HTTP Basic Authentication

Peter Kelly

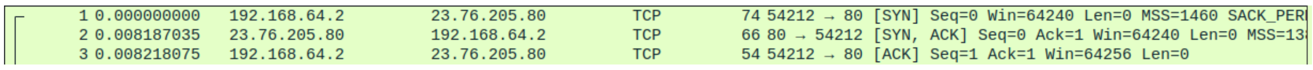
September 21, 2023

1. To begin my connection, my computer and the server complete the TCP handshake [1]. Then my computer requests the HTML for the web-page at index "/basicauth/", but is rejected with the error code "401 Unauthorized" [2] as I need authorization to see a web page in the "Protected Area" realm. The web client sees this and then prompts me to enter my login information. Once I type in my password, my computer then sends the GET request again, this time with an Authorization : Basic header that gives the server my credentials in base64 [3]. In this case the base64 credential was "Y3MzMzg6cGFzc3dvcmQ=" and the plain text credentials were "cs338:password." The server then acknowledges my GET request with an "ACK" packet [4]. Then the server sends a "200 OK" packet that includes the HTML in its "Line-based text data" header [5]. My web browser then caches my credentials so that if I need to request a page from the "Protected Area" realm again I do not need to reenter my username and password [6]. The web browser remembers my credentials and automatically sends them with the GET request, bypassing the need for the server to prompt the client with the "401 Unauthorized" error.

2. The password is sent to the server, though once the web browser has the username and password it caches them so that it can resend the credentials in subsequent GET requests.

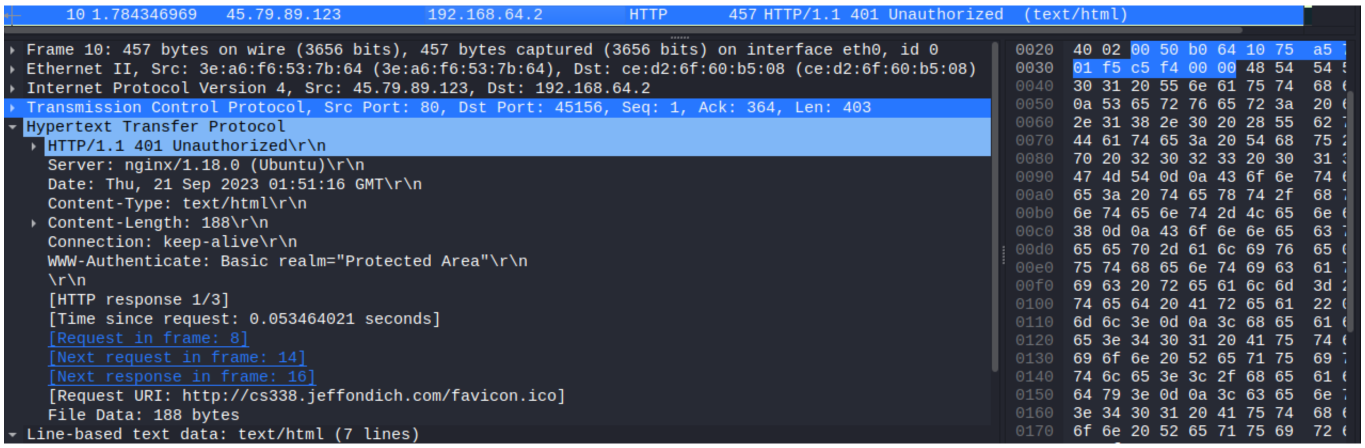
3. When the password is sent, it is sent in base64. Wireshark automatically translates this, as it is not encrypted, back into readable text.

4. One of the easiest ways to see how Wireshark shows the implementation of Basic Authorization is in the Authorization header. The RFC states that it must be of the form "Authorization: Basic base64Credentials" where the code after basic is the base64 username and password separated by a colon. In Wireshark I get the string "Authorization: Basic Y3MzMzg6cGFzc3dvcmQ=\r\n" where the \r \n denotes white space. This is of the same form as the specification.



```
  1 0.000000000   192.168.64.2     23.76.205.80     TCP      74 54212 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PER
  2 0.008187035   23.76.205.80     192.168.64.2     TCP      66 80 → 54212 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=13
  3 0.008218075   192.168.64.2     23.76.205.80     TCP      54 54212 → 80 [ACK] Seq=1 Ack=1 Win=64256 Len=0
```

