

John Woods

## A Basic Auth Story

The process begins with the TCP handshake: my machine sending SYN packets to the server, and the server responding with SYN ACK, with the handshake being completed by my machine sending an ACK. Notably, my machine appears to connect to two separate ports, port 80 and port 443.

1	0.000000000	172.16.222.128	45.79.89.123	TCP	74 58916 → 80 [SYN] Seq=0 Win=64240 Len=0 MS
2	0.007393480	172.16.222.128	45.79.89.123	TCP	74 41794 → 443 [SYN] Seq=0 Win=64240 Len=0 M
3	0.045217187	45.79.89.123	172.16.222.128	TCP	60 80 → 58916 [SYN, ACK] Seq=0 Ack=1 Win=642
4	0.045270952	172.16.222.128	45.79.89.123	TCP	54 58916 → 80 [ACK] Seq=1 Ack=1 Win=64240 Le
5	0.052372829	45.79.89.123	172.16.222.128	TCP	60 443 → 41794 [SYN, ACK] Seq=0 Ack=1 Win=64
6	0.052427157	172.16.222.128	45.79.89.123	TCP	54 41794 → 443 [ACK] Seq=1 Ack=1 Win=64240 L

Port 80 is used for general plaintext data transmission, whereas port 443 allows secure network data transmission (<https://www.ssl2buy.com/wiki/port-80-http-vs-port-443-https#:~:text=The%20main%20difference%20between%20Port,a%20non%2DHTTPS%20web%20page>)

Next, my machine sends a “Client Hello,” which contains the address cs338.jeffondich.com. The hello message includes the TLS version I’m running, as well as some other information (<https://www.cloudflare.com/learning/ssl/what-happens-in-a-tls-handshake/>)

This Hello is acknowledged by the server, and it responds with a Server Hello, which contains the phrase “let’s encrypt!” a few times, alongside the address accioapps.com, which (I believe) is the website for an app of Jeff’s given that the copyright points to Ultralingua Inc.

The server hello also establishes the certificate of the server. Additionally, the client and server exchange client and server keys and certificates, as seen below:

11	0.101612457	45.79.89.123	172.16.222.128	TLSv1.2	534 Certificate, Server Key Exchange, Server Hello Done
12	0.101619317	172.16.222.128	45.79.89.123	TCP	54 41794 → 443 [ACK] Seq=518 Ack=4577 Win=62780 Len=0
13	0.108224258	172.16.222.128	45.79.89.123	TLSv1.2	212 Client Key Exchange, Change Cipher Spec, Encrypted Handshake

The message from the server contains both its public key, as well as its signature.

0070	10 e9 6f 82 c8 11 42 0d	fb e9 ec e3 86 00 de 9d	...	0 B
0080	10 e3 38 fa a4 7d b1 d8	e8 49 82 84 06 9b 2b e8	...	8 } . . . I . . . +
0090	6b 4f 01 0c 38 77 2e f9	dd e7 39 16 03 03 01 6d	...	kO . 8w. . . . 9 . . . m
00a0	0c 00 01 09 03 00 18 61	04 f9 df cd 7d 17 27 4a	...	. . . i . . . a . . . } . . . J
00b0	e6 c1 e8 50 0c 6a 16 47	9a 18 8e 6a 29 08 ba 10	...	. . . P . . . j . . . } . . .
00c0	45 52 a9 96 cf 25 a2 35	31 0f 6c 56 53 8b bd 63	...	ER . . . % . 5 1 . lVS . . c
00d0	51 09 39 15 17 e5 10 4f	81 e8 02 af e8 05 de 98	...	Q19 . . . . 0 . . . . .
00e0	d7 f3 ce 25 bb c9 d8 77	f0 00 c0 48 0c be d0 56	...	. . . % . . . w . . . H . . V
00f0	f1 35 a6 6b 35 a0 f5 a0	23 ce 15 97 0f a6 da 0e	...	. . . 5 . k5 . . . # . . . 0 . . .
0100	8c e4 8c d0 db ea 08 40	90 08 04 01 00 3a f0 0f	...	. . . . . @ . . . . . : . . .
0110	06 33 7e 01 1f 51 77 3b	f4 f9 1f c9 6f 2f bd 4d	...	. . . 3 . . . Qw; . . . o / M
0120	31 56 df 8e 7c 8c 76 d1	fd 31 d5 f4 b6 5a 3c 3f	...	1V . . .   . . . v . . . 1 . . . Z < ?
0130	f0 2f 13 cb 11 75 42 bb	9a 8c 0c 22 5e 5d 13 56	...	. / . . . uB . . . " ^ ] . V
0140	01 7a d5 1f 4c 81 82 0b	a9 81 b8 34 88 7e 0b 4e	...	. z . . . L . . . . . 4 . . ~ N
0150	db d9 c2 70 fc 1b 67 87	06 8f 7d 38 68 49 80 1d	...	. . . p . . . g . . . } 8hI . .
0160	9b ad f4 4e b5 62 b2 ef	ec 23 b9 35 07 c0 39 bc	...	. . . N . . . b . . . # . 5 . 9

