Klient IMAP s podporou TLS

Matej Olexa (xolexa03) 16.11.2024

Obsah

Popis

Implementácia IMAP klienta ktorý pomocou protokolu RFC3501 [5] umožňuje sťahovanie správ uložených na serveri. Po spustení sa klient autentifikuje, pomocou vybraných argumentov stiahne potrebné správy do vybraného adresára a preruší spojenie.

Spustenie

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

Povinné argumenty

- `<server>`

- `-a <auth_file>` - súbor s autentifikáčnymi údajmi
- `-o <outdir>` - priečinok na uloženie mailov

Nepovinné argumenty

- `-p <port>`
- `-T` - zapnutie šifrovania
- `-c <certfile>` - súbor s certifikátom
- `-C <certaddr>` - adresár s certifikátmi
- `-n` - iba maili typu `NEW`

- `-h` - iba hlavičky mailov
- `-b <MAILBOX>` - schránka

```
./imapcl imap.centrum.sk -T -p 993 -a auth_file -o saved_emails
```

Zoznam odovzdaných súborov

- `Makefile`
- `mockup.py`
- `README`
- `documentation.pdf`
- `src/`
  - `arg_parser.cpp`
  - `arg_pasrer.h`
  - `client.cpp`
  - `client.h`
  - `commands.cpp`
  - `commands.h`
  - `connection.cpp`
  - `connection.h`
  - `file_manager.cpp`
  - `file_manager.h`
  - `main.cpp`
  - `msg_parser.cpp`
  - `msg_parser.h`
  - `runner.cpp`
  - `runner.h`

Teória

IMAP

IMAP (Internet Message Access Protocol) [5] umožňuje klientom prístup k emailom umiestneným na mail serveru. Narozdiel od POP3, ktorý email po stiahnutí vymaže zo serveru, IMAP tieto maily po stiahnutí na serveri ponechá a synchronizuje svoje akcie s ostatnými zariadeniami.
Každá správa počas komunikácie s IMAP serverom sa začína s unikátnym reťazcom `tag`, tento tag sa mení (inkrementuje) každou správou poslanou z klienta - server svoje odpovede posiela s rovnakým tagom. Vďaka tomu vie klient zistiť, na ktorú spravu server odpovedá.
Dôležitou časťou je, že IMAP server môže kedykoľvek počas komunikácie poslať klientovi správu `BYE`, ktorá ukončuje spojenie.

Dôležité Operácie

- `LOGIN` - prihlásenie uživateľa na server
- `LIST` - vypísanie všetkých schránok

- `SELECT` - otváranie konkrétnej schránky
- `FETCH` - získanie konkrétného obsahu schránky
- `SEARCH` - vyhľadanie podľa konkrétnych údajov
- `LOGOUT` - odhlásenie a ukončenie spojenia

Workflow Príklad možnej postupnosti využitia IMAPu

1. `Pripojenie sa na server`
2. `Prihlásenie sa`
3. `Vybratie si schránky`
4. `Stiahnutie chcených emailov`
5. `Odhlásenie sa`

Odpovede od servera Odpovede poslané od servera majú na začiatku kód statusu

- `OK` - príkaz uspel
- `NO` - príkaz neuspel
- `BAD` - nesprávna syntax

Flags Pre vyberanie emailov môžeme použiť aj tzv. `flags` (príznaky), tieto nám umožňuju vylučiť nechcené emaily. Každý mail server môže mať svoje vlastné príznaky, avšak existujú štandartné ktoré sa vyskytujú na všetkých.

Štandartné príznaky

- `\Seen`
- `\Flagged`
- `\Answered`
- `\Draft`

Štruktúra emailu

Každý email je rozdelený na 2 hlavné časti – Hlavička (Header) a telo (body). Okrem toho, sa k ním ukladajú metadata pre bližšie informácie.

Hlavička V hlavičke sa nachádzajú informácie o smerovaní emailu. Vyskytuje sa vo vrchnej časti emailu.
Zvykne obsahovat -

- `From:` - Odosielateľ emailu
- `To:` - Príjmateľ emailu
- `Subject:` - Predmet emailu
- `Date:` - Dátum emailu
- `Message-ID:` - Unikátny identifikátor emailu
- `Reply-To:` - Špecifikuje adresu na ktorej správu email odpovedal

Telo Telo mailu obsahuje hlavný obsah, väčšinou je štrukturované ako plaintext alebo HTML.

