AddressSanitizer: heap-buffer-overflow in function pdf_write_names #480

Closed hdthky opened this issue on Mar 24 · 7 comments

Assignees

Labels unable-to-reproduce

hdthky commented on Mar 24

Description

Whilst experimenting with `htmldoc` , built from commit 31f7804, we are able to induce a vulnerability in function `pdf_write_names` , using a harness compiled from `htmldoc/htmldoc.cxx` .

Because there is no bounds checking, a heap-based out-of-bound read will be triggered when the software encounters a malformed file, result in information disclosure or denial of service.

Proof of Concept

The POC is: poc_heap_overflow1

The command is: `./htmldoc --webpage -t pdf -f /dev/null poc_heap_overflow1

The ASAN report is:

```
=================================================================
==50540==ERROR: AddressSanitizer: heap-buffer-overflow on address 0x6250000020f4 at pc
0x0000003cf241 bp 0x7fffffffaf90 sp 0x7fffffffaf88
READ of size 4 at 0x6250000020f4 thread T0
    #0 0x3cf240 in pdf_write_names(_IO_FILE*) /work/libraries/htmldoc/htmldoc/ps-pdf.cxx:3589:39
    #1 0x3cf240 in pdf_write_document(unsigned char*, unsigned char*, unsigned char*, unsigned
char*, unsigned char*, unsigned char*, tree_str*, tree_str*) /work/libraries/htmldoc/htmldoc/ps-
pdf.cxx:2301:5
    #2 0x3cf240 in pspdf_export /work/libraries/htmldoc/htmldoc/ps-pdf.cxx:910:7
    #3 0x39a254 in main /work/libraries/htmldoc/htmldoc/htmldoc.cxx:1291:3
    #4 0x7ffff75070b2 in __libc_start_main (/lib/x86_64-linux-gnu/libc.so.6+0x270b2)
    #5 0x2a051d in _start (/work/libraries/htmldoc/htmldoc/htmldoc+0x2a051d)

0x6250000020f4 is located 20 bytes to the right of 8160-byte region
[0x625000000100,0x6250000020e0)
allocated by thread T0 here:
    #0 0x31b6f9 in realloc (/work/libraries/htmldoc/htmldoc/htmldoc+0x31b6f9)
    #1 0x3ddaf8 in check_pages(int) /work/libraries/htmldoc/htmldoc/ps-pdf.cxx:8859:24

SUMMARY: AddressSanitizer: heap-buffer-overflow /work/libraries/htmldoc/htmldoc/ps-pdf.cxx:3589:39
in pdf_write_names(_IO_FILE*)
Shadow bytes around the buggy address:
  0x0c4a7fff83c0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x0c4a7fff83d0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x0c4a7fff83e0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x0c4a7fff83f0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x0c4a7fff8400: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
=>0x0c4a7fff8410: 00 00 00 00 00 00 00 00 00 00 00 00 fa fa[fa]fa
  0x0c4a7fff8420: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c4a7fff8430: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c4a7fff8440: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c4a7fff8450: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c4a7fff8460: 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
==50540==ABORTING
```

Impact

This vulnerability is capable of inducing information disclosure or denial of service.

michaelrsweet commented on Mar 24 Owner

OK, so I am unable to reproduce when "leak_check_at_exit=false" is set. When not set I get completely different results.

michaelrsweet self-assigned this on Mar 24

michaelrsweet added the unable-to-reproduce label on Mar 24

hdthky commented on Mar 24 Author

It is a little weird. Even if I turn off leak_check_at_exit, I can still reproduce it.
The compile command I used is:
`CFLAGS="-g -fsanitize=address" CXXFLAGS="-g -fsanitize=address" LDFLAGS="-fsanitize=address" ./configure`

hdthky commented on Mar 24 Author

The causes of buffer overflow and memory leak are totally different. In theory, turning leak_check_at_exit off or not doesn't affect the result.

michaelrsweet commented on Mar 24 Owner

@hdthky I am well aware of the differences. Theory doesn't matter, actual run evidence shows a difference on three difference systems at my immediate disposal (iMac Pro running current macOS, Ubuntu VM on that system, and Ubuntu VM on an M1 MacBook Pro). No errors on macOS, different results on both Ubuntu VMs.

michaelrsweet added a commit that referenced this issue on Mar 24

`Call check_pages when writing links (Issue #480)` ✗ `46c8ec2`

michaelrsweet commented on Mar 24 Owner

@hdthky Assuming the cause is similar to the others, the following change should fix things for you:

[master `46c8ec2`] Call check_pages when writing links (Issue #480)

hdthky commented on Mar 24 Author

Yes, it has been fixed now.

michaelrsweet closed this as completed on Mar 24

hdthky commented on May 19 Author

The vulnerability was found by Xingyuan Mo, Hui Lu, Zhihong Tian from Guangzhou University.

Assignees

michaelrsweet

Labels

unable-to-reproduce

Projects

None yet

Milestone

No milestone

Development

No branches or pull requests

2 participants