



Bug 2437675 (CVE-2026-2239) - CVE-2026-2239 gimp: GIMP: Application crash (DoS) via crafted PSD file due to heap-buffer-overflow

Keywords: Security

Reported: 2026-02-09 09:07 UTC by OSIDB Bzimport

Status: NEW

Modified: 2026-02-09 09:24 UTC ([History](#))

Alias: CVE-2026-2239

CC List: 0 users

Product: Security Response

Fixed In Version:

Component: vulnerability

Clone Of:

Version: unspecified

Environment:

Hardware: All

Last Closed:

OS: Linux

Embargoed:

Priority: low

Severity: low

Target Milestone: ---

Assignee: Product Security DevOps Team

QA Contact:

Docs Contact:

URL:

Whiteboard:

Depends On: [2437676](#) [2437677](#)

Blocks:

TreeView+ [depends on](#) / [blocked](#)

Attachments ([Terms of Use](#))

OSIDB Bzimport 2026-02-09 09:07:09 UTC

[Description](#)

Description of the bug

In plug-ins/file-psd/psd-util.c, the function fread_pascal_string() allocates a buffer with g_malloc(len) at line 277 and reads len bytes from the file into it. The buffer is not null-terminated. It is then passed to gimp_any_to_utf8(str, len, NULL) at line 302. Downstream (e.g. in code that treats the result as a C string, or in a stub that calls g_strdup()), strlen() is used on memory that is not guaranteed to be null-terminated. For a crafted PSD where the pascal string has length 14 and the 14 bytes do not contain a null, strlen() reads past the end of the 14-byte region, causing a heap-buffer-overflow (READ of size 15 at address 0

bytes after the 14-byte region). This can crash the application when loading a malformed PSD (e.g. when reading layer names in `read_layer_info` at `psd-load.c:1245`). Denial-of-service when opening such a file.

Component: `plug-ins/file-psd/psd-util.c` (`fread_pascal_string`), used from `read_layer_info` (`psd-load.c:1245`).

CWE: Heap-buffer-overflow (missing null terminator / use of buffer as C string). Severity: Medium (DoS).

Reproduction

Is the bug reproducible? Yes.

Reproduction steps:

Build GIMP with AddressSanitizer (ASan).

Open the attached PoC PSD file in GIMP (File → Open).

Expected result: Parser should either null-terminate the buffer before passing it to code that expects a C string, or the code that uses the string should respect the length and not call `strlen()` on the raw buffer. No read past the end of the allocation.

Actual result: AddressSanitizer reports `heap-buffer-overflow: READ` of size 15 at address 0 bytes after a 14-byte region, in `strlen` → `g_strdup` → `gimp_any_to_utf8` → `fread_pascal_string` (`psd-util.c:302`). Application aborts.

Additional information

Suggested fix: In `fread_pascal_string()` in `psd-util.c`, after reading `len` bytes into `str`, either (1) allocate `len+1` and set `str[len] = '\0'` before passing to `gimp_any_to_utf8`, or (2) ensure every caller of `gimp_any_to_utf8` and downstream code use the length parameter and never assume null termination of the raw buffer.

Note: The backtrace below was obtained with a fuzzing harness (`gimp_any_to_utf8` in `stubs`). In vanilla GIMP the same heap overflow occurs in `fread_pascal_string` (`psd-util.c:302`) when loading the PoC.

