Use-after-free via pointer aliasing in `png_set_tRNS` and `png_set_PLTE`

High **ctruta** published **GHSA-m4pc-p4q3-4c7j** last week

Package

libpng

Affected versions

>= 1.2.1, <= 1.6.55

Patched versions

1.6.56, 1.8.0 (trunk)

Description

Summary

Use-after-free via pointer aliasing between `png_struct` and `png_info` in `png_set_tRNS` and `png_set_PLTE`.

Description

In libpng versions through 1.6.55, `png_set_tRNS` and `png_set_PLTE` each alias a heap-allocated buffer between `png_struct` and `png_info`, sharing a single allocation across two structs with independent lifetimes. The `trans_alpha` aliasing has been present since at least libpng 1.0, and the `palette` aliasing since at least 1.2.1. Both affect all prior release lines:

- `png_set_tRNS` sets `png_ptr->trans_alpha = info_ptr->trans_alpha` (256-byte buffer).
- `png_set_PLTE` sets `info_ptr->palette = png_ptr->palette` (768-byte buffer).

In both cases, calling `png_free_data` (with `PNG_FREE_TRNS` or `PNG_FREE_PLTE`) frees the buffer through `info_ptr` while the corresponding `png_ptr` pointer remains dangling. Subsequent row-transform functions dereference and, in some code paths, write to the freed memory. A second call to `png_set_tRNS` or `png_set_PLTE` has the same effect, because both functions call `png_free_data` internally before reallocating the `info_ptr` buffer.

`trans_alpha`

In `png_set_tRNS`, the pointer aliasing is a direct assignment:

```
png_ptr->trans_alpha = info_ptr->trans_alpha;  /* pointer aliasing */
```

An application that calls `png_free_data(png_ptr, info_ptr, PNG_FREE_TRNS, 0)` or `png_set_tRNS` a second time after `png_read_info`, and then calls `png_read_update_info`, triggers a use-after-free in `png_init_palette_transformations`, which reads from; and, when `PNG_READ_INVERT_ALPHA` is enabled, writes to the freed 256-byte buffer:

```
/* png_init_palette_transformations --- reads from freed buffer */
if (png_ptr->trans_alpha[i] == 255)

/* png_init_palette_transformations --- writes to freed buffer */
for (i = 0; i < istop; i++)
    png_ptr->trans_alpha[i] = (png_byte)(255 - png_ptr->trans_alpha[i]);
```

The attacker controls the tRNS chunk values in the PNG file. The complement operation `255 - x` is an involution: the values written to the freed buffer are deterministically controlled by the attacker. Per-row transforms such as `png_do_expand_palette` subsequently dereference the same dangling pointer.

`palette`

In `png_set_PLTE`, the pointer aliasing is an assignment in the opposite direction:

```
info_ptr->palette = png_ptr->palette;  /* pointer aliasing */
```

Additionally, `png_set_quantize` aliases `png_ptr->palette` directly to a caller-supplied pointer (`png_ptr->palette = palette`). This is a distinct variant: not `info_ptr` / `png_ptr` cross-aliasing, but aliasing to an externally owned buffer. The fix for the `palette` UAF makes `png_read_destroy` free `png_ptr->palette` unconditionally; without also fixing `png_set_quantize`, that unconditional free would free unmanaged memory.

An application that calls `png_free_data(png_ptr, info_ptr, PNG_FREE_PLTE, 0)` or `png_set_PLTE` a second time after reading a palette-based PNG, and then continues decoding, triggers reads of up to 768 bytes of freed heap memory per row in `png_do_expand_palette`, gamma/background compositing, and the ARM Neon palette riffle. The bit-shift transforms perform read-modify-write on the freed buffer, enabling heap corruption.

Impact

- **Read-after-free**: Freed heap memory read via the dangling pointer in row-transform functions. This may leak sensitive heap contents.

