Bug 701791 - global-buffer-overflow at devices/gdevpjet.c:177 in pj_common_print_page

Status:	RESOLVED FIXED	**Reported:**	2019-10-26 06:14 UTC by Suhwan
		Modified:	2019-10-26 14:08 UTC (History)
Alias:	None	**CC List:**	0 users
Product:	Ghostscript		
Component:	General (show other bugs)	**See Also:**	
Version:	master	**Customer:**	
Hardware:	PC Linux	**Word Size:**	---
Importance:	P4 normal		
Assignee:	Default assignee		
URL:			
Keywords:			
Depends on:			
Blocks:			

Attachments	
poc (16.24 KB, application/pdf) 2019-10-26 06:14 UTC, Suhwan	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.

Suhwan 2019-10-26 06:14:54 UTC **Description**

```
Created attachment 18375 [details]
poc

Hello.

I found a global-buffer-overflow bug in GhostScript.

Please confirm.

Thanks.

OS: Ubuntu 18.04 64bit

Steps to reproduce:
1. Download the .POC files.
2. Compile the source code with ASan.
3. Run following cmd.

gs -sOutputFile=tmp -sDEVICE=pjetxl $PoC

Here's ASAN report.
=================================================================
==35530==ERROR: AddressSanitizer: global-buffer-overflow on address 0x0000042b13d0
at pc 0x000001f6bc40 bp 0x7fff15776b70 sp 0x7fff15776b68
READ of size 8 at 0x0000042b13d0 thread T0
    #0 0x1f6bc3f in pj_common_print_page ghostpdl/./devices/gdevpjet.c:177:39
    #1 0x13f07d9 in gx_default_print_page_copies ghostpdl/./base/gdevprn.c:1231:12
    #2 0x13ef028 in gdev_prn_output_page_aux ghostpdl/./base/gdevprn.c:1133:27
    #3 0x2bc0289 in gx_forward_output_page ghostpdl/./base/gdevnfwd.c:184:10
    #4 0x22b6f20 in gs_output_page ghostpdl/./base/gsdevice.c:212:17
    #5 0x3054b9f in zoutputpage ghostpdl/./psi/zdevice.c:416:12
    #6 0x2e8bdb6 in interp ghostpdl/./psi/interp.c:1300:28
    #7 0x2e8bdb6 in gs_call_interp ghostpdl/./psi/interp.c:520
    #8 0x2e8bdb6 in gs_interpret ghostpdl/./psi/interp.c:477
    #9 0x2e3f451 in gs_main_interpret ghostpdl/./psi/imain.c:253:12
    #10 0x2e3f451 in gs_main_run_string_end ghostpdl/./psi/imain.c:791
    #11 0x2e3f451 in gs_main_run_string_with_length ghostpdl/./psi/imain.c:735
    #12 0x2e548f0 in run_string ghostpdl/./psi/imainarg.c:1117:12
    #13 0x2e548f0 in runarg ghostpdl/./psi/imainarg.c:1086
    #14 0x2e5302a in argproc ghostpdl/./psi/imainarg.c:1008:16
    #15 0x2e479f7 in gs_main_init_with_args01 ghostpdl/./psi/imainarg.c:241:24
    #16 0x2e539d0 in gs_main_init_with_args ghostpdl/./psi/imainarg.c:288:16
    #17 0x57b86f in main ghostpdl/./psi/gs.c:95:16
    #18 0x7ffa45c04b96 in __libc_start_main /build/glibc-OTsEL5/glibc-
2.27/csu/../csu/libc-start.c:310
    #19 0x482e79 in _start (gs+0x482e79)

0x0000042b13d0 is located 16 bytes to the left of global variable '<string
literal>' defined in './devices/gdevl31s.c:34:70' (0x42b13e0) of size 3
  '<string literal>' is ascii string 'dl'
0x0000042b13d0 is located 45 bytes to the right of global variable '<string
literal>' defined in './devices/gdevl31s.c:34:63' (0x42b13a0) of size 3
  '<string literal>' is ascii string 'c5'
SUMMARY: AddressSanitizer: global-buffer-overflow
ghostpdl/./devices/gdevpjet.c:177:39 in pj_common_print_page
Shadow bytes around the buggy address:
  0x00008084e220: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x00008084e230: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x00008084e240: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x00008084e250: 00 00 00 00 03 f9 f9 f9 f9 f9 f9 f9 07 f9 f9 f9
  0x00008084e260: f9 f9 f9 f9 06 f9 f9 f9 f9 f9 f9 f9 06 f9 f9 f9
=>0x00008084e270: f9 f9 f9 f9 03 f9 f9 f9 f9 f9[f9]f9 03 f9 f9 f9
  0x00008084e280: f9 f9 f9 f9 03 f9 f9 f9 f9 f9 f9 f9 00 f9 f9 f9
  0x00008084e290: f9 f9 f9 f9 00 02 f9 f9 f9 f9 f9 f9 00 00 00 00
  0x00008084e2a0: 00 00 00 00 00 00 00 f9 f9 f9 f9 f9 f9 00 f9 f9 f9
  0x00008084e2b0: f9 f9 f9 f9 00 01 f9 f9 f9 f9 f9 f9 00 03 f9 f9
  0x00008084e2c0: f9 f9 f9 f9 05 f9 f9 f9 f9 f9 f9 f9 00 00 04 f9
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
==35530==ABORTING
```

Ken Sharp 2019-10-26 13:58:28 UTC **Comment 1**

```
The line cuaing the error is:

                                      (spr40[dp[2]] >> 1) +
When i is 1528 (ie its the last 'chunk' because DATA_SIZE is 1536 and we stop when
i >= DATA_SIZE).

The problem is that we copy 'line_size' bytes from the device into the line buffer:

                        gdev_prn_copy_scan_lines(pdev, lnum,
                                                 (byte *)data, line_size);

where line_size in this case is 1530 bytes. But we transpose the data using a limit
of 'DATA_SIZE' for i:

                                /* Transpose the data to get pixel planes. */
                                for ( i = 0, odp = plane_data; i < DATA_SIZE;
                                      i += 8, odp++
                                    )
And Data_SIZE is 1536 bytes. Which means we read off the end of the buffer by 6
bytes, which means we are using uninitialised data. Its the use of uninitialised
data which causes the buffer overflow, trying to read 95 bytes along an array of 8
bytes.

We simply need to set those bytes to 0. Its possible there is code which is trying
to do that already:

                                /* Pad with 0s to fill out the last */
                                /* block of 8 bytes. */
                                memset(end_data, 0, 7);

But if that's what its doing, its in the wrong place, end_data is decremented
before we reach here if the line has any white space at the right edge. Which means
that code is simply overwriting 0x00 bytes with more 0x00 bytes.

So I've chosen to simply set the entire buffer to 0x00 before we start copying and
processing, its a one-time setup cost so its probably cheaper than trying to clean
the buffer on every line.

While this is probably benign (the data we are writing is unused), its poor
practice at best.
```

Ken Sharp **2019-10-26 14:08:26 UTC** **Comment 2**

```
Fixed in commit aba3375ac24f8e02659d9b1eb9093909618cdb9f
```