Figure 1: TCP Handshake

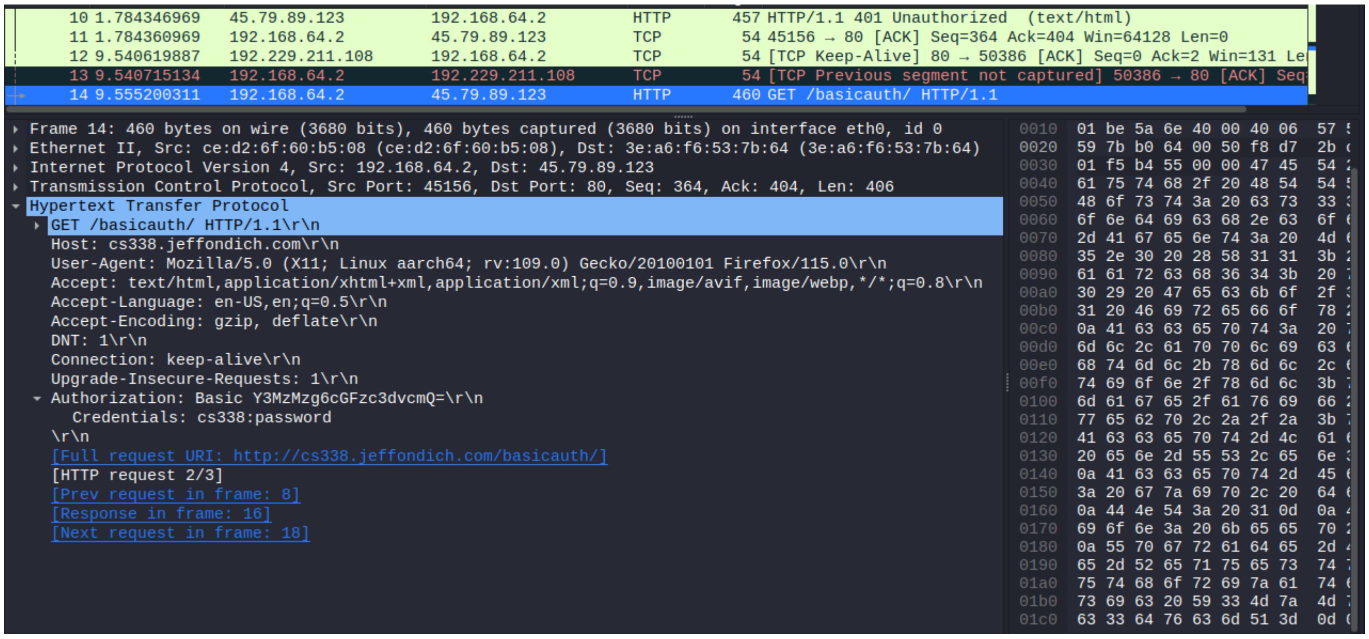


```
 10 1.784346969   45.79.89.123     192.168.64.2     HTTP     457 HTTP/1.1 401 Unauthorized  (text/html)
```

```
› Frame 10: 457 bytes on wire (3656 bits), 457 bytes captured (3656 bits) on interface eth0, id 0
› Ethernet II, Src: 3e:a6:f6:53:7b:64 (3e:a6:f6:53:7b:64), Dst: ce:d2:6f:60:b5:08 (ce:d2:6f:60:b5:08)
› Internet Protocol Version 4, Src: 45.79.89.123, Dst: 192.168.64.2
› Transmission Control Protocol, Src Port: 80, Dst Port: 45156, Seq: 1, Ack: 364, Len: 403
˅ Hypertext Transfer Protocol
  › HTTP/1.1 401 Unauthorized\r\n
    Server: nginx/1.18.0 (Ubuntu)\r\n
    Date: Thu, 21 Sep 2023 01:51:16 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
    [HTTP response 1/3]
    [Time since request: 0.053464021 seconds]
    [Request in frame: 8]
    [Next request in frame: 14]
    [Next response in frame: 16]
    [Request URI: http://cs338.jeffondich.com/favicon.ico]
    File Data: 188 bytes
˅ Line-based text data: text/html (7 lines)
```

```
0020  40 02 00 50 b0 64 10 75  a5
0030  01 f5 c5 f4 00 00 48 54  54
0040  30 31 20 55 6e 61 75 74  68
0050  0a 53 65 72 76 65 72 3a  20
0060  2e 31 38 2e 30 20 28 55  62
0070  44 61 74 65 3a 20 54 68  75
0080  70 20 32 30 32 33 20 30  31
0090  47 4d 54 0d 0a 43 6f 6e  74
00a0  65 3a 20 74 65 78 74 2f  68
00b0  6e 74 65 6e 74 2d 4c 65  6e
00c0  38 0d 0a 43 6f 6e 6e 65  63
00d0  65 65 70 2d 61 6c 69 76  65
00e0  75 74 68 65 6e 74 69 63  61
00f0  69 63 20 72 65 61 6c 6d  3d
0100  74 65 64 20 41 72 65 61  22
0110  6d 6c 3e 0d 0a 3c 68 65  61
0120  65 3e 34 30 31 20 41 75  74
0130  69 6f 6e 20 52 65 71 75  69
0140  74 6c 65 3e 3c 2f 68 65  61
0150  64 79 3e 0d 0a 3c 63 65  6e
0160  3e 34 30 31 20 41 75 74  68
0170  6f 6e 20 52 65 71 75 69  72
```

