ezXML Bugs

Status: Beta
Brought to you by: voisine

#27 Out-of-bounds read/write in ezxml_parse_str() in ezxml.c:586/587

Milestone: v1.0 (example) **Status:** open **Owner:** Aaron Voisine **Labels:** None
Priority: 5
Updated: 2021-04-16 **Created:** 2021-04-16 **Creator:** rc0r **Private:** No

Description

Function `ezxml_parse_str()` performs incorrect memory handling while parsing crafted XML files which may lead to an out-of-bounds read and write in `ezxml.c:586` and `ezxml.c:587`.

The OOB write leads to a crash because it attempts to write past the `mmap()`'ed memory region used for reading the crafted XML file in case `EZXML_NOMMAP` is not set. Memory mapping occurs in `ezxml_parse_fd()`:

```
#ifndef EZXML_NOMMAP
    l = (st.st_size + sysconf(_SC_PAGESIZE) - 1) & ~(sysconf(_SC_PAGESIZE) -1);
    if ((m = mmap(NULL, l, PROT_READ | PROT_WRITE, MAP_PRIVATE, fd, 0)) !=
        MAP_FAILED) {
        madvise(m, l, MADV_SEQUENTIAL); // optimize for sequential access
        root = (ezxml_root_t)ezxml_parse_str(m, st.st_size);
        madvise(m, root->len = l, MADV_NORMAL); // put it back to normal
    }
    else { // mmap failed, read file into memory
#endif // EZXML_NOMMAP
```

The invalid read occurs in case the call to `strcspn(s + 1, "[]>") + 1` in the code below does not find any of the `[]>` characters in `s`. Then the length of the string `s` is returned. Thus the addition `s += strcspn(s + 1, "[]>") + 1;` points to the last byte in `s`. Later on `s` is incremented pointing past the allocated buffer.

```
else if (! strncmp(s, "!DOCTYPE", 8)) { // dtd
    for (l = 0; *s && ((! l && *s != '>') || (l && (*s != ']' ||
         *(s + strspn(s + 1, EZXML_WS) + 1) != '>')));
         l = (*s == '[') ? 1 : l) s += strcspn(s + 1, "[]>") + 1;
    if (! *s && e != '>')
        return ezxml_err(root, d, "unclosed <!DOCTYPE");
    d = (l) ? strchr(d, '[') + 1 : d;
    if (l && ! ezxml_internal_dtd(root, d, s++ - d)) return &root->xml;
```

Depending on the contents of the memory following `s` the check in `ezxml.c:586` may or may not fail. During the check dereferencing `s` performs an out-of-bounds read.
In case the checks pass the out-of-bounds write of constant `\0` occurs in `ezxml.c:587`:

```
if (! s || ! *s) break;
*s = '\0';
```

MITRE assigned **CVE-2021-31347** for the out-of-bounds write issue and **CVE-2021-31348** for the out-of-bounds read issue.

