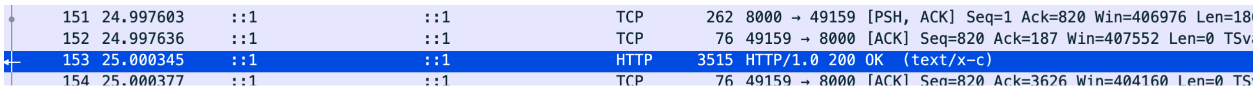
1. I accomplished this task by using the Python http server class.
Specifically, I ran python3 -m http.server 8000 in a folder on my computer
which contains cFS which is NASA's core flight system.

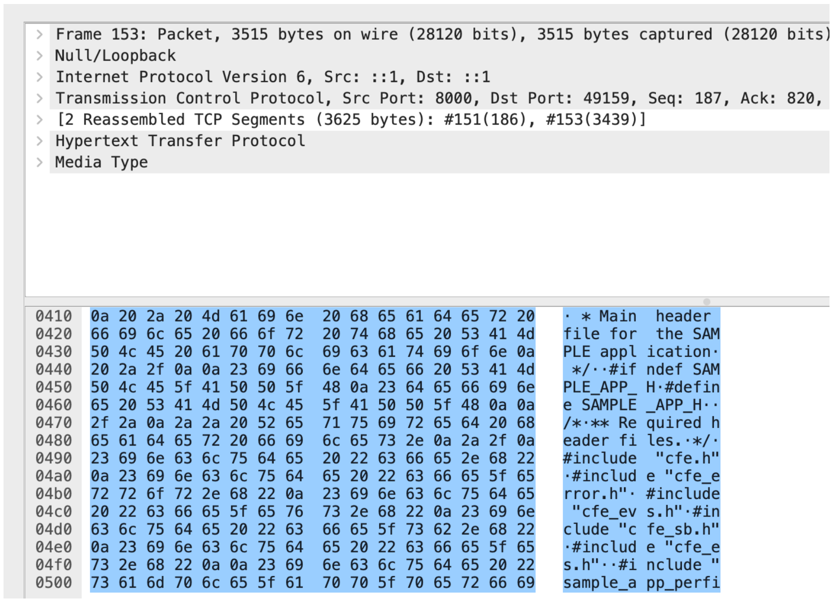
2. HTTP is not secure because it is unencrypted. An eavesdropper can see http requests
because they are sent across the internet between the client and the server. HTTP alone
is unencrypted though so any person looking can see the request and the response. If
people are sending sensitive data this is quite bad.

In order to accomplish the task, I downloaded wireshark and chmodbpf as this was in the
installer instructions. I then set it to specifically capture loopbakc lo0 traffic rather
than the default mode it is in, en0. Then I set it to specifically use tcp port 8000 because
this is what I set for my webserver. Then I started recording with the blue fin and started the
webserver to begin capturing the traffic. Then I clicked around entering different files in the
folder I was accessing.

Using wireshark I was able to read the request and response between my local client and server
double clicking on the request and reading the pop up window. An example of reading the
response is shown below.



```
151  24.997603     ::1          ::1          TCP     262  8000 → 49159 [PSH, ACK] Seq=1 Ack=820 Win=406976 Len=18
152  24.997636     ::1          ::1          TCP      76  49159 → 8000 [ACK] Seq=820 Ack=187 Win=407552 Len=0 TSv
153  25.000345     ::1          ::1          HTTP   3515  HTTP/1.0 200 OK  (text/x-c)
154  25.000377     ::1          ::1          TCP      76  49159 → 8000 [ACK] Seq=820 Ack=3626 Win=404160 Len=0 TS
```