Frame (534 bytes) Reassembled TCP (4086 bytes)

Bytes 168-264: Pubkey (tls.handshake.server\_point) Packets: 31 - Displayed: 31 (100.0%) Profile: Default

```

0120 31 56 df 8e 7c 8c 76 d1 fd 31 d5 f4 b6 5a 3c 3f 1V...|v...1...Z<?
0130 f0 2f 13 cb 11 75 42 bb 9a 8c 0c 22 5e 5d 13 56 ./...uB...^...V
0140 01 7a d5 1f 4c 81 82 0b a9 81 b8 34 88 7e 0b 4e .z...L...4...N
0150 db d9 c2 70 fc 1b 67 87 06 8f 7d 38 68 49 80 1d ...p...g...}8hI...
0160 9b ad f4 4e b5 62 b2 ef ec 23 b9 35 97 c0 39 bc ...N...b...#...5...9...
0170 59 ce af f9 b9 9a 73 32 d4 dc 69 b9 b9 67 73 cd Y...s2...i...gs...
0180 ee 1a 40 6c 0d 2d 1e c9 66 2b 84 35 80 86 88 d0 ...@1...f+;5...
0190 fc b2 09 db 09 6f 69 6a d5 64 e3 f1 7d 6b ea 71 ...oij...d...}k.q
01a0 46 f8 8b 76 3a ac 7b 80 42 da 55 50 6c 7b fb 38 F...v...(-B-UP1{-8
01b0 60 2d eb ad ef 26 25 06 71 b8 ad 0d a3 0b 7a 4a ...8%...q...z3
01c0 36 35 cc c6 b1 a7 df 2b 51 49 d0 3f e4 02 8d ef 65-F...+Q@...?...
01d0 29 09 f3 dd 1e c3 4e 60 a0 0b e2 fb a8 e7 b9 a0 )...N...
01e0 76 85 28 9f 03 d5 10 9c f0 5b d6 14 b9 cb 7b 2e V...{...[...{...
01f0 c9 da 71 7f d1 c5 76 fb e4 7c a8 af cb 92 27 1b ...q...v...|...{...
0200 d8 93 94 61 53 1c 06 54 c5 ad 17 e4 b5 16 03 03 ...aS...T...
0210 00 04 0e 00 00 00

```

Frame (534 bytes) Reassembled TCP (4086 bytes)

Bytes 269-524: Signature (tls.handshake.sig) Packets: 31 · Displayed: 31 (100.0%) Profile: Default

Meanwhile, the client key message only contains its pubkey, as well as its handshake protocol, and the message is marked as a Change Cypher Spec message, which is letting the server know that the keys have been set up and that it's switching to encrypted messaging.

```

04 d6 c0 a3 5a 7a a9 31 37 2b 6f ac 20 7a dd bf ...Zz.1 7+o. z...
03 dc ef 7d 05 2e 9b 4b c2 30 32 41 a3 ba 7b 31 ...}.K.02A...{1
55 f2 c1 59 1f 05 20 12 7a f2 74 62 c2 79 2c c8 U...Y... z.tb.y,
1c 52 35 9a 08 14 ec ed bd df c8 58 d5 8c 21 27 R5...X...!
cb 16 da d7 6c 68 95 5d f7 96 d0 18 ec 42 95 a6 ...lh.]...B...
23 a8 d2 30 d2 cc 40 9e 0f 9a e5 f3 6a 1c be 09 #...@...j...
18 14 03 03 00 01 01 16 03 03 00 28 00 00 00 00 ...(.
00 00 00 00 75 7a 47 95 d9 b3 92 9f 6a 92 9e 66 ...uzG...j..f
6c aa ab cc b8 64 c7 13 58 18 91 3c 10 96 79 7b l...d...X...<.y{
0c 3a 03 85

