New issue

Jump to bottom

SEGV in function dwarf::cursor::uleb128 at dwarf/internal.hh:154 #50

Open xiaoxiongwang opened this issue on Aug 15, 2020 · 1 comment

xiaoxiongwang commented on Aug 15, 2020 • edited

Tested in Ubuntu 16.04, 64bit.

The tested program is the example program dump-tree.

The testcase is dump_tree_segv.

I use the following command:

```
/path-to-libelfin/examples/dump-tree dump_tree_segv
```

and get:

```
Segmentation fault (core dumped)
```

I use **valgrind** to analysis the bug and get the below information (absolute path information omitted):

```
valgrind /path-to-libelfin/examples/dump-tree dump_tree_segv
==21176== Memcheck, a memory error detector
==21176== Copyright (C) 2002-2015, and GNU GPL'd, by Julian Seward et al.
==21176== Using Valgrind-3.11.0 and LibVEX; rerun with -h for copyright info
==21176== Command: /path-to-libelfin/examples/dump-tree dump_tree_segv
==21176==
==21176== Invalid read of size 1
==21176==    at 0x431161: uleb128 (internal.hh:154)
==21176==    by 0x431161: dwarf::die::read(unsigned long) (die.cc:35)
==21176==    by 0x44869E: dwarf::value::as_reference() const (value.cc:215)
==21176==    by 0x44C482: dwarf::to_string[abi:cxx11](dwarf::value const&) (value.cc:324)
==21176==    by 0x404A3B: dump_tree(dwarf::die const&, int) (dump-tree.cc:19)
==21176==    by 0x4035C1: dump_tree (dump-tree.cc:21)
==21176==    by 0x4035C1: main (dump-tree.cc:43)
==21176==  Address 0x5b02809b is not stack'd, malloc'd or (recently) free'd
==21176==
==21176==
==21176== Process terminating with default action of signal 11 (SIGSEGV)
==21176==  Access not within mapped region at address 0x5B02809B
==21176==    at 0x431161: uleb128 (internal.hh:154)
==21176==    by 0x431161: dwarf::die::read(unsigned long) (die.cc:35)
==21176==    by 0x44869E: dwarf::value::as_reference() const (value.cc:215)
==21176==    by 0x44C482: dwarf::to_string[abi:cxx11](dwarf::value const&) (value.cc:324)
==21176==    by 0x404A3B: dump_tree(dwarf::die const&, int) (dump-tree.cc:19)
==21176==    by 0x4035C1: dump_tree (dump-tree.cc:21)
==21176==    by 0x4035C1: main (dump-tree.cc:43)
==21176==  If you believe this happened as a result of a stack
==21176==  overflow in your program's main thread (unlikely but
==21176==  possible), you can try to increase the size of the
==21176==  main thread stack using the --main-stacksize= flag.
==21176==  The main thread stack size used in this run was 8388608.
--- <0>
<b> DW_TAG_compile_unit
      DW_AT_producer
      DW_AT_language 4 byte block: cb 0 0 0
      DW_AT_name
      DW_AT_comp_dir
      DW_AT_low_pc 0x0
      DW_AT_high_pc 0x1500000000000000
      DW_AT_stmt_list <line 0x0>
 <2d> DW_TAG_base_type
       DW_AT_byte_size 0x8
       DW_AT_encoding 0x7
       DW_AT_name long unsigned int
 <34> DW_TAG_base_type
       DW_AT_byte_size 0x1
       DW_AT_encoding 0x8
       DW_AT_name
 <3b> DW_TAG_base_type
       DW_AT_byte_size 0x2
       DW_AT_encoding 0x7
       DW_AT_name
 <42> DW_TAG_base_type
       DW_AT_byte_size 0x4
       DW_AT_encoding 0x7
       DW_AT_name
 <49> DW_TAG_base_type
       DW_AT_byte_size 0x1
       DW_AT_encoding 0x6
       DW_AT_name
 <50> DW_TAG_base_type
       DW_AT_byte_size 0x2
       DW_AT_encoding 0x5
       DW_AT_name
 <57> DW_TAG_base_type
       DW_AT_byte_size 0x4
       DW_AT_encoding 0x5
       DW_AT_name int
 <5e> DW_TAG_base_type
       DW_AT_byte_size 0x8
       DW_AT_encoding 0x5
       DW_AT_name
 <65> DW_TAG_base_type
```

```
        DW_AT_byte_size 0x8
        DW_AT_encoding 0x7
        DW_AT_name
    <6c> DW_TAG_base_type
        DW_AT_byte_size 0x1
        DW_AT_encoding 0x6
        DW_AT_name
    <73> DW_TAG_subprogram
        DW_AT_external true
        DW_AT_name
        DW_AT_decl_file 0x1
        DW_AT_decl_line 0x3
==21176==
==21176== HEAP SUMMARY:
==21176==     in use at exit: 81,552 bytes in 68 blocks
==21176==   total heap usage: 182 allocs, 114 frees, 92,963 bytes allocated
==21176==
==21176== LEAK SUMMARY:
==21176==    definitely lost: 0 bytes in 0 blocks
==21176==    indirectly lost: 0 bytes in 0 blocks
==21176==      possibly lost: 0 bytes in 0 blocks
==21176==    still reachable: 81,552 bytes in 68 blocks
==21176==         suppressed: 0 bytes in 0 blocks
==21176== Rerun with --leak-check=full to see details of leaked memory
==21176==
==21176== For counts of detected and suppressed errors, rerun with: -v
==21176== ERROR SUMMARY: 1 errors from 1 contexts (suppressed: 0 from 0)
Segmentation fault (core dumped)
```

