

Proiect Retele de Calculatoare

MySSH

Andrei-Ioan Ianău^[0000-0002-0826-1809]

Universitatea "Alexandru Ioan Cuza", Iași, Iași, România
secretariat@info.uaic.ro
<http://www.info.uaic.ro>

Abstract. Acest document descrie functionalitatea a unei aplicatii de tip "clienti-server" ce deservește ca scop executarea de catre server a unor comenzi primite de la un client. Comunicarea va fi efectuata prin diferite metode cum ar fi sockets, pipes si fisiere fifo.

Keywords: Retele · Sockets · Client-Server.

1 Tehnologiile utilizate

Pentru aplicatia pe care dorim sa o realizam, vom folosi protocolul TCP. Acesta este cel mai de dorit deoarece ne permite siguranta transmiterii datelor la client si la server. Mai ales daca vom folosi encriptare, este de ajutor daca toate datele sunt transmise, astfel decriptarea sa se poata realiza.

Ce este TCP/IP? Spus în termeni simpli, TCP/IP este numele unei familii de protocoale de rețea. Protocoalele sunt seturi de reguli pe care toate companiile si toate produsele software trebuie să le respecte, pentru ca produsele lor să fie compatibile între ele. Un protocol definește felul cum programele comunică între ele. Un protocol de asemenea definește felul cum fiecare parte a pachetului are grijă de transferal de informație. În esență, un protocol este un set scris de directive care definește felul în care două aplicații sau mașini pot comunica între ele, fiecare conformându-se cu aceleași standarde. TCP/IP nu este restricționat doar la Internet. Este protocolul de rețea cel mai larg folosit în lume, folosit pentru rețele mari, cât si pentru rețele mici. TCP/IP vine de la Protocolul de Control al Transmisiei/Internet Protocol, care sunt de fapt două protocoale separate. Contrar a ce gândesc unii oameni, termenul TCP/IP se referă la o întreagă familie de protocoale înrudite, toate proiectate pentru a transfera informații prin intermediul rețelei. TCP/IP este proiectat pentru a fi componenta software a unei rețele. Toate părțile protocolului TCP/IP au anumite sarcini, cum ar fi trimiterea de scrisori electronice, transferal de fișiere, livrarea de servicii de logare la distanță, dirijarea de mesaje, sau manipularea căderilor de rețea. Serviciile care intră în protocolul TCP/IP și funcțiile lor pot fi grupate după scopul lor. Protocoalele de transport controlează mișcarea datelor între 2 mașini si include următoarele: - TCP (Transmission Control Protocol) Un serviciu bazat pe conexiuni, însemnând că mașinile care trimit și cele care primesc sunt conectate și

comunică una cu cealaltă tot timpul. - UDP (User Datagram Protocol) Un serviciu fără conexiuni, însemnând că datele sunt trimise fără ca mașinile care trimit și care primesc să aibă contact unele cu celelalte. Este ca și cum am trimite o scrisoare prin poșta normală, la o anumită adresă, neavând cum să știm dacă scrisoarea ajunge sau nu la acea adresă.

TCP/IP, Internet-ul și arhitectura stratificată Internet-ul nu este o singură rețea, ci mai degrabă o colecție de multe rețele care comunică între ele prin protocolul TCP/IP. TCP/IP și Internetul sunt așa de strâns legate încât arhitectura TCP/IP este adesea denumită și arhitectură Internet. Aproape de la începuturile internetului ca ARPAnet, a devenit evident că protocoalele existente nu puteau să se descurce cu volumul foarte mare al traficului pe care rețeaua trebuia să-l suporte, astfel un proiect a fost inițiat pentru a dezvolta noi protocoale. Protocoalele TCP/IP au fost propuse în anul 1973 și au condus la o versiune standardizată, apărută în 1982. Una dintre paginile de cercetare pentru software-ul de rețea a fost la universitatea din Berkeley, California. Aceasta a fost centrul de dezvoltare pentru sistemul de operare UNIX de-a lungul mai multor ani; cercetarea făcută acolo a ajutat la rafinarea protocolului TCP/IP. În 1983, universitatea a emis o versiune a UNIXului care avea incorporate protocolul TCP/IP. Protocolul a devenit foarte popular deoarece UNIX a era folosit la scară largă, mai că din ce în ce mai multe site-uri se conectau la ARPAnet. Când TCP/IP a fost proiectat, toate serviciile care trebuiau livrate au fost luate în considerare. Cea mai bună abordare pentru a implementa toate serviciile a fost cea de a divide serviciile diferite în categorii, cum ar fi serviciile end-user (transfer de fișiere și logare la distanță), serviciile de transport (modul în care datele sunt trimise în mod invizibil utilizatorului) și serviciile de rețea (modul în care informația este împachetată în vederea transferului). A fost dezvoltată o arhitectura stratificată, care izolează fiecare set de servicii. O abordare stratificată în proiectarea software-ului cere mai multă muncă inițial, dar are mai multe beneficii importante. Mai întâi, deoarece fiecare strat este independent de celelalte, schimbările la un anumit serviciu nu provoacă probleme cu celelalte servicii. Pe măsură ce noi servicii sunt dezvoltate, acestea pot fi adăugate fără a schimba celelalte părți ale sistemului software. Cel mai important, stratificarea face posibil ca un set de programe mici și eficiente să fie dezvoltate pentru scopuri specifice, fiecare fiind independent de celelalte. O condiție necesară pentru a permite ca arhitectura stratificată să funcționeze corespunzător este că fiecare strat trebuie să știe ceea ce vine de la stratul de deasupra sau de dedesubt. Poate ca stratul să nu fie interesat de conținutul mesajului, dar trebuie să știe ce să facă cu el. De exemplu, dacă trimiteți un e-mail, scrieți mesaje și comandați stratul aplicației să transmită mesajul către destinație. Fiecare strat se ocupă de mesajul e-mail, dar nu-l interesează conținutul mesajului. Pentru a simplifica această sarcină, fiecare strat adaugă un bloc de date în fața și în spatele mesajului, ceea ce indică stratul care este implicat, precum și un set de biți reprezentând informațiile adăugate de alte straturi, informații de care mașinile care primesc mesajul cum trebuie. Informația din mesaj este ignorată. Fiecare strat face propria "încapsulare", în

