global buffer overflow in decode_CABAC_bit when decoding file #236

Open leonzhao7 opened this issue on Dec 24, 2019 · 1 comment

leonzhao7 commented on Dec 24, 2019

global buffer overflow in decode_CABAC_bit 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-decode_CABAC_bit-overflow.crash
WARNING: CTB outside of image area (concealing stream error...)
WARNING: end_of_sub_stream_one_bit not set to 1 when it should be
WARNING: slice header invalid
=================================================================
==58539==ERROR: AddressSanitizer: global-buffer-overflow on address 0x00000054625f at pc 0x0000004fc1cf bp 0x7fffa287c990 sp 0x7fffa287c980
READ of size 1 at 0x00000054625f thread T0
    #0 0x4fc1ce in decode_CABAC_bit(CABAC_decoder*, context_model*) /root/src/libde265/libde265/cabac.cc:180
    #1 0x46fca1 in decode_cu_skip_flag /root/src/libde265/libde265/slice.cc:1679
    #2 0x4797c7 in read_coding_unit(thread_context*, int, int, int, int) /root/src/libde265/libde265/slice.cc:4289
    #3 0x47b6fe in read_coding_quadtree(thread_context*, int, int, int, int) /root/src/libde265/libde265/slice.cc:4647
    #4 0x47b53f in read_coding_quadtree(thread_context*, int, int, int, int) /root/src/libde265/libde265/slice.cc:4630
    #5 0x47338a in read_coding_tree_unit(thread_context*) /root/src/libde265/libde265/slice.cc:2861
    #6 0x47beb1 in decode_substream(thread_context*, bool, bool) /root/src/libde265/libde265/slice.cc:4736
    #7 0x47db9f in read_slice_segment_data(thread_context*) /root/src/libde265/libde265/slice.cc:5049
    #8 0x40bf17 in decoder_context::decode_slice_unit_sequential(image_unit*, slice_unit*) /root/src/libde265/libde265/decctx.cc:843
    #9 0x40c6d7 in decoder_context::decode_slice_unit_parallel(image_unit*, slice_unit*) /root/src/libde265/libde265/decctx.cc:945
    #10 0x40b589 in decoder_context::decode_some(bool*) /root/src/libde265/libde265/decctx.cc:730
    #11 0x40b2f2 in decoder_context::read_slice_NAL(bitreader&, NAL_unit*, nal_header&) /root/src/libde265/libde265/decctx.cc:688
    #12 0x40dbb3 in decoder_context::decode_NAL(NAL_unit*) /root/src/libde265/libde265/decctx.cc:1230
    #13 0x40e17b in decoder_context::decode(int*) /root/src/libde265/libde265/decctx.cc:1318
    #14 0x405a61 in de265_decode /root/src/libde265/libde265/de265.cc:346
    #15 0x404972 in main /root/src/libde265/dec265/dec265.cc:764
    #16 0x7f83f76d982f in __libc_start_main (/lib/x86_64-linux-gnu/libc.so.6+0x2082f)
    #17 0x402b28 in _start (/root/dec265+0x402b28)

0x00000054625f is located 31 bytes to the right of global variable 'next_state_MPS' defined in 'cabac.cc:112:22' (0x546200) of size 64
0x00000054625f is located 1 bytes to the left of global variable 'next_state_LPS' defined in 'cabac.cc:120:22' (0x546260) of size 64
SUMMARY: AddressSanitizer: global-buffer-overflow /root/src/libde265/libde265/cabac.cc:180 decode_CABAC_bit(CABAC_decoder*, context_model*)
Shadow bytes around the buggy address:
  0x0000800a0bf0: 00 00 00 01 f9 f9 f9 f9 00 00 00 00 00 02 f9 f9
  0x0000800a0c00: f9 f9 f9 f9 00 00 00 00 00 00 00 00 01 f9 f9 f9
  0x0000800a0c10: f9 f9 f9 f9 00 00 00 00 00 00 00 00 00 00 00 00
  0x0000800a0c20: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x0000800a0c30: 00 00 00 00 f9 f9 f9 f9 00 00 00 00 f9 f9 f9 f9
=>0x0000800a0c40: 00 00 00 00 00 00 00 00 f9 f9 f9 f9[f9]00 00 00
  0x0000800a0c50: 00 00 00 00 f9 f9 f9 f9 00 01 f9 f9 f9 f9 f9 f9
  0x0000800a0c60: 00 04 f9 f9 f9 f9 f9 f9 00 00 00 f9 f9 f9 f9 f9
  0x0000800a0c70: 00 01 f9 f9 f9 f9 f9 f9 00 00 00 00 00 00 00 00
  0x0000800a0c80: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x0000800a0c90: 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
  ==58539==ABORTING
```

POC file

libde265-decode_CABAC_bit-overflow.zip
password: leon.zhao.7

CREDIT

Zhao Liang, Huawei Weiran Labs

ist199099 commented on Oct 20 • edited

This was assigned CVE-2020-21596.

Assignees

No one assigned

Labels

None yet

Projects

None yet

Milestone

No milestone

Development

No branches or pull requests

2 participants