|  |  |
| --- | --- |
| sigla_usv_nou | Universitatea „Ştefan cel Mare” - Suceava  Facultatea de Inginerie Electrică şi Ştiinţa Calculatoarelor |

Proiect la Rețele de Calculatoare din cadrul cursurilor de Licenţă

Inginerie Electronică şi Telecomunicaţii

Conducător ştiintific: drd. ing. Cătălin Beguni

**PROIECT**

**Simularea funcţionării unor servicii de reţea:**

**Tema: Firewall**

Student:Țurcanu Vlad

- SUCEAVA 2024 -

**CUPRINS**

[1. INTRODUCERE](#_Toc528314028) 4

[1.1. Scopul](#_Toc528314029) 4

[1.2. Obiective](#_Toc528314031) 4

[1.3. Teoria Esențială](#_Toc528314031) 4

[2. CREAREA PROGRAMULUI](#_Toc528314032) 5

[2.1. Pașii inițiali](#_Toc528314033) 5

[2.2. Serverul](#_Toc528314034) 5

2.2.1 Rolul și structura Firewall.cpp și Firewall.h…………………………………...6

2.2.2. Rolul și structura Main\_Func.cpp și Main\_Func.h…………………………….6

2.2.3. Funcția și structura Server.cpp........................................................................7-9

2.3. Clientul.........................................................................................................................9

2.3.1. Rolul FUNCS.cpp și FUNCS.h...........................................................................9

2.3.2. Funcția și structura Client.cpp......................................................................10-11

[3. CONCLUZII](#_Toc528314037) 12

[4. BIBLIOGRAFII](#_Toc528314037) 13

**FIGURI ȘI TABELE**

[*Fig. 1.*](#_Toc528314045) *7*

[*Fig. 2.*](#_Toc528314046) *7*

[*Fig. 3.*](#_Toc528314047) *8*

[*Fig. 4.*](#_Toc528314048) *8*

[*Fig. 5.*](#_Toc528314049) *8*

[*Fig. 6.*](#_Toc528314049) *8*

[*Fig. 7.*](#_Toc528314049) *9*

[*Fig. 8.*](#_Toc528314049) *9*

[*Fig. 9.*](#_Toc528314049) *9*

[*Fig. 10.*](#_Toc528314049) *10*

[*Fig. 11.*](#_Toc528314049) *11*

[*Fig. 12.*](#_Toc528314049) *11*

[*Fig. 13.*](#_Toc528314049) *11*

*Tabelul.1....................................................................................................................................5*

*Tabelul.2....................................................................................................................................6Tabelul.3...................................................................................................................................10*

**1.INTRODUCERE:**

* 1. **Scopul**

Proiectul constă în realizarea unui program server/client TCP cu implementarea și simularea unui instrument Firewall ce va securiza rețeaua internă.

* 1. **Obiective**
* Alegerea limbajului și a unui mediu de programare;
* Implementarea unui server/client TCP folosind o librărie specială;
* Realizarea unui Firewall pentru programul scris, testat cu ajutorul unui client local;
* Testarea funcționării corecte a programului;
* Opțional : Efectuarea transmiterii datelor prin rețea;
  1. **Teoria Esențială**

Un **Firewall** este un sistem de securitate de rețea care monitorizează și controlează traficul de intrare și ieșire pe baza unor reguli de securitate prestabilite. Un firewall poate fi hardware, software, sau o combinație a ambelor. Rolul său principal este de a crea o barieră între rețeaua internă de încredere și surse externe neîncrezute, cum ar fi internetul, pentru a preveni accesul neautorizat și atacurile cibernetice.

Momentele importante ale unui Firewall:

* ***Filtrarea Traficului****:* Firewall-urile filtrează traficul pentru a permite sau a bloca datele pe baza unor reguli. Aceste reguli pot fi bazate pe adrese IP, porturi, protocoale, conținut și alte criterii.
* ***Protecția Împotriva Atacurilor****:* Firewall-urile pot proteja împotriva diferitelor tipuri de atacuri, cum ar fi atacurile de tip Denial-of-Service (DoS),DDoS(„Distributed Denial of Service”) atacurile bazate pe scripturi și alte vulnerabilități de securitate.

