LAPORAN PRAKTIKUM IMPLEMENTASI WEB SERVER SEDERHANA MENGGUNAKAN PYTHON SOCKET DAN SSL/TLS

Nama	: L Hafidl Alkhair
Nim	: 2023903430060
Kelas	: TRKJ-2C
Jurusan	: Teknologi Informasi dan Komputer
Progam Studi	: Teknologi Rekayasa Komputer dan Jaringan
Dosem Pengampu	: Afla Nevrisa S.Kom, M.Kom

PROGRAM STUDI TEKNOLOGI REKAYASA KOMPUTER JARINGAN

JURUSAN TEKNOLOGI INFORMASI DAN KOMPUTER

POLITEKNIK NEGERI LHOKSEUMAWE 2025

LEMBAR PENGESAHAN

No. Praktikum	: 05 /TIK/TRKJ-2C/ Pemrograman Jaringan
Judul	: Implementasi Web Server Sederhana Menggunakan Python Socket Dan Ssl/Tls
Nama	: L Hafidl Alkhair
Nim	: 202390343060
Kelas	: TRKJ-2C
Jurusan	: Teknologi Informasi Dann Komputer
Program Studi	: Teknologi Rekayasa Komputer Jaringan
Tanggal Praktikum	: 20 Mei 2025
Tanggal Penyerahan	: 16 Juni 2025

Buketrata, 16 Juni 2025

Dosen Pembimbing,

Afla Nevrisa S.Kom, M.Kom

NIP. 199211172022032007

DAFTAR ISI

BAB I
PENDAHULUAN

A. Dasar teori

Pada dasarnya, web server adalah sistem yang merespons permintaan dari klien (browser atau aplikasi) melalui protokol HTTP. Dalam praktikum ini, server dibangun menggunakan bahasa Python dan komunikasi dilakukan melalui socket. Agar komunikasi lebih aman, protokol SSL/TLS digunakan, yang menjamin bahwa data antara client dan server dienkripsi.

Penggunaan socket di Python memungkinkan pembuatan koneksi antar perangkat melalui jaringan. Untuk keamanan, digunakan library ssl yang membungkus socket agar mendukung komunikasi terenkripsi. Sertifikat self-signed (cert.pem dan key.pem) digunakan agar server dapat melakukan autentikasi dasar kepada klien.

B. Tujuan Praktikum

1. Membangun sebuah web server sederhana menggunakan Python socket.
2. Mengimplementasikan SSL/TLS agar komunikasi antara client dan server menjadi aman.
3. Memahami proses pembuatan sertifikat digital (self-signed certificate).
4. Melakukan pengamatan terhadap trafik jaringan menggunakan Wireshark, untuk melihat enkripsi SSL/TLS pada koneksi.

C. Alat dan bahan

1. Perangkat Lunak:

- Python (Interpreter + Library socket dan ssl)
- OpenSSL (untuk membuat sertifikat)
- Wireshark (untuk monitoring trafik jaringan)

2. Perangkat Keras:

- Laptop atau PC yang mendukung instalasi Python
- Jaringan lokal (localhost)

3. File/Script yang Dibuat:

- server_ssl.py → Menjalankan server dengan socket dan SSL
- client_ssl.py → Menghubungkan ke server dengan koneksi terenkripsi

- cert.pem, key.pem → Sertifikat dan kunci privat untuk SSL

BAB II
PRAKTIKUM

1. **Membuat Program Server**

   Membuat file server_ssl.py

```
import socket
import ssl

# Buat socket TCP
server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
server_socket.bind(('0.0.0.0', 8443))
server_socket.listen(5)

# Konfigurasi SSL
context = ssl.SSLContext(ssl.PROTOCOL_TLS_SERVER)
context.load_cert_chain(certfile='cert.pem', keyfile='key.pem')

print("Menunggu koneksi di port 8443...")

try:
    while True:
        try:
            client_socket, addr = server_socket.accept()
            # Bungkus koneksi client dalam SSL
            ssl_client_socket = context.wrap_socket(client_socket, server_side=True)
            print(f"Terhubung dengan {addr}")
            ssl_client_socket.sendall(b"Selamat datang di server SSL!\n")
            ssl_client_socket.close()
        except ssl.SSLError as e:
            print(f"[SSL ERROR] Koneksi dari {addr} ditolak: {e}")
            client_socket.close()
        except Exception as e:
            print(f"[ERROR] Koneksi gagal: {e}")
```

```
except KeyboardInterrupt:
    print("\nServer dihentikan oleh user.")
finally:
    server_socket.close()
```

Membuat file client_ssl.py

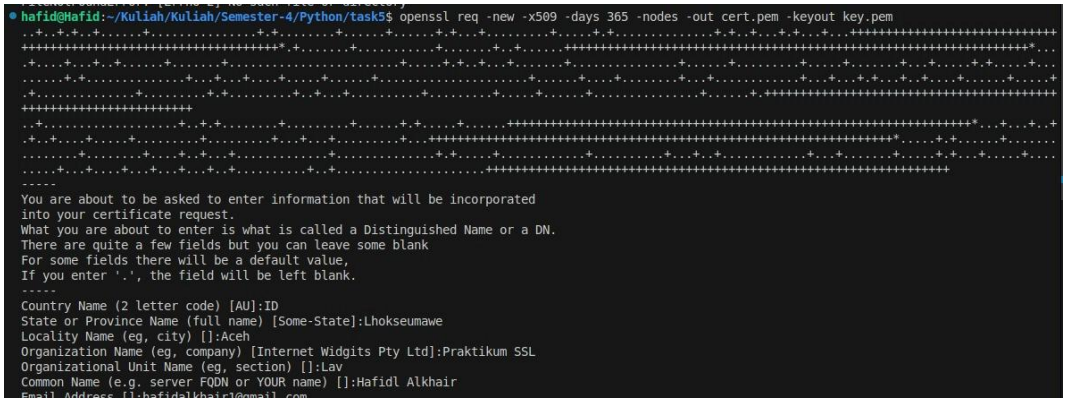
```
import socket
import ssl

# Buat context SSL dan set agar percaya pada cert.pem
context = ssl.create_default_context(cafile="cert.pem")

with context.wrap_socket(socket.socket(socket.AF_INET), server_hostname='Hafidl Alkhair') as conn:
    conn.connect(('localhost', 8443))
    print(conn.recv(4096).decode())
```