```

Bytes 64-160: Pubkey (tls.handshake.client\_point) Packets: 31 · Displayed: 31 (100.0%)

```

04 d6 c0 a3 5a 7a a9 31 37 2b 6f ac 20 7a dd bf ...Zz.1 7+o. z...
03 dc ef 7d 05 2e 9b 4b c2 30 32 41 a3 ba 7b 31 ...}.K.02A...{1
55 f2 c1 59 1f 05 20 12 7a f2 74 62 c2 79 2c c8 U...Y... z.tb.y,
1c 52 35 9a 08 14 ec ed bd df c8 58 d5 8c 21 27 R5...X...!
cb 16 da d7 6c 68 95 5d f7 96 d0 18 ec 42 95 a6 ...lh.]...B...
23 a8 d2 30 d2 cc 40 9e 0f 9a e5 f3 6a 1c be 09 #...@...j...
18 14 03 03 00 01 01 16 03 03 00 28 00 00 00 00 ...(.
00 00 00 00 75 7a 47 95 d9 b3 92 9f 6a 92 9e 66 ...uzG...j..f
6c aa ab cc b8 64 c7 13 58 18 91 3c 10 96 79 7b l...d...X...<.y{
0c 3a 03 85

```

Bytes 172-211: Handshake Protocol (tls.handshake) Packets: 31 · Displayed: 31 (100.0%)

Notably, these messages have occurred over port 443, since security is necessary until this point. Additionally, the server sends a Change Cypher Spec message, of its own. This is followed by the RST packet from the client, letting the server know that it's done using port 443, and since the encryption keys have been set up, they can both switch to using port 80. This message also contains data that lets both parties ensure the handshake hasn't been tampered with.

(<https://www.thesslstore.com/blog/explaining-ssl-handshake/#:~:text=The%20%E2%80%9CCchange%20Cipher%20Spec%E2%80%9D%20message,complete%20on%20the%20client%20side>)

20 0.157036301	45.79.89.123	172.16.222.128	TLSv1.2	105 Change Cipher Spec, Encrypted Handshake Message
21 0.157060169	172.16.222.128	45.79.89.123	TCP	54 41796 → 443 [RST] Seq=708 Win=0 Len=0

From there on, all messages seem to be encrypted, and only use port 80.

```
21 0.157060169 172.16.222.128 45.79.89.123 TCP 54 41796 → 443 [RST] Seq=708 Win=0 Len=0
22 0.158021041 45.79.89.123 172.16.222.128 TCP 60 80 → 58924 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
23 0.158057114 172.16.222.128 45.79.89.123 TCP 54 58924 → 80 [ACK] Seq=1 Ack=1 Win=64240 Len=0
24 5.058202067 172.16.222.128 45.79.89.123 TCP 54 58920 → 80 [FIN, ACK] Seq=1 Ack=1 Win=64240 Len=0
25 5.058483310 45.79.89.123 172.16.222.128 TCP 60 80 → 58920 [ACK] Seq=1 Ack=2 Win=64239 Len=0
26 5.103099675 45.79.89.123 172.16.222.128 TCP 60 80 → 58920 [FIN, PSH, ACK] Seq=1 Ack=2 Win=64239 Len=0
27 5.103118925 172.16.222.128 45.79.89.123 TCP 54 58920 → 80 [ACK] Seq=2 Ack=2 Win=64240 Len=0
28 6.058513834 172.16.222.128 45.79.89.123 TCP 54 58924 → 80 [FIN, ACK] Seq=1 Ack=1 Win=64240 Len=0
29 6.058878331 45.79.89.123 172.16.222.128 TCP 60 80 → 58924 [ACK] Seq=1 Ack=2 Win=64239 Len=0
30 6.103679581 45.79.89.123 172.16.222.128 TCP 60 80 → 58924 [FIN, PSH, ACK] Seq=1 Ack=2 Win=64239 Len=0
31 6.103707502 172.16.222.128 45.79.89.123 TCP 54 58924 → 80 [ACK] Seq=2 Ack=2 Win=64240 Len=0

Internet Protocol Version 4, Src: 172.16.222.128, Dst: 45.79.89.123
00 50 56 f2 f3 24 00 0c 29 38 74 c0 08 00 45 00 .PV..S... )8t...E
00 28 00 00 40 00 40 06 29 75 ac 10 de 80 2d 4f .(. @. @.)u....-0
59 7b e6 2c 00 50 ca 86 16 4c 17 1d c1 d9 50 10 Y{., P.. L...P
fa f0 03 42 00 00 ...B...
```

