Out of Bounds Read In static void elm_close(tree_node_t *nodo) #19

Closed Halcy0nic opened this issue on Oct 24 · 2 comments

Halcy0nic commented on Oct 24

Hi there!

Great work on html2xhtml, I find myself using it quite often. While I was using the tool I created some fuzz tests to run in the background. A couple of test cases led to a segfault when using the '-t frameset' option, which led me to further investigate the crash.

Valgrind

I started with Valgrind, which reported an invalid read of size 4 in each of the test cases:

```
==1040381== Memcheck, a memory error detector
==1040381== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==1040381== Using Valgrind-3.18.1 and LibVEX; rerun with -h for copyright info
==1040381== Command: ./src/html2xhtml -t frameset
report/vuln/id:000000,sig:11,src:001386+001369,time:12081510,execs:2336913,op:splice,rep:16
==1040381==
==1040381== Invalid read of size 4
==1040381==    at 0x40E911: elm_close (procesador.c:944)
==1040381==    by 0x410617: err_html_struct (procesador.c:1889)
==1040381==    by 0x40F20A: err_content_invalid (procesador.c:0)
==1040381==    by 0x40F20A: elm_close (procesador.c:959)
==1040381==    by 0x40E7C4: saxEndDocument (procesador.c:233)
==1040381==    by 0x40DF7A: main (html2xhtml.c:117)
==1040381==  Address 0x6f20d4 is not stack'd, malloc'd or (recently) free'd
==1040381==
==1040381==
==1040381== Process terminating with default action of signal 11 (SIGSEGV)
==1040381==  Access not within mapped region at address 0x6F20D4
==1040381==    at 0x40E911: elm_close (procesador.c:944)
==1040381==    by 0x410617: err_html_struct (procesador.c:1889)
==1040381==    by 0x40F20A: err_content_invalid (procesador.c:0)
==1040381==    by 0x40F20A: elm_close (procesador.c:959)
==1040381==    by 0x40E7C4: saxEndDocument (procesador.c:233)
==1040381==    by 0x40DF7A: main (html2xhtml.c:117)
```

