sep 2024 03:41:34 GMT
Content-Type: text/html
Connection: keep-alive
Content-Length: 509

<html>
  <head>
    <title>
      Index of /basicauth/
    </title>
  </head>
  <body>
    <h1>
      Index of /basicauth/
    </h1>
    <hr>
    <pre>
      <a href="../">
        ../
      </a>
      <a href="amateurs.txt">
        amateurs.txt
      </a>
                                             04-Apr-2022 14:10                  75
```

Fig 3. Successful authentication request to the server. Y3MzMzg6cGFzc3dvcmQ= (Base64) > cs338:password (ASCII)

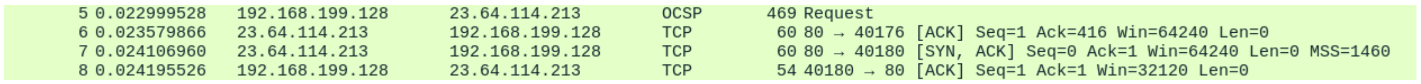
Wireshark Observations:

When attempting to load the page the client sets up a successful TCP Handshake, which establishes communication between the client and the server. After establishing communication, the client sends an OCSP protocol to the server, which is used to check the user's certification status. The server typically responds to this protocol in three ways, good meaning the certificate is valid, revoked meaning the certificate has been revoked and is no longer trustworthy, and unknown meaning the responder can't determine the certificate's status. Additionally, the client sends a GET request to the server, GET /basicauth/ HTTP/1.1, but is denied by the server and receives a 401 error message, meaning that the client is unauthorized to view contents.



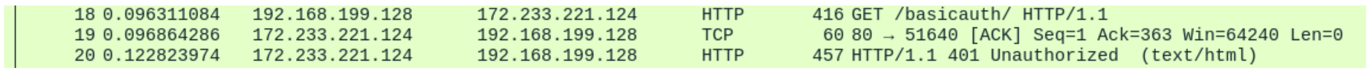
```
1 0.000000000   192.168.199.128      23.64.114.213        TCP       74 40176 → 80 [SYN] Seq=0 Win=32120 Len=0 MSS=1460 SACK_PERM TSval=119242793 TSecr=0 WS=128
2 0.000136344   192.168.199.128      23.64.114.213        TCP       74 40180 → 80 [SYN] Seq=0 Win=32120 Len=0 MSS=1460 SACK_PERM TSval=119242794 TSecr=0 WS=128
3 0.022513669   23.64.114.213        192.168.199.128      TCP       60 80 → 40176 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
```

Fig 4. Successful TCP connection established



```
5 0.022999528   192.168.199.128      23.64.114.213        OCSP      469 Request
6 0.023579866   23.64.114.213        192.168.199.128      TCP        60 80 → 40176 [ACK] Seq=1 Ack=416 Win=64240 Len=0
7 0.024106960   23.64.114.213        192.168.199.128      TCP        60 80 → 40180 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
8 0.024195526   192.168.199.128      23.64.114.213        TCP        54 40180 → 80 [ACK] Seq=1 Ack=1 Win=32120 Len=0
```

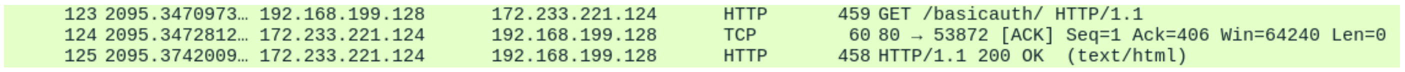
Fig 5. OCSP Protocol being sent by the client



```
18 0.096311084   192.168.199.128      172.233.221.124      HTTP      416 GET /basicauth/ HTTP/1.1
19 0.096864286   172.233.221.124      192.168.199.128      TCP        60 80 → 51640 [ACK] Seq=1 Ack=363 Win=64240 Len=0
20 0.122823974   172.233.221.124      192.168.199.128      HTTP      457 HTTP/1.1 401 Unauthorized  (text/html)
```

Fig 6. Failed authentication request to the server

After successfully loading the page the client is prompted to input credentials to authenticate themselves. Similar to the Burp Suite observations the client encodes the credentials inputted by the user with base64 and sends it to the server for authentication. After the server has decoded and processed the data sent by the client it will either respond with a 401 unauthorized message or a 200 successful message, redirecting the client to the page containing the secret files.



```
123 2095.3470973… 192.168.199.128      172.233.221.124      HTTP      459 GET /basicauth/ HTTP/1.1
124 2095.3472812… 172.233.221.124      192.168.199.128      TCP        60 80 → 53872 [ACK] Seq=1 Ack=406 Win=64240 Len=0
125 2095.3742009… 172.233.221.124      192.168.199.128      HTTP      458 HTTP/1.1 200 OK  (text/html)
```

Fig 7. Successful authentication request to the server.

Works Cited

Thakkar, Megha. “What Is OCSP? OCSP Security Explained - InfoSec Insights.” *Sectigo*, 30 June 2022, https://sectigostore.com/blog/what-is-ocsp-ocsp-security-explained/. Accessed 25 September 2024.

“WWW-Authenticate - HTTP | MDN.” *MDN Web Docs*, 9 September 2024, https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/WWW-Authenticate. Accessed 25 September 2024.