



TALOS-2026-2364

# LibRaw deflate\_dng\_load\_raw integer overflow vulnerability

APRIL 7, 2026

CVE NUMBER

SUMMARY

CONFIRMED VULNERABLE VERSIONS

PRODUCT URLS

CVSSV3 SCORE

CWE

## DETAILS

max\_raw\_memory\_mb

unpack.cpp

deflate\_dng\_load\_raw() src/decoders/fp\_dng.cpp

```
void LibRaw::deflate_dng_load_raw()
{
    [...]
    tile_stripe_data_t tiles;
    tiles.init(ifd, imgdata.sizes, libraw_internal_data.unpacker_data,
              libraw_internal_data.unpacker_data.order,
              libraw_internal_data.internal_data.input);

    if (ifd->sample_format == 3)
    {
[1]     INT64 raw_bytes = tiles.tileCnt * tiles.tileWidth *
                        tiles.tileHeight * ifd->samples * sizeof(float);
[2]     if (raw_bytes > INT64(imgdata.rawparams.max_raw_memory_mb) * INT64(1024 *
1024))
        throw LIBRAW_EXCEPTION_TOOBIG;
[3]     float_raw_image = (float *)calloc(tiles.tileCnt * tiles.tileWidth *
                                         tiles.tileHeight * ifd->samples,
                                         sizeof(float));
    }
    [...]

    for (size_t y = 0, t = 0; y < imgdata.sizes.raw_height; y += tiles.tileHeight)
    {
        for (size_t x = 0; x < imgdata.sizes.raw_width; x += tiles.tileWidth, ++t)
        {
            [...]
            for (size_t row = 0; row < rowsInTile; ++row)
            {
                [...]
[4]         unsigned char *dst2 = (unsigned char *)&float_raw_image
                                [((y + row) * imgdata.sizes.raw_width + x) * ifd->samples];
[5]         memmove(dst2, dst, colsInTile * ifd->samples * sizeof(float));
            }
        }
    }
}
```

```

    }
  }
}

```

```

    tiles.tileWidth    tiles.tileHeight    TileWidth    TileLength
tiles.tileCnt        ceil(ImageWidth/TileWidth) * ceil(ImageLength/TileHeight)
ifd->samples        SamplesPerPixel

[1]                                                         INT64

```

```

INT64 raw_bytes = tiles.tileCnt * tiles.tileWidth * tiles.tileHeight * ifd->samples *
sizeof(float);

```

```

    tiles.tileCnt int    tiles.tileWidth unsigned
tiles.tileHeight unsigned    ifd->samples int    sizeof(float) size_t

```

```

sizeof(float)

```

```

[2]    raw_bytes

```

```

[3]                                                         float_raw_image

```

```

[4]    dst2    float_raw_image
      ((y + row) * raw_width + x) * samples

```

```

[5] memmove()    dst2

```

```

[1] [2]

```

```

LibRaw::unpack() src/decoders/unpack.cpp

```

```

if (INT64(rwidth) * INT64(rheight + 8) *
    INT64(sizeof(imgdata.rawdata.raw_image[0])) * 4
    + INT64(libraw_internal_data.unpacker_data.meta_length) >
    INT64(imgdata.rawparams.max_raw_memory_mb) * INT64(1024 * 1024))
    throw LIBRAW_EXCEPTION_TOOBIG;

```

deflate\_dng\_load\_raw()

max\_raw\_memory\_mb

max\_raw\_memory\_mb >= 11'017 MB

max\_raw\_memory\_mb

[1] [2]

ImageWidth=38000 ImageLength=38000 TileWidth=9500 TileHeight=9500

SamplesPerPixel=3

[1]

```

tileCnt * tileWidth * tileHeight * samples = 16 * 9500 * 9500 * 3 = 4'332'000'000

```

UINT32\_MAX

37'032'704

sizeof(float)

```

37'032'704 * 4 = 148'130'816 bytes (~141 MB)

```

[2]

[3]

```
16 * 9500 * 9500 * 3 * 4 = 17'328'000'000 bytes (~16.1 GB)
```

```
[4] [5]                                     y   row
      memmove()
      unpack()                               max_raw_memory_mb
```

## Crash Information

```
AddressSanitizer:DEADLYSIGNAL
=====
==69447==ERROR: AddressSanitizer: SEGV on unknown address 0x000110bb2240 (pc
0x0001017cfc28 bp 0x00016eeb6550 sp 0x00016eeb5d00 T0)
==69447==The signal is caused by a WRITE memory access.
   #0 0x0001017cfc28 in __sanitizer_internal_memmove+0x84
(libclang_rt.asan_osx_dynamic.dylib:arm64e+0x53c28)
   #1 0x000100f16c1c in LibRaw::deflate_dng_load_raw() fp_dng.cpp:419
   #2 0x000100fcba84 in LibRaw::unpack() unpack.cpp:447
   #3 0x000100e8d400 in main poc_deflate_dng_overflow.cpp:75
   #4 0x00018f671d50 (<unknown module>)

==69447==Register values:
  x[0] = 0x0000000110bb2240   x[1] = 0x00000005b2e89e90   x[2] = 0x000000000001bd50
  x[3] = 0x00000005b0b574a4
  x[4] = 0x00000005b2ea5bdc   x[5] = 0x0000000000000000   x[6] = 0x00000005b2ea5bdf
  x[7] = 0x0000000000000000
  x[8] = 0x000000000001bd40   x[9] = 0x00000005b2e89ef0   x[10] = 0x0000000110bb2260
  x[11] = 0x000000000001bd40
  x[12] = 0x0000007022199bf0   x[13] = 0x00000000000006f5   x[14] = 0x00000000000006f0
  x[15] = 0x00000005b0b6534c
  x[16] = 0x0000000302803ea8   x[17] = 0x00000005b0b6c2a0   x[18] = 0x0000000000000000
  x[19] = 0x000000000001bd50
  x[20] = 0x00000005b2e89e90   x[21] = 0x0000000110bb2240   x[22] = 0x0000000000000514
  x[23] = 0x000000010220a368
  x[24] = 0x00000001022095e8   x[25] = 0x000000000001bd0f   x[26] = 0x0000007000020000
  x[27] = 0x000000000001bd50
  x[28] = 0x000000002dde41ac   fp = 0x000000016eeb6550   lr = 0x0000000101801598
  sp = 0x000000016eeb5d00
AddressSanitizer can not provide additional info.
SUMMARY: AddressSanitizer: SEGV fp_dng.cpp:419 in LibRaw::deflate_dng_load_raw()
==69447==ABORTING
```

## TIMELINE

CREDIT

---

VULNERABILITY REPORTS

PREVIOUS REPORT

TALOS-2026-2363



© 2026 Cisco Systems, Inc. and/or its affiliates. All rights reserved. View our [Privacy Policy](#).