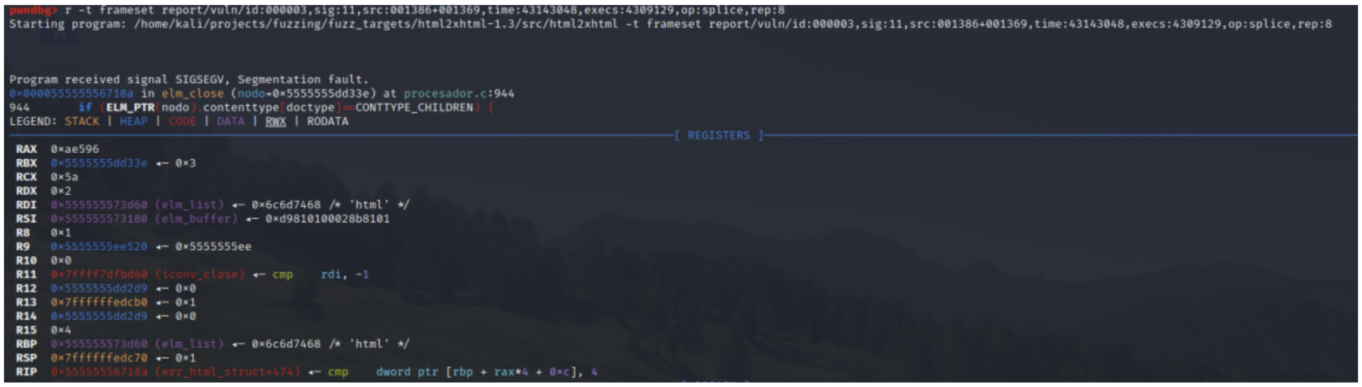
```
==1040381==  If you believe this happened as a result of a stack
==1040381==  overflow in your program's main thread (unlikely but
==1040381==  possible), you can try to increase the size of the
==1040381==  main thread stack using the --main-stacksize= flag.
==1040381==  The main thread stack size used in this run was 8388608.
==1040381==
==1040381== HEAP SUMMARY:
==1040381==     in use at exit: 88,190 bytes in 13 blocks
==1040381==   total heap usage: 22 allocs, 9 frees, 2,218,413 bytes allocated
==1040381==
==1040381== LEAK SUMMARY:
==1040381==    definitely lost: 0 bytes in 0 blocks
==1040381==    indirectly lost: 0 bytes in 0 blocks
==1040381==      possibly lost: 0 bytes in 0 blocks
==1040381==    still reachable: 88,190 bytes in 13 blocks
==1040381==         suppressed: 0 bytes in 0 blocks
==1040381== Rerun with --leak-check=full to see details of leaked memory
==1040381==
==1040381== For lists of detected and suppressed errors, rerun with: -s
==1040381== ERROR SUMMARY: 1 errors from 1 contexts (suppressed: 0 from 0)
==1040419== Memcheck, a memory error detector
==1040419== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==1040419== Using Valgrind-3.18.1 and LibVEX; rerun with -h for copyright info
==1040419== Command: ./src/html2xhtml -t frameset
report/vuln/id:000001,sig:11,src:001386+001369,time:12316330,execs:2651995,op:splice,rep:16
==1040419==
==1040419== Invalid read of size 4
==1040419==    at 0x40E911: elm_close (procesador.c:944)
==1040419==    by 0x410617: err_html_struct (procesador.c:1889)
==1040419==    by 0x40F20A: err_content_invalid (procesador.c:0)
==1040419==    by 0x40F20A: elm_close (procesador.c:959)
==1040419==    by 0x40E7C4: saxEndDocument (procesador.c:233)
==1040419==    by 0x40DF7A: main (html2xhtml.c:117)
==1040419==  Address 0x6efc84 is not stack'd, malloc'd or (recently) free'd
==1040419==
==1040419==
==1040419== Process terminating with default action of signal 11 (SIGSEGV)
==1040419==  Access not within mapped region at address 0x6EFC84
==1040419==    at 0x40E911: elm_close (procesador.c:944)
==1040419==    by 0x410617: err_html_struct (procesador.c:1889)
==1040419==    by 0x40F20A: err_content_invalid (procesador.c:0)
==1040419==    by 0x40F20A: elm_close (procesador.c:959)
==1040419==    by 0x40E7C4: saxEndDocument (procesador.c:233)
==1040419==    by 0x40DF7A: main (html2xhtml.c:117)
==1040419==  If you believe this happened as a result of a stack
==1040419==  overflow in your program's main thread (unlikely but
==1040419==  possible), you can try to increase the size of the
==1040419==  main thread stack using the --main-stacksize= flag.
==1040419==  The main thread stack size used in this run was 8388608.
