Bug 704945 - Null pointer dereference in gx_default_create_buf_device()

Status: RESOLVED FIXED

Alias: None

Product: Ghostscript
Component: Fuzzing (show other bugs)
Version: 9.55.0
Hardware: PC Linux

Importance: P4 normal
Assignee: Robin Watts

URL:
Keywords:

Depends on:
Blocks:

Reported: 2022-02-14 12:11 UTC by zhailiangliang
Modified: 2022-02-16 17:02 UTC (History)
CC List: 1 user (show)

See Also:
Customer:
Word Size: ---

Attachments	
poc file (130.71 KB, application/pdf) 2022-02-14 12:11 UTC, zhailiangliang	Details
Add an attachment (proposed patch, testcase, etc.)	

Note
You need to log in before you can comment on or make changes to this bug.

zhailiangliang 2022-02-14 12:11:49 UTC **Description**

```
Created attachment 22082 [details]
poc file

Hello.

I found a NULL pointer dereference bug in gx_default_create_buf_device
Please confirm.
Thanks.

1. steps to reproduce:
OS: Ubuntu 20.04.3 LTS
Version: ghostscript-9.55.0
Steps to reproduce:
1. Download the .POC files.
2. Compile the source code with ASan.
3. gs -dBATCH -dNOPAUSE -dSAFER -r2 -sOutputFile=tmp -sDEVICE=devicen $PoC

2. proof of concept
AddressSanitizer:DEADLYSIGNAL
=================================================================
==196052==ERROR: AddressSanitizer: SEGV on unknown address 0x000000000000 (pc
0x000000000000 bp 0x7ffd61c808a0 sp 0x7ffd61c806d8 T0)
==196052==Hint: pc points to the zero page.
==196052==The signal is caused by a READ memory access.
==196052==Hint: address points to the zero page.

AddressSanitizer can not provide additional info.
SUMMARY: AddressSanitizer: SEGV (<unknown module>)
==196052==ABORTING

bt
#0  0x0000000000000000 in ?? ()
#1  0x0000555555ca8b74 in gx_default_create_buf_device (pbdev=0x7fffffffbbc0,
target=0x62a00033c268, y=0, render_plane=0x0, mem=0x0, color_usage=<optimized out>)
at ./base/gdevprn.c:1399
#2  0x0000555555cab9ff in gdev_create_buf_device (color_usage=0x0, mem=0x0,
render_plane=0x0, y=0, target=0x62a00033c268, pbdev=0x7fffffffbbc0, cbd_proc=
<optimized out>)
    at ./base/gdevprn.c:1352
#3  gdev_prn_allocate (pdev=pdev@entry=0x62a00033c268,
new_space_params=new_space_params@entry=0x0, new_width=new_width@entry=0,
new_height=new_height@entry=0,
    reallocate=reallocate@entry=0) at ./base/gdevprn.c:446
#4  0x0000555555cacc53 in gdev_prn_allocate_memory (new_height=0, new_width=0,
new_space_params=0x0, pdev=0x62a00033c268) at ./base/gdevprn.c:510
#5  gdev_prn_open (pdev=<optimized out>, pdev@entry=0x62a00033c268) at
./base/gdevprn.c:92
#6  0x0000555555c92ff3 in spotcmyk_prn_open (pdev=0x62a00033c268) at
./base/gdevdevn.c:1054
#7  0x0000555555cd3b1a in default_subclass_open_device ()
#8  0x0000555556e47f21 in gs_opendevice (dev=dev@entry=0x62a000054268) at
./base/gsdevice.c:461
#9  0x0000555556e49758 in gs_opendevice (dev=0x62a000054268) at
./base/gsdevice.c:594
#10 gs_setdevice_no_erase (pgs=0x62a00000c558, dev=dev@entry=0x62a000054268) at
./base/gsdevice.c:580
```

```
#11 0x00005555572dd9cc in zputdeviceparams (i_ctx_p=0x62a00000c268) at
./psi/zdevice.c:495
#12 0x000055555724b6f3 in interp (perror_object=0x7fffffffd540, pref=<optimized
out>, pi_ctx_p=0x613000000110) at ./psi/interp.c:1722
#13 gs_call_interp (pi_ctx_p=pi_ctx_p@entry=0x613000000110,
pref=pref@entry=0x7fffffffd460, user_errors=user_errors@entry=1,
pexit_code=pexit_code@entry=0x7fffffffd530,
    perror_object=<optimized out>) at ./psi/interp.c:520
#14 0x0000555557250e0b in gs_interpret (pi_ctx_p=pi_ctx_p@entry=0x613000000110,
pref=0x7fffffffd460, user_errors=user_errors@entry=1,
pexit_code=pexit_code@entry=0x7fffffffd530,
    perror_object=perror_object@entry=0x7fffffffd540) at ./psi/interp.c:477
#15 0x0000555557227276 in gs_main_interpret (perror_object=0x7fffffffd540,
pexit_code=0x7fffffffd530, user_errors=1, pref=0x7fffffffd460,
minst=0x613000000070) at ./psi/imain.c:257
#16 gs_main_run_string_end (perror_object=0x7fffffffd540,
pexit_code=0x7fffffffd530, user_errors=1, minst=0x613000000070) at
./psi/imain.c:945
#17 gs_main_run_string_with_length (minst=minst@entry=0x613000000070, str=
<optimized out>, length=<optimized out>, user_errors=1,
pexit_code=pexit_code@entry=0x7fffffffd530,
    perror_object=perror_object@entry=0x7fffffffd540) at ./psi/imain.c:889
#18 0x0000555557227362 in gs_main_run_string (minst=minst@entry=0x613000000070,
str=<optimized out>, user_errors=<optimized out>,
pexit_code=pexit_code@entry=0x7fffffffd530,
    perror_object=perror_object@entry=0x7fffffffd540) at ./psi/imain.c:870
#19 0x000055555722e797 in run_string (minst=minst@entry=0x613000000070,
    str=str@entry=0x610000000270 "
<2f686f6d652f7a6c6c2f67686f73747363726970742d392e35352e302f657874726163742f746573742f746578745f677261706869635f696d6167652e706466>.runfile"

    options=options@entry=3, user_errors=user_errors@entry=1,
pexit_code=0x7fffffffd530, pexit_code@entry=0x0, perror_object=0x7fffffffd540,
perror_object@entry=0x0)
    at ./psi/imainarg.c:1166
#20 0x000055555722ebc9 in runarg (minst=minst@entry=0x613000000070,
arg=arg@entry=0x7fffffffd758 "/home/zll/ghostscript-
9.55.0/extract/test/text_graphic_image.pdf",
    post=post@entry=0x5555580e8060 ".runfile", options=options@entry=3,
user_errors=1, pexit_code=pexit_code@entry=0x0, perror_object=0x0,
pre=0x5555580e7be0 "") at ./psi/imainarg.c:1125
#21 0x000055555722f2fe in argproc (arg=0x7fffffffd758 "/home/zll/ghostscript-
9.55.0/extract/test/text_graphic_image.pdf", minst=0x613000000070) at
./psi/imainarg.c:1047
#22 argproc (minst=0x613000000070, arg=0x7fffffffd758 "/home/zll/ghostscript-
9.55.0/extract/test/text_graphic_image.pdf") at ./psi/imainarg.c:1032
#23 0x00005555572332cd in gs_main_init_with_args01
(minst=minst@entry=0x613000000070, argc=<optimized out>, argv=<optimized out>) at
./psi/imainarg.c:242
#24 0x0000555557233abd in gs_main_init_with_args (minst=0x613000000070, argc=
<optimized out>, argv=<optimized out>) at ./psi/imainarg.c:289
#25 0x00005555559fe9b3 in main (argc=8, argv=<optimized out>) at ./psi/gs.c:95

3. repair advice:
diff --git a/base/gdevprn.c b/base/gdevprn.c
index a78a28cbf..f9c0f19e1 100644
--- a/base/gdevprn.c
+++ b/base/gdevprn.c
@@ -1405,6 +1405,9 @@ gx_default_create_buf_device(gx_device **pbdev, gx_device
*target, int y,
         dev_t_proc_dev_spec_op((*orig_dso), gx_device) = dev_proc(mdev,
dev_spec_op);
         /* The following is a special hack for setting up printer devices. */
         assign_dev_procs(mdev, mdproto);
+        if (mdproto->initialize_device_procs == NULL) {
+            return_error(gs_error_rangecheck);
+        }
         mdev->initialize_device_procs = mdproto->initialize_device_procs;
         mdev->initialize_device_procs((gx_device *)mdev);
         /* We know mdev->procs.initialize_device is NULL! */
```

