DOCUMENTAȚIE 2FA

REȚELE DE CALCULATOARE

MISTREANU EMANUELA-2A4

FACULTATEA DE INFORMATICĂ-IAȘI

My2FA (B)

Dezvoltati o aplicatie de tip client/server, care oferă gestionarea codurilor si notificarilor de tip 2FA (two-factor authentication) pentru o lista predefinita de aplicații. In cadrul serverului, codurile 2FA vor fi regenerate recurent pentru fiecare aplicatie si vor putea fi aratate la cererea clientului dezvoltat pentru aceasta aplicatie. In momentul in care serverul 2FA primeste o notificare de autentificare pentru una din aplicatiile pe care le are in gestiune, acesta il va notifica pe clientul 2FA mai departe si ii va cere aprobarea sau respingerea acesteia. Pentru a valida functionalitatea aplicatiei 2FA, un client si un server aditional vor fi dezvoltate. Serverul aditional va juca rolul aplicatiei unde utilizatorul incearca sa se autentifice prin intermediul clientului aditional. In momentul in care clientul initializeaza o cerere de autentificare la serverul aditional prin intermediul clientului aditional, i se vor prezenta doua optiuni: trimiterea unei notificari in aplicatia de 2FA pentru a confirma identitatea sau introducerea unui cod din aplicatia de 2FA. In cazul notificarii, daca se primeste un raspuns pozitiv de la serverul 2FA, autentificarea in aplicatia aditionala va functiona, altfel va esua. In cazul utilizarii mecanismului bazat pe coduri, codul introdus de utilizator in clientul aditional, va fi preluat de serverul aditional si verificat cu serverul de 2FA, pentru a decide daca autentificarea s-a realizat cu succes. Bonus: mecanism de stocare criptat, interogari criptate.

Sus

1 INTRODUCERE

Proiectul 2FA reprezinta o aplicatie de tip client/server care permite autentificarea unui utilizator intr-o aplicatie dintr-o lista predefinita. Autentificarea se realizeaza in 2 pasi, initial utilizatorul introduce un username si o parola, iar pentru validarea autenticitatii trebuie sa mai parcurga inca un pas.

Utilizatorii pot sa se inregistreze pentru prima data intr-o aplicatie cu un username si o parola iar apoi vor fi adaugati in baza de date. Daca clientul doreste sa se autentifice, avand cont deja, acesta primeste doua optiuni: autentificare ca utilizator sau autentificare ca dezvoltator. Pentru amandoua optiuni lucrurile decurg in acelasi mod: utilizatorul introduce username-ul si parola iar apoi dupa validarea existentei acestora in baza de date, primeste 2 variante: autentificare prin cod sau prin notificare. In cazul codului, utilizatorul trebuie sa introduca codul din aplicatia in care doreste sa se autentifice, validarea acestuia facandu-se in serverul aplicatiei, iar in cazul notificarii, o notificare este trimisa clientului in aplicatia in care doreste sa se conecteze, de unde poate sa accepte sau nu, aceasta cerere. Pentru utilizatorii conectati ca dezvoltatori, acestia au acces la codurile aplicatiei sale, asadar la cerere, serverul trimite informatiile din fisierul cu codurile respective.

2 TEHNOLOGII UTILIZATE

2.1 STOCAREA DATELOR

Codurile de la fiecare aplicatie sunt stocare in fisiere cu nume specifice app1.txt, ap2.txt, ap3.txt , sub forma unui int corespuzator unui cod pe fiecare linie (pentru a putea pastra un index pentru fiecare apliatie in server si sa atribuim un cod o singura data unui singur utilizator).

Pentru baza de date ce retine toti utilizatorii inregistrati pe fiecare aplicatie de asemenea exista cate un fisier care contine datele sub forma username1, parola1, username2, parola2, etc. (primul username, parola fiind al clientului dezvoltator), in acest mod putand valida corectitudinea datelor atunci cand un utilizator sau un dezvoltator doreste sa se autentifice..

2.2 LIMBAJ

Aplicatia MY2FA foloseste ca limbaj de programare C.