Finally, my browser requests the basicauth folder, and is denied with a 401 unauthorized error, and we begin the authorization process.

```
24 0.161333164 172.16.64.129 45.79.89.123 HTTP 403 GET /basicauth/ HTTP/1.1
25 0.161600876 45.79.89.123 172.16.64.129 TCP 60 80 → 46306 [ACK] Seq=1 Ack=350 Win=64240 Len=0
26 0.207128663 45.79.89.123 172.16.64.129 HTTP 457 HTTP/1.1 401 Unauthorized (text/html)
```

Note that the auth error also contains the WWW-Authenticate header which informs my browser about what authorization schemes will be accepted (<https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Authorization>).

```
65 65 70 2d 61 6c 69 76 65 0d 0a 57 57 57 2d 41 eep-aliv e .WWW-A
75 74 68 65 6e 74 69 63 61 74 65 3a 20 42 61 73 uthentic ate: Bas
69 63 20 72 65 61 6c 6d 3d 22 50 72 6f 74 65 63 ic realm ="Protec
74 65 64 20 41 72 65 61 22 0d 0a 0d 0a 3c 68 74 ted Area "...<ht
6d 6c 3e 0d 0a 3c 68 65 61 64 3e 3c 74 69 74 6c ml> <he ad><titl
65 3e 3a 30 31 20 41 75 74 68 6f 72 69 7a 61 74 e>401 Au thorizat
69 6f 6e 20 52 65 71 75 69 72 65 64 3c 2f 74 69 ion Requ ired</ti
74 6c 65 3e 3c 2f 68 65 61 64 3e 0d 0a 3c 62 6f tle></he ad>...<bo
64 79 3e 0d 0a 3c 63 65 6e 74 65 72 3e 3c 68 31 dy> <ce nter><h1
3e 3a 30 31 20 41 75 74 68 6f 72 69 7a 61 74 69 >401 Aut horizati
6f 6e 20 52 65 71 75 69 72 65 64 3c 2f 68 31 3e on Requ ired</h1>
3c 2f 63 65 6e 74 65 72 3e 0d 0a 3c 68 72 3e 3c </center>...<hr>
63 65 6e 74 65 72 3e 6e 67 69 6e 78 2f 31 2e 31 center>n ginx/1.1

Bytes 219-266: WWW-Authenticate (http.www_authenticate) | Packets: 74 · Displayed: 74 (100.0%) · Marked: 4 (5.4%) | Profile: Default
```

Next, I input the extremely secure username and password that the server requested, and we start the “second phase” of the connection. Once again, we see the TCP handshake, and my browser requests the folder again.

```
48 112.281775872 172.16.64.129 45.79.89.123 TCP 74 46308 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=3575.
49 112.327170023 45.79.89.123 172.16.64.129 TCP 60 80 → 46308 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
50 112.327221916 172.16.64.129 45.79.89.123 TCP 54 46308 → 80 [ACK] Seq=1 Ack=1 Win=64240 Len=0
51 112.327571178 172.16.64.129 45.79.89.123 HTTP 446 GET /basicauth/ HTTP/1.1
```

This GET, however, is different from the last. In the screenshot below, notice that the GET request from before I entered my credentials ends with the request line.

```
6c 2b 78 6d 6c 2c 61 70 70 6c 69 63 61 74 69 6f l+xml,ap plicatio
6e 2f 78 6d 6c 3b 71 3d 30 2e 39 2c 69 6d 61 67 n/xml;q= 0.9,imag
65 2f 77 65 62 70 2c 2a 2f 2a 3b 71 3d 30 2e 38 e/webp,* /*;q=0.8
0d 0a 41 63 63 65 70 74 2d 4c 61 6e 67 75 61 67 . Accept -Languag
65 3a 20 65 6e 2d 55 53 2c 65 6e 3b 71 3d 30 2e e: en-US ,en;q=0.
35 0d 0a 41 63 63 65 70 74 2d 45 6e 63 6f 64 69 5 .Accep t-Encodi
6e 67 3a 20 67 7a 69 70 2c 20 64 65 66 6c 61 74 ng: gzip , deflat
65 0d 0a 44 4e 54 3a 20 31 0d 0a 43 6f 6e 6e 65 e .DNT: 1 .Conne
63 74 69 6f 6e 3a 20 6b 65 65 70 2d 61 6c 69 76 ction: kee p-aliv
65 0d 0a 55 70 67 72 61 64 65 2d 49 6e 73 65 63 e .Upgra de-Insec
75 72 65 2d 52 65 71 75 65 73 74 73 3a 20 31 0d ure-Requ ests: 1
0a 0d 0a ...

Request line (http.request.line), 30 bytes | Packets: 74 · Displayed: 74 (100.0%) · Marked: 3 (4.1%) | Profile: Default
```

