



TALOS-2026-2363

LibRaw uncompressed_fp_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

uncompressed_fp_dng_load_raw() src/decoders/fp_dng.cpp

```
void LibRaw::uncompressed_fp_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);

    [1] INT64 allocsz = INT64(tiles.tileCnt) * INT64(tiles.tileWidth) *
                  INT64(tiles.tileHeight) * INT64(ifd->samples) *
                  INT64(sizeof(float));
    [2] if (allocsz > INT64(imgdata.rawparams.max_raw_memory_mb) * INT64(1024 * 1024))
        throw LIBRAW_EXCEPTION_TOOBIG;

    if (ifd->sample_format == 3)
    [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 (unsigned x = 0; x < imgdata.sizes.raw_width && t <
              (unsigned)tiles.tileCnt;
             x += tiles.tileWidth, ++t)
        {
            libraw_internal_data.internal_data.input->seek(tiles.tOffsets[t],
            SEEK_SET);
            size_t rowsInTile = y + tiles.tileHeight > imgdata.sizes.raw_height ?
            imgdata.sizes.raw_height - y : tiles.tileHeight;
            size_t colsInTile = x + tiles.tileWidth > imgdata.sizes.raw_width ?
            imgdata.sizes.raw_width - x : tiles.tileWidth;
```

```

size_t inrowbytes = colsInTile * bytesps * ifd->samples;
int fullrowbytes = tiles.tileWidth * bytesps * ifd->samples;
size_t outrowbytes = colsInTile * sizeof(float) * ifd->samples;

for (size_t row = 0; row < rowsInTile; ++row)
{
[4]     unsigned char *dst = fullrowbytes > int(inrowbytes) ? rowbuf.data():
        (unsigned char *)&float_raw_image
        [((y + row) * imgdata.sizes.raw_width + x) * ifd->samples];
[5]     libraw_internal_data.internal_data.input->read(dst, 1, fullrowbytes);
        [...]
    }
}
}
}
}

```

```

[4] fullrowbytes          tileWidth * bytesps * samples      bytesps = bps /
8   inrowbytes
    dst                  float_raw_image

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

[1]                                     INT64()
[2]

[3]

```

```

calloc(tiles.tileCnt * tiles.tileWidth * tiles.tileHeight * ifd->samples,
sizeof(float))

```

```

tiles.tileCnt int tiles.tileWidth
unsigned tiles.tileHeight unsigned ifd->samples int

UINT32_MAX

sizeof(float)      size_t

```

[1] [2]

```
[3]         tileCnt * tileWidth * tileHeight * samples > UINT32_MAX
           UINT32_MAX + 1 = 4'294'967'296
```

```
[1]         4'294'967'296 * 4 = 17'179'869'184 bytes
max_raw_memory_mb >= 17'179'869'184 / (1'024 * 1'024) = 16'384 MB
```

max_raw_memory_mb

```
           fullrowbytes <= inrowbytes           [4]
dst         float_raw_image
```

```
dst = &float_raw_image[((y + row) * imgdata.sizes.raw_width + x) * ifd->samples];
```

[5]

raw_width raw_height

```
           width = 64000 height = 22370 samples = 3           64000 *
22370 * 3 = 4'295'040'000           72'704           72'704 * 4
= 290'816           64000 * 3 * 4 = 768'000
```

```
           unpack()           max_raw_memory_mb
```

Crash Information

```
=====
==50166==ERROR: AddressSanitizer: heap-buffer-overflow on address 0x000104a53c00 at
pc 0x000103121740 bp 0x00016d4d2530 sp 0x00016d4d1ce0
WRITE of size 768000 at 0x000104a53c00 thread T0
#0 0x00010312173c in __asan_memmove+0x234
(libclang_rt.asan_osx_dynamic.dylib:arm64e+0x8573c)
#1 0x000102937ccc in LibRaw_buffer_datastream::read(void*, unsigned long,
unsigned long) libraw_datastream.cpp:339
#2 0x0001028fc79c in LibRaw::uncompressed_fp_dng_load_raw() fp_dng.cpp:652
#3 0x0001029af79c in LibRaw::unpack() unpack.cpp:447
#4 0x0001028710c8 in main poc_fp_dng_overflow.cpp:54
#5 0x00018f671d50 (<unknown module>)
```

```
0x000104a53c00 is located 0 bytes after 291840-byte region
[0x000104a0c800,0x000104a53c00)
allocated by thread T0 here:
  #0 0x0001030d9620 in calloc+0x80
(libclang_rt.asan_osx_dynamic.dylib:arm64e+0x3d620)
  #1 0x0001029b4e04 in LibRaw::calloc(unsigned long, unsigned long)
utils_libraw.cpp:275
  #2 0x0001028fc4ac in LibRaw::uncompressed_fp_dng_load_raw() fp_dng.cpp:626
  #3 0x0001029af79c in LibRaw::unpack() unpack.cpp:447
  #4 0x0001028710c8 in main poc_fp_dng_overflow.cpp:54
  #5 0x00018f671d50 (<unknown module>)
```

```
SUMMARY: AddressSanitizer: heap-buffer-overflow libraw_datastream.cpp:339 in
LibRaw_buffer_datastream::read(void*, unsigned long, unsigned long)
```

```
Shadow bytes around the buggy address:
```

```
0x000104a53980: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x000104a53a00: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x000104a53a80: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x000104a53b00: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x000104a53b80: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
=>0x000104a53c00:[fa]fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
0x000104a53c80: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
0x000104a53d00: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
0x000104a53d80: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
0x000104a53e00: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
0x000104a53e80: fa 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
```

```
==50166==ABORTING
```

TIMELINE

CREDIT

VULNERABILITY REPORTS

NEXT REPORT

PREVIOUS REPORT

TALOS-2026-2364

TALOS-2026-2359



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