Secure SSL VPN System with raw sockets and AES-256 encryption

Aasma, Akshaya Krishna M
PES University
Github : https://github.com/Aasma1306/VPN

Abstract—**This paper presents the development of a secure SSL-based Virtual Private Network (VPN) system featuring a graphical user interface (GUI) for enhanced usability and control. The system comprises a C-based multithreaded server and client utilizing OpenSSL and AES-256-CBC encryption, with a PyQt5-based GUI to initiate, monitor, and manage VPN connections. The integration of modern cryptographic techniques with a user-friendly interface makes this system an effective solution for secure communication in personal and enterprise environments.**

I. Introduction

The rise in cyber threats and the growing need for secure communication channels has made VPNs a critical technology. While many commercial VPNs are available, they often operate as black boxes. This project aims to create an open, educational, and modular VPN system leveraging OpenSSL for transport security and a Python-based GUI for ease of access.

II. System Architecture

The proposed system consists of three core components:

A. VPN Server (vpn_server.c)

The server is a C-based multithreaded program using OpenSSL for secure socket layer communication. It performs mutual TLS authentication and handles data encryption/decryption using AES-256-CBC. The server can accept and maintain multiple client connections simultaneously.

B. VPN Client (vpn_client.c)

The client program connects securely to the server using TLS. After authentication, encrypted communication is maintained using the AES-256-CBC algorithm. The client encrypts outgoing messages and decrypts incoming responses.

C. GUI Interface (vpn_gui.py)

Developed using PyQt5, the GUI allows users to:

- Start/stop the VPN connection
- Select IP, port, and protocol (UDP, TCP, WireGuard as a placeholder)
- Monitor real-time logs and connection duration
- Toggle between dark and light modes
- Save and manage VPN connection profiles

The GUI uses the `cryptography` module to encrypt saved configurations and profiles using Fernet encryption.

III. Features and Implementation

- **Security:** Mutual TLS authentication and symmetric AES encryption ensure end-to-end confidentiality.
- **Usability:** The GUI provides an intuitive interface for managing VPN operations without command-line interactions.
- **Extensibility:** Profiles can be managed through the interface, and additional protocols can be added.

IV. Challenges

The integration of a C-based client with a Python GUI required careful management of system calls and inter-process communication. Buffer size mismatches and encryption padding in AES were also handled meticulously to avoid memory leaks and data corruption.

V. Results

The system was successfully deployed on a local network. The VPN connection was stable and secure, and the GUI interface provided reliable status monitoring and control. The connection timer and log panel effectively tracked session durations and client-server communication.

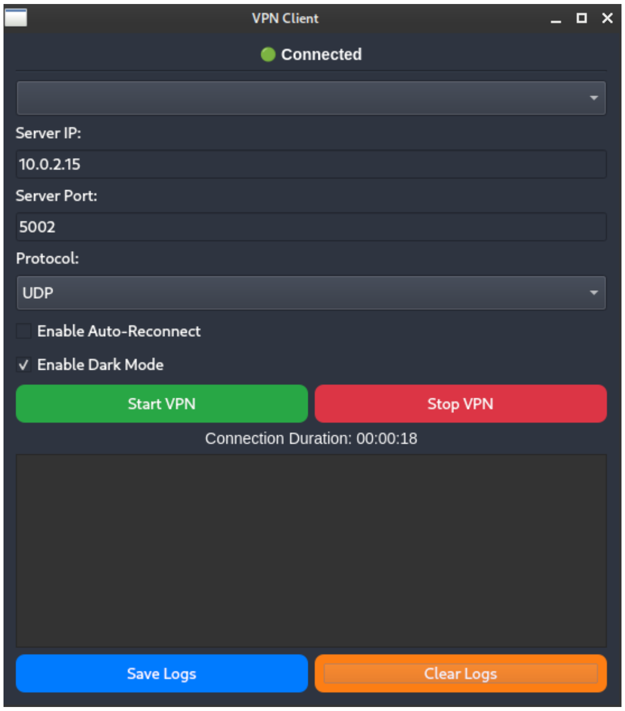
VI. Screenshot

VII. Conclusion

This project demonstrates the practical implementation of a secure VPN system integrated with a modern GUI. The modularity and usability of the tool make it suitable for both educational and real-world applications. Future enhancements may include dynamic IP filtering, support for WireGuard, and mobile app integration.

References

[1] OpenSSL Project, https://www.openssl.org/.
[2] PyQt5 Documentation, https://doc.qt.io/qtforpython/.
[3] Cryptography Python Package, https://cryptography.io/.



Server IP: 10.0.2.15

Server Port: 5002

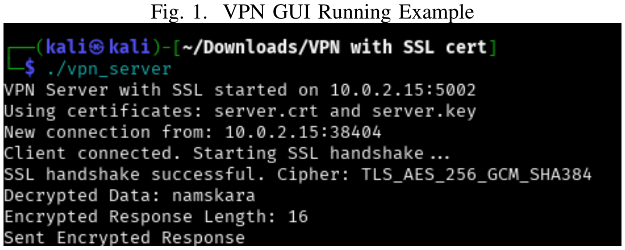
Protocol: UDP

□ Enable Auto-Reconnect

☑ Enable Dark Mode

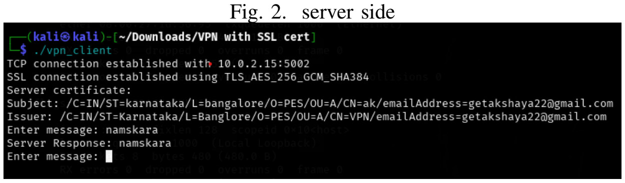
Connection Duration: 00:00:18

Fig. 1. VPN GUI Running Example



```
┌──(kali㉿kali)-[~/Downloads/VPN with SSL cert]
└─$ ./vpn_server
VPN Server with SSL started on 10.0.2.15:5002
Using certificates: server.crt and server.key
New connection from: 10.0.2.15:38404
Client connected. Starting SSL handshake...
SSL handshake successful. Cipher: TLS_AES_256_GCM_SHA384
Decrypted Data: namskara
Encrypted Response Length: 16
Sent Encrypted Response
```

Fig. 2. server side



```
┌──(kali㉿kali)-[~/Downloads/VPN with SSL cert]
└─$ ./vpn_client
TCP connection established with 10.0.2.15:5002
SSL connection established using TLS_AES_256_GCM_SHA384
Server certificate:
Subject: /C=IN/ST=karnataka/L=bangalore/O=PES/OU=A/CN=ak/emailAddress=getakshaya22@gmail.com
Issuer: /C=IN/ST=Karnataka/L=Banglore/O=PES/OU=A/CN=VPN/emailAddress=getakshaya22@gmail.com
Enter message: namskara
Server Response: namskara
Enter message: █
```

Fig. 3. client side