Misaligned write in LossyDctDecoder_execute leading to undefined behavior (DWA/DWAB decompression)

High **cary-ilm** published **GHSA-w88v-vqhq-5p24** yesterday

Package

openexr

Affected versions

3.2.0-3.2.6, 3.3.0-3.3.8, 3.4.0–3.4.8

Patched versions

3.2.7, 3.3.9, 3.4.9

Description

Summary

A misaligned memory write vulnerability exists in `LossyDctDecoder_execute()` in `src/lib/OpenEXRCore/internal_dwa_decoder.h:749`. When decoding a DWA or DWAB-compressed EXR file containing a `FLOAT`-type channel, the decoder performs an in-place HALF→FLOAT conversion by casting an unaligned `uint8_t *` row pointer to `float *` and writing through it. Because the row buffer may not be 4-byte aligned, this constitutes undefined behavior under the C standard and crashes immediately on architectures that enforce alignment (ARM, RISC-V, etc.). On x86 it is silently tolerated at runtime but remains exploitable via compiler optimizations that assume aligned access.

Details

Vulnerable file: `src/lib/OpenEXRCore/internal_dwa_decoder.h`, line 749
Function: `LossyDctDecoder_execute()`

After DCT decoding, the function expands HALF (16-bit) pixel data to FLOAT (32-bit) in-place by walking each row right-to-left:

```
/* internal_dwa_decoder.h:737-752 */
for (int chan = 0; chan < numComp; ++chan)
{
    if (chanData[chan]->_type != EXR_PIXEL_FLOAT) continue;

    for (int y = 0; y < d->_height; ++y)
    {
        float*    floatXdrPtr = (float*) chanData[chan]->_rows[y]; // ← unaligned cast
        uint16_t* halfXdr     = (uint16_t*) floatXdrPtr;

        for (int x = d->_width - 1; x >= 0; --x)
        {
            floatXdrPtr[x] = one_from_native_float(              // ← UB: misaligned wri
                half_to_float(one_to_native16(halfXdr[x])));
        }
    }
}
```

`_rows[y]` is set in `DwaCompressor_uncompress()` by slicing the output buffer sequentially per channel (`internal_dwa_compressor.h:1035-1040`):

```
rv = DctCoderChannelData_push_row(
    me->alloc_fn, me->free_fn, &(cd->_dctData), outBufferEnd);
cd->_dctData._type = chan->data_type;
outBufferEnd += chan->width * chan->bytes_per_element;  // no alignment padding
```

No alignment padding is added between channels, so a FLOAT channel that follows a HALF channel (or any channel whose total byte size is not a multiple of 4) will receive a `_rows[y]` pointer that is **not 4-byte aligned**. Casting this to `float *` and writing through it at line 749 triggers undefined behavior.

UBSan (`-fsanitize=undefined`) reports:

```
/tmp/openexr_v347/src/lib/OpenEXRCore/internal_dwa_decoder.h:749:17:
runtime error: store to misaligned address 0x7592d9a3eeea for type 'float',
which requires 4 byte alignment
```

Full crash log:

```
/tmp/openexr_v347/src/lib/OpenEXRCore/internal_dwa_decoder.h:749:17: runtime error
store to misaligned address 0x7592d9a3eeea for type 'float', which requires 4 byte
alignment
0x7592d9a3eeea: note: pointer points here
 00 00  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  00
00 00 00 00 00 00
              ^
    #0 0x... in LossyDctDecoder_execute
/src/lib/OpenEXRCore/internal_dwa_decoder.h:749:32
```

```
    #1 0x... in DwaCompressor_uncompress
/src/lib/OpenEXRCore/internal_dwa_compressor.h:1079:18
    #2 0x... in internal_exr_undo_dwab /src/lib/OpenEXRCore/internal_dwa.c:237:18
    #3 0x... in exr_uncompress_chunk /src/lib/OpenEXRCore/compression.c:542:14
    #4 0x... in exr_decoding_run /src/lib/OpenEXRCore/decoding.c:580:14

SUMMARY: UndefinedBehaviorSanitizer: undefined-behavior
         internal_dwa_decoder.h:749:17
```

Fix direction: Use `memcpy`-based unaligned write (`unaligned_store32`) instead of dereferencing a `float *`, consistent with the pattern already used in `unpack.c`:

```
// replace:
floatXdrPtr[x] = one_from_native_float(half_to_float(one_to_native16(halfXdr[x])));
// with:
uint32_t fval = one_from_native_float(half_to_float(one_to_native16(
                    unaligned_load16(&halfXdr[x]))));
unaligned_store32(&floatXdrPtr[x], fval);
```

PoC

Requirements: OpenEXR ≤ v3.4.7, clang with `-fsanitize=address,undefined`

Build:

```
git clone https://github.com/AcademySoftwareFoundation/openexr.git
cd openexr && git checkout v3.4.7

cmake -B build \
  -DCMAKE_C_FLAGS="-fsanitize=address,undefined -g -O1" \
  -DCMAKE_CXX_FLAGS="-fsanitize=address,undefined -g -O1" \
  -DBUILD_SHARED_LIBS=OFF -DBUILD_TESTING=OFF
cmake --build build --target OpenEXRCore -j$(nproc)
```

Compile the attached `poc.c` harness (see attachment) against the built library:

```
clang -fsanitize=address,undefined -g -O1 \
  -I openexr/src/lib/OpenEXRCore \
  -I build/cmake/OpenEXRCore -I build/cmake \
  -I build/_deps/imath-build/config \
  poc.c build/src/lib/OpenEXRCore/libOpenEXRCore-3_4.a \
  build/external/OpenJPH/src/core/libopenjph.a \
  -lz -lm -lstdc++ -lubsan -o poc
```

Reproduce:

```
ASAN_OPTIONS="halt_on_error=1:print_stacktrace=1" \
UBSAN_OPTIONS="print_stacktrace=1:halt_on_error=1" \
./poc crash.exr
```

Expected output:

```
internal_dwa_decoder.h:749:17: runtime error: store to misaligned address
... for type 'float', which requires 4 byte alignment
SUMMARY: UndefinedBehaviorSanitizer: undefined-behavior
```

Impact

Any application using OpenEXRCore to decode untrusted EXR files with DWA/DWAB compression and FLOAT-type channels is affected.

- **Crash / Denial of Service**: reliable on architectures that enforce alignment (ARM, RISC-V, MIPS). Reproducible with the attached PoC on x86 under UBSan.
- **Silent data corruption / potential code execution**: on x86, the misaligned write succeeds at the hardware level but violates C aliasing and alignment rules. Compilers may generate incorrect code around such accesses when auto-vectorizing (SSE/AVX require 16-byte alignment for some instructions), potentially leading to memory corruption.

Affected: all versions **≤ v3.4.7** and current `main` branch.

Severity

High 7.1 / 10

CVSS v3 base metrics

Attack vector	Network
Attack complexity	Low
Privileges required	None
User interaction	Required
Scope	Unchanged
Confidentiality	None
Integrity	Low
Availability	High

Learn more about base metrics

CVSS:3.1/AV:N/AC:L/PR:N/UI:R/S:U/C:N/I:L/A:H

CVE ID

CVE-2026-34379

Weaknesses

- CWE-704
- CWE-787
- CWE-843

Credits

pwn2woot Reporter