Ken Sharp 2022-02-14 13:37:41 UTC **Comment 1**

```
This does indeed seg fault on both Windows and Linux, including on current master.

The problem is something to do with trying to render a large number of bits in
memory. In gx_default_create_buf_device() if 'mem' is NULL (a signal from
gdev_prn_allocate that we should render entirely in memory, see line 442 of
gdevprn.c) then at line 1402 we set mdev to be *pbdev, which in this case means
that target and mdev are the same.

Since they are the same (devicen) we pick up the initialise routine from the
mdproto prototype. In this case that prototype is 'mem_x_device' (see
base/gdevmx.h) which is some kind of special device prototype as it has **no**
function prototypes at all.

We don't check device function pointers for NULL, because device function tables
are not supposed to contain NULL (with the special exception of fill_rectangle).

And at this point, I'm at a loss.....
```

Robin Watts 2022-02-16 17:02:54 UTC **Comment 2**

```
Fixed in:

commit ae1061d948d88667bdf51d47d918c4684d0f67df
```

```
Author: Robin Watts <Robin.Watts@artifex.com>
Date:   Wed Feb 16 15:22:50 2022 +0000

    ~~Bug 704945~~: Add init_device_procs entry for mem_x_device.

    When allocating a buffer device, we rely on an init_device_procs
    being defined for the device we are using as a prototype. Which
    device we use as a prototype depends upon the number of bits per
    pixel we are using. For bpp > 64, we use mem_x_device, which does
    not currently have an init_device_procs defined.

    This is a fairly hard case to tickle, as very few devices use
    more than 64 bits per pixel. The DeviceN device is one of the
    few that does, and then the problem only kicks in if the
    MaxBitmap figure is high enough (or conversely the resolution is
    low enough).
```