Sarah Tse(748537)
I-Hsin Lin(754961)

Assignment 5 - Secure Sockets Layer(SSL)

Section 1: Goals of the experiment

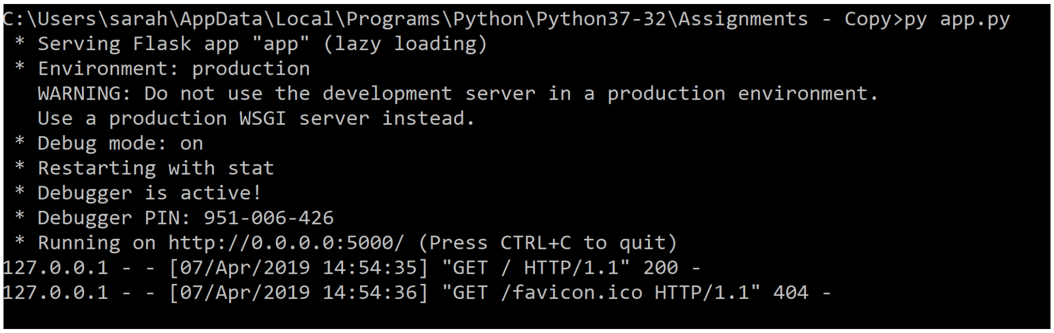
The main goal in this experiment is to understand how SSL provides security at the application layer and observe the differences. For this assignment, we have chosen to create a web server to exchange information between the server and client. In choosing this option, some of our sub goals are to learn how to locally host a web server and utilize Wireshark to read unencrypted HTML packets.

Section 2: Experimental Setup

To build a fast and simple website, we utilized Flask which is a Python based micro website application framework. The plan is to set up a locally hosted website where the client will exchange data with the server both unsecured and secured with SSL.

First, we programmed the back-end in python which would set up the site to be locally hosted and accept data from the front-end. On the front-end, the interface prompted users to enter their age and display their data after submitting the answer. Appendix A contains the back-end of the site and Appendix B and C contain the html of the site. Due to the location of the program, the files had to be stored within the program files in order to use the "pip" and "py" commands. The whole project was stored in a folder and within that folder a "templates" folder contained the html files.

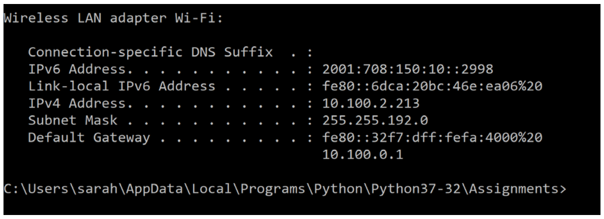
The to start locally hosting, we ran the python code on the server computer using command line as seen in Figure 1 below.



```
C:\Users\sarah\AppData\Local\Programs\Python\Python37-32\Assignments - Copy>py app.py
 * Serving Flask app "app" (lazy loading)
 * Environment: production
   WARNING: Do not use the development server in a production environment.
   Use a production WSGI server instead.
 * Debug mode: on
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 951-006-426
 * Running on http://0.0.0.0:5000/ (Press CTRL+C to quit)
127.0.0.1 - - [07/Apr/2019 14:54:35] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [07/Apr/2019 14:54:36] "GET /favicon.ico HTTP/1.1" 404 -
```

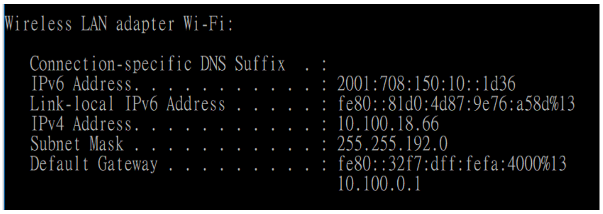
Figure 1:Starting web server on local host

Both the server and client are on the same network(aalto open). The client was able to access the website by typing in the server's IPv4 address and port. Figures 2a and 2b show the IPv4 addresses of the server and client.



```
Wireless LAN adapter Wi-Fi:

   Connection-specific DNS Suffix  . :
   IPv6 Address. . . . . . . . . . . : 2001:708:150:10::2998
   Link-local IPv6 Address . . . . . : fe80::6dca:20bc:46e:ea06%20
   IPv4 Address. . . . . . . . . . . : 10.100.2.213
   Subnet Mask . . . . . . . . . . . : 255.255.192.0
   Default Gateway . . . . . . . . . : fe80::32f7:dff:fefa:4000%20
                                       10.100.0.1

C:\Users\sarah\AppData\Local\Programs\Python\Python37-32\Assignments>
```

