

File: stories/basic-authentication.pdf

Owner: Lysander Miller

Sources: A lot of the information on this page is information I learned from

<https://datatracker.ietf.org/doc/html/rfc7617#section-2>.

Basic Authentication

When we use Wireshark to track the packets passing between a web browser on kali and cs338.jeffondich.com's nginx server – specifically focusing on the packets we see when we try to access cs338.jeffondich.com's secrets folder – we can see the following steps occur:

1. The web browser client sends a [SYN] packet to cs338.jeffondich.com. Here is the packet summary for that packet:

```
1      0.0000000000      192.168.152.128      45.79.89.123      TCP
74      33422 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM
TSval=3935832193 TSecr=0 WS=128
```

In this [SYN] packet, the client provides a port number 33422. This is so that, when cs338.jeffondich.com responds, it knows specifically which piece of software to send its response to.

This packet is the first part of the TCP handshake between port 33422 on the web browser and port 80 on cs338.jeffondich.com. When the TCP handshake is complete, 2-way communication will be set up between port 33422 on the web browser and port 80 on cs338.jeffondich.com.

2. The web browser client sends another [SYN] packet to cs338.jeffondich.com. Here is the packet summary:

```
2      0.039079782 192.168.152.128      45.79.89.123 TCP 74      33432
→ 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM
TSval=3935832232 TSecr=0 WS=128
```

While the previous [SYN] packet came from port 33422, this packet comes from port 33432. Thus, this packet is the first part of the TCP handshake between port 80 on cs338.jeffondich.com and port **33432** on the web browser.

3. The following two packets finish setting up the TCP connection between port 33422 on the web browser and port 80 on cs338.jeffondich.com:

```
3      0.054117328 45.79.89.123 192.168.152.128    TCP    60      80 →
33422 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
```

```
4      0.054189205 192.168.152.128    45.79.89.123  TCP    54      33422
→ 80 [ACK] Seq=1 Ack=1 Win=64240 Len=0
```

2-way communication has now been set up between port 33422 on the web browser and port 80 on cs338.jeffondich.com.

4. The following packet issues a GET request for something called “/basicauth/”:

```
5      0.054580630 192.168.152.128    45.79.89.123  HTTP 408    GET
/basicauth/ HTTP/1.1
```

Although not evident in the packet summary, if we look at the TCP header of this packet, we can see that this request is from port 33422 on the web browser client:

```
▼ Transmission Control Protocol, Src Port: 33422, Dst Port: 80, Seq: 1, Ack: 1, Len: 354
  Source Port: 33422
```

In this HTTP request, port 33422 is requesting that port 80 on cs338.jeffondich.com sends over whatever /basicauth/ is.

5. The following packet acknowledges the packet/message sent by port 33422 on the web browser:

```
6      0.056906702 45.79.89.123 192.168.152.128    TCP    60      80 →
33422 [ACK] Seq=1 Ack=355 Win=64240 Len=0
```

It’s important to note that this acknowledgement has nothing to do with whether or not cs338.jeffondich.com will send /basicauth/ to port 33422 on the web browser.

6. The following two packets finish setting up the TCP connection between port 33432 on the web browser and port 80 on cs338.jeffondich.com:

```
7      0.088742287 45.79.89.123 192.168.152.128    TCP    60      80 →
33432 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
```

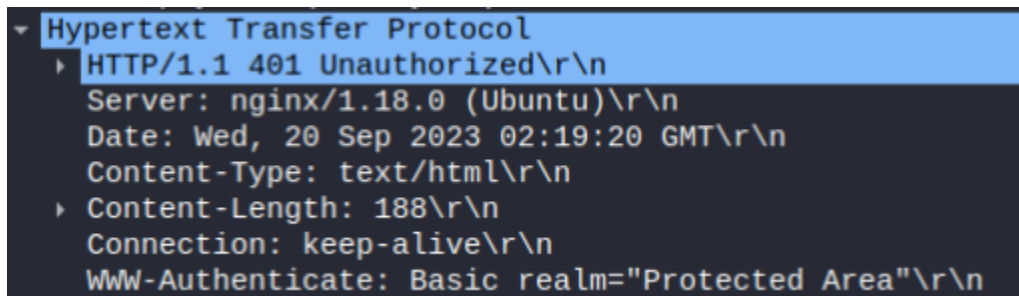
```
8      0.088809736 192.168.152.128    45.79.89.123  TCP    54      33432
→ 80 [ACK] Seq=1 Ack=1 Win=64240 Len=0
```