 **Tipuri de Firewall-uri**:

**1) *Firewall-uri bazate pe pachete****:* Acestea inspectează pachetele individuale de date și decid dacă să le permită să treacă pe baza surselor, destinațiilor și tipurilor de port.

**2) *Firewall-uri de stat****:* Acestea țin evidența stării sesiunilor de comunicație și pot lua decizii mai complexe bazate pe contextul traficului.

**3) *Firewall-uri de aplicație (Proxy Firewall-uri)****:* Acestea filtrează traficul la nivel de aplicație, examinând conținutul datelor pentru a identifica și a bloca traficul nedorit.

* ***Zone Demilitarizate (DMZ)****:* Unele firewall-uri permit configurarea unei zone DMZ, unde serverele accesibile publicului (cum ar fi serverele web și de email) sunt separate de rețeaua internă, oferind un nivel suplimentar de securitate.
* **Reguli și Politici**: Administratorii de rețea configurează firewall-urile cu reguli și politici specifice pentru a controla accesul la resursele de rețea și pentru a proteja împotriva traficului nedorit sau periculos.
* **Audit și Log-uri**: Firewall-urile înregistrează activitatea rețelei, permițând administratorilor să monitorizeze și să analizeze evenimentele suspecte sau încercările de intruziune.

1. **CREAREA PROGRAMULUI ȘI EXPLICAREA ROLULUI FIECĂREI ETAPE**
   1. **Pașii inițiali**

Firewall-ul implementat verifică statutul clientului prin controlarea IP-ului,portului sau IP+port;

Proiectul trebuie sa fie împărțit în 2 programe separete.Unul va răspunde pentru executarea serverului,Firewall-ului, si altul pentru client ca scop de simulare;

Pentru o programare mai eficientă, împărțim codul în mai multe fișiere, .cpp, .h.

Structura serverului va arăta în acest fel :

1. **Server.cpp** [*File principal,în care sunt incluse toate celelalte file-uri*]
2. **Main\_Func.cpp și Main\_Func.h** [*Include toate funcțiile de bază pentru lucrul cu biblioteca Winsock2 și una care folosește funcțiile din Firewall.cpp* ]
3. Firewall.cpp și Firewall.h [*Include toate funcțiile implementate pentru securizarea rețelei*]
4. **BLOCKED\_ip.txt, Blocked\_IP\_Ports.txt, BLOCKED\_PORTS.txt** [*File-uri txt care conțin regulile pentru filtrarea clienților potențiali* ]

Structura clientului:

1. **Client.cpp** [ *File care efectuiază conectarea la server și în care sunt incluse toate funcțiile de baza* ]
2. **FUNCS.cpp și FUNCS.h** [ *Conține funcțiile ce operează cu biblioteca Winsock2*]
   1. **Serverul**

**2.2.1. Rolul și structura Firewall.cpp și Firewall.h**

În file-ul Firewall.h am creat un class *Firewall* ce va conține toate declarațiile funcțiilor, ca urmare a fi implementate in file-ul Firewall.cpp

Funcțiile de bază a acestor file-uri sunt :

TABELUL.1

|  |  |
| --- | --- |
| Numele Funcției | Rolul Funcției |
| AddBlocked\_IP\_Port() | Adaugă IP-ul si portul dorit în file-ul *Blocked\_IP\_Ports.txt*,ce ulterior va fi controlat pentru validare |
| AddBlockedIP() | Adaugă IP-ul dorit în file-ul *BLOCKED\_ip.txt,*ce ulterior va fi controlat pentru validare |
| AddBlocked\_Port() | Adaugă portul dorit în file-ul *BLOCKED\_PORTS.txt,*ce ulterior va fi controlat pentru validare |
| Deleting\_IP() | Șterge din fișierul *BLOCKED\_ip.txt* toate IP -urile,sau un ip anume |
| Deleting\_Port() | Șterge din fișierul *BLOCKED\_PORTS.txt,*toate porturile,sau un port anume |
| Deleting\_Ip\_Port() | Șterge din fișierul *Blocked\_IP\_Ports.txt* toate IP-urile și porturile,sau un IP/port anume |
| IsBlocked\_Ip\_Port(const char\* \_IP, const char\* \_PORT) | Se folosește la verificarea ip-ului/portului clientului prezente în fișierul *Blocked\_IPPorts.txt* |
| IsBlockedIP(const char\* \_IP) | Se folosește la verificarea ip-ului clientului prezente în fișierul *BLOCKED\_ip.txt* |
| IsBlockedPort(const char\* \_PORT) | Se folosește la verificarea ip-ului clientului prezente în fișierul *BLOCKED\_PORTS.txt* |

