Universitatea Alexandru Ioan Cuza

Facultatea de Informatica

Password Manager

Glodeanu Irina-Elena

Anul 2, grupa A6

irina.glodeanu@info.uaic.ro

**Cuprins**

1. Introducere
2. Tehnologiile utilizate (e.g. TCP, UDP... - caracteristicile care au fost utilizate in cadrul proiectului)
3. Arhitectura aplicatiei (conceptele implicate, diagrama aplicatiei)
4. Detalii de implementare (cod relevant, comentat; *use-case*-uri)
5. Concluzii (e.g. cum ar putea fi imbunatatita solutia propusa? )
6. Bibliografie

**1.Introducere**

Aceasta este o aplicatie client/server care sa ofere functionalitatea unui manager de parole.

Aplicatia va suporta conturi de utilizator multiple, fiecare utilizator autentificandu-se pe baza unei parole master.

Fiecare utilizator va putea sa-si organizeze parolele salvate in diferite categorii, iar pentru fiecare parola va avea posibilitatea introducerii unor informatii precum titlu, username, url, notite etc.

**2. Tehnologiile utilizate**

Pentru aceasta aplicatie folosesc un server TCP concurrent (engl. Transmission Control Protocol , Protocolul de control al transmisiei).

Acest protocol realizeaza o conexiune viruala full-duplex intre client si server intre care se face schimb de informatie(flux de octeti).

Am considerat ca acest protocol este mai potrivit pentru aplicatia in cauza decat UDP deoarece comunicarea se face constant intre client si server deopotriva( TCP-ul fiind un protocol orientat conexiune). De asemenea, avem nevoie de fiabilitatea data de acest protocol deoarece pentru o buna functionare a aplicatiei, informatiile transmise intre cilent si server nu pot fi pierdute sub nicio forma (orice pierdere poate duce a nefunctionarea aplicatiei). Din acelasi motiv, folosim TCP-ul deoarece ne este asigurata livrarea pachetelor de date in ordinea in care au fost transmise. In plus, fiind un protocol folosit de obicei de aplicații care au nevoie de confirmare de primire a datelor, error checking-ul, message acknoledgement-ul (receprorul raspunde cu un mesaj de confirmare) si retransmiterea in caz de octeti pierduti (caracteristici pe care UDP-ul nu le are), consider ca acest protocol reprezinta cea mai potrivita alegere din punctul de vedere al functionalitatii necesare aplicatiei mele.

In cazul detectiei de probleme precum pierderea, duplicarea sau livrarea in alta ordine decat cea initiala a datelor(datorita congestiilor de retea, incarcarea traficului, etc.), TCP-ul solicita retransmisia pachetelor pierdute, rearanjează pachetele în ordine si ajută la minimizarea traficul din rețea în vederea reducerii apariției altor probleme.

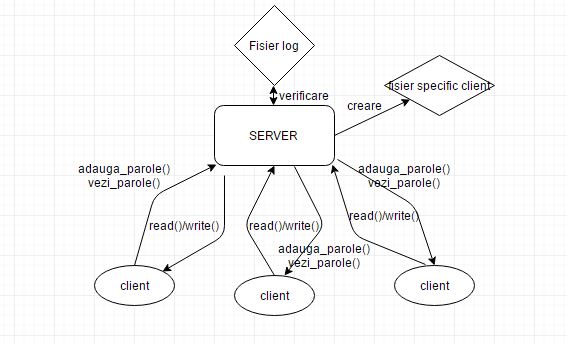
Desi TCP-ul nu are ca principal scop livrarea foarte rapida a datelor (pentru aceasta ar fi trebuit sa folosim UDP), ci mai degraba a livrarii lor exacte, acest lucru nu ne deranjeaza, deoarece , in cazul aplicatitei de management al parolelor, transmisia sigura a datelor este prioritara.

**3.Arhitectura aplicatiei (conceptele implicate, diagrama aplicatiei)**

Aplicatia va folosi o arhitectura de tipul CLIENT/SERVER TCP concurent.

Serverul va primi cererile de conectare de la clienti, dupa care va oferi acestora posibilitatea de a adauga sau vedea parole doar in cazul in care clientii detin un cont si o parola master. Daca numele clientului sau parola nu corespund cu un user autorizat sa acceseze aplicatia, acestuia i se da ocazia sa reintroduca un nume si o parola autorizata.

Odata cu verificarea de catre server existentei acelui user, clientului i se ofera posibilitatea sa adauge mai intai o categorie si mai apoi o noua parola apartinand acesteia impreuna cu niste informatii despre ea (Titlu, Username, URL, Notite) sau pur si simplu sa vizualizeze parolele si informatiile aferente apartinand acelei categorii.

****

**4.Detalii de implementare (cod relevant, comentat; *use-case*-uri)**

Dupa activarea serverului si mai apoi a conectarii concurente a clientilor la portul aferent, acestia au posibilitatea logarii doar daca introduc un nume si parola sa master care sa se gaseasca intr-un fisier cu aceste informatii (nume\_pass.txt). Se creeaza un anumit numar de thread-uri cu ajutorul functiei *threadCreate(...)* care vor fi alocate de server pentru fiecare cient in parte

Numele si parola sunt transmise serverului cu un *write()* care le primeste cu un *read()* astfel incat sa se poata face verificarea existentei lor in fisierul de mai sus. Pentru evetualitatea greselii informatiilor de logare, atat serverul cat si clientul au o bucla care nu lasa inaintarea in cadrul aplicatiei decat in cazul introducerii unor informatii valide.

