New issue

Jump to bottom

heap-buffer-overflow in libsixel/src/quant.c:867 #25

Closed a4865g opened this issue on Sep 2, 2021 · 6 comments · Fixed by #26

Assignees

Labels bug

a4865g commented on Sep 2, 2021

Hi,I found a heap-buffer-overflow in the current master 705d991
It sames with the saitoha/libsixel/issue#156 (I found this problem 2 days ago)

OS: Ubuntu 20.04.3 LTS x86_64
Kernel: 5.11.0-27-generic

POC: poc.zip

It's the command line's report:

```
$ ./img2sixel -o ./a.sixel -7 -p 1 -C 5 -d stucki -E size poc
double free or corruption (out)
Aborted
```

and here is the ASAN report for saitoha/libsixel (the current master [`6a5be8b`]):

```
$ ./img2sixel -o ./a.sixel -7 -p 1 -C 5 -d stucki -E size ~/Downloads/poc
=================================================================
==2216856==ERROR: AddressSanitizer: heap-buffer-overflow on address 0x62e00000c3fd at pc 0x7ffff74e5a7e bp 0x7fffffffc340 sp 0x7fffffffc330
READ of size 1 at 0x62e00000c3fd thread T0
    #0 0x7ffff74e5a7d in error_diffuse /home/wulearn/Desktop/testtt/libsixel/src/quant.c:876
    #1 0x7ffff74e6027 in diffuse_stucki /home/wulearn/Desktop/testtt/libsixel/src/quant.c:1002
    #2 0x7ffff74e8154 in sixel_quant_apply_palette /home/wulearn/Desktop/testtt/libsixel/src/quant.c:1417
    #3 0x7ffff74eab2b in sixel_dither_apply_palette /home/wulearn/Desktop/testtt/libsixel/src/dither.c:801
    #4 0x7ffff74d9d9c in sixel_encode_dither /home/wulearn/Desktop/testtt/libsixel/src/tosixel.c:830
    #5 0x7ffff74e1c75 in sixel_encode /home/wulearn/Desktop/testtt/libsixel/src/tosixel.c:1551
    #6 0x7ffff7535f3b in sixel_encoder_output_without_macro /home/wulearn/Desktop/testtt/libsixel/src/encoder.c:825
    #7 0x7ffff75371e2 in sixel_encoder_encode_frame /home/wulearn/Desktop/testtt/libsixel/src/encoder.c:1056
    #8 0x7ffff753b0af in load_image_callback /home/wulearn/Desktop/testtt/libsixel/src/encoder.c:1679
    #9 0x7ffff752b085 in load_with_builtin /home/wulearn/Desktop/testtt/libsixel/src/loader.c:963
    #10 0x7ffff752b5cb in sixel_helper_load_image_file /home/wulearn/Desktop/testtt/libsixel/src/loader.c:1418
    #11 0x7ffff753b513 in sixel_encoder_encode /home/wulearn/Desktop/testtt/libsixel/src/encoder.c:1743
    #12 0x555555558a3b in main /home/wulearn/Desktop/testtt/libsixel/converters/img2sixel.c:457
    #13 0x7ffff72c60b2 in __libc_start_main (/usr/lib/x86_64-linux-gnu/libc.so.6+0x270b2)
    #14 0x55555555638d in _start (/home/wulearn/Desktop/testtt/libsixel/converters/.libs/img2sixel+0x238d)

0x62e00000c3fd is located 3 bytes to the left of 47208-byte region [0x62e00000c400,0x62e000017c68)
allocated by thread T0 here:
    #0 0x7ffff76a2bc8 in malloc (/usr/lib/x86_64-linux-gnu/libasan.so.5+0x10dbc8)
    #1 0x555555558c4e in rpl_malloc /home/wulearn/Desktop/testtt/libsixel/converters/malloc_stub.c:45
    #2 0x7ffff7549243 in sixel_allocator_malloc /home/wulearn/Desktop/testtt/libsixel/src/allocator.c:162
    #3 0x7ffff7535cab in sixel_encoder_output_without_macro /home/wulearn/Desktop/testtt/libsixel/src/encoder.c:789
    #4 0x7ffff75371e2 in sixel_encoder_encode_frame /home/wulearn/Desktop/testtt/libsixel/src/encoder.c:1056
    #5 0x7ffff753b0af in load_image_callback /home/wulearn/Desktop/testtt/libsixel/src/encoder.c:1679
    #6 0x7ffff752b085 in load_with_builtin /home/wulearn/Desktop/testtt/libsixel/src/loader.c:963
    #7 0x7ffff752b5cb in sixel_helper_load_image_file /home/wulearn/Desktop/testtt/libsixel/src/loader.c:1418
    #8 0x7ffff753b513 in sixel_encoder_encode /home/wulearn/Desktop/testtt/libsixel/src/encoder.c:1743
    #9 0x555555558a3b in main /home/wulearn/Desktop/testtt/libsixel/converters/img2sixel.c:457
    #10 0x7ffff72c60b2 in __libc_start_main (/usr/lib/x86_64-linux-gnu/libc.so.6+0x270b2)

SUMMARY: AddressSanitizer: heap-buffer-overflow /home/wulearn/Desktop/testtt/libsixel/src/quant.c:876 in error_diffuse
Shadow bytes around the buggy address:
  0x0c5c7fff9820: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c5c7fff9830: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c5c7fff9840: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c5c7fff9850: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c5c7fff9860: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
=>0x0c5c7fff9870: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa[fa]
  0x0c5c7fff9880: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x0c5c7fff9890: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x0c5c7fff98a0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x0c5c7fff98b0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x0c5c7fff98c0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
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
==2216856==ABORTING
```

