Project (CP3)
A Secure Multi-Client TCP Socket Application

Network Security

(CS3403)

24 Feb 2025

Date of Submission: 6th Apr 2025 (Sun)

Description: Build a **client-server application** of your choice using a secure TCP socket connection.
Note: This **project** will be evaluated for **15 marks** (**CP3**) as per course design document. The remaining **10 marks (CP3)** will be based on the Lab submissions from Lab 2 onwards.
Ref: All the labs done from Lab 0 to Lab 3

You have been provided with the sample implementation of secure TCP sockets both in C and Python.

You are required to use the C language for the Server implementation and use the POSIX APIs to create threads to handle different client connections (**Ref Lab 3**) with a single Server process running on the WSL or Ubuntu VM (for Mac users).

You are free to choose either Python or Java on the client side where you can develop GUI for user interactions with your application for configuring it or to interact with it.

You are required to integrate the following in your application:

1. GUI to interact with your applications in two modes, admin and user.
2. Have login for admin and users.
3. Account creation and maintenance using a database (MySQL is fine or any other DB you are familiar with)
4. User specific preferences and data maintained by the Server.
5. Take any sample application where multiple users login for a service from the server.
6. Demonstrate user account management.
7. Cloud integration is optional.
8. Purpose is to use the knowledge that you have gained through different courses so far in developing one application.

Detailed guidelines and evaluation criteria are provided below.

Guidelines for the project design and implementation:

1. Choose an application where the User data is maintained and exchanged over the Internet between the Server and Client through a secure connection.
2. Bring out innovative design ideas in terms of data structures, execution efficiency, data handling, session keys handling by the server, etc.
3. **Use of both Asymmetric and symmetric keys** for effective and secure communication between the Server and the client.
   Developing a server-client application that utilizes asymmetric encryption for establishing secure connections and then transitions to symmetric encryption for efficient bulk data transfer is a common and effective approach in **network security**. This method leverages the strengths of both encryption types: the security of asymmetric encryption for key exchange and the speed of symmetric encryption for data transmission.
   Ref: Use of **session keys** **(or secret key** using **symmetric encryption)** by the server with each client.
4. You are encouraged to have a simple GUI on the client side to interact with your application server.

Evaluation criteria:

1. You need to submit the sources files of your project both Server and client side organized into directory structure, as a zip file.
2. Write a project report by providing the screenshots of client-side usage and interactions with your server. Explaining the supported features of your application.
3. Document your design and add a section on the testing you have done to validate your application.
4. Highlight the secure features of your application and how the user data maintained by the server is protected from illegal access or tampering.
5. You have to demonstrate running of your application during the viva.
6. You will be asked to explain and make any quick changes to either on the Server or Client side to add a feature or modify some configuration parameter or make changes to one of the features supported by your application during the viva.
7. **You need to be able to explain every line of code in your project.**
8. All the very best!!!

The main idea of this project component is to give you hands-on training on implementing a secure application.

You are required to provide the project title and other implementation details in the excel sheet shared here, by 28th Feb (Fri) midnight.

Other Challenging Project Ideas

You are free to take up any of these topics as well if you like challenges and want to explore the unknown. Study and analyze each of the ideas well and take a calculative risk to take up these project ideas or something else in a similar line.

CAUTION: You are free to choose after analysing well about the effort involved and what needs to be implemented to build it.

1. Network Traffic Analysis and Intrusion Detection System (IDS)

- **Description**: Build a simple system to monitor and analyze network traffic to detect potential intrusions or malicious activity.
- **Tools**: Wireshark (for packet capturing), Python (for analysis), Snort (IDS).
- **Objective**: Capture network traffic, identify patterns, and flag suspicious activities (like DDoS attacks, port scans, etc.).
- **Features**:
  - Capture and analyze packets.
  - Detect potential network anomalies or malicious traffic.
  - Generate alerts for suspicious activity.

2. Password Cracking Simulation

- **Description**: Develop a small program to demonstrate how password cracking can be done using brute force or dictionary attacks.
- **Tools**: Python, Hashlib, and wordlist files for dictionary attacks.
- **Objective**: Simulate a password cracking attack on encrypted passwords and show how weak passwords can be cracked.
- **Features**:
    - Implement brute force and dictionary-based attacks.
    - Demonstrate the importance of strong passwords.
    - Show password hashing and the vulnerabilities.