Figure 2a: Server ip address

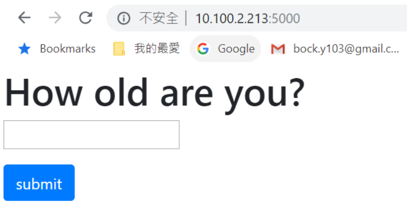


```
Wireless LAN adapter Wi-Fi:

   Connection-specific DNS Suffix  . :
   IPv6 Address. . . . . . . . . . . : 2001:708:150:10::1d36
   Link-local IPv6 Address . . . . . : fe80::81d0:4d87:9e76:a58d%13
   IPv4 Address. . . . . . . . . . . : 10.100.18.66
   Subnet Mask . . . . . . . . . . . : 255.255.192.0
   Default Gateway . . . . . . . . . : fe80::32f7:dff:fefa:4000%13
                                       10.100.0.1
```

Figure 2b: Client ip address

Figures 3a and 3b show a successful connection from the client side and the elements of the website.

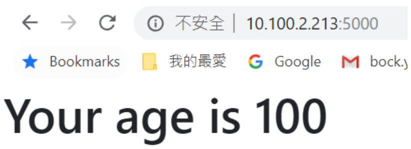
Sarah Tse(748537)
I-Hsin Lin(754961)



How old are you?

submit

Figure 3a: Index.html



Your age is 100

Figure 3b: age.html

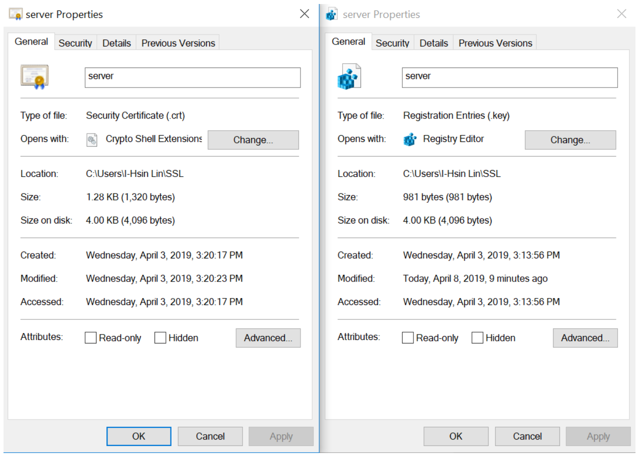
Section 2b: Experimental setup with SSL

(Python+flask: Configuring https website with ssl security certification)

In order to implement SSL, a security certificate and key were creating using OpenSSL following these steps.

1. Install pyOpenSSL
   ```
   pip install pyOpenSSL
   ```
2. Generate a private key and follow the prompts to fill in the content.
   ```
   openssl genrsa -des3 -out server.key 1024
   ```
3. Generate a csr file and follow the prompts to fill in the content.
   ```
   openssl req -new -key server.key -out server.csr
   ```
4. Step4: Generate crt file, valid for 1 year
   ```
   openssl x509 -req -days 365 -in server.csr -signkey server.key -out server.crt
   ```

Once the command above is executed, the server.crt and server.key files will be generated in the folder.



server

Type of file: Security Certificate (.crt)
Opens with: Crypto Shell Extensions
Location: C:\Users\I-Hsin Lin\SSL
Size: 1.28 KB (1,320 bytes)
Size on disk: 4.00 KB (4,096 bytes)
Created: Wednesday, April 3, 2019, 3:20:17 PM
Modified: Wednesday, April 3, 2019, 3:20:23 PM
Accessed: Wednesday, April 3, 2019, 3:20:17 PM
Attributes: □ Read-only □ Hidden

