

Forum:[Hack Forums](#)

Thread: [Network\(Socket\) Programming - Q & A's \(Part-3\)](#)

?si?qo? - December 27, 2008, 6:25 am

Part One: <https://hackforums.net/showthread.php?tid=39070>

Part Two: <https://hackforums.net/showthread.php?tid=40287>

In this part 3, i'm going to talk about socket programming in various aspects. You will find this thread useful if you are into socket programming in C/C++. Otherwise, you will find some difficulties in understanding everything i posted in here.

Q: What is a port scanner? Whats the point behind the port scanner?

Port Scanning is one of the most popular reconnaissance (examination or exploration of an area) techniques attackers use to discover services they can break into. All machines connected to a Local Area Network (LAN) or Internet run many services that listen at well-known and not so well known ports. A port scan helps the attacker find which ports are available (i.e., what service might be listing to a port). Essentially, a port scan consists of sending a message to each port, one at a time. The kind of response received indicates whether the port is used and can therefore be probed further for weakness.

The reason port scanners are used by hackers is because they can find out what type of programs you are running. After that, he may try to communicate with it to find out more about that app's name, version and so forth. This will allow the hacker to use this information to find exploits in your client's side app(buffer overflow) and attack them. This will allow them to have access to your computer. Also a port scanner can detect if you have a shared folder and if they can have access to it if I am not mistaken.

For more info: http://www.auditmypc.com/freescan/readingroom/port_scanning.asp

==*==*==*==*==*==*==*==*==*==*==*==*==*==*==*==

Q: What are the major steps to be followed while creating a server?

- A: 1. create a socket
2. bind the socket (to a port)
3. listen on the socket
4. accept connections (from clients)

==*==*==*==*==*==*==*==*==*==*==*==*==*==*==*==

Q: What are the major steps to be followed while creating a client?

- A: 1. create a socket
2. connect to the server (at whatever port/hostname the server is using)

==*==*==*==*==*==*==*==*==*==*==*==*==*==*==*==

Q: Can you tell me some of connect() errors?

A: When 'connect' returns -1, you need to check errno to find out what the error was. Only when it's ECONNREFUSED can you decide that the port is either not listening or blocked. The other error codes indicate another kind of problem that you'll probably have to resolve first.

Visit for more details:

<http://www.opengroup.org/onlinepubs/000095399/functions/connect.html>

[illegible]

Q: Where can i declare errno in Unix & Linux?

A: just add this header `#include<errno.h>`

— * — * — * — * — * — * — * — * — * — * — * — * — * — * — * —

Q: What "errno" Represents the Port is Currently in Use?

A: If a server is listening on a port, and you are trying to connect to that port, your connection will either be accepted or refused. You will not get an error like "port already in use", since multiple connections to the same port can exist as long as the server allows that.

—*—

Q: I created local port scanner under linux, all of the ports show as blocked, how can i open one port to test my code whether it can catch it or not?

A: use "sudo nc -l -p portnumber"

This command open the port specified temporarily & then close it, so you have to test your code before the nc closes the opened port.

—*—

General Note:

The server opens the port, and starts listening on it for incoming connections from a client. Not the client opens the port and starts listening from a server. A listening port would either allow you to connect to it or refuse your connection for some reason (in which case it's blocked).

-*-**--*-*--*-*--*-*--*-*--*-*--*-*--*-*--*-*--*-*--*-*--*-*--

Q: Does socket programming differs alot from OS to another?

Not really, just slight changes needed.

LINUX:

If you want to do socket programming under LINUX Ubuntu, you should add the appropriate headers. No manual linknig required.

Look at RemotePortScanner_LINUX.cpp, to learn how the structure of socket programming should be in LINUX Ubuntu.

UNIX:

If you want to do socket programming under Unix Sun Solaris OS, you should link your code to the socket library while compiling:

```
g++ RemotePortScanner_UNIX.cpp -lsocket -lnsl
```

Don't forget to add this header `#include<unistd.h>` in your code

Look at RemotePortScanner_UNIX.cpp, to learn how the structure of socket programming should be in UNIX SunOS.

