

Sean Ji and Alex Miller

## Security, Privacy, and Consumer Protection: Lab #1

### 1) Server Implementation

Please see ‘server.js’ to see how we implemented our local web server(s); the file defines two local servers, one serving content over http on port 7000 and the other over https on port 8000. It is built on Express for Node.js. Our servers both serve the file ‘index.html’ to local clients querying our server through a browser.

### 2) Why is HTTP not secure?

Source	Destination	Protocol	Length	Info
20 127.0.0.1	127.0.0.1	TCP	74	43898 -> 7000 [SYN] Seq=0 Win=65536 Len=0 MSS=65536 SACK_PERM=1 TSval=2112668986 TSecr=0 WS=128
55 127.0.0.1	127.0.0.1	TCP	74	74 7000 -> 43898 [SYN, ACK] Seq=9 Ack=1 Win=65483 Len=0 MSS=65495 SACK_PERM=1 TSval=2112668986 TSecr=2112668986
70 127.0.0.1	127.0.0.1	TCP	66	43898 -> 7000 [ACK] Seq=1 Ack=1 Win=65536 Len=0 TSval=2112668986 TSecr=2112668986
83 127.0.0.1	127.0.0.1	TCP	512	43898 -> 7000 [PSH, ACK] Seq=1 Ack=1 Win=65536 Len=446 TSval=2112668979 TSecr=2112668986 [TCP segment of w/o length 512]
21 127.0.0.1	127.0.0.1	TCP	66	7000 -> 43898 [ACK] Seq=1 Ack=447 Win=65152 Len=0 TSval=2112668979 TSecr=2112668979
10 127.0.0.1	127.0.0.1	TCP	834	7000 -> 43898 [PSH, ACK] Seq=1 Ack=447 Win=65536 Len=768 TSval=2112668983 TSecr=2112668979 [TCP segment of w/o length 834]
70 127.0.0.1	127.0.0.1	TCP	66	43898 -> 7000 [ACK] Seq=447 Ack=769 Win=64768 Len=0 TSval=2112668984 TSecr=2112668983
62 127.0.0.1	127.0.0.1	TCP	449	43898 -> 7000 [PSH, ACK] Seq=447 Ack=769 Win=65536 Len=383 TSval=2112669029 TSecr=2112668983 [TCP segment of w/o length 449]
14 127.0.0.1	127.0.0.1	TCP	66	7000 -> 43898 [ACK] Seq=769 Ack=830 Win=65280 Len=0 TSval=2112669029 TSecr=2112669029
70 127.0.0.1	127.0.0.1	TCP	488	7000 -> 43898 [PSH, ACK] Seq=769 Ack=830 Win=65536 Len=422 TSval=2112669031 TSecr=2112669029 [TCP segment of w/o length 488]
99 127.0.0.1	127.0.0.1	TCP	66	43898 -> 7000 [ACK] Seq=830 Ack=1191 Win=65152 Len=0 TSval=2112669031 TSecr=2112669031
56 127.0.0.1	127.0.0.1	TCP	66	7000 -> 43898 [FIN, ACK] Seq=1191 Ack=830 Win=65536 Len=0 TSval=2112674832 TSecr=2112669031
16 127.0.0.1	127.0.0.1	TCP	66	43898 -> 7000 [FIN, ACK] Seq=830 Ack=1192 Win=65536 Len=0 TSval=2112674832 TSecr=2112674832
.....0.. = Syn: Not set .....0.. = Fin: Not set [TCP Flags: .....AP...] Window size value: 512 [Calculated window size: 65536] [Window size scaling factor: 128] Checksum: 0x129 [unverified] [Checksum Status: Unverified] Urgent pointer: 0				
0000	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	45 00	.....E