3. Firewall Implementation

- **Description**: Build a simple firewall using either hardware or software.
- **Tools**: Python, iptables (Linux), or pfSense.
- **Objective**: Create a basic firewall system to control incoming and outgoing network traffic based on predefined rules.
- **Features**:
    - Allow/block specific IP addresses, ports, or protocols.
    - Implement packet filtering and logging of denied traffic.
    - Monitor firewall performance.

4. SSL/TLS Encryption for Web Communication

- **Description**: Implement a basic secure communication system using SSL/TLS to encrypt data between a client and a server.
- **Tools**: OpenSSL, Python's `ssl` module.
- **Objective**: Demonstrate secure data transfer between client and server, protecting against eavesdropping and man-in-the-middle attacks.
- **Features**:
    - Set up a secure HTTP server (HTTPS).
    - Encrypt and decrypt data using SSL/TLS certificates.
    - Show the importance of certificate validation.

5. Network Vulnerability Scanner

- **Description**: Build a basic tool to scan networks and identify common vulnerabilities (e.g., open ports, outdated software, etc.).
- **Tools**: Python (Socket programming), OpenVAS, or Nmap.
- **Objective**: Scan a network for potential security weaknesses like open ports, misconfigurations, or unpatched software.
- **Features**:
    - Identify open ports and services.
    - Detect common vulnerabilities (CVEs).
    - Provide a report with recommended fixes.

6. Man-in-the-Middle (MITM) Attack Simulation

- **Description**: Create a demo showing a Man-in-the-Middle attack where an attacker intercepts and alters communication between two parties.
- **Tools**: Wireshark, Ettercap, or MITMf.

- **Objective**: Show how MITM attacks work and how attackers can intercept or alter traffic.
- **Features**:
    - Intercept communication between two devices.
    - Demonstrate SSL stripping or DNS spoofing.
    - Educate on how to prevent such attacks.

7. Network Segmentation and Virtual LANs (VLANs)

- **Description**: Implement network segmentation using VLANs to isolate different parts of a network for improved security.
- **Tools**: Cisco Packet Tracer, GNS3, or real routers and switches.
- **Objective**: Demonstrate how VLANs can enhance security by isolating traffic between departments or types of devices.
- **Features**:
    - Set up multiple VLANs.
    - Configure inter-VLAN routing.
    - Show the security benefits of network segmentation.

8. Secure File Sharing Using Encryption

- **Description**: Develop a secure file-sharing system where files are encrypted before transmission and decrypted by the receiver.
- **Tools**: Python, PyCryptodome.
- **Objective**: Ensure that files shared over a network are encrypted and safe from unauthorized access.
- **Features**:
    - File encryption and decryption.
    - Secure file transfer protocol (e.g., SFTP, SCP).
    - Use of symmetric and asymmetric encryption algorithms.

9. VPN (Virtual Private Network) Setup and Implementation

- **Description**: Create a simple VPN service to secure communication between devices over an untrusted network.
- **Tools**: OpenVPN, WireGuard.
- **Objective**: Set up a private network over the internet, ensuring privacy and security for data transmission.
- **Features**:
    - Establish a secure VPN tunnel.
    - Encrypt traffic between client and server.
    - Demonstrate secure remote access to internal resources.

10. Phishing Attack Simulation

- **Description**: Develop a system that simulates a phishing attack, demonstrating how attackers trick users into providing sensitive information.
- **Tools**: Kali Linux, SET (Social-Engineer Toolkit).
- **Objective**: Show how phishing works and raise awareness on avoiding such attacks.
- **Features**:
    - Simulate fake login pages.

  - Use social engineering techniques to collect user information.
  - Demonstrate how phishing emails are crafted.

11. Security of Internet of Things (IoT) Devices

- **Description**: Analyze the security of IoT devices by identifying common vulnerabilities and providing solutions.
- **Tools**: Kali Linux, Nmap, IoT devices.
- **Objective**: Explore the security flaws in IoT devices (e.g., weak passwords, unencrypted data) and suggest remedies.
- **Features**:
  - Scan IoT devices for vulnerabilities.
  - Analyze device data transmission for encryption.
  - Demonstrate IoT security practices.

****