MySSH - Raport Tehnic

Trofin Dragoș-Neculai

Facultatea de Informatică, Universitatea "Alexandru Ioan Cuza" din Iași

1 Introducere

Într-o lume digitală tot mai interconectată, necesitatea unui canal sigur și eficient de comunicare între sisteme este esențială. **MySSH** vine ca o soluție inovativă pentru gestionarea sesiunilor de comandă de la distanță, oferind un protocol propriu de securizare și administrare a comenzilor.

MySSH permite utilizatorilor să se conecteze la un server de la distanță, să trimită comenzi și să primească răspunsurile într-un mediu securizat, asigurând integritatea datelor. Proiectul utilizează concepte moderne de rețea, cum ar fi TCP pentru fiabilitate și comunicații bidirecționale, oferind o alternativă personalizată față de soluțiile tradiționale de transfer securizat de comenzi. Printr-o abordare simplificată, MySSH își propune să reducă complexitatea pentru utilizatori, menținând în același timp un nivel ridicat de securitate și performanță. . Obiectivul principal este de a oferi funcționalitățile de bază ale unui serviciu SSH simplificat, prin care clientul trimite comenzi către server și primește rezultatele execuției direct în consola sa.

2 Tehnologii Aplicate

Proiectul utilizează protocolul **TCP/IP** [3] pentru comunicarea între client și server, datorită fiabilității și garanției livrării corecte a datelor. Protocolul TCP este ideal pentru acest tip de aplicații deoarece asigură o comunicare ordonată și fără pierderi de pachete .

Pe server, un socket TCP este inițializat pentru a asculta pe portul 8090, iar clientul se conectează la acest port pentru a trimite comenzi către server. Proiectul utilizează mai multe tehnologii esențiale:

- **Criptarea datelor:** Mesajele între client și server sunt criptate folosind o metodă simplă de XOR, împreună cu o parolă partajată și contori pentru a asigura confidențialitatea. Această metodă este implementată în funcțiile `encrypt_decrypt`.
- **Multiprocesare și thread-uri:** Serverul creează un proces separat pentru fiecare client folosind `fork`[4] și gestionează execuția comenzilor pe server, iar în modul interactiv folosește `forkpty` [7] pentru a crea terminale virtuale.
- **Select pentru multiplexare:** Sistemul utilizează funcția `select` pentru a monitoriza mai multe descriere de fișier (file descriptor) pentru operații de citire/scriere, asigurând o gestionare eficientă a intrărilor și ieșirilor între client și server.

3 Structura Aplicației

Aplicația MySSH este formată din trei componente: serverul și clientul care comunică prin intermediul socket-urilor TCP și fișierul JSON în care se află user-ul împreună cu parola acestuia care vor oferi acces la folosirea comenzilor. Diagrama detaliată a aplicației este prezentată în Figura 1.

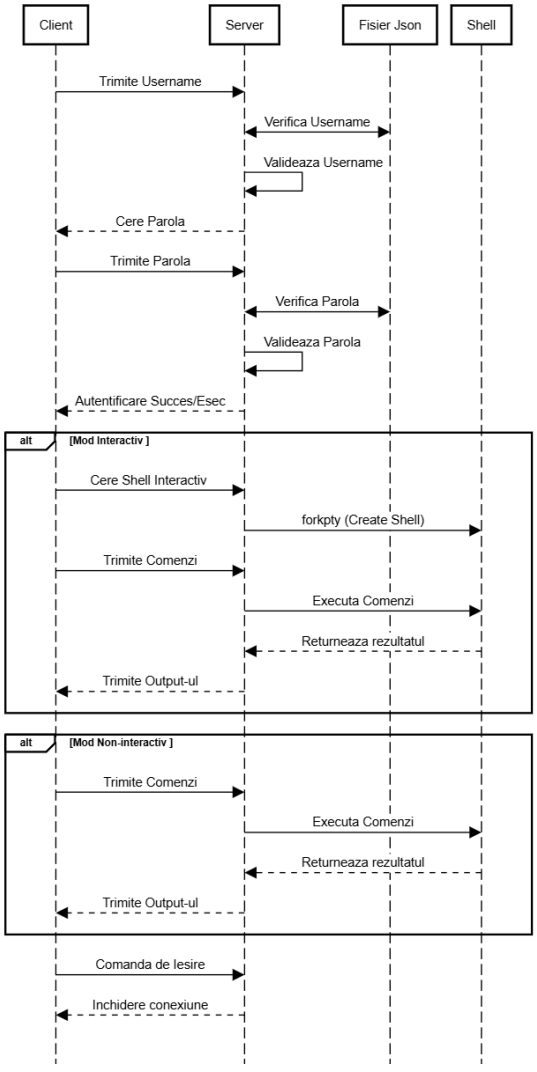

Fig. 1. Diagrama structurii aplicației MySSH

4 Aspecte de Implementare

4.1 Autentificarea utilizatorului

Serverul folosește un fișier `users.json` pentru a gestiona autentificarea utilizatorilor. Funcția `authenticateUser` verifică dacă utilizatorul și parola corespund unui cont valid. Dacă autentificarea eșuează, conexiunea este închisă.

```
bool authenticateUser(const std::vector<User>& users,
                      const std::string& username,
                      const std::string& password)
{
    for (const auto& user : users) {
        if (user.username == username && user.password == password)
            return true;
    }
    return false;
}
```