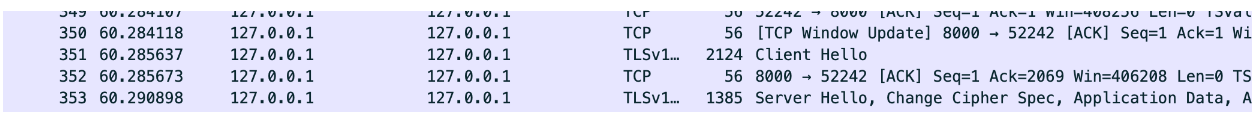
```
> Frame 153: Packet, 3515 bytes on wire (28120 bits), 3515 bytes captured (28120 bits)
> Null/Loopback
> Internet Protocol Version 6, Src: ::1, Dst: ::1
> Transmission Control Protocol, Src Port: 8000, Dst Port: 49159, Seq: 187, Ack: 820,
> [2 Reassembled TCP Segments (3625 bytes): #151(186), #153(3439)]
> Hypertext Transfer Protocol
> Media Type
```

```
0410  0a 20 2a 20 4d 61 69 6e  20 68 65 61 64 65 72 20   · * Main  header
0420  66 69 6c 65 20 66 6f 72  20 74 68 65 20 53 41 4d   file for  the SAM
0430  50 4c 45 20 61 70 70 6c  69 63 61 74 69 6f 6e 0a   PLE appl ication·
0440  20 2a 2f 0a 0a 23 69 66  6e 64 65 66 20 53 41 4d    */··#if ndef SAM
0450  50 4c 45 5f 41 50 50 5f  48 0a 23 64 65 66 69 6e   PLE_APP_ H·#defin
0460  65 20 53 41 4d 50 4c 45  5f 41 50 50 5f 48 0a 0a   e SAMPLE _APP_H··
0470  2f 2a 0a 2a 2a 20 52 65  71 75 69 72 65 64 20 68   /*·** Re quired h
0480  65 61 64 65 72 20 66 69  6c 65 73 2e 0a 2a 2f 0a   eader fi les.·*/·
0490  23 69 6e 63 6c 75 64 65  20 22 63 66 65 2e 68 22   #include  "cfe.h"
04a0  0a 23 69 6e 63 6c 75 64  65 20 22 63 66 65 5f 65   ·#includ e "cfe_e
04b0  72 72 6f 72 2e 68 22 0a  23 69 6e 63 6c 75 64 65   rror.h"· #include
04c0  20 22 63 66 65 5f 65 76  73 2e 68 22 0a 23 69 6e    "cfe_ev s.h"·#in
04d0  63 6c 75 64 65 20 22 63  66 65 5f 73 62 2e 68 22   clude "c fe_sb.h"
04e0  0a 23 69 6e 63 6c 75 64  65 20 22 63 66 65 5f 65   ·#includ e "cfe_e
04f0  73 2e 68 22 0a 0a 23 69  6e 63 6c 75 64 65 20 22   s.h"··#i nclude "
0500  73 61 6d 70 6c 65 5f 61  70 70 5f 70 65 72 66 69   sample_a pp_perfi
```

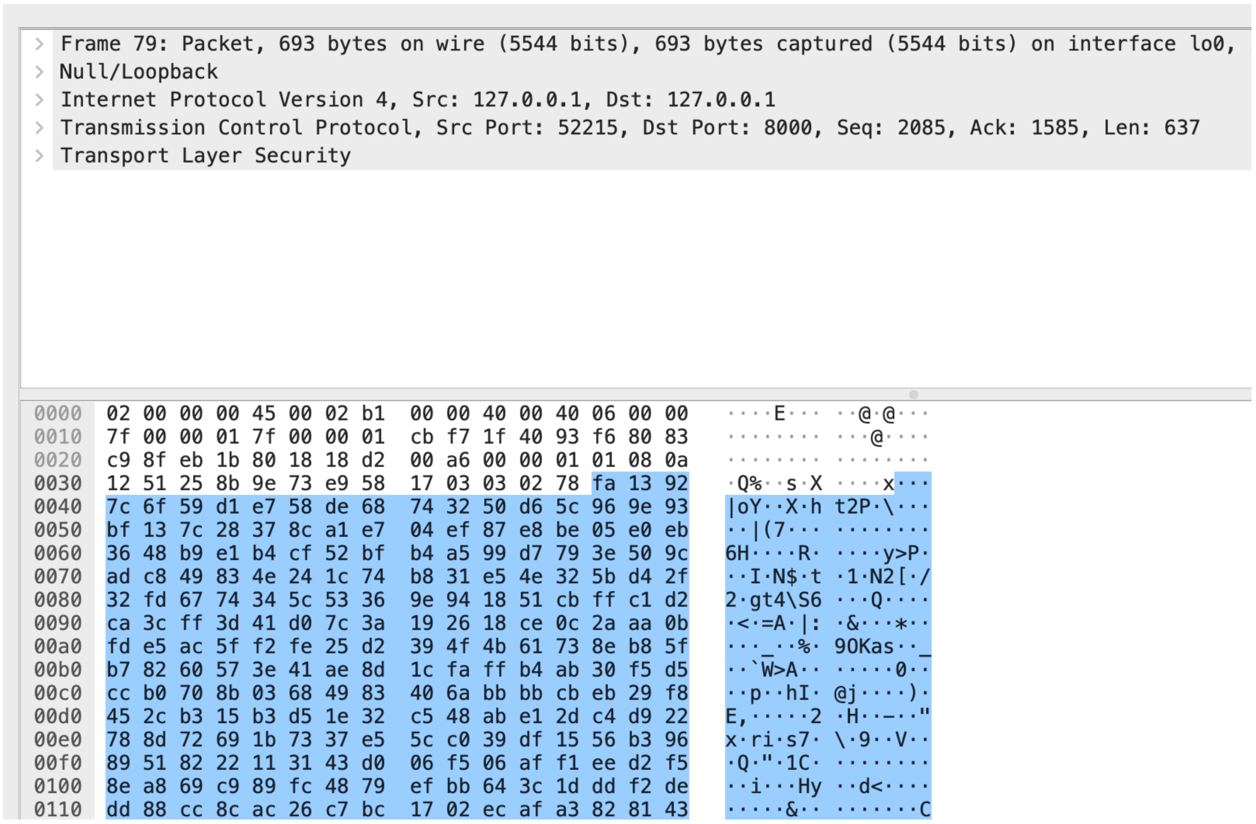
3. I cannot obtain a certificate from a certificate authority because this is a locally hosted web server and certificate authorities will only issue certificates to publicly available domains.

