My2FA
Raport tehnic

Mocanita Robert-Daniel (2A3)

06 - Decembrie - 2022

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.

1 Introducere

Proiectul My2FA se bazeaza pe o comunicare de tip client/server prin care se simuleaza functionarea unui sistem de tip Two-Factor Authentication. Acesta este compus din doua perechi server-client : o pereche ce gestioneaza sistemul de autentificare si inca una aditionala care joaca rolul aplicatiei in care se incearca autentificarea.
Serverul are o lista de aplicatii predefinite pentru care genereaza in mod continuu coduri de tip 2FA, acestea incepand sa fie create din momentul in care serverul este pornit. La cererea serverului aditional, serverul 2FA poate manipula doua tipuri de autentificare : prin confirmare directa sau manuala. In cazul confirmarii directe, acesta asteapta un raspuns *da/nu* de la clientul 2FA si realizeaza (sau nu) autentificarea iar in cazul metodei manuale, acesta trimite un cod 2FA, clientul il afiseaza iar utilizatorul va putea introduce in clientul aditional codul de autentificare.

2 Tehnologiile utilizate

2.1 TCP/IP

Protocolul utilizat de aplicatie este TCP/IP (protocol orientat-conexiune) deoarece, fiind o aplicatie de securitate, e nevoie sa asiguram o conexiune fiabila, fara o posibila pierdere de informatii.
Protocolul de Control al Transmisiei efectueaza o conectare virtuala full duplex intre doua puncte terminale, fiecare punct fiind definit de catre o adresa IP si de catre un port TCP. Acesta ofera incredere, asigurand livrarea ordonata a fluxurilor de octeti de la un program la altul din retea.

2.2 Socket-uri

Pentru realizarea conexiunii client-server a fost necesara utilizarea socket-urilor.
Programarea prin socket-uri reprezinta o abordare low-level prin care doua calculatoare (programe) pot fi conectate pentru a realiza un schimb de date. Ca principiu de baza programarea prin socketuri face posibila comunicarea in full-duplex intre client si server, mesajele fiind fluxuri de octeti.

Un socket reprezinta un punct de conexiune intr-o retea TCP/IP. Serverul deschide un socket si asteapta conexiuni iar clientul, cunoscand adresa socket-ului si portul acestuia, se poate conecta la server pentru a incepe schimbul de informatii.

2.3 Thread-uri

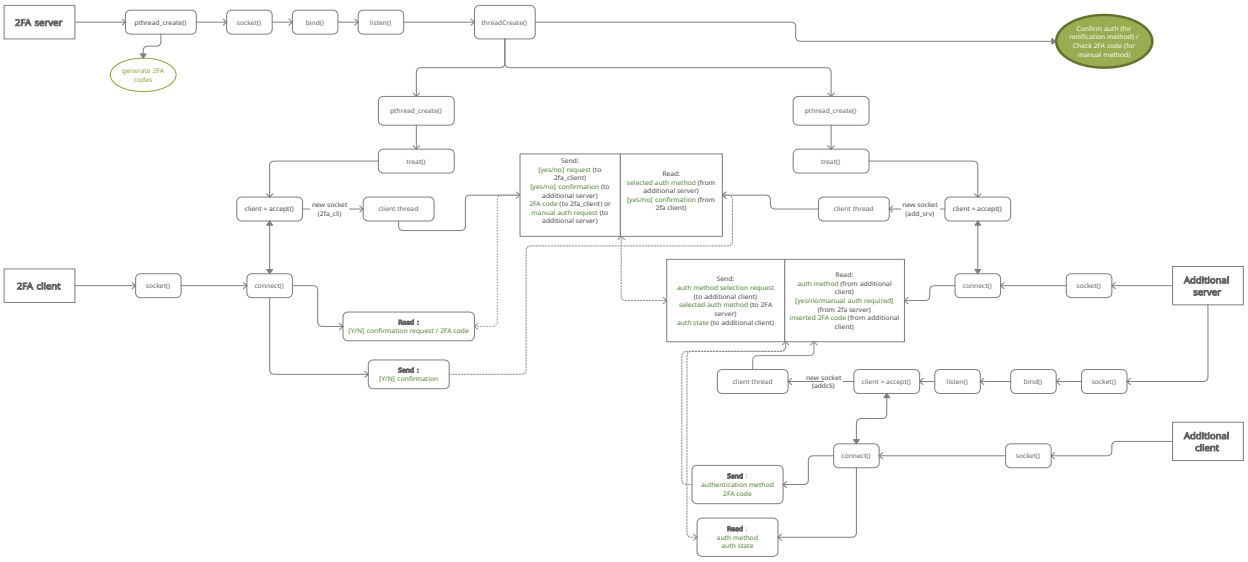
Am ales sa folosesc modelul *prethreaded* pentru realizarea serverului care gestioneaza autentificarea 2FA. Astfel creez doua threaduri, unul ce deserveste clientului 2FA iar al doilea pentru serverul aditional.
Acest fapt imi permite servirea celor doi clienti in mod concurent si schimbarea de mesaje intre ei, intr-un mod mai avantajos (din punct de vedere al timpului si resurselor) decat crearea de procese pentru fiecare client.
De asemenea, un thread separat este folosit si pentru generarea de coduri 2FA, pentru ca aceasta generare sa se produca in mod continuu fara a se bloca in timpul apelurilor de citire/scriere efectuate de server.

