VYSOKÉ UČENIE TECHNICKÉ V BRNE

Fakulta informačných technológií

Sieťové aplikácie a správa sietí

2017/2018

POP3 klient

Daniel Florek (xflore02) 8.11.2017

1 Úvod

Úlohou projektu, ktorý popisuje táto dokumentácia bolo vytvoriť POP3 klient aplikáciu, ktorá dokáže stiahnúť všetky emaily, prípadne iba nové emaily zo serveru šifrovanou aj nešifrovanou formou vo formáte IMF. Užívateľovi musí byť takisto poskytnutá možnosť overovať server vlastnými certifikátmi. Aplikácia musí byť takisto schopná správy zo serveru vymazať. Referenčný stroj na testovanie a preklad aplikácie je server merlin.fit.vutbr.cz

2 Úvod do problematiky

Skôr, ako popíšem svoju variantu riešenia, vysvetlím niekoľko termínov, ktoré je nutné poznať pre pochopenie danej problematiky.

2.1 POP – POST OFFICE PROTOCOL

POP je internetový protokol, ktorý sa využíva na prijímanie emailov prostredníctvom TCP/IP protokolu. Všeobecná myšlienka POP je pripojiť sa na server, stiahnúť emaily k používateľovi, a potom ich následne zo serveru vymazať, ale väčšina mail klientov má možnosť nechať správy na serveri. POP má nezabezpečenú aj zabezpečenú variantu komunikácie pomocou SSL/TLS. Pre nezabezpečenú komunikáciu server načúva na porte 110, pre zabezpečenú na porte 995. Niektoré servery podporujú nadviazať zabezpečenú komunikáciu na porte 110 až po nadviazaní nezabezpečeného spojenia pomocou prijatia príkazu „STLS“. Posledná verzia protokolu je POP3.

2.2 POP3

POP3 je najaktuálnejšia verzia POP protokolu. POP3 server podľa RFC normy musí mať implementované určité príkazy a niekoľko príkazov je voliteľných a nie každý server ich musí podporovať.

Pre riešenie tohto projektu nás budú zaujímať iba povinné príkazy ktoré sú:

- **STAT** – Vypíše počet emailov na serveri a ich celkovú veľkosť v bajtoch
- **LIST** – Server odošle zoznam všetkých emailov a ich veľkostí
- **DELE** – Server vymaže danú správu po tom čo sa ukončí spojenie
- **NOOP** – Server odošle kladnú správu aby sa zistilo, či je v poriadku
- **RSET** – Odznačí správy, ktoré boli označené na zmazanie
- **QUIT** – Ukončí spojenie.

Forma odpovedí serveru začína vždy „+OK“ v prípade úspechu alebo „-ERR“ v prípade neúspechu a ďalej nasleduje odpoveď na príkaz, prípadne chybová hláška v prípade neúspechu. Odpovede serveru sú ukončené pomocou CRLF (\r\n). Viacriadkové odpovede serveru sú ukončené pomocou „.CRLF“ (.\r\n).

2.3 TLS/SSL

TLS (Transport Layer Security) a jeho predchodca SSL (Secure Sockets Layer) sú protokoly, ktoré slúžia na šifrovanie dát. Zabezpečujú bezpečnú komunikáciu cez internet. Zabezpečujú hlavne prehliadanie webu, prácu s emailami, výmeny správ a iné prenosy. Medzi TLS a SSL sú drobné rozdiely, ale protokol je prakticky ten istý. Protokoly zabezpečujú komunikáciu voči odpočúvaniu. Zväčša býva autorizovaný iba server a klient ostáva neautorizovaný a klient si overuje identitu servera. Zabezpečené spojenie nastáva takzvaným „handshake“, kedy sa server autorizuje užívateľovi svojím certifikátom. Takisto mu odošle svoj verejný kľúč ktorým klient zašifruje nejaký spoločný kľúč ktorým budú komunikáciu šifrovať a dešifrovať. Zašifrovaný kľúč následne klient odošle serveru a ten si ho dešifruje súkromným kľúčom (asymetrické šifrovanie). Ďalej sa už komunikácia šifruje kľúčom, ktorý vygeneroval klient symetricky.

3 Vlastné riešenie

3.1 Spustenie programu

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

Povinné argumenty programu sú:

- **Server** – na ktorý sa pripájame. Vo forme mena, IPv4 aj IPv6 adresy.
- **-a Autentizačný súbor** – Súbor obsahujúci prihlasovacie údaje na daný server.
- **-o Output priečinok** – Priečinok, kde sa uložia všetky prijaté emaily.