2.3 PROTOCOL

Proiectul are un protocol de comunicare de tipul TCP, un protocol in care clientul si serverul sunt conectate, deoarece:

- TCP confera o siguranta a datelor: mesajele trimise ajung intotdeauna la destinatar, de asemenea sunt primite integral (nedorindu-ne ca de exemplu dezvoltatorul sa trimita o cerere pentru obtinerea codurilor aplicatiei sale si sa primeasca doar jumatate din acestea, sau nici macar sa nu ajunga la dezvoltator);

- TCP poate fi folosit pentru bulk data transfer mai exact transfer de fisiere (aplicatia transfera fisierul cu codurile clientului dezvoltator);

- TCP asigura transmiterea in ordine a mesajelor(importatant in cazul in care transmitem username ul si parola, vrem sa fie exact in acesta ordine altfel consecutivitatea nu va fi respectata in baza de date folosita (fisiere text) si utilizatorul nu poate fi autentificat desi este inregistrat in aplicatia respectiva);

- TCP este orientat spre conexiune. TCP necesită ca conexiunea între două puncte la distanță să fie stabilită înainte de trimiterea datelor reale (asadar mereu datele ajung la destinatia dorita);

- TCP oferă server full duplex, adică poate îndeplini atât roluri de receptor, cât și de expeditor. (proprietate folosita de exemplu in urmatoare situatie: serverul primeste codul la care trebuie sa trimita un raspuns- bidirectional).

3 ARHITECTURA APLICATIEI

Aplicatia MY2FA este alcatuita din client, server, un client additional, un server aditional si baza de date (fisiere de tip txt).
Comenzile initiale care apar la compilarea programului:

- **APP1** -autentificarea la aplicatia app1;
- **APP2**- autentificarea la aplicatia app2;
- **APP3** -autentificarea la aplicatia app3;
- **EXIT** -iesirea din program.

Dupa alegerea aplicatiei urmatoarele comenzi vor fi afisate:
- **INREGISTRARE** - pentru un nou utilizator, se cere un username si o parola de la utilizator, iar daca username-ul nu se gaseste in baza de date, introduce datele in baza de date;
- **AUTENTIFICARE CA UTILIZATOR** -in cazul in care utilizatorul vrea sa se conecteze se cere username-ul si parola.;
- **AUTENTIFICARE CA DEZVOLTATOR** - in cazul in care utilizatorul vrea sa se conecteze ca dezvoltator pentru a avea acces la unele informatii din aplicatie, de asemenea se cere username-ul si parola.;
- **EXIT**-iesire din program.

Daca una din comenzile de autentificare sunt alese, apar urmatoarele comenzi:
- **AUTENTIFICARE PRIN NOTIFICARE** -In clientul aplicatiei apare o notificare de forma “INCERCATI SA VA CONECTATI DE PE ALT DISPOZITIV”, de unde utilizatorul poate trimite un raspuns de forma “DA”/”NU”;
- **AUTENTIFICARE PRIN COD** -Un cod va aparea in clientul aplicatiei 2FA, pe care utilizatorul il poate vizualiza si il poate introduce in clientul aditional pentru a fi validat de aplicatie;
- **EXIT**-iesire din program.

Daca autentificarea s-a produs cu success si utilizatorul s-a conectat ca dezvoltator o noua comanda apare:
- **CODURI APLICATIE** -prin care dezvoltatorul poate cere codurile aplicatiei, pe care serverul le trimite sub forma unui fisier

Pentru a ilustra functionalitatea proiectului am adaugat urmatoarele diagrame:
- Diagrama generala a aplicatiei (Fig 1)
- Diagrama explicita a aplicatiei (Fig 2)

