

## Public Key Infrastructure

Veronica Sokoloff Cortes

Fall 2025

**Goal:** Show why HTTP is insecure (plaintext) and how HTTPS (TLS) fixes it (encrypted).

### Setup:

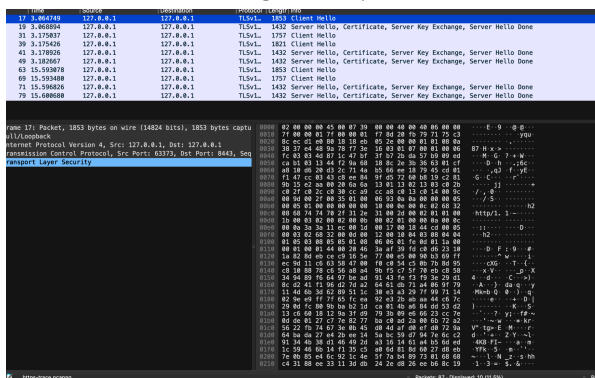
- OS: macOS (loopback 100)
- Tools: Python 3 (http.server), OpenSSL, Wireshark

1. **Host a local web server:** To host the local web server I used Python's built-in http.server. I started by making a working folder with an `index.html` so there was something to serve (`mkdir -p ~/http-demo && cd ~/http-demo`, then I wrote a basic "Hello, HTTP" page). Next, I started the server in the terminal and binded it to my own PC for safety (`python3 -m http.server 8000 --bind 127.0.0.1`). Terminal confirmed it was running, and I double checked by visiting `http://127.0.0.1:8000/`.

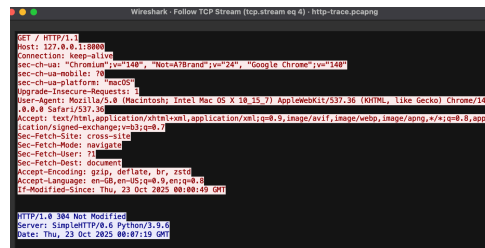
- ★ *ChatGBT told me that binding to 127.0.0.1 keeps the server local, nothing exposed to other devices, which made this the fastest, lowest-friction way to demonstrate an HTTP server without configuring nginx or Apache.*

2. **Identify why HTTP is not secure:** HTTP isn't secure because it sends everything unencrypted (requests, headers, and bodies) so literally anyone on the path could read it if they wanted to. An eavesdropper on the same network can "sniff" packets and see exactly which resources are fetched (`GET /index.html`), plus sensitive data in headers or the page/form content. To demonstrate this, I ran my local HTTP server and captured traffic on the loopback interface (100) with Wireshark using the http (or `tcp.port == 8000`) display filter. The capture clearly shows the plaintext request line, Host, other headers, and the `HTML` body in the bytes pane—proof that HTTP provides no confidentiality. (Screenshots included below.)

*TLS 1.3 handshake begins; only metadata visible*



*End-to-end HTTP dialogue is readable to an eavesdropper.*



*HTTP request is fully readable.*

Protocol	Length	Info
HTTP/...	114	POST /notifications/libraries/sync HTTP/1.1 ,
HTTP	792	GET / HTTP/1.1
HTTP	159	HTTP/1.0 304 Not Modified
HTTP/...	114	POST /notifications/libraries/sync HTTP/1.1 ,

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

- I found out that public certificate authorities only issue certificates for public DNS names after you prove domain control (via **DNS/HTTP** challenges). **localhost** and **127.0.0.1** are local-only identifiers (not publicly resolvable) and are explicitly disallowed by CA/B rules, so there's no way for me to get validated or issued a cert for my local dev server.

So, I instead generated a self-signed X.509 certificate with Subject Alternative Names for localhost, 127.0.0.1, and ::1, then added it to my macOS System Keychain as Always Trust so the browser would accept it. I restarted my server wrapped in TLS (Python http.server behind `ssl.SSLContext`) on port 8443 and verified I could load `https://127.0.0.1:8443/` without warnings. I recorded my traffic while loading the page and I was shocked to see you can read everything: the request (`GET /`), the Host header, and the page's HTML. In the HTTPS capture, I first saw a "hello" exchange between the browser and server to set up encryption, but after that all the packets just look like scrambled data, hard to read.

