New issue

Jump to bottom

Heap-buffer-overflow in fallback-motion.cc in put_epel_16_fallback #347

Open FDU-Sec opened this issue on Oct 10 · 0 comments

FDU-Sec commented on Oct 10

Description

Heap-buffer-overflow (/libde265/build/libde265/liblibde265.so+0x1465fb) in put_epel_16_fallback(short*, long, unsigned short const*, long, int, int, int, int, short*, int)

Version

```
$ ./dec265 -h
 dec265  v1.0.8
--------------
usage: dec265 [options] videofile.bin
The video file must be a raw bitstream, or a stream with NAL units (option -n).

options:
  -q, --quiet       do not show decoded image
  -t, --threads N   set number of worker threads (0 - no threading)
  -c, --check-hash  perform hash check
  -n, --nal         input is a stream with 4-byte length prefixed NAL units
  -f, --frames N    set number of frames to process
  -o, --output      write YUV reconstruction
  -d, --dump        dump headers
  -0, --noaccel     do not use any accelerated code (SSE)
  -v, --verbose     increase verbosity level (up to 3 times)
  -L, --no-logging  disable logging
  -B, --write-bytestream FILENAME  write raw bytestream (from NAL input)
  -m, --measure YUV compute PSNRs relative to reference YUV
  -T, --highest-TID select highest temporal sublayer to decode
      --disable-deblocking   disable deblocking filter
      --disable-sao          disable sample-adaptive offset filter
  -h, --help        show help
```

Replay

```
git clone https://github.com/strukturag/libde265.git
cd libde265
mkdir build
cd build
cmake ../ -DCMAKE_CXX_FLAGS="-fsanitize=address"
make -j$(nproc)
./dec265/dec265 poc13
```

ASAN

```
WARNING: end_of_sub_stream_one_bit not set to 1 when it should be
WARNING: end_of_sub_stream_one_bit not set to 1 when it should be
=================================================================
==64370==ERROR: AddressSanitizer: heap-buffer-overflow on address 0x62b00001b510 at pc 0x7f47d023f5fc
READ of size 2 at 0x62b00001b510 thread T0
    #0 0x7f47d023f5fb in put_epel_16_fallback(short*, long, unsigned short const*, long, int, int, in
    #1 0x7f47d026ffe8 in acceleration_functions::put_hevc_epel(short*, long, void const*, long, int,
    #2 0x7f47d0271d75 in void mc_chroma<unsigned short>(base_context const*, seq_parameter_set const*
    #3 0x7f47d0262b2d in generate_inter_prediction_samples(base_context*, slice_segment_header const*
    #4 0x7f47d026f90f in decode_prediction_unit(base_context*, slice_segment_header const*, de265_ima
    #5 0x7f47d02aa7e3 in read_prediction_unit(thread_context*, int, int, int, int, int, int, int, int
    #6 0x7f47d02ac264 in read_coding_unit(thread_context*, int, int, int, int) (/libde265/build/libde
    #7 0x7f47d02ad250 in read_coding_quadtree(thread_context*, int, int, int, int) (/libde265/build/l
    #8 0x7f47d02a4726 in read_coding_tree_unit(thread_context*) (/libde265/build/libde265/liblibde265
    #9 0x7f47d02ad9ea in decode_substream(thread_context*, bool, bool) (/libde265/build/libde265/libl
    #10 0x7f47d02af70f in read_slice_segment_data(thread_context*) (/libde265/build/libde265/liblibde
    #11 0x7f47d020e6d2 in decoder_context::decode_slice_unit_sequential(image_unit*, slice_unit*) (/l
    #12 0x7f47d020eec1 in decoder_context::decode_slice_unit_parallel(image_unit*, slice_unit*) (/lib
    #13 0x7f47d020dc0f in decoder_context::decode_some(bool*) (/libde265/build/libde265/liblibde265.s
    #14 0x7f47d020d93d in decoder_context::read_slice_NAL(bitreader&, NAL_unit*, nal_header&) (/libde
    #15 0x7f47d021043e in decoder_context::decode_NAL(NAL_unit*) (/libde265/build/libde265/liblibde26
    #16 0x7f47d0210ab3 in decoder_context::decode(int*) (/libde265/build/libde265/liblibde265.so+0x11
    #17 0x7f47d01f7e95 in de265_decode (/libde265/build/libde265/liblibde265.so+0xfee95)
    #18 0x555f566e3bc9 in main (/libde265/build/dec265/dec265+0x6bc9)
    #19 0x7f47cfd29c86 in __libc_start_main (/lib/x86_64-linux-gnu/libc.so.6+0x21c86)
    #20 0x555f566e19b9 in _start (/libde265/build/dec265/dec265+0x49b9)

0x62b00001b510 is located 0 bytes to the right of 25360-byte region [0x62b000015200,0x62b00001b510)
allocated by thread T0 here:
    #0 0x7f47d0720790 in posix_memalign (/usr/lib/x86_64-linux-gnu/libasan.so.4+0xdf790)
    #1 0x7f47d02491cb in ALLOC_ALIGNED(unsigned long, unsigned long) (/libde265/build/libde265/liblib
    #2 0x7f47d024999d in de265_image_get_buffer(void*, de265_image_spec*, de265_image*, void*) (/libd
    #3 0x7f47d024bd1a in de265_image::alloc_image(int, int, de265_chroma, std::shared_ptr<seq_paramet
    #4 0x7f47d02300cc in decoded_picture_buffer::new_image(std::shared_ptr<seq_parameter_set const>,
    #5 0x7f47d02173ff in decoder_context::process_slice_segment_header(slice_segment_header*, de265_e
    #6 0x7f47d020d246 in decoder_context::read_slice_NAL(bitreader&, NAL_unit*, nal_header&) (/libde2
    #7 0x7f47d021043e in decoder_context::decode_NAL(NAL_unit*) (/libde265/build/libde265/liblibde265
    #8 0x7f47d0210ab3 in decoder_context::decode(int*) (/libde265/build/libde265/liblibde265.so+0x117
    #9 0x7f47d01f7e95 in de265_decode (/libde265/build/libde265/liblibde265.so+0xfee95)
    #10 0x555f566e3bc9 in main (/libde265/build/dec265/dec265+0x6bc9)
    #11 0x7f47cfd29c86 in __libc_start_main (/lib/x86_64-linux-gnu/libc.so.6+0x21c86)

SUMMARY: AddressSanitizer: heap-buffer-overflow (/libde265/build/libde265/liblibde265.so+0x1465fb) in
```

```
Shadow bytes around the buggy address:
  0x0c567fffb650: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x0c567fffb660: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x0c567fffb670: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x0c567fffb680: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x0c567fffb690: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
=>0x0c567fffb6a0: 00 00[fa]fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c567fffb6b0: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c567fffb6c0: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c567fffb6d0: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c567fffb6e0: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c567fffb6f0: 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
  Freed heap region:       fd
  Stack left redzone:      f1
  Stack mid redzone:       f2
  Stack right redzone:     f3
  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
  Left alloca redzone:     ca
  Right alloca redzone:    cb
==64370==ABORTING
```

POC

https://github.com/FDU-Sec/poc/blob/main/libde265/poc13

Environment

```
Ubuntu 18.04.5 LTS
Clang 10.0.1
gcc 7.5.0
```

Credit

Peng Deng (Fudan University)

Assignees

No one assigned

Labels

None yet

Projects

None yet

Milestone

No milestone

Development

No branches or pull requests

1 participant