I created a self signed certificate for my web server first by installing openssl. I then created a private key using openssl and then a certificate for my web server using openssl and the private key I had just generated. Then I used flask to create a web server in the same folder because I could no longer use the python http server class.

I then used the lo0 tcp port 8000 wireshark settings and captured traffic. This time I could not see the specific data because it was encrypted so it looked like random characters as seen below. Additionally instead of seeing HTTP GET and OK which I got with HTTPS. The traces now had TLSv1.3 as the type which is because it is now encrypted.



```
349 60.284107   127.0.0.1    127.0.0.1    TCP       56 52242 → 8000 [ACK] Seq=1 Ack=1 Win=408256 Len=0 TSval
350 60.284118   127.0.0.1    127.0.0.1    TCP       56 [TCP Window Update] 8000 → 52242 [ACK] Seq=1 Ack=1 Wi
351 60.285637   127.0.0.1    127.0.0.1    TLSv1…  2124 Client Hello
352 60.285673   127.0.0.1    127.0.0.1    TCP       56 8000 → 52242 [ACK] Seq=1 Ack=2069 Win=406208 Len=0 TS
353 60.290898   127.0.0.1    127.0.0.1    TLSv1…  1385 Server Hello, Change Cipher Spec, Application Data, A
```



```
> Frame 79: Packet, 693 bytes on wire (5544 bits), 693 bytes captured (5544 bits) on interface lo0,
> Null/Loopback
> Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
> Transmission Control Protocol, Src Port: 52215, Dst Port: 8000, Seq: 2085, Ack: 1585, Len: 637
> Transport Layer Security
```

```
0000  02 00 00 00 45 00 02 b1  00 00 40 00 40 06 00 00   ····E···  ··@·@···
0010  7f 00 00 01 7f 00 00 01  cb f7 1f 40 93 f6 80 83   ········  ···@····
0020  c9 8f eb 1b 80 18 18 d2  00 a6 00 00 01 01 08 0a   ········  ········
0030  12 51 25 8b 9e 73 e9 58  17 03 03 02 78 fa 13 92   ·Q%··s·X  ····x···
0040  7c 6f 59 d1 e7 58 de 68  74 32 50 d6 5c 96 9e 93   |oY··X·h  t2P·\···
0050  bf 13 7c 28 37 8c a1 e7  04 ef 87 e8 be 05 e0 eb   ··|(7···  ········
0060  36 48 b9 e1 b4 cf 52 bf  b4 a5 99 d7 79 3e 50 9c   6H····R·  ····y>P·
0070  ad c8 49 83 4e 24 1c 74  b8 31 e5 4e 32 5b d4 2f   ··I·N$·t  ·1·N2[·/
0080  32 fd 67 74 34 5c 53 36  9e 94 18 51 cb ff c1 d2   2·gt4\S6  ···Q····
0090  ca 3c ff 3d 41 d0 7c 3a  19 26 18 ce 0c 2a aa 0b   ·<·=A·|:  ·&···*··
00a0  fd e5 ac 5f f2 fe 25 d2  39 4f 4b 61 73 8e b8 5f   ···_··%·  9OKas··_
00b0  b7 82 60 57 3e 41 ae 8d  1c fa ff b4 ab 30 f5 d5   ··`W>A··  ·····0··
00c0  cc b0 70 8b 03 68 49 83  40 6a bb bb cb eb 29 f8   ··p··hI·  @j····)·
00d0  45 2c b3 15 b3 d5 1e 32  c5 48 ab e1 2d c4 d9 22   E,·····2  ·H··-··"
00e0  78 8d 72 69 1b 73 37 e5  5c c0 39 df 15 56 b3 96   x·ri·s7·  \·9··V··
00f0  89 51 82 22 11 31 43 d0  06 f5 06 af f1 ee d2 f5   ·Q"·1C·  ········
0100  8e a8 69 c9 89 fc 48 79  ef bb 64 3c 1d dd f2 de   ··i···Hy  ··d<····
0110  dd 88 cc 8c ac 26 c7 bc  17 02 ec af a3 82 81 43   ·····&··  ·······C
```