==1040419==
==1040419== HEAP SUMMARY:
==1040419==     in use at exit: 88,190 bytes in 13 blocks
==1040419==   total heap usage: 22 allocs, 9 frees, 2,218,413 bytes allocated
==1040419==
==1040419== LEAK SUMMARY:
==1040419==    definitely lost: 0 bytes in 0 blocks
```

```
==1040419==    indirectly lost: 0 bytes in 0 blocks
==1040419==      possibly lost: 0 bytes in 0 blocks
==1040419==    still reachable: 88,190 bytes in 13 blocks
==1040419==         suppressed: 0 bytes in 0 blocks
==1040419== Rerun with --leak-check=full to see details of leaked memory
==1040419==
==1040419== For lists of detected and suppressed errors, rerun with: -s
==1040419== ERROR SUMMARY: 1 errors from 1 contexts (suppressed: 0 from 0)
==1040433== Memcheck, a memory error detector
==1040433== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==1040433== Using Valgrind-3.18.1 and LibVEX; rerun with -h for copyright info
==1040433== Command: ./src/html2xhtml -t frameset
report/vuln/id:000002,sig:11,src:001386+001369,time:43142960,execs:4309058,op:splice,rep:8
==1040433==
==1040433== Invalid read of size 4
==1040433==    at 0x40E911: elm_close (procesador.c:944)
==1040433==    by 0x410617: err_html_struct (procesador.c:1889)
==1040433==    by 0x40F20A: err_content_invalid (procesador.c:0)
==1040433==    by 0x40F20A: elm_close (procesador.c:959)
==1040433==    by 0x40E7C4: saxEndDocument (procesador.c:233)
==1040433==    by 0x40DF7A: main (html2xhtml.c:117)
==1040433==  Address 0x6efc84 is not stack'd, malloc'd or (recently) free'd
==1040433==
==1040433==
==1040433== Process terminating with default action of signal 11 (SIGSEGV)
==1040433==  Access not within mapped region at address 0x6EFC84
==1040433==    at 0x40E911: elm_close (procesador.c:944)
==1040433==    by 0x410617: err_html_struct (procesador.c:1889)
==1040433==    by 0x40F20A: err_content_invalid (procesador.c:0)
==1040433==    by 0x40F20A: elm_close (procesador.c:959)
==1040433==    by 0x40E7C4: saxEndDocument (procesador.c:233)
==1040433==    by 0x40DF7A: main (html2xhtml.c:117)
==1040433==  If you believe this happened as a result of a stack
==1040433==  overflow in your program's main thread (unlikely but
==1040433==  possible), you can try to increase the size of the
==1040433==  main thread stack using the --main-stacksize= flag.
==1040433==  The main thread stack size used in this run was 8388608.
==1040433==
==1040433== HEAP SUMMARY:
==1040433==     in use at exit: 92,286 bytes in 14 blocks
==1040433==   total heap usage: 23 allocs, 9 frees, 2,222,509 bytes allocated
==1040433==
==1040433== LEAK SUMMARY:
==1040433==    definitely lost: 0 bytes in 0 blocks
==1040433==    indirectly lost: 0 bytes in 0 blocks
==1040433==      possibly lost: 0 bytes in 0 blocks