sensul că fiecare strat adaugă o capsulă de informații în jurul informației inițiale, constituind blocuri de început și sfârșit. Aceasta se materializează în câteva seturi de header-e și trailer-e până când mesajul ajunge în rețea. [1]

2 Arhitectura aplicatiei

2.1 Organizarea fisierelor

Vom avea urmatoarele fisiere:

client.c – va contine codul ce reprezinta implementarea aplicatiei din partea clientului.

server.c – va contine codul ce reprezinta implementarea aplicatiei din partea serverului.

README – un fisier ce contine scurte instructiuni pentru utilizarea aplicatiei.

primes.pr – fisier ce contine un numar mare de numere prime, ce vor fi folosite in crearea cheilor de criptare

users.txt – fisier ce contine utilizatorii si parolele lor

run.sh – fisier de foloseste la testarea aplicatiei

Makefile – fisier de foloseste la dezvoltarea aplicatiei

constants.h – contine defines ce sunt folosite in intreaga aplicatie.

macros.h – contine macros ce sunt folosite in intreaga aplicatie

rsa.c – contine codul pentru criptarea de tip rsa.

rsa.h – contine declarile functiilor ce ajuta la criptare.

sha256.c – contine codul functiilor cu care se poate face hash.

sha256.h – contine declararile functiilor cu care se poate face hash.

utilities.c – contine codul functiilor ce se reutilizeaza atat la client cat si la server.

utilities.h – contine declararea functiilor ce se reutilizeaza atat la client cat si la server.

pentru semnătura electronică. Algoritmul a fost dezvoltat în 1977 și publicat în 1978 de Ron Rivest, Adi Shamir și Leonard Adleman la MIT și își trage numele de la inițialele numelor celor trei autori. [3]

Puterea sa criptografică se bazează pe dificultatea problemei factorizării numerelor întregi, problemă la care se reduce criptanaliza RSA și pentru care toți algoritmi de rezolvare cunoscuți au complexitate exponențială. Există însă câteva metode de criptanaliză care ocolesc factorizarea efectivă, exploatând maniere eronate de implementare efectivă a schemei de criptare.

3.2 Extinderea usoara a aplicatiei

Un lucru care permite modularizarea si extinderea acestei aplicatii este modul in care este realizat serverul.

```
typedef int (*Command)(int *auth, char *message, char args[MAX_ARGS][BUFFER_SIZE]);

enum COMMANDS
{
    HELP_COMMAND,
    LOGNEW_COMMAND,
    LOGIN_COMMAND,
    LOGOUT_COMMAND,
    MSTAT_COMMAND,
    MYFIND_COMMAND,
    CWD_COMMAND,
    CD_COMMAND,
    QUIT_COMMAND,
    EXEC_COMMAND,
    UNKOWN_COMMAND,
    COMMANDS_NUMBER
};

// @return enum of type of the command.
int Command_type(const char *command)
{ ...
}
```

Fig. 2. Modularizare.

Folosind astfel o modalitate de extindere a functionalitatii, vom putea apela `command[command_type](args);`

3.3 Inserarea unui utilizator nou

```
int Insert_new_user(char *username, char *password)
{
    FILE *fp;

    fp = fopen("users.txt", "a");
    CHECK(fp != NULL, 1, "Unable to open users.txt");

    char line[BUFFER_SIZE];
    char password_hashed[BUFFER_SIZE];

    calculate_sha256(password_hashed, password, strlen(password));

    sprintf(line, "%s %s\n", username, password_hashed);
    fwrite(line, sizeof(char), strlen(line), fp);
    fclose(fp);
    return 0;
}
```

Fig. 3. Inserare utilizator nou.

3.4 Pastrarea credentialelor

Credentialele se pastreaza intr-un fisier sub forma *username**sha256**custom(password)*. Pentru a calcula sha256 pentru parola, am folosit o librerie si modificat-o astfel incat sa imi functioneze. [4]

References

1. Tutorial tradus de Ionuț Lucașl , 4–6
https://profs.info.uaic.ro/busaco/teach/courses/net/docs/tcpip_net-ro.pdf
2. Wikipedia page pentru RSA , 4–6
<https://ro.wikipedia.org/wiki/RSA>
3. Wikipedia page pentru RSA , 4–6
<https://ro.wikipedia.org/wiki/RSA>
4. Sha256 library
<https://github.com/amosnier/sha-2>