From: Hyunwoo Kim <imv4bel@gmail.com>
To: eli.billauer@gmail.com, arnd@arndb.de, gregkh@linuxfoundation.org
Cc: linux-kernel@vger.kernel.org
Subject: [PATCH] char: xillybus: Fix use-after-free in xillyusb_open()
Date: Sat, 22 Oct 2022 10:54:04 -0700 [thread overview]
Message-ID: <20221022175404.GA375335@ubuntu> (raw)

A race condition may occur if the user physically removes
the USB device while calling open() for this device node.

This is a race condition between the xillyusb_open() function and
the xillyusb_disconnect() function, which may eventually result in UAF.

So, add a mutex to the xillyusb_open() and xillyusb_disconnect()
functions to avoid race contidion.

Signed-off-by: Hyunwoo Kim <imv4bel@gmail.com>
---
 drivers/char/xillybus/xillyusb.c | 21 +++++++++++++++++++++-
 1 file changed, 20 insertions(+), 1 deletion(-)

diff --git a/drivers/char/xillybus/xillyusb.c b/drivers/char/xillybus/xillyusb.c
index 39bcbfd908b4..d615f74dcf36 100644
--- a/drivers/char/xillybus/xillyusb.c
+++ b/drivers/char/xillybus/xillyusb.c
@@ -63,6 +63,7 @@ static const struct usb_device_id xillyusb_table[] = {
 };

 MODULE_DEVICE_TABLE(usb, xillyusb_table);
+static DEFINE_MUTEX(disconnect_mutex);

 struct xillyusb_dev;

@@ -1237,9 +1238,13 @@ static int xillyusb_open(struct inode *inode, struct file *filp)
        int rc;
        int index;

+   mutex_lock(&disconnect_mutex);
+
        rc = xillybus_find_inode(inode, (void **)&xdev, &index);
-       if (rc)
+   if (rc) {
+   mutex_unlock(&disconnect_mutex);
                return rc;
+   }

        chan = &xdev->channels[index];
        filp->private_data = chan;
@@ -1379,6 +1384,8 @@ static int xillyusb_open(struct inode *inode, struct file *filp)
                        request_read_anything(chan, OPCODE_SET_PUSH);
        }

+   mutex_unlock(&disconnect_mutex);
+
        return 0;

 unfifo:
@@ -1410,10 +1417,14 @@ static int xillyusb_open(struct inode *inode, struct file *filp)

        kref_put(&xdev->kref, cleanup_dev);

+   mutex_unlock(&disconnect_mutex);
+
        return rc;

 unmutex_fail:
        mutex_unlock(&chan->lock);
+   mutex_unlock(&disconnect_mutex);
+
        return rc;
 }

@@ -1698,6 +1709,8 @@ static int xillyusb_release(struct inode *inode, struct file *filp)
        struct xillyusb_dev *xdev = chan->xdev;
        int rc_read = 0, rc_write = 0;

+   mutex_lock(&disconnect_mutex);
+
        if (filp->f_mode & FMODE_READ) {
                struct xillyfifo *in_fifo = chan->in_fifo;

@@ -1760,6 +1773,8 @@ static int xillyusb_release(struct inode *inode, struct file *filp)

        kref_put(&xdev->kref, cleanup_dev);

+   mutex_unlock(&disconnect_mutex);
+
        return rc_read ? rc_read : rc_write;
 }

@@ -2172,6 +2187,8 @@ static void xillyusb_disconnect(struct usb_interface *interface)
        int rc;
        int i;

+   mutex_lock(&disconnect_mutex);
+
        xillybus_cleanup_chrdev(xdev, &interface->dev);

        /*
@@ -2228,6 +2245,8 @@ static void xillyusb_disconnect(struct usb_interface *interface)
        xdev->dev = NULL;

        kref_put(&xdev->kref, cleanup_dev);
+
+   mutex_unlock(&disconnect_mutex);
 }

 static struct usb_driver xillyusb_driver = {
--
2.25.1

Dear,

This race condition can occur in the flow of:
```
                cpu0                                                    cpu1
        1. xillyusb_open()
           xillybus_find_inode()
           mutex_lock(&unit_mutex);
           unit = iter;
           mutex_unlock(&unit_mutex);
                                                                2. xillyusb_disconnect()
                                                                   xillybus_cleanup_chrdev()
                                                                   mutex_lock(&unit_mutex);
                                                                   kfree(unit);
                                                                   mutex_unlock(&unit_mutex);
        3. *private_data = unit->private_data;   // UAF

```

Because the interval between 1 and 3 in xillyusb_open() is very short, this race condition is usually rare.

However, if someone wants to trigger this UAF, they can use the technique presented in this paper:
https://www.usenix.org/system/files/sec21-lee-yoochan.pdf

This is a method to increase the execution time between No. 1 and No. 3 using the Reschedule IPI.
This means you will be able to trigger the UAF much more reliably.