Dupa ce un anumit client va transmite serverului informatiil corecte de autentificare , serverul va da posibilitatea clientului sa aleaga ce doreste sa faca mai departe: sa vada parolele dintr-o categorie( executia aplicatiei va fi directionata spre functia *vezi\_parole())* sau sa adauge o noua parola intr-o categorie cu informatiile de rigoare(functia *adauga\_parole()).*

**Avem structura account (explicatia in comentariu)**

struct account {

char id[20];

char password[20];

};

int numar\_de\_conturi;

static struct account accounts[10];

/\*imi verifica cate linii (conturi) am de toate

si imi pune pentru fiecare linie in structura

accounts[i].id-numele de pe linia i

si in accounts[i].password- parola pentru

numele de pe linia i\*/

void read\_file(struct account accounts[])

{

FILE \*fp;

int i=0; // cate finii sunt in fisier

int c;

fp=fopen("nume\_pass.txt", "r");

while(!feof(fp)) {

c=fgetc(fp);

if(c=='\n')

++i;

}

int j=0;

rewind(fp); //seteaza pozitia indicatorului asociat cu fp

// la inceputul fisierului

while(j!=i-1) {

fscanf(fp, "%s %s", accounts[j].id, accounts[j].password);

++j;

}

numar\_de\_conturi=j;

}

**Connect-ul dintre client(sus) si server(jos)**

***client:***

if ((sd = socket (AF\_INET, SOCK\_STREAM, 0)) == -1)

{

perror ("Eroare la socket().\n");

return errno;

}

/\* umplem structura folosita pentru realizarea conexiunii cu serverul \*/

/\* familia socket-ului \*/

server.sin\_family = AF\_INET;

/\* adresa IP a serverului \*/

server.sin\_addr.s\_addr = inet\_addr(argv[1]);

/\* portul de conectare \*/

server.sin\_port = htons (port);

/\* ne conectam la server \*/

if (connect (sd, (struct sockaddr \*) &server,sizeof (struct sockaddr)) == -1)

{

perror ("[client]Eroare la connect().\n");

return errno;

}

***server:***

/\* crearea unui socket \*/

if ((sd = socket (AF\_INET, SOCK\_STREAM, 0)) == -1)

{

perror ("[server]Eroare la socket().\n");

return errno;

}

/\* utilizarea optiunii SO\_REUSEADDR \*/

int on=1;

setsockopt(sd,SOL\_SOCKET,SO\_REUSEADDR,&on,sizeof(on));

/\* pregatirea structurilor de date \*/

bzero (&server, sizeof (server));

/\* umplem structura folosita de server \*/

/\* stabilirea familiei de socket-uri \*/

server.sin\_family = AF\_INET;

/\* acceptam orice adresa \*/

server.sin\_addr.s\_addr = htonl (INADDR\_ANY);

/\* utilizam un port utilizator \*/

server.sin\_port = htons (PORT);

/\* atasam socketul \*/

if (bind (sd, (struct sockaddr \*) &server, sizeof (struct sockaddr)) == -1)

{

perror ("[server]Eroare la bind().\n");

return errno;

}

/\* punem serverul sa asculte daca vin clienti sa se conecteze \*/

if (listen (sd, 2) == -1)

{

perror ("[server]Eroare la listen().\n");

return errno;

}

//printf("Nr threaduri %d \n", nthreads);

fflush(stdout);

int i;

for(i=0; i<nthreads;i++) threadCreate(i);

**Functia de creare a thread-ului din server(cea de mai sus)**

void threadCreate(int i)

{

void \*treat(void \*);

pthread\_create(&threadsPool[i].idThread,NULL,&treat,(void\*)i);

return; /\* threadul principal returneaza \*/

}

void \*treat(void \* arg)

{

int client;

struct sockaddr\_in from;

bzero (&from, sizeof (from));

fflush(stdout);

for( ; ; )

{

int length = sizeof (from);

pthread\_mutex\_lock(&mlock);

if ( (client = accept (sd, (struct sockaddr \*) &from, &length)) < 0)

{

perror ("[thread]Eroare la accept().\n");

}

printf("Thread %d trezit\n",(int)arg);

pthread\_mutex\_unlock(&mlock);

threadsPool[(int)arg].thCount++;

raspunde(client,(int)arg); //procesarea cererii

/\* am terminat cu acest client, inchidem conexiunea \*/

close (client);

}

};

**Loop-ul din functia “raspunde()”care nu lasa sa operam cu un username/parola care nu se gaseste in nume\_pass.txt**

***server:***

while(gasit==0) //loop-ul pentru introducerea de info valide

{if (read (cl, &msg,sizeof(msg)) <= 0) //citesc ce imi trimite clientul

{

printf("[Thread %d]\n",idThread);

perror ("Eroare la primul read() \n");