However, in the new GET, we see a different ending: the authorization header:

```
65 2f 77 65 62 70 2c 2a 2f 2a 3b 71 3d 30 2e 38 e/webp,*/*;q=0.8
0d 0a 41 63 63 65 70 74 2d 4c 61 6e 67 75 61 67 . Accept -Languag
65 3a 20 65 6e 2d 55 53 2c 65 6e 3b 71 3d 30 2e e: en-US,en;q=0.
35 0d 0a 41 63 63 65 70 74 2d 45 6e 63 6f 64 69 5: Accept-Encodi
6e 67 3a 20 67 7a 69 70 2c 20 64 65 66 6c 61 74 ng: gzip, deflat
65 0d 0a 44 4e 54 3a 20 31 0d 0a 43 6f 6e 6e 65 e. DNT: 1. Conne
63 74 69 6f 6e 3a 20 6b 65 65 70 2d 61 6c 69 76 ction: keep-aliv
65 0d 0a 55 70 67 72 61 64 65 2d 49 6e 73 65 63 e. Upgra de-Insec
75 72 65 2d 52 65 71 75 65 73 74 73 3a 20 31 0d ure-Requ ests: 1.
0a 41 75 74 68 6f 72 69 7a 61 74 69 6f 6e 3a 20 .Authori zation:
42 61 73 69 63 20 59 33 4d 7a 4d 7a 67 36 63 47 Basic Y3 MzMzg6cG
46 7a 63 33 64 76 63 6d 51 3d 0d 0a 0d 0a Fzc3dvcm Q=..
```

Bytes 401-443: Authorization (http.authorization) | Packets: 74 - Displayed: 74 (100.0%) - Marked: 3 (4.1%) | Profile: Default

This authorization header not only contains the information about what authorization type my browser is choosing to use, it also contains the password and username in plain text (!!!).

Although it's been converted into base64, the information is all available, so therefore using HTTPS or TLS is essentially mandatory when using basic authentication

([https://developer.mozilla.org/en-US/docs/Web/HTTP/Authentication#basic\\_authentication\\_scheme](https://developer.mozilla.org/en-US/docs/Web/HTTP/Authentication#basic_authentication_scheme) ).

To prove that this is truly the username and password in plaintext, we can simply take the string that is sent following the “Basic” declaration, (Y3MzMzg6cGFzc3dvcmQ=..) and input it into a base64 conversion tool. Doing so provides us with the promised information, as seen below:

## Decode from Base64 format

Simply enter your data then push the decode button.

Y3MzMzg6cGFzc3dvcmQ=..

For encoded binaries (like images, documents, etc.) use the file upload form a little further down on this page.

UTF-8 Source character set.

☐ Decode each line separately (useful for when you have multiple entries).

Live mode OFF Decodes in real-time as you type or paste (supports only the UTF-8 character set).

**DECODE** Decodes your data into the area below.

cs338:password

(courtesy of [base64decode.org](https://base64decode.org))

Servers and proxies implementing Basic authentication need to store user passwords in some form in order to authenticate a request. These passwords ought to be stored in such a way that a leak of the password data doesn't make them trivially recoverable. This is especially important when users are allowed to set their own passwords, since users are known to choose weak passwords and to reuse them across authentication realms. While a full discussion of