- HTTP → anyone watching the network can see the whole conversation.

```
Hypertext Transfer Protocol
> GET / HTTP/1.1\r\n
Host: 127.0.0.1:8000\r\n
Connection: keep-alive\r\n
sec-ch-ua: "Chromium";v="140", "Not=A?Brand";v="24", "Google Chrome";v=
sec-ch-ua-mobile: ?0\r\n
sec-ch-ua-platform: "macOS"\r\n
Upgrade-Insecure-Requests: 1\r\n
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avi
Sec-Fetch-Site: cross-site\r\n
Sec-Fetch-Mode: navigate\r\n
Sec-Fetch-User: ?1\r\n
Sec-Fetch-Dest: document\r\n
Accept-Encoding: gzip, deflate, br, zstd\r\n
Accept-Language: en-GB,en-US;q=0.9,en;q=0.8\r\n
If-Modified-Since: Thu, 23 Oct 2025 00:00:49 GMT\r\n
\r\n
[Response in frame: 37]
[Full request URI: http://127.0.0.1:8000/]
```

Wireshark shows the full request line, headers, and (in the response) the page body in clear text—confirming that HTTP provides no confidentiality.

- HTTPS → the same conversation is encrypted after the initial setup, so it's hidden.

```
(server contiguous streams: 1) 0030 38 37 c4 48 9a 78 f7 2e 15 03 01 07 00 01 00 06 87 H-x> .....
TCP payload (1797 bytes) 0040 fc 03 03 4d 87 1c 47 bf 3f b7 2b da 57 b0 09 ed ...M-x< 7+...W...
Transport Layer Security 0050 ca b1 03 13 44 f2 9a 68 18 8c 2e 3b 36 03 01 cf ...D-h...;6c...
[Stream index: 0] 0060 a8 10 d6 20 d3 2c 71 4a b5 66 ee 18 79 45 cd 01 ...qJ f-yE...
0070 f1 47 cc 03 43 c8 ee 84 9f d5 72 60 b0 19 c2 01 ...C...r...
0080 9b 15 e2 aa 00 20 6a 6a 13 01 13 02 13 03 c0 2b ...j...+
0090 c0 2f c0 2c c0 30 cc a9 cc a8 c0 13 c0 14 00 9c ...0...
00a0 00 00 00 2f 00 35 01 00 86 93 0a 00 00 00 05 .../S...
00b0 00 05 01 00 00 00 00 00 10 00 0e 00 0c 02 08 32 ...h2...
00c0 00 68 74 74 70 2f 31 2e 31 00 2d 00 02 01 01 00 http/1.1-...
00d0 1b 00 03 02 00 02 00 0b 00 02 01 00 00 0a 00 0c ...h2...
00e0 00 0a 3a 3a 11 ec 00 1d 00 17 00 16 44 cd 00 05 ...D...
00f0 00 03 02 68 32 00 0d 00 12 00 10 04 03 08 04 04 ...h2...
0100 01 05 03 08 05 01 00 06 06 01 fe 0d 01 1a 00 ...D...
0110 00 01 00 01 44 00 20 46 3a af 30 fe c0 d6 23 10 ...D F :9...#
0120 1a 82 8d eb ce c9 16 5e 77 00 e5 00 90 b3 69 ff ...W...1...
0130 ec 9d 11 c6 63 58 47 00 fe c0 54 c5 0b 7b 8d 95 ...XG...T...E...
0140 c5 10 82 78 c6 36 a8 a4 9b f5 cf 51 70 eb c8 38 ...xV...D...X
0150 34 94 89 f6 64 97 be ad 91 43 fe f3 f9 3e 29 d1 4...d...C...>...
0160 8c d2 41 f1 96 d2 7d a2 64 61 d9 71 a4 06 9f 79 ...A...> da q...y
0170 11 4d 6b 3d 62 09 51 1c 38 e3 a3 20 7f 99 71 14 ...Meb D 0 0...q
0180 02 9e e9 ff 7f 65 fc ea 92 e3 2b ab aa 44 c6 7c ...e...+...D...J
0190 29 0d fc 00 9b ba b2 1d ca 01 4b a6 84 dd 53 d2 ...)...K...S...
01a0 13 c6 60 10 12 0a 2f 09 79 3a 00 e6 66 23 c6 7e ......F...y...f...a...
01b0 0d de 01 27 c7 7e 82 77 ba c0 ad 2a 00 6b 72 a2 ......W...+...K...r...
01c0 56 22 fb 74 67 3e 0b 45 d0 4d af d0 ef d0 72 9a ...V...tq> E .M...r...
01d0 64 ba d0 27 6b 2b ee 14 5a bc 59 d7 94 7e c6 c2 ...d...x...2...Y...n...
01e0 91 34 4b 38 01 46 49 2d a3 16 14 61 a4 b5 6d ed ...4K8-FI...>...a...m...
Type of handshake message (0: handshake.type), 1: byte Packets: 8
```

Browser begins the secure session & only the setup metadata is visible rest on left is unreadable

My screenshots show HTTP is readable and HTTPS is unreadable.