Starred by 8 users

Owner:	wtc@google.com
CC:	----
Status:	Fixed *(Closed)*
Components:	----
Modified:	Apr 13, 2021

Type-Defect
Priority-Medium
Hotlist-AOM-OKR

Issue 2913: global-buffer-overflow in av1/encoder/partition_search.h:63

Reported by zodf0...@gmail.com on Wed, Dec 23, 2020, 11:37 PM EST

What version / commit were you testing with?
commit a5d214

What steps will reproduce the problem?
1. ./aomenc --i422 --monochrome -h 10 -w 10 -o /dev/null ./poc4

What is the expected output?

This is ASAN report:

```
➜ Yuan-fuzz ~/aom/build/aomenc --i422 --monochrome -h 10 -w 10 -o /dev/null ./poc4
Warning: automatically updating to profile 2 to match input format.
Pass 1/2 frame   10/11      2288B    1830b/f   54900b/s   13898 us (719.53 fps)
Warning: automatically updating to profile 2 to match input format.
Pass 2/2 frame   10/0          0B   16530 us 604.96 fps [ETA  unknown] =================================================================
==11116==ERROR: AddressSanitizer: global-buffer-overflow on address 0x5595dc9c46df at pc 0x5595dbef2b29 bp 0x7fff4d70efc0 sp 0x7fff4d70efb0
READ of size 1 at 0x5595dc9c46df thread T0
    #0 0x5595dbef2b28 in update_cb_offsets /home/yuan/afl-target/aom/av1/encoder/partition_search.h:63
    #1 0x5595dbef2b28 in encode_b /home/yuan/afl-target/aom/av1/encoder/partition_search.c:1178
    #2 0x5595dbef414b in encode_sb /home/yuan/afl-target/aom/av1/encoder/partition_search.c:1402
    #3 0x5595dbef4c17 in encode_sb /home/yuan/afl-target/aom/av1/encoder/partition_search.c:1371
    #4 0x5595dbef4c17 in encode_sb /home/yuan/afl-target/aom/av1/encoder/partition_search.c:1371
    #5 0x5595dbef4c17 in encode_sb /home/yuan/afl-target/aom/av1/encoder/partition_search.c:1371
    #6 0x5595dbef4c17 in encode_sb /home/yuan/afl-target/aom/av1/encoder/partition_search.c:1371
    #7 0x5595dbf2df6e in av1_rd_pick_partition /home/yuan/afl-target/aom/av1/encoder/partition_search.c:3797
    #8 0x5595dbdad867 in encode_rd_sb /home/yuan/afl-target/aom/av1/encoder/encodeframe.c:710
    #9 0x5595dbdb7ae9 in encode_sb_row /home/yuan/afl-target/aom/av1/encoder/encodeframe.c:848
    #10 0x5595dbdb7ae9 in av1_encode_sb_row /home/yuan/afl-target/aom/av1/encoder/encodeframe.c:957
    #11 0x5595dbdba5a4 in av1_encode_tile /home/yuan/afl-target/aom/av1/encoder/encodeframe.c:997
    #12 0x5595dbdc2c3d in encode_tiles /home/yuan/afl-target/aom/av1/encoder/encodeframe.c:1027
    #13 0x5595dbdc2c3d in encode_frame_internal /home/yuan/afl-target/aom/av1/encoder/encodeframe.c:1430
    #14 0x5595dbdc89d9 in av1_encode_frame /home/yuan/afl-target/aom/av1/encoder/encodeframe.c:1598
    #15 0x5595da7d566b in encode_with_recode_loop /home/yuan/afl-target/aom/av1/encoder/encoder.c:2508
    #16 0x5595da7d566b in encode_with_recode_loop_and_filter /home/yuan/afl-target/aom/av1/encoder/encoder.c:2612
    #17 0x5595da7f6398 in encode_frame_to_data_rate /home/yuan/afl-target/aom/av1/encoder/encoder.c:3097
    #18 0x5595da83250d in av1_encode /home/yuan/afl-target/aom/av1/encoder/encoder.c:3231
    #19 0x5595dbe5c365 in denoise_and_encode /home/yuan/afl-target/aom/av1/encoder/encode_strategy.c:974
    #20 0x5595dbe5c365 in av1_encode_strategy /home/yuan/afl-target/aom/av1/encoder/encode_strategy.c:1360
    #21 0x5595da8347d4 in av1_get_compressed_data /home/yuan/afl-target/aom/av1/encoder/encoder.c:3512
    #22 0x5595da65aaec in encoder_encode /home/yuan/afl-target/aom/av1/av1_cx_iface.c:2313
    #23 0x5595da50062c in aom_codec_encode /home/yuan/afl-target/aom/aom/src/aom_encoder.c:155
    #24 0x5595da3150e1 in encode_frame /home/yuan/afl-target/aom/apps/aomenc.c:2064
    #25 0x5595da2f452c in main /home/yuan/afl-target/aom/apps/aomenc.c:2719
    #26 0x7f332e653bf6 in __libc_start_main (/lib/x86_64-linux-gnu/libc.so.6+0x21bf6)
    #27 0x5595da309739 in _start (/home/yuan/afl-target/aom/build/aomenc+0x93739)

0x5595dc9c46df is located 1 bytes to the left of global variable 'mi_size_wide_log2' defined in '/home/yuan/afl-target/aom/av1/common/common_data.h:25:22'
(0x5595dc9c46e0) of size 22
0x5595dc9c46df is located 41 bytes to the right of global variable 'mi_size_high_log2' defined in '/home/yuan/afl-target/aom/av1/common/common_data.h:29:22'
(0x5595dc9c46a0) of size 22
SUMMARY: AddressSanitizer: global-buffer-overflow /home/yuan/afl-target/aom/av1/encoder/partition_search.h:63 in update_cb_offsets
Shadow bytes around the buggy address:
  0x0ab33b930880: 00 04 f9 f9 f9 f9 f9 f9 00 00 03 f9 f9 f9 f9 f9
  0x0ab33b930890: 00 00 06 f9 f9 f9 f9 f9 00 00 00 00 00 00 00 04
  0x0ab33b9308a0: f9 f9 f9 f9 00 00 06 f9 f9 f9 f9 f9 00 00 06 f9
  0x0ab33b9308b0: f9 f9 f9 f9 00 00 06 f9 f9 f9 f9 f9 00 00 06 f9
  0x0ab33b9308c0: f9 f9 f9 f9 00 00 06 f9 f9 f9 f9 f9 00 00 06 f9
=>0x0ab33b9308d0: f9 f9 f9 f9 00 00 06 f9 f9 f9 f9[f9]00 00 06 f9
  0x0ab33b9308e0: f9 f9 f9 f9 00 00 00 00 00 00 00 00 00 00 00 00
  0x0ab33b9308f0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x0ab33b930900: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x0ab33b930910: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x0ab33b930920: 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
==11116==ABORTING