AddressSanitizer output (full):

```
=1869886==ERROR: AddressSanitizer: heap-buffer-overflow on
address 0x502000020a5e at pc 0x76f1afa7d96f bp 0x7ffea58bf1d0
sp 0x7ffea58be978
READ of size 15 at 0x502000020a5e thread T0
#0 0x76f1afa7d96e in strlen
../../../../src/libsanitizer/sanitizer_common/sanitizer_common
_interceptors.inc:391
#1 0x76f1aeb843c in g_strdup (/lib/x86_64-linux-
gnu/libglib-2.0.so.0+0x7843c) (BuildId:
2b4ac191b9964e6c52d861865b20c49b20916169)
#2 0x620787d0a1fd in g_strdup_inline /usr/include/glib-
2.0/glib/gstrfuncs.h:321
#3 0x620787d0a1fd in gimp_any_to_utf8
/home/wjddn0623/fuzzing/harness/gimp_stubs.c:242
#4 0x620787d28ed4 in fread_pascal_string
/home/wjddn0623/fuzzing/gimp/plug-ins/file-psd/psd-util.c:302
#5 0x620787d0e374 in read_layer_info
/home/wjddn0623/fuzzing/gimp/plug-ins/file-psd/psd-load.c:1245
#6 0x620787d0ebc2 in read_layer_block
/home/wjddn0623/fuzzing/gimp/plug-ins/file-psd/psd-load.c:1406
#7 0x620787d1b4d1 in load_image
/home/wjddn0623/fuzzing/gimp/plug-ins/file-psd/psd-load.c:234
#8 0x620787d094d8 in main
/home/wjddn0623/fuzzing/harness/fuzz_psd.c:56
```

```

#9 0x76f1ae42a1c9 in __libc_start_call_main
../sysdeps/nptl/libc_start_call_main.h:58
#10 0x76f1ae42a28a in __libc_start_main_impl ../csu/libc-
start.c:360
#11 0x620787d09224 in _start
(/home/wjddn0623/fuzzing/harness/fuzz_psd+0xe224) (BuildId:
03de7aaed914728e879387867c3bcd01829e511c)

0x502000020a5e is located 0 bytes after 14-byte region
[0x502000020a50,0x502000020a5e)
allocated by thread T0 here:
#0 0x76f1afafd9c7 in malloc
../../../../src/libsanitizer/asan/asan_malloc_linux.cpp:69
#1 0x76f1aebd2a79 in g_malloc (/lib/x86_64-linux-
gnu/libglib-2.0.so.0+0x62a79) (BuildId:
2b4ac191b9964e6c52d861865b20c49b20916169)
#2 0x620787d28d33 in fread_pascal_string
/home/wjddn0623/fuzzing/gimp/plug-ins/file-psd/psd-util.c:277
#3 0x620787d0e374 in read_layer_info
/home/wjddn0623/fuzzing/gimp/plug-ins/file-psd/psd-load.c:1245
#4 0x620787d0ebc2 in read_layer_block
/home/wjddn0623/fuzzing/gimp/plug-ins/file-psd/psd-load.c:1406
#5 0x620787d1b4d1 in load_image
/home/wjddn0623/fuzzing/gimp/plug-ins/file-psd/psd-load.c:234
#6 0x620787d094d8 in main
/home/wjddn0623/fuzzing/harness/fuzz_psd.c:56
#7 0x76f1ae42a1c9 in __libc_start_call_main
../sysdeps/nptl/libc_start_call_main.h:58
#8 0x76f1ae42a28a in __libc_start_main_impl ../csu/libc-
start.c:360
#9 0x620787d09224 in _start
(/home/wjddn0623/fuzzing/harness/fuzz_psd+0xe224) (BuildId:
03de7aaed914728e879387867c3bcd01829e511c)

```

```

SUMMARY: AddressSanitizer: heap-buffer-overflow
../../../../src/libsanitizer/sanitizer_common/sanitizer_common
_interceptors.inc:391 in strlen
Shadow bytes around the buggy address:
 0x502000020780: fa fa 00 00 fa fa fd fd fa fa 00 00 fa fa fd
fd
 0x502000020800: fa fa 00 fa fa fa 00 00 fa fa 00 fa fa fa 00
00
 0x502000020880: fa fa 00 00 fa fa fd fd fa fa 00 00 fa fa fd
fd
 0x502000020900: fa fa 00 fa fa fa 00 00 fa fa fd fd fa fa fd
fd
 0x502000020980: fa fa fd fd fa fa fd fa fa fd fd fa fa fd
fd
=>0x502000020a00: fa fa fd fa fa fa 00 fa fa fa 00[06]fa fa fa
fa
 0x502000020a80: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
fa
 0x502000020b00: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
fa
 0x502000020b80: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
fa
 0x502000020c00: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
fa
 0x502000020c80: 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
==1869886==ABORTING
```

Note

You need to [log in](#) before you can comment on or make changes to this bug.