**2.2.2. Rolul și structura Main\_Func.cpp și Main\_Func.h**

Main\_Func.h are ca scop includerea tuturor bibliotecilor necesare a lucrului funcțiilor declarate.Aceste funcții sunt ulterior implementate in Main\_Func.cpp

Rolul funcțiilor implementate:

TABELUL.2

|  |  |
| --- | --- |
| Numele Funcției | Rolul Funcției |
| Create\_Socket() | Crează socket-ul pentru server,specificând familia adresei,tipul socket-ului, si tipul protocolului.Dacă nu se poate crea,returneaza eroare |
| Bind\_Socket(int port, SOCKET serverSocket) | Specifica IP -ul,portul și protocoulul serviciului dorit |
| Listening(SOCKET serverSocket); | Etapa de așteptare pentru conectarea clientului,în caz ca nu apar erori |
| Accept\_Area(SOCKET serverSocket, int Conns\_nr) | Etapa de acceptare a conectărilor din exterior |
| Blocking\_Connection(char\* clientIP, SOCKET clientSocket) | În cazul conexiunii blocate de regulile firewall-ului,transmite mesajul respectiv clientului și termină conectarea înfăptuită. |
| Accepting\_Connection(const char\* clientIP, SOCKET clientSocket) | În cazul conexiunii acceptate de regulile firewall-ului,transmite mesajul respectiv clientului, si înfaptuiește legătura între client si server |
| Receiving\_Message(SOCKET clientSocket, char\* received\_Buffer, const int LEN\_MESSG) | Etapa de acceptare a mesajului transmis de client |
| Send\_Message(char \* send\_Buffer, const int LEN\_MESSG, SOCKET clientSocket) | Etapa de transmitere a mesajului către client |
| Firewall\_Commands(Firewall FR) | Bloc de comenzi din Firewall.h, ce oferă utilizatorului(server) manipularea regulilor de restricții |

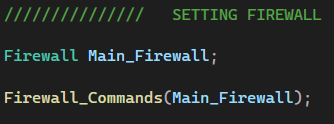
**Din funțiile 6 și 7 se observa implementarea optionala a transferului de date prin rețea.**

**2.2.3. Funcția și structura Server.cpp**

Are ca rol folosirea,manipularea tuturor funcțiilor expuse mai sus pentru crearea și setarea serverului dorit.

Executarea file-ului este împărțit pe mai multe etape:

1. Crearea obiectului Firewall și setarea sa



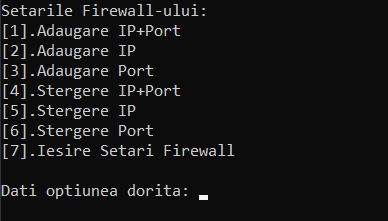


Fig.1

1. Încărcarea bibliotecii Windows și verificarea deschiderii acesteia

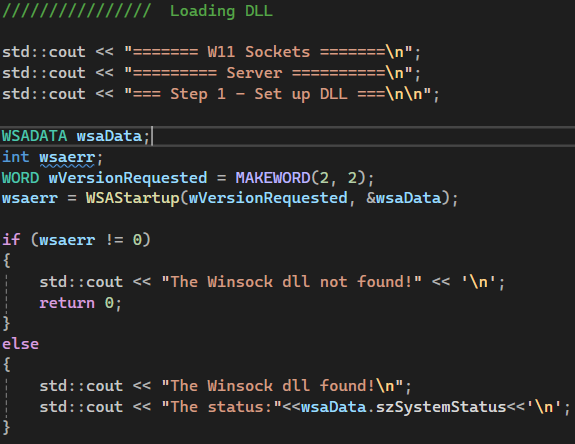


Fig.2

1. Crearea socketului

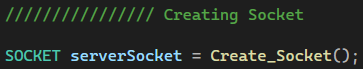


Fig.3

1. Inițializarea serviciului, setând IP -ul,port-ul,protocolul serverului

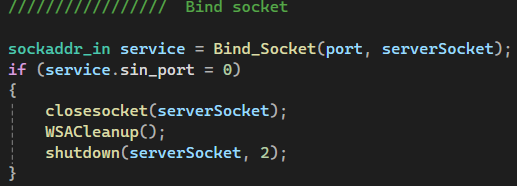


Fig.4

1. Etapa de ascultare/pregatire de acceptare a clientului

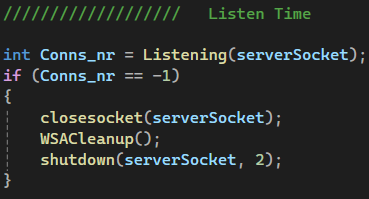


Fig.5

1. Acceptarea clientului în rețea

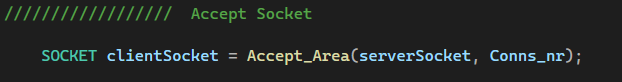


Fig.6

1. Preluarea informației necesare(IP și portul) pentru verificare

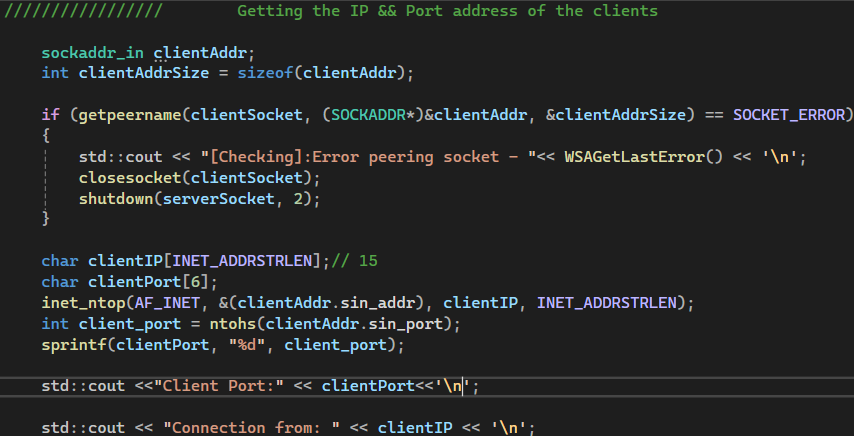
**

Fig.7

1. Verificarea statultului de validare a clientului conectat, 9) primirea și 10) transmiterea informației



Fig.8

1. Închiderea serverului

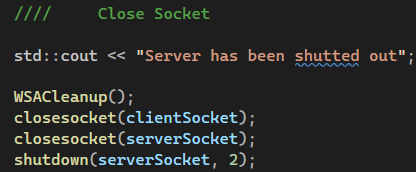


Fig.9

* 1. **Clientul**
     1. **Rolul FUNCS.cpp și FUNCS.h**

Conțin toate funcțiile necesare lucrului file-ului Client.cpp

TABELUL.3

|  |  |
| --- | --- |
| Numele Funcției | Rolul Funcției |
| Create\_Socket() | Crează socket-ul pentru client,specificând familia adresei,tipul socket-ului, si tipul protocolului.Dacă nu se poate crea,returnează eroare |
| Connecting\_Client(int port, SOCKET clientSocket) | Are ca scop crearea serviciului clientului,setarea IP-ului,portului,protocolului și conectarea către server |
| Receiving\_Message(SOCKET clientSocket, char\* received\_Buffer, const int LEN\_MESSG) | Acceptă date de la server |
| Send\_Message(char\* send\_Buffer, const int LEN\_MESSG, SOCKET clientSocket) | Transmite date către server |
| comparing(const char\* recv, const char\* source) | Funcție folosită pentru compararea mesajului de validare transmis de către server : “BLOCKED” sau “ACCEPTED” |

* + 1. **Funcția și structura Client.cpp**

1),2) Încărcarea bibliotecii și crearea socketului este aceeași ca și in punctele 2),3) în secțiunea 2.2.3.