```

poc4
2.1 KB View Download

Comment 1 by zodf0...@gmail.com on Tue, Dec 29, 2020, 2:17 AM EST

This is environment:
OS : ubuntu 18.04.3
kernel : gnu/linux 5.4.0-52-generic
CPU :   Intel(R) Core(TM) i7-10700 CPU @ 2.90GHz
compiler : gcc version 7.5.0

This is How I build
1. git clone https://aomedia.googlesource.com/aom
2. cd aom/build
3. cmake ..

Comment 2 by ryoh@chromium.org on Tue, Dec 29, 2020, 7:46 AM EST

Hi. I'm using libaom from cavif(https://github.com/link-u/cavif), AVIF encoder.

I observed that assertion error is reported when converting YUV422 images only on Linux.

> cavif --enable-full-color-range -i hato.png -o hato.profile2.8bpc.yuv422.monochrome.avif --tune psnr --profile 2 --bit-depth 8 --pix-fmt yuv422 --monochrome --cpu-used 0 --rate-control q --crf 18
> cavif: /github/workspace/external/libaom/av1/encoder/partition_search.c:1170: encode_b: Assertion `x->cb_offset[PLANE_TYPE_UV] < ((1 << num_pels_log2_lookup[cpi->common.seq_params.sb_size]) >> (subsampling_x + subsampling_y))' failed.

https://github.com/link-u/avif-sample-images/runs/1621679812?

I think this Assertion error might be related to this issue.

