New issue

Jump to bottom

heap-buffer-overflow in put_weighted_pred_avg_16_fallback when decoding file #243

Open leonzhao7 opened this issue on Dec 24, 2019 · 2 comments

leonzhao7 commented on Dec 24, 2019

heap-buffer-overflow in put_weighted_pred_avg_16_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_weighted_pred_avg_16_fallback-heap_overflow.crash
WARNING: pps header invalid
WARNING: non-existing PPS referenced
=================================================================
==103499==ERROR: AddressSanitizer: heap-buffer-overflow on address 0x62a000005310 at pc 0x000000432cd8 bp 0x7ffe393c5a50 sp 0x7ffe393c5a40
WRITE of size 2 at 0x62a000005310 thread T0
    #0 0x432cd7 in put_weighted_pred_avg_16_fallback(unsigned short*, long, short const*, short const*, long, int, int, int) /root/src/libde265/libde265/fallback-motion.cc:246
    #1 0x52bc12 in acceleration_functions::put_weighted_pred_avg(void*, long, short const*, short const*, long, int, int, int) const ../libde265/acceleration.h:251
    #2 0x52085c in generate_inter_prediction_samples(base_context*, slice_segment_header const*, de265_image*, int, int, int, int, int, int, PBMotion const*)
/root/src/libde265/libde265/motion.cc:513
    #3 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
    #4 0x478f4a in read_prediction_unit(thread_context*, int, int, int, int, int, int, int, int, int) /root/src/libde265/libde265/slice.cc:4137
    #5 0x47a704 in read_coding_unit(thread_context*, int, int, int, int) /root/src/libde265/libde265/slice.cc:4492
    #6 0x47b6fe in read_coding_quadtree(thread_context*, int, int, int, int) /root/src/libde265/libde265/slice.cc:4647
    #7 0x47338a in read_coding_tree_unit(thread_context*) /root/src/libde265/libde265/slice.cc:2861
    #8 0x47beb1 in decode_substream(thread_context*, bool, bool) /root/src/libde265/libde265/slice.cc:4736
    #9 0x47db9f in read_slice_segment_data(thread_context*) /root/src/libde265/libde265/slice.cc:5049
    #10 0x40bf17 in decoder_context::decode_slice_unit_sequential(image_unit*, slice_unit*) /root/src/libde265/libde265/decctx.cc:843
    #11 0x40c6d7 in decoder_context::decode_slice_unit_parallel(image_unit*, slice_unit*) /root/src/libde265/libde265/decctx.cc:945
    #12 0x40b589 in decoder_context::decode_some(bool*) /root/src/libde265/libde265/decctx.cc:730
    #13 0x40b2f2 in decoder_context::read_slice_NAL(bitreader&, NAL_unit*, nal_header&) /root/src/libde265/libde265/decctx.cc:688
    #14 0x40dbb3 in decoder_context::decode_NAL(NAL_unit*) /root/src/libde265/libde265/decctx.cc:1230
    #15 0x40e17b in decoder_context::decode(int*) /root/src/libde265/libde265/decctx.cc:1318
    #16 0x405a61 in de265_decode /root/src/libde265/libde265/de265.cc:346
    #17 0x404972 in main /root/src/libde265/dec265/dec265.cc:764
    #18 0x7fa63f83582f in __libc_start_main (/lib/x86_64-linux-gnu/libc.so.6+0x2082f)
    #19 0x402b28 in _start (/root/dec265+0x402b28)

0x62a000005310 is located 0 bytes to the right of 20752-byte region [0x62a000000200,0x62a000005310)
allocated by thread T0 here:
    #0 0x7fa640736076 in __interceptor_posix_memalign (/usr/lib/x86_64-linux-gnu/libasan.so.2+0x99076)
    #1 0x43e00d in ALLOC_ALIGNED /root/src/libde265/libde265/image.cc:54
    #2 0x43e725 in de265_image_get_buffer /root/src/libde265/libde265/image.cc:132
    #3 0x440639 in de265_image::alloc_image(int, int, de265_chroma, std::shared_ptr<seq_parameter_set const>, bool, decoder_context*, long, void*, bool)
/root/src/libde265/libde265/image.cc:384
    #4 0x43afa4 in decoded_picture_buffer::new_image(std::shared_ptr<seq_parameter_set const>, decoder_context*, long, void*, bool) /root/src/libde265/libde265/dpb.cc:262
    #5 0x414467 in decoder_context::process_slice_segment_header(slice_segment_header*, de265_error*, long, nal_header*, void*) /root/src/libde265/libde265/decctx.cc:2012
    #6 0x40acad in decoder_context::read_slice_NAL(bitreader&, NAL_unit*, nal_header&) /root/src/libde265/libde265/decctx.cc:639
    #7 0x40dbb3 in decoder_context::decode_NAL(NAL_unit*) /root/src/libde265/libde265/decctx.cc:1230
    #8 0x40e17b in decoder_context::decode(int*) /root/src/libde265/libde265/decctx.cc:1318
    #9 0x405a61 in de265_decode /root/src/libde265/libde265/de265.cc:346
    #10 0x404972 in main /root/src/libde265/dec265/dec265.cc:764
    #11 0x7fa63f83582f in __libc_start_main (/lib/x86_64-linux-gnu/libc.so.6+0x2082f)

SUMMARY: AddressSanitizer: heap-buffer-overflow /root/src/libde265/libde265/fallback-motion.cc:246 put_weighted_pred_avg_16_fallback(unsigned short*, long, short const*, short
const*, long, int, int, int)
```

```
  Shadow bytes around the buggy address:
    0x0c547fff8a10: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
    0x0c547fff8a20: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
    0x0c547fff8a30: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
    0x0c547fff8a40: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
    0x0c547fff8a50: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  =>0x0c547fff8a60: 00 00[fa]fa fa fa fa fa fa fa fa fa fa fa fa
    0x0c547fff8a70: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
    0x0c547fff8a80: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
    0x0c547fff8a90: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
    0x0c547fff8aa0: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
    0x0c547fff8ab0: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  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
  ==103499==ABORTING
```

POC file

libde265-put_weighted_pred_avg_16_fallback-heap_overflow.zip
password: leon.zhao.7

CREDIT

Zhao Liang, Huawei Weiran Labs

farindk commented on Oct 17 — Contributor

I cannot reproduce this even going back to v1.0.3.
clang 14.0.0

ist199099 commented on Oct 20

This has been assigned CVE-2020-21600.

Assignees
No one assigned

Labels
None yet

Projects
None yet

Milestone
No milestone

Development
No branches or pull requests

3 participants