I use **AddressSanitizer** to build ffjpeg and running it with the following command:

```
/path-to-libelfin/examples/dump-tree dump_tree_segv
```

This is the ASAN information (absolute path information omitted):

```
/path-to-libelfin-address/examples/dump-tree dump_tree_segv
ASAN:SIGSEGV
=================================================================
==21215==ERROR: AddressSanitizer: SEGV on unknown address 0x7f519be3409b (pc 0x000000417cb5 bp 0x7ffddf8f6830 sp 0x7ffddf8f6730 T0)
    #0 0x417cb4 in dwarf::cursor::uleb128() /path-to-libelfin-address/dwarf/internal.hh:154
    #1 0x417cb4 in dwarf::die::read(unsigned long) /path-to-libelfin-address/dwarf/die.cc:35
    #2 0x422a25 in dwarf::value::as_reference() const /path-to-libelfin-address/dwarf/value.cc:215
    #3 0x425711 in dwarf::to_string[abi:cxx11](dwarf::value const&) /path-to-libelfin-address/dwarf/value.cc:324
    #4 0x403aec in dump_tree(dwarf::die const&, int) /path-to-libelfin-address/examples/dump-tree.cc:19
    #5 0x403bea in dump_tree(dwarf::die const&, int) /path-to-libelfin-address/examples/dump-tree.cc:21
    #6 0x403361 in main /path-to-libelfin-address/examples/dump-tree.cc:43
    #7 0x7f514331582f in __libc_start_main (/lib/x86_64-linux-gnu/libc.so.6+0x2082f)
    #8 0x403878 in _start (/path-to-libelfin-address/examples/dump-tree+0x403878)

AddressSanitizer can not provide additional info.
SUMMARY: AddressSanitizer: SEGV /path-to-libelfin-address/dwarf/internal.hh:154 dwarf::cursor::uleb128()
==21215==ABORTING
```

An attacker can exploit this vulnerability by submitting a malicious elf file that exploits this bug which will result in a Denial of Service (DoS).

👍 1

fgeek commented on Aug 6, 2021

CVE-2020-24822 has been assigned for this issue.

Assignees

No one assigned

Labels

None yet

Projects

None yet

Milestone

No milestone

Development

No branches or pull requests

2 participants