2.4 Randomizer

Pentru generarea de coduri 2FA am ales folosirea primitivelor *rand*(), respectiv *srand*().
Prin *rand*() generez fiecare simbol (pseudo)aleator dintr-un cod 2FA iar cu ajutorul *srand*() setez la fiecare cod nou o noua valoare de seed (bazata pe timpul la care s-a executat generarea) pentru a ma asigura ca se vor genera coduri diferite (diferite de la cod la cod intr-o aplicatie dar diferite si intre aplicatii diferite).

3 Arhitectura aplicatiei

Aplicatia 'My2FA' se foloseste de doua perechi server-client.
Serverul 2FA va accepta conexiuni de la clientul 2FA dar si de la serverul aditional. Clientul 2FA va schimba mesaje doar cu serverul sau. Serverul aditional se va conecta la serverul 2FA si va primi mesaje de la clientul aditional, cel din urma comunicand doar cu serverul aditional.
Pornim serverul 2FA si conectam clientul 2FA. In momentul pornirii serverului, acesta va genera continuu coduri 2FA pentru fiecare aplicatie predefinita. Utilizatorul va interactiona initial cu clientul aditional. Clientul se va conecta la serverul aditional iar utilizatorul va putea alege metoda de autentificare dorita : prin notificare in clientul 2FA sau prin introducerea manuala a unui cod. Daca se alege metoda notificarii, serverul 2FA primeste acest request si intreaba clientul 2FA daca se doreste autentificarea. In functie de raspuns, clientul aditional va primi (sau nu) acces. Daca se alege metoda manuala de autentificare, serverul 2FA trimite catre clientul sau ultimul cod generat pentru aplicatia in cauza iar clientul 2FA afiseaza acest cod. In acest moment, utilizatorul poate introduce codul primit. Daca se introduce gresit aceasta parola, utilizatorului i se ofera inca o incercare. Daca si a doua este gresita, clientul se va opri neprimind acces. In cazul in care parola este corecta, clientul va primi acces iar conexiunea se inchide.



4 Detalii de implementare

4.1 Comunicarea server-client este realizata pe baza socket-urilor.

Acestea ne permit o comunicare bidirectionala.

Crearea unui socket TCP

```
if ((_2fa_sd = socket(AF_INET, SOCK_STREAM, 0)) == -1)
{
    perror("Error creating socket for communicating with 2FA server");
    return errno;
}
```

AF_INET specifica familia de protocoale folosita de socket si anume IPv4.
SOCK_STREAM specifica tipul de conexiune si anume una bazata pe fluxuri de octeti secventiale, fiabile si bidirectionale. (Specific protocolului TCP).

