Lab 1: Socket Programming – Stateful Client–Server Chat Application

Course: FSCT 8561 – Security Applications
Instructor: Dr. Maryam R. Aliabadi
Lab Duration: 3 Hours

Overview

This lab builds directly on Lab 0. Students are expected to reuse and extend their existing Python socket code. Rather than repeating basic socket operations, this lab introduces application-level protocol design, session state, robustness, and deeper security analysis.

Learning Objectives

- Design a simple application-layer protocol on top of TCP
- Maintain session state across multiple client messages
- Implement robust input validation and error handling
- Analyze security implications of stateful network applications

Pre-requisite

Completed Lab0

Required Reading & Tutorials

- Mastering Python for Networking and Security – **Chapter 3**
    - https://learning.oreilly.com/library/view/mastering-python-for/9781839217166/
    - Source code on: https://github.com/PacktPublishing/Mastering-Python-for-Networking-and-Security-Second-Edition
- Socket programing tutorials
    - Official Python Documentation – socket Module
      https://docs.python.org/3/library/socket.html
    - Real Python – Socket Programming in Python
      https://realpython.com/python-sockets/
    - GeeksforGeeks – Socket Programming in Python
      https://www.geeksforgeeks.org/socket-programming-python/
    - DigitalOcean – How To Use Sockets in Python 3

Lab Scenario

You will implement a stateful client–server chat service. Clients must identify themselves using a username, communicate using a defined message protocol, and disconnect cleanly. The server enforces protocol rules and maintains session state. Please read all requirements before starting the implementation.

Part 1 – Application-Level Message Protocol

All messages must follow this format:

COMMAND|DATA

HELLO|username → Client introduces itself
MSG|text → Client sends a chat message
EXIT| → Client requests disconnection

The server must reject any message sent before a valid HELLO command and respond with OK| or ERROR|reason messages.

Part 2 – TCP Server Requirements

Support persistent connections (multiple messages per client)
Maintain session state (username, connection status)
Validate all incoming messages
Handle malformed input without crashing
Log connections, disconnections, and errors

Part 3 – TCP Client Requirements

Prompt user for a username
Send HELLO upon connection
Allow multiple messages per session
Display server responses clearly
Exit cleanly using EXIT command

Part 4 – Robustness and Error Handling

You must demonstrate handling of at least three of the following:

Empty messages
Messages exceeding a fixed length
Invalid command format
Client sends data before HELLO

- Unexpected server or client disconnect

Part 5 – Reflection Questions

1. What happens if the server crashes while the client is connected?
2. How can the server handle multiple clients?
3. Why can recv(1024) split messages unexpectedly?
4. How would you add basic security (authentication, encryption) to this chat application?

Part 6 – Security Analysis

In 300–400 words, analyze the security implications of your design. Your analysis must reference specific parts of your implementation and address:

New attack surfaces introduced by protocol and state
Potential abuse scenarios
Why TCP does not provide security guarantees
Proposed mitigations for a real deployment

Deliverables

`server.py`
`client.py`

- A single recording demonstrating successful client–server communication and robustness testing (mp3 format)

Security analysis and reflection report

- Submit **one PDF file only**. This PDF is the Security analysis and Reflection Report and must include cloud links to all required technical artifacts (code base, recordings.).

Filename format:

Lab1-FirstName-LastName-StudentNumber.pdf

Good Luck!