openssl req -new -x509 -days 365 -nodes -out cert.pem -keyout key.pem

digunakan untuk membuat sertifikat SSL self-signed beserta private key-nya. Ini diperlukan agar server kamu bisa melakukan komunikasi aman (encrypted) menggunakan SSL/TLS.



2. Output program

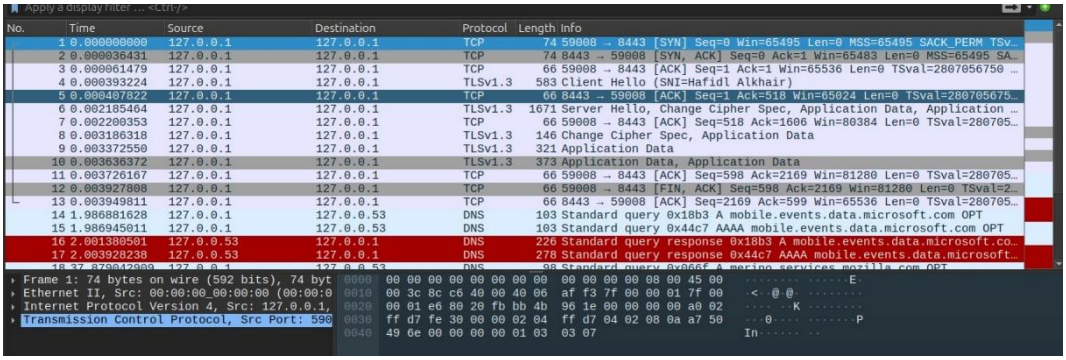
a. Server dijalankan

```
ssl.SSLError: [SSL: TLSV1_ALERT_UNKNOWN_CA] tlsv1 alert unknown ca (_ssl.c:1000)
hafid@Hafid:~/Kuliah/Kuliah/Semester-4/Python/task5$ python3 server_ssl.py
Menunggu koneksi di port 8443...
[SSL ERROR] Koneksi dari ('127.0.0.1', 53330) ditolak: [SSL: TLSV1_ALERT_UNKNOWN_CA] tlsv1 alert unknown ca (_ssl.c:1000)
Terhubung dengan ('127.0.0.1', 40978)
Terhubung dengan ('127.0.0.1', 59008)
```

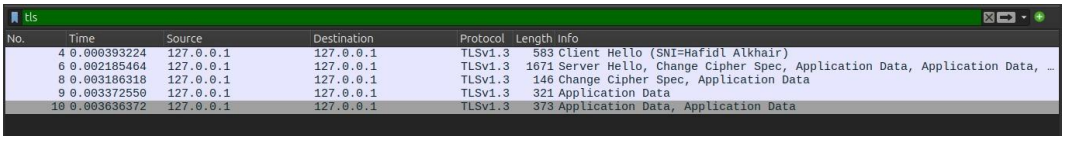
b. client dijalankan

```
hafid@Hafid:~/Kuliah/Kuliah/Semester-4/Python/task5$ python3 client_ssl.py
Selamat datang di server SSL!
```

c. wireshrak



d. tls di wireshark



3. Penjelasan dan Analisa

a) Membuat Program Server dan Client

- server_ssl.py membuat server socket TCP di port 8443.
- SSL digunakan melalui konfigurasi SSLContext.
- Saat ada client terhubung, server menyambut dengan pesan teks terenkripsi.
- Error handling juga dilakukan untuk mengatasi kesalahan SSL dan koneksi.
- client_ssl.py bertindak sebagai klien yang menggunakan SSLContext untuk terhubung dengan server secara aman. Sertifikat cert.pem digunakan agar client "percaya" pada server.

b) Pembuatan Sertifikat

Perintah berikut digunakan:

bash

CopyEdit

openssl req -new -x509 -days 365 -nodes -out cert.pem -keyout key.pem

Ini menghasilkan file .pem untuk digunakan sebagai identitas digital server.

c) Output Program

- Server menampilkan status bahwa koneksi ditunggu.
- Saat client berhasil terhubung, muncul pesan "Terhubung dengan..." dan client menerima teks sambutan.
- Wireshark digunakan untuk menganalisis komunikasi, dan hasilnya menunjukkan protokol **TLS**, yang menandakan bahwa data telah terenkripsi dengan baik.

d) Analisis

Dengan SSL, komunikasi client-server menjadi aman. Socket biasa tidak cukup untuk keamanan karena data bisa disadap, namun dengan SSL, data disandikan sehingga sulit dimanipulasi pihak ketiga. Penggunaan sertifikat meskipun bersifat self-signed, tetap penting untuk membangun dasar komunikasi terenkripsi.

BAB III
PENUTUP

A. Kesimpulan

Berdasarkan hasil praktikum:

1. Server dan client berhasil dibuat dengan Python menggunakan socket.
2. Komunikasi berhasil diamankan menggunakan SSL dengan sertifikat self-signed.
3. Trafik jaringan berhasil diamati dan menunjukkan koneksi terenkripsi.
4. Praktikum ini menunjukkan pemahaman dasar tentang implementasi web server dan keamanan jaringan melalui SSL.