Starred by 2 users

Owner:	saelo@google.com
CC:	proje...@google.com
Status:	Fixed *(Closed)*
Components:	----
Modified:	Nov 18, 2020

Deadline-90
CCProjectZeroMembers
Severity-High
Methodology-Fuzzing
Finder-saelo
Reported-2019-Dec-12
Vendor-ASWF
Product-OpenEXR
CVE-2020-11764
CVE-2020-11763
CVE-2020-11762
CVE-2020-11760
CVE-2020-11761
CVE-2020-11765
CVE-2020-11758
CVE-2020-11759
Fixed-2020-Feb-12

Issue 1987: OpenEXR: Multiple Memory Safety Issues

Reported by saelo@google.com on Thu, Dec 12, 2019, 4:23 AM EST

Description #2 by saelo@google.com (Apr 17, 2020)

Through fuzzing, I found multiple memory safety issues in the OpenEXR library [1]. The issues should be reproducible with the exrmakepreview binary compiled with address sanitizer and in Debug configuration as follows (this is on macOS, but it should be the same on linux):

```
% git clone https://github.com/AcademySoftwareFoundation/openexr.git
% cd openexr
% mkdir build && cd build
% CFLAGS="-g -fsanitize=address" CXXFLAGS="-g -fsanitize=address" LDFLAGS="-fsanitize=address" cmake .. -DCMAKE_BUILD_TYPE=Debug
% make -j
% ./bin/exrmakepreview /path/to/crash.exr /tmp/out
```

Generally, most of the issues appear to be out-of-bounds reads and/or writes and could be exploitable (for information disclosure or remote code execution) depending on the usage scenario of the OpenEXR library.

I've attempted to deduplicate the crashes and have summarized them below. However, it is still possible that multiple issues share the same root cause and just crash in different ways later on.
In addition to the unique crashes, the attached archive also contains the crashes that are likely duplicates so that hopefully it will be easier to verify the fixes.

bug1 (CVE-2020-11764)

This sample causes an out-of-bounds write (of presumably image pixels) on the heap in the copyIntoFrameBuffer function.

```
==32235==ERROR: AddressSanitizer: heap-buffer-overflow on address 0x608000000200 at pc 0x0001076def57 bp 0x7ffee92c0d30 sp 0x7ffee92c04e0
WRITE of size 2 at 0x608000000200 thread T0
    #0 0x1076def56 in __asan_memcpy (libclang_rt.asan_osx_dynamic.dylib:x86_64h+0x5ef56)
    #1 0x106b32385 in Imf_2_4::copyIntoFrameBuffer(char const*&, char*, char*, unsigned long, bool, double, Imf_2_4::Compressor::Format, Imf_2_4::PixelType,
Imf_2_4::PixelType) ImfMisc.cpp:324
    #2 0x106bc5e60 in Imf_2_4::(anonymous namespace)::TileBufferTask::execute() ImfTiledInputFile.cpp:619
    #3 0x1075386a7 in IlmThread_2_4::(anonymous namespace)::NullThreadPoolProvider::addTask(IlmThread_2_4::Task*) IlmThreadPool.cpp:456
    #4 0x107537204 in IlmThread_2_4::ThreadPool::addTask(IlmThread_2_4::Task*) IlmThreadPool.cpp:838
    #5 0x1075373cc in IlmThread_2_4::ThreadPool::addGlobalTask(IlmThread_2_4::Task*) IlmThreadPool.cpp:858
    #6 0x106bb56d1 in Imf_2_4::TiledInputFile::readTiles(int, int, int, int, int, int) ImfTiledInputFile.cpp:1214
    #7 0x106bb6a1c in Imf_2_4::TiledInputFile::readTiles(int, int, int, int, int, int) ImfTiledInputFile.cpp:1270
    #8 0x1069dea18 in Imf_2_4::(anonymous namespace)::bufferedReadPixels(Imf_2_4::InputFile::Data*, int, int) ImfInputFile.cpp:283
    #9 0x1069dda9d in Imf_2_4::InputFile::readPixels(int, int) ImfInputFile.cpp:818
    #10 0x106a25fd0 in Imf_2_4::RgbaInputFile::readPixels(int, int) ImfRgbaFile.cpp:1313
    #11 0x106942c9a in (anonymous namespace)::generatePreview(char const*, float, int, int&, Imf_2_4::Array2D<Imf_2_4::PreviewRgba>&) makePreview.cpp:114
    #12 0x106941f00 in makePreview(char const*, char const*, int, float, bool) makePreview.cpp:158
    #13 0x10693fe30 in main main.cpp:185
    #14 0x7fff66ba52e4 in start (libdyld.dylib:x86_64+0x112e4)
```

bug2 (CVE-2020-11763)