4.2 Creearea de thread-uri

Pentru a asigura concurenta serverului 2FA, am folosit implementarea cu threaduri si pentru ca stim numarul de clienti am ales sa aloc de la inceput cele doua fire de executie, folosind mutex-uri pentru protejarea apelului de accept(). [acest mutex_lock se regaseste in functia treat()]

Initializare thread-uri (prethreaded server)

```
pthread_t tid;
pthread_create(&tid, NULL, &gen_2fa_onthread, NULL);

void threadCreate(int);

nthreads = NO_THREADS;
threadsPool = calloc(sizeof(Thread), nthreads);

void threadCreate(int i)
{
    void *treat(void *);

    pthread_create(&threadsPool[i].idThread, NULL, &treat, (void *)i);
    return; // returning from principal thread
}
```

De asemenea, in dezvoltarea aplicatiei, am ales sa creez un thread separat si pentru generarea de coduri de autentificare, pentru ca acest proces sa se realizeze continuu :

Initializare thread pentru generarea 2FA

```
pthread_t tid;
pthread_create(&tid, NULL, &gen_2fa_onthread, NULL);

void* gen_2fa_onthread(void *arg)
{
    while(1)
    {
        ret_2fa_pass(); // Genereaza pentru fiecare aplicatie predefinita
        sleep(TIME_FOR_2FA_GEN); // Asteptam inainte de alta generare
    }
    return 0;
}
```

4.3 Folosirea de mutex-uri

Obiectul *mlock* de tip mutex este blocat de apelul *pthread_mutex_lock*(). Astfel, apelul threadului se va bloca pana cand mutex-ul devine disponibil.
Functia *pthread_mutex_unlock*() elibereaza obiectul referentiat, devenind disponibil.

Folosire mutex pentru protejarea acceptului

```
pthread_mutex_t mlock=PTHREAD_MUTEX_INITIALIZER;
pthread_mutex_lock(&mlock);

/* accepting client */
if ((client = accept(_2fa_sd, (struct sockaddr *)&from, &length)) < 0)
{
    perror("[2fa_thread:]Error accepting.\n");
}

pthread_mutex_unlock(&mlock);
```

4.4 Generare de parole 2FA

In implementarea generatorului de 2FA, se foloseste o functie bazata pe primitivele *rand*() si *srand*(). Aceasta genereaza $\_2FA\_CODE\_LEN$ simboluri si returneaza intreaga parola.

Generarea unui singur cod 2FA

```
char* random_2fa_pass()
{
    char numbers[] = "0123456789";
    char letter[] = "qwertyuiopasdfghjklzxcvbnm";
    char LETTER[] = "QWERTYUIOPASDFGHJKLZXCVBNM";
    srand((unsigned int)(time(NULL))); // seed rand() by time
    char *pass = malloc(_2FA_CODE_LEN);
    int randomizer = rand() % 3; // [0/1/2] for number, letter or LETTER

    for (int i=0; i < _2FA_CODE_LEN; i++)
    {
        switch(randomizer)
        {
        case 0:
            {
                pass[i] = numbers[rand() % NO_DIGITS];
                randomizer = rand() % 3;
            }break;
        case 1:
            {
                pass[i] = letter[rand() % NO_LETTERS];
                randomizer = rand() % 3;
            }break;
        case 2:
            {
                pass[i] = LETTER[rand() % NO_LETTERS];
                randomizer = rand() % 3;
            }break;
        }
    }
    return pass;
}
```

5 Concluzii

Am ales proiectul 'My2FA' deoarece necesita un sistem de comunicare complex intre clientii si serverele implicate si prezinta un element de securitate, ramura de care sunt pasionat.

Imbunatatiri ce pot fi aduse solutiei :

- Un important element al unei aplicatii de securitate ar fi confidentialitatea schimbului de informatii intre client-server. Acest schimb de date ar putea fi securizat cu un sistem de criptare a mesajelor.
- O alta imbunatatire ce poata fi adusa aplicatiei 'My2FA' este o interfata grafica pentru clientul 2FA. (e.g. Butoane pentru acceptul/refuzul autentificarii pe baza de notificare)
- Un alt element ce ar mari utilitatea sistemului ar fi parametrizarea aplicatiilor pentru care serverul 2FA genereaza coduri. Momentan, solutia propusa lucreaza pe doar 4 aplicatii predefinite.

6 Bibliografie

[1] Computer Networks - Facultatea de Informatica Iasi

- https://profs.info.uaic.ro/ computernetworks/files/7rc_ProgramareaInReteaIII_En.pdf
- https://profs.info.uaic.ro/ computernetworks/files/NetEx/S12/ServerPreThread/servTcpPreTh.c

[2] Wikipedia

- https://ro.wikipedia.org/wiki/Transmission_Control_Protocol

[3] Linux Man Pages

- https://man7.org/linux/man-pages
- https://pubs.opengroup.org/onlinepubs/007908799/