Public Key Infrastructure Lab

1.) Host a local server

To host a server, I first created a directory containing a PNG image and a text file. While in the directory in my terminal, I used the command “

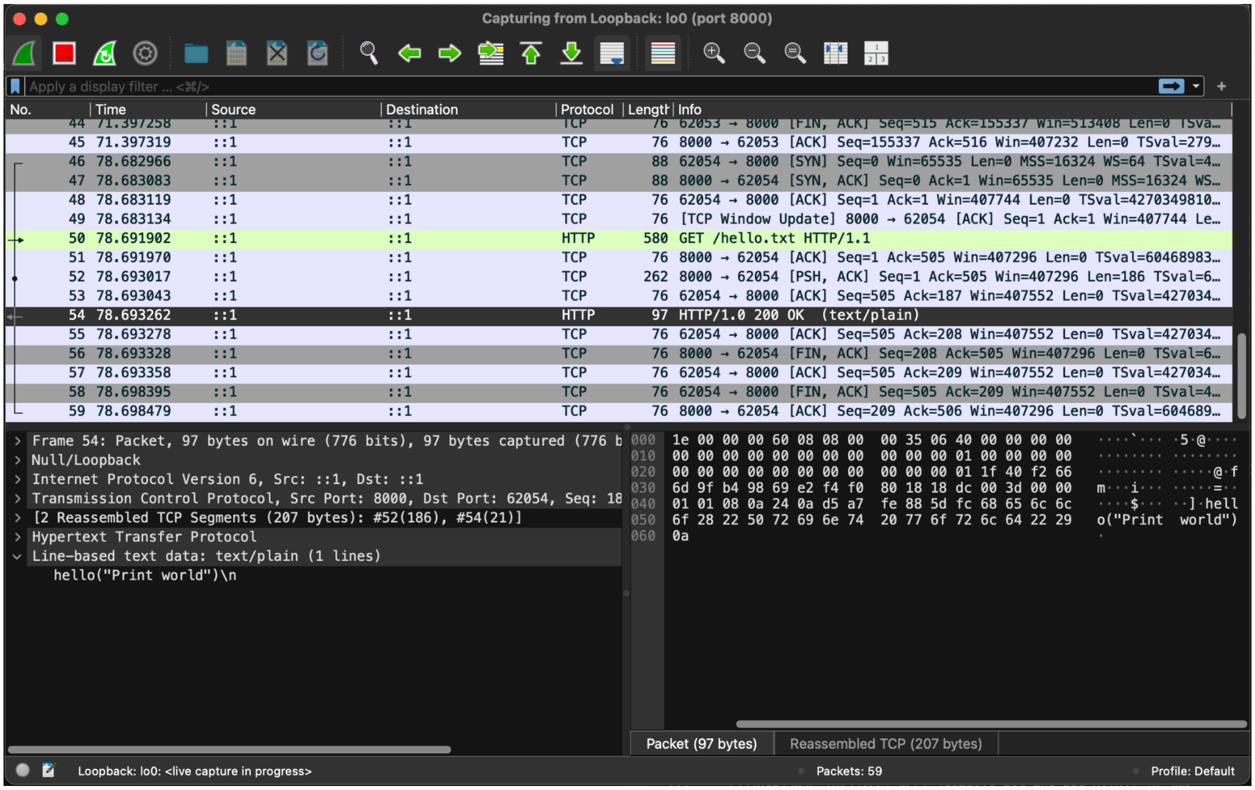
2.) Identify why HTTP is not secure

Data sent from an HTTP server is sent unencrypted, as plain text. As such, an eavesdropper present at any point between the client and web server that’s capable of tracking traffic can see exactly what resources are being sent between the two parties.

With wireshark, I was able to track the numerous communications related to TCP sent between the two parties. The majority of these were acknowledgements between systems. Wireshark was also able to track messages through HTTP, which are the text and image files that the client requested from the website.

When I requested the text file using a local client, a web browser, and my local server sent them to my client, Webshark tracked what was communicated. Because it was sent in plain text, Wireshark could see all of the bytes being sent in the form of bytes.

This is an example of traffic I was able to eavesdrop on using wireshark:



Capturing from Loopback: lo0 (port 8000)
Apply a display filter ... <⌘/>

```
No.  Time       Source  Destination  Protocol  Length  Info
 44  71.397258  ::1     ::1          TCP        76  62053 → 8000 [FIN, ACK] Seq=515 Ack=155337 Win=513408 Len=0 TSva…
 45  71.397319  ::1     ::1          TCP        76  8000 → 62053 [ACK] Seq=155337 Ack=516 Win=407232 Len=0 TSval=279…
 46  78.682966  ::1     ::1          TCP        88  62054 → 8000 [SYN] Seq=0 Win=65535 Len=0 MSS=16324 WS=64 TSval=4…
 47  78.683083  ::1     ::1          TCP        88  8000 → 62054 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=16324 WS…
 48  78.683119  ::1     ::1          TCP        76  62054 → 8000 [ACK] Seq=1 Ack=1 Win=407744 Len=0 TSval=4270349810…
 49  78.683134  ::1     ::1          TCP        76  [TCP Window Update] 8000 → 62054 [ACK] Seq=1 Ack=1 Win=407744 Le…
 50  78.691902  ::1     ::1          HTTP      580  GET /hello.txt HTTP/1.1
 51  78.691970  ::1     ::1          TCP        76  8000 → 62054 [ACK] Seq=1 Ack=505 Win=407296 Len=0 TSval=60468983…
 52  78.693017  ::1     ::1          TCP       262  8000 → 62054 [PSH, ACK] Seq=1 Ack=505 Win=407296 Len=186 TSval=6…
 53  78.693043  ::1     ::1          TCP        76  62054 → 8000 [ACK] Seq=505 Ack=187 Win=407552 Len=0 TSval=427034…
 54  78.693262  ::1     ::1          HTTP       97  HTTP/1.0 200 OK  (text/plain)
 55  78.693278  ::1     ::1          TCP        76  62054 → 8000 [ACK] Seq=505 Ack=208 Win=407552 Len=0 TSval=427034…
 56  78.693328  ::1     ::1          TCP        76  8000 → 62054 [FIN, ACK] Seq=208 Ack=505 Win=407296 Len=0 TSval=6…
 57  78.693358  ::1     ::1          TCP        76  62054 → 8000 [ACK] Seq=505 Ack=209 Win=407552 Len=0 TSval=427034…
 58  78.698395  ::1     ::1          TCP        76  62054 → 8000 [FIN, ACK] Seq=505 Ack=209 Win=407552 Len=0 TSval=4…
 59  78.698479  ::1     ::1          TCP        76  8000 → 62054 [ACK] Seq=209 Ack=506 Win=407296 Len=0 TSval=604689…
```

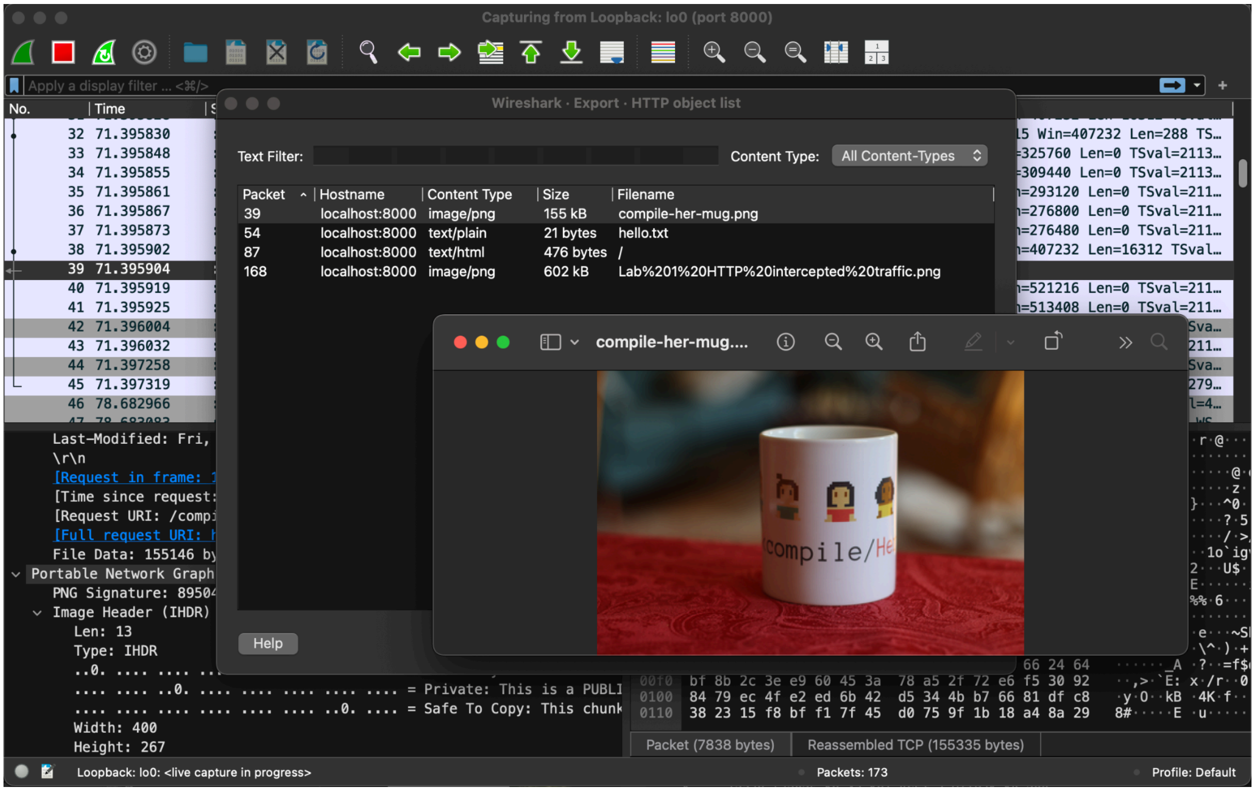
```
> Frame 54: Packet, 97 bytes on wire (776 bits), 97 bytes captured (776 b
> Null/Loopback
> Internet Protocol Version 6, Src: ::1, Dst: ::1
> Transmission Control Protocol, Src Port: 8000, Dst Port: 62054, Seq: 18
> [2 Reassembled TCP Segments (207 bytes): #52(186), #54(21)]
> Hypertext Transfer Protocol
v Line-based text data: text/plain (1 lines)
    hello("Print world")\n
```

```
000  1e 00 00 00 60 08 08 00  00 35 06 40 00 00 00 00   ····`··· ·5·@····
010  00 00 00 00 00 00 00 00  00 00 00 01 00 00 00 00   ········ ········
020  00 00 00 00 00 00 00 00  00 00 00 01 1f 40 f2 66   ········ ·····@·f
030  6d 9f b4 98 69 e2 f4 f0  80 18 18 dc 00 3d 00 00   m···i··· ·····=··
040  01 01 08 0a 24 0a d5 a7  fe 88 5d fc 68 65 6c 6c   ····$··· ··]·hell
050  6f 28 22 50 72 69 6e 74  20 77 6f 72 6c 64 22 29   o("Print  world")
060  0a                                                 ·
```

