New issue

Jump to bottom

Bug: Buffer Overflow into Out-of-Bounds Write #8

Closed carter-yagemann opened this issue on Feb 28, 2020 · 7 comments

Assignees

carter-yagemann commented on Feb 28, 2020

Description

In v0.12 and newer, the function `get_type()` in `pdf.c` has the following logic:

pdfresurrect/pdf.c
Lines 1299 to 1304 in `e4de322`

```
1299    /* Return the value by storing it in static mem */
1300    memcpy(buf, c, (((c - obj) < sizeof(buf)) ? c - obj : sizeof(buf)));
1301    c = buf;
1302    while (!(isspace(*c) || *c=='/' || *c=='>'))
1303      ++c;
1304    *c = '\0';
```

If `buf` does not contain one of the expected terminating characters (whitespace, `/`, `>`), `c` can point to an address outside `buf`, causing a `\x00` byte to be written out-of-bounds.

Example

Instead of creating a PoC, I found a benign PDF that happens to trigger this bug: http://ftpcontent.worldnow.com/wbbh/documents/Remoteattacksurfaces.pdf
(sha256: `371d87d27666d1f97678cbf4eec03704f4c1e85029009ee2439690303f7dde28`)

The problem occurs while parsing the following data:

```
obj\r\n<</Type/FontDescriptor/FontName/ABCDEE+Calibri/Flags 32/ItalicAngle 0/Ascent 750/Descent -250/CapHeight 750/AvgWidth 521/MaxWidth 1743/FontWeight 400/XHeight 250/StemV
52/FontBBox[ -503 -250 1240 750] /FontFile2 5812 0 R>>\r\nendobj
```

Due to the reuse of `buf` between invocations of the function, `buf` will eventually contain:

```
"FontDescriptor\000FontName\000DeviceRG"
```

This benign example causes a *read* to segfault, but a more carefully crafted input could cause an out-of-bounds write.

Valgrind

```
<removed for brevity>
Remoteattacksurfaces.pdf: --A-- Version 1 -- Object 2029 (FontDescriptor)
Remoteattacksurfaces.pdf: --A-- Version 1 -- Object 2030 (FontDescriptor)
Remoteattacksurfaces.pdf: --A-- Version 1 -- Object 2031 (FontDescriptor)
Remoteattacksurfaces.pdf: --A-- Version 1 -- Object 2032 (FontDescriptor)
Remoteattacksurfaces.pdf: --A-- Version 1 -- Object 2033 (FontDescriptor)
Remoteattacksurfaces.pdf: --A-- Version 1 -- Object 2034 (FontDescriptor)
Remoteattacksurfaces.pdf: --A-- Version 1 -- Object 2035 (FontDescriptor)
Remoteattacksurfaces.pdf: --A-- Version 1 -- Object 2036 (FontDescriptor)
Remoteattacksurfaces.pdf: --A-- Version 1 -- Object 2037 (FontDescriptor)
Remoteattacksurfaces.pdf: --A-- Version 1 -- Object 2038 (FontDescriptor)
Remoteattacksurfaces.pdf: --A-- Version 1 -- Object 2039 (FontDescriptor)
==18759== Invalid read of size 1
==18759==    at 0x10D12D: get_type (pdf.c:1296)
==18759==    by 0x10B012: pdf_summarize (pdf.c:503)
==18759==    by 0x109F95: main (main.c:337)
==18759==  Address 0x112000 is not stack'd, malloc'd or (recently) free'd
==18759==
==18759==
==18759== Process terminating with default action of signal 11 (SIGSEGV)
==18759==  Access not within mapped region at address 0x112000
==18759==    at 0x10D12D: get_type (pdf.c:1296)
==18759==    by 0x10B012: pdf_summarize (pdf.c:503)
==18759==    by 0x109F95: main (main.c:337)
==18759==  If you believe this happened as a result of a stack
==18759==  overflow in your program's main thread (unlikely but
==18759==  possible), you can try to increase the size of the
==18759==  main thread stack using the --main-stacksize= flag.
==18759==  The main thread stack size used in this run was 8388608.
==18759==
==18759== HEAP SUMMARY:
==18759==     in use at exit: 284,202 bytes in 9 blocks
==18759==   total heap usage: 35,756 allocs, 35,747 frees, 2,340,979,569 bytes allocated
==18759==
==18759== LEAK SUMMARY:
==18759==    definitely lost: 0 bytes in 0 blocks
==18759==    indirectly lost: 0 bytes in 0 blocks
==18759==      possibly lost: 0 bytes in 0 blocks
==18759==    still reachable: 284,202 bytes in 9 blocks
==18759==         suppressed: 0 bytes in 0 blocks
==18759== Rerun with --leak-check=full to see details of leaked memory
==18759==
==18759== For counts of detected and suppressed errors, rerun with: -v
==18759== ERROR SUMMARY: 1 errors from 1 contexts (suppressed: 0 from 0)
```

👍 1

enferex commented on Feb 28, 2020 Owner

Thanks a ton for this! I'll work on a patch.

enferex self-assigned this on Feb 28, 2020

enferex commented on Feb 29, 2020 • edited Owner

I've fixed the issue with the overflow; however, we are dropping some type names (e.g., less descriptive and returning "Unknown" more frequently than I expect. This doesn't concern me too much, as I don't really know if reporting type names is all that valuable to users. The fix is currently in its own branch: https://github.com/enferex/pdfresurrect/tree/carter-fix

Edit: I plan on merging this into master once I get a better understanding of why we are loosing more type names.

enferex commented on Feb 29, 2020 Owner

I've fixed the type name information, now we should maintain consistency with reporting names as we were in v.19, but with the added sanity check now.

carnil commented on Mar 2, 2020

This issue appears to have been assigned CVE-2020-9549.

enferex commented on Mar 2, 2020 Owner

> This issue appears to have been assigned CVE-2020-9549.

Yep, thanks for following up with that. Master has the latest fixes.

carnil commented on Mar 3, 2020

@enferex `36b67e5` and `bfa81b9` specifically?

enferex commented on Mar 5, 2020 Owner

> @enferex 36b67e5 and bfa81b9 specifically?

Yep, they should be the ones.

enferex closed this as completed on Mar 5, 2020

Assignees

enferex

Labels

None yet

Projects

None yet

Milestone

No milestone

Development

No branches or pull requests

3 participants