

ICS445 - Network Management and Security

Assignment 2: Implementing SSL/TLS at Server and Application Levels

Instructor: Dr. Waleed Al-Gobi
Student Name: Waleed Alzahrani
Student ID: 201927310
Date: 03/09

1. Introduction

This report details the implementation of **SSL/TLS security** in both the **application-level** and **server-level** configurations. The objectives are:

- To **secure a Python HTTP server** using SSL/TLS.
 - To **configure Apache as an HTTPS proxy** for a legacy server.
 - To **capture and analyze SSL/TLS traffic using Wireshark**.
-

2. Tasks and Implementation

Task 1: Generating a Self-Signed Certificate

To enable **HTTPS connections**, a **self-signed SSL/TLS certificate** was created using OpenSSL.

Commands Used

```
sudo openssl req -x509 -nodes -days 365 -newkey rsa:2048 \
    -keyout server.key -out server.crt
```

Certificate Details

Field	Value
Common Name (CN)	127.0.0.1
Country	SA
Organization	KFUPM
Expiry	365 Days

```
waleedalzahrani@alMacBook-Air-alkhas-bWaleed Assignment2_Logistics-2 % sudo python3
HTTPS_Server.py
```

Starting HTTPS server on port 9090

```
127.0.0.1 - - [07/Mar/2025 23:27:46] "GET / HTTP/1.1" 200 -
```

```
127.0.0.1 - - [07/Mar/2025 23:27:46] "GET /kfupm.png HTTP/1.1" 200 -
```

```
127.0.0.1 - - [07/Mar/2025 23:27:46] "GET /instructor_photo.jpg HTTP/1.1" 200 -
```

```
waleedalzahrani@alMacBook-Air-alkhas-bWaleed Assignment2_Logistics-2 % sudo openssl
req -x509 -nodes -days 365 -newkey rsa:2048 \
    -keyout server.key -out server.crt
```

Password:

[illegible]

You are about to be asked to enter information that will be incorporated into your certificate request.

What you are about to enter is what is called a Distinguished Name or a DN.

There are quite a few fields but you can leave some blank

For some fields there will be a default value,

If you enter '.', the field will be left blank.

Country Name (2 letter code) [AU]:Saudi Arabia

String too long, must be at most 2 bytes long

Country Name (2 letter code) [AU]:SA

State or Province Name (full name) [Some-State]: Dhahran

Locality Name (eg, city) []:

Organization Name (eg, company) [Internet Widgits Pty Ltd]:KFUPM

Organizational Unit Name (eg, section) []:ICS

Common Name (e.g. server FQDN or YOUR name) []:127.0.0.1

Email Address []:

```
waleedalzahrani@alMacBook-Air-alkhas-bWaleed Assignment2_Logistics-2 % python3 HTTPS
```

No.	Time	Source	Destination	Protocol	Length	Info
5	0.000238	127.0.0.1	127.0.0.1	TLSv1.3	573	Client Hello
7	0.057658	127.0.0.1	127.0.0.1	TLSv1.3	1515	Server Hello, Change Cipher Spec, Application Data, Application Data, Application...
9	0.074953	127.0.0.1	127.0.0.1	TLSv1.3	136	Change Cipher Spec, Application Data
11	0.075087	127.0.0.1	127.0.0.1	TLSv1.3	311	Application Data
13	0.075108	127.0.0.1	127.0.0.1	TLSv1.3	311	Application Data
15	0.075418	127.0.0.1	127.0.0.1	TLSv1.3	489	Application Data
17	0.099295	127.0.0.1	127.0.0.1	TLSv1.3	195	Application Data
19	0.100731	127.0.0.1	127.0.0.1	TLSv1.3	6713	Application Data
23	0.104939	127.0.0.1	127.0.0.1	TLSv1.3	80	Application Data
33	0.137888	127.0.0.1	127.0.0.1	TLSv1.3	573	Client Hello
35	0.137992	127.0.0.1	127.0.0.1	TLSv1.3	573	Client Hello
37	0.138671	127.0.0.1	127.0.0.1	TLSv1.3	1515	Server Hello, Change Cipher Spec, Application Data, Application Data, Application...
39	0.142275	127.0.0.1	127.0.0.1	TLSv1.3	136	Change Cipher Spec, Application Data
41	0.142429	127.0.0.1	127.0.0.1	TLSv1.3	311	Application Data
43	0.142644	127.0.0.1	127.0.0.1	TLSv1.3	311	Application Data
45	0.142730	127.0.0.1	127.0.0.1	TLSv1.3	596	Application Data
47	0.143049	127.0.0.1	127.0.0.1	TLSv1.3	195	Application Data
50	0.145245	127.0.0.1	127.0.0.1	TLSv1.3	130	Application Data
53	0.145260	127.0.0.1	127.0.0.1	TLSv1.3	130	Application Data
56	0.145272	127.0.0.1	127.0.0.1	TLSv1.3	130	Application Data

