**Utilizarea soclurilor de windows pentru realizarea de aplicatii client - server**

**-scanner de porturi pentru sistemele dintr-o retea-**

**Limbajul de programare, mediul de dezvoltare**

Aplicatia este dezvoltata in limbajul C++. Am folosit librariile MFC ( Michrosoft Foundation Classes ) pentru interfata si pentru ferestrele folossite de sockets.

Aplicatia de client foloseste si Iphlpapi.lib, librarie cu ajutorul careia se obtin informatii despre conexuinile TCP si UDP deschise.

Ca mediu de dezvoltare am folosit Microsoft Visual studio 2005.

**Motivatia lucrarii**

In zilele, cand aproape toata lumea are acces la internet, iar atacurile asupra sistemelor sunt din ce in ce mai numeroase, administratorii si specialistii in securitatea retelelor au din ce in ce mai mult de lucru. Atunci cand un program distructiv infecteaza un sistem, actiunea lui poate deteriora date importante sau poate sa nu faca nimic altceva in afara de a se raspandi mai depate. Pentru a evita pericolul, trebuie in primul rand inteles care este acesta, modalitatile de infiltrare si metodele prin care acestea pot fi combatute. Printre aceste amenintari se numara virusi, viermi, cai troieni, dar si atacuri asupra porturilor.

O retea de calculatoare prezintă mai multe puncte vulnerabile; asupra acestora trebuie sa-si indrepte atentia administratorul de sistem, deoarece ele vor fi in mod special vizate de diferitii atacatori (hackeri, crackeri etc.).

Scanarea porturilor reprezinta o modalitate prin care atacatorii descopera porturile deschise si serviciile asociate acestor porturi si gasesc vulnerabilitati in sistem pe care pe exploateaza.

Dar scanarea porturilor reprezinta si o modalitate folosita de administratori pentru a descoporii punctele slabe din sistem si a gasi metode de protectie impotriva atacatorilor. Un administrator poate sa verifice ce porturi sunt deschise, inchise sau blocate si ce porturi pot fi blocate pentru a evita posibilitatea unei scanari. Managementul porturilor reprezinta o metoda buna de a spori securitatea sistemului, si implicit a retelei.

Pornind de la aceasta solutie si de la dorintea mea de a aprofunda programarea cu socluri de windows, am inceput sa realizez o aplicatie care va ajuta un administrator de retea sa tina sub observatie sistemele dintr-o retea. Prin implementarea acestei aplicatii am reusit sa ating mai multe puncte pe care le-am considerat interesante cum ar fi crearea unei aplicatii de securitate, utilizarea de socluri de windows intr-o aplicatie client server sau crearea de servicii de Windws.

**Aplicatia**

Aplicatia propriu – zisa este alcatuita din 3 module. Modulul principal este reprezentat de un Server care va fi instalat pe unul din sistemele din LAN.Prin conectarea la acest server al doilea modul denumit AdminTool va putea afisa informatiile cerute pentru toti clientii din retea. Clienti sunt denumiti toti utilizatorii care au instalat cel de-al treilea modul, modulul de Client. Acest modul poate fi instalat pe oricate sisteme din retea si la pornirea lui se creeaza o conexiune cu Serverul, conexiune prin care acesta poate obtine toate informatiile necesare.

Modulele Client si Server vor rula ca servicii de Windows, numai modulul de Admin avand o interfata grafica (GUI – Graphical User Interface). Acest modul este dezvoltat pe arhitectura claselor MFC(Microsoft Foundation Classes). Această arhitectura permite dezvoltarea cu uşurinţă a aplicaţiilor tip MDI(Multiple Document Interface) dar şi accesul la tehnologiile Microsoft. Alegerea acestei librării este motivată şi de documentaţia existenta pe site-ului Microsoft legată de structura Windows-ului şi a claselor MFC.

**Comunicarea intre module**

Pentru a realiza comunicarea intre cele trei module ale aplicatiei, am implementat o clasa wrapper peste un handle de socket – SOCKET, suprascriind toate metodele necesare cum ar fi *Create, Bind, Connect, Listen, Send* sau *Receive.*Prin aceasta suprascriere nu am facut altceva decat sa apelez functiile de WINSOCK API

Fiecare din cele trei module contine cate o fereastra, vizibila numai la modulul de Admin, fereastra utilizata in prinderea si tratarea mesajelor socketilor. Astfel am evitat utilizarea socketilor blocanti, metoda ce ar fi ingreunat considerabil functionarea aplicatiei. Asignarea ferestrei drept destinatarul mesajelor socketilor se realizeaza prin apelarea functiei *WSAAsyncSelect*, suprascrisa drept AsyncSelect.