I'm compiling libaom under those condition:
 - commit: 563fbff
- Ubuntu 18.04 with gcc-8, Ubuntu 20.04 with gcc-9 and gcc-10
- Binaries are built by github actions:
https://github.com/link-u/cavif/actions/runs/450715597

Curiously, I also find these assertion error cannot be observed in Windows (Mingw64, gcc-10).

Comment 3 by jz...@google.com on Mon, Jan 11, 2021, 1:53 PM EST

Status: Assigned (was: New)
Owner: wtc@google.com

Comment 4 Deleted

Comment 5 by jz...@google.com on Mon, Feb 22, 2021, 2:46 PM EST

Labels: Hotlist-AOM-OKR

Comment 6 by wtc@google.com on Tue, Apr 6, 2021, 9:02 PM EDT

Status: Started (was: Assigned)

zodf0055980: Thank you very much for the bug report. I apologize for the long delay in responding to this bug report.

ryoh: The build logs you mentioned in comment #2 have expired because it took me so long to look into this bug. I am so sorry!

I can reproduce the ASan error with commit a5d214. In the current HEAD, there is a different ASan error. Fortunately, the fix I came up with works for both commit a5d214 and the current HEAD.

I think this bug is an example of libaom's general weakness in monochrome image support -- libaom still does not have an image format type for monochrome (YUV 4:0:0) images. So we need to pass a non-monochrome input image to the libaom encoder and set the 'monochrome' flag to 1. If the non-monochrome input image has the YUV

4:2:2 format, then we may run into problems like this bug.

The poc4 file is apparently a webm file. Since it is not a y4m or ivf file, aomenc treats it as the "raw" format.

The ASan error in commit a5d214 occurs during the fourth call to update_cb_offsets(). Here is a gdb session that shows the arguments passed to the first four update_cb_offsets() calls and the single stepping in the fourth call:

```
Breakpoint 1, update_cb_offsets (x=0x7ffff2e0a820, bsize=BLOCK_16X16, subsampling_x=1, subsampling_y=0)
    at /usr/local/google/home/wtc/tmp/aom/av1/encoder/partition_search.h:59
59            get_plane_block_size(bsize, subsampling_x, subsampling_y);
(gdb) p x->e_mbd.is_chroma_ref
$2 = true
(gdb) c
Continuing.

Breakpoint 1, update_cb_offsets (x=0x7ffff2e0a820, bsize=BLOCK_16X16, subsampling_x=1, subsampling_y=0)
    at /usr/local/google/home/wtc/tmp/aom/av1/encoder/partition_search.h:59
59            get_plane_block_size(bsize, subsampling_x, subsampling_y);
(gdb) p x->e_mbd.is_chroma_ref
$3 = true
(gdb) c
Continuing.

Breakpoint 1, update_cb_offsets (x=0x7ffff2e0a820, bsize=BLOCK_4X8, subsampling_x=1, subsampling_y=0)
    at /usr/local/google/home/wtc/tmp/aom/av1/encoder/partition_search.h:59
59            get_plane_block_size(bsize, subsampling_x, subsampling_y);
(gdb) p x->e_mbd.is_chroma_ref
$4 = false
(gdb) c
Continuing.

Breakpoint 1, update_cb_offsets (x=0x7ffff2e0a820, bsize=BLOCK_4X8, subsampling_x=1, subsampling_y=0)
    at /usr/local/google/home/wtc/tmp/aom/av1/encoder/partition_search.h:59
59            get_plane_block_size(bsize, subsampling_x, subsampling_y);
(gdb) p x->e_mbd.is_chroma_ref
$5 = true
(gdb) list
54
55      static AOM_INLINE void update_cb_offsets(MACROBLOCK *x, const BLOCK_SIZE bsize,
56                                               const int subsampling_x,
57                                               const int subsampling_y) {
58        const BLOCK_SIZE plane_bsize =
59            get_plane_block_size(bsize, subsampling_x, subsampling_y);
60        x->cb_offset[PLANE_TYPE_Y] += block_size_wide[bsize] * block_size_high[bsize];
61        if (x->e_mbd.is_chroma_ref)
62          x->cb_offset[PLANE_TYPE_UV] +=
63              block_size_wide[plane_bsize] * block_size_high[plane_bsize];
(gdb) n
60        x->cb_offset[PLANE_TYPE_Y] += block_size_wide[bsize] * block_size_high[bsize];
(gdb) n
61        if (x->e_mbd.is_chroma_ref)
(gdb) n
62          x->cb_offset[PLANE_TYPE_UV] +=
(gdb) p plane_bsize
$6 = BLOCK_INVALID
(gdb)
```

So in the fourth call, get_plane_block_size(bsize, subsampling_x, subsampling_y) returns BLOCK_INVALID.

