Faculty of Information Technology
Brno University of Technology

ISA

Network Applications and Network Administration

IMAP client with TLS support

Andrii Bondarenko (xbonda06)

November 18, 2024

Contents

1 Introduction

This project focuses on implementing an IMAP (Internet Message Access Protocol) client with support for secure connections using SSL/TLS. The application allows users to connect to email servers, download emails, and save them locally. The main goal of the project was to develop a robust client that supports both unencrypted and encrypted connections, adhering to the IMAP4rev1 standard (RFC 3501).

2 Problem Statement

In the current digital age, email remains a crucial communication tool. However, securing email communication is vital due to potential security threats like eavesdropping and data breaches. The objective of this project was to develop an IMAP client that supports secure connections over SSL/TLS, providing users with a way to safely access their emails.

3 Design of the Application

The application is designed using the C++ programming language and follows modern design patterns for maintainability and scalability. The key components of the application are:

3.1 Modules Overview

- **ArgParser**: Handles parsing of command-line arguments.
- **IMAPClient**: The core class responsible for interacting with the IMAP server.
- **ConnectionStrategy**: Strategy pattern used to switch between TCP and SSL connections.
- **SSLWrapper**: A wrapper for managing SSL connections using OpenSSL.
- **IMAPCommandFactory**: Factory pattern used to generate main IMAP commands like LOGIN, SELECT, SEARCH, FETCH and LOGOUT.

3.2 Design Patterns

The application utilizes several design patterns:

- **Strategy Pattern**: Allows switching between unencrypted (TCP) and encrypted (SSL/TLS) connections.
- **Singleton Pattern**: Used in the ‘SSLWrapper‘ class to manage a single SSL context.
- **Factory Pattern**: The ‘IMAPCommandFactory‘ generates IMAP commands simplifying command creation and improving code maintainability.

4 Implementation Details

The implementation of the IMAP client is structured using modern C++ features and design patterns to ensure modularity, scalability, and maintainability. Below is a detailed overview of the key components and how they were implemented.

4.1 Command-Line Arguments Parsing

The `ArgParser` class is responsible for handling the parsing of command-line arguments. It extracts and validates the parameters provided by the user, such as the server address, port number, and SSL-related options. The program uses 'getopt' for argument parsing.

- `<server>`: The domain or IP address of the IMAP server.
- `-p port`: Specify the server port (default: 143 for non-TLS, 993 for TLS).
- `-T`: Enable SSL/TLS encryption.
- `-c certfile`: Specify a certificate file to use for SSL/TLS verification.
- `-C certdir`: Specify a directory containing certificates for SSL/TLS verification (default: `/etc/ssl/certs`).
- `-n`: Download only new messages.
- `-h`: Download only message headers.
- `-a auth_file`: Path to the file containing authentication credentials.
- `-o out_dir`: Output directory where emails will be saved.

The `ArgParser` is also responsible for validating the authentication file. It ensures that the file contains both the username and password in the correct format:

4.2 Configuration File

The authentication file (`auth_file`) must contain the username and password in a simple format as follows:

```
username = your_username   password = your_password
```

The file should be a Unix text file, with each line terminated by a newline character (`\n`).

4.3 Network Connection Strategies

The application employs the **Strategy Pattern** to handle both encrypted (SSL/TLS) and unencrypted (TCP) connections. This is achieved through two classes:

- `TCPConnectionStrategy`: Manages unencrypted connections over port 143 by default.
- `SSLConnectionStrategy`: Manages secure connections over port 993 using OpenSSL. It supports loading custom certificate files and directories for SSL verification.

The `IMAPClient` class dynamically selects the appropriate strategy based on the command-line arguments. This design allows for flexible switching between secure and insecure connections without modifying the core client logic.

4.4 IMAP Client Functionality

The `IMAPClient` class is the core component of the application, responsible for interacting with the IMAP server. It performs the following tasks:

- **Connecting to the server**: Uses the selected connection strategy (TCP or SSL) to establish a connection with the IMAP server.
- **Authentication**: Authenticates the user by sending a `LOGIN` command with credentials extracted from the authentication file.
- **Selecting the mailbox**: Uses the `SELECT` command to choose the mailbox specified by the user (default is `INBOX`).
- **Searching for messages**: the client sends a `SEARCH` command to retrieve the UIDs of messages to fetch.
- **Fetching messages**:
  - For new messages (`-n` flag), the client sends individual `FETCH` commands for each UID.
  - For all messages, a single `FETCH` command with the range `1:*` is used to fetch all messages at once.

