Commit 35e7aa3

cary-ilm committed 2 weeks ago

Fix B44/B44A integer overflow: use uint64_t for row offset (#2312)

The B44 and B44A decoder and encoder use channel width (`nx`) and
height (`ny`) in row pointer math. `nx` and `ny` are `int`; the
scratch buffer is correctly sized with `(uint64_t)ny * (uint64_t)nx *
bytes_per_element`, but row bases were computed as:

```
  row0 = (uint16_t*)scratch;
  row0 += y * nx;   // int * int -> signed overflow when y*nx > INT_MAX
```

For large `nx` (e.g. 268435456), `y*nx` overflows, so `row0`/`row1`/`row2`/`row3`
point before the scratch buffer.

Fix: compute the row offset in `uint64_t` before pointer arithmetic in both
`uncompress_b44_impl` (decoder) and `compress_b44_impl` (encoder).

Analysis and solution with the help of Curor / Claude Opus 4.5

Signed-off-by: Cary Phillips <cary@ilm.com>

RB-3.4 + release · v3.4.9 ••• v3.4.8-rc

1 parent bc44767 commit 35e7aa3

1 file changed +12 -11 lines changed

Top

Filter files…

src/lib/OpenEXRCore

internal_b44.c

1 file changed +12 -11 lines changed

Search within code

src/lib/OpenEXRCore/internal_b44.c

```
@@ -427,13 +427,13 @@ compress_b44_impl (exr_encode_pipeline_t* encode, int flat_field)
427 427                 // rightmost column and the bottom row.
428 428                 //
429 429                 uint16_t *row0, *row1, *row2, *row3;
    430 +             /* row offset in elements: use uint64_t so y*nx cannot overflow int */
    431 +             uint64_t row_off = (uint64_t) (y) * (uint64_t) (nx);
430 432
431     -             row0 = (uint16_t*) scratch;
432     -             row0 += y * nx;
433     -
434     -             row1 = row0 + nx;
435     -             row2 = row1 + nx;
436     -             row3 = row2 + nx;
    433 +             row0 = (uint16_t*) scratch + row_off;
    434 +             row1 = row0 + (uint64_t) nx;
    435 +             row2 = row1 + (uint64_t) nx;
    436 +             row3 = row2 + (uint64_t) nx;
437 437
438 438                 if (y + 3 >= ny)
439 439                 {
@@ -557,11 +557,12 @@ uncompress_b44_impl (
557 557
558 558             for (int y = 0; y < ny; y += 4)
559 559             {
560     -                 row0 = (uint16_t*) scratch;
561     -                 row0 += y * nx;
562     -                 row1 = row0 + nx;
563     -                 row2 = row1 + nx;
564     -                 row3 = row2 + nx;
    560 +                 /* row offset in elements: use uint64_t so y*nx cannot overflow int */
    561 +                 uint64_t row_off = (uint64_t) (y) * (uint64_t) (nx);
    562 +                 row0 = (uint16_t*) scratch + row_off;
    563 +                 row1 = row0 + (uint64_t) nx;
    564 +                 row2 = row1 + (uint64_t) nx;
    565 +                 row3 = row2 + (uint64_t) nx;
```

```
565  566              for (int x = 0; x < nx; x += 4)
566  567              {
567  568                  if (bIn + 3 > comp_buf_size) return EXR_ERR_OUT_OF_MEMORY;
```

Comments 0

Please sign in to comment.