Proiect MySSH

Amihaesii Simona
Grupa: 2E3

amihaesii_simona@yahoo.com

Abstract. Acest proiect propune dezvoltarea unui sistem client-server bazat pe utilizarea socket-urilor in C. Programul permite comunicarea encriptata intre un server si mai multi clienti, avand capacitatea de a transmite datele solicitate de acestia.

Keywords: server · client · TCP · baze de date · SQLite · socket · server concurent · fork

1 Introducere

Aplicatia MySSH implementeaza un sistem de comunicare encriptata intre un server si clienti, folosind criptografia asimetrica si protocolul TCP, cu scopul de a returna clientilor output-ul comenzilor cerute, asigurand integritatea si confidentialitatea datelor transmise.

2 Tehnologii Aplicate

1) Aplicatia utilizeaza un model TCP/IP concurent, creand un proces nou pentru fiecare client care se conecteaza la server pentru a putea servi mai multi clienti simultan si pentru a asigura ca datele ajung integral la destinatar.
2) Pentru schimbul encriptat de date, se implementeaza criptografia asimetrica cu algoritmul RSA. Se genereaza chei publice si private la nivel de server si client, comunicarea este securizata prin encriptarea si decriptarea datelor.
3) Pentru gestionarea informatiilor despre utilizatori, programul utilizeaza o baza de date SQLite locala, prin intermediul bibliotecii sqlite3. Aceasta stocheaza informatiile de autentificare si permite verificarea existentei informatiilor utilizatorilor in baza de date prin interogari.
4) In plus, in cazurile in care executia unor comenzi esueaza am folosit biblioteca libexplain/execvp.h pentru a gasi eroarea exacta.

3 Structura Aplicatiei

Aplicatia este structurata intr-un model client-server concurent, unde serverul este capabil sa serveasca mai multi clienti simultan, comunicarea este encriptata, si doar utilizatorii autentificati pot primi output-ul comenzilor cerute.

