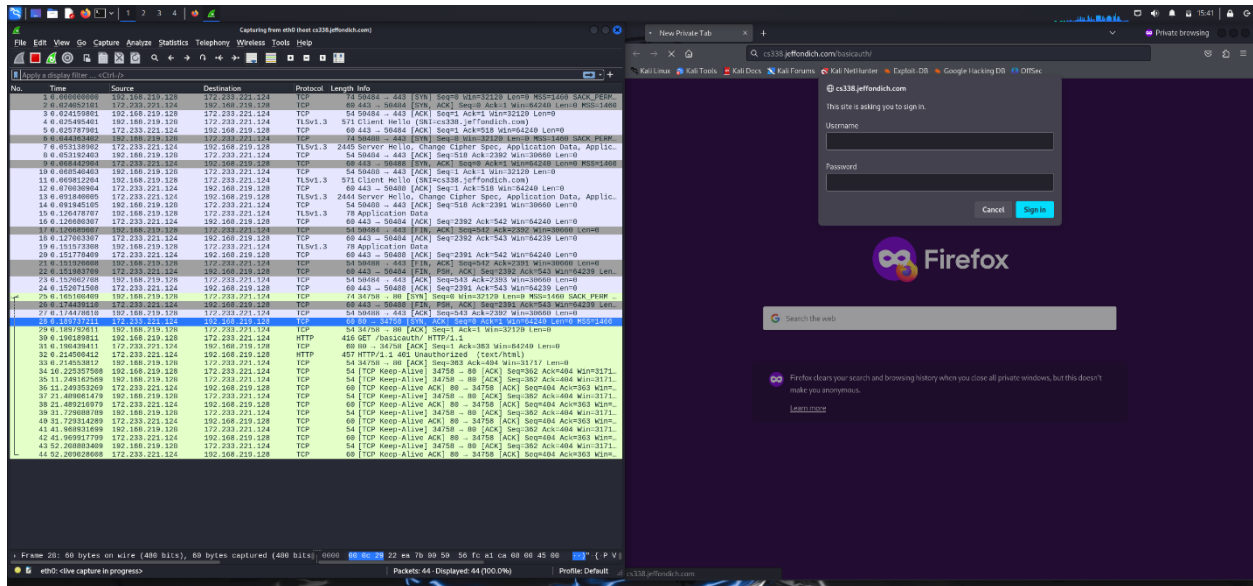


User authentication when accessing webservers

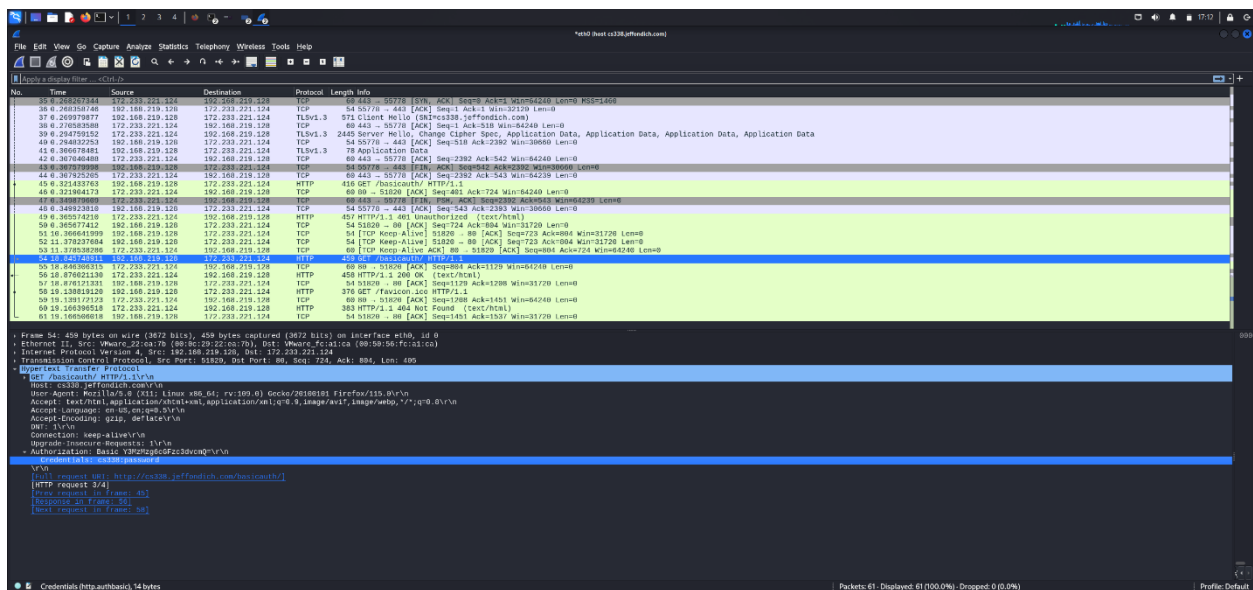
Most websites we visit nowadays want us to create some kind of account, often blocking access to content behind certain kinds of accounts. Certain webservers have easily configurable authentication abilities built in. In this case, we will be looking at Nginx.