Voliteľné argumenty sú:

- **-T** – Zapína šifrovanú komunikáciu od začiatku, v prípade, že nie je nastavený prepínač pre nastavenie portu, tak sa implicitne nastaví port 995 na ktorom pop3s za normálnych prípadov funguje.
- **-S** – Zapína nešifrovanú komunikáciu, v prípade nenastaveného portu na porte 110 a po prijatí welcome správy od serveru pomocou „STLS“ príkazu zapína šifrovanú komunikáciu.
- **-c certfile** – v prípade prepínačov –T alebo –S môžeme špecifikovať súbor, ktorým budeme overovať certifikát, ktorý nám odošle server.
- **-C certaddr** – v prípade prepínačov –T alebo –S môžeme špecifikovať adresár s certifikátmi, ktorými budeme overovať certifikát, ktorý nám odošle server.
- **-p port** – týmto prepínačom môžeme špecifikovať vlastný port na ktorý sa chceme pripojiť.
- **-d** – prepínač, ktorý definuje, že sa majú vymazať správy. V mojej variante riešenia mažem správy ktoré reálne prijímam a zapisujem do output directory, takže v prípade kombinácie s prepínačom –n mažem iba nové správy.

- **-n** – prepínač, ktorý definuje, že sa majú prijať len nové správy. V mojej variante riešenia prijímam tie správy, ktoré náš klient pre daného používateľa ešte neprijal.

V prípade použití prepínačov pre šifrovanú komunikáciu –T a –S a nešpecifikovaní žiadnych certifikátov sa musí použiť predvolený systémový adresár s certifikátmi. V mojej variante riešenia takéto správanie nastáva aj v prípade, že používateľ špecifikuje do certfile alebo certaddr prípadne do oboch naraz prázdny reťazec.

3.2 Prijímanie Argumentov

Prijímanie argumentov je implementované v module „arguments.h“. Stará sa o ne funkcia *getArguments(int argc, char** argv)*, ktorá berie ako parametre počet argumentov programu a samotné argumenty programu. Spracovanie argumentov pozostáva z klasického cyklu obsahujúceho getopt a ukladanie argumentov do globálnych premenných, ktoré sú deklarované v module „globvars.h“. V prípade, že nie je zadaný port, tak je implicitne nastavený na 110. V prípade, že je zadaný –T prepínač, ktorý povoľuje POP3s komunikáciu, tak sa port implicitne nastaví na 995.

3.3 Pripájanie na server

Po tom, čo sa úspešne prijmú argumenty, tak sa zavolá funkcia *int bindSocket(std::string server_hostname, int port_numer)* v nej sa vykoná preklad adries a následne sa pokúsi vytvoriť socket a pripojiť sa s ním na adresu s daným portom. Keď sa program úspešne napojí k serveru, tak sa vykoná test, či sme zapli program s prepínačmi –S alebo –T, v tomto prípade sa vykoná prepojenie otvoreného socketu s SSL kontextom vo funkcií *bindSSLSocket(int client_socket, std::string certfile, std::string certaddr)* v ktorej sa nastaví kontext šifrovanej komunikácie, nastaví sa do nej certifikát alebo zložka s certifikátmi, prípadne oboje. Potom sa vykoná SSL handshake a nakoniec sa overí certifikát serveru pomocou zadaného certifikátu prípadne priečinku certifikátov. Ako klientská metóda šifrovania sa použije TLSv1.2. V prípade prepínača –S sa ešte pred prepojením SSL kontextu so socketom prijme program welcome hlášku od servera a následne sa odošle príkaz „STLS“, ktorý zapína šifrovanú komunikáciu.

3.4 Autentizácia užívateľa

Keď sme sa úspešne pripojili na server, tak sa zavolá funkcia *int authorize(int client_socket, std::string authfile)* prebehne overenie prihlasovacích údajov, ktoré sa načítajú zo súboru zadaného ako argument funkciami *getUsername(std::string authfile)* a *getPassword(std::string authfile)*. Následne sa tieto údaje odošlú pomocou POP3 príkazov USER username a PASS password serveru funkciou *bool sendCommand(std::string command)* na prijímanie správ zo serveru sa používa funkcia *std::string recieveResponse()*. Ktorá prijíma znaky od serveru až po CRLF (\r\n). Následne sa správy od serveru kontrolujú

funkciou *bool isERR(std::string response)* ktorá kontroluje, či náhodou server neodpovedal chybovou hláškou.

