New issue

Jump to bottom

stack-buffer-overflow in put_qpel_fallback when decoding file #241

Open leonzhao7 opened this issue on Dec 24, 2019 · 1 comment

leonzhao7 commented on Dec 24, 2019

stack-buffer-overflow in put_qpel_fallback when decoding file

I found some problems during fuzzing

Test Version

dev version, git clone https://github.com/strukturag/libde265

Test Environment

root@ubuntu:~# lsb_release -a
No LSB modules are available.
Distributor ID: Ubuntu
Description: Ubuntu 16.04.6 LTS
Release: 16.04
Codename: xenial

root@ubuntu:~# uname -a
Linux ubuntu 4.15.0-45-generic #4816.04.1-Ubuntu SMP Tue Jan 29 18:03:48 UTC 2019 x86_64 x86_64 x86_64 GNU/Linux

Test Configure

./configure
configure: --
configure: Building dec265 example: yes
configure: Building sherlock265 example: no
configure: Building encoder: yes
configure: --

Test Program

```
dec265 [infile]
```

Asan Output

```
root@ubuntu:~# ./dec265 libde265-put_qpel_fallback-stack_overflow.crash
WARNING: pps header invalid
WARNING: CTB outside of image area (concealing stream error...)
WARNING: pps header invalid
WARNING: end_of_sub_stream_one_bit not set to 1 when it should be
WARNING: pps header invalid
WARNING: end_of_sub_stream_one_bit not set to 1 when it should be
=================================================================
==91107==ERROR: AddressSanitizer: stack-buffer-overflow on address 0x7ffebaa90b7f at pc 0x00000043836d bp 0x7ffebaa8e510 sp 0x7ffebaa8e500
READ of size 2 at 0x7ffebaa90b7f thread T0
    #0 0x43836c in void put_qpel_fallback<unsigned short>(short*, long, unsigned short const*, long, int, int, short*, int, int, int) /root/src/libde265/libde265/fallback-
motion.cc:520
    #1 0x433c33 in put_qpel_1_3_fallback_16(short*, long, unsigned short const*, long, int, int, short*, int) /root/src/libde265/libde265/fallback-motion.cc:646
    #2 0x52c405 in acceleration_functions::put_hevc_qpel(short*, long, void const*, long, int, int, short*, int, int, int) const ../libde265/acceleration.h:338
    #3 0x52d7d6 in void mc_luma<unsigned char>(base_context const*, seq_parameter_set const*, int, int, int, int, short*, int, unsigned char const*, int, int, int, int)
/root/src/libde265/libde265/motion.cc:156
    #4 0x51f6f2 in generate_inter_prediction_samples(base_context*, slice_segment_header const*, de265_image*, int, int, int, int, int, int, int, PBMotion const*)
/root/src/libde265/libde265/motion.cc:376
    #5 0x52b8f9 in decode_prediction_unit(base_context*, slice_segment_header const*, de265_image*, PBMotionCoding const&, int, int, int, int, int, int, int, int)
/root/src/libde265/libde265/motion.cc:2107
    #6 0x478f4a in read_prediction_unit(thread_context*, int, int, int, int, int, int, int, int, int) /root/src/libde265/libde265/slice.cc:4137
    #7 0x47a704 in read_coding_unit(thread_context*, int, int, int, int, int) /root/src/libde265/libde265/slice.cc:4492
    #8 0x47b6fe in read_coding_quadtree(thread_context*, int, int, int, int, int) /root/src/libde265/libde265/slice.cc:4647
    #9 0x47338a in read_coding_tree_unit(thread_context*) /root/src/libde265/libde265/slice.cc:2861
    #10 0x47beb1 in decode_substream(thread_context*, bool, bool) /root/src/libde265/libde265/slice.cc:4736
    #11 0x47db9f in read_slice_segment_data(thread_context*) /root/src/libde265/libde265/slice.cc:5049
    #12 0x40bf17 in decoder_context::decode_slice_unit_sequential(image_unit*, slice_unit*) /root/src/libde265/libde265/decctx.cc:843
    #13 0x40c6d7 in decoder_context::decode_slice_unit_parallel(image_unit*, slice_unit*) /root/src/libde265/libde265/decctx.cc:945
    #14 0x40b589 in decoder_context::decode_some(bool*) /root/src/libde265/libde265/decctx.cc:730
    #15 0x40b2f2 in decoder_context::read_slice_NAL(bitreader&, NAL_unit*, nal_header&) /root/src/libde265/libde265/decctx.cc:688
    #16 0x40dbb3 in decoder_context::decode_NAL(NAL_unit*) /root/src/libde265/libde265/decctx.cc:1230
    #17 0x40e17b in decoder_context::decode(int*) /root/src/libde265/libde265/decctx.cc:1318
    #18 0x405a61 in de265_decode /root/src/libde265/libde265/de265.cc:346
    #19 0x404972 in main /root/src/libde265/dec265/dec265.cc:764
    #20 0x7fd98d83582f in __libc_start_main (/lib/x86_64-linux-gnu/libc.so.6+0x2082f)
    #21 0x402b28 in _start (/root/dec265+0x402b28)

Address 0x7ffebaa90b7f is located in stack of thread T0 at offset 9151 in frame
    #0 0x52cf34 in void mc_luma<unsigned char>(base_context const*, seq_parameter_set const*, int, int, int, int, short*, int, unsigned char const*, int, int, int, int)
/root/src/libde265/libde265/motion.cc:49

  This frame has 2 object(s):
    [32, 9120) 'mcbuffer'
    [9152, 14832) 'padbuf' <== Memory access at offset 9151 partially underflows this variable
HINT: this may be a false positive if your program uses some custom stack unwind mechanism or swapcontext
      (longjmp and C++ exceptions *are* supported)
SUMMARY: AddressSanitizer: stack-buffer-overflow /root/src/libde265/libde265/fallback-motion.cc:520 void put_qpel_fallback<unsigned short>(short*, long, unsigned short const*,
```

```
long, int, int, short*, int, int, int)
Shadow bytes around the buggy address:
  0x10005754a110: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x10005754a120: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x10005754a130: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x10005754a140: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x10005754a150: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
=>0x10005754a160: 00 00 00 00 00 00 00 00 00 00 00 00 f2 f2 f2[f2]
  0x10005754a170: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x10005754a180: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x10005754a190: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x10005754a1a0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x10005754a1b0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
Shadow byte legend (one shadow byte represents 8 application bytes):
  Addressable:           00
  Partially addressable: 01 02 03 04 05 06 07
  Heap left redzone:       fa
  Heap right redzone:      fb
  Freed heap region:       fd
  Stack left redzone:      f1
  Stack mid redzone:       f2
  Stack right redzone:     f3
  Stack partial redzone:   f4
  Stack after return:      f5
  Stack use after scope:   f8
  Global redzone:          f9
  Global init order:       f6
  Poisoned by user:        f7
  Container overflow:      fc
  Array cookie:            ac
  Intra object redzone:    bb
  ASan internal:           fe
==91107==ABORTING
```

POC file

libde265-put_qpel_fallback-stack_overflow.zip
libde265-put_qpel_fallback-stack_overflow2.zip
password: leon.zhao.7

CREDIT

Zhao Liang, Huawei Weiran Labs

coldtobi commented last week

According to Debian this is CVE-2020-21601

Assignees

No one assigned

Labels

None yet

Projects

None yet

Milestone

No milestone

Development

No branches or pull requests

2 participants