4.2 Utilizarea TCP

Codul proiectului MySSH folosește protocolul TCP pentru a crea o conexiune fiabilă între client și server. TCP este implementat în cod prin utilizarea următoarelor funcții standard de rețea:

Listing 1.1. Inițializarea socket-ului TCP pe server

```
int server_fd;
sockaddr_in address;

// Crearea socket-ului TCP
if ((server_fd = socket(AF_INET, SOCK_STREAM, 0)) == 0) {
    perror("Socket failed");
    exit(EXIT_FAILURE);
}


address.sin_family = AF_INET;
address.sin_addr.s_addr = INADDR_ANY;
address.sin_port = htons(PORT);

// Legarea socket-ului la port
if (bind(server_fd, (struct sockaddr*)&address, sizeof(address)) < 0) {
    perror("Bind failed");
    exit(EXIT_FAILURE);
}
```

Funcția `socket()` inițializează socket-ul TCP, folosind familia de adrese `AF_INET` pentru IPv4 și `SOCK_STREAM` pentru a indica utilizarea TCP. După aceasta, serverul este legat de o adresă specifică folosind `bind()`, urmată de ascultarea conexiunilor prin `listen()`.

4.3 Executarea comenzilor shell

Comenzile sunt procesate și executate folosind funcția `execvp`[6]. În cazul pipeline-urilor între comenzi (comenzi separate prin '|'), ieșirea unei comenzi este transmisă ca intrare pentru următoarea. Aceasta este realizată prin utilizarea pipe-urilor[10] și funcției `fork`.

```
void CommandShell::executeCommand(const Command& cmd,
                                  int input_fd, int output_fd)
{
    pid_t pid = fork();
    if (pid == 0) {
    .
    .
    .

        dup2(input_fd, STDIN_FILENO);
        .
        .
        dup2(output_fd, STDOUT_FILENO);
        .
        .
        .
        execvp(args[0], args.data());
    }
}
.
.
.
for (size_t i = 0; i < pipeline.commands.size(); ++i)
    {
        int pipe_fds[2] = {-1, -1};

        // Create pipe for all but the last command
        if (i < pipeline.commands.size() - 1)
        {
            if (pipe(pipe_fds) == -1)
            {
                perror("pipe failed");
                return;
            }
        }
```

4.4 Scenarii Reale de Utilizare

1. **Sesiune interactivă**: Clientul trimite comenzi precum `ls`, `pwd`, `cat file.txt`, iar serverul le execută și returnează rezultatele execuției. 2. **Deconectare**: La primirea comenzii `exit`, serverul închide sesiunea de shell și deconectează clientul.

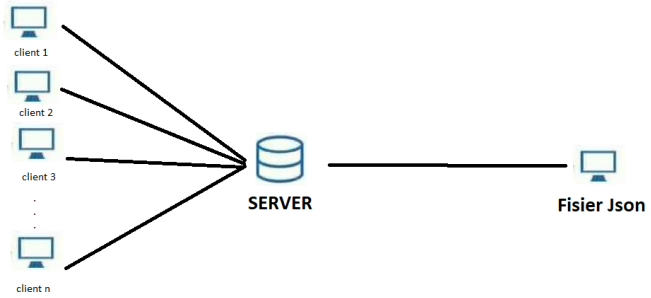

Fig. 2. Diagrama Arhitecturala

5 Concluzii

Proiectul **MySSH** oferă o bază funcțională pentru execuția comenzilor la distanță prin intermediul unei conexiuni TCP. Ca posibile îmbunătățiri, soluția ar putea avea o îmbunătățire a criptării. De asemenea, o interfață grafică plăcută utilizatorului ar putea aduce un plus acestei aplicații.

6 Referințe Bibliografice

1. Online Diagram Tool. `https://sequencediagram.org/`
2. Overleaf. Scriere Raport Tehnic `https://www.overleaf.com/learn/latex/Code_listing`
3. Cursul Retele de Calculatoare saptamana 4 `https://edu.info.uaic.ro/computer-networks/files/5rc_ProgramareaInReteaI_ro.pdf`
4. fork() — Linux manual page `https://man7.org/linux/man-pages/man2/fork.2.html`
5. Linux Documentation dup2 `https://man7.org/linux/man-pages/man2/dup.2.html`
6. How to use execvp `https://stackoverflow.com/questions/27541910/how-to-use-execvp`
7. Linux Documentation forkpty `https://linux.die.net/man/3/forkpty`
8. Linux Documentation fnctl `https://man7.org/linux/man-pages/man2/fcntl.2.html`

9. Cursul Retele de Calculatore saptamana 6 https://edu.info.uaic.ro/computer-networks/files/6rc_ProgramareaInReteaII_Ro.pdf
10. Linux Documentation pipe https://man7.org/linux/man-pages/man2/pipe.2.html
11. Michael Kerrisk (Linux manual pages).listen(). https://man7.org/linux/man-pages/man2/listen.2.html
12. Linux Documentation.socket() https://linux.die.net/man/7/socket