Heap-buffer-overflow in image_set_mask(image_t*, int, int, unsigned char) #461

Closed Jorgecmartins opened this issue on Dec 30, 2021 · 2 comments

Assignees	
Labels	bug priority-medium
Milestone	Stable

Jorgecmartins commented on Dec 30, 2021 Contributor

Heap-buffer-overflow in image_set_mask(image_t*, int, int, unsigned char)

I've attached poc.zip that contains a malicious gif and a html file.

To reproduce and get the segmentation fault:

```
$ unzip poc.zip
$ htmldoc --webpage -f output.pdf crash.html
```

AddressSanitizer report:

```
=================================================================
==8922==ERROR: AddressSanitizer: heap-buffer-overflow on address 0x7f951ffffb90 at pc
0x0000006588e3 bp 0x7ffdeac61760 sp 0x7ffdeac61758
READ of size 1 at 0x7f951ffffb90 thread T0
    #0 0x6588e2 in image_set_mask(image_t*, int, int, unsigned char)
./htmldoc/htmldoc/image.cxx:1810:13
    #1 0x65607f in gif_read_image(_IO_FILE*, image_t*, unsigned char (*) [3], int, int)
./htmldoc/htmldoc/image.cxx:325:7
    #2 0x64e5c7 in image_load_gif(image_t*, _IO_FILE*, int, int)
./htmldoc/htmldoc/image.cxx:1360:12
    #3 0x64bfe8 in image_load ./htmldoc/htmldoc/image.cxx:830:14
    #4 0x575153 in write_image(_IO_FILE*, render_str*, int) ./htmldoc/htmldoc/ps-pdf.cxx:10305:5
    #5 0x554517 in pdf_write_document(unsigned char*, unsigned char*, unsigned char*, unsigned
char*, unsigned char*, unsigned char*, tree_str*, tree_str*) ./htmldoc/htmldoc/ps-pdf.cxx:2292:7
    #6 0x53f367 in pspdf_export ./htmldoc/htmldoc/ps-pdf.cxx:910:7
    #7 0x510cb6 in main ./htmldoc/htmldoc/htmldoc.cxx:1291:3
    #8 0x7f95234c10b2 in __libc_start_main /build/glibc-eX1tMB/glibc-2.31/csu/../csu/libc-
start.c:308:16
    #9 0x42110d in _start (./htmldoc/htmldoc_asan+0x42110d)

0x7f951ffffb90 is located 3631 bytes to the right of 181601-byte region
[0x7f951ffd2800,0x7f951fffed61)
allocated by thread T0 here:
    #0 0x4999c2 in calloc (./htmldoc/htmldoc_asan+0x4999c2)
    #1 0x655be0 in image_need_mask(image_t*, int) ./htmldoc/htmldoc/image.cxx:1756:24
    #2 0x64dd93 in image_load_gif(image_t*, _IO_FILE*, int, int)
./htmldoc/htmldoc/image.cxx:1343:13
    #3 0x64bfe8 in image_load ./htmldoc/htmldoc/image.cxx:830:14
    #4 0x611ff3 in compute_size(tree_str*) ./htmldoc/htmldoc/htmllib.cxx:3239:11
    #5 0x605528 in htmlReadFile ./htmldoc/htmldoc/htmllib.cxx:981:11
    #6 0x515181 in read_file(char const*, tree_str**, char const*)
./htmldoc/htmldoc/htmldoc.cxx:2492:9
    #7 0x510381 in main ./htmldoc/htmldoc/htmldoc.cxx:1177:7
    #8 0x7f95234c10b2 in __libc_start_main /build/glibc-eX1tMB/glibc-2.31/csu/../csu/libc-
start.c:308:16

SUMMARY: AddressSanitizer: heap-buffer-overflow ./htmldoc/htmldoc/image.cxx:1810:13 in
image_set_mask(image_t*, int, int, unsigned char)
Shadow bytes around the buggy address:
  0x0ff323ff7f20: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0ff323ff7f30: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0ff323ff7f40: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0ff323ff7f50: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0ff323ff7f60: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
=>0x0ff323ff7f70: fa fa[fa]fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0ff323ff7f80: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0ff323ff7f90: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0ff323ff7fa0: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0ff323ff7fb0: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0ff323ff7fc0: 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
  Shadow gap:
```

The overflow occurs on the `img->mask` buffer.
The `img->mask` is allocated on `image_need_mask()`, line 1343, using `img->width` and `img->height` values that were initialized at line 1267.

After the call to `image_need_mask()`, `img->width` and `img->height` are changed at lines 1346, 1347. By setting a higher `img->height` it its possible to go out of bounds on the buffer `img->mask` in:

```
1808 maskptr  = img->mask + y * img->maskwidth + x / 8;
1809 if (alpha <= dither[x & 3][y & 3])
1810    *maskptr |= masks[x & 7];
```

michaelrsweet self-assigned this on Dec 30, 2021

michaelrsweet added the investigating label on Dec 30, 2021

michaelrsweet commented on Dec 30, 2021 Owner

Reproduced.

michaelrsweet added bug priority-medium and removed investigating labels on Dec 30, 2021

michaelrsweet added this to the **Stable** milestone on Dec 30, 2021

michaelrsweet commented on Dec 30, 2021 Owner

[master `71fe878`] Fix potential heap overflow bug with GIF images (Issue #461)

michaelrsweet closed this as completed on Dec 30, 2021

michaelrsweet added a commit that referenced this issue on Dec 30, 2021

`Fix potential heap overflow bug with GIF images (Issue #461)` 71fe878

Assignees

michaelrsweet

Labels

bug priority-medium

Projects

None yet

Milestone

Stable

Development

No branches or pull requests

2 participants