MySSH: Implementarea Securizată a unui Sistem Client-Server

Andrei Popa

Universitatea "Alexandru Ioan Cuza"

Abstract. Proiectul "MySSH" reprezintă o soluție inovatoare în domeniul comunicațiilor securizate client-server. Scopul principal este de a implementa un sistem robust, capabil să gestioneze autentificarea și comunicarea criptată între un client și un server.

1 Introducere

Proiectul "MySSH" răspunde nevoii securității comunicațiilor în rețelele de calculatoare prin dezvoltarea unei soluții client-server care asigură comunicații securizate și autentificate, atât pentru utilizatorii individuali, cât și pentru organizații. Principalul obiectiv al acestui proiect este de a crea un sistem capabil să execute comenzi pe un server de la distanță, menținând în același timp integritatea și confidențialitatea datelor transmise.

Caracteristica cheie a "MySSH" este implementarea sa flexibilă și sigură, care permite utilizatorilor să execute o gamă variată de comenzi **Unix/Linux**, inclusiv comenzi complexe care implică redirecționări, pipe-uri și alte operații specifice shell-ului. Această funcționalitate este deosebit de relevantă în scenarii unde accesul la distanță și gestionarea serverelor sunt esențiale, oferind utilizatorilor un control extins asupra sistemelor remote fără a compromite securitatea.

În plus, "MySSH" se concentrează pe utilizarea tehnologiilor avansate de criptare pentru a asigura că toate comunicațiile între client și server sunt protejate împotriva interceptărilor și accesului neautorizat. Acest aspect este vital în protejarea datelor sensibile și a informațiilor confidențiale care sunt adesea transmise în cadrul acestor sesiuni.

Prin abordarea acestor provocări, "MySSH" nu doar că îmbunătățește securitatea și eficiența comunicațiilor în rețele, dar deschide și noi posibilități pentru administrarea sistemelor și infrastructurilor IT într-un mod mai sigur și mai eficient. Această introducere prezintă fundamentul pe care se construiește restul documentației, detaliind tehnicile, implementarea și aplicabilitatea "MySSH" în contextul securității cibernetice moderne.

2 Tehnologii Utilizate

Proiectul "MySSH" integrează o serie de tehnologii avansate pentru a asigura securitatea, eficiența și fiabilitatea comunicațiilor client-server. Aceste tehnologii au fost alese strategic pentru a răspunde nevoilor specifice de securitate și performanță ale aplicației.

2.1 TCP - Transmission Control Protocol

Am ales protocolul TCP pentru comunicația client-server din cauza fiabilității sale superioare în livrarea datelor. TCP asigură o comunicare securizată și ordonată, garantând că pachetele de date ajung la destinație complet și fără pierderi. Această caracteristică este esențială pentru asigurarea integrității și coerenței comunicațiilor în "MySSH".

2.2 SHA-256 pentru Hashuirea Parolei

Securitatea informațiilor utilizatorilor este de o importanță capitală în "MySSH". Pentru a asigura protecția parolelor utilizatorilor, am folosit algoritmul SHA-256 din libraria OpenSSL, un standard industrial de hashuire. Acesta transformă parolele în hash-uri unice și ireversibile, ceea ce înseamnă că parolele sunt stocate într-o formă securizată și sunt protejate împotriva accesului neautorizat sau a atacurilor brute-force.

2.3 AES pentru Criptarea Comenzilor și a Output-urilor

Pentru a asigura confidențialitatea comunicațiilor între client și server, "MySSH" utilizează Advanced Encryption Standard (AES). AES este un algoritm robust de criptare care protejează atât comenzile trimise de client, cât și răspunsurile serverului. Implementarea AES garantează că toate comunicațiile sunt criptate eficient, prevenind astfel interceptarea și citirea neautorizată a datelor sensibile.

Aceste tehnologii formează coloana vertebrală a "MySSH", asigurând că aplicația nu doar că răspunde cerințelor de funcționalitate și eficiență, dar și celor de securitate, un aspect vital în orice sistem de comunicație modern.