This sample appears to cause a std::vector to be read out-ouf-bounds. Afterwards, the calling code will write into an element slot of this vector, thus likely corrupting memory.

```
==16398==ERROR: AddressSanitizer: heap-buffer-overflow on address 0x6070000016d0 at pc 0x000106ff023e bp 0x7ffee8eedb50 sp 0x7ffee8eedb48
READ of size 8 at 0x6070000016d0 thread T0
    #0 0x106ff023d in std::__1::vector<unsigned long, std::__1::allocator<unsigned long> >::operator[](unsigned long) vector:1544
    #1 0x106fefc68 in Imf_2_4::TileOffsets::operator()(int, int, int, int) ImfTileOffsets.cpp:483
    #2 0x106fbaac8 in Imf_2_4::(anonymous namespace)::writeTileData(Imf_2_4::OutputStreamMutex*, Imf_2_4::TiledOutputFile::Data*, int, int, int, int, char const*, int)
ImfTiledOutputFile.cpp:467
    #3 0x106fb8864 in Imf_2_4::TiledOutputFile::copyPixels(Imf_2_4::TiledInputFile&) ImfTiledOutputFile.cpp:1543
    #4 0x106fbc70f in Imf_2_4::TiledOutputFile::copyPixels(Imf_2_4::InputFile&) ImfTiledOutputFile.cpp:1564
    #5 0x106d16211 in makePreview(char const*, char const*, int, float, bool) makePreview.cpp:176
    #6 0x106d13e30 in main main.cpp:185
    #7 0x7fff6a1fe3d4 in start (libdyld.dylib:x86_64+0x163d4)
```

bug3 (CVE-2020-11762)

This sample causes an out-of-bounds memcpy in DwaCompressor::uncompress in the UNKNOWN compression case. While the sample crashes on an out-of-bounds read, it looks like the code would afterwards also write the data outside the bounds of the output buffer.

```
==58195==ERROR: AddressSanitizer: heap-buffer-overflow on address 0x0009004f904e at pc 0x0001023a8e12 bp 0x7ffeee6e0c90 sp 0x7ffeee6e0440
READ of size 17171232 at 0x0009004f904e thread T0
    #0 0x1023a8e11 in __asan_memcpy (libclang_rt.asan_osx_dynamic.dylib:x86_64h+0x5ee11)
    #1 0x101661060 in Imf_2_4::DwaCompressor::uncompress(char const*, int, Imath_2_4::Box<Imath_2_4::Vec2<int> >, char const*&) ImfDwaCompressor.cpp:2819
    #2 0x10165cb80 in Imf_2_4::DwaCompressor::uncompress(char const*, int, int, char const*&) ImfDwaCompressor.cpp:2314
    #3 0x101774f03 in Imf_2_4::(anonymous namespace)::LineBufferTask::execute() ImfScanLineInputFile.cpp:551
    #4 0x1021156a7 in IlmThread_2_4::(anonymous namespace)::NullThreadPoolProvider::addTask(IlmThread_2_4::Task*) IlmThreadPool.cpp:456
    #5 0x102114204 in IlmThread_2_4::ThreadPool::addTask(IlmThread_2_4::Task*) IlmThreadPool.cpp:838
    #6 0x1021143cc in IlmThread_2_4::ThreadPool::addGlobalTask(IlmThread_2_4::Task*) IlmThreadPool.cpp:858
    #7 0x10175c2e4 in Imf_2_4::ScanLineInputFile::readPixels(int, int) ImfScanLineInputFile.cpp:1631
    #8 0x1015bbb6a in Imf_2_4::InputFile::readPixels(int, int) ImfInputFile.cpp:822
    #9 0x101603fd0 in Imf_2_4::RgbaInputFile::readPixels(int, int) ImfRgbaFile.cpp:1313
    #10 0x101522c9a in (anonymous namespace)::generatePreview(char const*, float, int, int&, Imf_2_4::Array2D<Imf_2_4::PreviewRgba>&) makePreview.cpp:114
    #11 0x101521f00 in makePreview(char const*, char const*, int, float, bool) makePreview.cpp:158
    #12 0x10151fe30 in main main.cpp:185
    #13 0x7fff6a1fe3d4 in start (libdyld.dylib:x86_64+0x163d4)
```