Here, we can see our first attempt to go to the website <http://cs338.jeffondich.com/basicauth/>



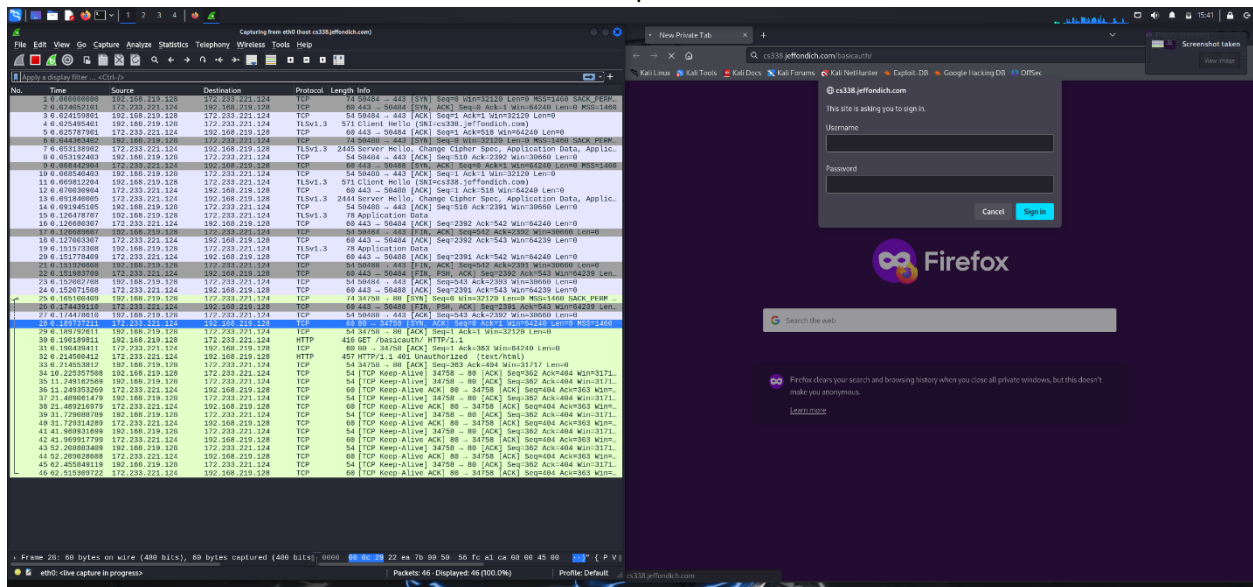
On the left, we have wireshark: a program that lets us see network traffic. That is, we can see what signals our computer is sending and receiving to the webserver that hosts [cs338.jeffondich.com](http://cs338.jeffondich.com/basicauth/). When we try to go to <http://cs338.jeffondich.com/basicauth/>, we don't even have the chance to load the page before the password popup appears. In frames number 1-3 and 7-10, we can see our computer opening up TCP connections with the webserver. Normally, all of this information would be unencrypted because of we are using http to access the website instead of https. However, possibly do to the password prompt, these are secure.

Once we enter a password however, it is not send via an encrypted channel:

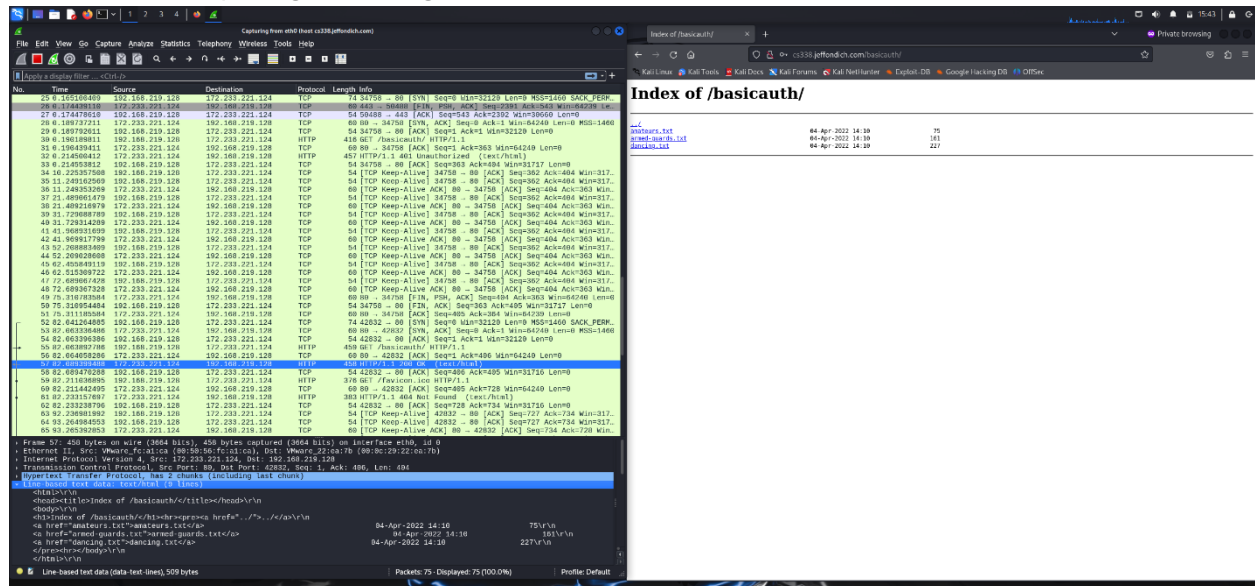


Here we can clearly see the username and password we used to access the website. Some research into Nginx shows that the basic authentication it provides [only uses base64](#) encoding which is easily unencoded with any need for an encryption key.

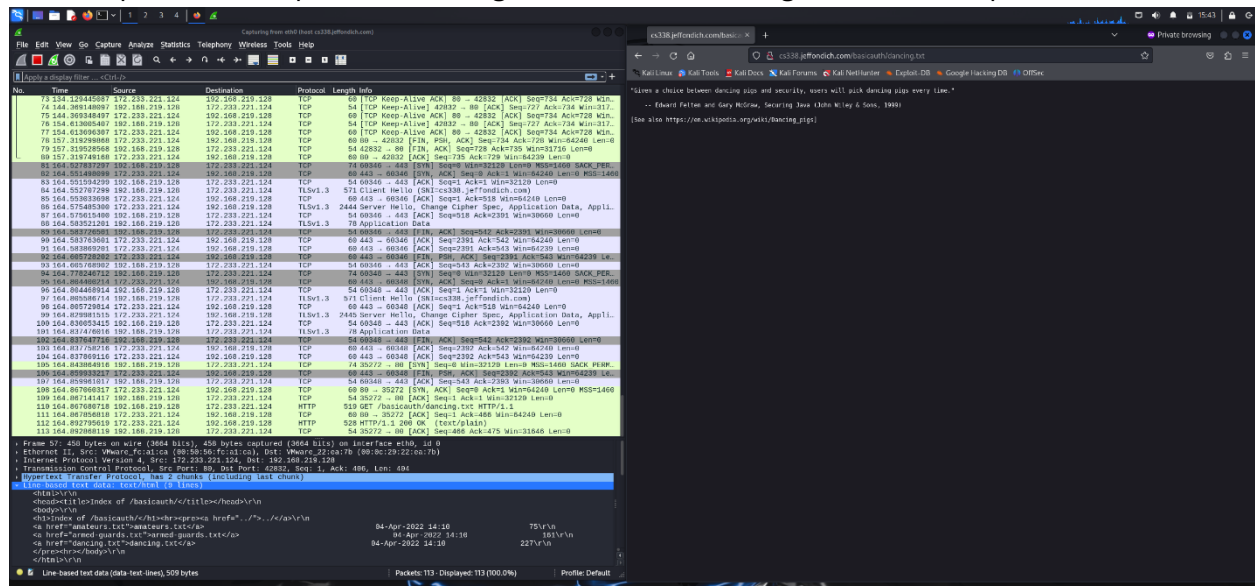
Attempts to access the website with incorrect credentials will get a reply from the server saying that access is unauthorized. See Frame 30 for an example.



Once we have successfully entered our credentials, we can access the desired page. On Frame 57, our HTTP GET request goes through.



As we try to access other pages on the website, our credentials are checked again. Thankfully, our browser can do this for us, without having to reenter anything. The grey lines are a repeat of the same requests and responses that we got when first accessing the /basicauth path.