Ostatné Ďalšie informácie v emaili sú napr. - `prílohy` - dokumenty, obrázky, videá, Posielajú sa ako osobitné časti v tele mailu pomocou Multipurpose Internet Mail Extensions (MIME) formátu - `UID` - unikátny identifikátor pridelený danou schránkou - `príznaky`

SSL

Slúží na šifrovanie dát pomocou symetrickej a asymetrickej enkrypcie. Pri prvotnom pripojení, klient kontroluje validitu certifikátu servera. [1] [2] [3]

Implementácia

Program impactl je implementovaný v programovacom jazyku C++ 20. Je rozdelený do 4 hlavných častí, ktoré sú používané v `main.cpp`. Najprv, sa pomocou `argument_parser` uložia argumenty z volania programu do pomocného objektu triedy `Connection`. Tento objekt sa používa na držanie potrebných informácii. Následne sa vytvorí inštancia triedy `File_manager`, ktorý slúži na prístup k súborom. Nad týmto objektom sa zavolá metóda `get_auth_data` s referenciou na inštanciu triedy `connection` a uloží doň dáta.
Nakoniec sa vytvorí objekt triedy `runner`, ktorý využíva dáta získané z `connection` a pomocou metódy `.run()` spustí hlavnú funkcionalitu programu.

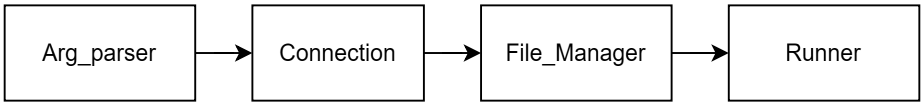

Figure 1: Main flow

`Runner` je trieda ktorá slúži pre hlavný beh programu. V jej konštruktore sa vytvárajú inštancie tried `Client`, `Commands` a `MsgParser`. Metóda `run` tieto objekty využíva a volá ich metódy pre implementovanie funkcionality IMAP klientu.
Na začiatku sa vytvóri spojenie pomocou `initialize_connection`, ktorá najprv pošle prázdnu správu serveru, potom pošle údaje na prihlásenie, získa možnosti serveru, vyberie schránku a získa počet správ.
Následne sa podľa zadaných argumentov rozhodne, či bude získavať iba nové správy pomocou `fetch_new_messages` alebo všetky, Podľa toho si uloží dáta do vektoru `messages_to_process`. Nakoniec sa zavolá metóda `process_messages`, ktorá na ID každej uloženej správy zavolá `process_single_message` a tá skontroluje existenciu jednotlivých správ - v prípade že neexistujú, tak sú stiahnuté a uložené do adresára.
Ak počas sťahovania správ nedošla klientovi odpoveď typu `BAD` alebo `NO`, považuje sa beh za úspešný a nakoniec sa odošle iba príkaz `logout`.

`Runner` na komunikáciu so serverom používa metódu `send_and_receive` - táto metóda slúži na jednotné poslanie a prijatie správy - pretože každá poslaná správa čaká na odpoveď. Taktiež je to spôsob ktorým sa spravujú tagy. Pre získanie príkazu v správnom formáte, ktorý príjme IMAP server sa používa trieda `Commands`, ktorého metódy tieto príkazy navrátia.

