



TALOS-2026-2359

# LibRaw x3f\_load\_huffman heap-based buffer overflow vulnerability

APRIL 7, 2026

CVE NUMBER

SUMMARY

CONFIRMED VULNERABLE VERSIONS

PRODUCT URLS

CVSSV3 SCORE

CWE

## DETAILS

`max_raw_memory_mb``x3f_load_huffman() src/x3f/x3f_utils_patched.cpp`

```
static void x3f_load_huffman(x3f_info_t *I, x3f_directory_entry_t *DE, ...)
{
    x3f_directory_entry_header_t *DEH = &DE->header;
    x3f_image_data_t *ID = &DEH->data_subsection.image_data;
    x3f_huffman_t *HUF = new_huffman(&ID->huffman);

    uint32_t size;

    [...]

    switch (ID->type_format)
    {
    case X3F_IMAGE_RAW_HUFFMAN_X530:
    case X3F_IMAGE_RAW_HUFFMAN_10BIT:
[1]     size = ID->columns * ID->rows * 3;
        [...]
[2]     HUF->x3rgb16.buf = malloc(sizeof(uint16_t) * size);
        HUF->x3rgb16.data = (uint16_t *)HUF->x3rgb16.buf;
        break;
        [...]
    }

    [...]
}
```

`ID->columns ID->rows``[1]``columns * rows * 3``ID->columns`

```
ID->rows    uint32_t
          UUINT32_MAX  [2]
```

```
static void huffman_decode_row(x3f_info_t * /*I*/, x3f_directory_entry_t *DE,
                             int /*bits*/, int row, int offset, int *minimum)  {
    [...]
    int16_t c[3] = {(int16_t)offset, (int16_t)offset, (int16_t)offset};
    [...]

    for (col = 0; col < (int)ID->columns; col++)
    {
        int color;

        for (color = 0; color < 3; color++)
        {
            uint16_t c_fix;

[3]         c[color] += get_huffman_diff(&BS, &HUF->tree);
            if (c[color] < 0)
            {
                c_fix = 0;
                [...]
            }
            else
            {
                c_fix = c[color];
            }

            switch (ID->type_format)
            {
                case X3F_IMAGE_RAW_HUFFMAN_X530:
                case X3F_IMAGE_RAW_HUFFMAN_10BIT:
[4]         HUF->x3rgb16.data[3 * (row * ID->columns + col) + color] =
                    (uint16_t)c_fix;
                    break;
                    [...]
            }
        }
    }
}
```

```
[3]                                     int16_t      c
huffman_decode()  huffman_decode_row()  rows
                  columns                [4]      rows *
```

```
columns * 3
color
```

```
[4] 3 * (row * ID->columns + col) +
```

```
c_fix
int16_t
```

```
columns * rows * 3      UINT32_MAX
```

```
src/decoders/unpack.cpp
```

```
if (INT64(MAX(S.width, S.raw_width)) *
    INT64(MAX(S.height, S.raw_height) + 8) *
    INT64(sizeof(*imgdata.image)) // 8, accounts for max 4 channels * 2 bytes
    + INT64(libraw_internal_data.unpacker_data.meta_length)
    > INT64(imgdata.rawparams.max_raw_memory_mb) * INT64(1024 * 1024))
    throw LIBRAW_EXCEPTION_TOOBIG;
```

```
[1]      columns * rows > 1,431,655,765      [1]
          max_raw_memory_mb
[2]      sizeof(uint16_t) *
size      columns * rows >
715,827,882      sizeof(uint16_t)      size_t
[2]
unpack()      max_raw_memory_mb
-DUSE_X3FTOOLS
```

#### Crash Information

```
=====
==43504==ERROR: AddressSanitizer: heap-buffer-overflow on address 0x00011fbe6200 at
pc 0x0001005236d8 bp 0x00016f9766b0 sp 0x00016f9766a8
WRITE of size 2 at 0x00011fbe6200 thread T0
#0 0x0001005236d4 in huffman_decode_row(x3f_info_s*, x3f_directory_entry_s*, int,
```

```

int, int, int*) x3f_utils_patched.cpp:1083
  #1 0x000100521efc in x3f_load_huffman(x3f_info_s*, x3f_directory_entry_s*, int,
int, int) x3f_utils_patched.cpp:1483
  #2 0x00010051f43c in x3f_load_data(x3f_s*, x3f_directory_entry_s*)
x3f_utils_patched.cpp:2076
  #3 0x0001005159a4 in LibRaw::x3f_load_raw() x3f_parse_process.cpp:579
  #4 0x00010050b34c in LibRaw::unpack() unpack.cpp:447
  #5 0x0001003cd100 in main poc_x3f_overflow.cpp:43
  #6 0x00018f671d50 (<unknown module>)

```

0x00011fbe6200 is located 0 bytes after 410065408-byte region  
[0x0001074d4800,0x00011fbe6200)

allocated by thread T0 here:

```

#0 0x000100ce1330 in malloc+0x78
(libclang_rt.asan_osx_dynamic.dylib:arm64e+0x3d330)
  #1 0x00010052146c in x3f_load_huffman(x3f_info_s*, x3f_directory_entry_s*, int,
int, int) x3f_utils_patched.cpp
  #2 0x00010051f43c in x3f_load_data(x3f_s*, x3f_directory_entry_s*)
x3f_utils_patched.cpp:2076
  #3 0x0001005159a4 in LibRaw::x3f_load_raw() x3f_parse_process.cpp:579
  #4 0x00010050b34c in LibRaw::unpack() unpack.cpp:447
  #5 0x0001003cd100 in main poc_x3f_overflow.cpp:43
  #6 0x00018f671d50 (<unknown module>)

```

SUMMARY: AddressSanitizer: heap-buffer-overflow x3f\_utils\_patched.cpp:1083 in  
huffman\_decode\_row(x3f\_info\_s\*, x3f\_directory\_entry\_s\*, int, int, int, int\*)

Shadow bytes around the buggy address:

```

0x00011fbe5f80: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x00011fbe6000: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x00011fbe6080: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x00011fbe6100: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x00011fbe6180: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
=>0x00011fbe6200:[fa]fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
0x00011fbe6280: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
0x00011fbe6300: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
0x00011fbe6380: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
0x00011fbe6400: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
0x00011fbe6480: 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

```

==43504==ABORTING

TIMELINE

CREDIT

---

VULNERABILITY REPORTS

NEXT REPORT

PREVIOUS REPORT

TALOS-2026-2363

TALOS-2026-2358



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