New issue

Jump to bottom

Heap overflow in heif_colorconversion.cc:2263 #207

Open **gaintcome** opened this issue on Feb 23, 2020 · 7 comments

gaintcome commented on Feb 23, 2020

I spotted this overflow in `heif_colorconversion.cc` .

how to reproduce:

```
./heif-convert heap-overflow.poc test.png
```

heap-overflow.poc.zip

Here is the report of ASAN

```
==19327==ERROR: AddressSanitizer: heap-buffer-overflow on address 0x62300000350e at pc 0x0000007cf8f3 bp 0x7ffd1d38dbd0 sp 0x7ffd1d38dbc8
READ of size 2 at 0x62300000350e thread T0
    #0 0x7cf8f2 in Op_YCbCr420_to_RRGGBBaa::convert_colorspace(std::shared_ptr<heif::HeifPixelImage const> const&, heif::ColorState, heif::ColorConversionOptions)
libheif/heif_colorconversion.cc:2263:58
    #1 0x7fd4d8 in heif::ColorConversionPipeline::convert_image(std::shared_ptr<heif::HeifPixelImage> const&) libheif/heif_colorconversion.cc:2446:15
    #2 0x781134 in heif::convert_colorspace(std::shared_ptr<heif::HeifPixelImage> const&, heif_colorspace, heif_chroma) libheif/heif_image.cc:396:19
    #3 0x5d4e46 in heif::HeifContext::Image::decode_image(std::shared_ptr<heif::HeifPixelImage>&, heif_colorspace, heif_chroma, heif_decoding_options const*) const
libheif/heif_context.cc:896:11
    #4 0x549151 in heif_decode_image libheif/heif.cc:660:33
    #5 0x522111 in main examples/heif_convert.cc:226:11
    #6 0x7f3e8328182f in __libc_start_main /build/glibc-LK5gWL/glibc-2.23/csu/../csu/libc-start.c:291
    #7 0x41e308 in _start (examples/heif-convert+0x41e308)

0x62300000350f is located 0 bytes to the right of 6159-byte region [0x623000001d00,0x62300000350f)
allocated by thread T0 here:
    #0 0x51a178 in operator new[](unsigned long) (examples/heif-convert+0x51a178)
    #1 0x77b052 in heif::HeifPixelImage::add_plane(heif_channel, int, int, int) libheif/heif_image.cc:151:27

SUMMARY: AddressSanitizer: heap-buffer-overflow libheif/heif_colorconversion.cc:2263:58 in Op_YCbCr420_to_RRGGBBaa::convert_colorspace(std::shared_ptr<heif::HeifPixelImage const>
const&, heif::ColorState, heif::ColorConversionOptions)
Shadow bytes around the buggy address:
  0x0c467fff8650: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x0c467fff8660: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x0c467fff8670: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x0c467fff8680: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x0c467fff8690: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
=>0x0c467fff86a0: 00[07]fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c467fff86b0: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c467fff86c0: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c467fff86d0: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c467fff86e0: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c467fff86f0: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
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
==19327==ABORTING
```

fgeek commented on Nov 4, 2021

CVE-2020-23109 has been assigned for this issue.

