

pnggroup / libpng Public

[Code](#) [Issues](#) 149 [Pull requests](#) 28 [Discussions](#) [Actions](#) [Projects](#)

Commit 2301926



Oblivionsage authored and ctruta committed 2 weeks ago

fix: Resolve use-after-free on `png_ptr->trans_alpha`

The function `png_set_tRNS` sets `png_ptr->trans_alpha` to point at `info_ptr->trans_alpha` directly, so both structs share the same heap buffer. If the application calls `png_free_data(PNG_FREE_TRNS)`, or if `png_set_tRNS` is called a second time, the buffer is freed through `info_ptr` while `png_ptr` still holds a dangling reference. Any subsequent row read that hits the function `png_do_expand_palette` will dereference freed memory.

The fix gives `png_struct` its own allocation instead of aliasing the `info_ptr` pointer. This was already flagged with a TODO in `png_handle_tRNS` ("horrible side effect ... Fix this.") but it was never addressed.

Verified with AddressSanitizer. All 34 existing tests pass without regressions.

Reviewed-by: Cosmin Truta <ctruta@gmail.com>
Signed-off-by: Cosmin Truta <ctruta@gmail.com>

[libpng18](#) · [v1.6.56](#)1 parent [0c37b8f](#) commit 2301926

4 files changed +30 -22 lines changed

[↑ Top](#) 

- pngread.c
- pngutil.c
- pngset.c
- pngwrite.c

4 files changed +30 -22 lines changed

Search within code



▼ pngread.c

```

@@ -788,12 +788,11 @@ png_read_destroy(png_structrp png_ptr)
788 788
789 789     #if defined(PNG_TRNS_SUPPORTED) || \
790 790         defined(PNG_READ_EXPAND_SUPPORTED) ||
           defined(PNG_READ_BACKGROUND_SUPPORTED)
791 -     if ((png_ptr->free_me & PNG_FREE_TRNS) != 0)
792 -     {
793 -         png_free(png_ptr, png_ptr->trans_alpha);
794 -         png_ptr->trans_alpha = NULL;
795 -     }
796 -     png_ptr->free_me &= ~PNG_FREE_TRNS;
791 +     /* png_ptr->trans_alpha is always independently allocated (not aliased
792 +      * with info_ptr->trans_alpha), so free it unconditionally.
793 +      */
794 +     png_free(png_ptr, png_ptr->trans_alpha);
795 +     png_ptr->trans_alpha = NULL;
797 796     #endif
798 797
799 798     inflateEnd(&png_ptr->zstream);

```

▼ pngutil.c

```

@@ -1772,10 +1772,6 @@ png_handle_tRNS(png_structrp png_ptr, png_inforp
info_ptr, png_uint_32 length)
1772 1772         return handled_error;
1773 1773     }
1774 1774
1775 -     /* TODO: this is a horrible side effect in the palette case because the
1776 -      * png_struct ends up with a pointer to the tRNS buffer owned by the
1777 -      * png_info. Fix this.
1778 -      */
1779 1775     png_set_tRNS(png_ptr, info_ptr, readbuf, png_ptr->num_trans,
1780 1776         &(png_ptr->trans_color));
1781 1777     return handled_ok;

```

```

  ▾ pngset.c
  ↑
  @@ -1155,28 +1155,35 @@ png_set_tRNS(png_structrp png_ptr, png_inforp
  info_ptr,
1155 1155
1156 1156     if (trans_alpha != NULL)
1157 1157     {
1158 -     /* It may not actually be necessary to set png_ptr->trans_alpha here;
1159 -     * we do it for backward compatibility with the way the
  png_handle_tRNS
1160 -     * function used to do the allocation.
1161 -     *
1162 -     * 1.6.0: The above statement is incorrect; png_handle_tRNS
  effectively
1163 -     * relies on png_set_tRNS storing the information in png_struct
1164 -     * (otherwise it won't be there for the code in pngtran.c).
1165 -     */
1166 -
1167 1158     png_free_data(png_ptr, info_ptr, PNG_FREE_TRNS, 0);
1168 1159
1169 1160     if (num_trans > 0 && num_trans <= PNG_MAX_PALETTE_LENGTH)
1170 1161     {
1171 -     /* Changed from num_trans to PNG_MAX_PALETTE_LENGTH in version 1.2.1
  */
1162 +     /* Allocate info_ptr's copy of the transparency data. */
1172 1163     info_ptr->trans_alpha = png_voidcast(png_bytep,
1173 1164         png_malloc(png_ptr, PNG_MAX_PALETTE_LENGTH));
1174 1165     memcpy(info_ptr->trans_alpha, trans_alpha, (size_t)num_trans);
1175 -
1176 1166     info_ptr->free_me |= PNG_FREE_TRNS;
1177 1167     info_ptr->valid |= PNG_INFO_TRNS;
1168 +
1169 +     /* Allocate an independent copy for png_struct, so that the
1170 +     * lifetime of png_ptr->trans_alpha is decoupled from the
1171 +     * lifetime of info_ptr->trans_alpha. Previously these two
1172 +     * pointers were aliased, which caused a use-after-free if
1173 +     * png_free_data freed info_ptr->trans_alpha while
1174 +     * png_ptr->trans_alpha was still in use by the row transform
1175 +     * functions (e.g. png_do_expand_palette).
1176 +     */

```

```

1177 +         png_free(png_ptr, png_ptr->trans_alpha);
1178 +         png_ptr->trans_alpha = png_voidcast(png_bytep,
1179 +             png_malloc(png_ptr, PNG_MAX_PALETTE_LENGTH));
1180 +         memcpy(png_ptr->trans_alpha, trans_alpha, (size_t)num_trans);
1181 +     }
1182 +     else
1183 +     {
1184 +         png_free(png_ptr, png_ptr->trans_alpha);
1185 +         png_ptr->trans_alpha = NULL;
1178 1186     }
1179 -     png_ptr->trans_alpha = info_ptr->trans_alpha;
1180 1187 }
1181 1188
1182 1189     if (trans_color != NULL)

```

▼ pngwrite.c

```

@@ -1010,6 +1010,12 @@ png_write_destroy(png_structrp png_ptr)
1010 1010     png_ptr->chunk_list = NULL;
1011 1011     #endif
1012 1012
1013 + #if defined(PNG_tRNS_SUPPORTED)
1014 +     /* Free the independent copy of trans_alpha owned by png_struct. */
1015 +     png_free(png_ptr, png_ptr->trans_alpha);
1016 +     png_ptr->trans_alpha = NULL;
1017 + #endif
1018 +
1013 1019     /* The error handling and memory handling information is left intact at
1014 1020     this
1015 1021     * point: the jmp_buf may still have to be freed. See
1016 1022     png_destroy_png_struct
1017 1023     * for how this happens.

```

Comments 0



Please [sign in](#) to comment.