WINDOWS:

If you want to do socket programming under Windows, you should link your code (project) to winsock library:

File -> New -> Project -> Win32 -> Win32 Console Application or Win32 Project for GUI -> Click Next on the New Window -> Leave Empty Project Unchecked -> Go to Project Tab -> fileName Properties (or click alt+f7) -> Configuration Properties -> Linker -> Input -> Additional Dependencies field -> type in WS2_32.lib. However, in VC++ Express 2008, libraries are stored under Microsoft SDKs folder, so you should enter the whole path name "C:\Program Files\Microsoft SDKs\Windows\v6.0A\Lib\WS2_32.lib". Do not forget to put the quotation around the path name. Otherwise, you will be getting the following error:

```
LINK : fatal error LNK1104: cannot open file 'C:\Program.obj'
```

for more info:

[http://www.errorbase.net/LINK : fatal_error_LNK1104: cannot open file %27\(X\).obj%27](http://www.errorbase.net/LINK:_fatal_error_LNK1104:_cannot_open_file_%27(X).obj%27)

You could link to winsock library by adding this code after the includes headers:

```
#pragma comment(lib, "Ws2_32.lib")
```

Look at RemotePortScanner_WINDOWS.cpp, to learn how the structure of socket programming should be in Windows OS.

Please note this code works good, but the style need re-structured in more efficient & correct way - [Download](#)

```
==*==*==*==*==*==*==*==*==*==*==*==*==*==*==*==*
```

Q: My Socket Programming Code Compiled Without Using winsock - `#include"WinSock.h"?! It works when i put this #include<Ws2tcpip.h> but not this #include"WinSock.h"`

A: I am fairly certain that Ws2tcpip is Winsock2. Your code might be using parts of Winsock that were introduced in Winsock2 and not available in Winsock.h.

Ws2tcpip.h This is where the addrinfo lives. ETIMEDOUT and ECONNREFUSED are probably the Linux names for these constants. Windows names them slightly differently. Try WSAETIMEDOUT and WSAECONNREFUSED. These values are declared in winsock2.h and winerror.h header files.

—*—

Q: What does error 10060 refer to?

A: When `WSAGetLastError()` returns 10060, `WSAETIMEDOUT` this means
A connection attempt failed because the connected party did
not properly respond after a period of time, or established
connection failed because connected host has failed to
respond.

Error: 10060 is fairly similar to 10061, it just depends what kind of connection you are using, wireless or LAN ... I do not know there could be other reasons why sometimes you get 10061 & sometime you get 10060, i can't remember the proper reasons on top of my head - Google it if you really care!

==*==

Q: In which library does `getaddrinfo()` reside?

A: Windows Sockets: winsock.h, and wsock32.lib

[illegible]

Q: What is the Best C++ Compiler that is Good for Writing Client-Server Apps (GUI Client-Server Apps.)?

A: what is your operating system! For windows its been said Visual C++ is a good one. If you are dealing with unix, gnu is good. If the client is running on one operating system, and the server on another, then you will need a different compiler for each operating system and use corba as a middle layer to talk between the client and server. What the heck is corba! visit <http://www.webopedia.com/TERM/C/CORBA.html>

—*—

Q: What is DLL?

A: "Well for starters, what the heck is a DLL? DLL stands for Dynamic Link Library. That means that the "dll executable code" isn't linked to until it's needed by your "main executable code".

DLL's are used for two main reasons:

- 1) They give programmers the ability to have "a piece of code" shared by many applications easily and effectively.

2) They make updating an existing .exe (that uses DLL's) easy. Simply pop in a new DLL and new things will happen. Take Winamp as example. DLL's can also hold things like bitmaps, so you could use them to update "player skins" like in Unreal Tournament."

this above part of explanation is quoted from GameTutorials CD under C++ category in "Using a

DLL" section. Just not break the copyright, here is the source of the website if you want to grab the CD:

<http://www.gametutorials.com/gtstore/pc-35-6-gametutorials-squared-cd.aspx>

—*—

Q: What does the following code do? Please explain each line...

Why we need to put in Windows Socket Programs?

Why we do not use it in Linux or Unix?

Why at the end we do cleanup?

