Vysoké učenie technické v Brne

Fakulta informačných technológií

Dokumentácia k projektu pre predmet ISA

FTP klient

20.11.2016

Autor: Samuel Macko, xmacko10@stud.fit.vutbr.cz

Obsah

1 Úvod

Zadaním projektu je vytvoriť zjednodušený LDAP server. Server pracuje s databázou obsahujúcou meno, login a email. Klient môže získavať záznami vyhovujúce zadanému filtru.

2 LDAP protokol

2.1 LDAP všeobecne:

LDAP (Lightweight Directory Access Protocol) je internetový protokol pre prístup k adresárovým službám. Pracuje nad vrstvou TCP/IP a štandardne beží na porte 389. Systém LDAP je definovaný pomocou množiny 4 modelov:

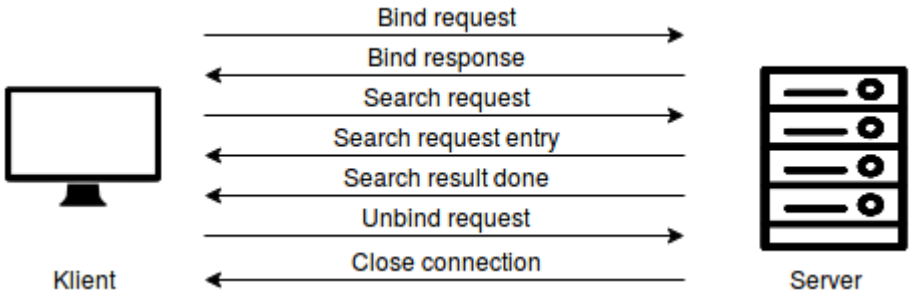
1. Informačný model: popisuje uloženie informácií v adresári.
2. Menný model: definuje ako sú dáta hierarchicky usporiadané.
3. Funkčný model: popisuje prístup k dátam.
4. Bezpečtnostný model: má na starosti zabezpečenie dát v adresári.

Základné operácie:

- Bind: prihlásenie do adresára
- Unbind: ukončenie spojenia
- Search: vyhľadávanie v adresári. Môžeme zadať ďalšie parametre ako napr. bázu, rozsah vyhľadávania, masku a pod.
- Compare: porovnávanie hodnôt atribútov
- Abandon: zruší predchádzajúcu operáciu
- Modify: úprava záznamu
- Add: pridanie záznamu
- Delete: zrušenie záznamu

Správy zakódované v ASN.1.

Príklad bežnej komunikácie medzi klientom a serverom:



2.2 Obmedzenia vyplívajúce zo zadania:

Tento zjednodušený LDAP server podporuje iba správy: Bind, Unbind a Search requesty. Prihlasovanie do adresára je anonymné (nevyžaduje prihlasovacie údaje). V Search správe server rozozná okrem filtra iba maximálny počet vrátených záznamov, ostatné údaje sú ignorované. Klient môže zahájiť komunikáciu správou Search request bez predchádzajúcej správy Bind request. Server nedokáže spracovať bloky správy ktorých dĺžka je reprezentovaná troma a viac bajtmi no vzhľadom na databázu a dĺžku bežne používaných filtrov by to nemalo predstavovať príliš veľký problém.

3 Implementácia

3.1 trieda Network:

Trieda nemá žiadne triedne premenné a všetky metódy sú statické, slúžia na vytvorenie a prácu so soketom. Obsahuje globálnu premennú BUFFER_SIZE nastavenú na hodnotu 512.

3.1.1 create_socket(int port):

Metóda dostane ako parameter port. Naplní štruktúru sockaddr_in, vytvorí soket a pripojí ho k zadanému portu.

3.1.2 read_message(int socket):

Zo soketu zadaného parametrom sa v cykle načítavajú znaky do poľa buffer ktorý má veľkosť globálnej konštanty BUFFER_SIZE.

3.1.3 send_message(int socket vector<uint8_t > message):

Metóda zapíše správu message na zadaný soket.

3.2 trieda Params

Slúži na spracovanie a ukladanie parametrov programu.

Obsahuje 2 privátne premenné:

1. int port – port na ktorom server počúva
2. string database_file_name – názov databázového súboru

3.2.1 handle_parameters(int argc, char** argv):

Načíta parametre s ktorými bol program spustený. Ak sú parametre validné uloží zadaný port do premennej port a názov súboru do database_file_name. Ak nieje zadaný žiaden port tak server počúva na porte 389.

3.2.2 get_port():

Vráti obsah premennej port.

3.2.3 get_database_file_name():

Vráti obsah premennej database_file_name.

3.3 trieda LDAP_tools:

Premenné:

- int size_limit - obsahuje maximálny počet odpovedí ktoré server odošle, ak je hodnota záporná, odpovedí môže odoslať neobmedzene.
- int num_rows - počet riadkov vstupného súboru.
- bool negation - ak má hodnotu true tak je výraz s ktorým sa aktuálne pracuje negovaný, pri vytvorení objektu sa nastaví na false.
- vector<string> cn_vec, vector<string> uid_vec, vector<string> mail_vec - vektory obsahujúce hodnoty cn, uid a mail jednotlivých riadkov databázového súboru.

3.3.1 bool validate_bind_request(string message):

Skontroluje správu a keď sa jedná o validný bind request vráti true, inak false.

3.3.2 bool validate_search_request(string message):

Skontroluje správu a keď sa jedná o validný search request vráti true, inak false.

3.3.3 bool validate_unbind_request(string message):

Skontroluje správu a keď sa jedná o validný unbind request vráti true, inak false.