3 Structura Aplicației

Proiectul "MySSH" a fost construit folosind o serie de concepte și paradigme de programare care îmbunătățesc securitatea, performanța și ușurința de utilizare. Structura aplicației este divizată în mai multe componente cheie, fiecare având un rol specific în funcționarea sistemului.

3.1 Concepte implicate

Aplicația "MySSH" utilizează următoarele concepte cheie în design-ul său:

- **Comunicare Client-Server**: Bazată pe modelul client-server, aplicația asigură un schimb de date securizat între un client și un server prin protocolul TCP.
- **Criptare**: Pentru a proteja confidențialitatea datelor, "MySSH" folosește criptarea AES pentru mesaje și SHA-256 pentru hashuirea parolelor.
- **Autentificare**: Autentificarea utilizatorilor se face prin verificarea parolelor hashuite, asigurând securitatea accesului la sistem.
- **Interfața Utilizator**: Interacțiunea utilizatorilor cu sistemul se face printr-o interfață simplă, unde comenzi pot fi introduse și răspunsurile vizualizate.

3.2 Diagrama proiectului

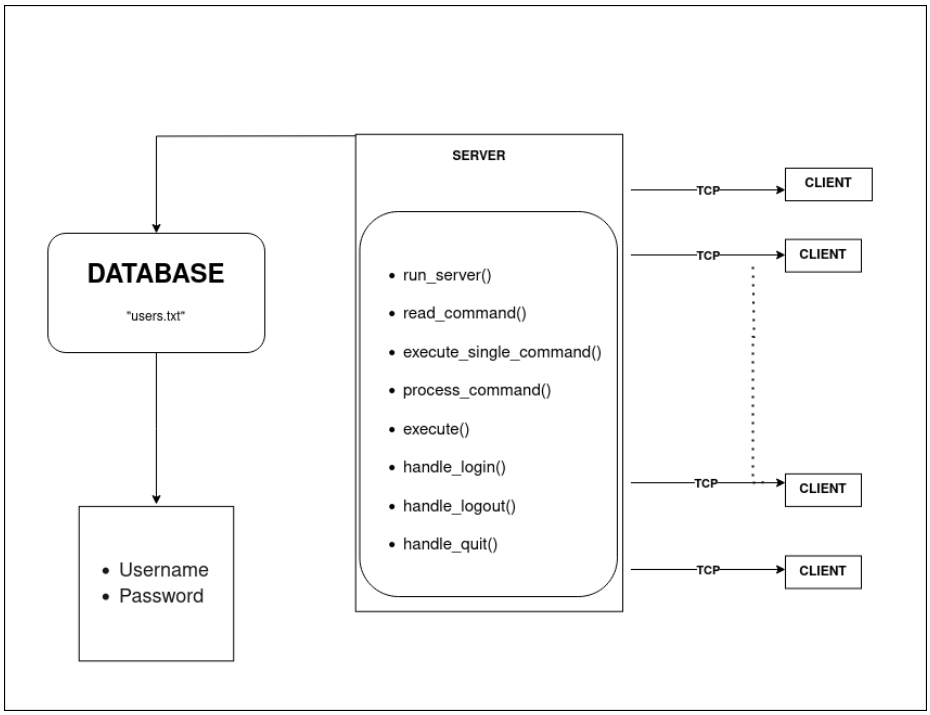

Fig. 1. Diagrama arhitecturii sistemului MySSH

4 Aspecte de Implementare

4.1 Arhitectura Generală

Aplicația "MySSH" adoptă un model client-server, unde clientul inițiază conexiunea și trimite comenzi, iar serverul răspunde la aceste comenzi. Acest model permite o separare clară a responsabilităților și facilitează comunicații securizate.

Clientul este responsabil pentru autentificarea utilizatorului, introducerea comenzilor și primirea răspunsurilor. Este implementat astfel încât să ofere securitate în transmiterea datelor și să asigure integritatea și confidențialitatea informațiilor transmise.

Serverul gestionează cererile venite de la clienți, execută comenzile și returnează rezultatele. Este echipat cu mecanisme de criptare pentru a proteja datele transmise și pentru a asigura securitatea în cadrul sesiunilor de comunicare.

