

Ruben Boero and Vanessa Heynes

What queries are sent from the browser, and what responses does it receive?

Browser sends a query to the server, asking for the /basicauth/ page:

```
37      0.232054426      172.16.15.128      45.79.89.123      HTTP      417
GET /basicauth/ HTTP/1.1
```

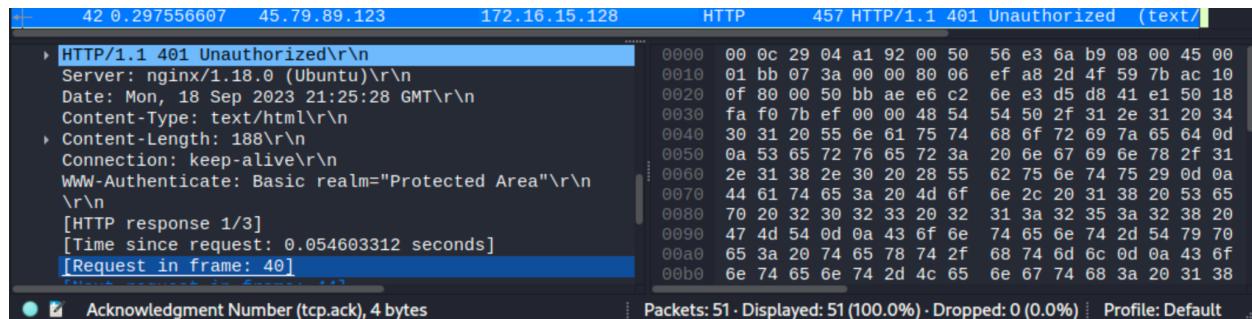
In response, server acknowledges the client's request:

```
41      0.243207591      45.79.89.123      172.16.15.128      TCP      60      80
→ 48046 [ACK] Seq=1 Ack=364 Win=64240 Len=0
```

But refuses to send the page without authorizing the user first:

```
42      0.297556607      45.79.89.123      172.16.15.128      HTTP      457
HTTP/1.1 401 Unauthorized (text/html)
```

The 401 status code is how nginx communicates to the client that they have been denied authorization.



Here, the server is saying that the requested page is protected by Basic HTTP security

After the password is typed by the user, what sequence of queries and responses do you see?

The password is sent by the client to the server:

```
44      7.247785473      172.16.15.128      45.79.89.123      HTTP      460
GET /basicauth/ HTTP/1.1
```

The server acknowledges that the password has been received:

```
45      7.248124978      45.79.89.123      172.16.15.128      TCP      60      80
→ 48046 [ACK] Seq=404 Ack=770 Win=64240 Len=0
```

And that the password is correct:

```
46      7.309787275      45.79.89.123      172.16.15.128      HTTP      458
HTTP/1.1 200 OK (text/html)
```

46 7.309787275	45.79.89.123	172.16.15.128	HTTP	458 HTTP/1.1 200 OK (text/html)
47 7.309816067	172.16.15.128	45.79.89.123	TCP	54 48046 → 80 [ACK] Seq=770 Ack=808
48 7.365140723	172.16.15.128	45.79.89.123	HTTP	377 GET /favicon.ico HTTP/1.1

Frame 46: 458 bytes on wire (3664 bits), 458 bytes captured
 Ethernet II, Src: VMware_e3:6a:b9 (00:50:56:e3:6a:b9), Dst: 172.16.15.128 (45.79.89.123)
 Internet Protocol Version 4, Src: 45.79.89.123, Dst: 172.16.15.128
 Transmission Control Protocol, Src Port: 80, Dst Port: 457989123
 Hypertext Transfer Protocol
 HTTP/1.1 200 OK\r\nServer: nginx/1.18.0 (Ubuntu)\r\nContent-Type: text/html\r\nTransfer-Encoding: chunked\r\nConnection: keep-alive\r\nDate: Mon, 18 Sep 2023 21:25:35 GMT\r\nContent-Length: 509\r\n\r\n

0000	00 0c 29 04 a1 92 00 50 56 e3 6a b9 08 00 45 00
0010	01 bc 07 3e 00 00 80 06 ef a3 2d 4f 59 7b ac 10
0020	0f 80 00 50 bb ae e6 c2 70 76 d5 d8 43 77 50 18
0030	fa f0 5c f1 00 00 48 54 54 50 2f 31 2e 31 20 32
0040	30 30 20 4f 4b 0d 0a 53 65 72 76 65 72 3a 20 6e
0050	67 69 6e 78 2f 31 2e 31 38 2e 30 20 28 55 62 75
0060	6e 74 75 29 0d 0a 44 61 74 65 3a 20 4d 6f 6e 2c
0070	20 31 38 20 53 65 70 20 32 30 32 33 20 32 31 3a
0080	32 35 3a 33 35 20 47 4d 54 0d 0a 43 6f 6e 74 65
0090	6e 74 2d 54 79 70 65 3a 20 74 65 78 74 2f 68 74

Notice that the header contains "OK", rather than "Unauthorized" like in packet 42 shown above.