0010	03 3a 18 2e 40 00 40 06	21 94 7f 00 00 01 7f 00	4 0 0	.....
0020	00 01 1b 58 ab 72 3a 3d	c6 cd 07 1b 20 c7 80 18	.....X:ri	.....
0030	02 00 01 29 00 00 01 01	00 0a 70 ec c5 37 70 ec	.....)	.....
0040	53 33 18 2e 40 00 40 06	21 94 7f 00 00 01 7f 00	4 0 0	.....
0050	4d 0d 0a 58 2d 50 6f 77	65 72 65 64 2d 42 79 3a	.....X:K-Pow	.....By:
0060	20 45 70 72 65 73 3a 20	62 0a 41 63 65 70 74	.....E	.....
0070	2d 52 61 6e 67 65 73 3a	20 62 79 74 65 73 0d 0a	.....R	.....
0080	43 61 63 68 65 2d 43 6f	6e 74 72 6f 6c 3a 20 70	.....C	.....
0090	15 62 6e 69 63 2c 20 6d	61 70 20 61 67 65 30 38	.....h	.....
00a0	0d 0a 4c 61 73 74 2d 4d	6f 64 69 6e 69 65 64 3a	.....L	.....
00b0	20 54 75 65 2c 20 32 35	20 4a 61 6e 20 32 30 32	Tue, 25 Jan 202	.....
00c0	52 20 61 30 3a 20 32 3a	30 36 20 47 4d 54 6d 0a	2 20:02:00 GMT	.....
00d0	45 54 61 67 3a 20 57 2f	22 31 63 3a 2d 31 37 65	Tag: W/ "i04-176	.....
00e0	39 32 36 65 64 66 36	22 0d 0a 43 6f 6e 74 65	9265edf6 "	.....Conte
00f0	6e 74 2d 54 79 65 3a	20 74 65 70 74 2f 68 74	.....t	.....
0100	0d 6c 30 20 63 68 61 72	73 65 74 3d 55 54 46 2d	.....l	.....
0110	38 0d 0a 43 6f 6e 74 65	6e 74 2d 4c 65 6e 67 74	.....C	.....
0120	08 3a 20 34 35 32 0d 0a	44 61 74 65 3a 20 57 65	.....v	.....
0130	04 2c 20 32 36 20 4a 61	6e 20 32 30 32 32 20 32	.....4	.....
0140	33 3a 31 37 3a 34 31 20	47 4d 54 0d 0a 43 6f 6e	3:17:41 GMT-Con	.....
0150	6e 65 63 74 6f 6e 3a	20 0b 65 65 70 2d 61 65	.....w	.....
0160	09 76 65 6d 0a 40 65 65	70 2d 41 6c 69 76 65 3a	.....ive	.....
0170	20 74 69 6d 65 6f 75 74	3d 35 6d 0a 0d 0a 3c 68	.....t	.....
0180	74 6d 6c 3e 69 65 63 65	65 61 64 3a 6a 69 69 6a	.....t	.....
0190	61 62 31 0a 09 3c 2f 68	69 61 64 3a 0a 0a 09 3c	.....h	.....
01a0	62 6f 64 79 20 69 64 3d	27 64 65 73 63 27 3e 0a	.....b	.....
01b0	39 3c 2f 62 6f 6f 6f 6f	6a 69 63 73 63 72 69	.....</body>	.....
01c0	70 74 3a 0a 09 69 69 66	20 28 6c 6f 63 61 74 69	.....t	.....
01d0	6f 6e 2e 70 72 6f 74 6f	63 6f 6c 20 21 3d 3d 29	.....n	.....
01e0	27 69 74 74 70 73 2f 27	20 70 0a 09 69 69 64	.....("https")	.....
01f0	6f 63 75 6d 65 6e 74 2e	67 65 74 45 6c 65 6d 65	.....t	.....
0200	6e 74 42 79 49 64 28 22	64 65 73 63 22 29 2e 69	.....n	.....
0210	6e 65 72 4e 14 4d 4e	45 3d 20 27 3c 68 31 3a	.....n	.....
0220	54 68 69 73 20 69 73 20	61 6e 20 49 4e 45 43 43	.....t	.....
0230	55 52 45 20 63 6f 6e 6e	65 63 74 69 6f 6e 3c 2f	.....n	.....
0240	69 31 3e 27 30 0a 09 69	70 20 05 6c 73 65 20 69	.....t	.....
0250	66 20 28 6c 6f 63 61 74	69 6f 6e 2e 70 72 6f 74	.....r	.....
0260	6f 63 6f 6c 2d 3d 3d 3d	20 27 68 74 74 70 73 3a	.....s	.....