==1040433==    still reachable: 92,286 bytes in 14 blocks
==1040433==         suppressed: 0 bytes in 0 blocks
==1040433== Rerun with --leak-check=full to see details of leaked memory
==1040433==
==1040433== For lists of detected and suppressed errors, rerun with: -s
==1040433== ERROR SUMMARY: 1 errors from 1 contexts (suppressed: 0 from 0)
==1040439== Memcheck, a memory error detector
==1040439== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==1040439== Using Valgrind-3.18.1 and LibVEX; rerun with -h for copyright info
==1040439== Command: ./src/html2xhtml -t frameset
```

```
report/vuln/id:000003,sig:11,src:001386+001369,time:43143048,execs:4309129,op:splice,rep:8
==1040439==
==1040439== Invalid read of size 4
==1040439==    at 0x40E911: elm_close (procesador.c:944)
==1040439==    by 0x410617: err_html_struct (procesador.c:1889)
==1040439==    by 0x40F20A: err_content_invalid (procesador.c:0)
==1040439==    by 0x40F20A: elm_close (procesador.c:959)
==1040439==    by 0x40E7C4: saxEndDocument (procesador.c:233)
==1040439==    by 0x40DF7A: main (html2xhtml.c:117)
==1040439==  Address 0x6e7074 is not stack'd, malloc'd or (recently) free'd
==1040439==
==1040439==
==1040439== Process terminating with default action of signal 11 (SIGSEGV)
==1040439==  Access not within mapped region at address 0x6E7074
==1040439==    at 0x40E911: elm_close (procesador.c:944)
==1040439==    by 0x410617: err_html_struct (procesador.c:1889)
==1040439==    by 0x40F20A: err_content_invalid (procesador.c:0)
==1040439==    by 0x40F20A: elm_close (procesador.c:959)
==1040439==    by 0x40E7C4: saxEndDocument (procesador.c:233)
==1040439==    by 0x40DF7A: main (html2xhtml.c:117)
==1040439==  If you believe this happened as a result of a stack
==1040439==  overflow in your program's main thread (unlikely but
==1040439==  possible), you can try to increase the size of the
==1040439==  main thread stack using the --main-stacksize= flag.
==1040439==  The main thread stack size used in this run was 8388608.
==1040439==
==1040439== HEAP SUMMARY:
==1040439==     in use at exit: 92,286 bytes in 14 blocks
==1040439==   total heap usage: 23 allocs, 9 frees, 2,222,509 bytes allocated
==1040439==
==1040439== LEAK SUMMARY:
==1040439==    definitely lost: 0 bytes in 0 blocks
==1040439==    indirectly lost: 0 bytes in 0 blocks
==1040439==      possibly lost: 0 bytes in 0 blocks
==1040439==    still reachable: 92,286 bytes in 14 blocks
==1040439==         suppressed: 0 bytes in 0 blocks
==1040439== Rerun with --leak-check=full to see details of leaked memory
==1040439==
==1040439== For lists of detected and suppressed errors, rerun with: -s
==1040439== ERROR SUMMARY: 1 errors from 1 contexts (suppressed: 0 from 0)
```