4.2 Structura Codului

Codul pentru "MySSH" este divizat în două componente principale: clientul și serverul. Ambele sunt implementate în limbajul C/C++, profitând de capabilitățile extinse ale acestor limbaje în ceea ce privește gestionarea rețelelor și a proceselor.

Clientul este conceput pentru a iniția conexiuni securizate, a trimite comenzi și a primi rezultatele. Interfața sa simplă și directă permite utilizatorilor să interacționeze eficient cu serverul.

Serverul, pe de altă parte, ascultă și gestionează cererile venite de la client. Implementarea sa include un mecanism robust pentru executarea comenzilor și returnarea output-ului către client, tratând în mod eficient redirecționările și pipe-urile.

4.3 Exemplu de Implementare a Clientului

Următorul fragment de cod ilustrează implementarea clientului în "MySSH":

```
int main (int argc, char *argv[])
{
  /* initializarea socketului si a structurii serverului*/

  /*initializarea encriptiei*/
  init_crypto(&encrypt_ctx, &decrypt_ctx, key, iv);

  /* exista toate argumentele in linia de comanda? */
  if (argc != 3)
    {
      printf ("[client] Sintaxa: %s <adresa_server> <port>\n", argv[0]);
      return -1;
    }

  /* stabilim portul */
  port = atoi (argv[2]);

  while(1){

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

  server.sin_family = AF_INET;
  server.sin_addr.s_addr = inet_addr(argv[1]);
```

```
server.sin_port = htons (port);

if (connect(sd, (struct sockaddr *) &server, sizeof(struct sockaddr)) == -1) {
          perror("[client]Eroare la connect().\n");
          close(sd);
          continue;
      }


  /*prelucrearea credentialelor si trimiterea lor*/
  SHA256Hash(password, hashedPassword);

  /*verificarea logarii clientului*/

  while(1){
      /*encriptarea comenzii si trimiterea lui*/
      if (!EVP_EncryptUpdate(encrypt_ctx, encrypted_msg, &encrypted_len, msg, strlen((cha
          perror("EncryptUpdate error");
          return 1; // or handle the error accordingly
      }

      if (!EVP_EncryptFinal_ex(encrypt_ctx, encrypted_msg + encrypted_len, &tmplen)) {
          perror("EncryptFinal error");
          return 1; // or handle the error accordingly
      }
      encrypted_len += tmplen;

      /*citirea outputului si printarea lui*/

      // Decrypt the command
      if (!EVP_DecryptUpdate(decrypt_ctx, decrypted_buffer, &decrypted_len, msg, bytes)) {
          perror("DecryptUpdate error");
          return 0;
      }

      int final_len = 0;
      if (!EVP_DecryptFinal_ex(decrypt_ctx, decrypted_buffer + decrypted_len, &final_len)
          perror("DecryptFinal error");
          return 0;
      }
      decrypted_len += final_len;
      decrypted_buffer[decrypted_len] = '\0';

  }
```

```
    close(sd);
  }

  return 0;
}
```

Această secțiune de cod demonstrează cum clientul "MySSH" inițiază o conexiune TCP cu serverul și utilizează criptarea AES pentru a securiza comunicațiile.

