☆ Starred by 7 users

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

Type-Defect
Priority-Medium

Issue 2912: stack-buffer-overflow in src/aom_image.c:334 or null pointer dereference in src/aom_image.c:311

Reported by zodf0...@gmail.com on Wed, Dec 23, 2020, 11:22 PM EST

Code ‹ Prev 389 of 2974 Next ›
Back to list

What version / commit were you testing with?
commit a5d214

What steps will reproduce the problem?

1. ./aomenc --pass=2 --usage=1 -o /dev/null ./poc3

What is the expected output?

It has two behaviors.
This is ASAN report:

```
➜ Yuan-fuzz ~/aom/build/aomenc --pass=2 --usage=1 -o /dev/null ./poc3
Warning: Assuming --pass=2 implies --passes=2

Warning: Enforcing one-pass encoding in realtime mode

Warning: non-zero lag-in-frames option ignored in realtime mode.

ASAN:DEADLYSIGNAL
=================================================================
==9159==ERROR: AddressSanitizer: SEGV on unknown address 0x000000000000 (pc 0x5578673b12c0 bp 0x62696c2f7273752f sp 0x7ffd99443d90 T0)
==9159==The signal is caused by a READ memory access.
==9159==Hint: address points to the zero page.
    #0 0x5578673b12bf in aom_img_metadata_free /home/yuan/afl-target/aom/aom/src/aom_image.c:311
    #1 0x5578673b12bf in aom_img_metadata_array_free /home/yuan/afl-target/aom/aom/src/aom_image.c:336
    #2 0x5578673b12bf in aom_img_remove_metadata /home/yuan/afl-target/aom/aom/src/aom_image.c:369
    #3 0x5578673b12bf in aom_img_free /home/yuan/afl-target/aom/aom/src/aom_image.c:270
    #4 0x55786719f0a1 in main /home/yuan/afl-target/aom/apps/aomenc.c:2874
    #5 0x7f9c599d4bf6 in __libc_start_main (/lib/x86_64-linux-gnu/libc.so.6+0x21bf6)
    #6 0x5578671b1739 in _start (/home/yuan/afl-target/aom/build/aomenc+0x93739)

AddressSanitizer can not provide additional info.
SUMMARY: AddressSanitizer: SEGV /home/yuan/afl-target/aom/aom/src/aom_image.c:311 in aom_img_metadata_free
==9159==ABORTING

```

```
➜ Yuan-fuzz ~/aom/build/aomenc --pass=2 --usage=1 -o /dev/null ./poc3
Warning: Assuming --pass=2 implies --passes=2

Warning: Enforcing one-pass encoding in realtime mode
```

```
Warning: non-zero lag-in-frames option ignored in realtime mode.

=================================================================
==9156==ERROR: AddressSanitizer: stack-buffer-overflow on address 0x7ffd471bfa48 at pc 0x5644e712db4e bp 0x7ffd471bf460 sp 0x7ffd471bf450
READ of size 8 at 0x7ffd471bfa48 thread T0
    #0 0x5644e712db4d in GetActualMallocAddress /home/yuan/afl-target/aom/aom_mem/aom_mem.c:46
    #1 0x5644e712db4d in aom_free /home/yuan/afl-target/aom/aom_mem/aom_mem.c:74
    #2 0x5644e71255e4 in aom_img_free /home/yuan/afl-target/aom/aom/src/aom_image.c:271
    #3 0x5644e6f130a1 in main /home/yuan/afl-target/aom/apps/aomenc.c:2874
    #4 0x7f809c720bf6 in __libc_start_main (/lib/x86_64-linux-gnu/libc.so.6+0x21bf6)
    #5 0x5644e6f25739 in _start (/home/yuan/afl-target/aom/build/aomenc+0x93739)

Address 0x7ffd471bfa48 is located in stack of thread T0 at offset 1112 in frame
    #0 0x5644e6f09edf in main /home/yuan/afl-target/aom/apps/aomenc.c:2309

  This frame has 17 object(s):
    [32, 36) 'q'
    [96, 104) 'iter'
    [160, 176) 'diff'
    [224, 240) 'cfg'
    [288, 320) 'timer'
    [352, 392) 'arg'
    [448, 680) 'global'
    [736, 752) 'y'
    [800, 816) 'u'
    [864, 880) 'v'
    [928, 1096) 'raw' <== Memory access at offset 1112 overflows this variable
    [1152, 1320) 'raw_shift'
    [1376, 1544) 'enc_img'
    [1600, 1768) 'dec_img'
    [1824, 1992) 'enc_hbd_img'
    [2048, 2216) 'dec_hbd_img'
    [2272, 2504) 'input'
HINT: this may be a false positive if your program uses some custom stack unwind mechanism or swapcontext
      (longjmp and C++ exceptions *are* supported)
SUMMARY: AddressSanitizer: stack-buffer-overflow /home/yuan/afl-target/aom/aom_mem/aom_mem.c:46 in GetActualMallocAddress
Shadow bytes around the buggy address:
  0x100028e2fef0: f2 f2 f2 f2 f2 f2 00 00 00 00 00 00 00 00 00 00
  0x100028e2ff00: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x100028e2ff10: 00 00 00 f2 f2 f2 f2 f2 f2 f2 00 00 f2 f2 f2 f2
  0x100028e2ff20: f2 f2 00 00 f2 f2 f2 f2 f2 f2 00 00 f2 f2 f2 f2
  0x100028e2ff30: f2 f2 00 00 00 00 00 00 00 00 00 00 00 00 00 00
=>0x100028e2ff40: 00 00 00 00 00 00 00 f2 f2[f2]f2 f2 f2 f2 00 00
  0x100028e2ff50: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x100028e2ff60: 00 00 00 f2 f2 f2 f2 f2 f2 f2 00 00 00 00 00 00
  0x100028e2ff70: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 f2
  0x100028e2ff80: f2 f2 f2 f2 f2 f2 00 00 00 00 00 00 00 00 00 00
  0x100028e2ff90: 00 00 00 00 00 00 00 00 00 00 00 f2 f2 f2 f2 f2
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
==9156==ABORTING

