Closed Issue created 3 months ago by **Maddie Stone**

Integer overflow in pixman_sample_floor_y leading to heap out-of-bounds write

There is an out-of-bounds write in rasterize_edges_8 due to an integer overflow in `pixman_sample_floor_y`.

```
pixman_sample_floor_y (pixman_fixed_t y, int n)
{
    pixman_fixed_t f = pixman_fixed_frac (y);
    pixman_fixed_t i = pixman_fixed_floor (y);

    f = DIV (f - pixman_fixed_e - Y_FRAC_FIRST (n), STEP_Y_SMALL (n)) * STEP_Y_SMALL (n) +
    Y_FRAC_FIRST (n);

    if (f < Y_FRAC_FIRST (n))
    {
        if (pixman_fixed_to_int (i) == 0x8000)  ←(1)
        {
            f = 0; /* saturate */
        }
        else
        {
            f = Y_FRAC_LAST (n);
            i -= pixman_fixed_1; ← (2)
        }
    }
    return (i | f);
}
```

The condition at (1) will never be true because if i = 0x80000000, then `pixman_fixed_to_int` would return 0xffff8000, not 0x8000. The subtraction at (2) would then overflow back to 0x7fffffff.

Using the example from the attached POC: let's say y=0x80000700. At (1), i= 0x80000000 and f=0xfffff778. So `pixman_fixed_to_int(i) = 0xffff8000`. Therefore the code falls into the else block and completes `i -= pixman_fixed_1` at (2) which causes i to overflow to 0x7fff0000. `pixman_sample_floor_y(0x80000700)` returns `0x7ffff777` instead of `0x80000000`.

`pixman_rasterize_trapezoid` then passes a much too large `b` to `pixman_rasterize_edges` leading to the heap out-of-bounds write in the `memset` in `rasterize_edges_8`.

===

PROOF OF CONCEPT

Tested as of commit `285b9a907caffeb979322e629d4e57aa42061b5a`.

Copy to pixman/pixman directory and build: `$gcc poc.c -ldl -fsanitize=address -o poc`

==

```
==473984==ERROR: AddressSanitizer: heap-buffer-overflow on address 0x61b0000009b1 at pc 0x7f269bb092
WRITE of size 10 at 0x61b0000009b1 thread T0
    #0 0x7f269bb0928d in __interceptor_memset ../../../../src/libsanitizer/sanitizer_common/sanitize
    #1 0x7f2698a8d421 in rasterize_edges_8 /usr/local/google/home/maddiestone/pixman/pixman/pixman-e
    #2 0x7f2698a8d421 in pixman_rasterize_edges_no_accessors /usr/local/google/home/maddiestone/pixm
    #3 0x7f2698a8d421 in pixman_rasterize_edges /usr/local/google/home/maddiestone/pixman/pixman/pix
    #4 0x7f2698ab3894 in pixman_rasterize_trapezoid /usr/local/google/home/maddiestone/pixman/pixman
    #5 0x55ca2f9ec7a1 in main (/usr/local/google/home/maddiestone/pixman/pixman/poc+0x17a1)
    #6 0x7f269b9147fc in __libc_start_main ../csu/libc-start.c:332
    #7 0x55ca2f9ec129 in _start (/usr/local/google/home/maddiestone/pixman/pixman/poc+0x1129)

0x61b0000009b1 is located 769 bytes to the right of 1584-byte region [0x61b000000080,0x61b0000006b0)
allocated by thread T0 here:
    #0 0x7f269bb7e987 in __interceptor_calloc ../../../../src/libsanitizer/asan/asan_malloc_linux.cp
    #1 0x7f2698a71209 in create_bits /usr/local/google/home/maddiestone/pixman/pixman/pixman-bits-im
    #2 0x7f2698a71209 in _pixman_bits_image_init /usr/local/google/home/maddiestone/pixman/pixman/pi

SUMMARY: AddressSanitizer: heap-buffer-overflow ../../../../src/libsanitizer/sanitizer_common/saniti
```

```
Shadow bytes around the buggy address:
  0x0c367fff80e0: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c367fff80f0: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c367fff8100: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c367fff8110: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c367fff8120: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
=>0x0c367fff8130: fa fa fa fa fa fa[fa]fa fa fa fa fa fa fa fa fa
  0x0c367fff8140: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c367fff8150: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c367fff8160: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c367fff8170: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c367fff8180: 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
  Shadow gap:              cc
==473984==ABORTING
```

This bug is subject to a 90-day disclosure deadline. If a fix for this issue is made available to users before the end of the 90-day deadline, this bug report will become public 30 days after the fix was made available. Otherwise, this bug report will become public at the deadline. **The scheduled deadline is 2022-11-03**. For more details, see the Project Zero vulnerability disclosure policy: https://googleprojectzero.blogspot.com/p/vulnerability-disclosure-policy.html

poc.c

Drag your designs here or click to upload.

Tasks 0

No tasks are currently assigned. Use tasks to break down this issue into smaller parts.

Linked items 0

Activity

Maddie Stone @maddiestone · 3 months ago — Author

Hi folks, Can you please confirm that you've received this? Thanks!

Matt Turner @mattst88 · 3 months ago — Owner

Yes, received. Thank you!

Maddie Stone @maddiestone · 1 month ago — Author

Hey folks, We're one month out from the deadline. Do you expect to have a fix released by then? Thanks!

Edited by Maddie Stone 1 month ago

Matt Turner @mattst88 · 1 month ago Owner

I hope to have time to investigate this next week.

Matt Turner @mattst88 · 1 month ago Owner

Thanks for the POC. I can reproduce the issue.

Is it your expectation that

```
diff --git a/pixman/pixman-trap.c b/pixman/pixman-trap.c
index 91766fd..7560405 100644
--- a/pixman/pixman-trap.c
+++ b/pixman/pixman-trap.c
@@ -74,7 +74,7 @@ pixman_sample_floor_y (pixman_fixed_t y,

     if (f < Y_FRAC_FIRST (n))
     {
-        if (pixman_fixed_to_int (i) == 0x8000)
+        if (pixman_fixed_to_int (i) == 0xffff8000)
         {
             f = 0; /* saturate */
         }
```

is the appropriate fix?

With that, the failure is gone. Unfortunately the test suite doesn't test this path, as placing an `abort()` inside the `if` above the `f = 0;` doesn't trigger.

Maddie Stone @maddiestone · 1 month ago Author

Hi Matt, I believe that fix should work and address the oob write.

Thanks!

Maddie Stone @maddiestone · 4 weeks ago Author

Hey Matt, We're one week out from the deadline. Do you expect to have the fix released by then?

Matt Turner @mattst88 · 4 weeks ago Owner

Yes.

I don't know the process for announcing a security issue (and I don't actually know that this is a security issue?). What should I do?

I'm happy to push my patch and immediately make a point release.

Matt Turner closed via commit a1f88e84 3 weeks ago

Matt Turner made the issue visible to everyone 3 weeks ago

Matt Turner @mattst88 · 3 weeks ago Owner

This has been assigned CVE-2022-44638.

Please register or sign in to reply