New issue

Jump to bottom

Heap-buffer-overflow in color.c:379:42 in sycc420_to_rgb #1347

Open **yuawn** opened this issue on Apr 29, 2021 · 18 comments · May be fixed by #1362

yuawn commented on Apr 29, 2021 • edited

Hi,
I found a vulnerability in current master 0bda718, and I also reproduced it on latest released version v2.4.0.

Crash Summary
A heap-buffer-overflow in color.c:379:42 in sycc420_to_rgb, it can lead to heap-based buffer overflow via a crafted `.j2k` file when decompress it.

Crash Analysis
There is insufficient validation of `*cb`.

openjpeg/src/bin/common/color.c
Lines 375 to 381 in `0bda718`

```
375            ++cb;
376            ++cr;
377        }
378        if (j < maxw) {
379            sycc_to_rgb(offset, upb, *y, *cb, *cr, r, g, b);
380        }
381    }
```

PoC:
poc.j2k.gz

To reproduce (x86-64 Ubuntu 20.04.2 with gcc 9.3.0):

```
CFLAGS='-g -fsanitize=address' cmake .. -DCMAKE_BUILD_TYPE=Release
make

./bin/opj_decompress -i ./poc.j2k -o out.png
```

ASAN report:

```
=================================================================
==2371124==ERROR: AddressSanitizer: heap-buffer-overflow on address 0x612000000760 at pc 0x0000004f278c bp 0x7ffd11a3eca0 sp 0x7ffd11a3ec98
READ of size 4 at 0x612000000760 thread T0
    #0 0x4f278b in sycc420_to_rgb /home/yuawn/fuzz-targets/openjpeg/reproduce/openjpeg/src/bin/common/color.c:379:42
    #1 0x4f278b in color_sycc_to_rgb /home/yuawn/fuzz-targets/openjpeg/reproduce/openjpeg/src/bin/common/color.c:416:9
    #2 0x4cb136 in main /home/yuawn/fuzz-targets/openjpeg/reproduce/openjpeg/src/bin/jp2/opj_decompress.c:1589:13
    #3 0x7f653bef10b2 in __libc_start_main /build/glibc-YbNSs7/glibc-2.31/csu/../csu/libc-start.c:308:16
    #4 0x41d4fd in _start (/home/yuawn/fuzz-targets/openjpeg/reproduce/openjpeg/build/bin/opj_decompress+0x41d4fd)

0x612000000760 is located 0 bytes to the right of 288-byte region [0x612000000640,0x612000000760)
allocated by thread T0 here:
    #0 0x498027 in posix_memalign (/home/yuawn/fuzz-targets/openjpeg/reproduce/openjpeg/build/bin/opj_decompress+0x498027)
    #1 0x7f653c38aa5f in opj_aligned_alloc_n /home/yuawn/fuzz-targets/openjpeg/reproduce/openjpeg/src/lib/openjp2/opj_malloc.c:61:9
    #2 0x7f653c38aa5f in opj_aligned_malloc /home/yuawn/fuzz-targets/openjpeg/reproduce/openjpeg/src/lib/openjp2/opj_malloc.c:209:12
    #3 0x7f653c37c257 in opj_alloc_tile_component_data /home/yuawn/fuzz-targets/openjpeg/reproduce/openjpeg/src/lib/openjp2/tcd.c:697:39
    #4 0x7f653c37c257 in opj_tcd_decode_tile /home/yuawn/fuzz-targets/openjpeg/reproduce/openjpeg/src/lib/openjp2/tcd.c:1561:18
    #5 0x7f653c2d6131 in opj_j2k_decode_tile /home/yuawn/fuzz-targets/openjpeg/reproduce/openjpeg/src/lib/openjp2/j2k.c:9727:11
    #6 0x7f653c2f275e in opj_j2k_decode_tiles /home/yuawn/fuzz-targets/openjpeg/reproduce/openjpeg/src/lib/openjp2/j2k.c:11568:15
    #7 0x7f653c2dba43 in opj_j2k_exec /home/yuawn/fuzz-targets/openjpeg/reproduce/openjpeg/src/lib/openjp2/j2k.c:8871:33
    #8 0x7f653c2dba43 in opj_j2k_decode /home/yuawn/fuzz-targets/openjpeg/reproduce/openjpeg/src/lib/openjp2/j2k.c:11871:11

SUMMARY: AddressSanitizer: heap-buffer-overflow /home/yuawn/fuzz-targets/openjpeg/reproduce/openjpeg/src/bin/common/color.c:379:42 in sycc420_to_rgb
Shadow bytes around the buggy address:
  0x0c247fff8090: fa fa fa fa fa fa fa fa 00 00 00 00 00 00 00 00
  0x0c247fff80a0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x0c247fff80b0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x0c247fff80c0: fa fa fa fa fa fa fa fa 00 00 00 00 00 00 00 00
  0x0c247fff80d0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
=>0x0c247fff80e0: 00 00 00 00 00 00 00 00 00 00 00 00[fa]fa fa fa
  0x0c247fff80f0: fa fa fa fa fa fa fa fa 00 00 00 00 00 00 00 00
  0x0c247fff8100: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x0c247fff8110: 00 00 00 00 00 00 00 00 00 00 00 00 fa fa fa fa
  0x0c247fff8120: fa fa fa fa fa fa fa fa 00 00 00 00 00 00 00 00
  0x0c247fff8130: 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
==2371124==ABORTING
```