Packet (97 bytes) | Reassembled TCP (207 bytes)

Loopback: lo0: <live capture in progress>   Packets: 59   Profile: Default

I was also able to eavesdrop and capture a PNG image sent to the client:



Capturing from Loopback: lo0 (port 8000)
Apply a display filter ... <⌘/>
Wireshark · Export · HTTP object list
Help
compile-her-mug....
compile/He

Text Filter: Content Type: All Content-Types

Packet	Hostname	Content Type	Size	Filename
39	localhost:8000	image/png	155 kB	compile-her-mug.png
54	localhost:8000	text/plain	21 bytes	hello.txt
87	localhost:8000	text/html	476 bytes	/
168	localhost:8000	image/png	602 kB	Lab%201%20HTTP%20intercepted%20traffic.png

```
No.  Time
 32  71.395830     5 Win=407232 Len=288 TS…
 33  71.395848     =325760 Len=0 TSval=2113…
 34  71.395855     =309440 Len=0 TSval=2113…
 35  71.395861     n=293120 Len=0 TSval=211…
 36  71.395867     n=276800 Len=0 TSval=211…
 37  71.395873     n=276480 Len=0 TSval=211…
 38  71.395902     n=407232 Len=16312 TSval…
 39  71.395904
 40  71.395919     n=521216 Len=0 TSval=211…
 41  71.395925     n=513408 Len=0 TSval=211…
 42  71.396004     Sva…
 43  71.396032     211…
 44  71.397258     Sva…
 45  71.397319     279…
 46  78.682966     l=4…
```

```
    Last-Modified: Fri,
    \r\n
    [Request in frame: 1
    [Time since request:
    [Request URI: /compi
    [Full request URI: h
    File Data: 155146 by
v Portable Network Graph
    PNG Signature: 89504
  v Image Header (IHDR)
      Len: 13
      Type: IHDR
      ..0. .... .... ...
      .... .... ..0. .... .... .... .... .... = Private: This is a PUBLI
      .... .... .... .... .... .... ..0. .... = Safe To Copy: This chunk
      Width: 400
      Height: 267
```

```
                                               66 24 64   ······_A ·?··=f$
00f0  bf 8b 2c 3e e9 60 45 3a  78 a5 2f 72 e6 f5 30 92   ··,>·`E: x·/r··0
0100  84 79 ec 4f e2 ed 6b 42  d5 34 4b b7 66 81 df c8   ·y·O··kB ·4K·f··
0110  38 23 15 f8 bf f1 7f 45  d0 75 9f 1b 18 a4 8a 29   8#·····E ·u·····
```

Packet (7838 bytes) | Reassembled TCP (155335 bytes)

Loopback: lo0: <live capture in progress>   Packets: 173   Profile: Default

3.) Create a self-signed certificate and upgrade your web server to HTTPS

a. My local server does not have a unique domain, and can be accessed by going to localhost:8000 in a web browser. Certificates bind a domain name to a public key to validate the identity of the server. Because the identity of the owner of localhost:8000 changes depending on whose hosting the server, it wouldn't make sense for a CA to issue a certificate that essentially says "this domain is associated with this entity."

For this task, I downloaded mkcert and generated a certificate for localhost. Now, when I use wireshark to track traffic, I find that the data being sent is encrypted. It wasn't even clear what type of data was being sent due to the encryption, I was not able to track whether data was png or txt files. I can still track the acknowledgements between the client and my local server, however.

AI Acknowledgement:

I utilized an AI tool to help guide me in using Wireshark to track network traffic, and for help generating a self-signed certificate and running an HTTPS server using it.