53	112	374724364	45	79	89	123	172	16	64	129	HTTP	458	HTTP/1.1 200 OK	(text/html)
54	112	374744625	172	16	64	129	45	79	89	123	HTTP	54	46308 - 80 [ACK]	Seq=393 Ack=405 Win=63836 Len=0
55	112	428983162	172	16	64	129	45	79	89	123	HTTP	363	GET /favicon.ico	HTTP/1.1
56	112	421277737	45	79	89	123	172	16	64	129	TCP	60	80 -> 46308 [ACK]	Seq=405 ACK=702 Win=64240 Len=0
57	112	467325500	45	79	89	123	172	16	64	129	HTTP	583	HTTP/1.1 404 Not Found	(text/html)
58	112	467339992	172	16	64	129	45	79	89	123	TCP	54	46308 -> 80 [ACK]	Seq=702 ACK=734 Win=63836 Len=0
Data: 1fb08000000000004038d91cb0ab3301045f785fe4370af6382ad50a6812efb19a38946...														
0030	fa	00	87	62	00	00	48	54	50	27	31	26	31	20 32
0040	39	f0	78	4f	4b	0d	0a	83	65	72	76	65	72	31 20 6e
0050	67	69	66	78	2f	31	2e	31	38	2e	30	20	25	62 75
0060	6e	74	75	29	0d	0a	44	61	74	65	3a	20	46	72 69 2c
0070	29	30	38	20	41	70	72	20	32	30	32	32	20	32 32 3a
0080	30	33	3a	32	37	20	47	4d	54	0d	0a	43	6f	6e 74 65
0090	6e	74	20	54	79	70	65	3a	20	74	65	78	74	2f 68 74
00a0	6d	6c	0d	0a	54	72	61	6e	73	66	65	72	24	45 6e 63
00b0	64	64	69	6e	67	3a	20	63	68	75	6e	6b	65	64 0d 0a
00c0	43	6f	6e	6e	65	63	74	69	6f	6e	3a	20	6b	65 65 70
00d0	2d	61	6c	69	76	65	0d	0a	43	6f	6e	74	65	6e 74 2d
00e0	45	6e	63	6f	64	69	6e	67	3a	20	67	74	69	70 0d 0a
00f0	0d	0a	63	64	0d	0a	1f	8b	08	00	00	00	00	04 03
0100	8d	91	cb	0a	83	30	10	45	f7	85	fe	43	70	af 63 82
0110	ad	50	a6	81	2e	fb	19	a3	89	46	f0	45	8c	60 ff be
0120	3e	2a	48	4b	d1	2c	02	33	99	c3	61	72	d1	b8 aa 94
0130	e7	13	1a	4d	4a	a2	2b	5c	a9	e5	b3	56	7a	60 4d c6
0140	20	a1	ae	48	a9	77	06	10	96	37	84	79	72	24 92 46

53	112	374724364	45.79.89.123	172.16.64.129	HTTP	458	HTTP/1.1 200 OK (text/html)										
54	112	374744625	172.16.64.129	45.79.89.123	TCP	54	60308 → 80 [ACK] Seq=393 Ack=405 Win=63836 Len=0										
55	112	420983162	172.16.64.129	45.79.89.123	TCP	363	GET /favicon.ico HTTP/1.1										
56	112	421277737	45.79.89.123	172.16.64.129	TCP	60	80 → 46308 [ACK] Seq=405 Ack=702 Win=64240 Len=0										
57	112	467325500	45.79.89.123	172.16.64.129	HTTP	383	HTTP/1.1 404 Not Found (text/html)										
58	112	467325500	45.79.89.123	172.16.64.129	TCP	54	60308 → 80 [ACK] Seq=702 Ack=734 Win=63836 Len=0										
Data: 1f8b0800000000004938d91c8ba83301045f785fe4370af6382ad50a6812efb19a38946...																	
00c0	43	6f	6e	6e	65	63	74	69	6f	6e	3a	20	6b	65	65	70	Connection: keep
00d0	2d	61	6c	69	76	65	0d	0a	43	6f	6e	74	65	6e	74	2d	-alive: Content-
00e0	45	6e	63	6f	64	0e	67	3a	20	6f	7a	69	70	0d	0a		Encoding : gzip;
00f0	04	6b	63	64	0e	67	3a	20	6f	7a	69	70	0d	0a			cf
0100	80	31	1b	6a	93	30	10	45	f7	85	fe	43	70	af	63	82	...
0110	ad	50	a8	81	2e	fb	19	a3	89	46	f0	45	8c	60	ff	be	...E...
0120	3e	2a	48	4b	d1	2c	02	33	99	c3	61	72	d1	b8	aa	94	...P...F.E...
0130	e7	13	14	ad	a4	2b	5c	a9	e5	b3	56	7a	60	4d	c6		>HK\3 ar...
0140	20	a1	ae	48	a9	77	06	10	96	37	84	79	72	24	92	46	...M3\ VZ'M...
0150	bd	66	92	ff	01	0c	97	68	ac	c4	d6	6a	89	c4	8c	d5	...Hw\7yr\$F...
0160	d9	dd	0b	02	f0	e4	78	21	d0	04	af	6d	aa	c8	e9	de	...f...h...j...
0170	76	81	1b	9c	27	b7	d5	34	c8	8e	9d	30	f2	1f	ad	f5	...x!...m...
0180	45	28	04	e3	d1	8d	87	bf	58	7c	d9	4a	6d	a5	95	9f	y...4...0...
0190	f7	64	05	2a	fe	ea	1c	94	ef	8b	f9	95	6f	c4	8a	ea	E(.....X Jm...
01a0	b4	a8	f3	65	d9	d4	71	50	37	af	b5	ef	14	22	1e	9d	...d...o...
01b0	30	7f	ff	94	03	ac	c1	27	f4	37	86	48	91	bf	fd		...eMqP7...H...
01c0	01	00	00	0d	0a	30	0d	0a	0d	0a							0.....7.H...