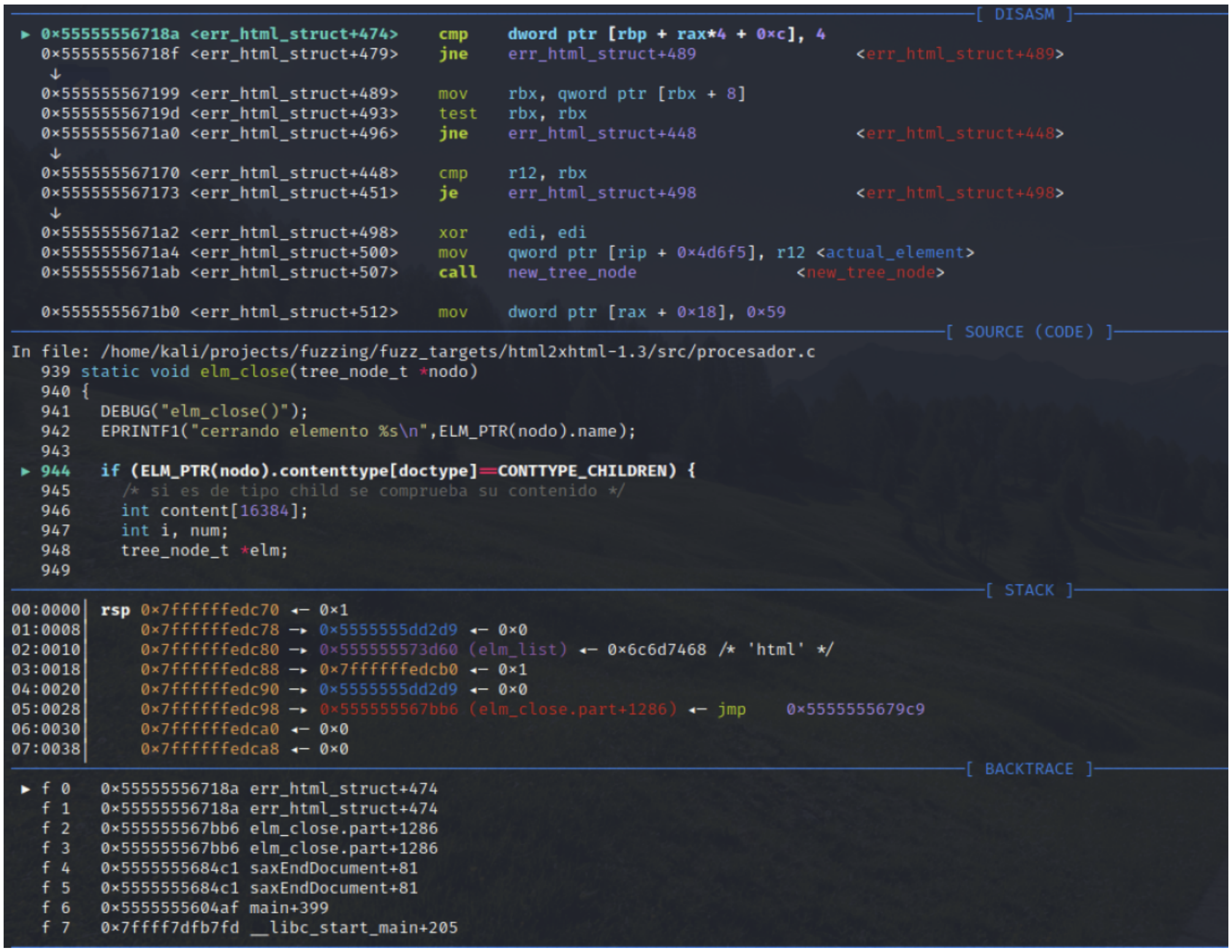
GDB Backtrace and Source Code

I attached gdb to html2xhtml in an attempt to find where the Out of Bounds Read was taking place:



```
pwndbg> r -t frameset report/vuln/id:000003,sig:11,src:001386+001369,time:43143048,execs:4309129,op:splice,rep:8
Starting program: /home/kali/projects/fuzzing/fuzz_targets/html2xhtml-1.3/src/html2xhtml -t frameset report/vuln/id:000003,sig:11,src:001386+001369,time:43143048,execs:4309129,op:splice,rep:8


Program received signal SIGSEGV, Segmentation fault.
0x000055555556718a in elm_close (nodo=0x5555555dd33e) at procesador.c:944
944         if (ELM_PTR(nodo).contenttype[doctype]==CONTTYPE_CHILDREN) {
LEGEND: STACK | HEAP | CODE | DATA | RWX | RODATA
──────────────────────────────────────────────────[ REGISTERS ]──────────────────────────────────────────────────
 RAX  0x ae596
 RBX  0x5555555dd33e ◂— 0x3
 RCX  0x5a
 RDX  0x2
 RDI  0x555555573d60 (elm_list) ◂— 0x6c6d7468 /* 'html' */
 RSI  0x555555573180 (elm_buffer) ◂— 0xd9810100028b8101
 R8   0x1
 R9   0x5555555ee520 ◂— 0x5555555ee
 R10  0x0
 R11  0x7ffff7dfbd60 (iconv_close) ◂— cmp    rdi, -1
 R12  0x5555555dd2d9 ◂— 0x0
 R13  0x7fffffffedcb0 ◂— 0x1
 R14  0x5555555dd2d9 ◂— 0x0
 R15  0x4
 RBP  0x555555573d60 (elm_list) ◂— 0x6c6d7468 /* 'html' */
 RSP  0x7fffffffedc70 ◂— 0x1
 RIP  0x55555556718a (err_html_struct+474) ◂— cmp    dword ptr [rbp + rax*4 + 0xc], 4
```



```
────────────────────────────────────────────────────────[ DISASM ]──────────
 ▶ 0x55555556718a <err_html_struct+474>    cmp    dword ptr [rbp + rax*4 + 0xc], 4
   0x55555556718f <err_html_struct+479>    jne    err_html_struct+489               <err_html_struct+489>
    ↓
   0x555555567199 <err_html_struct+489>    mov    rbx, qword ptr [rbx + 8]
   0x55555556719d <err_html_struct+493>    test   rbx, rbx
   0x5555555671a0 <err_html_struct+496>    jne    err_html_struct+448               <err_html_struct+448>
    ↓
   0x555555567170 <err_html_struct+448>    cmp    r12, rbx
   0x555555567173 <err_html_struct+451>    je     err_html_struct+498               <err_html_struct+498>
    ↓
   0x5555555671a2 <err_html_struct+498>    xor    edi, edi
   0x5555555671a4 <err_html_struct+500>    mov    qword ptr [rip + 0x4d6f5], r12 <actual_element>
   0x5555555671ab <err_html_struct+507>    call   new_tree_node               <new_tree_node>

   0x5555555671b0 <err_html_struct+512>    mov    dword ptr [rax + 0x18], 0x59
───────────────────────────────────────────────────────[ SOURCE (CODE) ]──────
In file: /home/kali/projects/fuzzing/fuzz_targets/html2xhtml-1.3/src/procesador.c
   939 static void elm_close(tree_node_t *nodo)