server

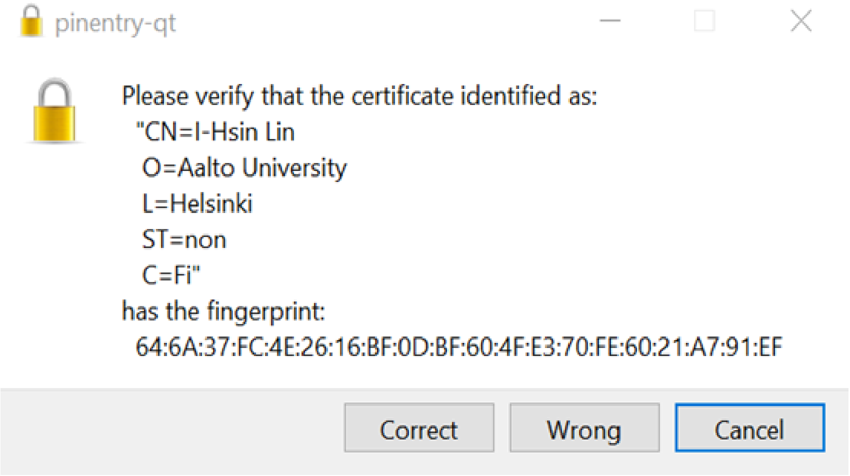
Type of file: Registration Entries (.key)
Opens with: Registry Editor
Location: C:\Users\I-Hsin Lin\SSL
Size: 981 bytes (981 bytes)
Size on disk: 4.00 KB (4,096 bytes)
Created: Wednesday, April 3, 2019, 3:13:56 PM
Modified: Today, April 8, 2019, 9 minutes ago
Accessed: Wednesday, April 3, 2019, 3:13:56 PM
Attributes: □ Read-only □ Hidden

Figure 4: Security certificate and key

After inserting the security key and certificate into the code (last line of code in Appendix D) and accessing the website again, the browser said that the certification had to be trusted by root.

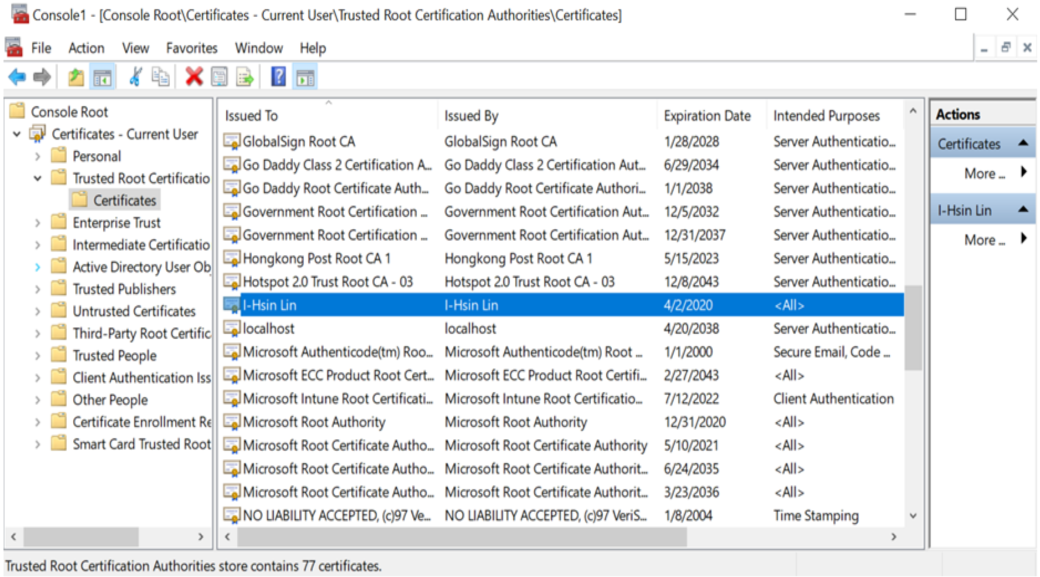
This is because the generated certificate is not issued by a trusted certificate authority so it is required to configure the trust settings on the client side.