Debugging Output

```
$ gdb ~/tmp/ezxml/ezxml_test
$ r CVE-2021-31347-OOBRW-000.sample

Program received signal SIGSEGV, Segmentation fault.
ezxml_parse_str (s=<optimized out>, s@entry=0x7f55fb00c000 "<?xNl", len=<optimized out>) at
587             *s = '\0';

Assembly
 0x0000562214c04398  48 85 c0           ezxml_parse_str+1160 test   %rax,%rax
 0x0000562214c0439b  0f 84 22 fd ff ff  ezxml_parse_str+1163 je     0x562214c040c3 <ezxml_p
 0x0000562214c043a1  0f b6 10           ezxml_parse_str+1169 movzbl (%rax),%edx
 0x0000562214c043a4  84 d2              ezxml_parse_str+1172 test   %dl,%dl
 0x0000562214c043a6  0f 84 17 fd ff ff  ezxml_parse_str+1174 je     0x562214c040c3 <ezxml_p
>0x0000562214c043ac  c6 00 00          ezxml_parse_str+1180 movb   $0x0,(%rax)
 0x0000562214c043af  48 8b 44 24 48     ezxml_parse_str+1183 mov    0x48(%rsp),%rax
 0x0000562214c043b4  48 8d 68 01        ezxml_parse_str+1188 lea    0x1(%rax),%rbp
 0x0000562214c043b8  48 89 6c 24 48     ezxml_parse_str+1192 mov    %rbp,0x48(%rsp)
 0x0000562214c043bd  0f b6 50 01        ezxml_parse_str+1197 movzbl 0x1(%rax),%edx

>>> i r
rax            0x7f55fb00e000      140007260086272
rbx            0x1                 1
rcx            0x562215b38         23121189688
rdx            0x7f                127
rsi            0x562215b36010      94704392953872
rdi            0x7                 7
rbp            0x562214c0614b      0x562214c0614b
rsp            0x7ffe733d2b50      0x7ffe733d2b50
r8             0x562215b38480      94704392963200
r9             0x7f55fb00dfba      140007260086202
r10            0x562215b384b0      94704392963248
r11            0x4                 4
r12            0x562214c0603a      94704377028666
r13            0x7f55fb00dfff      140007260086271
r14            0x7f55fb00e000      140007260086272
r15            0x7f55fadcfb20      140007257733920
rip            0x562214c043ac      0x562214c043ac <ezxml_parse_str+1180>
eflags         0x10202             [ IF RF ]
cs             0x33                51
ss             0x2b                43
ds             0x0                 0
es             0x0                 0
fs             0x0                 0
gs             0x0                 0
fs_base        0x7f55fadcfb80      0x7f55fadcfb80
gs_base        0x0                 0x0

>>> bt
#0  ezxml_parse_str (s=<optimized out>, s@entry=0x7f55fb00c000 "<?xNl", len=<optimized out>)
#1  0x0000562214c04b59 in ezxml_parse_fd (fd=fd@entry=3) at ezxml.c:641
#2  0x0000562214c04bfb in ezxml_parse_file (file=<optimized out>) at ezxml.c:659
#3  0x0000562214c0126a in main (argc=<optimized out>, argv=<optimized out>) at ezxml.c:1008

>>> up
#1  0x0000562214c04b59 in ezxml_parse_fd (fd=fd@entry=3) at ezxml.c:641
641             root = (ezxml_root_t)ezxml_parse_str(m, st.st_size);

>>> p m
$1 = (void *) 0x7f55fb00c000

>>> p m + st.st_size
$2 = (void *) 0x7f55fb00e000

>>> b ezxml.c:572

>>> r
>>> c // until the breakpoints hits the last time before SEGFAULT
0x0000562214c04866      572                     l = (*s == '[') ? 1 : l) s += strcspn(s + 1

Assembly

 0x0000562214c04854  3c 3e              ezxml_parse_str+2372 cmp    $0x3e,%al
 0x0000562214c04856  0f 84 37 fb ff ff  ezxml_parse_str+2374 je     0x562214c04393 <ezxml_p
 0x0000562214c0485c  4d 8d 75 01        ezxml_parse_str+2380 lea    0x1(%r13),%r14
 0x0000562214c04860  48 89 ee           ezxml_parse_str+2384 mov    %rbp,%rsi
 0x0000562214c04863  4c 89 f7           ezxml_parse_str+2387 mov    %r14,%rdi
>0x0000562214c04866  e8 a5 c8 ff ff     ezxml_parse_str+2390 call   0x562214c01110 <strcspn@
 0x0000562214c0486b  4d 8d 6c 05 01     ezxml_parse_str+2395 lea    0x1(%r13,%rax,1),%r13
 0x0000562214c04870  4c 89 6c 24 48     ezxml_parse_str+2400 mov    %r13,0x48(%rsp)
 0x0000562214c04875  41 0f b6 45 00     ezxml_parse_str+2405 movzbl 0x0(%r13),%eax
 0x0000562214c0487a  3c 5b              ezxml_parse_str+2410 cmp    $0x5b,%al

>>> p (char *) $rdi    // = s + 1
$3 = 0x7f55fb00dff7 " <driver"

>>> ni
>>> p $rax
$4 = 8

>>> p (char *) $r13   // = s
$5 = 0x7f55fb00dfff ""

>>> b ezxml.c:576
```

```
>>> c
Breakpoint 2, ezxml_parse_str (s=<optimized out>, s@entry=0x7f55fb00c000 "<?xNl", len=<optir
576                 if (l && ! ezxml_internal_dtd(root, d, s++ - d)) return &root->xml;

Assembly

 0x0000562214c048b6  48 8b 7c 24 20     ezxml_parse_str+2470 mov    0x20(%rsp),%rdi
 0x0000562214c048bb  4c 89 74 24 48     ezxml_parse_str+2475 mov    %r14,0x48(%rsp)
 0x0000562214c048c0  48 8d 70 01        ezxml_parse_str+2480 lea    0x1(%rax),%rsi
 0x0000562214c048c4  48 29 f2           ezxml_parse_str+2484 sub    %rsi,%rdx
 0x0000562214c048c7  48 89 74 24 18     ezxml_parse_str+2487 mov    %rsi,0x18(%rsp)
>0x0000562214c048cc  e8 ef dc ff ff     ezxml_parse_str+2492 call   0x562214c025c0 <ezxml_i
 0x0000562214c048d1  66 85 c0           ezxml_parse_str+2497 test   %ax,%ax
 0x0000562214c048d4  0f 85 b9 fa ff ff  ezxml_parse_str+2500 jne    0x562214c04393 <ezxml_p
 0x0000562214c048da  48 8b 44 24 20     ezxml_parse_str+2506 mov    0x20(%rsp),%rax
 0x0000562214c048df  e9 09 f7 ff ff     ezxml_parse_str+2511 jmp    0x562214c03fed <ezxml_p

>>> b ezxml.c:587
>>> c
586             if (! s || ! *s) break;

Assembly

 0x0000562214c0438b  48 89 df           ezxml_parse_str+1147 mov    %rbx,%rdi
 0x0000562214c0438e  e8 fd dd ff ff     ezxml_parse_str+1150 call   0x562214c02190 <ezxml_c
 0x0000562214c04393  48 8b 44 24 48     ezxml_parse_str+1155 mov    0x48(%rsp),%rax
 0x0000562214c04398  48 85 c0           ezxml_parse_str+1160 test   %rax,%rax
 0x0000562214c0439b  0f 84 22 fd ff ff  ezxml_parse_str+1163 je     0x562214c040c3 <ezxml_p
>0x0000562214c043a1  0f b6 10           ezxml_parse_str+1169 movzbl (%rax),%edx
 0x0000562214c043a4  84 d2              ezxml_parse_str+1172 test   %dl,%dl
 0x0000562214c043a6  0f 84 17 fd ff ff  ezxml_parse_str+1174 je     0x562214c040c3 <ezxml_p
 0x0000562214c043ac  c6 00 00           ezxml_parse_str+1180 movb   $0x0,(%rax)
 0x0000562214c043af  48 8b 44 24 48     ezxml_parse_str+1183 mov    0x48(%rsp),%rax

>>> p/x $rax
$6 = 0x7f55fb00e000

>>> p (char *) $rax
$7 = 0x7f55fb00e000 "\177ELF\002\001\001"
```

