stack-buffer-overflow in fallback-motion.cc when decoding file #301

Open dhbbb opened this issue on Jun 24, 2021 · 2 comments

dhbbb commented on Jun 24, 2021

Hello,
A stack-buffer-overflow has occurred when running program dec265
System info：
Ubuntu 20.04.1 : clang 10.0.0 , gcc 9.3.0

Dec265 v1.0.8

poc (4).zip

Verification steps：
1.Get the source code of libde265
2.Compile

```
cd libde265
mkdir build && cd build
cmake ../ -DCMAKE_CXX_COMPILER=clang++ -DCMAKE_CXX_FLAGS="fsanitize=address"
make -j 32
```

3.run dec265

```
./dec265 poc
```

asan info

```
=================================================================
==1262407==ERROR: AddressSanitizer: stack-buffer-overflow on address 0x7ffeacbd65e3 at pc 0x7ff9ff7de308 bp 0x7ffeacbd3f00 sp 0x7ffeacbd3ef0
READ of size 2 at 0x7ffeacbd65e3 thread T0
    #0 0x7ff9ff7de307 in void put_epel_hv_fallback<unsigned short>(short*, long, unsigned short const*, long, int, int, int, int, short*, int) /home/dh/sda3/AFLplusplus/libde265-
master/libde265-master-afl++/libde265/fallback-motion.cc:352
    #1 0x7ff9ff830067 in acceleration_functions::put_hevc_epel_hv(short*, long, void const*, long, int, int, int, int, short*, int) const /home/dh/sda3/AFLplusplus/libde265-
master/libde265-master-afl++/libde265/acceleration.h:328
    #2 0x7ff9ff830067 in void mc_chroma<unsigned char>(base_context const*, seq_parameter_set const*, int, int, int, int, short*, int, unsigned char const*, int, int, int, int)
/home/dh/sda3/AFLplusplus/libde265-master/libde265-master-afl++/libde265/motion.cc:254
    #3 0x7ff9ff8262ab in generate_inter_prediction_samples(base_context*, slice_segment_header const*, de265_image*, int, int, int, int, int, int, int, PBMotion const*)
/home/dh/sda3/AFLplusplus/libde265-master/libde265-master-afl++/libde265/motion.cc:388
    #4 0x7ff9ff828626 in decode_prediction_unit(base_context*, slice_segment_header const*, de265_image*, PBMotionCoding const&, int, int, int, int, int, int, int, int)
/home/dh/sda3/AFLplusplus/libde265-master/libde265-master-afl++/libde265/motion.cc:2107
    #5 0x7ff9ff89c8aa in read_coding_unit(thread_context*, int, int, int, int) /home/dh/sda3/AFLplusplus/libde265-master/libde265-master-afl++/libde265/slice.cc:4314
    #6 0x7ff9ff8a48f2 in read_coding_quadtree(thread_context*, int, int, int, int) /home/dh/sda3/AFLplusplus/libde265-master/libde265-master-afl++/libde265/slice.cc:4652
    #7 0x7ff9ff8a4e43 in read_coding_quadtree(thread_context*, int, int, int, int) /home/dh/sda3/AFLplusplus/libde265-master/libde265-master-afl++/libde265/slice.cc:4638
    #8 0x7ff9ff8a4ace in read_coding_quadtree(thread_context*, int, int, int, int) /home/dh/sda3/AFLplusplus/libde265-master/libde265-master-afl++/libde265/slice.cc:4645
    #9 0x7ff9ff8a4db9 in read_coding_quadtree(thread_context*, int, int, int, int) /home/dh/sda3/AFLplusplus/libde265-master/libde265-master-afl++/libde265/slice.cc:4641
    #10 0x7ff9ff8a6564 in decode_substream(thread_context*, bool, bool) /home/dh/sda3/AFLplusplus/libde265-master/libde265-master-afl++/libde265/slice.cc:4741
    #11 0x7ff9ff8a8ddb in read_slice_segment_data(thread_context*) /home/dh/sda3/AFLplusplus/libde265-master/libde265-master-afl++/libde265/slice.cc:5054
    #12 0x7ff9ff78dd75 in decoder_context::decode_slice_unit_sequential(image_unit*, slice_unit*) /home/dh/sda3/AFLplusplus/libde265-master/libde265-master-
afl++/libde265/decctx.cc:843