3.5 Prijímanie emailov

Po tom, čo sa užívateľ serveru predstaví, tak sa zavolá funkcia *int getMails(outdir)* v ktorej sa hneď na začiatku zavolá funkcia *getIDS(),* ktorá zo súboru pre daného používateľa vytiahne uchované ID správ, ktoré sa už prijali. Následne sa na server odošle príkaz STAT aby sa získala správa o počte správ na serveri. Následne funkcia *int getAmount(std::string response)* ktorá vyparsuje počet správ na serveri. A zavolá sa funkcia *int recieveAllMessages(int mails, std::cout outdir)*. Ktorá sa precyklí cez všetky správy na serveri. Prijme celú správu, vytiahne z nej Message-ID, v prípade, že bol zadaný –n prepínač, tak sa porovná, či sa také Message-ID ešte neprijalo, v prípade, že už také v zozname je, tak sa správa nezapisuje a prijme sa ďalšia správa na serveri. Message-ID sa zároveň používa ako názov pre emaily. V prípade, že je nastavený aj –d argument, tak každá správa, ktorá sa zapíše do output priečinka sa vymaže. V prípade, že sa nezadal –n argument, tak sa vymažú všetky správy, v prípade, že sa zadal, tak sa vymažú len nové správy.

3.6 Problémy pri implementácií a popis zvolených riešení

- **-n** prepínač prijíma nové správy na základe Message-ID. Ale pri testovaní sa prišlo na to, že niektoré správy Message-ID nemajú, takže tieto správy sa ukladajú ako non-ID + poradové číslo. Tieto správy ale keďže nemajú Message-ID, tak ich nevieme ako evidovať, že sme ich už prijali, takže sa prijmú vždy, aj keď prijímame iba nové správy. To isté platí aj pri mazaní. Tieto správy sa zmažú vždy, aj keď by sme mali zmazať iba nové správy.
- Keď sa sťahujú emaily bez **–n** prepínača, tak sa stiahnuté emaily nezaznamenávajú ako stiahnuté, takže pri použití **–n** prepínača sa emaily stiahnú znovu, aj keď už boli stiahnuté bez **–n** prepínača.
- **-d** prepínač maže správy, ktoré sa zapisujú na disk, takže v prípade bez prepínača **–n** sa vymažú všetky správy. V prípade že bol zadaný prepínač **–n** tak sa zmažú iba nové správy.
- Pri testovaní prijímania správ som narazil na problém, že pri niektorých správach minimálne na serveroch seznam.cz, centrum.cz, centrum.sk sa mi stávalo to, že server pri odosielaní správy deklaroval, že odošle určitý počet bajtov, ale po prijatí daného počtu bajtov socket nebol prázdny a ostalo tam pár bajtov navyše, takže som bol odkázaný ešte precykliť a konkatenovať k správe tie zvyšné bajty, až kým sa socket nevyprázdnil.
- Output priečinok musí existovať, v opačnom prípade sa nezapíše nič.
- V prípade použití prepínačov pre šifrovanú komunikáciu –T a –S a nešpecifikovaní žiadnych certifikátov sa musí použiť predvolený systémový adresár s certifikátmi. V mojej variante riešenia takéto správanie nastáva aj v prípade, že používateľ špecifikuje do certfile alebo certaddr prípadne do oboch naraz prázdny reťazec.

3.7 Návratové kódy

- **0** – v prípade, že sa program ukončil správne.
- **1** – v prípade, že program zlyhal vo fáze argumentov.
- **-1** – v prípade, že program zlyhal pri vytváraní socketu alebo pripájaní na server
- **2** – v prípade, že program zlyhal pri vytváraní SSL spojenia alebo pri zlyhaní overovania certifikátu
- **3** – v prípade, že program zlyhal pri welcome message serveru
- **4** – v prípade, že program zlyhal vo fázi, keď by mal začať TLS spojenie (prípadne ho nepodporuje)
- **5** – v prípade, že program zlyhal pri autentizácií používateľa
- **6** – v prípade, že program zlyhal pri príkaze STAT (získavanie počtu správ na serveri) prípade je formát správy v zlom formáte.
- **-3** – v prípade, že program zlyhal pri prijímaní niektorej zo správ.

Referencie

[1] Post Office Protocol. Wikipedia: the free encyclopedia [online].
https://sk.wikipedia.org/wiki/Post_Office_Protocol

[2] RFC1939 POP3. IETF.org [online]

https://tools.ietf.org/html/rfc1939

[3] Transport Layer Security. Wikipedia: the free encyclopedia [online].

https://sk.wikipedia.org/wiki/Transport_Layer_Security