5. Open Microsoft Management Control and import the certification to root



Please verify that the certificate identified as:
"CN=I-Hsin Lin
O=Aalto University
L=Helsinki
ST=non
C=Fi"
has the fingerprint:
64:6A:37:FC:4E:26:16:BF:0D:BF:60:4F:E3:70:FE:60:21:A7:91:EF

Figure 5a: Trusting the certificate on client side


Console1 - [Console Root\Certificates - Current User\Trusted Root Certification Authorities\Certificates]
File
Action
View
Favorites
Window
Help
Console Root
Certificates - Current User
Personal
Trusted Root Certificatio
Certificates
Enterprise Trust
Intermediate Certificatio
Active Directory User Ob
Trusted Publishers
Untrusted Certificates
Third-Party Root Certific
Trusted People
Client Authentication Iss
Other People
Certificate Enrollment Re
Smart Card Trusted Root
Actions
Certificates
More ...
I-Hsin Lin
More ...

Issued To	Issued By	Expiration Date	Intended Purposes
GlobalSign Root CA	GlobalSign Root CA	1/28/2028	Server Authenticatio...
Go Daddy Class 2 Certification A...	Go Daddy Class 2 Certification Aut...	6/29/2034	Server Authenticatio...
Go Daddy Root Certificate Auth...	Go Daddy Root Certificate Authori...	1/1/2038	Server Authenticatio...
Government Root Certification ...	Government Root Certification Aut...	12/5/2032	Server Authenticatio...
Government Root Certification ...	Government Root Certification Aut...	12/31/2037	Server Authenticatio...
Hongkong Post Root CA 1	Hongkong Post Root CA 1	5/15/2023	Server Authenticatio...
Hotspot 2.0 Trust Root CA - 03	Hotspot 2.0 Trust Root CA - 03	12/8/2043	Server Authenticatio...
I-Hsin Lin	I-Hsin Lin	4/2/2020	<All>
localhost	localhost	4/20/2038	Server Authenticatio...
Microsoft Authenticode(tm) Roo...	Microsoft Authenticode(tm) Root ...	1/1/2000	Secure Email, Code ...
Microsoft ECC Product Root Cert...	Microsoft ECC Product Root Certifi...	2/27/2043	<All>
Microsoft Intune Root Certificati...	Microsoft Intune Root Certificatio...	7/12/2022	Client Authentication
Microsoft Root Authority	Microsoft Root Authority	12/31/2020	<All>
Microsoft Root Certificate Autho...	Microsoft Root Certificate Authority	5/10/2021	<All>
Microsoft Root Certificate Autho...	Microsoft Root Certificate Authorit...	6/24/2035	<All>
Microsoft Root Certificate Autho...	Microsoft Root Certificate Authorit...	3/23/2036	<All>
NO LIABILITY ACCEPTED, (c)97 Ve...	NO LIABILITY ACCEPTED, (c)97 VeriS...	1/8/2004	Time Stamping

Trusted Root Certification Authorities store contains 77 certificates.

Figure 5b: Trusting the certificate on client side, Microsoft Management Console

6. Once the certificate is added to the root, access the website again and the certification would be trusted. The site is still classified as “not secured” status but it now considered “ok” since we have manually added the trust settings.

Sarah Tse(748537)
I-Hsin Lin(754961)