}

char find=' ';

int i=0, index=0;

char \*ptr;

ptr=strchr(msg,find);

index=(ptr-msg);

//impart stringul trimis de client in numele si parola aferenta

strncpy(user,msg,index);

strncpy(pass,msg+index+1,(strlen(msg)-index)-2); //asa da

for(i=0;i<numar\_de\_conturi;i++)

if(strcmp(user,accounts[i].id)==0 && strcmp(pass,accounts[i].password)==0)

{ //daca acestea se gasesc in fisierul nume\_pass.txt ies din loop

nr2=2;

gasit=1;

write (cl, &nr2, sizeof(nr2)); //trimit la client sa iasa si el din loop

}

if(gasit==0) write (cl, &nr2, sizeof(nr2));

//daca nu, trimit la client o valoare care il

//va face s intre si el inca odata in loop

}

***client:***

int nr2=0;

while(nr2!=2) //loop-ul pana se introduce ceva valid de la consola

{

/\* citirea mesajului \*/

printf ("[client]Introduceti un nume si parola master: ");

fflush (stdout);

read (0, buf, sizeof(buf));

printf("[client] Am citit %s\n",buf);

/\* trimiterea mesajului la server \*/

if (write (sd,&buf,sizeof(buf)) <= 0)

{

perror ("[client]Eroare la write() din loop\n");

return errno;

}

if(read (sd, &nr2,sizeof(nr2))<0)

{

perror ("[client]Eroare la al doilea read() \n");

return errno;

}

}

**Loop-ul care nu ne lasa sa alegem o alta optiune decat cele listate(adauga parole/vizualizare)**

***server:***

**if (read (cl, &optiune,sizeof(int)) <= 0)**

**{**

**perror ("[server]Eroare la read() la primirea optiunii clientului.\n");**

**}**

**if (optiune==1) //daca am ales sa vizualizam informatiile**

**{**

**printf("Clientul %d a ales sa vada \n", idThread);**

**vezi\_parole(parolele\_userului,cl);**

**}**

**if(optiune==2) //daca am ales sa adaugam parola si informatii**

**{**

**printf("Clientul %d a ales sa adauge \n", idThread);**

**adauga\_parole(parolele\_userului,cl);**

**}**

***client:***

int ok=0;

while(ok==0)

{

printf("Doriti sa vedeti parolele sau sa adaugati? \n 1.Vad \n 2.Adaug \n");

scanf("%d", &optiune);

if((optiune==1)||(optiune==2)) ok=1;

else printf("Alege intre 1 sau 2!! \n");

}

if (write (sd,&optiune,sizeof(int)) <= 0) //trimitem optiunea la server

//pentru a face actiunea specifica

{

perror ("[client]Eroare la write() spre server la alegere.\n");

return errno;

}

Restul functiilor care realizeaza efectiv functionalitatea aplicatiei se gasesc in arhiva adaugata.Acestea se gasesc in server deoarece singura functionalitate a clientului e sa faca cereri pe care serverul sa le indeplineasca.

Astfel avem(inafara de ceea ce am specificat deja):

Functii:

* vezi\_parole(FILE \*fisier, int cl) – cauta in fisierul specific unui anumit user parolele si informatiile unei categorii cerute de catre client si le afiseaza
* adauga\_parole(FILE \* filename, int cl) – primeste de la client ce trebuie sa introduca in fisierul cu parole si cu ajutorul functiei adauga\_info(...) acestea sunt introduse in fisierul specific fiecari admin
* adauga\_info(FILE \*fisier,char categorie[100],char parola[100],char titlu[100],char user[100],char url[100],char notite[100]) -pune in fisierul aferent fiecarui admin informatiile trimise de client

Fisiere:

* nume\_pass.txt –fisierul unde se gasesc pe fiecare linie numele si parola clientilor cu drept de a folosi aplicatia
* ion2.txt –fisierl in care clientul cu numele ion si parola 2 are dreptul de a-si adauga parole cu informatiile lor de rigoare clasate pe categorii sau sa vizualizeze parolele aferente unei categorii pe care trebuie s-o specifice de la tastatura
* alex123.txt –idem ion2.txt
* .....

**5.Concluzii (e.g. cum ar putea fi imbunatatita solutia propusa? )**

Utilitare care se mai pot aduce(implementa) aplicatiei pot fi:

-posibilitatea stergerii unei parole impreuna cu informatiile ce tin de ea

-posibilitatea modificarii unor informatii ce tin de o anumita parola

-posibilitatea de inregistrare/stergere a unui cont din fisierul nume\_pass.txt

**6.Bibliografie**

<http://en.wikipedia.org/wiki/Transmission_Control_Protocol>

<http://students.info.uaic.ro/~adria/teach/courses/net/cursullaboratorul.php>

<http://www.diffen.com/difference/TCP_vs_UDP#How_TCP_and_UDP_work>

Informatiile acumulate in timpul cursurilor si laboratorului de “Retele de calculatoare”

<http://stackoverflow.com/questions/5970383/difference-between-tcp-and-udp>