Fig. 1. DIAGRAMA GENERALA APLICATIE

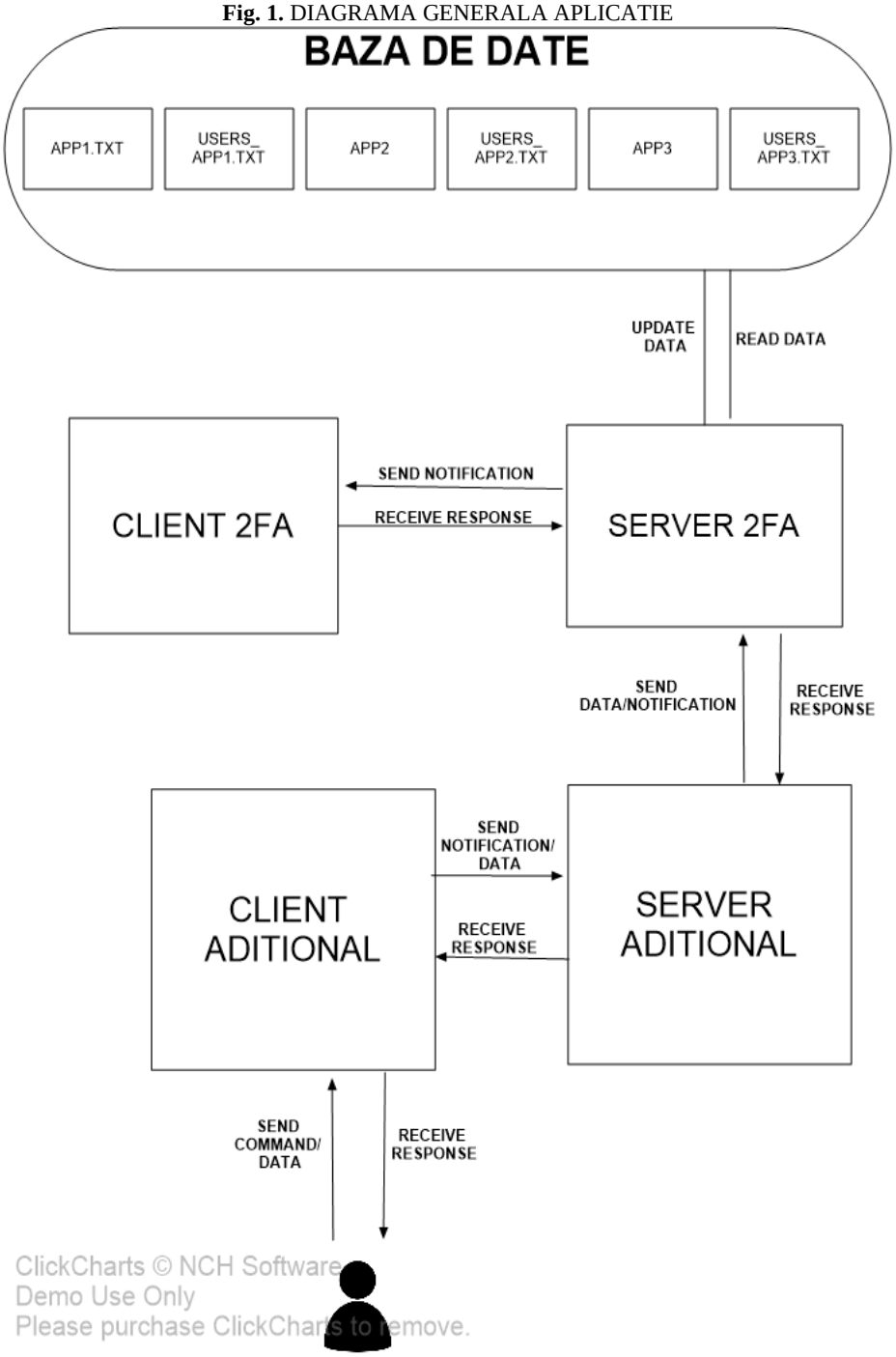