Figure 2: 401 Unauthorized Request from the server to the client, the page requested is in a

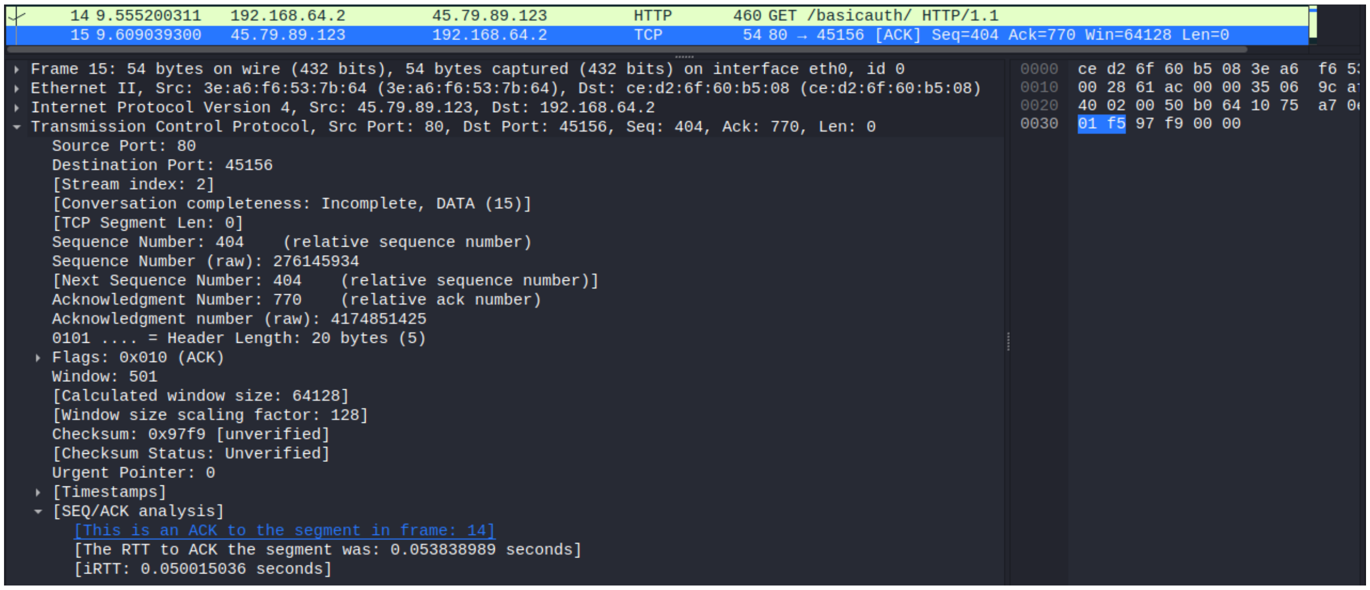


```
 10 1.784346969   45.79.89.123     192.168.64.2     HTTP     457 HTTP/1.1 401 Unauthorized  (text/html)
 11 1.784360969   192.168.64.2     45.79.89.123     TCP       54 45156 → 80 [ACK] Seq=364 Ack=404 Win=64128 Len=0
 12 9.540619887   192.229.211.108  192.168.64.2     TCP       54 [TCP Keep-Alive] 80 → 50386 [ACK] Seq=0 Ack=2 Win=131 Le
 13 9.540715134   192.168.64.2     192.229.211.108  TCP       54 [TCP Previous segment not captured] 50386 → 80 [ACK] Seq
 14 9.555200311   192.168.64.2     45.79.89.123     HTTP     460 GET /basicauth/ HTTP/1.1
```

```
› Frame 14: 460 bytes on wire (3680 bits), 460 bytes captured (3680 bits) on interface eth0, id 0
› Ethernet II, Src: ce:d2:6f:60:b5:08 (ce:d2:6f:60:b5:08), Dst: 3e:a6:f6:53:7b:64 (3e:a6:f6:53:7b:64)
› Internet Protocol Version 4, Src: 192.168.64.2, Dst: 45.79.89.123
› Transmission Control Protocol, Src Port: 45156, Dst Port: 80, Seq: 364, Ack: 404, Len: 406
˅ Hypertext Transfer Protocol
  › GET /basicauth/ HTTP/1.1\r\n
    Host: cs338.jeffondich.com\r\n
    User-Agent: Mozilla/5.0 (X11; Linux aarch64; 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
    DNT: 1\r\n
    Connection: keep-alive\r\n
    Upgrade-Insecure-Requests: 1\r\n
  ˅ Authorization: Basic Y3MzMzg6cGFzc3dvcmQ=\r\n