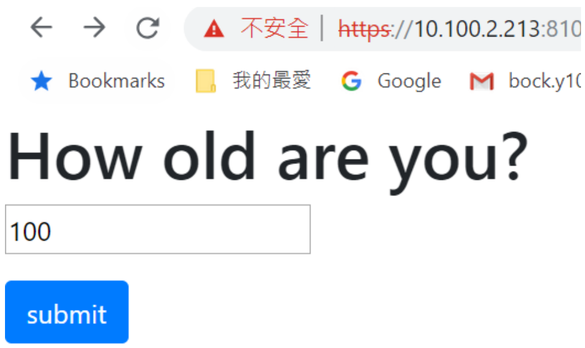


Figure 6a: Accessing website again

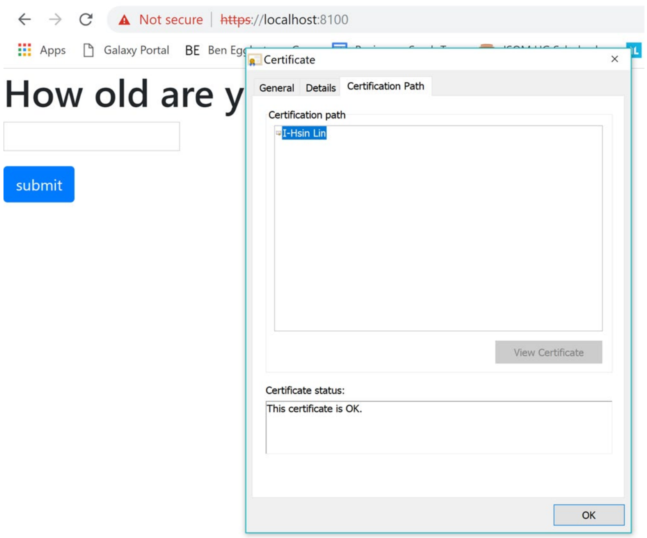
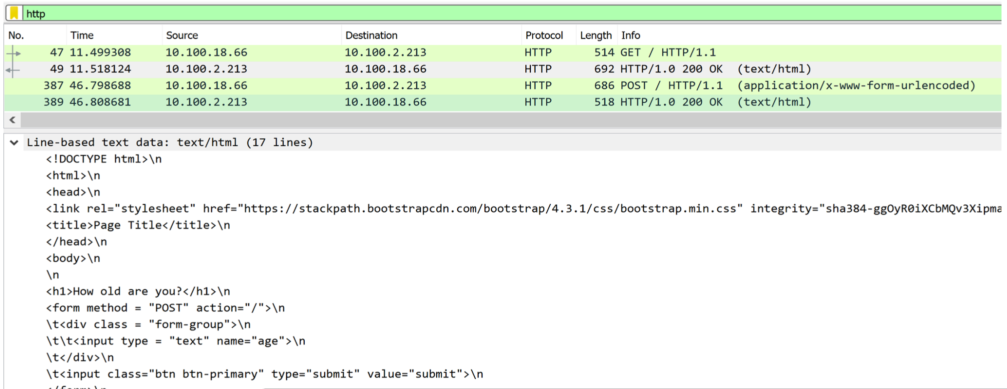


Figure 6b: Certificate status is “ok”

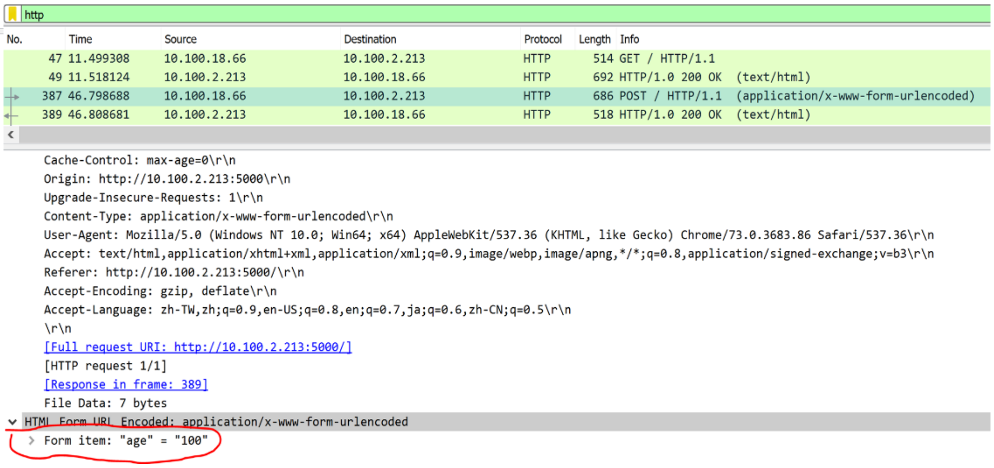
Section 3a: Results without TLS

When the client is accessing the unsecured website, Wireshark (running on server side) captured HTTP packets that show the html of the webpage and also the user submitted data all in plaintext.



```
http
No.   Time        Source          Destination     Protocol  Length  Info
  47  11.499308   10.100.18.66    10.100.2.213    HTTP       514 GET / HTTP/1.1
  49  11.518124   10.100.2.213    10.100.18.66    HTTP       692 HTTP/1.0 200 OK  (text/html)
 387  46.798688   10.100.18.66    10.100.2.213    HTTP       686 POST / HTTP/1.1  (application/x-www-form-urlencoded)
 389  46.808681   10.100.2.213    10.100.18.66    HTTP       518 HTTP/1.0 200 OK  (text/html)

Line-based text data: text/html (17 lines)
    <!DOCTYPE html>\n
    <html>\n
    <head>\n
    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css" integrity="sha384-ggOyR0iXCbMQv3Xipma
    <title>Page Title</title>\n
    </head>\n
    <body>\n
    \n
    <h1>How old are you?</h1>\n
    <form method = "POST" action="/">\n
    \t<div class = "form-group">\n
    \t\t<input type = "text" name="age">\n
    \t</div>\n
    \t<input class="btn btn-primary" type="submit" value="submit">\n
    </form>\n
    </body>\n
```