It happens on:
https://github.com/saitoha/libsixel/blob/6a5be8b72d84037b83a5ea838e17bcf372ab1d5f/src/quant.c#L1002

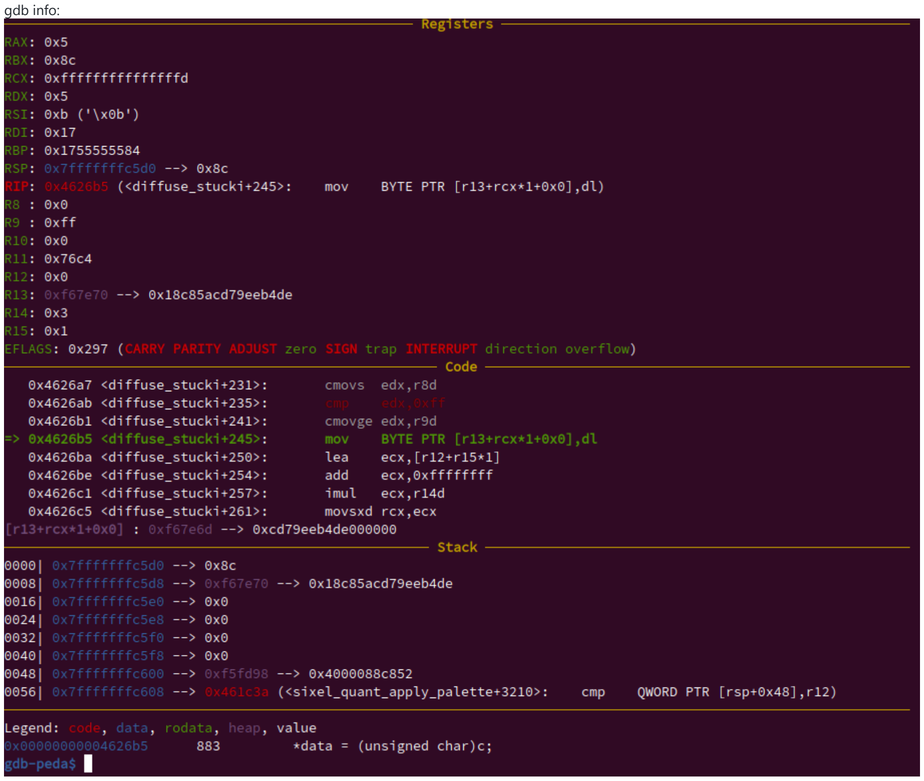
same with:

libsixel/src/quant.c
Line 993 in `705d991`

```
993      error_diffuse(data, pos + width * 1 - 2, depth, error, 1, 24);
```