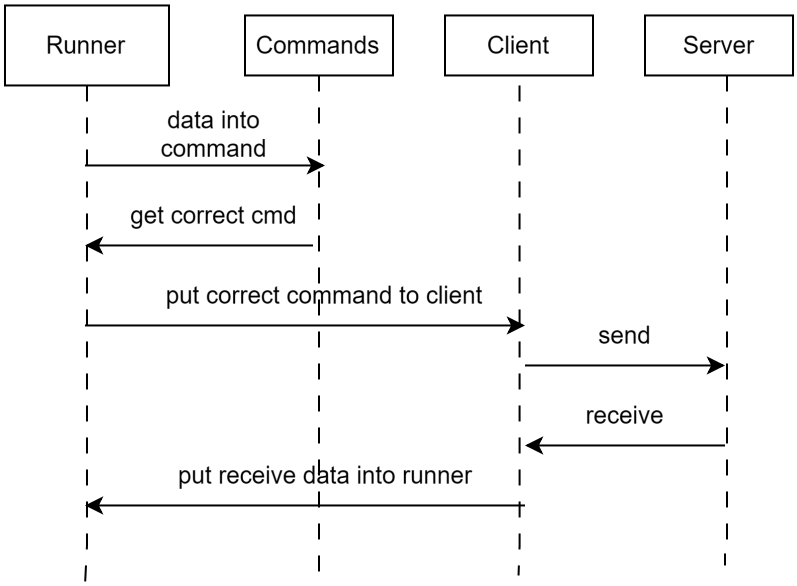

Figure 2: Flowchart

Komunikácia so serverom

Komunikácia so serverom je implementovaná v súboroch `client.cpp` a `client.h`. Na TCP komunikáciu sa používajú knižnice `netdb`, `sys/socket`, `sys/types` [4]. Metódy tejto triedy sú - `connect` - na vytvorenie socketu a vytvorenie pripojenie so serverom - `send` - posielanie správ - `receive` - príjmanie správ, táto metóda vráti `std::pair` reťazca a bool - v prípade, že server pošle správu `BYE`, teda ukončí pripojenie tak sa bool vráti v hodnote `true`, čo naznačí programu aby ukončil spojenie a neposielal ďalšie správy. Táto funkcia cyklicky čaká na koniec správy (pod reťazcom OK/NO/BAD), ale je implementovaný timeout ktorý ukončí čakanie predčasne aby nedošlo k úplnému zaseknutiu. Dôležitý je `TIMEOUT` - nastavený na 15 sekund. Ak od serveru nepríde do 15 sekúnd odpoveď obsahujúca `OK/BAD/NO` (so správnym tagom), tak sa program ukončí.[5][6][7]

Šifrovanie SSL

Enkrypcia komunikácie je implementovaná v rovnakých súboroch ako aj komunikácia so serverom - `client.cpp` a `client.h`. Používajú sa na to knižnice `openssl/ssl`, `openssl/err` [1][2][3]. Metódy implementujúce šifrovanie sú -

`init_openssl` - inicializuje šifrovanie, pridá šifrovacie algoritmy, načíta správy pre errory, vytvorí SSL kontext pre TLS pripojenie a načíta verifikáciu certifikátov - `verify_certificate` - získa certifikát od serveru, ktorý následne verifikuje a v prípade neúspechu vypíše chybovú hlášku

Okrem iného sa v metódach triedy `Client` kontroluje príznak ktorý rozhoduje o šifrovanej komunikácii a volá potrebné metódy knižníc a pomocných metód. V prípade šifrovanej komunikácie sa pre posielanie a príjmanie správ používajú rovnaké metódy z triedy `Client`, avšak namiesto funkcií `::send` a `::receive` sa používa `SSL_write` a `SSL_read`.

Posielanie príkazov

Príkazy posielané na server sú implementované v triede `Commands`, ktorá má metódy na každý potrebný príkaz.

- `login`
- `logout`
- `list`
- `select`
- `fetch_header_important`
- `fetch_header`
- `fetch`
- `get_new_messages`

Tieto metódy vezmú parametre `tag` a informácie špecifické pre daný príkaz a vrátia reťazec príkazu v RFC3501 formáte. Dôležitou metódou je `fetch_header_important`, ktorý pomocou `PEEK` získa zo správy informácie ktoré sa používajú na pomenovanie správ - predmet, dátum, odosielateľa a message id. Tieto správy sa používajú na pomenovanie uložených správ. Táto kombinácia informácii nám zaručí unikátnosť názvov a zároveň nám zaručí prehľadnosť v priečinku.

Spracovavanie správ

Reťazce prijaté po odoslaní príkazu na server sú spracované metódami triedy `MsgParser`. Tieto metódy slúžia hlavne na extrakciu dôležitých dát ako napr. ID nových správ.

- `get_message_count` - získa počet správ zo schránky
- `get_new_messages` - vráti vektor prvkov typu integer s hodnotami ID všetkých nových správ
- `get_capability` - do vektora s prvkami typu string vloží všetky `capabilities` serveru
- `get_mailbox_names` - do vektora s prvkami typu string vloží všetky názvy schránok mailu
- `extract_header_field` - z hlavičky emailu získa dáta konkrétneho poľa, napr. subject

- `get_file_name` - z hlavičky mailu vezme odosielateľa, dátum, predmet a message id ktoré vráti ako reťazec - tento reťazec sa používa ako názov súboru do ktorého sa mail následne ukláda