Each new page on the website goes through this process again. Interestingly, pressing the back arrow can avoid this. This likely means that the connection is remembered from out previous access of the page and that there is no need to go back to the server to update the page. This is probably done through browser caching. Since private browsing windows promise that they are not caching, I assume that this is done only in memory, although browsers have been known to lie about not

caching things before.

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

Apply a display filter: «Clear»

No.	Time	Source	Destination	Protocol	Length	Info
111	176.0392562	192.168.219.128	172.233.221.124	TCP	54	TCP Reset-Alive! 35272 → 88 [ACK] Seq=450 Acc=475 Win=0
115	176.131242181	192.168.219.128	172.233.221.124	TCP	54	TCP Keep-Alive! 35272 → 88 [ACK] Seq=450 Acc=475 Win=0
116	176.213289387	172.233.221.124	192.168.219.128	TCP	60	TCP Keep-Alive ACK 88 → 35272 [ACK] Seq=475 Acc=460 Win=0
117	186.263248097	192.168.219.128	172.233.221.124	TCP	54	TCP Keep-Alive! 35272 → 88 [ACK] Seq=450 Acc=475 Win=0
118	186.353476987	172.233.221.124	192.168.219.128	TCP	60	TCP Keep-Alive ACK 88 → 35272 [ACK] Seq=475 Acc=460 Win=0
119	186.567829181	192.168.219.128	172.233.221.124	TCP	54	TCP Keep-Alive! 35272 → 88 [ACK] Seq=450 Acc=475 Win=0
120	186.638945187	172.233.221.124	192.168.219.128	TCP	60	TCP Keep-Alive ACK 88 → 35272 [ACK] Seq=475 Acc=460 Win=0
121	206.048689987	192.168.219.128	172.233.221.124	TCP	54	TCP Keep-Alive! 35272 → 88 [ACK] Seq=450 Acc=475 Win=0
122	206.872515157	172.233.221.124	192.168.219.128	TCP	60	TCP Keep-Alive ACK 88 → 35272 [ACK] Seq=475 Acc=460 Win=0
123	206.798644887	192.168.219.128	172.233.221.124	TCP	54	TCP Keep-Alive! 35272 → 88 [ACK] Seq=450 Acc=475 Win=0
124	206.458427284	172.233.221.124	192.168.219.128	TCP	60	TCP Keep-Alive ACK 88 → 35272 [ACK] Seq=475 Acc=460 Win=0
125	206.615764384	192.168.219.128	172.233.221.124	TCP	54	TCP Keep-Alive! 35272 → 88 [ACK] Seq=450 Acc=475 Win=0
126	206.817472484	192.168.219.128	172.233.221.124	TCP	54	TCP Keep-Alive! 35272 → 88 [ACK] Seq=450 Acc=475 Win=0
127	206.617644884	172.233.221.124	192.168.219.128	TCP	60	TCP Keep-Alive ACK 88 → 35272 [ACK] Seq=475 Acc=460 Win=0
128	206.832689786	172.233.221.124	192.168.219.128	TCP	60	TCP Keep-Alive ACK 88 → 35272 [ACK] Seq=475 Acc=460 Win=0
129	206.642144884	192.168.219.128	172.233.221.124	TCP	54	TCP Keep-Alive! 35272 → 88 [ACK] Seq=450 Acc=475 Win=0
130	206.833535986	192.168.219.128	172.233.221.124	TCP	60	TCP Keep-Alive ACK 88 → 35272 [ACK] Seq=475 Acc=460 Win=0
131	206.648444884	192.168.219.128	172.233.221.124	TCP	54	TCP Keep-Alive! 35272 → 88 [ACK] Seq=450 Acc=475 Win=0
132	206.845010786	172.233.221.124	192.168.219.128	TCP	60	TCP Keep-Alive ACK 88 → 35272 [ACK] Seq=475 Acc=460 Win=0
133	206.648620986	172.233.221.124	192.168.219.128	TCP	60	TCP Keep-Alive ACK 88 → 35272 [ACK] Seq=475 Acc=460 Win=0
134	206.872515157	172.233.221.124	192.168.219.128	TCP	60	TCP Keep-Alive ACK 88 → 35272 [ACK] Seq=475 Acc=460 Win=0
135	206.617644884	192.168.219.128	172.233.221.124	TCP	54	TCP Keep-Alive! 35272 → 88 [ACK] Seq=450 Acc=475 Win=0