  The response from the server is then processed, and messages are saved as separate files.
- **Saving messages**: Each email is saved in the specified output directory. If the directory does not exist, it is created automatically. The client checks if a message has already been saved in the output directory. If a message is already present, it is skipped.

4.5 File Naming and Subject Decoding

One of the key features of the `IMAPClient` is the ability to generate meaningful filenames for saved emails. The naming convention used is:

`msg_<UID>_<Subject>`

Steps for generating filenames:

- Each message is assigned a filename based on its UID.
- The `Subject` field of the email is decoded to make it human-readable.
- Special characters in the `Subject` are sanitized to ensure compatibility with the file system.

Decoding the Subject:

- The `extractAndDecodeSubject` method handles decoding of the `Subject` field. It supports decoding both `Base64` and `Quoted-Printable` encoded subjects, following the IMAP protocol standards.
- If the `Subject` is missing or cannot be decoded, a default value of `no_subject` is used.

This approach ensures that saved emails have descriptive filenames, making it easier for users to identify and organize their downloaded emails.

4.6 IMAP Command Handling

The client uses the **Factory Pattern** through the `IMAPCommandFactory` class to generate various IMAP commands. The factory is responsible for creating instances of commands such as `LOGIN`, `SELECT`, `FETCH‘`, and `LOGOUT`. This encapsulates the command creation process, ensuring that the IMAP protocol syntax is correctly followed.

4.7 SSL/TLS Management

The `SSLWrapper` class is a singleton responsible for managing the initialization and handling of SSL contexts using OpenSSL. It provides methods to:

- Initialize the SSL library and set up the context.
- Load SSL certificates for server verification.
- Create and manage secure sockets for encrypted communication.
- Send and receive data securely over the network.

4.8 Error Handling and Exceptions

The client is designed to handle various error conditions gracefully. Custom exceptions, defined in `IMAPExceptions.h`, capture issues related to network communication, SSL, and IMAP protocol violations. This ensures that the client can provide informative error messages and terminate gracefully in the event of unexpected conditions.

5 Usage Instructions

To compile the project, run:

```
make
```

To execute the client, use:

```
./imapcl server [-p port] [-T [-c certfile] [-C certaddr]]
                [-n] [-h] -a auth_file [-b MAILBOX] -o out_dir
```

5.1 Examples

```
./imapcl eva.fit.vutbr.cz -o maildir -a cred
Saved 246 messages from the INBOX.

./imapcl eva.fit.vutbr.cz -T -p 993 -c cert.pem -C /etc/ssl/certs -a cred -o maildir -n
Saved 2 messages from the INBOX.

./imapcl eva.fit.vutbr.cz -o maildir -a /dev/null
Error: Bad auth file format
```

6 Testing

Extensive testing was conducted to ensure the stability and security of the application. The tests included:

- Connection to various IMAP servers (both with and without SSL).
- Downloading and saving emails.
- Handling invalid inputs and network errors.

7 Results

The IMAP client successfully connects to both encrypted and unencrypted IMAP servers, downloads emails, and saves them in the specified directory. The client handles errors gracefully and provides clear messages to the user.

8 Conclusion

The project demonstrated the importance of secure email communication and provided a solution to access emails safely over the IMAP protocol. The use of modern design patterns ensured the flexibility and maintainability of the code.

9 References

[1] Crispin, M. *INTERNET MESSAGE ACCESS PROTOCOL - VERSION 4rev1* https://datatracker.ietf.org/doc/html/rfc3501. 2003. Accessed: 2024-11-18.

[2] Dierks, T. and Rescorla, E. *The Transport Layer Security (TLS) Protocol Version 1.2* https://datatracker.ietf.org/doc/html/rfc5246. 2008. Accessed: 2024-11-18.

[3] Foundation, M. *Using IMAP with Email Clients* https://support.mozilla.org/en-US/kb/imap-synchronization. 2024. Accessed: 2024-11-18.

[4] Foundation, O. *OpenSSL Documentation* https://www.openssl.org/docs/. 2024. Accessed: 2024-11-18.

[5] Gamma, E.; Helm, R.; Johnson, R. and Vlissides, J. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley Professional, 1994. ISBN 978-0201633610.

[6] Matoušek, P. *Síťové služby a jejich architektura*. Publishing house of Brno University of Technology VUTIUM, 2014. 396 p. ISBN 978-80-214-3766-1. Available at: https://www.fit.vut.cz/research/publication/10567.