bug4 (CVE-2020-11760)

This sample appears to cause image pixel data to be read out-of-bounds during RLE uncompression. The reason seems to be a missing check on the input length in the rleUncompress function.

```
==6170==ERROR: AddressSanitizer: heap-buffer-overflow on address 0x611000000b40 at pc 0x000100e59e12 bp 0x7ffeefbfcc60 sp 0x7ffeefbfc410
READ of size 17 at 0x611000000b40 thread T0
    #0 0x100e59e11 in __asan_memcpy (libclang_rt.asan_osx_dynamic.dylib:x86_64h+0x5ee11)
    #1 0x100540864 in Imf_2_4::rleUncompress(int, int, signed char const*, char*) ImfRle.cpp:132
    #2 0x1001ca640 in Imf_2_4::RleCompressor::uncompress(char const*, int, int, char const*&) ImfRleCompressor.cpp:168
    #3 0x1004445dd in Imf_2_4::(anonymous namespace)::readSampleCountForLineBlock(Imf_2_4::InputStreamMutex*, Imf_2_4::DeepScanLineInputFile::Data*, int)
ImfDeepScanLineInputFile.cpp:1852
    #4 0x100441a5a in Imf_2_4::DeepScanLineInputFile::readPixelSampleCounts(int, int) ImfDeepScanLineInputFile.cpp:1967
    #5 0x100507de5 in Imf_2_4::CompositeDeepScanLine::readPixels(int, int) ImfCompositeDeepScanLine.cpp:460
    #6 0x10016995e in Imf_2_4::InputFile::readPixels(int, int) ImfInputFile.cpp:813
    #7 0x1001b1fd0 in Imf_2_4::RgbaInputFile::readPixels(int, int) ImfRgbaFile.cpp:1313
    #8 0x100006c9a in (anonymous namespace)::generatePreview(char const*, float, int, int&, Imf_2_4::Array2D<Imf_2_4::PreviewRgba>&) makePreview.cpp:114
    #9 0x100005f00 in makePreview(char const*, char const*, int, float, bool) makePreview.cpp:158
    #10 0x100003e30 in main main.cpp:185
    #11 0x7fff6a1fe3d4 in start (libdyld.dylib:x86_64+0x163d4)
```

bug5 (CVE-2020-11761)