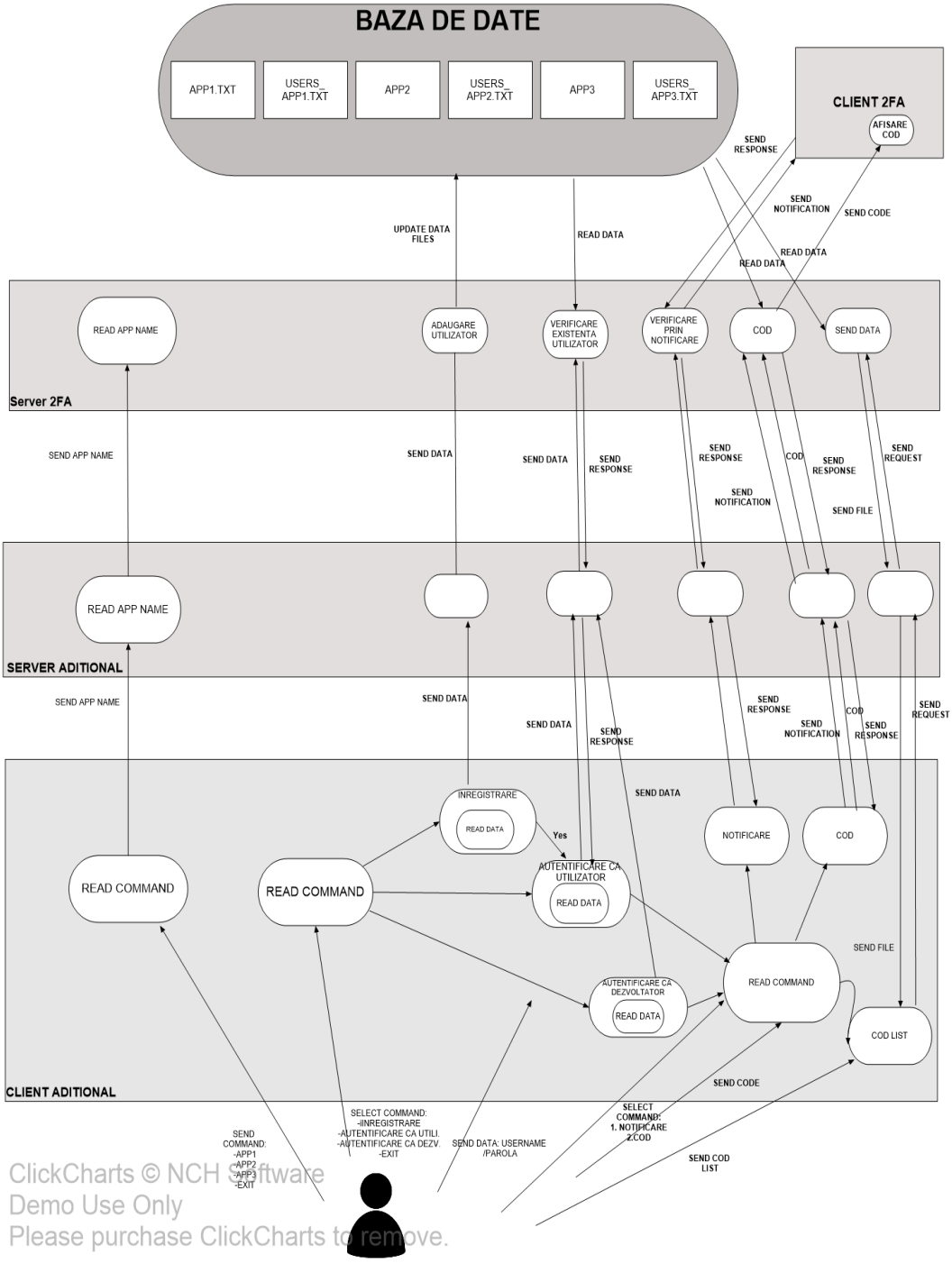