136	206.833535986	192.168.219.128	172.233.221.124	TCP	60	TCP Keep-Alive ACK 88 → 35272 [ACK] Seq=475 Acc=460 Win=0
137	206.617644884	192.168.219.128	172.233.221.124	TCP	54	TCP Keep-Alive! 35272 → 88 [ACK] Seq=450 Acc=475 Win=0
138	216.687786286	192.168.219.128	172.233.221.124	TCP	54	TCP Keep-Alive! 35272 → 88 [ACK] Seq=450 Acc=475 Win=0
139	216.689779686	192.168.219.128	172.233.221.124	TCP	54	TCP Keep-Alive! 35272 → 88 [ACK] Seq=450 Acc=475 Win=0
140	216.610879686	172.233.221.124	192.168.219.128	TCP	60	TCP Keep-Alive ACK 88 → 35272 [ACK] Seq=475 Acc=460 Win=0
141	216.633480807	172.233.221.124	192.168.219.128	TCP	60	TCP Keep-Alive ACK 88 → 35272 [ACK] Seq=475 Acc=460 Win=0
142	216.633480807	192.168.219.128	172.233.221.124	TCP	54	TCP Keep-Alive! 35272 → 88 [ACK] Seq=450 Acc=475 Win=0
143	216.633480807	192.168.219.128	172.233.221.124	TCP	54	TCP Keep-Alive! 35272 → 88 [ACK] Seq=450 Acc=475 Win=0
144	216.648689987	172.233.221.124	192.168.219.128	TCP	60	TCP Keep-Alive ACK 88 → 35272 [ACK] Seq=475 Acc=460 Win=0
145	216.648689987	192.168.219.128	172.233.221.124	TCP	60	TCP Keep-Alive ACK 88 → 35272 [ACK] Seq=475 Acc=460 Win=0
146	216.648689987	192.168.219.128	172.233.221.124	TCP	60	TCP Keep-Alive ACK 88 → 35272 [ACK] Seq=475 Acc=460 Win=0
147	216.648689987	192.168.219.128	172.233.221.124	TCP	60	TCP Keep-Alive ACK 88 → 35272 [ACK] Seq=475 Acc=460 Win=0
148	216.648689987	192.168.219.128	172.233.221.124	TCP	60	TCP Keep-Alive ACK 88 → 35272 [ACK] Seq=475 Acc=460 Win=0
149	216.648689987	192.168.219.128	172.233.221.124	TCP	60	TCP Keep-Alive ACK 88 → 35272 [ACK] Seq=475 Acc=460 Win=0
150	216.648689987	192.168.219.128	172.233.221.124	TCP	60	TCP Keep-Alive ACK 88 → 35272 [ACK] Seq=475 Acc=460 Win=0
151	216.648689987	192.168.219.128	172.233.221.124	TCP	60	TCP Keep-Alive ACK 88 → 35272 [ACK] Seq=475 Acc=460 Win=0
152	216.648689987	192.168.219.128	172.233.221.124	TCP	60	TCP Keep-Alive ACK 88 → 35272 [ACK] Seq=475 Acc=460 Win=0
153	216.648689987	192.168.219.128	172.233.221.124	TCP	60	TCP Keep-Alive ACK 88 → 35272 [ACK] Seq=475 Acc=460 Win=0
154	216.648689987	192.168.219.128	172.233.221.124	TCP	60	TCP Keep-Alive ACK 88 → 35272 [ACK] Seq=475 Acc=460 Win=0

Frame 07: 54 bytes on wire (432 bits), 54 bytes captured (432 bits) on interface eth0, 1d 0
Ethernet II, Src: VMware_22:6a:7b (00:0c:29:22:6a:7b), Dst: VMware_fca1ca (00:0c:29:fca1:ca)
Internet Protocol Version 4, Src: 192.168.219.128, Dst: 172.233.221.124
Transmission Control Protocol, Src Port: 60346, Dst Port: 443, Seq: 518, Ack: 2391, Len: 0



CS338 Sandbox

cs338.jeffondich.com

Kali Linux Kali Tools Kali Docs Kali Forums Kali NetHandler Exploit DB Google Hacking DB OJSec

Fun with security, or maybe insecurity

This page should be the page you retrieve for the "Getting started with Wireshark" assignment. Here's my head, as advertised:



Hopefully this illustrates some of the insecurity with Nginx's basic authentication and provides a good reason to use more secure methods of authentication. I am still unclear as to why there are some communications being secured with TLS but the password is not one of them. The application data sent over is always 78 bytes, so perhaps more investigation can be done into what is being sent.