when x=0, y=0, width=1,then

gdb info:



```
------------------------------------------------------------ Registers -------------------------------------------
RAX: 0x5
RBX: 0x8c
RCX: 0xfffffffffffffffd
RDX: 0x5
RSI: 0xb ('\x0b')
RDI: 0x17
RBP: 0x1755555584
RSP: 0x7fffffffc5d0 --> 0x8c
RIP: 0x4626b5 (<diffuse_stucki+245>:    mov    BYTE PTR [r13+rcx*1+0x0],dl)
R8 : 0x0
R9 : 0xff
R10: 0x0
R11: 0x76c4
R12: 0x0
R13: 0xf67e70 --> 0x18c85acd79eeb4de
R14: 0x3
R15: 0x1
EFLAGS: 0x297 (CARRY PARITY ADJUST zero SIGN trap INTERRUPT direction overflow)
--------------------------------------------------------------- Code ----------------------------------------------
   0x4626a7 <diffuse_stucki+231>:       cmovs  edx,r8d
   0x4626ab <diffuse_stucki+235>:       cmp    edx,0xff
   0x4626b1 <diffuse_stucki+241>:       cmovge edx,r9d
=> 0x4626b5 <diffuse_stucki+245>:       mov    BYTE PTR [r13+rcx*1+0x0],dl
   0x4626ba <diffuse_stucki+250>:       lea    ecx,[r12+r15*1]
   0x4626be <diffuse_stucki+254>:       add    ecx,0xffffffff
   0x4626c1 <diffuse_stucki+257>:       imul   ecx,r14d
   0x4626c5 <diffuse_stucki+261>:       movsxd rcx,ecx
[r13+rcx*1+0x0] : 0xf67e6d --> 0xcd79eeb4de000000
--------------------------------------------------------------- Stack ---------------------------------------------
0000| 0x7fffffffc5d0 --> 0x8c
0008| 0x7fffffffc5d8 --> 0xf67e70 --> 0x18c85acd79eeb4de
0016| 0x7fffffffc5e0 --> 0x0
0024| 0x7fffffffc5e8 --> 0x0
0032| 0x7fffffffc5f0 --> 0x0
0040| 0x7fffffffc5f8 --> 0x0
0048| 0x7fffffffc600 --> 0xf5fd98 --> 0x4000088c852
0056| 0x7fffffffc608 --> 0x461c3a (<sixel_quant_apply_palette+3210>:    cmp    QWORD PTR [rsp+0x48],r12)
-------------------------------------------------------------------------------------------------------------------
Legend: code, data, rodata, heap, value
0x00000000004626b5      883         *data = (unsigned char)c;
gdb-peda$
```

In this position,[r13+rcx*1+0x0] will be `0x100000000000f7e6d => 0xf7e6d`
So,writing to data will cause `overflow`
and then it writes to a location (chunk) in the heap that should not be written to.

heap info:
Before:

```
0xf5fda0         0x0              0x60             Freed             0x0             None
0xf5fe00         0x0              0x8060           Used             None             None
0xf67e60         0x0              0x70             Used             None             None
0xf67ed0         0x0              0x30             Used             None             None
0xf67f00         0x0              0x10010          Used             None             None
0xf77f10         0x0              0x7eb0           Freed    0x7ffff7c9cbe0   0x7ffff7c9cbe0
0xf7fdc0         0x7eb0           0xd0             Freed             0x0             None
```

After:

```
0xf5fda0         0x0              0x60             Freed             0x0             None
0xf5fe00         0x0              0x8060           Used             None             None
Corrupt ?!
```

dankamongmen commented on Sep 2, 2021 · Collaborator

great work! certainly looks valid to me. i'll try to have a fix up by tomorrow morning, but if anyone else wants to offer one in the meantime, i'll evaluate it. well done! =]

❤️ 1

dankamongmen self-assigned this on Sep 2, 2021

dankamongmen added the bug label on Sep 2, 2021

dankamongmen commented on Sep 3, 2021 · Collaborator

alright, so the fix seems pretty simple -- don't diffuse into the void.

dankamongmen mentioned this issue on Sep 3, 2021

[error_diffuse] don't diffuse into the void #26

Merged

dankamongmen commented on Sep 3, 2021 Collaborator

I've put up a PR which I'm pretty sure closes this problem. I still see plenty of uses of uninitialized data in a valgrind run, but no longer any invalid accesses.

dankamongmen closed this as completed in #26 on Sep 3, 2021

dankamongmen commented on Sep 3, 2021 Collaborator

Fix pushed in 1.10.0.

❤️ 1

a4865g commented on Sep 3, 2021 Author

Could I try to submit this problem to get CVE ID?

dankamongmen commented on Sep 3, 2021 Collaborator

> Could I try to submit this problem to get CVE ID?

you can do whatever you like, though i doubt it's very easy to turn this into an exploit.

👍 1

Assignees

dankamongmen

Labels

bug

Projects

None yet

Milestone

No milestone

Development

Successfully merging a pull request may close this issue.

[error_diffuse] don't diffuse into the void

2 participants