7. The following packet comes as a response to packet 5 (the packet in which port 33422 made a GET request for /basicauth/):

```
9      0.107042675      45.79.89.123      192.168.152.128      HTTP
457    HTTP/1.1 401 Unauthorized (text/html)
```

In the packet summary above, the status code 401 (which means unauthorized), demonstrates to us that /basicauth/ (what port 33422 on the web browser requested) is a resource within something called a protection space. A protection space is basically a password-protected folder/grouping of one or more resources. Thus, in order for port 80 on cs338.jeffondich.com to send /basicauth/, port 33422 of the web browser must send cs338.jeffondich.com a username and password.

Now, before we move on, let's look a bit deeper into this packet by examining the following screenshot:



```
Hypertext Transfer Protocol
  HTTP/1.1 401 Unauthorized\r\n
  Server: nginx/1.18.0 (Ubuntu)\r\n
  Date: Wed, 20 Sep 2023 02:19:20 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
```

Most of this screenshot shows basic information; for example, it details what type of content /basicauth/ is, the date and time at which /basicauth/ was requested, etc. However, one interesting point of note is the header WWW-Authenticate.

The presence of the WWW-Authenticate header lets us know that cs338.jeffondich.com is using the 'Basic' HTTP authentication scheme (scheme basically means protocol).

After the WWW-Authenticate header, we can see the following text:

```
Basic realm="Protected Area"\r\n
```

Let's break this down a bit.

- Basic is the scheme name. Or, in other words, Basic is the protocol being used for HTTP authentication.

- “Protected Area” is the realm. Or, in other words, the string assigned by the server to identify the protection space.

8. The following packet (sent by port 33422 on the web browser) acknowledges that it has received packet 9 from port 80 on cs338.jeffondich.com:

```
10      0.107107949 192.168.152.128      45.79.89.123  TCP   54      33422
→ 80 [ACK] Seq=355 Ack=404 Win=63837 Len=0
```

9. As indicated by the [FIN] flag, in packet 11, port 33432 on the web browser client closes the TCP connection with port 80 on cs338.jeffondich.com. Then, in packet 13, port 80 on cs338.jeffondich.com closes its connection to port 33432 on the web browser client. Finally, packets 12 and 14 acknowledge packets 11 and 13 respectively.

```
11      6.056676397 192.168.152.128      45.79.89.123  TCP   54      33432
→ 80 [FIN, ACK] Seq=1 Ack=1 Win=64240 Len=0
```

```
12      6.057493494 45.79.89.123 192.168.152.128      TCP   60      80 →
33432 [ACK] Seq=1 Ack=2 Win=64239 Len=0
```

```
13      6.108705842 45.79.89.123 192.168.152.128      TCP   60      80 →
33432 [FIN, PSH, ACK] Seq=1 Ack=2 Win=64239 Len=0
```

```
14      6.108749816 192.168.152.128      45.79.89.123  TCP   54      33432
→ 80 [ACK] Seq=2 Ack=2 Win=64240 Len=0
```

10. In the following packet, port 33422 on the web browser client makes another GET request for /basicauth/:

```
15      9.699747492 192.168.152.128      45.79.89.123  HTTP  451      GET
/basicauth/ HTTP/1.1
```

