New issue

Jump to bottom

Integer overflow bug in _parse_special_tag function, sxmlc.c #22

Closed Zzero00 opened this issue on Apr 13 · 2 comments

Zzero00 commented on Apr 13

Hi, there is a integer overflow bug in _parse_special_tag function, sxmlc.c.

epub2txt2/src/sxmlc.c
Lines 1203 to 1219 in 02f69e6

```
1203    static TagType _parse_special_tag(const SXML_CHAR* str, int len, _TAG* tag, XMLNode*
1204    {
1205            if (sx_strncmp(str, tag->start, tag->len_start))
1206                    return TAG_NONE;
1207
1208            if (sx_strncmp(str + len - tag->len_end, tag->end, tag->len_end)) /* There prc
1209                    return TAG_PARTIAL;
1210
1211            node->tag = __malloc((len - tag->len_start - tag->len_end + 1)*sizeof(SXML_CHA
1212            if (node->tag == NULL)
```

It passes ((len - tag->len_start - tag->len_end + 1)*sizeof(SXML_CHAR)) as a parameter to malloc function.
If (len - tag->len_start - tag->len_end) == -1, then (len - tag->len_start - tag->len_end + 1) == 0.
It is legal to use 0 as an argument to the malloc function, and it will return the address of a small heap successfully.

However, in line 1214, it passes (len - tag->len_start - tag->len_end) as a parameter to strncpy function.
-1 will be coerced to an unsigned integer: 0xffffffffffffffff. It is a huge size and will make the program crashed.

poc:
poc.zip

To reproduce:

```
$ wget https://github.com/kevinboone/epub2txt2/files/8482640/poc.zip
......
$ unzip poc.zip
Archive:  poc.zip
  inflating: poc
$ ls
epub2txt  poc  poc.zip
$ ./epub2txt --version
epub2txt version 2.04
Copyright (c)2013-2022 Kevin Boone and contributors
Distributed under the terms of the GNU Public Licence, v3.0
$ ./epub2txt poc
/tmp/epub2txt5552/OPS/epb.opf  bad CRC cb87c959  (should be 0192f2f4)
Segmentation fault (core dumped)
```

The epub2txt is built with:

```
git clone https://github.com/kevinboone/epub2txt2 && cd epub2txt2
make && sudo make install
```

Tested on: Ubuntu 20.04

kevinboone commented on Apr 14 — Owner

I'm not sure what to do about this. The failure occurs because epub2txt is fed with gibberish XML. So it crashes. It crashes in a clean and safe way because there's an immediate segfault. Sure, it would be nicer for the user to get a more helpful error message, but I suspect there's a gazillion ways you could formulate a broken EPUB that would crash the utility. The EPUB supplied here crashes the Calibre EPUB viewer as well, although you get a Python stack back-trace rather than segfault. But it still fails.

I use a lightweight XML parser to keep the size of epub2txt to a minimum, so it can run on embedded systems. Probably I'd get a nicer mode of failure if I used a full-fat XML parser. But -- and this is the main point I'm trying to make -- *it will still fail*.

So I'm unsure what merit there is to fixing this bug. I'm not denying it's a bug, but fixing it won't really make epub2txt a better utility. It still won't be able to process a corrupt EPUB. If epub2txt crashes with any well-formed EPUB, that can be displayed using other viewers, that's something I'd want to fix. This kind of thing, though, I'm not sure.

Comments welcome.

Zzero00 commented on Apr 14 — Author

> I'm not sure what to do about this. The failure occurs because epub2txt is fed with gibberish XML. So it crashes. It crashes in a clean and safe way because there's an immediate segfault. Sure, it would be nicer for the user to get a more helpful error message, but I suspect there's a gazillion ways you could formulate a broken EPUB that would crash the utility. The EPUB supplied here crashes the Calibre EPUB viewer as well, although you get a Python stack back-trace rather than segfault. But it still fails.
>
> I use a lightweight XML parser to keep the size of epub2txt to a minimum, so it can run on embedded systems. Probably I'd get a nicer mode of failure if I used a full-fat XML parser. But -- and this is the main point I'm trying to make -- *it will still fail*.
>
> So I'm unsure what merit there is to fixing this bug. I'm not denying it's a bug, but fixing it won't really make epub2txt a better utility. It still won't be able to process a corrupt EPUB. If epub2txt crashes with any well-formed EPUB, that can be displayed using other viewers, that's something I'd want to fix. This kind of thing, though, I'm not sure.
>
> Comments welcome.

Yes, I agree with you that if epub2txt2 is used in embedded systems, it is not very important to fix this bug.

👍 1

 **Zzero00** closed this as completed on Apr 14

Assignees

No one assigned

Labels

None yet

Projects

None yet

Milestone

No milestone

Development

No branches or pull requests

2 participants