# Two Integer Overflow bugs in image.cxx #471

Closed Zzero00 opened this issue on Mar 10 · 5 comments

Assignees

Labels bug priority-high

Milestone Stable

Zzero00 commented on Mar 10 • edited

Hi, there is two integer overflow bugs in the latest version of htmldoc.
They are similar to CVE-2021-20308.

os: ubuntu 20.04
version: 1.9.16(the latest)

## First

First, in image_load_jpeg function, image.cxx.
When it calls malloc, 'img->width' and 'img->height' are enough large to cause an integer overflow
So, the malloc function may return a heap block smaller than the expected size, and it will cause a buffer overflow/Address boundary error in the jpeg_read_scanlines function.

htmldoc/htmldoc/image.cxx
Lines 1390 to 1395 in cb4cdee

```
static int                      /* O - 0 = success, -1 = fail */
image_load_jpeg(image_t *img,   /* I - Image pointer */
                FILE    *fp,    /* I - File to load from */
                int     gray,   /* I - 0 = color, 1 = grayscale */
                int     load_data)/* I - 1 = load image data, 0 = just info */
{
```

htmldoc/htmldoc/image.cxx
Lines 1452 to 1466 in cb4cdee

```
    img->pixels = (uchar *)malloc((size_t)(img->width * img->height * img->depth));

    if (img->pixels == NULL)
    {
      jpeg_destroy_decompress(&cinfo);
      return (-1);
    }

    jpeg_start_decompress(&cinfo);

```

Asan report:

```
./htmldoc --webpage -f out.pdf ./test.html
PAGES: 4
Corrupt JPEG data: premature end of data segment
AddressSanitizer:DEADLYSIGNAL
=================================================================
==1326478==ERROR: AddressSanitizer: SEGV on unknown address 0x621000020000 (pc 0x7f0bd38812e1 bp
0x7ffd6a49c500 sp 0x7ffd6a49c460 T0)
==1326478==The signal is caused by a WRITE memory access.
    #0 0x7f0bd38812e1  (/lib/x86_64-linux-gnu/libjpeg.so.8+0x422e1)

AddressSanitizer can not provide additional info.
SUMMARY: AddressSanitizer: SEGV (/lib/x86_64-linux-gnu/libjpeg.so.8+0x422e1)
==1326478==ABORTING
```

And this is the poc file:
poc1.zip

## Second

There is another integer overflow bug in image_load_png function, image.cxx, similar to the first one.

**htmldoc/htmldoc/image.cxx**
Lines 1631 to 1647 in `cb4cdee`

```
      img->pixels = (uchar *)calloc(1, (size_t)(img->width * img->height * depth));

     /*
      * Allocate pointers...
      */

      rows = (png_bytep *)calloc(png_get_image_height(pp, info), sizeof(png_bytep));

      for (i = 0; i < (int)png_get_image_height(pp, info); i ++)
        rows[i] = img->pixels + i * img->width * depth;

```

It calls calloc to get heap block.
However, the width and height of the png file are both four bytes long, so 'img->width' and 'img->height' are enough large to cause an integer overflow.
The calloc function may return a heap block smaller than the expected size, and finally cause a heap overflow in the png_read_rows function when memcpy.