Ukladanie mailov

Ukladanie prijatých správ je implementované v triede `File_manager`. Táto trieda manažuje súbory a pomocou metódy `get_auth_data` vie získať autentifikačne dáta, ale používa sa aj na ukládanie správ. Na ukládanie sa používajú metódy

- `save_mail`
- `check_existence`
- `remove_file`

Cez získané dôležité dáta z hlavičky emailu si vieme skontrolovať existenciu správy v adresári. V tejto implementácii sa súbory obsahujúce čisto hlavičku mailu označujú prefixom `H-`. Ak chceme uložiť celú správu, najprv skontrolujeme jej existenciu v adresári - v prípade jej existencie sa nebude ukladať 2x. Taktiež sa kontroluje existencia hlavičkového súboru v adresári, ak existuje tak sa vymaže a vytvorí sa súbor s celou správou, už bez prefixu `H-`. Meno emailu je ale obmedzene počtom znakov na 250, kvôli limitáciam dĺžky názvy súboru v OS linux.
Stiahnuté súbory su teda pomenované pomocou kombinácie `odosielateľ-dátum-predmet-MessageID`

Prečo nepouživať UID?

V tejto implementácii nie je použitý UID na pomenovanie súborov; namiesto neho je využitý Message-ID, aj keď je dlhší a menej prehľadný. Najväčšou nevýhodou UID je možnosť skončenia UID Validity, čo by viedlo k nutnosti opätovného stiahnutia všetkých správ a potenciálnej desynchronizácii klienta so serverom. Message-ID je vždy jedinečný pre každý email odoslaný daným serverom, takže v kombinácii s emailom odosielateľa predstavuje spoľahlivejší spôsob unikátneho pomenovania.

Nové správy

Program pri získavaní nových správ používa parameter `NEW`. Ďalšie varianty mohli byť `UNSEEN` alebo `RECENT`, ale pri testovaní sa ako najspoľahlivejší ukázal `NEW` - stiahol najväčší počet potenciálne chcených správ.

Testovanie

Klient bol predovšetkým testovaný na serveri `imap.centrum.sk` cez poskytovateľa `pobox.sk` [9]. Tu bol vytvorený email na ktorý boli posielané správy. `imap.centrum.sk` poskytuje možnosť pripojenia bez TLS - takže bol ideálny pre túto implementáciu. Komunikácia bola sledovaná cez `wireshark` [8] - toto umožnilo správne otestovanie TLS. Na otestovanie konečnej implementácie bol na základe zadania použitý server `eva.fit.vutbr.cz`, na ktorom sa počas štúdia

nahromadilo ~1500 správ (pri prihlásení na xolexa03 účet). Schnopnosť tohto programu správne stiahnuť všetky správy (otestované aj na serveri `merlin` bez nájdených chýb pomocou programu `valgrind`) ukázala jeho výslednú funkcionalitu a spoľahlivosť.

Stiahnuté súbory vo formáte .eml boli otestované otvorením v programe microsoft outlook - čím sa otestovala správnosť zapisovania.

Okrem iného boli použité aj mockup testy, viz. `mockup.py`. Tento program, spustený pomocou `python mockup.py` spojazdní mockup server na porte 3143 a je schopný 6 scénarií.

- `Happy path` - správne spravanie serveru
- `Corrupt message` - pošle nesprávnu správu, klient správne vyhodí chybovú hlášku
- `Server disconnect` - server ukončí spojenie hneď na začiatku
- `Slow response` - server má oneskorenie 2 sekundy
- `Auth failure` - server nepríjme prihlásenie (pomocou `NO`)
- `Malformed response` - server odošle správu bez správneho formátovania

Všetky varianty je možné otestovať pomocou `python mockup.py ./imapcl localhost -p 3143 -o <mail directory> -a <authentification file>`

Program od uživateľa bude vyžadovať číslo scenária. Týmto spôsobom boli vyskúšané a odhalené možné chyby.

Zdroje

1. **OpenSSL**
   OpenSSL - Cryptography Basics
2. **OpenSSL Manual**
   OpenSSL Manual
3. **OpenSSL IBM**
   OpenSSL Tutorial by IBM
4. **TCP Socket**
   Socket Programming in C++
5. **IMAP**
   IMAP 101 - Manual IMAP Sessions
6. **IMAP Crib**
   IMAP Crib
7. **RFC3501**
   RFC3501 - IMAP Protocol
8. **Wireshark**
   Wireshark

9. **Pobox**
   Pobox