Report on the Development and Implementation of a Secure FTP Client-Server Application

M. Bharat Kumar
CS22B2038
Department of Computer Science and Engineering (Artificial Intelligence)

November 6, 2024

Contents

Abstract

This report provides a comprehensive overview of the design, development, and implementation of a secure FTP client-server application. The system integrates SSL/TLS encryption protocols to guarantee the confidentiality and integrity of file transfers. In addition to the core features of the FTP protocol, this document also discusses the resolution of technical challenges encountered during development, particularly addressing warnings related to buffer overflow and string truncation that arose during the construction of FTP command strings.

1 Introduction

The objective of this project was to develop an FTP client-server application capable of securely transferring files between a client and a server. The implementation of Secure Socket Layer (SSL) and Transport Layer Security (TLS) protocols ensures encrypted communication, safeguarding the data during transmission. The client interacts with the server to execute operations such as uploading and downloading files, as well as listing the contents of remote directories. On the other hand, the server is responsible for processing client commands, managing file transfers, and ensuring secure communication.

2 Project Structure

The project is composed of two primary components:

- **FTP Client:** The client component facilitates interaction with the user by receiving file transfer commands, formatting them appropriately, and sending them to the server via a secure SSL/TLS connection. The client supports various functions, including listing files, uploading files, and downloading files.
- **FTP Server:** The server listens for client connections, processes incoming FTP commands, and manages file transfers. SSL/TLS encryption is employed to secure all data transmissions, ensuring confidentiality and integrity.
- **SSL/TLS Encryption:** Both the client and server use SSL/TLS protocols to encrypt the data transmitted between them, providing secure communication channels and preventing unauthorized access during file transfer operations.

3 Problem Statement

The primary goal of this project was to design and implement a secure FTP client-server system. A notable challenge encountered during development was addressing security vulnerabilities such as buffer overflows and string truncation, which can occur when user input is mishandled during the construction of FTP command strings.

4 Development Challenges and Solutions

4.1 Buffer Overflow Warning

During the development of the FTP client, several buffer overflow warnings were triggered, particularly when constructing the `RETR` and `STOR` FTP commands. These warnings were generated by the `snprintf` function, indicating that the length of the formatted command string might exceed the allocated buffer size, resulting in potential data truncation.

4.2 Root Cause Analysis

The issue originated from the use of the `snprintf` function to format user input into FTP command strings, such as `RETR <filename>` or `STOR <filename>`. Given the variable length of the user-supplied filenames, there was a risk that the constructed string could exceed the allocated buffer size. The buffer was set to 1024 bytes, but after accounting for the fixed portion of the command (e.g., `"RETR "` or `"STOR "`), there was insufficient space to accommodate longer filenames.

4.3 Solution and Code Refinement

To resolve this issue, the buffer size was increased to accommodate the maximum possible size of the formatted FTP command string. Specifically, a constant `COMMAND_SIZE` was defined to ensure that the command string would always fit within the available buffer space. The revised code for formatting the `RETR` and `STOR` commands is as follows:

```
#define COMMAND_SIZE (BUFFER_SIZE - 5)  // Leave space for "RETR " or "STOR "

// For RETR command
snprintf(buffer, COMMAND_SIZE, "RETR %s", command);

// For STOR command
snprintf(buffer, COMMAND_SIZE, "STOR %s", command);
```

This adjustment ensures that the command strings will never exceed the buffer size, thereby preventing buffer overflow or string truncation.

5 Functional Workflow and Features

5.1 FTP Client Workflow

The FTP client performs the following operations:

- **Menu Options:** Upon establishing a connection, the client presents the user with a set of options, including the ability to list files, download files, upload files, and terminate the session.
- **Listing Files:** The client sends the `LIST` command to the server, which responds by listing the files available for download.
- **Downloading Files:** To download a file, the user provides the filename, and the client issues the `RETR <filename>` command. The server responds by sending the file's content, which the client stores locally.
- **Uploading Files:** To upload a file, the client sends the `STOR <filename>` command along with the file's content. The server receives the file and stores it in the appropriate directory.

5.2 Server-Side Operations

The FTP server listens for incoming client connections and processes FTP commands as follows:

- **File Transfer Requests:** The server processes the `RETR` and `STOR` commands, either sending the requested files to the client or receiving and storing uploaded files.

- **SSL/TLS Encryption:** All communication between the client and server is encrypted using SSL/TLS to ensure the security of sensitive data, such as credentials and file contents, during transmission.

6 Conclusion

In conclusion, this project successfully developed a secure FTP client-server application utilizing SSL/TLS encryption to ensure the protection of file transfers. By addressing the buffer overflow warnings and refining the process of formatting FTP commands, the project achieved a reliable and secure file transfer system. The lessons learned during the development process, particularly regarding buffer management and security, highlight the importance of robust string handling and encryption protocols in secure networked applications.