```

Comment 1 by zodf0...@gmail.com on Wed, Dec 23, 2020, 11:22 PM EST

poc3
2.1 KB View Download

Comment 2 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 3 by jz...@google.com on Mon, Jan 11, 2021, 1:52 PM EST Project Member

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

Comment 4 by wtc@google.com on Tue, Apr 13, 2021, 3:10 PM EDT Project Member

Status: Started (was: Assigned)

Hi zodf0055980,

Thank you very much for the bug report. My sincere apologies for the very late response!

Here are the steps I use to reproduce this bug.

$ cmake ../aom -DCMAKE_BUILD_TYPE=Debug -DSANITIZE=address

```
$ make -j
$ ./aomenc --pass=2 --usage=1 -o /dev/null ./poc3
```

As the following two warning messages indicate:

```
  Warning: Assuming --pass=2 implies --passes=2

  Warning: Enforcing one-pass encoding in realtime mode
```

the parse_global_config() function first sets global.pass to 2 and global.usage to 1 (AOM_USAGE_REALTIME) from the command-line options --pass=2 --usage=1, and then sets global.passes to global.pass (2) and subsequently to 1. So after parse_global_config() returns, we are left with global.pass=2 and global.passes=1.

Therefore, when we reach the long for loop in the main() function of aom/apps/aomenc.c that begins with this line:

```
   for (pass = global.pass ? global.pass - 1 : 0; pass < global.passes; pass++) {
```

we do not enter that for loop. The struct aom_img variable 'raw' is initialized inside that long for loop. So we do not initialize 'raw', and the call

```
   aom_img_free(&raw);
```

at the end of the main() function operates on an uninitialized 'raw' variable. This is why you observed two behaviors.

Hui Su's fix for ~~bug 2911~~ (https://aomedia-review.googlesource.com/c/aom/+/127342) also fixes this bug. So we can mark this bug as Fixed/Verified.

Note that the parse_global_config() function arguably should not set global.passes to a value less than global.pass. I can pursue that as an alternative fix for this bug.

Here is Hui Su's commit that fixes this bug:

https://aomedia.googlesource.com/aom/+/94bcbfe76b0fd5b8ac03645082dc23a88730c949

commit 94bcbfe76b0fd5b8ac03645082dc23a88730c949
Author: Hui Su <huisu@google.com>
Date: Wed Jan 13 23:01:41 2021

aomenc: initalize the image object

Otherwise it would cause problem when calling aom_img_free() at the end
if no frame is read.

~~BUG=aomedia:2911~~

Change-Id: I4350d5294706d2d84341e601e9ed6063229d0451

[modify] https://crrev.com/94bcbfe76b0fd5b8ac03645082dc23a88730c949/apps/aomenc.c

Comment 5 by Git Watcher on Tue, Apr 13, 2021, 5:16 PM EDT Project Member

The following revision refers to this bug:
https://aomedia.googlesource.com/aom/+/7a20d10027fd91fbe11e38182a1d45238e102c4a

commit 7a20d10027fd91fbe11e38182a1d45238e102c4a
Author: Wan-Teh Chang <wtc@google.com>
Date: Tue Apr 13 19:18:34 2021

Check global.pass when enforcing one-pass encoding

In parse_global_config(), when enforcing one-pass encoding in realtime
mode, check if global.pass is valid.

To reproduce this condition, pass --pass=2 --usage=1 to aomenc. After
setting global.pass to 2 and global.usage to 1 from the command-line
options, parse_global_config() performs the following two
transformations.

1. It first sets global.passes to 2 to match global.pass.

2. It then changes global.passes to 1 because global.usage is 1
(AOM_USAGE_REALTIME).

I propose that before changing global.passes to 1 we should check if
global.pass would be consistent with global.passes=1.

NOTE: This CL is an alternative way to fix the crash reported in
aomedia:2912.

~~BUG=aomedia:2912~~

Change-Id: I29e8f7a3cda1bbd9e2e1219873dcd152fe191ca4

[modify] https://crrev.com/7a20d10027fd91fbe11e38182a1d45238e102c4a/apps/aomenc.c

Comment 6 by wtc@google.com on Tue, Apr 13, 2021, 5:17 PM EDT Project Member

Status: Fixed (was: Started)