Sources:

I consulted this website for HTTP vs HTTPS information
https://www.cloudflare.com/learning/ssl/why-is-http-not-secure/

I used ChatGPT in this assignment and my list of **prompts** and the prompting methodology is below.

“I need to host a local web server”

I used this prompt to see what the initial instructions it suggested were to build a web server. I then decided to use the python http server class because this was also listed in the assignment.

“how would I use wireshark to capture traffic between my local webserver and client”

I used this prompt to get familiar with wireshark and how to use it because it was not something I had done before and I wanted my packet traces to be accurate.

“i have the app not cli”

I used this prompt because it gave instructions for using the cli instead of the app which I had downloaded onto my computer

“it says I don't have permission to capture on local interfaces and I need to install chmodbpf but i already did that”

I used this prompt because I was having trouble getting the permission to look at local interface traffic despite installing chmodbpf. The issue that I found was that I needed to restart wireshark after installing chmodbpf and I hadn’t done that

“ok I have permission now but I'm not seeing any traffic”

I used this to get the chatbot to move on from suggesting me fixes to the chmodbpf issue and to instead help me find a way to see traffic as I wasn’t seeing any with the tcp 8000 filter I had.

“ok I got it now, now what does it mean to submit "a packet trace of HTTP traffic"”

I asked this because I realized I was supposed to use lo0 instead of en0. So then I had a packet trace and I wanted to make sure I had the right format.

“now I need to generate an ssl certificate for my web server”

I used this prompt because I wanted to get instructions for generating the ssl certificate. To make my server https.

“ython3 -m http.server 8000 0.0.0.0 --directory . \ --ssl-key securityserver.key --ssl-cert securityserver.crt usage: server.py [-h] [--cgi] [-b ADDRESS] [-d DIRECTORY] [-p VERSION] [port] server.py: error: unrecognized arguments: 0.0.0.0 --ssl-key securityserver.key --ssl-cert securityserver.crt”

I used this prompt because I was confused at first why I couldn’t assign the certificate this way and then realized the http class wouldn’t work because it didn’t have ssl.

“will the flask app still allow me to access the stuff in the folder like I was doing before”

I used this prompt because I wanted to see how I could still generate traffic on the webserver using flask instead of http server.

“I'm not seeing any http traffic on wireshark now, thats normal right?”

I used this prompt because I wanted to make sure seeing TLS traffic instead of http traffic was correct.

“when I'm using the http and not the https how can I see the contents of what is being fetched on wireshark”

I used this prompt because I wanted to see the specific data like the contents of the file rather than just the file I was accessing on the web server and I wasn’t sure how to do that.

“why do I need to self sign the certificate, can I not use a CA for a local web server?”

I used this prompt because I wanted to confirm that my reasoning was correct for why I could not use a certificate authority. This reinforced my learning in the areas of certificate authorities.