After this is done, my machine ACKs the message, and also requests the favicon for the page, though the server is apparently unable to provide and sends along a 404 not found message, alongside an ACK of the request.



51	112.327571178	172.16.64.129	45.79.89.123	HTTP	446 GET /basicauth/ HTTP/1.1
52	112.327983533	45.79.89.123	172.16.64.129	TCP	60 80 → 46308 [ACK] Seq=1 Ack=393 Win=64240 Len=0
53	112.374724364	45.79.89.123	172.16.64.129	HTTP	458 HTTP/1.1 200 OK (text/html)
54	112.374744625	172.16.64.129	45.79.89.123	TCP	54 46308 → 80 [ACK] Seq=393 Ack=405 Win=63836 Len=0
55	112.4209983162	172.16.64.129	45.79.89.123	HTTP	363 GET /favicon.ico HTTP/1.1
56	112.421277737	45.79.89.123	172.16.64.129	TCP	60 80 → 46308 [ACK] Seq=405 Ack=702 Win=64240 Len=0
57	112.467325500	45.79.89.123	172.16.64.129	HTTP	383 HTTP/1.1 404 Not Found (text/html)

My client ACKs this error, and that's essentially it! If we request any of the text files in the folder, we see a very similar request that's authenticated in the same way, with the authorization header at the bottom.

Authorization: Basic Y3MzMzg6cGFzc3dvcmQ=\r\n					
00a0	20 47 65 63 6b 6f 2f 32	30 31 30 30 31 30 31 20	Gecko/2.0.0.101	.....	
00b0	46 69 72 65 66 6f 78 2f	39 31 2e 30 0d 0a 41 63	Firefox/91.0.0.0	Ac	
00c0	63 65 70 74 3a 20 74 65	78 74 2f 68 74 6d 6c 2c	cept: text/html,		
00d0	61 70 70 6c 69 63 61 74	69 6f 6e 2f 78 68 74 6d	application/xhtml		
00e0	6c 2b 78 6d 6c 2c 61 70	70 6c 69 63 61 74 69 6f	l+xml,application		
00f0	6e 2f 78 6d 6c 3b 71 3d	30 2e 39 2c 69 6d 61 67	n/xml;q=0.9,image		
0100	65 2f 77 65 62 70 2c 2a	2f 2a 3b 71 3d 30 2e 38	e/webp,*/*;q=0.8		
0110	0d 0a 41 63 63 65 70 74	2d 4c 61 6e 67 75 61 67	..Accept-Language		
0120	65 3a 20 65 6e 2d 55 53	2c 65 6e 3b 71 3d 30 2e	e: en-US,en;q=0.		
0130	35 0d 0a 41 63 63 65 70	74 2d 45 6e 63 6f 64 69	5..Accept-Encoding		
0140	6e 67 3a 20 67 7a 69 70	2c 20 64 65 66 6c 61 74	ng: gzip, deflate		
0150	65 0d 0a 44 4e 54 3a 20	31 0d 0a 41 75 74 68 6f	e..DNT: 1..Autho		
0160	72 69 7a 61 74 69 6f 6e	3a 20 42 61 73 69 63 20	rization: Basic		
0170	59 33 4d 7a 4d 7a 67 36	63 47 46 7a 63 33 64 76	Y3MzMzg6cGFzc3dv		
0180	63 6d 51 3d 0d 0a 43 6f	6e 6e 65 63 74 69 6f 6e	cmQ=..Co nnection		
0190	3a 20 6b 65 65 70 2d 61	6c 69 76 65 0d 0a 55 70	: keep-alive..Up		
01a0	67 72 61 64 65 2d 49 6e	73 65 63 75 72 65 2d 52	grade-In secure-R		
01b0	65 71 75 65 73 74 73 3a	20 31 0d 0a 0d 0a	quests: 1....		

