Faculty of Information Technology
Brno University of Technology

Network Applications and Network Administration
Project: POP3 Client with POP3S and TLS Support

2021

Jakub Bartko (xbartk07)

Contents

1 Introduction

The purpose of this documentation is to provide a basic introduction to the problem at hand, a summary of the implementation of its solution, and a usage guide.

2 Basic Terminology Overview

The executable part of this project is an **email client** using **POP3 protocol** for retrieving messages from remote email server.

Email client is generally a software used to access and manage user's email, implementing functions for message management, composition and reception. Usually, a remote Mail Transfer Agent server, hosted by an email service provider, is used for receipt and storage of user's email. The remote mail storage is referred to as the user's *mailbox* [4]. The emails are stored on the remote server to be either viewed or downloaded (presumably using an email client) by the user. Email client is activated either periodically or manually by user – to fetch emails from the mailbox. It is typically required to configure an email address, password, POP3/IMAP or SMTP address, port number and other preferences [5].

The two most provalent methods for email retrieval are **IMAP** and **POP** – both Internet standard protocols used by email clients to retrieve email messages. The main difference between the two is whether the user desires primarily to fetch new emails to store them locally, or to view their inbox repeatedly, from multiple devices and locations (*What are IMAP and POP?* [7]).

Internet Message Access Protocol (IMAP) allows the user to access the email messages from any device, as the contents aren't implicitly stored locally. They are rather read from the email service and downloaded only if requested by the user. Other advantages of using IMAP include simultaneous access to the same mailbox by multiple agents and ability to partially fetch portions of individual messages [1].

Post Office Protocol (POP) on the other hand primarilly supports download and delete operations. It's purpose is to connect to the server, retrieve messages and store them locally, which are subsequently deleted from the remote server. POP3 clients might also be instructed to leave the messages on the remote mail server, but, unlike the IMAP clients, it isn't their default behaviour nor does it reflect the original design (meant for users with only temporary Internet connection) [2].

POP version 3 is the version in most common use.

3 POP3 and Secure Communication

This section discusses the functionality requirements, implementation of which is described in section 4.

3.1 Basic Operation

This section is based on **RFC1939 [9]**. The communication between the remote server and email client roughly functions as follows:

- *the server host starts the POP3 service by listening on TCP port 110 ...*
- client host establishes a TCP connection with the server host,
- if connection is established successfully, the server sends a greeting,
- exchange of commands and server responses follows,
- *... until connection is closed or aborted.*

Each server response consists of: status indicator (`+OK` or `-ERR`) and the response itself – terminated by `CRLF`[1].

The lifetime of POP3 session is divided into following states: authorization state, transaction state, and update state.

After the server's greeting, the session enters the **authorization state**. The client is required to identify itself (e.g. using credentials). Only after successfull identification does the server acquire the necessary resources for the following operations – entering the **transaction state**.

This state is characterized by client's requests for actions on the POP3 server and server's responses. The main requests are for: message information, retrieval and deletion. It is terminated aither by aborting the connection, or by issuing a `QUIT` command, after which the session enters the **update state**.

Session enters this state *only* after a client-issued `QUIT` command. In this state, all messages marked as *deleted* are removed from the maildrop. The server then releases its lock on the maildrop and closes the TCP connection.

[1]In case of multi-line response, the terminating characters are "CRLF.CRLF"

3.2 TLS, SSL and OpenSSL

SSL stands for **Secure Socket Layer**. It is a standard behind secure communication on the Internet, integrating data cryptography into the protocol. Its operation is based on certificates and cryptographic algorithms [6]. SSL encrypts the data being transmitted, leaving only the user's computer and the secure server able to interpret the data.

Transport Layer Security (TLS) represents the successor of the now-deprecated SSL. TLS is a cryptographic protocol designed to provide communications security over a computer network [11]. Its use is preffered to SSL due to many known vulnerabilities of the latter.

The client might opt to communicate with or without encryption (TLS, SSL). It's common for both to be available – on different ports, e.g. **110** for *non-ecrypted* POP3 and **995** for *SSL-encrypted* POP3 service.

OpenSSL is a cryptography and SSL/TLS toolkit for general-purpose cryptography and secure communication [8]. It is one of the most widely known open library for secure communication. It is also capable of message digests, encryption and decryption of files, handling digital certificates and digital signatures. Some Linux distributions even come with a binary version of OpenSSL [6].

4 Code Structure and Implementation

The `popcl` program is implemented in C++17 and provided with Makefile which makes use of g++, GNU project C and C++ compiler[2], version 9 or above (see README for further details). Its operation consists roughly of: command-line options parsing, establishing connection with the POP3 server, message fetching, storing and deleting of messages, and status & error reporting. The following OpenSSL libraries were used for connection and communication handling: `BIO`, `SSL` and `Err`. These libraries provide OpenSSL with abstractions of numerous kinds of communication, including both secure and unsecure sockets.