   940 {
   941   DEBUG("elm_close()");
   942   EPRINTF1("cerrando elemento %s\n",ELM_PTR(nodo).name);
   943
 ▶ 944   if (ELM_PTR(nodo).contenttype[doctype]==CONTTYPE_CHILDREN) {
   945     /* si es de tipo child se comprueba su contenido */
   946     int content[16384];
   947     int i, num;
   948     tree_node_t *elm;
   949
────────────────────────────────────────────────────────────[ STACK ]──────
00:0000│ rsp 0x7fffffffedc70 ◂— 0x1
01:0008│     0x7fffffffedc78 —▸ 0x5555555dd2d9 ◂— 0x0
02:0010│     0x7fffffffedc80 —▸ 0x555555573d60 (elm_list) ◂— 0x6c6d7468 /* 'html' */
03:0018│     0x7fffffffedc88 —▸ 0x7fffffffedcb0 ◂— 0x1
04:0020│     0x7fffffffedc90 —▸ 0x5555555dd2d9 ◂— 0x0
05:0028│     0x7fffffffedc98 —▸ 0x555555567bb6 (elm_close.part+1286) ◂— jmp    0x5555555679c9
06:0030│     0x7fffffffedca0 ◂— 0x0
07:0038│     0x7fffffffedca8 ◂— 0x0
─────────────────────────────────────────────────────────[ BACKTRACE ]──────
 ▶ f 0   0x55555556718a err_html_struct+474
   f 1   0x55555556718a err_html_struct+474
   f 2   0x555555567bb6 elm_close.part+1286
   f 3   0x555555567bb6 elm_close.part+1286
   f 4   0x5555555684c1 saxEndDocument+81
   f 5   0x5555555684c1 saxEndDocument+81
   f 6   0x5555555604af main+399
   f 7   0x7ffff7dfb7fd __libc_start_main+205
```

Taking a look at the segfault in GDB led me to the following function:

html2xhtml/src/procesador.c
Line 940 in `ffb2f1f`

```
940      static void elm_close(tree_node_t *nodo)
```

A user could provide a malformed document with an invalid 'ELM_PTR(nodo).contenttype[doctype]', resulting in the following comparison in assembly:

```
cmp    dword ptr [rbp + rax*4 + 0xc], 4
```

This could be leveraged to read locations that they should not have access to. I have attached multiple crash files to help reproduce the issue.

Thanks again!

crashes.zip

jfisteus added a commit that referenced this issue on Oct 25

`Fix segmentation fault when closing an element` … 9523db7

jfisteus commented on Oct 25 Owner

Thanks for testing html2xhtml and for reporting the issue you've found. The detail of your report and your example files have been quite helpful to triage the bug.

I've found the cause of the issue: the node received by `elm_close` was expected to be of type element but was of type comment instead. I've fixed it and your files don't cause a segmentation fault anymore.

Halcy0nic commented on Oct 25 Author

Awesome, thanks!

Halcy0nic closed this as completed on Oct 25

Assignees

No one assigned

Labels

None yet

Projects

None yet

Milestone

No milestone

Development

No branches or pull requests

2 participants