This sample triggers an out-of-bounds read of what seems to be image pixel data during huffman uncompression. There appear to be multiple variants of this issue (crash1.exr, crash2.exr) that end up reading other datastructures out-of-bounds.

```
==14890==ERROR: AddressSanitizer: heap-buffer-overflow on address 0x63400003e800 at pc 0x00010a05c8b0 bp 0x7ffee600b4f0 sp 0x7ffee600b4e8
READ of size 1 at 0x63400003e800 thread T0
    #0 0x10a05c8af in Imf_2_4::FastHufDecoder::refill(unsigned long&, int, unsigned long&, int&, unsigned char const*&, int&) ImfFastHuf.cpp:492
    #1 0x10a05b969 in Imf_2_4::FastHufDecoder::decode(unsigned char const*, int, unsigned short*, int) ImfFastHuf.cpp:644
    #2 0x109cdf730 in Imf_2_4::hufUncompress(char const*, int, unsigned short*, int) ImfHuf.cpp:1082
    #3 0x109cf2fb6 in Imf_2_4::PizCompressor::uncompress(char const*, int, Imath_2_4::Box<Imath_2_4::Vec2<int> >, char const*&) ImfPizCompressor.cpp:583
    #4 0x109cf1960 in Imf_2_4::PizCompressor::uncompress(char const*, int, int, char const*&) ImfPizCompressor.cpp:285
    #5 0x109e3855f in Imf_2_4::(anonymous namespace)::LineBufferTaskIF::execute() ImfScanLineInputFile.cpp:866
    #6 0x10a7e86a7 in IlmThread_2_4::(anonymous namespace)::NullThreadPoolProvider::addTask(IlmThread_2_4::Task*) IlmThreadPool.cpp:456
    #7 0x10a7e7204 in IlmThread_2_4::ThreadPool::addTask(IlmThread_2_4::Task*) IlmThreadPool.cpp:838
    #8 0x10a7e73cc in IlmThread_2_4::ThreadPool::addGlobalTask(IlmThread_2_4::Task*) IlmThreadPool.cpp:858
    #9 0x109e2c2e4 in Imf_2_4::ScanLineInputFile::readPixels(int, int) ImfScanLineInputFile.cpp:1631
    #10 0x109c8bb6a in Imf_2_4::InputFile::readPixels(int, int) ImfInputFile.cpp:822
    #11 0x109cd3fd0 in Imf_2_4::RgbaInputFile::readPixels(int, int) ImfRgbaFile.cpp:1313
    #12 0x109bf2c9a in (anonymous namespace)::generatePreview(char const*, float, int, int&, Imf_2_4::Array2D<Imf_2_4::PreviewRgba>&) makePreview.cpp:114
    #13 0x109bf1f00 in makePreview(char const*, char const*, int, float, bool) makePreview.cpp:158
    #14 0x109befe30 in main main.cpp:185
    #15 0x7fff6a1fe3d4 in start (libdyld.dylib:x86_64+0x163d4)
```

bug6 (CVE-2020-11765)

This sample causes an out-of-bounds read on the stack. The underlying issue seems to be an off-by-one error in the ImfXdr.h read function (called during DwaCompressor::Classifier::Classifier), which for this sample instead of reading 255 bytes actually reads 256 bytes into a stack allocated buffer, thus overwriting the null terminator. This then causes a following strlen to read out-of-bounds.

```
==7113==ERROR: AddressSanitizer: stack-buffer-overflow on address 0x7ffee51a1190 at pc 0x00010b7ba5ae bp 0x7ffee51a0fe0 sp 0x7ffee51a0788
READ of size 263 at 0x7ffee51a1190 thread T0
    #0 0x10b7ba5ad in wrap_strlen (libclang_rt.asan_osx_dynamic.dylib:x86_64h+0x185ad)
    #1 0x10aa5f7a4 in std::__1::char_traits<char>::length(char const*) __string:218
    #2 0x10ac2b770 in std::__1::basic_string<char, std::__1::char_traits<char>, std::__1::allocator<char> >::basic_string<std::nullptr_t>(char const*) string:820
    #3 0x10ac11a0c in std::__1::basic_string<char, std::__1::char_traits<char>, std::__1::allocator<char> >::basic_string<std::nullptr_t>(char const*) string:818
    #4 0x10ac231aa in Imf_2_4::DwaCompressor::Classifier::Classifier(char const*&, int) ImfDwaCompressor.cpp:274
    #5 0x10aba1d22 in Imf_2_4::DwaCompressor::Classifier::Classifier(char const*&, int) ImfDwaCompressor.cpp:265
    #6 0x10ab9dba2 in Imf_2_4::DwaCompressor::uncompress(char const*, int, Imath_2_4::Box<Imath_2_4::Vec2<int> >, char const*&) ImfDwaCompressor.cpp:2437
    #7 0x10ab9cb80 in Imf_2_4::DwaCompressor::uncompress(char const*, int, int, char const*&) ImfDwaCompressor.cpp:2314
    #8 0x10acb4f03 in Imf_2_4::(anonymous namespace)::LineBufferTask::execute() ImfScanLineInputFile.cpp:551
```