However, when we look a bit deeper at this packet, we can see that this GET request differs from 33422’s previous GET request:

```

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:109.0) Gecko/20100101 Firefox/115.0\r\n
  Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8\r\n
  Accept-Language: en-US,en;q=0.5\r\n
  Accept-Encoding: gzip, deflate\r\n
  Connection: keep-alive\r\n
  Upgrade-Insecure-Requests: 1\r\n
  Authorization: Basic Y3MzMzg6cGFzc3dvcnQ=\r\n
    Credentials: cs338:password
\r\n
[Full request URI: http://cs338.jeffondich.com/basicauth/]
[HTTP request 2/2]
[Prev request in frame: 5]
[Response in frame: 17]

```

In this request, we see the Authorization header. Let's explain what comes after this header:

- Basic once again refers to the scheme name. Or, in other words, Basic is the protocol being used for HTTP authentication.
- The long string of letters and numbers following the word Basic is the username and password to get to <http://cs338.jeffondich.com/basicauth/> AFTER the username and password have been:
 - a. Concatenated together with a colon between them (as seen in clear text next to the header labeled "Credentials").
 - b. Encoded into an octet sequence (such as UTF-8).
 - c. Encoded again (using Base64) into a sequence of US-ASCII characters.

(for more information about this encryption process, examine <https://datatracker.ietf.org/doc/html/rfc7617#section-2>).

Thus, we can see that, in this packet, port 33422 on the web browser client sends the username and password to cs338.jeffondich.com.

Additionally, notice that username:password is not encrypted/secure. Not only is it sent to cs338.jeffondich.com in Base64 (which can easily be converted back to clear text) but Wireshark even converts username:password into clear text for its user!

11. With the following packet, port 80 on cs338.jeffondich.com acknowledges that it has received packet 15 from port 33422 on the web browser client:

```

16      9.700939032 45.79.89.123 192.168.152.128    TCP    60      80 →
33422 [ACK] Seq=404 Ack=752 Win=64240 Len=0

```

12. Just by looking at the following packet summary, we can tell that this packet is sending /basicauth/ to port 33422 on the web browser client:

```
17      9.756207708  45.79.89.123  192.168.152.128    HTTP 458
HTTP/1.1 200 OK (text/html)
```

After all, within the HTTP protocol, 200 OK is the status code that indicates that the client's request has succeeded. For a GET request, such as the one sent in packet 15, this specifically means that the requested resource has been fetched and is in this package's body. (<https://developer.mozilla.org/en-US/docs/Web/HTTP/Status/200> was my source for the information in this paragraph). In other words, the presence of the phrase "200 OK" means that, in this package, port 80 on cs338.jeffondich.com has sent /basicauth/.

If we look at the contents of this packet a little closer, we can see the text/html content of /basicauth/ sent in this packet:

```
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>
<a href="armed-guards.txt">armed-guards.txt</a>
<a href="dancing.txt">dancing.txt</a>
</pre><hr></body>\r\n
</html>\r\n
```

13. With the following packet, port 33422 on the web browser client acknowledges that it has received packet 17 from port 80 on cs338.jeffondich.com:

```
18      9.756303962  192.168.152.128    45.79.89.123  TCP    54      33422
→ 80 [ACK] Seq=752 Ack=808 Win=63837 Len=0
```

14. The final packets sent between port 33422 on the web browser client and port 80 on cs338.jeffondich.com just keep the TCP connection alive between the two. After all, after a certain amount of time goes by without any packets being sent across the connection, cs338.jeffondich.com will close the connection.

19	19.981832648	192.168.152.128	45.79.89.123	TCP	54 [TCP Keep-Alive] 33422 → 80 [ACK] Seq=751 A
20	21.005775498	192.168.152.128	45.79.89.123	TCP	54 [TCP Keep-Alive] 33422 → 80 [ACK] Seq=751 A
21	21.006906352	45.79.89.123	192.168.152.128	TCP	60 [TCP Keep-Alive ACK] 80 → 33422 [ACK] Seq=8
22	31.245597386	192.168.152.128	45.79.89.123	TCP	54 [TCP Keep-Alive] 33422 → 80 [ACK] Seq=751 A
23	31.245869154	45.79.89.123	192.168.152.128	TCP	60 [TCP Keep-Alive ACK] 80 → 33422 [ACK] Seq=8