If we pass --i444 or -i420 instead of -i422 to aomenc, we see the same arguments passed to the first four update_cb_offsets() calls, but in the fourth call, get_plane_block_size(bsize, subsampling_x, subsampling_y) returns a valid block size.

The fix I came up with is to change aomenc to change the pixel format of the input image to YUV 4:2:0 (and remove the UV planes) when monochrome is 1.

Comment 7 by Git Watcher on Wed, Apr 7, 2021, 2:21 AM EDT

The following revision refers to this bug:
https://aomedia.googlesource.com/aom/+/30d6b683ba4feb81897027e84d0e5bb75008fe2f

commit 30d6b683ba4feb81897027e84d0e5bb75008fe2f
Author: Wan-Teh Chang <wtc@google.com>
Date: Wed Apr 07 01:35:54 2021

Assert valid plane_bsize in update_cb_offsets()

Compute plane_bsize only when it will be used. Assert plane_bsize is
valid.

~~BUG=aomedia:2913~~

Change-Id: I02af1020ad629b8df813814b75c8b7184fe05aa1

[modify] https://crrev.com/30d6b683ba4feb81897027e84d0e5bb75008fe2f/av1/encoder/partition_search.h

Comment 8 Deleted

Comment 9 by wtc@google.com on Tue, Apr 13, 2021, 1:45 PM EDT

Here are the steps I use to reproduce this bug.

```
$ cmake ../aom -DCMAKE_BUILD_TYPE=Debug -DSANITIZE=address
$ make -j
$ ./aomenc --i422 --monochrome -h 10 -w 10 -o out.webm ./poc4
```

I pass -DCMAKE_BUILD_TYPE=Debug to cmake so that the stack trace has file names and line numbers. It can be omitted if you just want to reproduce the bug.

Note that I pass a file name instead of /dev/null to the -o option of aomenc because it seems that the WebM writer needs to perform the 'seek' operation on the output file.
This avoids the following error messages when the bug is fixed:

```
webmenc> Segment::Finalize failed.
Fatal: WebM writer finalization failed.
```

Comment 10 by wtc@google.com on Tue, Apr 13, 2021, 1:54 PM EDT

The CL in comment 7 merely changes the ASan "global-buffer-overflow" error to an assertion failure, so that the bug can be detected without using ASan. It does not fix this bug.

I just uploaded a CL that fixes this bug: https://aomedia-review.googlesource.com/c/aom/+/135481

Comment 11 by Git Watcher on Tue, Apr 13, 2021, 8:30 PM EDT

The following revision refers to this bug:
https://aomedia.googlesource.com/aom/+/5c9bc4181071684d157fc47c736acf6c69a85d85

```
commit 5c9bc4181071684d157fc47c736acf6c69a85d85
Author: Wan-Teh Chang <wtc@google.com>
Date: Tue Apr 13 17:12:20 2021

Convert input frames to monochrome before encoding

When encoding a color input to a monochrome output, convert input frames
to monochrome frames in the AOM_IMG_FMT_I420 format before passing them
to aom_codec_encode(). The libaom encoder apparently has some problems
with encoding input frames in the AOM_IMG_FMT_I422 format to a
monochrome output. Although I could not get to the bottom of those
problems, this change allows us to bypass those problems.

Do not upgrade profile 0 to profile 2 when encoding to a monochrome
output, even if the input is YUV 4:2:2.

~~BUG=aomedia:2913~~

Change-Id: I86cc6a5a533092e241ba26b95d65a9cbe5fdd67f

[modify] https://crrev.com/5c9bc4181071684d157fc47c736acf6c69a85d85/apps/aomenc.c
```

Comment 12 by wtc@google.com on Tue, Apr 13, 2021, 8:30 PM EDT

Status: Fixed (was: Started)