VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

Síťové aplikace a správa sítí

Klient POP3 s podporou TLS

14. listopadu 2021 Daniel Němec (xnemec92)

Obsah

1 Úvod

Cílem projektu bylo vytvořit síťovou aplikaci, která podle zadaných parametrů, stáhne nebo smaže zprávy v emailové schránce přes internetový protokol POP3.

2 POP3

Internetový protokol POP3 se využívá k přístupu k emailové schránce, která je na vzdáleném serveru. Zkratka POP znamená *Post Office Protocol* a číslo 3 značí verzi protokolu. Protokol stáhne ze serveru všechny zprávy(i spam) a uloží je do lokální složky v počítači, takže ke zprávám se dá přistupovat i offline. Nevýhodou je že stáhne i zprávy, které uživatele nezajímají. Defaultní port je 110. Jeho zabezpečená verze POP3S pomocí protokolu TLS, umožňuje vše provádět zabezpečeně na defualtním portu 995.

3 Spuštění programu

K správné kompilaci programu je potřebný překladač `gcc`. Také je potřebný program `ake . Testování pro` 4.2.1|. Ve složce s projektem se nachází Makefile, pomocí kterého můžeme program zkompilovat použitím příkazu:

```
$ make
```

Samotný zkompilovaný program se pouští s následujícími argumenty:

```
$ ./popcl <server> [-p <port>][-T|-S[-c <certfile>][-C <certaddr>]]
[-d][-n] -a <auth_file> -o <out_dir>
```

Pořadí argumentů, až na argument **<*server*>**, je libovolné.

- `<server>` - Hostname nebo IP adresa serveru, na který se chceme připojit. Tento argument je povinný.
- `-p <port>` - Nepovinný argument, specifikuje port na kterém se bude připojovat k serveru. V případě že tento parametr není zadán, zvolí se výchozí porty pro protokol POP3.
- `-T` - Nepovinný argument, zapíná šifrované připojení pop3s.
- `-S` - Nepovinný argument, zapíná připojení STLS.
- `-c <certfile>` - Nepovinný argument, zadává certifikační soubor.
- `-C <certaddr>` - Nepovinný argument, zadává složku s certifikačními soubory.
- `-d` - Nepovinný argument, vymaže všechny zprávy v emailové schránce.
- `-n` - Nepovinný argument, stáhne pouze nové zprávy, která již nejsou stažené.
- `-a <auth_file>` - Povinný argument, specifikující soubor s přihlašovacími údaji ke schránce.
- `-o <out_dir>` - Povinný argument, specifikující složku, kam se mají stažené zprávy uložit, pokud složka neexistuje, klient jí vytvoří.

Pokud je zadán neplatný argument, je vypsána nápověda použití programu.

4 Implementace

Program je implementovaný v jazyce C++, především kvůli podpoře řetě. K implementaci projektu jsem použil knihovnu `penssl, která ps`kytuje API pro zabezpečenou komunikaci. Při programování připojení a komunikace se serverem, jsem se inspiroval OpenSSL dokumentací.

4.1 Parsování argumentů

Argumenty se parsují ihned po spuštění programu v hlavní funkci `main()`. K parsování dochází pomocí knihovny `getopt.h` konkrétně funkcí `etopt() . Ar`umenty jsou ukládány do struktury `clargs`, která je definována v souboru `popcl.hpp`. Poté se kontroluje, zda jsou zadány povinné argumenty, pokud nejsou tak je vypsána nápověda.

4.2 Připojení k serveru

Pokud byl zadán parametr -T, program naváže zabezpečené připojení k serveru na daném portu ve funkci `connect_to_server_secured()`. Pokud je zadán parametr -S, program naváže nezabezpečené připojení k serveru funkcí `connect_to_server()` a poté se pokusí toto připojení "vylepšit"na zabezpečené pomocí příkazu STLS, k tomuto dochází ve funkci `init_stls()`. Zabezpečené připojení vyžaduje potvrzovací certifikáty, které načítá funkce `load_ca()`, pokud nejsou zadané parametrem, načtou se výchozí uložené certifikáty pomocí funkce `SSL_CTX_set_default_verify_paths()`. Pokud není zadán ani jeden z těchto argumentů, naváže nezabezpečené připojení a komunikaci. Následně se zašlou přihlašovací údaje. Pokud se naskytne někde chyba, je vypsána na standardní chybový výstup. Jsou-li zadány nesprávné přihlašovací údaje, je uživatel upozorněn.

4.3 Komunikace

Po každém zaslaném příkazu severu se volá funkce `response_ok()`. Tato funkce načítá odpověď v cyklu dokud není načtena celá zpráva. Pokud je zpráva jednořádková je zakončena `\r\n`. Více řádková zpráva je zakončena `\r\n.\r\n`. Poté zkontroluje zda odpověď není chybná.

4.4 Ukládání zpráv

K operaci se zprávami dochází ve funkci `process_message()` Program nejdříve zašle serveru příkaz `STAT`, pomocí kterého klient zjistí počet zpráv v emailové schránce. Poté se začnou stahovat zprávy přes příkaz `RETR n`(`n` je číslo zprávy), která je následně uložena do složky, pod příponou .eml a pojmenovaná identifikátorem "Message-ID", který se pomocí regexu najde ve zprávě.
Pokud zpráva neobsahuje "Message-ID", je pojmenována `email_i`, kde `i` je číslo zprávy dané serverem. Stahování probíhá v cyklu dokud nejsou staženy všechny zprávy.

4.5 Nové zprávy

Probíhá při zadání parametru -n. Klient prochází všechny zprávy jako při klasickém stahování, ale kontroluje i zda je zpráva již uložena ve výstupní složce. Tato kontrola probíhá pomocí porovnávání názvů již uložených zpráv a identifikátorů zpráv z emailové schránky, pokud složka zprávu neobsahuje, zpráva je stažena a navýší s počítadlo s novými zprávami. Zde je výhodou mít zprávy uložené s

identifikátorem jako názvem zprávy, protože identifikátor je vždy unikátní. Nevýhodou je někdy příliš dlouhé pojmenování zpráv a je nutné nahrazovat nepovolené znaky.

4.6 Mazání zpráv

Probíhá při zadání parametru -d. Zprávy jsou nejdříve staženy ze serveru do složky a poté pomocí příkazu `DELE n` smazány ze serveru. Pokud jsou zadány parametry -d a -n současně, jsou nové zprávy staženy a následně smazány ze serveru.

5 Zdroje

- Zadání projektu ve WIS
- RFC 1939: Post Office Protocol - Version 3
- Dokumentace OpenSSL