    #13 0x7ff9ff790c0f in decoder_context::decode_slice_unit_parallel(image_unit*, slice_unit*) /home/dh/sda3/AFLplusplus/libde265-master/libde265-master-
afl++/libde265/decctx.cc:945
    #14 0x7ff9ff791715 in decoder_context::decode_some(bool*) /home/dh/sda3/AFLplusplus/libde265-master/libde265-master-afl++/libde265/decctx.cc:730
    #15 0x7ff9ff7949bb in decoder_context::read_slice_NAL(bitreader&, NAL_unit*, nal_header&) /home/dh/sda3/AFLplusplus/libde265-master/libde265-master-afl++/libde265/decctx.cc:688
    #16 0x7ff9ff795839 in decoder_context::decode_NAL(NAL_unit*) /home/dh/sda3/AFLplusplus/libde265-master/libde265-master-afl++/libde265/decctx.cc:1230
    #17 0x7ff9ff796e1e in decoder_context::decode(int*) /home/dh/sda3/AFLplusplus/libde265-master/libde265-master-afl++/libde265/decctx.cc:1318
    #18 0x5573510028fd in main /home/dh/sda3/AFLplusplus/libde265-master/libde265-master-afl++/dec265/dec265.cc:764
    #19 0x7ff9ff2e50b2 in __libc_start_main (/lib/x86_64-linux-gnu/libc.so.6+0x270b2)
    #20 0x55735100576d in _start (/home/dh/sda3/AFLplusplus/libde265-master/libde265-master-afl++/out/dec265-afl+++0xa76d)

Address 0x7ffeacbd65e3 is located in stack of thread T0 at offset 9315 in frame
    #0 0x7ff9ff82e67f in void mc_chroma<unsigned char>(base_context const*, seq_parameter_set const*, int, int, int, int, short*, int, unsigned char const*, int, int, int, int)
/home/dh/sda3/AFLplusplus/libde265-master/libde265-master-afl++/libde265/motion.cc:174

  This frame has 2 object(s):
    [32, 9120) 'mcbuffer' (line 200)
    [9392, 14752) 'padbuf' (line 222) <== Memory access at offset 9315 underflows this variable
HINT: this may be a false positive if your program uses some custom stack unwind mechanism, swapcontext or vfork
      (longjmp and C++ exceptions *are* supported)
SUMMARY: AddressSanitizer: stack-buffer-overflow /home/dh/sda3/AFLplusplus/libde265-master/libde265-master-afl++/libde265/fallback-motion.cc:352 in void
put_epel_hv_fallback<unsigned short>(short*, long, unsigned short const*, long, int, int, int, int, short*, int)
Shadow bytes around the buggy address:
  0x100055972c60: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x100055972c70: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x100055972c80: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x100055972c90: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x100055972ca0: 00 00 00 00 f2 f2 f2 f2 f2 f2 f2 f2 f2 f2 f2 f2
=>0x100055972cb0: f2 f2 f2 f2 f2 f2 f2 f2 f2 f2 f2 f2[f2]f2 f2 f2
  0x100055972cc0: f2 f2 f2 f2 f2 f2 00 00 00 00 00 00 00 00 00 00
  0x100055972cd0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x100055972ce0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x100055972cf0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x100055972d00: 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
  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
    Shadow gap:              cc
  ==1262407==ABORTING
```

stevebeattie commented on Jan 12

This issue was assigned CVE-2021-36410.

farindk added a commit that referenced this issue on Apr 5

`fix MC with HDR chroma, but SDR luma (#301)` ✕ `697aa4f`

farindk commented on Apr 5 Contributor

Thank you.
Please confirm that the issue is resolved with the above change.

Assignees

No one assigned

Labels

None yet

Projects

None yet

Milestone

No milestone

Development

No branches or pull requests

3 participants