      Credentials: cs338:password
    \r\n
    [Full request URI: http://cs338.jeffondich.com/basicauth/]
    [HTTP request 2/3]
    [Prev request in frame: 8]
    [Response in frame: 16]
    [Next request in frame: 18]
```

```
0010  01 be 5a 6e 40 00 40 06  57
0020  59 7b b0 64 00 50 f8 d7  2b
0030  01 f5 b4 55 00 00 47 45  54
0040  61 75 74 68 2f 20 48 54  54
0050  48 6f 73 74 3a 20 63 73  33
0060  6f 6e 64 69 63 68 2e 63  6f
0070  2d 41 67 65 6e 74 3a 20  4d
0080  35 2e 30 20 28 58 31 31  3b
0090  61 61 72 63 68 36 34 3b  20
00a0  30 29 20 47 65 63 6b 6f  2f
00b0  31 20 46 69 72 65 66 6f  78
00c0  0a 41 63 63 65 70 74 3a  20
00d0  6d 6c 2c 61 70 70 6c 69  63
00e0  68 74 6d 6c 2b 78 6d 6c  2c
00f0  74 69 6f 6e 2f 78 6d 6c  3b
0100  6d 61 67 65 2f 61 76 69  66
0110  77 65 62 70 2c 2a 2f 2a  3b
0120  41 63 63 65 70 74 2d 4c  61
0130  20 65 6e 2d 55 53 2c 65  6e
0140  0a 41 63 63 65 70 74 2d  45
0150  3a 20 67 7a 69 70 2c 20  64
0160  0a 44 4e 54 3a 20 31 0d  0a
0170  69 6f 6e 3a 20 6b 65 65  70
0180  0a 55 70 67 72 61 64 65  2d
0190  65 2d 52 65 71 75 65 73  74
01a0  75 74 68 6f 72 69 7a 61  74
01b0  73 69 63 20 59 33 4d 7a  4d
01c0  63 33 64 76 63 6d 51 3d  0d
```

Figure 3: The second request the client sends. It sends this because it was refused with the 401 Unauthorized error so the client knows it needs to send the get request with the Authorization Basic credentials.