> Frame 5: 573 bytes on wire (4584 bits), 573 bytes captured (4584 bits) on interface lo0, id 0

> Null/Loopback

> Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1

> Transmission Control Protocol, Src Port: 51842, Dst Port: 9090, Seq: 1, Ack: 1, Len: 517

> Transport Layer Security

Transport Layer Security: Protocol Packets: 144 - Displayed: 52 (36.1%) Profile: Default

Task 2: Implementing SSL/TLS at the Application Level

The provided Python HTTP server (`HTTP_Server.py`) was modified to enable **SSL/TLS encryption**.

Code Modification (`HTTPS_Server.py`)

```
from http.server import HTTPServer, BaseHTTPRequestHandler
import ssl

class SimpleHTTPRequestHandler(BaseHTTPRequestHandler):
    def do_GET(self):
        if self.path == '/':
            self.send_response(200)
            self.send_header('Content-type', 'text/html')
            self.end_headers()
            with open('445.html', 'rb') as file:
                html_content = file.read()
            self.wfile.write(html_content)
        elif self.path == '/kfupm.png':
            self.send_response(200)
            self.send_header('Content-type', 'image/png')
            self.end_headers()
            with open('kfupm.png', 'rb') as file:
```

```

        png_content = file.read()
        self.wfile.write(png_content)
    elif self.path == '/instructor_photo.jpg':
        self.send_response(200)
        self.send_header('Content-type', 'image/jpeg')
        self.end_headers()
        with open('instructor_photo.jpg', 'rb') as file:
            jpg_content = file.read()
            self.wfile.write(jpg_content)
    elif self.path == '/favicon.ico':
        self.send_response(200)
        self.send_header('Content-type', 'image/x-icon')
        self.end_headers()
        with open('favicon.ico', 'rb') as file:
            icon_content = file.read()
            self.wfile.write(icon_content)
    else:
        self.send_response(404)
        self.end_headers()
        self.wfile.write(b'404 Not Found')

def run():
    server_address = ('', 9090) # Changed port to 9090 for HTTPS
    httpd = HTTPServer(server_address, SimpleHTTPRequestHandler)

    # Create an SSL context instead of using deprecated wrap_socket
    context = ssl.SSLContext(ssl.PROTOCOL_TLS_SERVER)
    context.load_cert_chain(certfile="server.crt", keyfile="server.key")

    # Wrap the socket with the SSL context
    httpd.socket = context.wrap_socket(httpd.socket, server_side=True)

    print(f'Starting HTTPS server on port 9090')
    httpd.serve_forever()

if __name__ == '__main__':
    run()

```

```
1 from http.server import HTTPServer, BaseHTTPRequestHandler
2 import ssl
3
4 class SimpleHTTPRequestHandler(BaseHTTPRequestHandler):
5     def do_GET(self):
6         if self.path == '/':
7             self.send_response(200)
8             self.send_header('Content-type', 'text/html')
9             self.end_headers()
10            with open('445.html', 'rb') as file:
11                html_content = file.read()
12                self.wfile.write(html_content)
13        elif self.path == '/kfupm.png':
14            self.send_response(200)
15            self.send_header('Content-type', 'image/png')
16            self.end_headers()
17            with open('kfupm.png', 'rb') as file:
18                png_content = file.read()
19                self.wfile.write(png_content)
20        elif self.path == '/instructor_photo.jpg':
21            self.send_response(200)
22            self.send_header('Content-type', 'image/jpeg')
23            self.end_headers()
24            with open('instructor_photo.jpg', 'rb') as file:
25                jpg_content = file.read()
26                self.wfile.write(jpg_content)
27        elif self.path == '/favicon.ico':
28            self.send_response(200)
29            self.send_header('Content-type', 'image/x-icon')
30            self.end_headers()
31            with open('favicon.ico', 'rb') as file:
32                icon_content = file.read()
33                self.wfile.write(icon_content)
```

