

New issue

[Jump to bottom](#)

libsolvable "testcase_read" function a heap overflow vulnerability #416

Closed

yangjiageng opened this issue on Dec 13, 2020 · 4 comments

yangjiageng commented on Dec 13, 2020 • edited

Description:

There is a heap overflow bug in function:

```
Solver * testcase_read(Pool *pool, FILE *fp, const char *testcase, Queue *job, char **resultp, int *resultflagsp)
at src/testcase.c: line 2334
```

```
FOR_JOB_SELECT(p, pp, jobset, what)
MAPCLR(pool->considered, p); // line 2334
```

The libsolvable defines MAPCLR(m, n) as following:

```
#define MAPCLR(m, n) ((m)->map[(n) >> 3] &= ~(1 << ((n) & 7)))
, which means that MAPCLR(pool->considered, p) is same as pool->considered->map[p >> 3] &= ~(1 << (p & 7)).
The type of variable "pool->considered" is a Map structure.
The definition of the structure Map as following:
```

```
typedef struct s_Map {
unsigned char *map;
int size;
} Map;
```

If the value of the index variable "p >> 3" is bigger than pool->considered->size, there will be a heap-buffer overflow bug.
Our PoC file can trigger this bug.

Please reproduce this issue through the following PoC: /libsolvableBuildDir/tools/testsolvable PoC-testcase_read-2334
If you configure CC with flag -fsanitize=address, you will get the following outputs:

```
disable: unknown package 'A-3-1.noarch@available'
=====
==24312==ERROR: AddressSanitizer: heap-buffer-overflow on address 0x602000000d1 at pc 0x7f474a02b45e bp 0x7ffff997970 sp 0x7ffff997968
READ of size 1 at 0x602000000d1 thread T0
#0 0x7f474a02b45d in testcase_read /root/Experiments/real-world/libsolvable/ext/testcase.c:2334:6
#1 0x4f144b in main /root/Experiments/real-world/libsolvable/tools/testsolvable.c:159:11
#2 0x7f473fa8abf6 in __libc_start_main /build/glibc-S7xCS9/glibc-2.27/csu/./csu/libc-start.c:310
#3 0x41e6f9 in _start (/root/Experiments/real-world/libsolvable/build/tools/testsolvable+0x41e6f9)

0x602000000d1 is located 0 bytes to the right of 1-byte region [0x602000000d0,0x602000000d1)
allocated by thread T0 here:
#0 0x4abe48 in calloc /root/Downloads/llvm-build/llvm/projects/compiler-rt/lib/asan/asan_malloc_linux.cpp:154
#1 0x7f4740be7f10 in solvable_alloc /root/Experiments/real-world/libsolvable/src/util.c:79:9
#2 0x7f4740a61dba in map_init /root/Experiments/real-world/libsolvable/src/bitmap.c:24:22
#3 0x7f474a02804a in testcase_read /root/Experiments/real-world/libsolvable/ext/testcase.c:2318:8
#4 0x4f144b in main /root/Experiments/real-world/libsolvable/tools/testsolvable.c:159:11
#5 0x7f473fa8abf6 in __libc_start_main /build/glibc-S7xCS9/glibc-2.27/csu/./csu/libc-start.c:310
```

SUMMARY: AddressSanitizer: heap-buffer-overflow /root/Experiments/real-world/libsolvable/ext/testcase.c:2334:6 in testcase_read
Shadow bytes around the buggy address:

```
0x0c047fff7fc0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x0c047fff7fd0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x0c047fff7fe0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x0c047fff7ff0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x0c047fff8000: fa fa fd fd fa 07 fa fa fa 00 00 fa fa 04 fa
=>0x0c047fff8010: fa fa 04 fa fa fa 00 00 fa fa[01]fa fa fa 00 02
0x0c047fff8020: fa fa 00 00 fa fa 04 fa fa fa fa fa fa fa fa
0x0c047fff8030: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
0x0c047fff8040: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
0x0c047fff8050: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
0x0c047fff8060: 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
Shadow gap: cc
==24312==ABORTING

The ASAN outputs information about this overflow bug.
And attacker can use this bug to achieve a DoS attack.
Please reproduce and fix this bug.

mlschroe commented on Dec 14, 2020

Member

Made testcase reader more robust.

 mlschroe closed this as completed on Dec 14, 2020

stevebeattie commented on May 18, 2021

It seems this issue was assigned [CVE-2021-3200](#).

Looking at the commit history, it appears [0077ef2](#) (included in 0.7.17) is the commit that addresses this issue?

mlschroe commented on May 20, 2021

Member

Yes, that's correct.



tcillum-rh commented on May 20, 2021 • edited

How can an attacker use this bug to cause a DoS attack? What is the service being denied?

This looks like a case where a file gets opened using testsolv, it causes a small out-of-bounds read, and at most it'll just crash testsolv (in the case of READ size 1 like here, I don't even think it would do that). So, to "mitigate" this attack, one would just not open that file again, right? Am I missing something? @mlschroe @yangjiageng



Assignees

No one assigned

Labels

None yet

Projects

None yet

Milestone

No milestone

Development

No branches or pull requests

4 participants