4.4 Exemplu de Implementare a Serverului

În mod similar, serverul "MySSH" este responsabil pentru ascultarea cererilor de la clienți, procesarea acestora și răspunsul corespunzător. Serverul gestionează autentificarea utilizatorilor și execuția comenzilor primite.

```
int run_server()
{

    // Crearea socket-ului
    if ((sd = socket(AF_INET, SOCK_STREAM, 0)) == -1)
    {
        perror("[server] Eroare la socket().\n");
        return errno;
    }

    // Setarea opțiunii SO_REUSEADDR
    setsockopt(sd, SOL_SOCKET, SO_REUSEADDR, &optval, sizeof(optval));

    // Pregătirea structurilor de date
    bzero(&server, sizeof(server));

    // Completarea structurii folosite de server
    server.sin_family = AF_INET;
    server.sin_addr.s_addr = htonl(INADDR_ANY);
    server.sin_port = htons(PORT);

    // Legarea socket-ului
    if (bind(sd, (struct sockaddr *)&server, sizeof(struct sockaddr)) == -1)
    {
        perror("[server] Eroare la bind().\n");
        return errno;
    }

    // Serverul începe să asculte dacă clienții se conectează
    if (listen(sd, 5) == -1)
    {
```

```
    perror("[server] Eroare la listen().\n");
    return errno;
}
while (1)
{

    bcopy((char *)&actfds, (char *)&readfds, sizeof(readfds));

    // Apelul select()
    if (select(nfds + 1, &readfds, NULL, NULL, &tv) < 0) {
        perror("[server] Eroare la select().\n");
        return errno;
    }

    // Verificarea dacă socket-ul server este pregătit să accepte clienți
    if (FD_ISSET(sd, &readfds)) {
        len = sizeof(from);
        bzero (&from, sizeof (from));
        client = accept(sd, (struct sockaddr *)&from, &len);
        if (client < 0) {
            perror("[server] Eroare la accept().\n");
            continue;
        }

        printf("[server] S-a conectat clientul cu descriptorul %d, de la adresa %s.\n",

        FD_SET(client, &actfds); // Adăugăm noul client în set
        if (client > nfds)
            nfds = client;
    }

    // Verificarea pentru un socket client activ
    for (fd = 0; fd <= nfds; fd++)
    {
        // Verificarea pentru un socket pregătit de citire
        if (fd != sd && FD_ISSET(fd, &readfds))
        {
            // // Autentificarea și procesarea comenzilor

            if(findUserBySocket(fd)){
                /* daca userul e logat citeste comanda si o executa*/
                /*trimite outputul la client*/
                }
            } else {
                /*conectarea clientului*/
```

```
                    /*daca e in baza de date, vom trimite clientului mesajul "Autentificare
                    /* daca nu, trimitem "Autentificare esuata"*/
                }
            }
        }
    }
}
```

Această secțiune de cod oferă o viziune asupra modului în care serverul gestionează conexiunile și comunică cu clienții.

4.5 Comunicarea și Protocolul

Comunicarea între client și server se bazează pe protocolul TCP, ales pentru fiabilitatea sa și capacitatea de a gestiona transmisii de date ordonate și orientate spre conexiune. Protocolul asigură că fiecare comandă și răspuns sunt transmise integral și în ordinea corectă.

Securitatea este un aspect crucial al "MySSH". Implementarea include mecanisme de criptare pentru protejarea datelor transmise între client și server, asigurând confidențialitatea și prevenind accesul neautorizat.

4.6 Execuția Comenzilor și Scenarii de Utilizare

Serverul "MySSH" este capabil să execute o varietate largă de comenzi Unix/Linux. Acest lucru este realizat prin interpretarea și procesarea comenzilor primite de la client, inclusiv suportul pentru operațiuni complexe precum pipe-uri și redirecționări.

De exemplu, pentru a lista fișierele și a găsi un fișier specific, un utilizator ar putea executa următoarea comandă:

```
ls -l | grep myfile
```

Într-un alt scenariu, pentru a redirecționa conținutul unui log într-un fișier de backup, comanda ar fi:

```
cat /var/log/syslog > syslog_backup.txt
```

Scenariile de utilizare variază de la administrarea simplă a serverului la executarea de sarcini complexe și automatizate.

4.7 Inovații și Diferențiatori

Proiectul "MySSH" introduce mai multe elemente inovatoare care îl diferențiază de alte soluții existente în domeniul comunicațiilor securizate client-server. În centrul acestor inovații stă un accent puternic pe securitate, ușurința în utilizare și adaptabilitate.