#9 0x10b6586a7 in IlmThread_2_4::(anonymous namespace)::NullThreadPoolProvider::addTask(IlmThread_2_4::Task*) IlmThreadPool.cpp:456
#10 0x10b657204 in IlmThread_2_4::ThreadPool::addTask(IlmThread_2_4::Task*) IlmThreadPool.cpp:838
#11 0x10b6573cc in IlmThread_2_4::ThreadPool::addGlobalTask(IlmThread_2_4::Task*) IlmThreadPool.cpp:858
#12 0x10ac9c2e4 in Imf_2_4::ScanLineInputFile::readPixels(int, int) ImfScanLineInputFile.cpp:1631
#13 0x10aafbb6a in Imf_2_4::InputFile::readPixels(int, int) ImfInputFile.cpp:822
#14 0x10aafd7fd in Imf_2_4::InputFile::readPixels(int) ImfInputFile.cpp:830
#15 0x10ab4168c in Imf_2_4::RgbaInputFile::FromYca::readYCAScanLine(int, Imf_2_4::Rgba*) ImfRgbaFile.cpp:1137
#16 0x10ab40356 in Imf_2_4::RgbaInputFile::FromYca::readPixels(int) ImfRgbaFile.cpp:1061
#17 0x10ab3f1a1 in Imf_2_4::RgbaInputFile::FromYca::readPixels(int, int) ImfRgbaFile.cpp:970
#18 0x10ab43f40 in Imf_2_4::RgbaInputFile::readPixels(int, int) ImfRgbaFile.cpp:1309
#19 0x10aa61c9a in (anonymous namespace)::generatePreview(char const*, float, int, int&, Imf_2_4::Array2D<Imf_2_4::PreviewRgba>&) makePreview.cpp:114
#20 0x10aa60f00 in makePreview(char const*, char const*, int, float, bool) makePreview.cpp:158
#21 0x10aa5ee30 in main main.cpp:185
#22 0x7fff6a1fe3d4 in start (libdyld.dylib:x86_64+0x163d4)

bug7 (No CVE as the bug does not affect the library)

This sample causes an out-of-bounds read of pixel data in the makePreview function. As such, it might be specific to the exrmakepreview binary.

==8573==ERROR: AddressSanitizer: heap-buffer-overflow on address 0x0001151d9740 at pc 0x00010cddae12 bp 0x7ffee3bcccd0 sp 0x7ffee3bcc480
READ of size 2 at 0x0001151d9740 thread T0
#0 0x10cddae11 in __asan_memcpy (libclang_rt.asan_osx_dynamic.dylib:x86_64h+0x5ee11)
#1 0x10c039377 in (anonymous namespace)::generatePreview(char const*, float, int, int&, Imf_2_4::Array2D<Imf_2_4::PreviewRgba>&) makePreview.cpp:134
#2 0x10c037f00 in makePreview(char const*, char const*, int, float, bool) makePreview.cpp:158
#3 0x10c035e30 in main main.cpp:185
#4 0x7fff6a1fe3d4 in start (libdyld.dylib:x86_64+0x163d4)

bug8 (CVE-2020-11758)

This sample appears to cause image pixel data to be read out-of-bounds.

==5934==ERROR: AddressSanitizer: heap-buffer-overflow on address 0x6230000036a0 at pc 0x000105ec2096 bp 0x7ffee9f8e780 sp 0x7ffee9f8e778
READ of size 16 at 0x6230000036a0 thread T0
#0 0x105ec2095 in long long vector[2] Imf_2_4::loadSSE<true>(long long vector[2]*&) ImfOptimizedPixelReading.h:99
#1 0x105ec4bfd in void Imf_2_4::writeToRGBAFillASSETemplate<true, true>(long long vector[2]*&, long long vector[2]*&, long long vector[2]*&, unsigned short const&, long long vector[2]*&, unsigned long const&) ImfOptimizedPixelReading.h:310
#2 0x105ebf0c0 in Imf_2_4::optimizedWriteToRGBAFillA(unsigned short*&, unsigned short*&, unsigned short*&, unsigned short const&, unsigned short*&, unsigned long const&, unsigned long const&) ImfOptimizedPixelReading.h:426
#3 0x105ebd5d4 in Imf_2_4::(anonymous namespace)::LineBufferTaskIF::execute() ImfScanLineInputFile.cpp:968
#4 0x1068686a7 in IlmThread_2_4::(anonymous namespace)::NullThreadPoolProvider::addTask(IlmThread_2_4::Task*) IlmThreadPool.cpp:456
#5 0x10686720 4 in IlmThread_2_4::ThreadPool::addTask(IlmThread_2_4::Task*) IlmThreadPool.cpp:838
#6 0x1068673cc in IlmThread_2_4::ThreadPool::addGlobalTask(IlmThread_2_4::Task*) IlmThreadPool.cpp:858
#7 0x105eb02e4 in Imf_2_4::ScanLineInputFile::readPixels(int, int) ImfScanLineInputFile.cpp:1631
#8 0x105d0fb6a in Imf_2_4::InputFile::readPixels(int, int) ImfInputFile.cpp:822
#9 0x105d57fd0 in Imf_2_4::RgbaInputFile::readPixels(int, int) ImfRgbaFile.cpp:1313
#10 0x105c75c9a in (anonymous namespace)::generatePreview(char const*, float, int, int&, Imf_2_4::Array2D<Imf_2_4::PreviewRgba>&) makePreview.cpp:114
#11 0x105c74f00 in makePreview(char const*, char const*, int, float, bool) makePreview.cpp:158
#12 0x105c72e30 in main main.cpp:185
#13 0x7fff6a1fe3d4 in start (libdyld.dylib:x86_64+0x163d4)