This is the Asan report:

```
./htmldoc --webpage -f out.pdf ./test.html
PAGES: 4
=================================================================
==1327797==ERROR: AddressSanitizer: heap-buffer-overflow on address 0x602000008631 at pc
0x000000434be3 bp 0x7ffc5424be70 sp 0x7ffc5424b630
WRITE of size 196608 at 0x602000008631 thread T0
    #0 0x434be2 in memcpy (/root/fuzz_workdir/htmldoc/install/bin/htmldoc+0x434be2)
    #1 0x7f847c9e379c  (/lib/x86_64-linux-gnu/libpng16.so.16+0x1d79c)
    #2 0x7f847c9d680b in png_read_row (/lib/x86_64-linux-gnu/libpng16.so.16+0x1080b)
    #3 0x7f847c9d81d8 in png_read_rows (/lib/x86_64-linux-gnu/libpng16.so.16+0x121d8)
    #4 0x5e8c0b in image_load_png(image_t*, _IO_FILE*, int, int)
/root/fuzz_workdir/tmp/htmldoc/htmldoc/image.cxx:1647:5
    #5 0x5e169d in image_load /root/fuzz_workdir/tmp/htmldoc/htmldoc/image.cxx:845:14
    #6 0x54314f in write_image(_IO_FILE*, render_str*, int)
/root/fuzz_workdir/tmp/htmldoc/htmldoc/ps-pdf.cxx:10305:5
    #7 0x55015e in pdf_write_page(_IO_FILE*, int) /root/fuzz_workdir/tmp/htmldoc/htmldoc/ps-
pdf.cxx:2695:13
    #8 0x5175c6 in pdf_write_outpage(_IO_FILE*, int) /root/fuzz_workdir/tmp/htmldoc/htmldoc/ps-
pdf.cxx:2607:9
    #9 0x5175c6 in pdf_write_document(unsigned char*, unsigned char*, unsigned char*, unsigned
char*, unsigned char*, unsigned char*, tree_str*, tree_str*)
/root/fuzz_workdir/tmp/htmldoc/htmldoc/ps-pdf.cxx:2321:5
    #10 0x5175c6 in pspdf_export /root/fuzz_workdir/tmp/htmldoc/htmldoc/ps-pdf.cxx:910:7
    #11 0x4e30ca in main /root/fuzz_workdir/tmp/htmldoc/htmldoc/htmldoc.cxx:1291:3
    #12 0x7f847c3d60b2 in __libc_start_main /build/glibc-sMfBJT/glibc-2.31/csu/../csu/libc-
start.c:308:16
    #13 0x41e86d in _start (/root/fuzz_workdir/htmldoc/install/bin/htmldoc+0x41e86d)

0x602000008631 is located 0 bytes to the right of 1-byte region [0x602000008630,0x602000008631)
allocated by thread T0 here:
    #0 0x499c42 in calloc (/root/fuzz_workdir/htmldoc/install/bin/htmldoc+0x499c42)
    #1 0x5e897d in image_load_png(image_t*, _IO_FILE*, int, int)
/root/fuzz_workdir/tmp/htmldoc/htmldoc/image.cxx:1631:26
    #2 0x5e169d in image_load /root/fuzz_workdir/tmp/htmldoc/htmldoc/image.cxx:845:14
    #3 0x54314f in write_image(_IO_FILE*, render_str*, int)
/root/fuzz_workdir/tmp/htmldoc/htmldoc/ps-pdf.cxx:10305:5
    #4 0x55015e in pdf_write_page(_IO_FILE*, int) /root/fuzz_workdir/tmp/htmldoc/htmldoc/ps-
pdf.cxx:2695:13

SUMMARY: AddressSanitizer: heap-buffer-overflow
(/root/fuzz_workdir/htmldoc/install/bin/htmldoc+0x434be2) in memcpy
Shadow bytes around the buggy address:
  0x0c047fff9070: fa fa 00 02 fa fa 00 02 fa fa 00 02 fa fa 00 02
  0x0c047fff9080: fa fa 00 02 fa fa 06 fa fa fa 06 fa fa fa 07 fa
  0x0c047fff9090: fa fa 06 fa fa fa 02 fa fa fa 00 07 fa fa fd fd
```

```
  0x0c047fff90a0: fa fa fd fa fa fa fd fa fa fa fd fa fa fa fd fd
  0x0c047fff90b0: fa fa fd fa fa fa fd fd fa fa fd fa fa fa fd fa
=>0x0c047fff90c0: fa fa fd fa fa fa[01]fa fa fa fa fa fa fa fa fa
  0x0c047fff90d0: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c047fff90e0: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c047fff90f0: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c047fff9100: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c047fff9110: 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:              cc
==1327797==ABORTING
```

And this is the poc file:
poc2.zip
(wrong poc)
This is the correct poc:
real_poc2.zip

**michaelrsweet** self-assigned this on Mar 10

**michaelrsweet** added **bug** priority-high labels on Mar 10

**michaelrsweet** added this to the **Stable** milestone on Mar 10

**michaelrsweet** added a commit that referenced this issue on Mar 10

`Fix a potential integer overflow bug in the JPEG and PNG loaders (Issue ···` ✕ `31f7804`

**michaelrsweet** commented on Mar 10 Owner

I wasn't able to reproduce the issue with poc2, but I added range checks to the JPEG and PNG load functions to limit images to <4GiB - a little lazy but for the intended usage I don't see a problem limiting images like this.

[master `31f7804` ] Fix a potential integer overflow bug in the JPEG and PNG loaders (Issue #471)

**michaelrsweet** closed this as completed on Mar 10

---

**Zzero00** commented on Mar 10 • edited Author

> I wasn't able to reproduce the issue with poc2, but I added range checks to the JPEG and PNG load functions to limit images to <4GiB - a little lazy but for the intended usage I don't see a problem limiting images like this.
>
> [master 31f7804] Fix a potential integer overflow bug in the JPEG and PNG loaders (Issue #471)

Oh, I'm sorry, this is the correct poc2 file:
real_poc2.zip

I forgot to modify the html file in the above poc2 file.
As follows:

```
$ unzip real_poc2.zip
Archive:  real_poc2.zip
  inflating: poc.png
  inflating: test.html
$ ls
htmldoc*  htmldoc_noasan*  poc.png  real_poc2.zip  test.html
$ ./htmldoc_noasan --webpage -f out.pdf ./test.html
PAGES: 4
fish: "./htmldoc_noasan --webpage -f o…" terminated by signal SIGSEGV (Address boundary error)
$ ./htmldoc --webpage -f out.pdf ./test.html
PAGES: 4
=================================================================
==1369912==ERROR: AddressSanitizer: heap-buffer-overflow on address 0x602000008631 at pc
0x000000434be3 bp 0x7ffe31cc2070 sp 0x7ffe31cc1830
WRITE of size 196608 at 0x602000008631 thread T0
......
```

**michaelrsweet** commented on Mar 14 Owner

Re-confirmed that the changes I pushed also address this test file.

**BrianInglis** commented on May 21 • edited

Currently tracked as `htmldoc` **1.9.16** vulnerabilty CVE-2022-27114 score 5.5 MEDIUM.
Please consider making an updated release soon or ask to have the affected version corrected.

**michaelrsweet** commented on May 22 Owner

**@BrianInglis** a release was done a few days ago.

Please refamiliarize yourself with the license terms of this free software that comes with no warranties or guarantees of any kind!

Repository owner locked as **resolved** and limited conversation to collaborators on May 22

Assignees

michaelrsweet

Labels

bug priority-high

Projects

None yet

Milestone

Stable

Development

No branches or pull requests

3 participants