- **Criptare Avansată**: "MySSH" implementează un sistem de criptare avansat folosind algoritmul AES, configurat pentru a opera în modul CBC cu un vector de inițializare specific. Acest lucru oferă o protecție robustă a datelor transmise, depășind nivelul standard de securitate oferit de protocoalele tradiționale SSH.
- **Autentificare Securizată**: Prin utilizarea algoritmului de hash SHA-256 pentru stocarea și verificarea parolelor, "MySSH" asigură că informațiile de autentificare ale utilizatorilor sunt protejate împotriva atacurilor brute-force și a compromiterii.
- **Gestionarea Eficientă a Conexiunilor**: Utilizarea funcției 'select()' permite serverului "MySSH" să gestioneze multiple conexiuni client într-o manieră eficientă și scalabilă, fără a recurge la multithreading, ceea ce simplifică arhitectura și reduce consumul de resurse.
- **Interoperabilitate și Flexibilitate**: "MySSH" a fost proiectat pentru a fi flexibil și ușor de integrat cu alte sisteme. Prin designul modular și protocolul de comunicație bine definit, "MySSH" poate fi adaptat pentru a se potrivi diferitelor medii de operare și cerințe specifice.

Aceste caracteristici inovatoare nu doar că îmbunătățesc securitatea și eficiența sistemului "MySSH", dar de asemenea îi conferă o flexibilitate și o scalabilitate care îl poziționează ca o soluție remarcabilă în peisajul tehnologiilor de comunicație securizate.

5 Concluzii

Proiectul "MySSH" a reprezentat o inițiativă ambițioasă de a îmbunătăți securitatea și eficiența în comunicațiile client-server. Prin abordarea inovatoare și implementarea riguroasă, "MySSH" a demonstrat cum tehnologiile avansate pot fi folosite pentru a asigura comunicații securizate și eficiente între clienți și servere.

5.1 Realizări și Impact

Principala realizare a proiectului "MySSH" este dezvoltarea unui sistem robust și flexibil, capabil să gestioneze o gamă largă de comenzi și operații într-un mediu securizat. Implementarea suportului pentru redirecționări complexe și pipe-uri, combinată cu o gestionare eficientă a sesiunilor și a securității, plasează "MySSH" în fruntea soluțiilor moderne de comunicare securizată.

Impactul "MySSH" se extinde dincolo de asigurarea securității, oferind utilizatorilor un nivel ridicat de control și flexibilitate în administrarea serverelor la distanță. Aceasta facilitează un management mai eficient al infrastructurii IT, îmbunătățind productivitatea și reducând riscurile asociate cu accesul remote.

5.2 Direcții pentru Dezvoltări Viitoare

În ciuda eficienței său, există întotdeauna loc pentru îmbunătățire și inovație. Direcții viitoare de dezvoltare pentru "MySSH" ar putea include:

- **Extinderea Platformei**: Adăugarea suportului pentru mai mulți clienți, îmbunătățind astfel accesibilitatea și flexibilitatea soluției.
- **Optimizarea Performanței**: Continuarea muncii asupra optimizării performanței și a eficienței, în special în medii cu resurse limitate sau rețele cu latență mare.
- **Funcționalități Avansate**: Integrarea unor funcționalități avansate, gestionand o gamă mai largă de comenzi(ex. cd, pwd).
- **Securitate Îmbunătățită**: Focalizarea pe dezvoltarea continuă a securității, inclusiv implementarea de noi algoritmi de criptare și protocoale de autentificare.

5.3 Reflecții Finale

"MySSH" reprezintă un pas important în evoluția comunicațiilor securizate client-server. Implementarea sa demonstrează cum inovația tehnologică poate fi utilizată pentru a aborda provocările securității cibernetice, oferind în același timp un nivel superior de funcționalitate și ușurință în utilizare. Așteptăm cu interes să vedem cum "MySSH" va continua să evolueze și să își găsească locul în peisajul tehnologic în schimbare.

6 Referințe Bibliografice

References

1. Server TCP Concurrent Select, `https://profs.info.uaic.ro/~computernetworks/files/NetEx/S9/servTcpCSel.c`
2. Client TCP, `https://profs.info.uaic.ro/~computernetworks/files/NetEx/S9/cliTcp.c`
3. OpenSSL Documentation, `https://www.openssl.org/docs/man1.1.1/man1/`
4. SHA-256 Algorithm Tutorial, `https://www.simplilearn.com/tutorials/cyber-security-tutorial/sha-256-algorithm`