3.3.4 void load_file(string file_name):

Načíta súbor do troch vektorov cn_vec, uid_vec, mail_vec. Slabé miesto aplikácie je v tom že server načítava celú databázu do 3 vektorov, v prípade príliš rozsiahlej databázy má server veľké pamäťové nároky.

3.3.5 vector<uint8_t> search_request_parse(string message, int socket):

Nastaví premennú size_limit a vytvotrí finálny vektor všetkých výsledkov ktoré sa budú posielať klientovi. Tento vektor sa naplní zavolaním metódy get_result. Keď sa dokončí plnenie tak vráti vektor ktorý obsahuje záznami vyhovujúce kritériám. Ak bol zadaný size_limit tak nadbytočné záznamy odstráni.

3.3.6 vector<uint8_t> form_search_result(int socket, string filter_result, uint8_t message_id):

Parameter filter_result príde vo forme CommonName;Mail;UID. Tento string rozparsuje a vytvorí z neho správu SearchRequestEntry ktorú vráti vo forme vektoru.

3.3.7 vector<string> find_results(string type, string value):

Podľa parametru type prehľadáva vektory obsahujúce položky vstupnej databázy a do vektoru result_vector ukladá záznamy ktoré vyhovujú kritériu parametru value. V databáze môže užívateľ vyhľadávať podľa reťazcov:

- “cn”, “commonname” - meno
- “uid”, “userid” - login
- “mail” - email

Vo vyhľadávaní nezáleží na veľkých a malých písmenách. Metóda vráti vektor obsahujúci všetky riadky databázy ktoré zodpovedajú daným kritériám.

3.3.8 vector<uint8_t> add_length(vector<uint8_t> arg_vector):

Využíva sa pri vytváraní správy SearchResEntry. Pripája hodnotu dĺžky pred vektor ktorý mu príde ako parameter. Vráti výsledný vektor.

3.3.9 int length_value_length(uint8_t value)

Vráti počet bajtov na ktorých je zapísaná dĺžka.

3.3.10 uint32_t length_value(string value)

Vezme postupnosť znakov ktoré udávajú dĺžku bloku a vráti hodnotu ktprú dohromady predstavujú.

3.3.11 vector<string> get_result(string filter):

Metóda vytvorí regulárny výraz ktorým sa bude prehľadávať databáza. Server podporuje 5 druhov filtrov:

1. EqualityMatch (1 parameter) - parameter sa musí presne rovnať záznamu v databázy
2. Substring (1 parameter) - parameter obsahuje jeden alebo viacero znakov “*” ktoré sa môžu rovnať 0 alebo viacerým ľubovoľným znakom. Dva znaky “*” nemôžu následovať za sebou, v tom prípade má parameter zlý formát.
3. Not (1 parameter) - zneguje filter zadaný parametrom - vyhľadáva položky databázy ktoré nevyhovujú danému filtru.
4. And (2 parametre) - filtru vyhovjú iba záznami ktoré vyhovujú obom filtrom zadaným parametrami súčasne. Parametre môžu obsahovať ďalšie filtre obsahujúce And a Or.
5. Or (2 parametre) - filtru vyhovjú iba záznami ktoré vyhovujú aspoň jednému z filtrov zadaných parametrami. Parametre môžu obsahovať ďalšie filtre obsahujúce And a Or.

Keď nieje zadaný žiaden filter tak server vráti všetky položky databázy. Metóda sa môže volať rekurzívne aby sme mohli spracovať filtre And, Or a Not. Server taktiež podporuje viacero filtrov And a OR zanorených v sebe. Keď sa ukončí prehľadávanie metóda vráti vektor obsahujúci všetky položky ktoré vyhovujú danému filtru.

4 Použitie

./myldap *{-p <port>} -f <súbor>*

./myldap -h

- *-p : nepovinný parameter, umožňuje špecifikovať port na ktorom bude server počúvať, ak nieje zadaný tak server počúva na porte 389*
- -f : cesta k textovému súboru vo formáte csv
- -h : zobrazí nápovedu použitia programu

5 Testovanie

Projekt bol testovaný na referenčnom serveri merlin.fit.vutbr.cz pomocou nástroja ldapsearch.

5.1 Na strane servera

- ./myldap -p 45678
- ./myldap -f test.csv

5.2 Na strane klienta

- ldapsearch -h merlin.fit.vutbr.cz -p 45679 -x
- ldapsearch -h merlin.fit.vutbr.cz -p 45679 -x "(userid=xzarsk03)"
- ldapsearch -h merlin.fit.vutbr.cz -p 45679 -x "(!(mail=x*))"
- ldapsearch -h merlin.fit.vutbr.cz -p 45679 -x "(commonname=*va*i*l)"
- ldapsearch -h merlin.fit.vutbr.cz -p 45679 -x "(common=*va*i*l)"
- ldapsearch -h merlin.fit.vutbr.cz -p 45679 -x "(|(&(cn=*a*)(!(cn=*B*)))(mail=*mil*))"
- ldapsearch -h merlin.fit.vutbr.cz -p 45679 -x "(|(cn=*mil*) (cn=fffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff*f))"

6 Bibliografia:

- Síťové aplikace a jejich architektura, Ing. Petr Matoušek, PhD.
- https://cwiki.apache.org/confluence/display/DIRxSRVx10/Ldap+ASN.1+Codec
- https://tools.ietf.org/html/rfc4511
- https://en.wikipedia.org/wiki/Lightweight_Directory_Access_Protocol

7 Záver

Tento projekt mi priblížil fungovanie LDAP protokolu. Pri vypracovávaní som sa naučil kódovať a dekódovať BER kódovanie a čítať a vytvárať správy v LDAP ASN.1 gramatike. Projekt neimplementuje žiadne bonusové rozšírenia.