👍 1

p4zuu commented on Nov 4, 2021

Is the overflow fixed on master? I can't reproduce the bug

pgajdos commented on Jun 7

I have tried to reproduce with version 1.6.2 (built against x265 and libde265), and I succeeded:

```
$ valgrind  -q ./heif-convert /192382/heap-overflow.poc out.png
File contains 1 images
==2255== Invalid read of size 2
==2255==    at 0x48A2E3C: Op_YCbCr420_to_RRGGBBaa::convert_colorspace(std::shared_ptr<heif::HeifPixelImage const> const&, heif::ColorState, heif::ColorConversionOptions)
```

```
(heif_colorconversion.cc:2263)
==2255==    by 0x489D86D: UnknownInlinedFun (heif_colorconversion.cc:2446)
==2255==    by 0x489D86D: heif::convert_colorspace(std::shared_ptr<heif::HeifPixelImage> const&, heif_colorspace, heif_chroma) (heif_image.cc:396)
==2255==    by 0x48A80E4: heif::HeifContext::Image::decode_image(std::shared_ptr<heif::HeifPixelImage>&, heif_colorspace, heif_chroma, heif_decoding_options const*) const [clone
.constprop.0] [clone .isra.0] (heif_context.cc:896)
==2255==    by 0x48862FD: heif_decode_image (heif.cc:663)
==2255==    by 0x10F626: main (in /home/abuild/rpmbuild/BUILD/libheif-1.6.2/examples/heif-convert)
==2255==  Address 0x535037e is 6,158 bytes inside a block of size 6,159 alloc'd
==2255==    at 0x484A20F: operator new[](unsigned long) (in /usr/libexec/valgrind/vgpreload_memcheck-amd64-linux.so)
==2255==    by 0x4898FEA: heif::HeifPixelImage::add_plane(heif_channel, int, int, int) (heif_image.cc:151)
==2255==    by 0x48868DB: heif_image_add_plane (heif.cc:745)
==2255==    by 0x48A8936: UnknownInlinedFun (heif_decoder_libde265.cc:126)
==2255==    by 0x48A8936: libde265_v1_decode_image(void*, heif_image**) (heif_decoder_libde265.cc:306)
==2255==    by 0x488E86E: heif::HeifContext::decode_image(unsigned int, std::shared_ptr<heif::HeifPixelImage>&, heif_decoding_options const*) const (heif_context.cc:944)
==2255==    by 0x48A8097: heif::HeifContext::Image::decode_image(std::shared_ptr<heif::HeifPixelImage>&, heif_colorspace, heif_chroma, heif_decoding_options const*) const [clone
.constprop.0] [clone .isra.0] (heif_context.cc:880)
==2255==    by 0x488D5CE: heif::HeifContext::decode_image(unsigned int, std::shared_ptr<heif::HeifPixelImage>&, heif_decoding_options const*) const (heif_context.cc:1017)
==2255==    by 0x48A8097: heif::HeifContext::Image::decode_image(std::shared_ptr<heif::HeifPixelImage>&, heif_colorspace, heif_chroma, heif_decoding_options const*) const [clone
.constprop.0] [clone .isra.0] (heif_context.cc:880)
==2255==    by 0x48862FD: heif_decode_image (heif.cc:663)
==2255==    by 0x10F626: main (in /home/abuild/rpmbuild/BUILD/libheif-1.6.2/examples/heif-convert)
==2255==
==2255== Invalid read of size 2
==2255==    at 0x48A2E4C: Op_YCbCr420_to_RRGGBBaa::convert_colorspace(std::shared_ptr<heif::HeifPixelImage const> const&, heif::ColorState, heif::ColorConversionOptions)
(heif_colorconversion.cc:2264)
==2255==    by 0x489D86D: UnknownInlinedFun (heif_colorconversion.cc:2446)
==2255==    by 0x489D86D: heif::convert_colorspace(std::shared_ptr<heif::HeifPixelImage> const&, heif_colorspace, heif_chroma) (heif_image.cc:396)
==2255==    by 0x48A80E4: heif::HeifContext::Image::decode_image(std::shared_ptr<heif::HeifPixelImage>&, heif_colorspace, heif_chroma, heif_decoding_options const*) const [clone
.constprop.0] [clone .isra.0] (heif_context.cc:896)
==2255==    by 0x48862FD: heif_decode_image (heif.cc:663)
==2255==    by 0x10F626: main (in /home/abuild/rpmbuild/BUILD/libheif-1.6.2/examples/heif-convert)
==2255==  Address 0x535037e is 6,158 bytes inside a block of size 6,159 alloc'd
==2255==    at 0x484A20F: operator new[](unsigned long) (in /usr/libexec/valgrind/vgpreload_memcheck-amd64-linux.so)
==2255==    by 0x4898FEA: heif::HeifPixelImage::add_plane(heif_channel, int, int, int) (heif_image.cc:151)
==2255==    by 0x48868DB: heif_image_add_plane (heif.cc:745)
==2255==    by 0x48A8936: UnknownInlinedFun (heif_decoder_libde265.cc:126)
==2255==    by 0x48A8936: libde265_v1_decode_image(void*, heif_image**) (heif_decoder_libde265.cc:306)
==2255==    by 0x488E86E: heif::HeifContext::decode_image(unsigned int, std::shared_ptr<heif::HeifPixelImage>&, heif_decoding_options const*) const (heif_context.cc:944)
==2255==    by 0x48A8097: heif::HeifContext::Image::decode_image(std::shared_ptr<heif::HeifPixelImage>&, heif_colorspace, heif_chroma, heif_decoding_options const*) const [clone
.constprop.0] [clone .isra.0] (heif_context.cc:880)
==2255==    by 0x488D5CE: heif::HeifContext::decode_image(unsigned int, std::shared_ptr<heif::HeifPixelImage>&, heif_decoding_options const*) const (heif_context.cc:1017)
==2255==    by 0x48A8097: heif::HeifContext::Image::decode_image(std::shared_ptr<heif::HeifPixelImage>&, heif_colorspace, heif_chroma, heif_decoding_options const*) const [clone
.constprop.0] [clone .isra.0] (heif_context.cc:880)
==2255==    by 0x48862FD: heif_decode_image (heif.cc:663)
==2255==    by 0x10F626: main (in /home/abuild/rpmbuild/BUILD/libheif-1.6.2/examples/heif-convert)

[..]  lot of  'Use of uninitialised value' in PngEncoder::Encode()

==2255==
Written to out.png
$
```

However, I cannot reproduce same way anymore when I update to latest 1.12.0:

```
$ valgrind  -q ./heif-convert /192382/heap-overflow.poc test.png
File contains 1 images
Could not decode image: 0: Unsupported feature: Unsupported color conversion
$
```

The key is in commit `bca0162` which, applied on top of 1.6.2, gives the same result as 1.12.0. This means the heap overflow is fixed or at least hidden by the commit in question.

p4zuu commented on Oct 24

I've investigated the issue and it's actually partially fixed for the given poc.