Antetul acestei functii este:

int WSAAsyncSelect(SOCKET *s*, HWND *hWnd*, unsigned int *wMsg*, long *lEvent*);

Parametrii:

s- handle al socketului ce urmeaza sa trimita mesajele catre fereastra

hWnd - handle-ul ferestrei destinatar al mesajelor.

wMsg – ID al mesajului ce urmeaza sa fie primit in urma primirii mesajului

de la socket.

lEvent – eventuri ale socketului ce va genera mesajul de mai sus.

Aceste eventuri pot fi spre exemplu FD\_READ, ce anunta ca socketul este gata sa citeasca date, sau FD\_CLOSE care anunta inchiderea socketului asociat.

Pentru modulul de Admin fereastra care primeste aceste mesaje este chiar fereastra principala a aplicatiei. La celelalte doua module a fost necesara crearea, in momentul pornirii serviciului, a unui nou thread pe care se construieste o noua ferestra invizibila. Acest thread se va opri in momentul opririi serviciului.

Astfel in momentul crearii ferestrei Serverului, se creaza si un socket de ascultare pentru clienti si unul de ascultare pentru Admin. In momentul in care un client sau un Admin va incerca conectarea la server se va crea un nou socket care se va lega cu cel ce a incercat conectarea si va fi retinut intr-o lista.

O data stabilita legatura intre Server si Clienti sau intre Server si Admin comunicarea propriu zisa se realizeaza prin comenzi trimise in bufferul functiilor *Send* sub forma de siruri de caractere cu urmatoarea structura:

Lungime\_comandaCOMMAND=NUME\_COMANDA&PARAM1=val1&PARAM2=val2…

Unde:

Lungime\_comanda – lungimea sirului de caractere reprezentat pe 4 caractere

COMMAND – un identificator predefinit

NUME\_COMANDA – poate avea o serie de valori printre care

LISTUSERS – cere serverului trimiterea tuturor clientilor

GETDETAILS - cere serverului trimiterea detaliilor pentru

un client

“&”, “=” – separator predefinit

PARAM1 – numele primului parametru, cum ar fi numele clientului

pentru care se cer detaliile

val1 – valoare primului parametru.

La notificarea unui socket ca este gata sa primeasca un buffer, se vor citi primele patru caractere ce reprezinta lungimea comenzii primite, apoi se va citi un buffer de lungime egala cu valoarea citita anterior. Acest buffer este apoi parsat in vederea obtinerii numelui comenzii a parametrilor aferenti.

Parsarea comenzii primite se realizeaza prin cautarea separatorilor predefiniti “&” si “=” precum si a identificatorului COMMAND. Pentru aceasta am utilizat o clasa ajutatoare CcommandTokenizer intr-o functie ce primeste ca parametri bufferul de parsat si o mapa de siruri de caractere in care cheie va fi numele parametrului si valoare chiar valoarea acestuia.

O data rezolvata problema conectarii modulelor, a stabilirii unui protocol de comunciare si a unei formatari a comenzilor a mai ramas de implementat acizitionarea datelor necesare la nivel de client, adica listarea conexiunilor deschise si a statisticilor TCP si UDP.

Pentru aceasta am utilizat o serie de functii ce permit obtinerea informatiilor despre parametrii de retea ce se gasesc in Internet Protocol Helper (IP Helper Api). Acest Api face posibila obtinerea si modificarea parametrilor de retea dintr-un sistem. Functiile pot fi folosite in programarea C/C++, aplicatiile tipice fiind protocoale de rutare IP si agenti SNMP (Simple Network Management Protocol).De asemenea ,IP Helper contine mecanisme de notificare ce pot fi folosite pentru a anunta ca anumiti parametrii de retea s-au modificat.

IP Helper face posibil accesarea informatiilor legate de protocoalele de retea ce sunt folosite de catre Windows.

**5.Metode alternative de implementare**

O alta metoda ce ar fi putut fi utilizata pentru a realiza aceasta aplicatie ar fi fost utilizarea *performance counter-elor* cu ajutorul librariei Pdh. Aceasta librarie pune la dispozitia programatorului o serie de functii prin care un server se poate conecta la sistemele dintr-o retea si poate cere datele citite de anumite *performance counter.*

Avantajul principal ar fi ca s-ar evita implementarea layer-ului de socketi implicit toate problemele legate de comunicare, de aceasta ocupandu-se intern libraria.

Ca dezavantaj poate fi privit faptul ca pentru a se putea conecta serverul trebuie sa primeasca o serie de drepturi asupra sistemelor tinta, printre care drepturi totale asupra unor chei din registrii si asupra unor fisiere de sistem, fapt ce ar putea crea anumite probleme de securitate.