Figure 7a: unencrypted HTTP packet with HTML inside



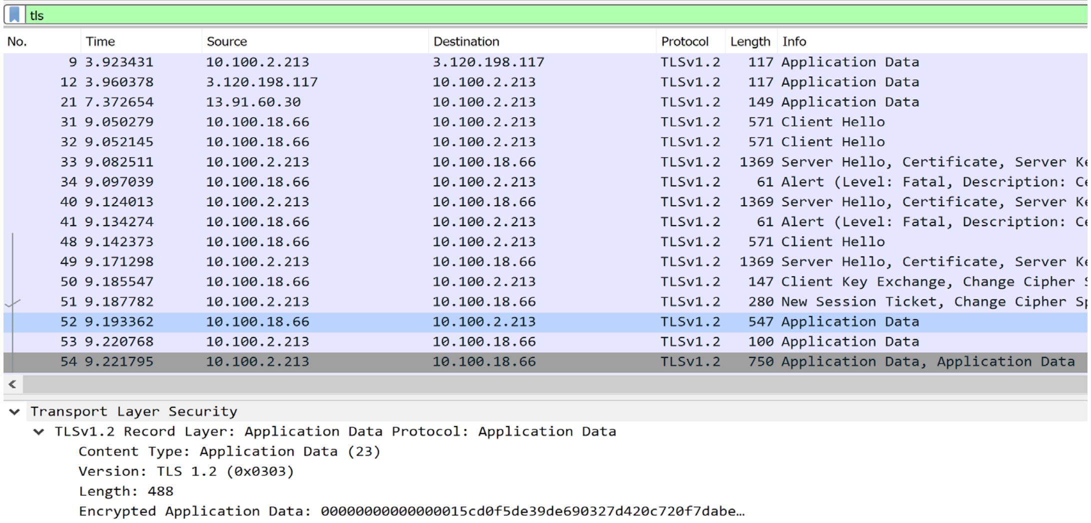
```
http
No.   Time        Source          Destination     Protocol  Length  Info
  47  11.499308   10.100.18.66    10.100.2.213    HTTP       514 GET / HTTP/1.1
  49  11.518124   10.100.2.213    10.100.18.66    HTTP       692 HTTP/1.0 200 OK  (text/html)
 387  46.798688   10.100.18.66    10.100.2.213    HTTP       686 POST / HTTP/1.1  (application/x-www-form-urlencoded)
 389  46.808681   10.100.2.213    10.100.18.66    HTTP       518 HTTP/1.0 200 OK  (text/html)

    Cache-Control: max-age=0\r\n
    Origin: http://10.100.2.213:5000\r\n
    Upgrade-Insecure-Requests: 1\r\n
    Content-Type: application/x-www-form-urlencoded\r\n
    User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/73.0.3683.86 Safari/537.36\r\n
    Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3\r\n
    Referer: http://10.100.2.213:5000/\r\n
    Accept-Encoding: gzip, deflate\r\n
    Accept-Language: zh-TW,zh;q=0.9,en-US;q=0.8,en;q=0.7,ja;q=0.6,zh-CN;q=0.5\r\n
    \r\n
    [Full request URI: http://10.100.2.213:5000/]
    [HTTP request 1/1]
    [Response in frame: 389]
    File Data: 7 bytes
HTML Form URL Encoded: application/x-www-form-urlencoded
  > Form item: "age" = "100"
```

Figure7b: unencrypted HTTP packet with user submitted data.