bug9 (CVE-2020-11759)

This sample causes a write to a wild pointer due to what appears to be integer overflow related issues in CompositeDeepScanLine::Data::handleDeepFrameBuffer and readSampleCountForLineBlock.

==54297==ERROR: AddressSanitizer: SEGV on unknown address 0x0005931f7800 (pc 0x000101193f27 bp 0x7ffeeede3c50 sp 0x7ffeeede2f20 T0)
==54297==The signal is caused by a WRITE memory access.
#0 0x101193f26 in Imf_2_4::(anonymous namespace)::readSampleCountForLineBlock(Imf_2_4::InputStreamMutex*, Imf_2_4::DeepScanLineInputFile::Data*, int) ImfDeepScanLineInputFile.cpp
#1 0x101190a5a in Imf_2_4::DeepScanLineInputFile::readPixelSampleCounts(int, int) ImfDeepScanLineInputFile.cpp:1967
#2 0x101256de5 in Imf_2_4::CompositeDeepScanLine::readPixels(int, int) ImfCompositeDeepScanLine.cpp:460
#3 0x100eb895e in Imf_2_4::InputFile::readPixels(int, int) ImfInputFile.cpp:813
#4 0x100f00fd0 in Imf_2_4::RgbaInputFile::readPixels(int, int) ImfRgbaFile.cpp:1313
#5 0x100e20c9a in (anonymous namespace)::generatePreview(char const*, float, int, int&, Imf_2_4::Array2D<Imf_2_4::PreviewRgba>&) makePreview.cpp:114
#6 0x100e1ff00 in makePreview(char const*, char const*, int, float, bool) makePreview.cpp:158
#7 0x100e1de30 in main main.cpp:185
#8 0x7fff66ba52e4 in start (libdyld.dylib:x86_64+0x112e4)

These bugs are subject to a 90 day disclosure deadline. After 90 days elapse
or a patch has been made broadly available (whichever is earlier), the
report will become visible to the public.

[1] https://github.com/AcademySoftwareFoundation/openexr

Comment 1 by saelo@google.com on Fri, Feb 7, 2020, 3:29 AM EST

attachment.zip
13.2 MB Download

Comment 2 by saelo@google.com on Wed, Feb 19, 2020, 10:39 AM EST

Status: Fixed (was: New)
Labels: -Restrict-View-Commit

These issues have been fixed in the 2.4.1 release: https://github.com/AcademySoftwareFoundation/openexr/releases/tag/v2.4.1

Comment 3 by saelo@google.com on Fri, Apr 17, 2020, 4:20 AM EDT

Description was changed.

Comment 4 by saelo@google.com on Fri, Apr 17, 2020, 4:21 AM EDT

Labels: CVE-2020-11764 CVE-2020-11763 CVE-2020-11762 CVE-2020-11760 CVE-2020-11761 CVE-2020-11765 CVE-2020-11758 CVE-2020-11759

Comment 5 by saelo@google.com on Wed, Nov 18, 2020, 5:50 AM EST

Labels: Fixed-2020-Feb-12