Nick Pandelakis & Owen Barnett

CS 231

9 April 2021

Basic Authentication Story

The first thing my computer does when we go to the http://cs231.jeffondich.com/ is try to establish two TCP connections to the server, shown below in frames 1 and 2. One of the connections is from port 60304 on my computer and the other is from port 60306, both connections are to port 80 on the server, which corresponds to the HTTP port. The likely reason two connections are opened is that HTTP allows parallel transfers, so connecting with two ports increases the total amount of traffic that can be sent between the server and my computer.

Frame No.	Source	Destination	Protocol	Info
1	10.0.2.15	45.79.89.123	TCP	60304 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=4265727647 TSecr=0 WS=128
2	10.0.2.15	45.79.89.123	TCP	Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1

				TSval=4265727647 TSecr=0 WS=128

Once the two TCP three-way handshakes are finished, my computer now sends a GET request for the website. The website sends back the HTML for the web page using HTTP along with a 200 status code since the transfer was successful. My computer now displays the webpage in the browser. The headers for frame 5 and 6 are displayed below. These correspond to the GET request and the server's response.

Frame No.	Source	Destination	Protocol	Info
5	10.0.2.15	45.79.89.123	HTTP	GET /index.html HTTP/1.1
6	45.79.89.123	10.0.2.15	TCP	80 → 60306 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460

Now I click on the secrets link in the browser. This causes my computer to send a HTTP request for the URI http://cs23.jeffondich.com/basicauth/, which is shown frame 11. Since the website is password protected, the server responds with a 401 error in frame 13, telling my computer that it is currently unauthorized to access this resource. This frame includes the HTML to display the popup asking for authentication. In frame

19 my computer again requests /basicauth/, this time sending a username and password in a base64 encoded string.

Frame No.	Source	Destination	Protocol	Info
11	10.0.2.15	45.79.89.123	HTTP	GET /basicauth/ HTTP/1.1
13	45.79.89.123	10.0.2.15	HTTP	HTTP/1.1 401 Unauthorized (text/html)
19	10.0.2.15	45.79.89.123	HTTP	GET /basicauth/ HTTP/1.1

Frame 19 Data:

```
Authorization: Basic Y3MyMzE6cGFzc3dvcmQ=\r\n
    Credentials: cs231:password
```

Although the credentials we provided the server with are not exactly plain text, the base64 encoding is so weak that Wireshark is able to decode and display the credentials without the user even asking. This obviously poses an enormous security risk, since anyone who is able to view this packet would immediately know our username and password for this site.

Although there could be use cases for HTTP where no sensitive information is ever needed--such as for a website with no credential requirements--or where the route the packets take from client to server is itself completely secured, this transaction shows one of the shortcomings of HTTP. If an actor has the ability to view packets at any point along the route, there will be no secrets between the client and the server.

One way to abuse this would be if I were to set up a free WiFi router in some public place. Since I control the router, I would be able to get all the network traffic that goes through it. If someone logs into a website through my router using HTTP, I would be able to get their username and password. Additionally, since most people only use a small number of passwords and usernames are easy to acquire, I would probably be able to access their private information.

HTTP is starkly different from a secure connection protocol, where packets contain encrypted information rather than plaintext. An example of this would be when my machine connects to Google, packets contain data such as

Encrypted Application Data: b5e92a8e7ebac2dd72b05bdf35f9e3

c5b2818154c77e34d488979a16e3d100b274db

This data is totally nonsensical without the proper keys, and is robust against an attack like the one mentioned in the previous paragraph.

While the security exchange is going on, my computer decided to close one of the two TCP connections. I think this happened because my computer decided that there was not enough traffic to justify having two TCP connections opened which both drain resources. The servers acknowledged the FIN request and closed the connection to port 60306. This interaction is shown in frames 15 and 17, and it finished before the authentication finished.

Frame No.	Source	Destination	Protocol	Info

15	10.0.2.15	45.79.89.123	TCP	60306 → 80 [FIN, ACK] Seq=1 Ack=1 Win=64240 Len=0
17	45.79.89.123	10.0.2.15	TCP	80 → 60306 [FIN, ACK] Seq=1 Ack=2 Win=65535 Len=0

After the packet that contained our credentials was sent to the server in frame 19, the server verified our connection and responded with frame 21, notifying the client that the credentials were verified by responding with HTTP 200, which contains the HTML code for http://cs231.jeffondich.com/basicauth/. After the page rendered, we stopped recording the traffic being sent to 45.79.89.123.

Frame No.	Source	Destination	Protocol	Info
21	45.79.89.123	10.0.2.15	HTTP	HTTP/1.1 200 OK (text/html)