Fig. 2. DIAGRAMA EXPLICITA APLICATIE



Mai explicit programul functioneaza in modul urmator:

Utilizatorul comunica cu serverul care gestioneaza autentificarile in doi pasi prin intermediul unui client si a unui server aditional. Utilizatorul comunica direct cu clientul in care trimite comenzile dorite: mai intai aplicatia in care vrea sa se conecteze, apoi modul in care vrea sa se conecteze(sau sa se inregistreze caz in care datele sunt trimise serverului aplicatiei 2FA care actualizeaza baza de date, utilizatorul revenind la comenzile de autentificare) si datele sale: username-ul si parola, acestea sunt transmise serverului aditional, iar din acesta in serverul aplicatiei 2FA pentru a fi validate, daca acestea se regasesc in baza de date un raspuns pozitiv este trimis de la serverul 2FA la serverul aditional, apoi in clientul aditional si in final la utilizator, apoi mai urmam un pas pentru autentificare: utilizatorul poate alege validarea prin notificare: caz in care o notificare este trimisa la serverul 2FA ce informeaza clientul 2FA care accepta sau respinge cererea, aplicatia trimite raspunsul server-ului aditional, care transmite clientului aditional si apoi utilizatorului. In cazul codului, utilizatorul trebuie sa introduca un cod din aplicatie (mai clar spus, atunci cand utilizatorul selecteaza comanda cod, o notificare este trimisa server-ului care trimite din baza de date un cod catre clientul aplicatiei, acesta afisandu-l in terminal de unde il poate prelua utilizatorul) care trece prin server-ul aditional, si este validat in baza de date a serverului aplicatie 2FA. De asemenea raspunsul este primit in cadrul serverului aditional, apoi ajunge la client si in cele din urma la utilizator. In cazul dezvoltatorilor, serverul aditional poate notifica serverul aplicatiei faptul ca acesta doreste codurile aplicatiei, de asemenea acesta trece prin pasii autentificarii.

4 DETALII IMPLEMENTARE:

Modul de comunicare intre server-client (atat cel aditional cat si cel al aplicatiei 2FA) este prin **SOCKET** (am ales acest mod de comunicare deoarece este bidirectional).

De asemenea intre serverul aditional si serverul aplicatiei comunicare este tot prin intermediul **SOCKET-ului** (serverul aditional trimite date/notificari iar serverul aplicatiei trimite confirmari/respingeri).

```
/* cream socketul */
if ((sd = socket (AF_INET, SOCK_STREAM, 0)) == -1)
  {
    perror ("Eroare la socket().\n");
    return errno;
  }
```

Concurenta server-ului am rezolvat-o folosint **THREAD-uri**, fiind o varianta mai rapida in defavoarea fork()-ului. (Se folosesc acelasi servere nefiind nevoie de crearea unor noi procese de fiecare data, lucru ce duce la evitarea incarcarii aplicatiei.

In cadrul proiectului am folosit urmatoarele functii:

- o functie care sa **genereze recurent** codurile pentru fiecare aplicatie

– Acesta functie creaza coduri adaugand la codul precedent o valoare (1) , iar apoi scrie fiecare cod pe cate o linie din fisier,atunci cand un nou client doreste autentificarea prin cod.

```
int cod(FILE *f)
  {
          char info[20], nr[20];
          while(fscanf(f, "%s", info)!=EOF){
            strcpy(nr,info);
            }
            int n;
            n=atoi(nr);
            n=n+1;
            char nr1 = (char) n;
           fseek(f, 0, SEEK_END);
           fprintf(f, "%d\n", nr1);
           return nr1;
           fclose(f);
          }
```

- o functie care sa **inregistreze un nou utilizator**

– clientul aditional preia informatiile de la utilizator, le transmite serverului aditional, iar acesta serverului 2FA, unde functia este apelata
– Mai exact acesta preia informatiile username si parola, cauta in fisierul text ca username-ul sa nu se regaseasca iar daca aceasta conditie este indeplinita, pune pe o nou linie din fisier username si parola cu un spatiu intre acestea.

```
int inregistrare(char user[20], char par[20], FILE *f)
{
          char info[100], info1[100];
          int g=0;
          while(fscanf(f, "%s", info)!=EOF){
          if(strcmp(info, "username")==0)
          {
           fscanf(f, "%s", info1);
             if(strcmp(info1,user)==0)
              {
                return 0;
                g=1;
               }
              }
              }
              if(g==0){
              fseek(f, 0, SEEK_END);
              fprintf(f, "username %s ", user);
              fseek(f, 0, SEEK_END);
              fprintf(f, "parola %s\n", par);
              fclose(f);
              return 1;
          }
          }
```