What WSADATA, WSASStartup, etc Stands for? I mean WSA?!

WSA stands for windows socket application.

WORD wVersionRequested;

```
WSADATA wsaData;
```

```
int err;
```

```
wVersionRequested = MAKEWORD(2, 2);
```

```
err = WSASStartup(wVersionRequested, &wsaData);
```

```
if (err != 0) {
```

```
printf("WSAStartup failed with error: %d\n", err);
```

```
return 1;
```

}

[illegible]

```
// All the networking information is included in one library. The header you need
```

```
// to include is winsock2.h Let's do that now to get started..
```

```
#include <winsock2.h>
```

•

■

■

■

```
// Now that we have winsock included, winsock must be initialized before we can
```

```
// use any networking functions. You do this with WSASStartup()..
```

```
// We need a couple variables first..
```

```
// All winsock functions return an error code, so we will use this to get more information
```

```
// about our errors..
```

```
int error;
```

```
// And a winSock data structure that will hold information about our network
```

// capabilities. We will not, however, use any of the information in it. It

// is included simply for completeness..

```
WSAData wsaData:
```

```
// And the call to start up winSock.
```

```
// Before we get going too fast, have no fear about the MAKEWORD macro. It is simply used so
winsock can
// identify what version of winsock we are trying to use. In this case, we are using v2.2.
// You can research more if you would like to inquire about earlier versions of winsock and/or what
their differences
// with v2.2 are if you chose to implement them..

// Now the nice thing about winsock is the way it handles return values and errors.
// Virtually EVERY winsock function will return SOCKET_ERROR if an error occurred, and
something else
// if it was successful. This makes error checking quite simple..
```

```
// So lets check for an error..
if (error == SOCKET_ERROR)
{
    printf("Could Not Start Up Winsock!\n");
    return 1;
}
```

•
•
•
•
•
•
•

```
// Finally, we have to shut down winsock. This MUST be done in order to clean up properly.
// WSASStartup() loads a .dll ... so if we do not free that .dll, we have caused a memory leak,
// which could eventually cause some problems. We shutdown winsock with a call similar to
// WSASStartup(). It is appropriately named WSACleanup(). This function requires
// no arguments and only needs to be called once..
```

```
// Shut down winsock.
WSACleanup();
```

[illegible]

Thats all to it!

Just not to break the law: the answer that is between the ampersand's are taken from GameTutroials LLC product under "C++" category in "Creating a Socket" tutorial.

source: <http://www.gametutorials.com/gtstore/pc-35-6-gametutorials-squared-cd.aspx>

Extra Info:

&wsaData (points to a structure WSADATA) will be filled with results by the function WSASStartup (output). WSADATA Contains:

will be the version less than or equal to the version requested, depending on what is available on the system. (2.2 as of Windows 98 SP2)

is a string with a description of the WinSock API

is a string, not usually used. Provides extra "start up" info.

the maximum amount of Sockets allowed.

is the maximum size of a UDP packet.

is a vendor specific struct, this is not normally used.

- It's a Microsoft Windows specific API

- To free up any used resources for other programs to use.

- Memory leak is a particular type of unintentional memory consumption by a computer program where the program fails to release memory when no longer needed.

==*-*-*-*==*-*-*-*==*-*-*-*==*-*-*-*==*-*-*-*==*-*-*-*==*-*-*-*==*-*-*-*==*-*-*-*==*-*-*-*==

```
getaddrinfo(remoteIP, REMOTEPORT, &hints, &results);
```

I do not think this would work, because based on what i have learned REMOTEPORT is considered as local port because 2nd Parameter only made to specify a local port.

A: `getaddrinfo` is simply used to retrieve information for building the sockets. It's the call to either `bind()` or `connect()` decides how this information is used.

==*-*==

Assume servInfo carries 5 ip addresses, the function bind() will bind to which IP address?

A: bind() will use the first element. the purpose of the bind() function is to choose exactly one source identification; in the case of IP, one source address and source port. So no matter how many IPs and source ports are available, you can only pick one address to bind() to.

In other words, the call to `bind()` will use only the first `addrinfo` structure in the list.