I'm not familiar with libheif codebase, so please correct me if my understandings are wrong.
The attached poc contains 1 image with an alpha channel. When decoding the image in `HeifContext::decode_image_planar()` is called recursively to decode the alpha channel. Then, an alpha plane is added to the image by copying a new decoded channel to the image `m_planes : img->transfer_plane_from_image_as()` is called, and it copies the corresponding channel as the alpha one. For this poc, the image has an `heif_colorspace_YCbCr` colorspace, thus the new `heif_channel_Y` is copied into a `heif_channel_Alpha` channel in the `m_planes` map. However, the new `heif_channel_Y` channel is not the same size of the current img. At the return of `transfer_plane_from_image_as()` call, `m_planes[heif_channel_Alpha].bit_depth = 8` even though other channels have a bit_depth equal to 9. Since the plane allocated size depends on `bit_depth`, the alpha channel is smaller:

```
int bytes_per_component = (m_bit_depth + 7) / 8;
int bytes_per_pixel = num_interleaved_pixels_per_plane(chroma) * bytes_per_component;

stride = m_mem_width * bytes_per_pixel;
stride = (stride + alignment - 1U) & ~(alignment - 1U);

try {
  allocated_mem = new uint8_t[m_mem_height * stride + alignment - 1];
```

The libheif version built for this crash finally calls `Op_YCbCr420_to_RRGGBBaa::convert_colorspace()` that triggers the out-of-bounds read. In this function, `in_a` points to the Alpha channel buffer, that is smaller that the other channels' ones (`in_y` , `in_cb` and `in_cr`), thus when copying the buffers data into the ouput buffer, the read in `in_a` goes out of bounds:

```
if (has_alpha) {
  in_a = (uint16_t*) input->get_plane(heif_channel_Alpha, &in_a_stride);
}

// TRUNCATED

for (int y = 0; y < height; y++) {
   for (int x = 0; x < width; x++) {

     // TRUNCATED

     if (has_alpha) {
       out_p[y * out_p_stride + 8 * x + 6 + le] = (uint8_t) (in_a[y * in_a_stride / 2 + x] >> 8);
       out_p[y * out_p_stride + 8 * x + 7 - le] = (uint8_t) (in_a[y * in_a_stride / 2 + x] & 0xff);
     }
   }
 }
```

With the current master version `ea78603d8e47096606813d221725621306789ff2`, the convert_colorspace method called is `Op_RGB_HDR_to_RRGGBBaa_BE::convert_colorspace()`, where a check breaks the execution flow:

```
std::shared_ptr<HeifPixelImage>
Op_RGB_HDR_to_RRGGBBaa_BE::convert_colorspace(const std::shared_ptr<const HeifPixelImage>& input,
                                              const ColorState& target_state,
                                              const ColorConversionOptions& options)
{

   // TRUNCATED

  if (input_has_alpha) {
    if (input->get_bits_per_pixel(heif_channel_Alpha) == 8) { // condition taken here
      return nullptr;
    }
```

Strictly speaking, the issue is fixed. Similar pocs could not trigger the out-of-bounds read anymore because `Op_RGB_HDR_to_RRGGBBaa_BE::convert_colorspace` perform an appropriate check. However, `Op_YCbCr420_to_RRGGBBaa::convert_colorspace()` doesn't perform such verification, so there could still be a flow leading the bug.

From my point-of-view, it would be more appropriate to check that the channels size are the same in `img->transfer_plane_from_image_as()` before copying the source channel into the destination one:

```
diff --git a/libheif/heif_image.cc b/libheif/heif_image.cc
index 338933d..927603f 100644
--- a/libheif/heif_image.cc
+++ b/libheif/heif_image.cc
@@ -486,6 +486,11 @@ void HeifPixelImage::transfer_plane_from_image_as(const std::shared_ptr<HeifPixe
   // TODO: check that dst_channel does not exist yet

   ImagePlane plane = source->m_planes[src_channel];
+
+  assert((m_planes[src_channel].m_bit_depth == plane.m_bit_depth));
+  assert((m_planes[src_channel].m_width == plane.m_width));
+  assert((m_planes[src_channel].m_height == plane.m_height));
+
   source->m_planes.erase(src_channel);

   m_planes.insert(std::make_pair(dst_channel, plane));
```

This is maybe too strong and maybe not appropriate to libheif use cases, but that would remove a whole class of bug.

👍 1

pgajdos commented on Oct 27

cc **@farindk**

lastzero commented on Nov 2

Do you consider it fixed in the master branch or already in the version tagged 1.13.0?

p4zuu commented on Nov 4 • edited

They're both fixed. Strictly speaking, the code path that follows the crasher was fixed with bca0162018df9a32d21c05aad1fa203881fa7813, introduced in 1.7.0

Assignees

No one assigned

Labels

None yet

Projects

None yet

Milestone

No milestone

Development

No branches or pull requests

5 participants