The central structure in the implementation is the class `Connection`. It provides an abstraction over the OpenSSL structures and operations for the purposes of establishing a connection with a POP3 server and running basic email management, implementing both lower-level operations, such as single command-response `read`/`write`, and more complex ones – `get_msgs` and `delete_msgs`. Its also aims to uphold the RAII principle[3] and provide a structed system of error handling and reporting.

[2]`https://linux.die.net/man/1/g++`

4.1 Initialization and Connecting

For the initialization of the `Connection` object, command-line options need to be parsed first. This process is handled using the C standard library `getopt` functionality encapsulated in the `options.h` module with minimal overhead – iterating parsed options and setting corresponding values and switches in `Options` struct, which is passed to the constructor of the `Connection` object.

The constructor is also responsible for initializing OpenSSL libraries, user's credentials, server address and correct port. Then, according to the specified CL options, either secured or unsecured connection (which might be secured afterwards) is established.

In case a secured connection is to be established, certificates are loaded and verified either from specified file/directory, or from the default locations from which CA certificates are loaded[4].

Success of creating the `BIO` connection object, establishing the connection itself, and checking the server's response – greeting, is checked afterwards.

If specified, the unsecure connection is made secure by appending an existing *connect* `BIO` to a new *SSL* `BIO` [3], and by using the `STLS` command – starting a TLS negotiation *(POP3 STARTTLS extension [10])*.

4.2 Commands and Message Manipulation

The basic I/O for communicating with the server is implemented in methods `Connection::read` and `Connection::write`. Reading uses a standard C char buffer with optional checking for `"+OK"` response. Writing passes a given command with an appended `CRLF` suffix to the server. Verification is handled in `Connection::login` method by sending the `USER` and `PASS` commands with provided credentials and awaiting server's response. The bulk of operations on server is encapsulated in methods `Connection::get_msgs` and `Connection::delete_msgs`.

[3]RAII – `https://en.wikipedia.org/wiki/Resource_acquisition_is_initialization`

[4]`https://www.openssl.org/docs/manmaster/man3/SSL_CTX_set_default_verify_paths.html`

The retrieval of messages starts by issuing the `STAT` command and parsing the data on number and size of remotely stored messages. After that, the output directory for downloaded messages is prepared. This is followed by repeated issuing of `RETR` command, reading the possibly multi-line response, fetching the ID of given message for filename – which is afterwards sanitized, as it might contain characters invalid for a filename (such as backslashes). The actual storing of the file might still be omitted – in case the `-n`, new messages only, flag was specified, and the fetched message ID is already present in the destination folder. Otherwise, message is stored, and finally a report on the number of downloaded messages is printed.

Afterwards, if specified by the respective CL option, messages are deleted from the remote server, in like manner as message retrieval – but issuing the `DELE` command instead, finally followed by the `QUIT` command, entering the **update state** (see section 3.1).

5 Usage Guide

See `README` beforehand! Extract the contents of the provided `.tar` file to an empty directory. Navigate to the source directory and build the project using `make`. Execute by running `make && ./popcl --help` or `make run` which expects user's credentials to be provided in `auth` file and downloads the messages to `out/` in the source directory. Run `./popcl --help` for detailed specification of CL options.

Minimal program execution requires at least the following options to be specified: server address, file with user's credentials and directory for saving messages; e.g. `./popcl pop.mail.com -a auth_file -o out_dir`. Operation of the email client can be amended by specifing *delete* or *new* flags (`-d` and `-n`, respectively). Options regarding the security of the communication include enabling POP3S and STL connections, and specifying file/directory containing certificates. Custom port can be specified as well.

References

[1] Internet message access protocol – definition and advantages from wikipedia.

[2] Post office protocol – definition from wikipedia.

[3] bio_new_ssl – linux man page. [online], 2021. URL `https://linux.die.net/man/3/bio_new_ssl`.

[4] Email client – Definition from Wikipedia. [online], 2021. URL `https://en.wikipedia.org/wiki/Email_client`.

[5] What is Email Client? – Definition from Technopedia. [online], 2021. URL `https://www.techopedia.com/definition/1656/email-client`.

[6] Secure programming with the openssl api – tutorial by ibm. [online], 2021. URL `https://developer.ibm.com/tutorials/l-openssl/`.

[7] Are IMAP and POP? – Definition from Microsoft Support. [online], 2021. URL `https://support.microsoft.com/en-us/office/what-are-imap-and-pop-ca2c5799-49f9-4079-aefe-ddca85d5b1c9`.

[8] Openssl. [online], 2021. URL `https://www.openssl.org/`.

[9] Post office protocol version 3 – rfc1939. [online], 2021. URL `https://datatracker.ietf.org/doc/html/rfc1939`.

[10] Pop3 starttls extension – using tls with imap, pop3 and acap. [online], 2021. URL `https://datatracker.ietf.org/doc/html/rfc2595`.

[11] Tls – definition from wikipedia. [online], 2021. URL `https://en.wikipedia.org/wiki/Transport_Layer_Security`.