**//Implement a program to Avoiding Server Hijacking.**

#include <winsock2.h>

#include <stdio.h>

#include <assert.h>

#include "SocketHelper.h"

//If you have an older version of winsock2.h

#ifndef SO\_EXCLUSIVEADDRUSE

#define SO\_EXCLUSIVEADDRUSE ((int)(~SO\_REUSEADDR))

#endif

/\*

  This application demonstrates a generic UDP-based server.

  It listens on port 8391. If you have something running there,

  change the port number and remember to change the client too.

\*/

int main(int argc, char\* argv[])

{

    SOCKET sock;

    sockaddr\_in sin;

    DWORD packets;

    bool hijack = false;

    bool nohijack = false;

    if(argc < 2 || argc > 3)

    {

        printf("Usage is %s [address to bind]\n", argv[0]);

        printf("Options are:\n\t-hijack\n\t-nohijack\n");

        return -1;

    }

    if(argc == 3)

    {

        //Check to see whether hijacking mode or no-hijack mode is

        //enabled.

        if(strcmp("-hijack", argv[2]) == 0)

        {

            hijack = true;

        }

        else

        if(strcmp("-nohijack", argv[2]) == 0)

        {

            nohijack = true;

        }

        else

        {

            printf("Unrecognized argument %s\n", argv[2]);

            return -1;

        }

    }

    if(!InitWinsock())

        return -1;

    //Create your socket.

    sock = socket(AF\_INET, SOCK\_DGRAM, IPPROTO\_UDP);

    if(sock == INVALID\_SOCKET)

    {

        printf("Cannot create socket -  err = %d\n", GetLastError());

        return -1;

    }

    //Now let’s bind the socket.

    //First initialize the sockaddr\_in.

    //I’m picking a somewhat random port that shouldn’t have

    //anything running.

    if(!InitSockAddr(&sin, argv[1], 8391))

    {

        printf("Can’t initialize sockaddr\_in - doh!\n");

        closesocket(sock);

        return -1;

    }

    //Let’s demonstrate the hijacking and

    //anti-hijacking options here.

    if(hijack)

    {

        BOOL val = TRUE;

        if(setsockopt(sock,

                      SOL\_SOCKET,

                      SO\_REUSEADDR,

                      (char\*)&val,

                      sizeof(val)) == 0)

        {

            printf("SO\_REUSEADDR enabled -  Yo Ho Ho\n");

        }

        else

        {

            printf("Cannot set SO\_REUSEADDR -  err = %d\n",

                   GetLastError());

            closesocket(sock);

            return -1;

        }

    }

    else

    if(nohijack)

    {

        BOOL val = TRUE;

        if(setsockopt(sock,

                      SOL\_SOCKET,

                      SO\_EXCLUSIVEADDRUSE,

                      (char\*)&val,

                      sizeof(val)) == 0)

        {

            printf("SO\_EXCLUSIVEADDRUSE enabled\n");

            printf("No hijackers allowed!\n");

        }

        else

        {

            printf("Cannot set SO\_ EXCLUSIVEADDRUSE -  err = %d\n",

                   GetLastError());

            closesocket(sock);

            return -1;

        }

    }

    if(bind(sock, (sockaddr\*)&sin, sizeof(sockaddr\_in))  == 0)

    {

        printf("Socket bound to %s\n", argv[1]);

    }

    else

    {

        if(hijack)

        {

            printf("Curses! Our evil warez are foiled!\ n");

        }

        printf("Cannot bind socket -  err = %d\n", GetLastError());

        closesocket(sock);

        return -1;

    }

    // OK, now we’ve got a socket bound. Let’s see whether someone

    //sends us any packets - put a limit so that we don’t have to

    //write special shutdown code.

    for(packets = 0; packets < 10; packets++)

    {

        char buf[512];

        sockaddr\_in from;

        int fromlen = sizeof(sockaddr\_in);

        // Remember that this function has a TRINARY return;

        //if it is greater than 0, we have some data;

        //if it is 0, there was a graceful shutdown

        //(shouldn’t apply here);

        //if it is less than 0, there is an error.

        if(recvfrom(sock, buf, 512, 0, (sockaddr\*)&from, &fromlen)> 0)

        {

            printf("Message from %s at port %d:\n%s\n",

                   inet\_ntoa(from.sin\_addr),

                   ntohs(from.sin\_port),

                   buf);

            // If we’re hijacking them, change the message and

            //send it to the real server.

            if(hijack)

            {

                sockaddr\_in local;

                if(InitSockAddr(&local, "127.0.0.1", 83 91))

                {

                    buf[sizeof(buf)-1] = ’\0’;

                    strncpy(buf, "You are hacked!", siz eof(buf) -1);

                    if(sendto(sock,

                              buf,

                              strlen(buf) + 1, 0,

                              (sockaddr\*)&local,

                              sizeof(sockaddr\_in)) < 1)

                    {

                        printf

                     ("Cannot send message to localhost - err = %d\n",

                      GetLastError());

                    }

                }

            }

        }

        else

        {

            printf("Ghastly error %d\n", GetLastError() );

            break;

        }

    }

    return 0;

}