👀 1

szukw000 commented on May 4, 2021 Contributor

@yuawn ,
I use the library from 10.01.2021.

szukw000: opj_decompress -i poc.j2k -o poc.j2k.png

[INFO] Start to read j2k main header (0).
[ERROR] Unknown progression order in COD marker
[WARNING] Unknown marker
[ERROR] Unknown progression order in COD marker
[WARNING] Unknown marker
[WARNING] Unknown marker
[WARNING] Unknown marker
[ERROR] Unknown progression order in COD marker
[WARNING] Unknown marker
[WARNING] Unknown marker
[WARNING] Unknown marker
[INFO] Main header has been correctly decoded.
[INFO] No decoded area parameters, set the decoded area to the whole image
[INFO] Psot value of the current tile-part is equal to zero, we assuming it is the last tile-part of the codestream.
[INFO] Header of tile 1 / 1 has been read.
[INFO] Tile 1/1 has been decoded.
[INFO] Image data has been updated with tile 1.

imagetopng: All components shall have the same subsampling, same bit depth, same sign.
Aborting
[ERROR] Error generating png file. Outfile poc.j2k.png not generated

winfried

szukw000 commented on May 4, 2021 Contributor

@yuawn ,
using a simple reader read_j2k:

```
NAME(/tmp/1347-poc.j2k)
LENG(1307)

ENTER read_jp2c
[0]marker(0xff4f)
    soc len(0)
[2]marker(0xff51)
    siz len(47)
    capabilities(11776)[extended: 0]
    x(7 : 32) y(7 : 19)
    xt(0 : 73760) yt(0 : 218793738)
    IMAGE w(25) h(12) TILE w(73760) h(218793738)
    nr_components(3)
      component[0] signed(1) prec(8) hsep(1) vsep(1)
      component[1] signed(0) prec(9) hsep(2) vsep(2)
      component[2] signed(0) prec(3) hsep(2) vsep(2)
[51]marker(0xff52)

    read_cod

             max_len 12
          prog_order 128
           nr_layers 38552
multi_comp_transform 0
Scod 0
       entropy_coder 0
      use_sop_marker 0
      use_eph_marker 0
      num_resolutions 1
     code_block_width 0
    code_block_height 0
     code_block_style 0
       transformation 0 (9-7 irreversible)

    [0]precinct_w 15
    [0]precinct_h 15


    cod len(12)

[65]marker(0xffff)
test_marker: type(0xffff) prefix(0xff) suffix(0xff)
 I :MARKER 0xffff is unknown.
EXIT read_jp2c
    end - s ==> -27531
EXIT with end - s ==> 0 (DEC:0)
```

winfried

yuawn commented on May 4, 2021 • edited Author

@szukw000,
you need to build it with address sanitizer to detect the bug.

yuawn commented on May 4, 2021 Author

I also reproduced it on released version 2.3.1 released on Apr 2, 2019.
This bug affects released versions 2.3.1 ~ 2.4.0.

CityOfLight77 commented on May 5, 2021

@yuawn I try to build openjpeg with AFL but got error it can't find clang... already install it beforehand.
Mind to know how you build openjpeg with AFL?

yuawn commented on May 5, 2021 • edited Author

@CityOfLight77 there is no need to build it with AFL.

