Heap-buffer-overflow in fallback-motion.cc: put_unweighted_pred_16_fallback #348

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+0x145b6b) in put_unweighted_pred_16_fallback(unsigned short*, long, short const*, long, int, int, 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 poc14
```

ASAN

```
WARNING: end_of_sub_stream_one_bit not set to 1 when it should be
=================================================================
==52042==ERROR: AddressSanitizer: heap-buffer-overflow on address 0x62b000006640 at pc 0x7fcb9155bb6c
WRITE of size 2 at 0x62b000006640 thread T0
    #0 0x7fcb9155bb6b in put_unweighted_pred_16_fallback(unsigned short*, long, short const*, long, i
    #1 0x7fcb9158cce4 in acceleration_functions::put_unweighted_pred(void*, long, short const*, long,
    #2 0x7fcb91581740 in generate_inter_prediction_samples(base_context*, slice_segment_header const*
    #3 0x7fcb9158c90f in decode_prediction_unit(base_context*, slice_segment_header const*, de265_ima
    #4 0x7fcb915c77e3 in read_prediction_unit(thread_context*, int, int, int, int, int, int, int, int
    #5 0x7fcb915c9264 in read_coding_unit(thread_context*, int, int, int, int) (/libde265/build/libde
    #6 0x7fcb915ca250 in read_coding_quadtree(thread_context*, int, int, int, int) (/libde265/build/l
    #7 0x7fcb915ca091 in read_coding_quadtree(thread_context*, int, int, int, int) (/libde265/build/l
    #8 0x7fcb915c1726 in read_coding_tree_unit(thread_context*) (/libde265/build/libde265/liblibde265
    #9 0x7fcb915ca9ea in decode_substream(thread_context*, bool, bool) (/libde265/build/libde265/libl
    #10 0x7fcb915cc70f in read_slice_segment_data(thread_context*) (/libde265/build/libde265/liblibde
    #11 0x7fcb9152b6d2 in decoder_context::decode_slice_unit_sequential(image_unit*, slice_unit*) (/l
    #12 0x7fcb9152bec1 in decoder_context::decode_slice_unit_parallel(image_unit*, slice_unit*) (/lib
    #13 0x7fcb9152ac0f in decoder_context::decode_some(bool*) (/libde265/build/libde265/liblibde265.s
    #14 0x7fcb9152a93d in decoder_context::read_slice_NAL(bitreader&, NAL_unit*, nal_header&) (/libde
    #15 0x7fcb9152d43e in decoder_context::decode_NAL(NAL_unit*) (/libde265/build/libde265/liblibde26
    #16 0x7fcb9152dab3 in decoder_context::decode(int*) (/libde265/build/libde265/liblibde265.so+0x11
    #17 0x7fcb91514e95 in de265_decode (/libde265/build/libde265/liblibde265.so+0xfee95)
    #18 0x55d2d5b14bc9 in main (/libde265/build/dec265/dec265+0x6bc9)
    #19 0x7fcb91046c86 in __libc_start_main (/lib/x86_64-linux-gnu/libc.so.6+0x21c86)
    #20 0x55d2d5b129b9 in _start (/libde265/build/dec265/dec265+0x49b9)

0x62b000006640 is located 48 bytes to the right of 25616-byte region [0x62b000000200,0x62b000006610)
allocated by thread T0 here:
    #0 0x7fcb91a3d790 in posix_memalign (/usr/lib/x86_64-linux-gnu/libasan.so.4+0xdf790)
    #1 0x7fcb915661cb in ALLOC_ALIGNED(unsigned long, unsigned long) (/libde265/build/libde265/liblib
    #2 0x7fcb9156692a in de265_image_get_buffer(void*, de265_image_spec*, de265_image*, void*) (/libd
    #3 0x7fcb91568d1a in de265_image::alloc_image(int, int, de265_chroma, std::shared_ptr<seq_paramet
    #4 0x7fcb9154d0cc in decoded_picture_buffer::new_image(std::shared_ptr<seq_parameter_set const>,
    #5 0x7fcb915343ff in decoder_context::process_slice_segment_header(slice_segment_header*, de265_e
    #6 0x7fcb9152a246 in decoder_context::read_slice_NAL(bitreader&, NAL_unit*, nal_header&) (/libde2
    #7 0x7fcb9152d43e in decoder_context::decode_NAL(NAL_unit*) (/libde265/build/libde265/liblibde265
    #8 0x7fcb9152dab3 in decoder_context::decode(int*) (/libde265/build/libde265/liblibde265.so+0x117
    #9 0x7fcb91514e95 in de265_decode (/libde265/build/libde265/liblibde265.so+0xfee95)
    #10 0x55d2d5b14bc9 in main (/libde265/build/dec265/dec265+0x6bc9)
    #11 0x7fcb91046c86 in __libc_start_main (/lib/x86_64-linux-gnu/libc.so.6+0x21c86)

SUMMARY: AddressSanitizer: heap-buffer-overflow (/libde265/build/libde265/liblibde265.so+0x145b6b) in
Shadow bytes around the buggy address:
    0x0c567fff8c70: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
    0x0c567fff8c80: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
    0x0c567fff8c90: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
    0x0c567fff8ca0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
    0x0c567fff8cb0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  =>0x0c567fff8cc0: 00 00 fa fa fa fa fa fa[fa]fa fa fa fa fa fa fa
    0x0c567fff8cd0: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
    0x0c567fff8ce0: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
    0x0c567fff8cf0: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
    0x0c567fff8d00: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
    0x0c567fff8d10: 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
  ==52042==ABORTING
```

POC

https://github.com/FDU-Sec/poc/blob/main/libde265/poc14

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