ezXML Bugs

Status: Beta
Brought to you by: voisine

#21 Out-of-bounds write caused by incorrect error handling of malloc in ezxml_new (ezxml.c:838)

Milestone: v1.0_(example)
Status: open
Owner: Aaron Voisine
Labels: None
Priority: 5
Updated: 2021-10-25
Created: 2021-01-24
Creator: CVE Reporting
Private: No

ezxml is vulnerable to OOB write when opening XML file after exhausting the memory pool.

Incorrect handling of the value returned by calloc in mg_tls_init may lead to:
- out-of-bound write attempt and segmentation fault error in case of restrictive memory protection,
- near NULL pointer overwrite in case of limited memory restrictions (e.g. in embedded environments).

Memory allocations are triggered during opening XML files, so the allocation error can be caused locally or remotely depending on the way of obtaining files.
In some embedded environments near zero memory areas are used to store device configuration, so in this case such configuration can be overwritten using this vulnerability.

Vulnerable code (ezxml.c):

```
833: // returns a new empty ezxml structure with the given root tag name
834: ezxml_t ezxml_new(const char *name)
835: {
836:     static char *ent[] = { "lt;", "&#60;", "gt;", "&#62;", "quot;", "&#34;",
837:                            "apos;", "&#39;", "amp;", "&#38;", NULL };
838:     ezxml_root_t root = (ezxml_root_t)memset(malloc(sizeof(struct ezxml_root)),
839:                                              '\0', sizeof(struct ezxml_root));
```

See following recommendations for details (especially the calloc example):
https://wiki.sei.cmu.edu/confluence/display/c/ERR33-C.+Detect+and+handle+standard+library+errors

The issue can be reproduced and tested using ErrorSanitizer (https://gitlab.com/ErrorSanitizer/ErrorSanitizer).

Reproduction steps:

1. Install gdb
2. Download and unpack code of ErrorSanitizer (https://gitlab.com/ErrorSanitizer/ErrorSanitizer)
3. Perform compilation of ErrorSanitizer according to the manual (https://gitlab.com/ErrorSanitizer/ErrorSanitizer#compilation)
   cd ErrorSanitizer; make
4. Set ESAN to the path of ErrorSanitizer directory
   export ESAN=/opt/...
5. Download and unzip attached map temp_0.cur_input
6. Download and compile ezxml 0.8.6
7. Run ezxml test program example with ErrorSanitizer in gdb using:
   gdb -batch -ex='run' -ex='backtrace' -ex='backtrace full' --args env LD_PRELOAD=$ESAN/error_sanitizer_preload.so ./ezxmltest temp_0.cur_input

You should receive similar output:

process 10435 is executing new program: ezxml/ezxmltest

```
Program received signal SIGSEGV, Segmentation fault.
__memset_avx2_unaligned_erms () at ../sysdeps/x86_64/multiarch/memset-vec-unaligned-erms.S:
#0  __memset_avx2_unaligned_erms () at ../sysdeps/x86_64/multiarch/memset-vec-unaligned-erm
#1  0x0000555555559991 in ezxml_new (name=0x0) at ezxml.c:838
#2  0x000055555555756d in ezxml_parse_str (s=0x7ffff7ff5000 "<TAG1>VALUE</TAG1>\n", len=19)
#3  0x00005555555584c4 in ezxml_parse_fd (fd=3) at ezxml.c:641
#4  0x00005555555585c4 in ezxml_parse_file (file=0x7fffffffe222 "temp_0.esn_input") at ezxm
#5  0x000055555555a53a in main (argc=2, argv=0x7fffffffde78) at ezxml.c:1008
#0  __memset_avx2_unaligned_erms () at ../sysdeps/x86_64/multiarch/memset-vec-unaligned-erm
No locals.
#1  0x0000555555559991 in ezxml_new (name=0x0) at ezxml.c:838
    ent = {0x55555555a97e "lt;", 0x55555555a982 "&#60;", 0x55555555a988 "gt;", 0x55555555a9
    root = 0x7ffff78a1dc8 <__GI___sysconf+872>
#2  0x000055555555756d in ezxml_parse_str (s=0x7ffff7ff5000 "<TAG1>VALUE</TAG1>\n", len=19)
    root = 0x555555554f80 <_start>
    q = 0 '\000'
    e = 0 '\000'
    d = 0x5400000054 <error: Cannot access memory at address 0x5400000054>
    attr = 0x1012
    a = 0x7fffffffdd20
    l = 64
    i = 0
    j = 84
#3  0x00005555555584c4 in ezxml_parse_fd (fd=3) at ezxml.c:641
    root = 0x0
    st = {st_dev = 66311, st_ino = 2527222, st_nlink = 1, st_mode = 33188, st_uid = 1000, s
    l = 4096
    m = 0x7ffff7ff5000
#4  0x00005555555585c4 in ezxml_parse_file (file=0x7fffffffe222 "temp_0.esn_input") at ezxm
    fd = 3
    xml = 0x7ffff7db3c79 <line+25>
#5  0x000055555555a53a in main (argc=2, argv=0x7fffffffde78) at ezxml.c:1008
    xml = 0x7fffffffde70
    s = 0x0
    i = 21845
```

1 Attachments

temp_0.cur_input

Discussion

Egbert Eich - *2021-10-25*

This bug, just like bug 22 and bug 23 takes advantage of the fact, that ezxml assumes that (re)allocation of memory will always succeed. There are many more cases where this can be exploited.
It seems that this project was chosen as a test case for a tool written for the purpose of testing whether software handles (re)allocation failures properly.
The issue demonstrated by the attached test case can be mitigated by the attached patch. This begs the question how useful this patch is as it only addresses one of many places a similar issue may occur.

Fix-CVE-2021-26221-bug-21.patch

Log in to post a comment.