VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

Síťové aplikace a správa sítí – Project

IMAP client with TLS support

November 10, 2024

Matúš Ďurica

Contents

Introduction

This project aimed to create a terminal-based IMAP client with TLS support. After execution, the client fetches messages stored on the server and stores them locally one by one. Its output is a message which tells the user how many messages were fetched and stored. The user can tweak the functionality further using command-line arguments.

1 Theory

1.1 Internet Message Access Protocol

1.2 Internet Message Format

1.3 Transmission Control Protocol

1.4 Secure Sockets Layer and Transport Layer Security

2 Implementation details

2.1 Return Codes

The application provides these return codes to narrow down errors during execution.

0	–	General success
1	–	General failure
2	–	Missing server argument
3	–	Missing required argument
4	–	Missing option for an argument
5	–	Unknown argument
6	–	Socket creating failed
7	–	Connecting to the socket failed
8	–	Bad host
9	–	Error while opening authentication file
10	–	Authentication file does not exist
11	–	Invalid credentials
12	–	Missing credentials
13	–	Output directory does not exist
14	–	Error while opening validity file
15	–	Can not access mailbox
16	–	Writing to a socket failed
17	–	Invalid response from the server
18	–	Certificate file or directory error
19	–	Error while creating the SSL context
20	–	Error while creating the SSL connection
21	–	Error while setting the socket descriptor to the SSL
22	–	Error while performing the SSL handshake

2.2 Argument parsing

CLI argument parsing is done using the `getopt` function provided by the `getopt.h` header file.

This function is utilized by the `Utils::CheckArguments` method, which checks all arguments received from the user. When the user inputs invalid or unknown arguments, an error message will be printed, and the application will exit with a relevant return code.

2.3 Sessions

2.3.1 Session

Class `Session` contains an interface for plain text communication with the IMAP server.

It utilizes the functionality of a BSD socket for communication with the IMAP server. Reading and writing to the socket are done by the `recv` and `send` functions, respectively. The communication itself is in plain text; therefore, it is insecure.

It is also a base class for the `EncryptedSession` class.

2.3.2 Encrypted Session

Class `EncryptedSession` contains an interface used for encrypted communication with the IMAP server.

It utilizes functions from the `openssl.h` header file to encrypt a BSD socket, which is then used to send or read encrypted data using the `SSL_write` and `SSL_read` functions, respectively.

Encrypting the BSD socket consists of creating a secure SSL context using `SSL_CTX_new` function, which takes a `method` parameter provided by the `TLS_client_method` function [3]. This is a general-purpose version-flexible SSL/TLS method. The actual protocol version used will be negotiated to the highest version mutually supported by the client and the server [4]. After creating the secure SSL context, the certificate directory (and the certificate itself, if the user supplied the path to it) is specified to the context using `SSL_CTX_load_verify_dir` (`SSL_CTX_load_verify_file`) [2]. Then, the SSL structure that holds the data for an SSL/TLS connection is created using the `SSL_new` function [5]. Then, setting the BSD socket file descriptor to the SSL context is done using `SSL_set_fd` function [6], and finally, the socket is ready to be connected using `SSL_connect` function [1].

2.4 Messages

2.4.1 Message

Class `Message` defines an object for the email message. It is a base class to the `HeaderMessage` class. The file name follows this convention:

```
<UID>_<Mailbox>_<Hostname>_<Subject>_<Sender>_<Hash>.eml
```

UID	–	UID of the message
Mailbox	–	Mailbox, where the message is located
Hostname	–	Hostname of the IMAP server
Subject	–	Subject of the message
Sender	–	Sender message address
Hash	–	Hashed Message-ID of the message

The message body is extracted from the server response using the `std::string::substr` method - as a first thing, the `RFC822.SIZE` is fetched from the server, representing the number of octets in the message. When parsing the message body, the first thing being done is applying apply a regular expression to remove the first line of the response, which contains the beginning of the IMAP response, which is not a part of the message body. Then the `RFC822.SIZE` is passed to the `substr` method to get the number of octets of the message. This substring is then stored as the message body.

It also provides methods for setting the file name and content and saving it to a directory.

2.4.2 Header Message

Class `HeaderMessage` defines an object of a message containing only the headers of an email message.

The file name follows the same convention as above, with the only difference being adding `_h` after the hashed Message-ID:
`<UID>_<Mailbox>_<Hostname>_<Subject>_<Sender>_<Hash>_h.eml`

Same as for the `Message` class, the headers are extracted from the IMAP response using a regular expression, which removes the first line of the IMAP response and making a substring based on the number of the octets of the headers. The only difference is that the number of the octets is extracted from the fetch response using another regular expression, as the RFC3501 protocol does not provide a way to obtain a number specifying the size of the headers.

3 Running the Application

3.1 Compilation

3.1.1 Prerequsites

- GNU make
- g++
- netinet/*
- sys/*
- arpa/*
- openssl/*
- libcrypto

3.1.2 Compilation of the application

The IMAP client can be compiled using the provided makefile by using `make` or `make all` from the project's root directory.

3.2 Running the Application

The application can be run using the following command:

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

Where:

- server - Required IP address/hostname of an IMAP server
- -p port - Optional port number

  DEFAULT VALUE:

  143 for unencrypted communication

  993 for encrypted communication
- -T - Turns on encrypted communication (imaps)

- -c certfile - Optional certificate file used for verifying SSL/TLS certificate from the server
- -C certaddr - Optional certificate directory, where certificates for verifying the SSL/TLS certificate from the server will be looked up
- -n - Only new messages will be fetched
- -h - Only headers will be fetched
- -a auth_file - Required path to a file containing username and password for authentication on the server
- -b MAILBOX - Optional mailbox name

  DEFAULT VALUE:

  INBOX
- -o out_dir - Required path to a directory to which messages will be fetched

3.2.1 Authentication File

The authentication file in the following format is used to store the username and password. which will be used by the application:

4 Testing

4.1 Compilation

4.1.1 Prerequsites

4.1.2 Compilation of automated tests

4.2 Running the Automated test

4.3 Tests output

Bibliography

[1] Authors, T. O. P.: SSL_connect. [online].
URL https://docs.openssl.org/1.0.2/man3/SSL_connect/

[2] Authors, T. O. P.: SSL_CTX_load_verify_locations. [online].
URL https://docs.openssl.org/3.0/man3/SSL_CTX_load_verify_locations/#synopsis

[3] Authors, T. O. P.: SSL_CTX_new. [online].
URL https://docs.openssl.org/master/man3/SSL_CTX_new

[4] Authors, T. O. P.: SSL_CTX_new. [online].
URL https://docs.openssl.org/master/man3/SSL_CTX_new/#notes

[5] Authors, T. O. P.: SSL_new. [online].
URL https://docs.openssl.org/master/man3/SSL_new/#description

[6] Authors, T. O. P.: SSL_set_fd. [online].
URL https://docs.openssl.org/master/man3/SSL_set_fd/