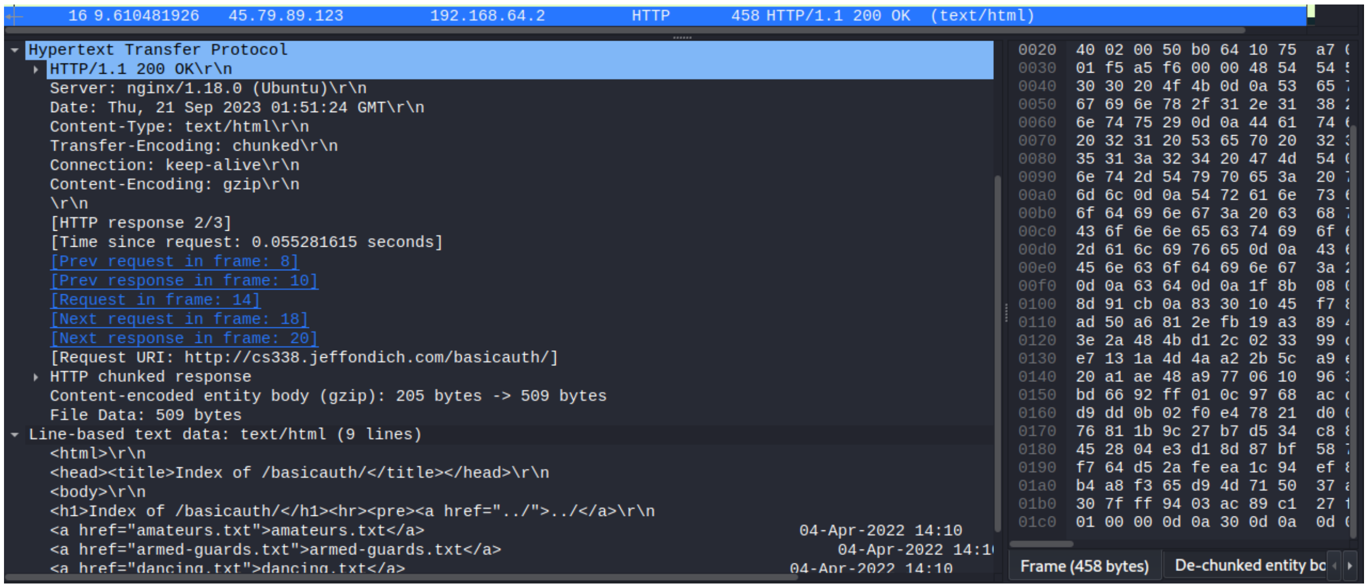


```
14 9.555200311  192.168.64.2     45.79.89.123     HTTP     460 GET /basicauth/ HTTP/1.1
15 9.609039300  45.79.89.123     192.168.64.2     TCP       54 80 → 45156 [ACK] Seq=404 Ack=770 Win=64128 Len=0
```

```
▸ Frame 15: 54 bytes on wire (432 bits), 54 bytes captured (432 bits) on interface eth0, id 0
▸ Ethernet II, Src: 3e:a6:f6:53:7b:64 (3e:a6:f6:53:7b:64), Dst: ce:d2:6f:60:b5:08 (ce:d2:6f:60:b5:08)
▸ Internet Protocol Version 4, Src: 45.79.89.123, Dst: 192.168.64.2
▾ Transmission Control Protocol, Src Port: 80, Dst Port: 45156, Seq: 404, Ack: 770, Len: 0
    Source Port: 80
    Destination Port: 45156
    [Stream index: 2]
    [Conversation completeness: Incomplete, DATA (15)]
    [TCP Segment Len: 0]
    Sequence Number: 404    (relative sequence number)
    Sequence Number (raw): 276145934
    [Next Sequence Number: 404    (relative sequence number)]
    Acknowledgment Number: 770    (relative ack number)
    Acknowledgment number (raw): 4174851425
    0101 .... = Header Length: 20 bytes (5)
  ▸ Flags: 0x010 (ACK)
    Window: 501
    [Calculated window size: 64128]
    [Window size scaling factor: 128]
    Checksum: 0x97f9 [unverified]
    [Checksum Status: Unverified]
    Urgent Pointer: 0
  ▸ [Timestamps]
  ▾ [SEQ/ACK analysis]
      [This is an ACK to the segment in frame: 14]
      [The RTT to ACK the segment was: 0.053838989 seconds]
      [iRTT: 0.050015036 seconds]
```

```
0000  ce d2 6f 60 b5 08 3e a6  f6 5
0010  00 28 61 ac 00 00 35 06  9c a
0020  40 02 00 50 b0 64 10 75  a7 0
0030  01 f5 97 f9 00 00
```

Figure 4: The server acknowledges it received my HTTP request with the proper credentials for the realm.