- **Write-after-free**: The `PNG_READ_INVERT_ALPHA` complement loop (`trans_alpha`) and the bit-shift transforms (`palette`) write attacker-influenced values to freed heap memory, enabling heap corruption. Because the complement `255 - x` is an involution, the attacker fully controls the values written to the freed buffer by choosing the corresponding tRNS chunk entries.
- **Arbitrary code execution (demonstrated)**: On allocators with deterministic reuse (e.g., glibc tcache, default since glibc 2.26+), the freed 256-byte buffer is returned by the next same-sized `malloc` . If the application places a struct containing function pointers or security-critical data in this allocation, uninitialized fields may retain attacker-controlled values written by the UAF, enabling control-flow hijack. This has been demonstrated with a proof-of-concept RCE exploit in `-no-pie` environments (common in embedded systems and legacy servers). In PIE/ASLR-enabled environments, exploitation requires an additional information leak vulnerability, or brute-forcing (practical on 32-bit systems).
- **Affected application pattern**: Applications that call `png_free_data()` to release memory between `png_read_info()` and `png_read_update_info()` are affected. This is a reasonable optimization pattern for memory-constrained embedded systems and server-side image processing pipelines.
- **Attack realism**: The crafted PNG is 100% standards-compliant and passes all PNG validators. No upstream filter or WAF can reject the payload without also rejecting valid PNG files.

Fix

Give `png_struct` its own independently allocated copies of both `trans_alpha` and `palette` , decoupling the lifetimes from `png_info` .

Fixing `trans_alpha`

Summary of commit `2301926` :

- **pngset.c**: Allocate an independent `png_ptr->trans_alpha` buffer via `png_malloc` + `memcpy` ; free the old buffer before reallocating.
- **pngread.c**: Free `png_ptr->trans_alpha` unconditionally in `png_read_destroy` .
- **pngwrite.c**: Free `png_ptr->trans_alpha` in `png_write_destroy` .
- **pngrutil.c**: Remove the TODO comment in `png_handle_tRNS` that flagged the aliasing side effect.

Post-fixing `trans_alpha` : defence in depth

Summary of commit `a3a2144` :

- Initialize all 256 bytes of both `trans_alpha` buffers to 0xff (fully opaque) via `memset` before `memcpy` , eliminating uninitialized-memory reads if a consumer indexes beyond `num_trans` .

Fixing `palette`

Summary of commit `7ea9eea` :

- **pngset.c**: Allocate independent `info_ptr->palette` and `png_ptr->palette` buffers via `png_calloc`; free the old `png_ptr->palette` before reallocating. Zero-fill is required because the ARM Neon palette riffle reads all 256 entries unconditionally.
- **pngread.c**: Free `png_ptr->palette` unconditionally in `png_read_destroy` (remove dead `free_me` conditional).
- **pngrtran.c**: In `png_set_quantize`, allocate an owned copy of the caller's palette instead of aliasing the user pointer, so that the unconditional free in `png_read_destroy` does not free unmanaged memory.
- **pngwrite.c**: Free `png_ptr->palette` in `png_write_destroy`.
- **pngrutil.c**: Remove the TODO/REVIEW/CONSIDER block in `png_handle_PLTE` that discussed the palette sharing problem.

Summary of commit `c1b0318` :

- **pngrtran.c**: In `png_read_transform_info`, copy `png_ptr->palette` into `info_ptr->palette` after `png_init_read_transformations` has applied in-place transforms (gamma correction, background compositing). Without this sync, `png_get_PLTE` returned the original untransformed palette, breaking indexed-colour background compositing.

Credits

- Halil Oktay (**@Oblivionsage**): Discovered the `trans_alpha` aliasing vulnerability and contributed the fix (PR #824).
- Ryo Shimada (**@shimarda**, University of Tsukuba; Powder Keg Technologies, Inc.): Independently discovered the `trans_alpha` aliasing vulnerability and demonstrated exploitability with ASan and RCE proof-of-concept exploits.

References

- Fix (`trans_alpha`): commit `2301926`
- Defence in depth (`trans_alpha`): commit `a3a2144`
- Fix (`palette`): commits `7ea9eea` and `c1b0318`
- GitHub PR #824

Severity

High 7.5 / 10

CVSS v3 base metrics

Attack vector	Network
Attack complexity	High

Privileges required	None
User interaction	Required
Scope	Unchanged
Confidentiality	High
Integrity	High
Availability	High

Learn more about base metrics

CVSS:3.1/AV:N/AC:H/PR:N/UI:R/S:U/C:H/I:H/A:H

CVE ID

CVE-2026-33416

Weaknesses

▶ CWE-416

Credits

shimarda	Reporter
Oblivionsage	Reporter