TLS based VPN Project - 2025.3.20

In this project, you'll design and implement a simple, secure VPN based on TLS protocol. The VPN uses a client-server architecture, tunnels traffic over TLS 1.3, and handles DNS securely. The project is split into stages, each building on the previous one, allowing you to learn networking, encryption, and system programming step-by-step.

Tools and Prerequisites

- Golang: Install from golang.org.
- Operating Systems: macOS/windows (client), Linux (server).
- Dependencies: Install using go get as needed.
- Root Access: Required for TUN interfaces and routing.
- Text Editor: VS Code, or similar.

Project Goals

1. Client: Capture all traffic via TUN and tunnel it to the server over TLS 1.3.
2. Server: Decrypt traffic, forward it to the internet via NAT.
3. DNS: Securely handle DNS requests to prevent leaks.

Stage 1: Setting Up a TLS 1.3 Connection

Objective: Establish a secure connection between client and server using TLS 1.3.
Tasks:
1 Generate a Self-Signed Certificate:
- Run: openssl req -x509 -newkey rsa:2048 -keyout server.key -out server.crt -days 365 -nodes. (Try ECC based certificate for better security)
- Fill in prompts (e.g., country, organization); these can be arbitrary for testing.
2 Server Code (server.go): package main

Run and Test:
- Start the server: go run server.go (on Linux).
- Then run the client: go run client.go.
- Check server logs for a connection message.
Learning Outcomes
- Understand TLS 1.3 and its role in secure communication.
- Learn basic client-server networking in Golang.

Stage 2: Capturing Traffic with TUN (Client)

Objective: Capture all traffic on the client using a TUN interface and send it to the server.

Tasks
1 Install Dependency:
- Run: go get github.com/songgao/water.
2. Update client.go
3. Configure TUN Interface:
- Run as root: sudo go run client.go.
- Manually configure: sudo ifconfig utunX 10.0.0.2 10.0.0.1 up (replace utunX with your TUN name from logs).
4. Check Server logs for received bytes (it won't process them yet)
Learning Outcomes

- Learn how TUN interfaces capture network traffic.
- Understand goroutines for concurrent traffic tunneling.

Stage 3: Server Traffic Echo

Objective: Make the server echo received traffic back to the client for testing.
Tasks
1 Update Server Code (server.go).
2. Test:
- Run the server: go run server.go.
- Run the client from Stage 2.
- Generate traffic (e.g., ping www.baidu.com) and check server logs.
Learning Outcomes
• Understand how to read and write network data in Golang.
• Test end-to-end traffic flow.

Stage 4: Server NAT Forwarding

Objective: Forward client traffic to the internet using NAT on the server.
Tasks
1 Update Server code(server.go).
2. Set Up NAT:
- Run: sudo iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE (replace eth0 with your server's interface).
3. Test:
- Run as root: sudo go run server.go.
- Note: Full forwarding requires additional work (e.g., TUN on server).
Learning Outcomes
• Learn about IP forwarding and NAT in Linux.
• Understand the basics of traffic routing.

Submission

You can follow the above 4 stages to code your VPN program, which should be easier. Submit your code, along with a brief report explaining necessary details, especially including the demonstration of packet analysis from wireshark or tcpdump to show the effectiveness of your VPN.