Sarah Tse(748537)
I-Hsin Lin(754961)

Section 3b: Results with SSL

When the client is accessing the site with the certificate, Wireshark (running on the server side) is capturing TLS packets and the data is encrypted.



tls

No.	Time	Source	Destination	Protocol	Length	Info
9	3.923431	10.100.2.213	3.120.198.117	TLSv1.2	117	Application Data
12	3.960378	3.120.198.117	10.100.2.213	TLSv1.2	117	Application Data
21	7.372654	13.91.60.30	10.100.2.213	TLSv1.2	149	Application Data
31	9.050279	10.100.18.66	10.100.2.213	TLSv1.2	571	Client Hello
32	9.052145	10.100.18.66	10.100.2.213	TLSv1.2	571	Client Hello
33	9.082511	10.100.2.213	10.100.18.66	TLSv1.2	1369	Server Hello, Certificate, Server Ke
34	9.097039	10.100.18.66	10.100.2.213	TLSv1.2	61	Alert (Level: Fatal, Description: Ce
40	9.124013	10.100.2.213	10.100.18.66	TLSv1.2	1369	Server Hello, Certificate, Server Ke
41	9.134274	10.100.18.66	10.100.2.213	TLSv1.2	61	Alert (Level: Fatal, Description: Ce
48	9.142373	10.100.18.66	10.100.2.213	TLSv1.2	571	Client Hello
49	9.171298	10.100.2.213	10.100.18.66	TLSv1.2	1369	Server Hello, Certificate, Server Ke
50	9.185547	10.100.18.66	10.100.2.213	TLSv1.2	147	Client Key Exchange, Change Cipher S
51	9.187782	10.100.2.213	10.100.18.66	TLSv1.2	280	New Session Ticket, Change Cipher Sp
52	9.193362	10.100.18.66	10.100.2.213	TLSv1.2	547	Application Data
53	9.220768	10.100.2.213	10.100.18.66	TLSv1.2	100	Application Data
54	9.221795	10.100.2.213	10.100.18.66	TLSv1.2	750	Application Data, Application Data

```
Transport Layer Security
  TLSv1.2 Record Layer: Application Data Protocol: Application Data
      Content Type: Application Data (23)
      Version: TLS 1.2 (0x0303)
      Length: 488
      Encrypted Application Data: 00000000000000015cd0f5de39de690327d420c720f7dabe…
```

Figure 8: Encrypted data within packet

Section 3c: Conclusions

Through this experiment, we were able to observe how an SSL certificate and key can encrypt data. As seen in the Figures 7a, 7b and 8, having an SSL certificate encrypts data and ensures safety for the client accessing the website in the application layer.

Section 4: Appendices

Appendix A - Unsecure Backend Python Code (app.py)

```
from flask import Flask, render_template, request
app = Flask(__name__)

@app.route("/", methods=['GET','POST'])
def send():
    if request.method == 'POST':
        age = request.form['age']

        return render_template('age.html', age=age)

    return render_template('index.html')

if __name__ == "__main__":
    app.run(debug=True, host='0.0.0.0')
```

Appendix B - Index.html

```
<!DOCTYPE html>
<html>
<head>
<link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap
.min.css" integrity="sha384-
```

```
ggOyR0iXCbMQv3Xipma34MD+dH/1fQ784/j6cY/iJTQUOhcWr7x9JvoRxT2MZw1T"
crossorigin="anonymous">
<title>Page Title</title>
</head>
<body>

<h1>How old are you?</h1>
<form method = "POST" action="/">
      <div class = "form-group">
            <input type = "text" name="age">
      </div>
      <input class="btn btn-primary" type="submit" value="submit">
</form>
</body>
</html>
```

Appendix C - age.html

```
<!DOCTYPE html>
<html>
<head>
<link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap
.min.css" integrity="sha384-
ggOyR0iXCbMQv3Xipma34MD+dH/1fQ784/j6cY/iJTQUOhcWr7x9JvoRxT2MZw1T"
crossorigin="anonymous">
<title>Page Title</title>
</head>
<body>

      <h1> Your age is {{ age }}</h1>
</body>
</html>
```

Appendix D - Secured app.py

```
from flask import Flask, render_template, request
app = Flask(__name__)

@app.route("/", methods=['GET','POST'])
def send():
    if request.method == 'POST':
        age = request.form['age']

        return render_template('age.html', age=age)

    return render_template('index.html')

if __name__ == "__main__":
    app.run(debug=True, host='0.0.0.0', port=8100,
ssl_context=('server.crt', 'server.key'))
```