TALOS-2026-2331

LibRaw lossless_jpeg_load_raw heap-based buffer overflow vulnerability

APRIL 7, 2026

CVE NUMBER

SUMMARY

CONFIRMED VULNERABLE VERSIONS

PRODUCT URLS

CVSSV3 SCORE

CWE

DETAILS

col

LibRaw::lossless_jpeg_load_raw() src/decoders/decoders_dcraw.cpp

```
void LibRaw::lossless_jpeg_load_raw()
{
    int jwide, jhigh, jrow, jcol, val, jidx, i, j, row = 0, col = 0;
    struct jhead jh;
    ushort *rp;

    if (!ljpeg_start(&jh, 0))
        return;

    [...]

[1] jwide = jh.wide * jh.clrs;

    [...]

    try
    {
        for (jrow = 0; jrow < jh.high; jrow++)
        {
            [...]
            rp = ljpeg_row(jrow, &jh);
            [...]
[2]         for (jcol = 0; jcol < jwide; jcol++)
            {
                val = curve[*rp++];
[3]             if (cr2_slice[0])
                {
                    jidx = jrow * jwide + jcol;
[4]                 i = jidx / (cr2_slice[1] * raw_height);
                    if ((j = i >= cr2_slice[0]))
                        i = cr2_slice[0];
                    if(!cr2_slice[1+j])
```

```
                        throw LIBRAW_EXCEPTION_IO_CORRUPT;

                    jidx -= i * (cr2_slice[1] * raw_height);
                    row = jidx / cr2_slice[1 + j];
[5]                 col = jidx % cr2_slice[1 + j] + i * cr2_slice[1];
                }
                [...]
[6]             if (row > raw_height)
                    throw LIBRAW_EXCEPTION_IO_CORRUPT;
[7]             if ((unsigned)row < raw_height)
                    RAW(row, col) = val;
                [...]
            }
        }
    }
    [...]
}
```

cr2_slice cr2_slice[0]
cr2_slice[1] cr2_slice[2]

[1] jwide
[2] jcol jwide-1 [3] cr2_slice[0]
jidx jrow * jwide + jcol
row col

[4] [5] [4] i
jidx / (cr2_slice[1] * raw_height) [5] col jidx % cr2_slice[1 + j] + i * cr2_slice[1]

RAW(row, col) [7]

```
imgdata.rawdata.raw_image[row * raw_width + col]
```

row [6] col col
raw_width

raw_image src/decoders/unpack.cpp

```
imgdata.rawdata.raw_alloc =
    calloc(size_t(rwidth) * (size_t(rheight) + 8),
sizeof(imgdata.rawdata.raw_image[0]));
```

raw_width × (raw_height + 8) row × raw_width + col

```
row × raw_width + col < raw_width × (raw_height + 8)
```

row < raw_height col < raw_width

[5]

```
col = jidx % cr2_slice[1 + j] + i × cr2_slice[1]
```

i ≥ 1 i × cr2_slice[1] col cr2_slice[1] col

cr2_slice[1] = 512 i = 3 col = X + 3 × 512 = X + 1536 raw_width = 32 col < 32 col ≥ 1536

val [7]

```
rp = ljpeg_row(jrow, &jh);
[...]
val = curve[*rp++];
```

rp ljpeg_row() *rp curve curve

linear_table() src/utils/curves.cpp read_shorts(curve, len)

curve *rp

ljpeg_row()

unpack() [7]

Crash Information

```
=================================================================
==56971==ERROR: AddressSanitizer: heap-buffer-overflow on address 0x620000000e80 at
pc 0x000102b69620 bp 0x00016d23aa10 sp 0x00016d23aa08
WRITE of size 2 at 0x620000000e80 thread T0
    #0 0x000102b6961c in LibRaw::lossless_jpeg_load_raw() decoders_dcraw.cpp:592
    #1 0x000102c46b64 in LibRaw::unpack() unpack.cpp:447
    #2 0x000102b08e4c in main poc_cr2_oob.cpp:38
    #3 0x00018f671d50  (<unknown module>)

0x620000000e80 is located 0 bytes after 3584-byte region
[0x620000000080,0x620000000e80)
allocated by thread T0 here:
    #0 0x00010353d330 in malloc+0x78
(libclang_rt.asan_osx_dynamic.dylib:arm64e+0x3d330)
    #1 0x000102c4b4ac in LibRaw::malloc(unsigned long) utils_libraw.cpp:260
    #2 0x000102c46834 in LibRaw::unpack() unpack.cpp:395
    #3 0x000102b08e4c in main poc_cr2_oob.cpp:38
    #4 0x00018f671d50  (<unknown module>)

SUMMARY: AddressSanitizer: heap-buffer-overflow decoders_dcraw.cpp:592 in
LibRaw::lossless_jpeg_load_raw()
Shadow bytes around the buggy address:
  0x620000000c00: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x620000000c80: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x620000000d00: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x620000000d80: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x620000000e00: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
=>0x620000000e80:[fa]fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x620000000f00: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x620000000f80: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x620000001000: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x620000001080: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x620000001100: 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
  ==56971==ABORTING
```

TIMELINE

CREDIT

VULNERABILITY REPORTS

NEXT REPORT

TALOS-2026-2358

PREVIOUS REPORT

TALOS-2026-2330