Both of GCC and Clang supports ASAN, just build it as I said above:

```
CFLAGS='-g -fsanitize=address' cmake .. -DCMAKE_BUILD_TYPE=Release
make
```

I reproduced this bug with gcc and clang on the versions from 2.3.1 to current master.

szukw000 commented on May 6, 2021 Contributor

@yuawn ,
I added:

```
    CFLAGS='-g -fsanitize=address'
```

opj_decompress -i /tmp/1347-poc.j2k -o 1347-poc.j2k.png

```
[INFO] Start to read j2k main header (0).
[ERROR] Unknown progression order in COD marker
[WARNING] Unknown marker
[ERROR] Unknown progression order in COD marker
[WARNING] Unknown marker
[WARNING] Unknown marker
[WARNING] Unknown marker
[ERROR] Unknown progression order in COD marker
[WARNING] Unknown marker
[WARNING] Unknown marker
[WARNING] Unknown marker
[INFO] Main header has been correctly decoded.
[INFO] No decoded area parameters, set the decoded area to the whole image
[INFO] Psot value of the current tile-part is equal to zero, we assuming it is the last tile-part of the codestream.
[INFO] Header of tile 1 / 1 has been read.
[INFO] Tile 1/1 has been decoded.
[INFO] Image data has been updated with tile 1.

imagetopng: All components shall have the same subsampling, same bit depth, same sign.
        Aborting
[ERROR] Error generating png file. Outfile 1347-poc.j2k.png not generated
```

By the way: I use gcc (GCC) 10.3.0.

winfried

yuawn commented on May 6, 2021 • edited Author

Hi **@szukw000**,
It seems like you didn't compile it with ASAN successfully, where did you add the compiler flags?

I can confirm that the following script can reproduce the bug successfully:

```
git clone https://github.com/uclouvain/openjpeg.git
cd openjpeg
git checkout v2.4.0

mkdir build
cd build

CFLAGS='-g -fsanitize=address' cmake .. -DCMAKE_BUILD_TYPE=Release
make

wget https://github.com/uclouvain/openjpeg/files/6402272/poc.j2k.gz
gunzip poc.j2k.gz

./bin/opj_decompress -i ./poc.j2k -o ./out.png
```

```
$ gcc --version
gcc (Ubuntu 10.2.0-5ubuntu1~20.04) 10.2.0

$ md5sum poc.j2k
c85153c022a7469d865a5a0b5e2781f8  poc.j2k
```

msabwat added a commit to msabwat/openjpeg that referenced this issue on May 6, 2021

`fix heap buffer overflow uclouvain#1347` f4cb033

ValZapod commented on May 17, 2021 • edited msabwat@ `f4cb033` will fix it, I hope. BTW, who knows why ffmpeg and openjpeg (thay is native decoder and libopenjpeg) are not bitperfect for sYCC stuff?

BECAUSE lossy jpeg 2000 is notr garanteered to be decoded the same way!

StayPirate commented on Jun 8, 2021

@rouault any idea if `f4cb033` will be merged or if an official ptach will be released instead?

rouault commented on Jun 8, 2021 Collaborator

> **@rouault** any idea if f4cb033 will be merged or if an official ptach will be released instead?

@msabwat can you issue a pull request with your proposed fix ?

msabwat commented on Jun 8, 2021 Contributor

> > **@rouault** any idea if f4cb033 will be merged or if an official ptach will be released instead?
>
> **@msabwat** can you issue a pull request with your proposed fix ?

Sure !

msabwat linked a pull request on Jun 9, 2021 that will close this issue

Draft: common: fix sycc420_to_rgb buffer overflow #1362

Open

StayPirate commented on Jun 10, 2021

This issue got assigned CVE-2021-3575. **@msabwat** would be worthy if you can add this CVE ID to your commit message.

yuawn mentioned this issue on Jun 10, 2021

heap-buffer-overflow in function sycc420_to_rgb() at openjpeg/src/bin/common/color.c:379 #1363

Open

ajakk commented on Jan 24

> This issue got assigned CVE-2021-3575. **@msabwat** would be worthy if you can add this CVE ID to your commit message.

Did you request it? Still seems reserved, so should be safe to make public now, right?

nanonyme commented on Apr 15

Any chance getting canonical fix merged? This is now public as severity 6.8 arbitrary code execution bug.

StayPirate commented on Apr 28

any update here?

ZaquL commented on May 16

On the sample kdu_jp2info.exe of kakadu v8.0.5 warns:

Kakadu Core Warning:
SIZ marker segment's Rsiz word must have bits 12 and 13 equal to 0 unless the
Part-2 flag (bit-15) is set.

So this is technically part 2 jpeg 2000 that is recognized by ffmpeg's native jpeg2000 decoder as yuv420p9, so 9 bits but it does not open it. All of that is not supported in libopenjpeg... So what should be done you should reject such files. For example:
`[jpeg2000 @ 0000024f436973c0] Missing EOC Marker.`

Leave ycbcr code alone.

❤️ 1

nanonyme commented on May 22

@ZaquL I guess that would explain why prior attempts to fix resulted in minor corruption.

👍 2

Assignees

No one assigned

Labels

None yet

Projects

None yet

Milestone

No milestone

Development

Successfully merging a pull request may close this issue.

Draft: common: fix sycc420_to_rgb buffer overflow
msabwat/openjpeg

10 participants