The server sends the previously requested /basicauth/ HTML to the client:

46 7.309787275	45.79.89.123	172.16.15.128	HTTP	458 HTTP/1.1 200 OK (text/html)
47 7.309816067	172.16.15.128	45.79.89.123	TCP	54 48046 → 80 [ACK] Seq=770 Ack=808
48 7.365140723	172.16.15.128	45.79.89.123	HTTP	377 GET /favicon.ico HTTP/1.1
49 7.365501020	45.79.89.123	172.16.15.128	TCP	60 80 → 48046 [ACK] Seq=808 Ack=1093
50 7.419076691	45.79.89.123	172.16.15.128	HTTP	383 HTTP/1.1 404 Not Found (text/html)
51 7.419104649	172.16.15.128	45.79.89.123	TCP	54 48046 → 80 [ACK] Seq=1093 Ack=113

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>..\r\namateurs.txt\r\narmed-guards.txt\r\ndancing.txt\r\n</pre><hr></body>\r\n</html>\r\n

0000	00 0c 29 04 a1 92 00 50 56 e3 6a b9 08 00 45 00
0010	01 bc 07 3e 00 00 80 06 ef a3 2d 4f 59 7b ac 10
0020	0f 80 00 50 bb ae e6 c2 70 76 d5 d8 43 77 50 18
0030	fa f0 5c f1 00 00 48 54 54 50 2f 31 2e 31 20 32
0040	30 30 20 4f 4b 0d 0a 53 65 72 76 65 72 3a 20 6e
0050	67 69 6e 78 2f 31 2e 31 38 2e 30 20 28 55 62 75
0060	6e 74 75 29 0d 0a 44 61 74 65 3a 20 4d 6f 6e 2c
0070	20 31 38 20 53 65 70 20 32 30 32 33 20 32 31 3a
0080	32 35 3a 33 35 20 47 4d 54 0d 0a 43 6f 6e 74 65
0090	6e 74 2d 54 79 70 65 3a 20 74 65 78 74 2f 68 74

Finally, the server then sends the favicon of the page.

Is the password sent by the browser to the server, or does the browser somehow do the password checking itself?

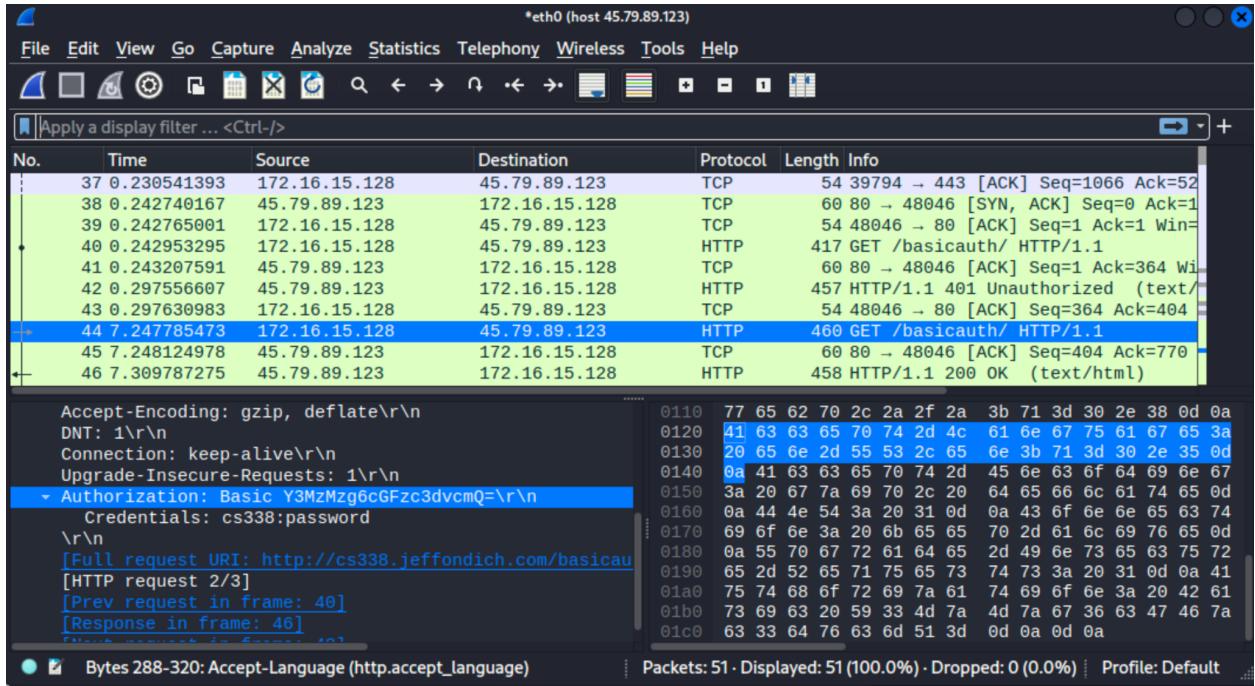
The server—running nginx—is doing the checking:

46 7.309787275	45.79.89.123	172.16.15.128	HTTP	458 HTTP/1.1 200 OK (text/html)
47 7.309816067	172.16.15.128	45.79.89.123	TCP	54 48046 → 80 [ACK] Seq=770 Ack=808
48 7.365140723	172.16.15.128	45.79.89.123	HTTP	377 GET /favicon.ico HTTP/1.1
49 7.365501020	45.79.89.123	172.16.15.128	TCP	60 80 → 48046 [ACK] Seq=808 Ack=1093
50 7.419076691	45.79.89.123	172.16.15.128	HTTP	383 HTTP/1.1 404 Not Found (text/html)
51 7.419104649	172.16.15.128	45.79.89.123	TCP	54 48046 → 80 [ACK] Seq=1093 Ack=113

Frame 46: 458 bytes on wire (3664 bits), 458 bytes captured
 Ethernet II, Src: VMware_e3:6a:b9 (00:50:56:e3:6a:b9), Dst: 172.16.15.128 (45.79.89.123)
 Internet Protocol Version 4, Src: 45.79.89.123, Dst: 172.16.15.128
 Transmission Control Protocol, Src Port: 80, Dst Port: 457989123
 Hypertext Transfer Protocol
 HTTP/1.1 200 OK\r\nServer: nginx/1.18.0 (Ubuntu)\r\nContent-Type: text/html\r\nTransfer-Encoding: chunked\r\nConnection: keep-alive\r\nDate: Mon, 18 Sep 2023 21:25:35 GMT\r\nContent-Length: 509\r\n\r\n

0000	3c 68 74 6d 6c 3e 0d 0a 3c 68 65 61 64 3e 3c 74
0010	69 74 6c 65 3e 49 6e 64 65 78 20 6f 66 20 2f 62
0020	61 73 69 63 61 75 74 68 2f 3c 2f 74 69 74 6c 65
0030	3e 3c 2f 68 65 61 64 3e 0d 0a 3c 62 6f 64 79 3e
0040	0d 0a 3c 68 31 3e 49 6e 64 65 78 20 6f 66 20 2f
0050	62 61 73 69 63 61 75 74 68 2f 3c 2f 68 31 3e 3c
0060	68 72 3e 3c 70 72 65 3e 3c 61 20 68 72 65 6d 3d
0070	22 2e 2e 2f 22 3e 2e 2e 2f 3c 2f 61 3e 0d 0a 3c
0080	61 20 68 72 65 66 3d 22 61 6d 61 74 65 75 72 73
0090	2e 74 78 74 22 3e 61 6d 61 74 65 75 72 73 2e 74

We came to this conclusion because the OK message comes from the server, not from the client.



Screenshot shows the client sending the password to the server.

The password is sent by the client to the server in this packet:

```
44      7.247785473      172.16.15.128      45.79.89.123      HTTP      460
GET /basicauth/ HTTP/1.1
```

Additionally, we found some [nginx documentation/examples](#) that makes it seem like the checking is happening within the server.

If the former, is the password sent in clear text or is it encrypted or something else?

The password is sent in clear text encoded with base64. The username and password are separated by a colon character:

```
Authorization: Basic Y3MzMzg6cGFzc3dvcmQ=\r\n
Credentials: cs338:password
```

If it's encrypted, where did the encryption key come from?

It's not encrypted.

How does what you observe via Wireshark connect to the relevant sections of the HTTP and HTTP Basic Authentication specification documents?

What we observed was directly connected to the section of the HTTP information under the 2.0 header.

- The username and password are sent unencrypted in plain text using base64 encoding.
- The username and password are sent as 1 string where the username and password are separated by a comma character.
- The HTTP headers were formatted in the same way as the documents describe.

Generally, the HTTP messages sent back and forth between the server and the client match the HTTP documentation.

How is the password verified? By whom?

The password is verified by nginx from the server side.

Sources:

- <https://docs.nginx.com/nginx/admin-guide/security-controls/configuring-subrequest-authentication/>
- <https://docs.nginx.com/nginx/admin-guide/security-controls/configuring-http-basic-authentication/>

Where is the authorized header?

The authorization header is found in packet 42:

HTTP/1.1 401 Unauthorized (text/)
Server: nginx/1.18.0 (Ubuntu)\r\nDate: Mon, 18 Sep 2023 21:25:28 GMT\r\nContent-Type: text/html\r\nContent-Length: 188\r\nConnection: keep-alive\r\nWWW-Authenticate: Basic realm="Protected Area"\r\n\r\n[HTTP response 1/3]
[Time since request: 0.054603312 seconds]
[Request in frame: 40]

Packets: 51 · Displayed: 51 (100.0%) · Dropped: 0 (0.0%) · Profile: Default

Here we can see that the server sends an HTTP response to the client. In this packet the authorized header can be found. The information in this packet tells the client that they are not yet an authorized user (status 401), as well as providing the client with information about the type of authentication that is being used (Basic)

Source:

- <https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Authorization>