For the text files, the server simply responds with the direct cleartext contents:

Authorization: Basic Y3MzMzg6cGFzc3dvcmQ=\r\n					
00a0	20 47 65 63 6b 6f 2f 32	30 31 30 30 31 30 31 20	Gecko/2.0.0.101	.....	
00b0	46 69 72 65 66 6f 78 2f	39 31 2e 30 0d 0a 41 63	Firefox/91.0.0.0	Ac	
00c0	63 65 70 74 3a 20 74 65	78 74 2f 68 74 6d 6c 2c	cept: text/html,		
00d0	61 70 70 6c 69 63 61 74	69 6f 6e 2f 78 68 74 6d	application/xhtml		
00e0	6c 2b 78 6d 6c 2c 61 70	70 6c 69 63 61 74 69 6f	l+xml,application		
00f0	6e 2f 78 6d 6c 3b 71 3d	30 2e 39 2c 69 6d 61 67	n/xml;q=0.9,image		
0100	65 2f 77 65 62 70 2c 2a	2f 2a 3b 71 3d 30 2e 38	e/webp,*/*;q=0.8		
0110	0d 0a 41 63 63 65 70 74	2d 4c 61 6e 67 75 61 67	..Accept-Language		
0120	65 3a 20 65 6e 2d 55 53	2c 65 6e 3b 71 3d 30 2e	e: en-US,en;q=0.		
0130	35 0d 0a 41 63 63 65 70	74 2d 45 6e 63 6f 64 69	5..Accept-Encoding		
0140	6e 67 3a 20 67 7a 69 70	2c 20 64 65 66 6c 61 74	ng: gzip, deflate		
0150	65 0d 0a 44 4e 54 3a 20	31 0d 0a 41 75 74 68 6f	e..DNT: 1..Autho		
0160	72 69 7a 61 74 69 6f 6e	3a 20 42 61 73 69 63 20	rization: Basic		
0170	59 33 4d 7a 4d 7a 67 36	63 47 46 7a 63 33 64 76	Y3MzMzg6cGFzc3dv		
0180	63 6d 51 3d 0d 0a 43 6f	6e 6e 65 63 74 69 6f 6e	cmQ=..Co nnection		
0190	3a 20 6b 65 65 70 2d 61	6c 69 76 65 0d 0a 55 70	: keep-alive..Up		
01a0	67 72 61 64 65 2d 49 6e	73 65 63 75 72 65 2d 52	grade-In secure-R		
01b0	65 71 75 65 73 74 73 3a	20 31 0d 0a 0d 0a	quests: 1....		

From there on, the only messages are keepalive exchanges, which occur until the server eventually loses its patience and sends along a FIN, to which my client responds with a FIN ACK, responded to once more with the server's ACK.

63	133.615469055	45.79.89.123	172.16.64.129	TCP	60 [TCP Keep-Alive ACK] 80 → 46308 [ACK] Seq=734 Ack=702 Win=64240 Len=0
64	143.854483096	172.16.64.129	45.79.89.123	TCP	54 [TCP Keep-Alive] 46308 → 80 [ACK] Seq=701 Ack=734 Win=63836 Len=0
65	143.854901772	45.79.89.123	172.16.64.129	TCP	60 [TCP Keep-Alive ACK] 80 → 46308 [ACK] Seq=734 Ack=702 Win=64240 Len=0
66	154.093849876	172.16.64.129	45.79.89.123	TCP	54 [TCP Keep-Alive] 46308 → 80 [ACK] Seq=701 Ack=734 Win=63836 Len=0
67	154.094118027	45.79.89.123	172.16.64.129	TCP	60 [TCP Keep-Alive ACK] 80 → 46308 [ACK] Seq=734 Ack=702 Win=64240 Len=0
68	164.334550711	172.16.64.129	45.79.89.123	TCP	54 [TCP Keep-Alive] 46308 → 80 [ACK] Seq=701 Ack=734 Win=63836 Len=0
69	164.334951565	45.79.89.123	172.16.64.129	TCP	60 [TCP Keep-Alive ACK] 80 → 46308 [ACK] Seq=734 Ack=702 Win=64240 Len=0
70	174.573921310	172.16.64.129	45.79.89.123	TCP	54 [TCP Keep-Alive] 46308 → 80 [ACK] Seq=701 Ack=734 Win=63836 Len=0
71	174.574166991	45.79.89.123	172.16.64.129	TCP	60 [TCP Keep-Alive ACK] 80 → 46308 [ACK] Seq=734 Ack=702 Win=64240 Len=0
72	177.455969281	45.79.89.123	172.16.64.129	TCP	60 80 → 46308 [FIN, PSH, ACK] Seq=734 Ack=702 Win=64240 Len=0
73	177.456254858	172.16.64.129	45.79.89.123	TCP	54 46308 → 80 [FIN, ACK] Seq=702 Ack=735 Win=63836 Len=0
74	177.456662759	45.79.89.123	172.16.64.129	TCP	60 80 → 46308 [ACK] Seq=735 Ack=703 Win=64239 Len=0

And so, the client and server lived happily ever after ☺