3) Conectarea clientului la serverul cu portul dorit

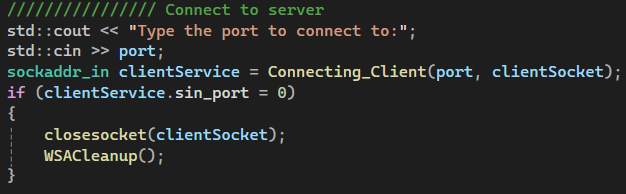


Fig.10

4) Etapa de primire a mesajului de validare de către server

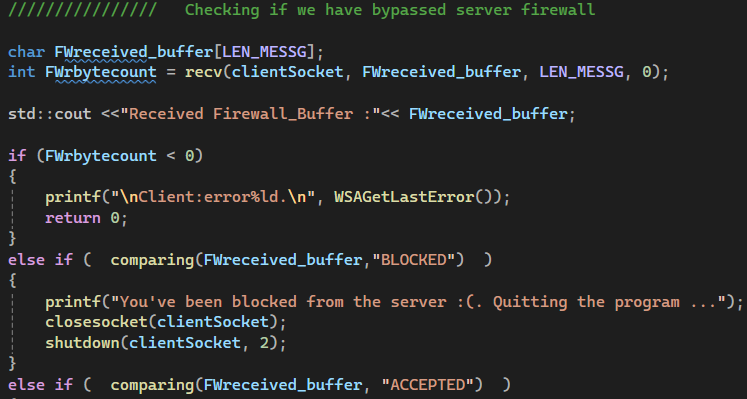


Fig.11

5) Comunicarea datelor prin rețea în cazul validării

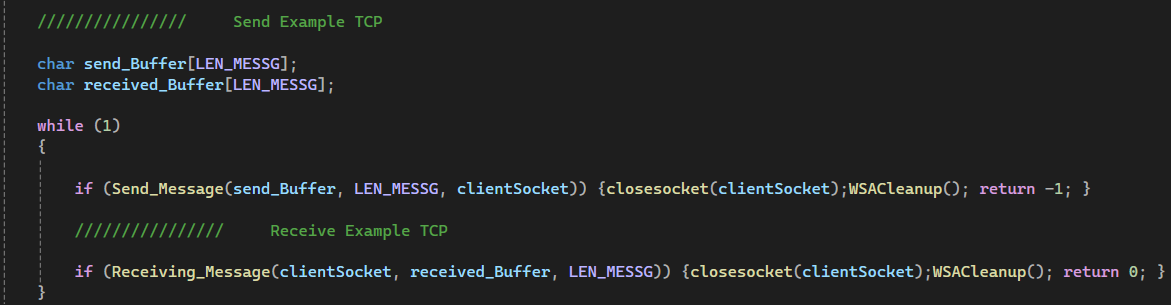


Fig.12

6) Închiderea socketului clientului și întreruperea comunicării

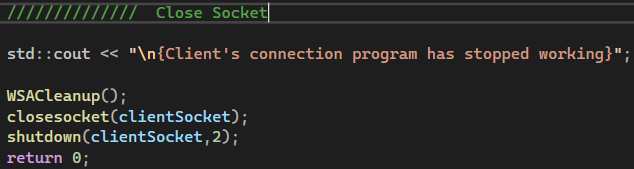


Fig.13

1. **CONCLUZII**

În acest fel,am implementat un Firewall pentru un program server/client.În cadrul acestui proiect am învațat bazele creării unui server și modul de lucru al bibliotecii WinSock2.La fel ,o experiența foarte importantă a fost elaborarea Firewall-ului,o parte componentă simplă în teorie,dar destul de dificilă în implementare.

Ca viitoarele îmbunătățiri acestui proiect aș putea propune :

1. Verificarea conexiunilor după protocoale
2. Conectări multiple
3. Verificarea performanțelor serverului și îmbunatățirea acesteia
4. Refacerea etapei de comunicare a datelor prin rețea
5. Implementarea analizei packetelor și metodelor de transmitere a acestora
6. Optimizarea codului și simplificarea sa
7. Implementarea Firewall-ului pentru un server UDP
8. Adăugarea administratorului pentru manipularea Firewall-ului din exterior
9. Protecția împotriva DDoS atacuri

Restul codului este atașat împreună cu acest proiect și nu a fost prezentat din cauza volumlui.

1. **BIBLIOGRAFIE**
2. <https://learn.microsoft.com/en-us/windows/win32/api/winsock2/>
3. <https://cplusplus.com/reference/>
4. <https://stackoverflow.com/>