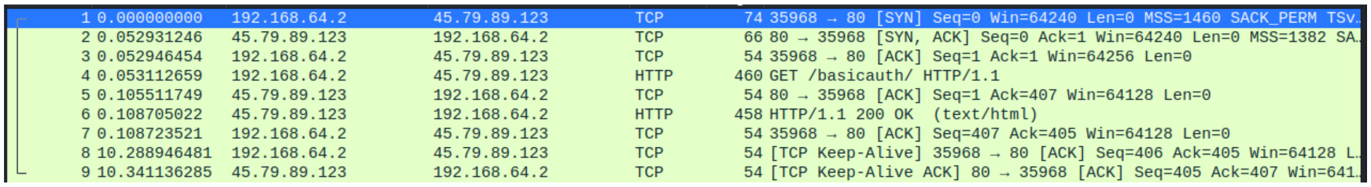


```
16 9.610481926  45.79.89.123     192.168.64.2     HTTP     458 HTTP/1.1 200 OK  (text/html)
```

```
▾ Hypertext Transfer Protocol
  ▸ HTTP/1.1 200 OK\r\n
    Server: nginx/1.18.0 (Ubuntu)\r\n
    Date: Thu, 21 Sep 2023 01:51:24 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
    [HTTP response 2/3]
    [Time since request: 0.055281615 seconds]
    [Prev request in frame: 8]
    [Prev response in frame: 10]
    [Request in frame: 14]
    [Next request in frame: 18]
    [Next response in frame: 20]
    [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 14:1
    <a href="dancing.txt">dancing.txt</a>                                            04-Apr-2022 14:10
```

```
0020  40 02 00 50 b0 64 10 75  a7
0030  01 f5 a5 f6 00 00 48 54  54
0040  30 30 20 4f 4b 0d 0a 53  65
0050  67 69 6e 78 2f 31 2e 31  38
0060  6e 74 75 29 0d 0a 44 61  74
0070  20 32 31 20 53 65 70 20  32
0080  35 31 3a 32 34 20 47 4d  54
0090  6e 74 2d 54 79 70 65 3a  20
00a0  6d 6c 0d 0a 54 72 61 6e  73
00b0  6f 64 69 6e 67 3a 20 63  68
00c0  43 6f 6e 6e 65 63 74 69  6f
00d0  2d 61 6c 69 76 65 0d 0a  43
00e0  45 6e 63 6f 64 69 6e 67  3a
00f0  0d 0a 63 64 0d 0a 1f 8b  08
0100  8d 91 cb 0a 83 30 10 45  f7
0110  ad 50 a6 81 2e fb 19 a3  89
0120  3e 2a 48 4b d1 2c 02 33  99
0130  e7 13 1a 4d 4a a2 2b 5c  a9
0140  20 a1 ae 48 a9 77 06 10  96
0150  bd 66 92 ff 01 0c 97 68  ac
0160  d9 dd 0b 02 f0 e4 78 21  d0
0170  76 81 1b 9c 27 b7 d5 34  c8
0180  45 28 04 e3 d1 8d 87 bf  58
0190  f7 64 d5 2a fe ea 1c 94  ef
01a0  b4 a8 f3 65 d9 4d 71 50  37
01b0  30 7f ff 94 03 ac 89 c1  27
01c0  01 00 00 0d 0a 30 0d 0a  0d
```

Figure 5: The HTML the server sends to me after receiving my GET request with the proper credentials for the ”Protected Area” realm.



```
1 0.000000000   192.168.64.2     45.79.89.123     TCP       74 35968 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSv…
2 0.052931246   45.79.89.123     192.168.64.2     TCP       66 80 → 35968 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1382 SA…
3 0.052946454   192.168.64.2     45.79.89.123     TCP       54 35968 → 80 [ACK] Seq=1 Ack=1 Win=64256 Len=0
4 0.053112659   192.168.64.2     45.79.89.123     HTTP     460 GET /basicauth/ HTTP/1.1
5 0.105511749   45.79.89.123     192.168.64.2     TCP       54 80 → 35968 [ACK] Seq=1 Ack=407 Win=64128 Len=0
6 0.108705022   45.79.89.123     192.168.64.2     HTTP     458 HTTP/1.1 200 OK  (text/html)
7 0.108723521   192.168.64.2     45.79.89.123     TCP       54 35968 → 80 [ACK] Seq=407 Ack=405 Win=64128 Len=0
8 10.288946481  192.168.64.2     45.79.89.123     TCP       54 [TCP Keep-Alive] 35968 → 80 [ACK] Seq=406 Ack=405 Win=64128 L…
9 10.341136285  45.79.89.123     192.168.64.2     TCP       54 [TCP Keep-Alive ACK] 80 → 35968 [ACK] Seq=405 Ack=407 Win=641…
```

Figure 6: The packet interaction that occurs when the client has cached the username and password