Reproduction

```
$ cd ~/tmp/ezxml
$ gcc -Wall -O2 -DEZXML_TEST -g -ggdb -o ezxml_test ezxml.c
$ ~/tmp/ezxml/ezxml_test CVE-2021-31347-OOBRW-000.sample
$ valgrind -s ~/tmp/ezxml/ezxml_test CVE-2021-31347-OOBRW-000.sample
$ gdb ~/tmp/ezxml/ezxml_test CVE-2021-31347-OOBRW-000.sample
```

Patch

```
--- ezxml.c     2006-06-08 04:33:38.000000000 +0200
+++ ezxml-fixed.c       2021-04-15 17:47:06.813825471 +0200
@@ -570,7 +570,7 @@
             for (l = 0; *s && ((! l && *s != '>') || (l && (*s != ']' ||
                  *(s + strspn(s + 1, EZXML_WS) + 1) != '>')));
                  l = (*s == '[') ? 1 : l) s += strcspn(s + 1, "[]>") + 1;
-            if (! *s && e != '>')
+            if (! *s)
                 return ezxml_err(root, d, "unclosed <!DOCTYPE");
             d = (l) ? strchr(d, '[') + 1 : d;
             if (l && ! ezxml_internal_dtd(root, d, s++ - d)) return &root->xml;
```

Files

- CVE-2021-31347-OOBRW-000.sample (Crash sample)
- CVE-2021-31347-OOBRW-000.patch (Patch fixing "unclosed <!DOCTYPE" check)

2 Attachments

CVE-2021-31347-OOBRW-000.patch

CVE-2021-31347-OOBRW-000.sample

Discussion

Log in to post a comment.

SourceForge

Create a Project

Open Source Software

Business Software

Top Downloaded Projects

Company

About
Team
SourceForge Headquarters
225 Broadway Suite 1600
San Diego, CA 92101
+1 (858) 454-5900

Resources

Support
Site Documentation
Site Status

© 2022 Slashdot Media. All Rights Reserved.