MINI PROJECT FOR COMPUTER AND NETWORK SECURITY

Secure FTP

Submitted to

Sirindhorn International Institute of Technology Thammasat University

April 2025

By
Sébastien Nourry - 6722808077
Samuel Liesch - 6722808341
Dhanabhat Lertarpakul - 6522770625
Peerawat Wongthai - 6522780673

Contents

1 Project Concept

1.1 Project Summary

The goal of this project is to implement a simplified but secure version of the FTP protocol using low-level TCP sockets in Python. The project is designed to give students real-world insight into both networking principles and modern cybersecurity practices. The server is built around the `select()` system call, allowing asynchronous and scalable client handling.

Key security features include role-based access control, directory isolation, password hashing, and anomaly detection. A honeypot is also implemented as a bonus feature, offering a way to mislead attackers and collect behavioral data for further analysis, as this part is a bonus, we do not guarantee to have it for the due date.

1.2 Typical Usage

Upon launch, the Secured FTP server waits for incoming TCP connections on a designated port. Users authenticate using a username and password. Based on the user's role and permissions, the server allows access to a personal or shared directory with read and/or write permissions.

Example usage scenarios:

- An authenticated user uploads or downloads files within their personal sandboxed directory.
- An admin user accesses broader server logs.
- A hacker attempts login with suspicious credentials, triggering the honeypot system.

1.3 Main Challenges

The project poses several technical and security challenges:

- Implementing non-blocking I/O to manage multiple clients simultaneously using `select()`.
- Designing a secure authentication system using hashed passwords.
- Restricting user access to specific directories (sandboxing).

- Monitoring user actions to detect suspicious patterns such as brute-force attacks or path traversal attempts.
- Managing session stability while integrating optional encryption or logging features.
- Designing and integrating a honeypot that does not interfere with legitimate operations.

2 Project Requirements

2.1 System Description

The system is a multi-client FTP server using TCP sockets. It uses the `select()` system call to manage concurrent client connections, ensuring that each user's commands are handled asynchronously.

Key features include:

- Command parsing and response handling
- Support for FTP commands like `LIST`, `RETR`
- Secure authentication with SHA-256 hashed passwords
- Role-based access control (Admin, User)
- Logging of all interactions for audit and intrusion detection
- Honeypot mode for suspicious behavior

2.2 Computational Tasks

The main computational tasks of the server are:

- Managing multiple client sockets simultaneously via `select()`
- Parsing and interpreting FTP commands

- Reading/writing files in isolated directories
- Hashing and comparing user passwords
- Detecting anomalies through login attempts and command frequency
- Recording logs with timestamps and user metadata
- Optionally compressing or encrypting files before transmission

2.3 Use Cases and Tests

Use Case 1: **Standard User Login and File Transfer**

> A valid user connects to the server, authenticates, lists available files, uploads and downloads documents within their sandbox.

Use Case 2: **Permission Violation Attempt**

> A user attempts to access a restricted directory. The server denies the request and logs the action.

Use Case 3: **Brute-force Login Detection**

> An attacker attempts multiple incorrect logins. After a threshold, the server delays response time and logs the source IP.

Use Case 4: **Honeypot Trigger**

> A user tries to log in with a suspicious username (e.g., `root`, `admin123`). The server redirects them to a fake environment and logs every interaction.

Test methods will include:

- Stress tests for multiple concurrent users
- Penetration testing using simulated attacks
- Functional testing of logging and permission systems

3 Algorithm Design and Implementation

3.1 Algorithm Design

The architecture is designed around a `select()` loop that:

- Initializes the listening socket
- Adds the socket to the monitored fd_set
- Accepts new connections when available
- Parses and processes commands from each active client

Each user session is isolated with its own state, including:

- Username
- Role and permissions
- Current working directory
- Failed login attempts

The command handler interprets FTP commands, checks permissions, and executes filesystem operations accordingly.

The honeypot mechanism is based on pattern recognition: certain usernames, behaviors or commands trigger a redirection to a mock file system and detailed logging (not fully implemented yet).

3.2 Implementation

The server is implemented in Python using the standard POSIX socket API. It uses:

- `select()` to monitor the listening socket and connected clients
- `read()` and `write()` for I/O
- `SHA-256` (via OpenSSL or custom) for hashing passwords

- A simple Json configuration file for defining users and roles
- Logging via standard file I/O in append mode

4 Useful resources

you can find further informations in the README.md file or directly access the Github repository of our project via this link: https://github.com/nourrysebastieN/secured_ftp