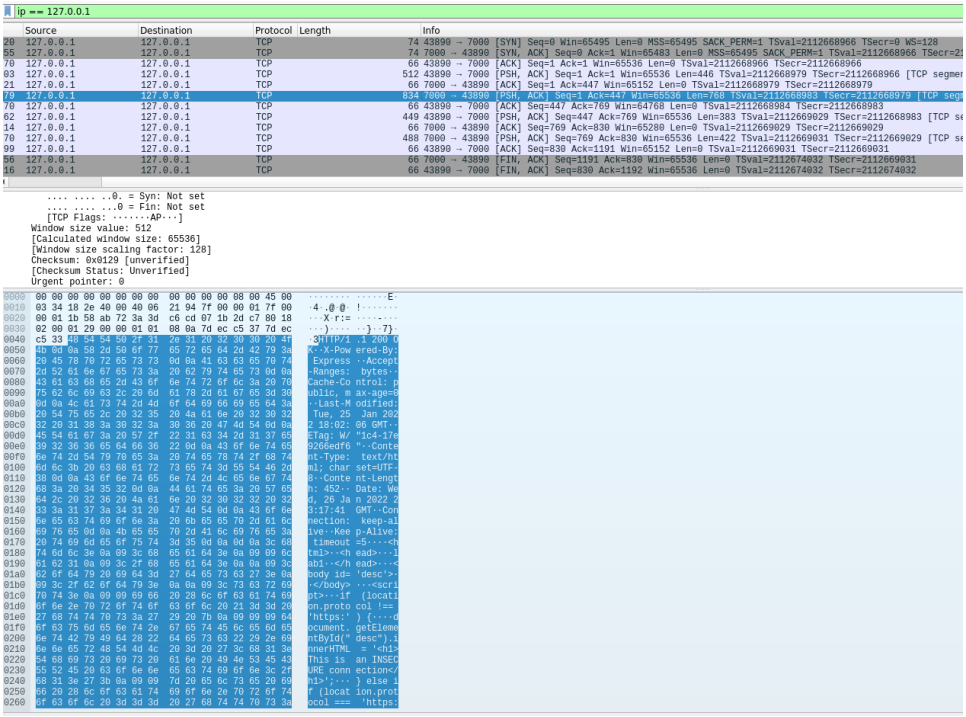
Sean Ji and Alex Miller

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?



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.

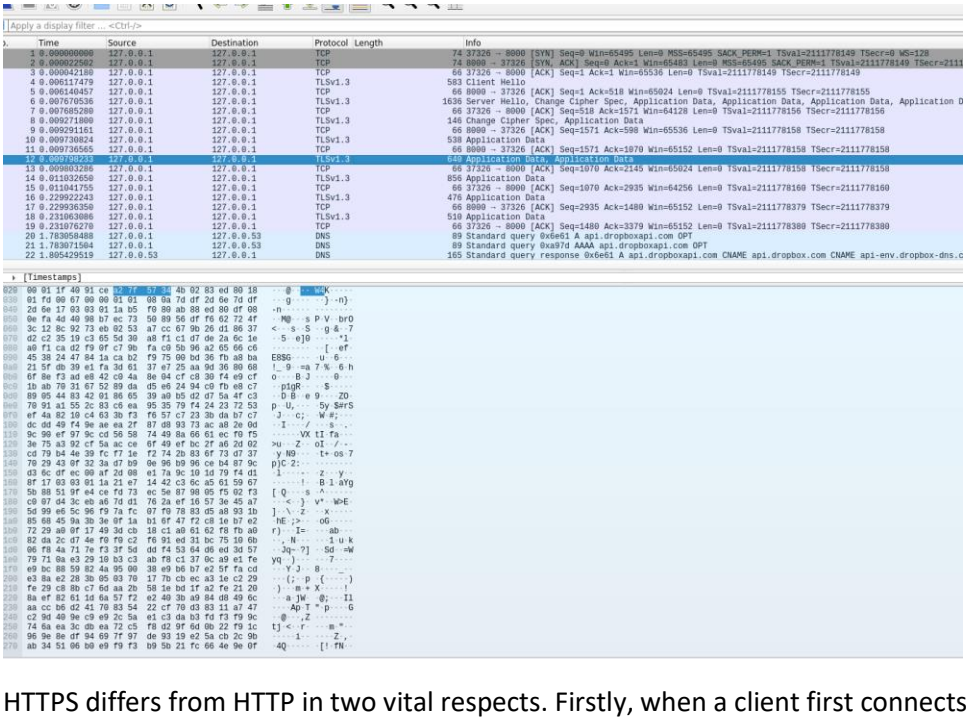
Please see 'insecure.pcapng' for our HTTP packet trace.

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 ay `localhost` server.



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://comodosslstore.com/resources/ssl-certificate-for-localhost/

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

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