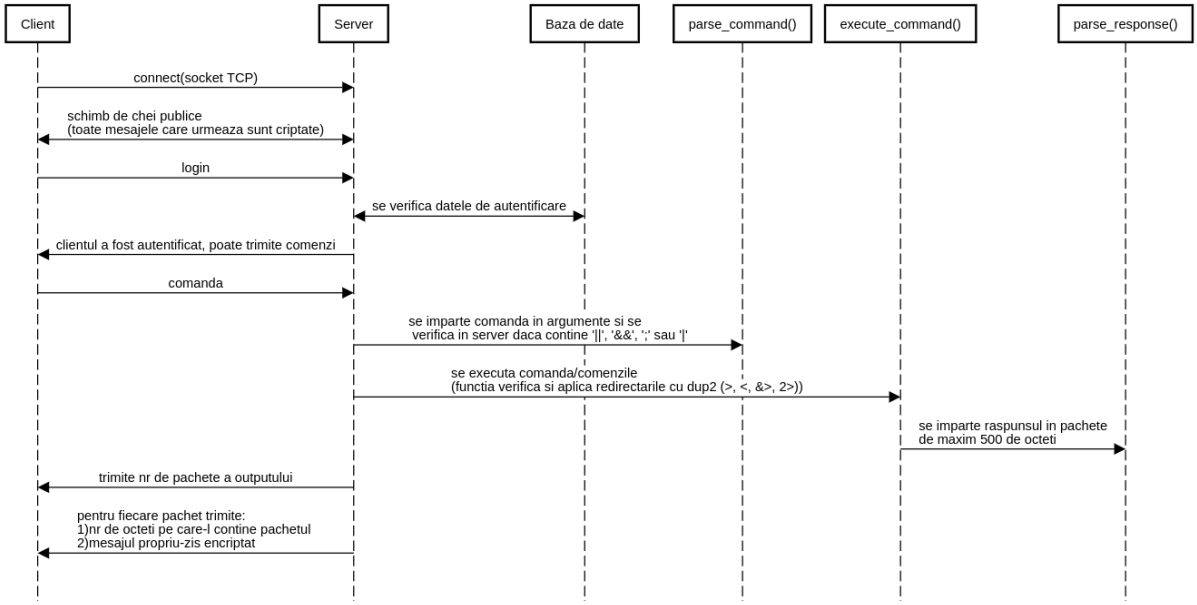

Fig. 1. Diagrama Aplicatiei

4 Aspecte de Implementare

Serverul si clientii isi genereaza propriile chei publice si private, iar dupa ce se conecteaza, fac schimb de cheile publice, cu ajutorul carora isi vor encripta mesajele inainte de a le transmite (ex: un client va encripta mesajul pe care vrea sa il transmita serverului folosind cheia publica a serverului), iar cu cheile private proprii vor decripta mesajele primite.
Functiile `encrypt_text` si `decrypt_text` executa criptarea si decriptarea mesajelor, utilizand valorile ASCII pentru a encoda textul inainte de a-l encripta, respectiv pentru a decoda un text decriptat.
Serverul este cel care gestioneaza comenzile primite de la clienti, inclusiv comenzile login, logout si alte comenzi specifice aplicatiei MySSH. Informatiile de autentificare ale utilizatorilor (numele de utilizator si parola), sunt stocate si verificate in baza de date SQLite. Comenzile SQL sunt folosite pentru a realiza interogari precum verificarea existentei unui utilizator si a corectitudinii parolei asociate.

```
char *username_to_check = nume_utilizator;
char *password_to_check = parola;
bzero(query, 200);
sprintf(query, "SELECT * FROM users WHERE username='%s' AND password='%s'", username_to_check, password_to_check);

if (sqlite3_exec(db, query, callback, &logat, &err_msg) == SQLITE_OK)
    if (logat == 1)
        strcpy(raspuns, "V-ati autentificat cu succes.");
    else
        strcpy(raspuns, "Datele introduse sunt incorecte.");
```

Fig. 2. Exemplu interogare cu SQLite3.

Daca utilizatorul s-a autentificat, poate introduce comenzi de la tastatura catre server. Comenzile sunt impartite in argumente separate de spatii cu ajutorul functiei `parse_command`. Serverul apoi verifica daca comanda introdusa de client contine '||', '|', '&&' sau ';' si daca da, o imparte in 2 comenzi, cea de dinainte de operator si cea de dupa, si se executa in functie de logica operatorului. Comenzile care urmeaza a fi executate sunt trimise catre functia `execute_command`, unde sunt verificate (intr-un proces copil care comunica cu parintele prin pipe) daca contin redirectari (>, <, 2>, &>) si se redirecteaza unde este cazul cu dup2, dupa care se executa cu execvp. Parintele verifica daca s-a executat comanda cu succes sau nu utilizand pipe-ul si biblioteca `libexplain/execvp.h`, dupa care transmite raspunsul final functiei `parse_response`, care-l imparte in pachete de maxim 500 octeti. Serverul este apoi cel care transmite raspunsul final clientului: trimite nr total de pachete si apoi, pt fiecare pachet: 1) trimite nr de octeti pe care-l contine; 2)encripteaza mesajul; 3)transmite mesajul encriptat.

5 Concluzii

Aplicatia poate fi imbunatatita in ceea ce priveste securitatea, cheia publica a serverului este momentan compromisa atacurilor de interceptare.
Alta imbunatatire ar fi ca aplicatia sa suporte mai mult de o singura redirectare (>, <, 2>, &>) si mai mult de o singura operatie logica ('||', '|', '&&', ';').

References

1. `https://www.sqlite.org/docs.html`
2. `https://www.gnu.org/software/bash/manual/html_node/Shell-Commands.html`
3. `https://cplusplus.com/reference/clibrary/`
4. `https://www.geeksforgeeks.org/rsa-algorithm-cryptography/`