New issue

Jump to bottom

SEGV in function dwarf::cursor::skip_form at dwarf/cursor.cc:191 #52

Open **xiaoxiongwang** opened this issue on Aug 15, 2020 · 2 comments

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_segv3.

I use the following command:

```
/path-to-libelfin/examples/dump-tree dump_tree_segv3
```

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_segv3
==423== Memcheck, a memory error detector
==423== Copyright (C) 2002-2015, and GNU GPL'd, by Julian Seward et al.
==423== Using Valgrind-3.11.0 and LibVEX; rerun with -h for copyright info
==423== Command: /path-to-libelfin/examples/dump-tree dump_tree_segv3
==423==
==423== Invalid read of size 1
==423==    at 0x42C998: uleb128 (internal.hh:154)
==423==    by 0x42C998: dwarf::cursor::skip_form(dwarf::DW_FORM) (cursor.cc:147)
==423==    by 0x433B4C: dwarf::die::read(unsigned long) (die.cc:51)
==423==    by 0x414B7C: dwarf::unit::root() const (dwarf.cc:195)
==423==    by 0x402CD0: main (dump-tree.cc:43)
==423==  Address 0x750280b3 is not stack'd, malloc'd or (recently) free'd
==423==
==423==
==423== Process terminating with default action of signal 11 (SIGSEGV)
==423==  Access not within mapped region at address 0x750280B3
==423==    at 0x42C998: uleb128 (internal.hh:154)
==423==    by 0x42C998: dwarf::cursor::skip_form(dwarf::DW_FORM) (cursor.cc:147)
==423==    by 0x433B4C: dwarf::die::read(unsigned long) (die.cc:51)
==423==    by 0x414B7C: dwarf::unit::root() const (dwarf.cc:195)
==423==    by 0x402CD0: main (dump-tree.cc:43)
==423==  If you believe this happened as a result of a stack
==423==  overflow in your program's main thread (unlikely but
==423==  possible), you can try to increase the size of the
==423==  main thread stack using the --main-stacksize= flag.
==423==  The main thread stack size used in this run was 8388608.
--- <0>
==423==
==423== HEAP SUMMARY:
==423==     in use at exit: 80,652 bytes in 63 blocks
==423==   total heap usage: 120 allocs, 57 frees, 88,208 bytes allocated
==423==
==423== LEAK SUMMARY:
==423==    definitely lost: 0 bytes in 0 blocks
==423==    indirectly lost: 0 bytes in 0 blocks
==423==      possibly lost: 0 bytes in 0 blocks
==423==    still reachable: 80,652 bytes in 63 blocks
==423==         suppressed: 0 bytes in 0 blocks
==423== Rerun with --leak-check=full to see details of leaked memory
==423==
==423== For counts of detected and suppressed errors, rerun with: -v
==423== 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_segv3
```

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

```
/path-to-libelfin-address/examples/dump-tree dump_tree_segv3
ASAN:SIGSEGV
=================================================================
==451==ERROR: AddressSanitizer: SEGV on unknown address 0x7f84237480b3 (pc 0x0000004167e8 bp 0x7fff99fc19f0 sp 0x7fff99fc18f0 T0)
    #0 0x4167e7 in dwarf::cursor::skip_form(dwarf::DW_FORM) /path-to-libelfin-address/dwarf/cursor.cc:191
    #1 0x4183b3 in dwarf::die::read(unsigned long) /path-to-libelfin-address/dwarf/die.cc:51
    #2 0x40f548 in dwarf::unit::root() const /path-to-libelfin-address/dwarf/dwarf.cc:195
    #3 0x403357 in main /path-to-libelfin-address/examples/dump-tree.cc:43
    #4 0x7f83b0c2982f in __libc_start_main (/lib/x86_64-linux-gnu/libc.so.6+0x2082f)
    #5 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/cursor.cc:191 dwarf::cursor::skip_form(dwarf::DW_FORM)
==451==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-2020-24821 has been assigned for this issue.

fgeek commented on Aug 6, 2021

@xiaoxiongwang for your information you can minimize your PoC files by running `AFL_TMIN_EXACT=1 afl-tmin` if you are using afl fuzzer.

Assignees

No one assigned

Labels

None yet

Projects

None yet

Milestone

No milestone

Development

No branches or pull requests

2 participants