New issue

Jump to bottom

heap-buffer-overflow in mc_chroma when decoding file #238

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

leonzhao7 commented on Dec 24, 2019

heap-buffer-overflow in mc_chroma 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-mc_chroma-heap_overflow.crash
WARNING: CTB outside of image area (concealing stream error...)
WARNING: faulty reference picture list
WARNING: slice segment address invalid
WARNING: faulty reference picture list
=================================================================
==78714==ERROR: AddressSanitizer: heap-buffer-overflow on address 0x61b00001bf10 at pc 0x00000052e002 bp 0x7ffc932b5930 sp 0x7ffc932b5920
READ of size 2 at 0x61b00001bf10 thread T0
    #0 0x52e001 in void mc_chroma<unsigned short>(base_context const*, seq_parameter_set const*, int, int, int, int, short*, int, unsigned short const*, int, int, int, int)
/root/src/libde265/libde265/motion.cc:244
    #1 0x51f88a 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:382
    #2 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
    #3 0x478f4a in read_prediction_unit(thread_context*, int, int, int, int, int, int, int, int, int, int) /root/src/libde265/libde265/slice.cc:4137
    #4 0x47a704 in read_coding_unit(thread_context*, int, int, int, int, int) /root/src/libde265/libde265/slice.cc:4492
    #5 0x47b6fe in read_coding_quadtree(thread_context*, int, int, int, int) /root/src/libde265/libde265/slice.cc:4647
    #6 0x47338a in read_coding_tree_unit(thread_context*) /root/src/libde265/libde265/slice.cc:2861
    #7 0x47beb1 in decode_substream(thread_context*, bool, bool) /root/src/libde265/libde265/slice.cc:4736
    #8 0x47db9f in read_slice_segment_data(thread_context*) /root/src/libde265/libde265/slice.cc:5049
    #9 0x40bf17 in decoder_context::decode_slice_unit_sequential(image_unit*, slice_unit*) /root/src/libde265/libde265/decctx.cc:843
    #10 0x40c6d7 in decoder_context::decode_slice_unit_parallel(image_unit*, slice_unit*) /root/src/libde265/libde265/decctx.cc:945
    #11 0x40b589 in decoder_context::decode_some(bool*) /root/src/libde265/libde265/decctx.cc:730
    #12 0x40b2f2 in decoder_context::read_slice_NAL(bitreader&, NAL_unit*, nal_header&) /root/src/libde265/libde265/decctx.cc:688
    #13 0x40dbb3 in decoder_context::decode_NAL(NAL_unit*) /root/src/libde265/libde265/decctx.cc:1230
    #14 0x40e17b in decoder_context::decode(int*) /root/src/libde265/libde265/decctx.cc:1318
    #15 0x405a61 in de265_decode /root/src/libde265/libde265/de265.cc:346
    #16 0x404972 in main /root/src/libde265/dec265/dec265.cc:764
    #17 0x7f97d894282f in __libc_start_main (/lib/x86_64-linux-gnu/libc.so.6+0x2082f)
    #18 0x402b28 in _start (/root/dec265+0x402b28)

0x61b00001bf10 is located 0 bytes to the right of 1424-byte region [0x61b00001b980,0x61b00001bf10)
allocated by thread T0 here:
    #0 0x7f97d9843076 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 0x40ee8b in decoder_context::generate_unavailable_reference_picture(seq_parameter_set const*, int, bool) /root/src/libde265/libde265/decctx.cc:1418
    #6 0x411722 in decoder_context::process_reference_picture_set(slice_segment_header*) /root/src/libde265/libde265/decctx.cc:1648
    #7 0x414cc9 in decoder_context::process_slice_segment_header(slice_segment_header*, de265_error*, long, nal_header*, void*) /root/src/libde265/libde265/decctx.cc:2066
    #8 0x40acad in decoder_context::read_slice_NAL(bitreader&, NAL_unit*, nal_header&) /root/src/libde265/libde265/decctx.cc:639
    #9 0x40dbb3 in decoder_context::decode_NAL(NAL_unit*) /root/src/libde265/libde265/decctx.cc:1230
    #10 0x40e17b in decoder_context::decode(int*) /root/src/libde265/libde265/decctx.cc:1318
    #11 0x405a61 in de265_decode /root/src/libde265/libde265/de265.cc:346
    #12 0x404972 in main /root/src/libde265/dec265/dec265.cc:764
```

```
    #13 0x7f97d894282f in __libc_start_main (/lib/x86_64-linux-gnu/libc.so.6+0x2082f)

SUMMARY: AddressSanitizer: heap-buffer-overflow /root/src/libde265/libde265/motion.cc:244 void mc_chroma<unsigned short>(base_context const*, seq_parameter_set const*, int, int,
int, int, short*, int, unsigned short const*, int, int, int, int)
Shadow bytes around the buggy address:
  0x0c367fffb790: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x0c367fffb7a0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x0c367fffb7b0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x0c367fffb7c0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x0c367fffb7d0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
=>0x0c367fffb7e0: 00 00[fa]fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c367fffb7f0: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c367fffb800: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c367fffb810: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x0c367fffb820: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x0c367fffb830: 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
==78714==ABORTING
```

POC file

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

CREDIT

Zhao Liang, Huawei Weiran Labs

hardik05 commented on Apr 3, 2021

+1, looks like this is still not fixed. i also found this issue. can send POC if required.

ist199099 commented on Oct 20

@hardik05 Can you comment here linking to your POC? I will reproduce this until Saturday.

This was assigned CVE-2020-21597.

Assignees

No one assigned

Labels

None yet

Projects

None yet

Milestone

No milestone

Development

No branches or pull requests

3 participants