- o functie care sa **valideze username-ului si parola** cand un client vrea sa se autentifice

— Acesta functie va prelua informatiile si va cauta in fisierul users_app existenta username-ului si a parolei pe o linie din acesta, daca se regasesc returneaza 1 altfel returneaza 0.

```
        }
int logare(char *user, char *par, FILE *f)
{
          char info[20], info1[20];
          int g=0;
          fscanf(f, "%s", info); //citim primele 4 cuvinte- linia dezvoltatorului
          fscanf(f, "%s", info);
          fscanf(f, "%s", info);
          fscanf(f, "%s", info);
          while(fscanf(f, "%s", info)!=EOF){
          if(strcmp(info, "username")==0)
          {
           fscanf(f, "%s", info1);
           if(strcmp(info1,user)==0)
             {
               fscanf(f, "%s", info1); //cuv: parola
               fscanf(f, "%s" , info1);
               if (strcmp(info1, par)==0)
                return 1;
                }
                }
                }
                return 0;
                }
```

- o functie care **valideze codul** primit de la utilizator in fisierele text

— acesta verifica ca codul trimis catre client si afisat in aplicatie sa corespunda codului introdus de utilizator

```
int validare_cod(int a, int b)
  {
    if (a==b)
       return 1;
    else
       return 0;
    }
```

- o functie care **verifica daca utilizatorul este dezvoltator** in cazul aceasta comanda este preluata:

– Acesta functie verifica ca username-ul si parola introduse sa se regaseasca pe prima linie a fisierului. (aceste date sunt completate by default in aplicatie inca de la creearea acesteea si nu pot fi modificate)

```
int logared(char *user, char *par, FILE *f)
{
          char info[20], info1[20];
          int g=0;
           fscanf(f, "%s", info1);
           fscanf(f, "%s", info1); //username
            if(strcmp(info1,user)==0)
              {
           fscanf(f, "%s", info);
           fscanf(f, "%s", info);
           if (strcmp(info1, par)==0)
                return 1;
                }
                return 0;
                }
```

Toate aceste functii sunt implementate in cadrul server-ului aplicatiei 2FA.

5 CONCLUZII

- Din punct de vedere a stocarii datelor, fisierele text nu sunt cele mai rapide cand vine vorba de cautarea unor date in raport de timp, asadar o baza de tipul SQL ar putea imbunatati semnificativ timpul de raspuns al aplicatiei 2FA. Comenzile select sunt pronuntat mai rapide decat cautarea linie cu linie intr-un fisier text, de asemenea cautarea unui utilizator nu mai implica relevanta consecutivitatii username, parola.

- Aplicatia 2FA reprezinta o modalitate de crestere a securitatii conectarii unui utilizator la o aplicatie, asadar un alt punct ce ar putea fi adaugat aici ar fi criptarea bazei de date, ce ridica expontial acesta securitate..

- Aplicatia dezvoltata ar putea fi imbunatatita la nivelul codului prin implementarea diferita in cazul unui raspuns negativ, utilizatorul nefiind scos de tot din aplicatie, ci cerandu-i-se un cod diferit, sau un username si o parola diferita..

6 References

1. LNCS Homepage, http://www.springer.com/lncs;
2. Computer Networks course : https://profs.info.uaic.ro/~computernetworks/files/5rc_ProgramareaInReteaI_ro.pdf
3. Computer Networks course : https://profs.info.uaic.ro/~computernetworks/files/6rc_ProgramareaInReteaII_Ro.pdf
4. Computer Networks course: https://profs.info.uaic.ro/~computernetworks/files/7rc_ProgramareaInReteaIII_Ro.pdf
5. Computer Networks lab: https://docs.google.com/document/d/1udCcy98st54i1zszvgtfxnIZnxoM-xevdXnxLLS-lqM/edit
6. https://duo.com/product/multi-factor-authentication-mfa/two-factor-authentication-2fa