Web-based and textual sent over HTTP is unencrypted. This means that an intermediary observer of HTTP network traffic between a client and a sever (the Man-in-the-Middle, so to speak) is able to both intercept IP packets sent between the client and server, and read them without difficulty. In the above photo, you can see the what we were able to capture when snooping on a local connection between a browser and our node server; the web-content of ‘index.html’ is plainly viewable in one of the packets we intercepted.

This suggests that HTTP an insecure protocol, as an unauthorized third party placed between a client and server (an ISP, someone on a shared Wi-Fi network, a government agency, etc.) is able to see personal or otherwise confidential information being sent between a client and a server over HTTP, as well as alter it (if they control a node in the IP path of the packets, such as a router or network switch). Additionally, besides not encrypting web-content, HTTP includes a number of sensitive data about a

client's machine configuration in headers, including client OS, installed fonts, browser version, etc., all of which can be used to identify and track clients across the web.

Please see 'insecure.pcapng' for our HTTP packet trace.

### 3) Upgrading to HTTPS

We generated a self-signed SSL certificate using 'openssl' with the following options:

```
openssl req -nodes -new -x509 -keyout server.key -out server.cert
```

Certificate Authorities don't sign certificates for 'localhost' servers; the domain is not considered to be 'owned' by anyone, so there's no reasonable way to verify ownership/legitimacy of any 'localhost' server.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	127.0.0.1	127.0.0.1	TCP	74	37225 → 8000 [SYN] Seq=0 Win=65535 Len=0 MSS=65535 SACK_PERM=1 TSval=2111778149 TSecr=0 WS=128
2	0.000022562	127.0.0.1	127.0.0.1	TCP	74	8000 → 37225 [SYN, ACK] Seq=0 Ack=1 Win=65483 Len=0 MSS=65495 SACK_PERM=1 TSval=2111778149 TSecr=2111778149
3	0.000042190	127.0.0.1	127.0.0.1	TCP	66	37225 → 8000 [ACK] Seq=1 Ack=1 Win=65536 Len=0 TSval=2111778149 TSecr=2111778149
4	0.000114779	127.0.0.1	127.0.0.1	TLSv1.3	583	Client Hello
5	0.000140457	127.0.0.1	127.0.0.1	TCP	66	8000 → 37225 [ACK] Seq=1 Ack=518 Win=65024 Len=0 TSval=2111778155 TSecr=2111778155
6	0.000170536	127.0.0.1	127.0.0.1	TLSv1.3	1636	Server Hello, Change Cipher Spec, Application Data, Application Data, Application Data
7	0.000185280	127.0.0.1	127.0.0.1	TCP	66	37225 → 8000 [ACK] Seq=518 Ack=1571 Win=64128 Len=0 TSval=2111778156 TSecr=2111778156
8	0.000211800	127.0.0.1	127.0.0.1	TLSv1.3	148	Change Cipher Spec, Application Data
9	0.000231151	127.0.0.1	127.0.0.1	TCP	66	8000 → 37225 [ACK] Seq=1571 Ack=598 Win=65536 Len=0 TSval=2111778158 TSecr=2111778158
10	0.000238824	127.0.0.1	127.0.0.1	TLSv1.3	538	Application Data
11	0.000238665	127.0.0.1	127.0.0.1	TCP	66	8000 → 37225 [ACK] Seq=1571 Ack=1870 Win=65152 Len=0 TSval=2111778158 TSecr=2111778158
12	0.000242221	127.0.0.1	127.0.0.1	TLSv1.3	640	Application Data, Application Data
13	0.000032298	127.0.0.1	127.0.0.1	TCP	66	37225 → 8000 [ACK] Seq=1070 Ack=2145 Win=65024 Len=0 TSval=2111778158 TSecr=2111778158
14	0.011032658	127.0.0.1	127.0.0.1	TLSv1.3	896	Application Data
15	0.011041755	127.0.0.1	127.0.0.1	TCP	66	37225 → 8000 [ACK] Seq=1070 Ack=2935 Win=64256 Len=0 TSval=2111778160 TSecr=2111778160
16	0.229022443	127.0.0.1	127.0.0.1	TLSv1.3	476	Application Data
17	0.229036350	127.0.0.1	127.0.0.1	TCP	66	8000 → 37225 [ACK] Seq=2935 Ack=1480 Win=65152 Len=0 TSval=2111778379 TSecr=2111778379
18	0.231063886	127.0.0.1	127.0.0.1	TLSv1.3	510	Application Data
19	0.231076278	127.0.0.1	127.0.0.1	TCP	66	37225 → 8000 [ACK] Seq=1480 Ack=3379 Win=65152 Len=0 TSval=2111778380 TSecr=2111778380
20	1.783058488	127.0.0.1	127.0.0.53	DNS	89	Standard query 0x6e61 A api.dropboxapi.com OPT
21	1.783071584	127.0.0.1	127.0.0.53	DNS	89	Standard query response 0x6e61 A api.dropboxapi.com CNAME api.dropbox.com CNAME api-env.dropbox-dns.c
22	1.895429519	127.0.0.53	127.0.0.1	DNS	165	Standard query response 0x6e61 A api.dropboxapi.com CNAME api-env.dropbox-dns.c

HTTPS differs from HTTP in two vital respects. Firstly, when a client first connects to a server over HTTPS, they carry a key-exchange (also called a TLS handshake) after the initial 3-way handshake. In this key exchange, a server sends the client its certificate containing its public key, which the client can verify against their list of known CAs. This qualifies the second difference between HTTP and HTTPS, which is that HTTPS traffic (web-based content and data, specifically) sent between a client is encrypted (asymmetrically). This makes it such that a Man-in-the-middle (MitM) is unable to *directly* snoop on traffic between a client on a server by intercepting IP packets.

You can observe in the above photo that, while we were able to see the content of 'index.html' in our HTTP trace, the corresponding packet of our HTTPS trace contains no readily understandable or exploitable content, data, or header information. Note our use of the word 'corresponding;' a MitM *can* use contextual data from other features of a client's HTTPS connection with a server in order to piece together the nature of their internet use. For example, a MitM could look at the amount of data sent by a server in response to a client's web request in order to distinguish whether a client requested one page over another.

However, a MitM is now not able to forge traffic between a client and a server, because HTTPS utilizes asymmetric encryption in order to ensure packet integrity.

Please see 'secure.pcapng' for our HTTP packet trace.

Sources:

<https://comodossllstore.com/resources/ssl-certificate-for-localhost/>

<https://letsencrypt.org/docs/certificates-for-localhost/>

<https://flaviocopes.com/express-https-self-signed-certificate/>