==*==*==*==*==*==*==*==*==*==*==*==*==*==*==*==

Q: Why IPv4 is returned as ::ffff:127.0.0.1?

A: This means your network interface supports IPv6 addresses, so that's what it uses. The IPv6 address ::ffff:127.0.0.1 is the exact same address as the IPv4 address 127.0.0.1

==*==*==*==*==*==*==*==*==*==*==*==*==*==*==*==

Q: The first IP address appears as ::aff:c0fd:7fbf the second appears as ::ffff:127.0.0.1 I think i only should get 127.0.0.1, i'm not using any IPv6. Do you know why? Why IP at first time appears differently than the rest?

Sent 35 bytes to client : ::aff:c0fd:7fbf
Sent 35 bytes to client : ::ffff:127.0.0.1
Sent 35 bytes to client : ::ffff:127.0.0.1
Sent 35 bytes to client : ::ffff:127.0.0.1

A: >>I think i only should get 127.0.0.1, i'm not using any IPv6.
This means your network interface supports IPv6 addresses, so that's what it uses. The IPv6 address ::ffff:127.0.0.1 is the exact same address as the IPv4 address 127.0.0.1

>>Why IP at first time appears differently than the rest
Your client has an address, which might be represented as either IPv4 or IPv6. In this case, your protocol stack chose to represent it as IPv6.

==*==*==*==*==*==*==*==*==*==*==*==*==*==*==*==

Q: How come sendto() sends to messages to any live address? How come it sends messages to closed ports? In short, How come it bypass the firewalls! And successfully send messages to any live address outhere?!

A: UDP is connectionless ... The success in sendto means that message was sent ... It does not imply recipient received it. If you need guaranteed delivery, use TCP. In UDP, packet goes & never responds back, it can get lost, it may loose some data on the way, or it may never reach.

==*==*==*==*==*==*==*==*==*==*==*==*==*==*==*==

Q: I wrote a port scanner that uses connect() on every port i test. I want to change it to UDP to test ports, i did it by sendto(), but it happens that i can not really know whether the port is open or closed since sendto() just works always as long as the machine is alive....Any idea?

A: You cant do that on UDP ... UDP is connectionless ... you need a connection oriented protocol to get connected.

Another solution, you will have to use ICMP Echo Requests and look for replies with the error "Destination port not found".

==*==*==*==*==*==*==*==*==*==*==*==*==*==*==*==

Q: My Windows Vista always bind my socket with my local IPv/port only. I want it to bind to my both local IP versions or at least binds to IPv4 instead of IPv6. How can i do it?

A: change hints.ai_family = AF_UNSPEC; to hints.ai_family = AF_INET;
As AF_UNSPEC, picks the highest numbered family protocol which happens to be IPV6 with Windows.

you may check the protocol binding settings on Vista...

Change the order of network protocol bindings (for Vista)

<http://windowshelp.microsoft.com/Windows/en-us/help/89554521-7ce7-4d70-aa29-88b8316d50761033.mspx>

[illegible]

Q: How come local IPv4 & local IPv6 are not related? They are in the same machine, same system!
Can you explain?

A: the two protocols are totally separated, as they use different stack and different addressing system. you may just simply think IPv6 as IPX, another different protocol.

>>They are in the same machine, same system!
theoretically, not same, just similar.

IPv4 and IPv6 do not share their ports, so IPv4:123 and IPv6:123 are different port, though they have the same port number, because, as i explained above, the two ports belong to different protocol stack, one for each.

—*—

Q: When I Make a Server Listens on 4432 Port for incoming connections. Does this mean, port 4432 is Open?

A: Yes it does mean that the port is open

—*—

There are more to come! Hope fully you have learned something today ?

I'm not responsible if you use anything i post in illegal manners

c u l8r

Aro - June 2, 2009, 6:10 pm

thanks man I am trying to learn more about sockets

scaryblackguy - June 2, 2009, 7:04 pm

epic.....

/tooshort

CppSean - June 2, 2009, 7:20 pm

Wow great tutorial man

H3LL - *June 4, 2009, 2:12 am*

Thanks, i bumped all your threads thanks to this