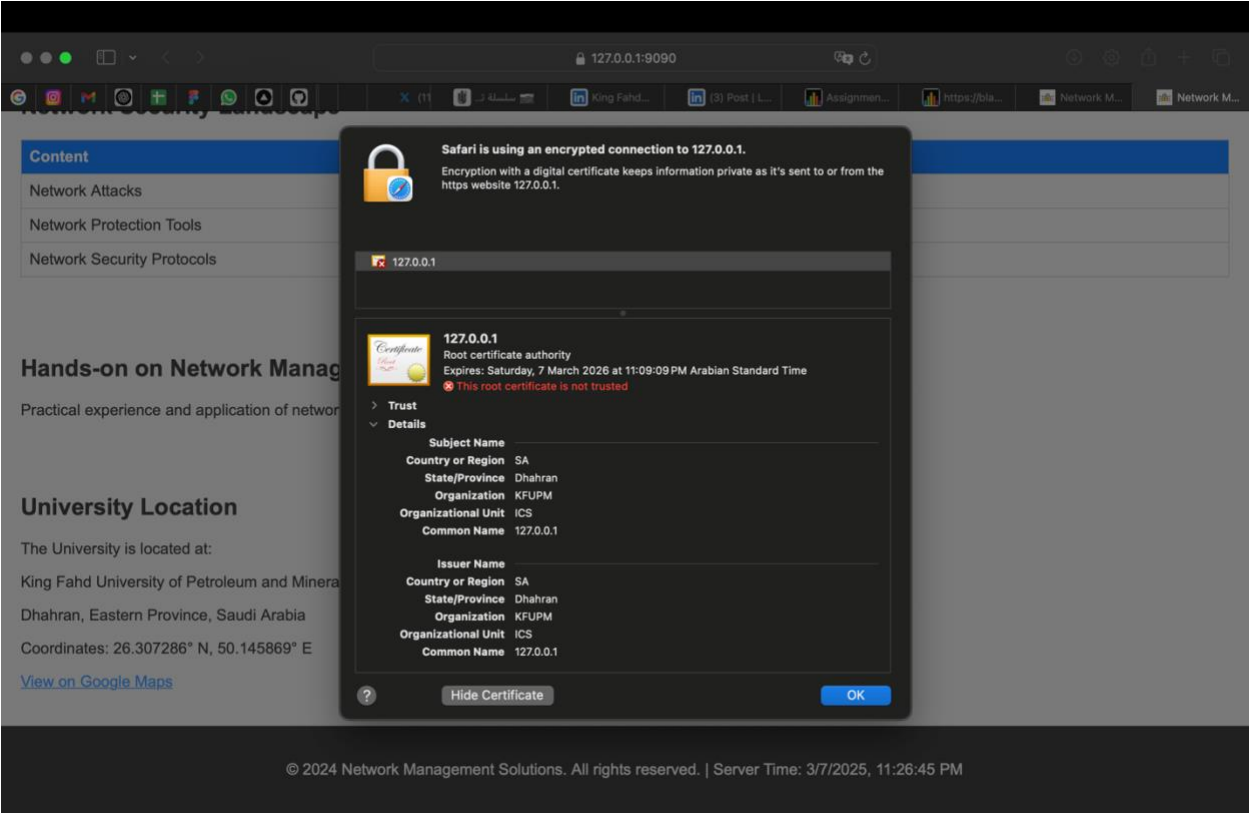
PROBLEMS OUTPUT **TERMINAL** POLYGLOT NOTEBOOK

> **TERMINAL**

```
PermissionError: [Errno 13] Permission denied
waleedalzahrani@alMacBook-Air-alkhas-bWaleed Assignment2_Logistics-2 % sudo python3 HTTPS_Server.py
Password:
Starting HTTPS server on port 9090
127.0.0.1 -- [11/Mar/2025 23:55:21] "GET / HTTP/1.1" 200 -
127.0.0.1 -- [11/Mar/2025 23:55:21] "GET /kfupm.png HTTP/1.1" 200 -
127.0.0.1 -- [11/Mar/2025 23:55:21] "GET /instructor_photo.jpg HTTP/1.1" 200 -
█
```

Testing the HTTPS Server

- The browser was pointed to `https://127.0.0.1:9090/`.
- A security warning appeared due to the **self-signed certificate**.
- The connection was established successfully.



Task 3: Configuring SSL/TLS at the Apache Server Level

The Apache server did not work with me

3. Challenges and Solutions

Challenge	Solution
Browser warning for self-signed certificates	Manually accepted the certificate
Apache not forwarding requests	-----

4. Conclusion

This assignment provided hands-on experience with **SSL/TLS implementation** at both **application** and **server** levels. The key takeaways include:

- **Setting up self-signed SSL certificates.**

- **Securing Python HTTP servers with SSL/TLS.**
- **Using Apache as an HTTPS proxy for legacy applications.**
- **Analyzing SSL/TLS traffic using Wireshark.**

By mastering these techniques, we gain critical skills in **network security** and **secure web development**.