Here is the KASAN log:
```
[   66.645661] ==================================================================
[   66.645678] BUG: KASAN: use-after-free in xillybus_find_inode+0x2c6/0x2ea [xillybus_class]
[   66.645712] Read of size 8 at addr ffff88811eb45f90 by task xillyusb_test/2143

[   66.645735] CPU: 2 PID: 2143 Comm: xillyusb_test Not tainted 6.1.0-rc1+ #10
[   66.645752] Hardware name: Gigabyte Technology Co., Ltd. B460MDS3H/B460M DS3H, BIOS F3 05/27/2020
[   66.645762] Call Trace:
[   66.645772]  <TASK>
[   66.645783]  dump_stack_lvl+0x49/0x63
[   66.645808]  print_report+0x177/0x46e
[   66.645828]  ? kasan_complete_mode_report_info+0x7c/0x210
[   66.645846]  ? xillybus_find_inode+0x2c6/0x2ea [xillybus_class]
[   66.645871]  kasan_report+0xb0/0x140
[   66.645891]  ? xillybus_find_inode+0x2c6/0x2ea [xillybus_class]
[   66.645917]  __asan_report_load8_noabort+0x14/0x20
[   66.645932]  xillybus_find_inode+0x2c6/0x2ea [xillybus_class]
[   66.645960]  xillyusb_open+0xa6/0x1030 [xillyusb]
[   66.645990]  ? endpoint_alloc+0x6d0/0x6d0 [xillyusb]
[   66.646014]  ? __kasan_check_write+0x14/0x20
[   66.646028]  ? _raw_spin_lock+0x88/0xe0
[   66.646044]  ? try_module_get.part.0+0xb8/0x1a0
[   66.646062]  chrdev_open+0x230/0x6d0
[   66.646082]  ? cdev_device_add+0x1f0/0x1f0
[   66.646099]  ? fsnotify_perm.part.0+0x1d9/0x4e0
[   66.646118]  do_dentry_open+0x449/0x1000
[   66.646132]  ? inode_permission+0x125/0x560
[   66.646148]  ? cdev_device_add+0x1f0/0x1f0
[   66.646167]  vfs_open+0x9f/0xd0
[   66.646181]  path_openat+0xd58/0x3eb0
[   66.646199]  ? kasan_save_alloc_info+0x1e/0x30
[   66.646212]  ? __kasan_slab_alloc+0x90/0xa0
[   66.646229]  ? kmem_cache_alloc+0x1f6/0x310
[   66.646244]  ? getname_flags.part.0+0x52/0x490
[   66.646260]  ? getname+0x7a/0xb0
[   66.646275]  ? __x64_sys_openat+0x128/0x210
[   66.646288]  ? do_syscall_64+0x59/0x90
[   66.646306]  ? path_lookupat+0x660/0x660
[   66.646328]  do_filp_open+0x1b1/0x3e0
[   66.646344]  ? debug_smp_processor_id+0x17/0x20
[   66.646363]  ? may_open_dev+0xd0/0xd0
[   66.646378]  ? getname_flags.part.0+0x52/0x490
[   66.646399]  ? _raw_spin_lock_bh+0xe0/0xe0
[   66.646422]  do_sys_openat2+0x132/0x450
[   66.646435]  ? recalc_sigpending+0x144/0x1c0
[   66.646450]  ? build_open_flags+0x450/0x450
[   66.646465]  ? sigprocmask+0x201/0x360
[   66.646478]  ? __ia32_sys_ssetmask+0x1d0/0x1d0
[   66.646495]  __x64_sys_openat+0x128/0x210
[   66.646509]  ? __ia32_compat_sys_open+0x1b0/0x1b0
[   66.646526]  ? debug_smp_processor_id+0x17/0x20
[   66.646545]  do_syscall_64+0x59/0x90
[   66.646560]  ? syscall_exit_to_user_mode+0x26/0x50
[   66.646578]  ? do_syscall_64+0x69/0x90
[   66.646593]  ? debug_smp_processor_id+0x17/0x20
[   66.646610]  ? fpregs_assert_state_consistent+0x52/0xc0
[   66.646630]  ? exit_to_user_mode_prepare+0x49/0x1a0
[   66.646644]  ? syscall_exit_to_user_mode+0x26/0x50
[   66.646662]  ? do_syscall_64+0x69/0x90
[   66.646675]  ? exit_to_user_mode_prepare+0x16a/0x1a0
[   66.646689]  ? syscall_exit_to_user_mode+0x26/0x50
[   66.646706]  entry_SYSCALL_64_after_hwframe+0x63/0xcd
[   66.646722] RIP: 0033:0x4534e4
[   66.646736] Code: 24 20 eb 8f 66 90 44 89 54 24 0c e8 d6 ab 02 00 44 8b 54 24 0c 44 89 e2 48 89 ee 41 89 c0 bf 9c ff ff ff b8 01 01 00 00 0f 05 <48> 3d 00 f0 ff ff 77 34 44 89 c7 89 44
24 0c e8 18 ac 02 00 8b 44
[   66.646751] RSP: 002b:00007f8130000140 EFLAGS: 00000293 ORIG_RAX: 0000000000000101
[   66.646771] RAX: ffffffffffffffda RBX: 00007f8130000640 RCX: 00000000004534e4
[   66.646784] RDX: 0000000000000000 RSI: 00000000004b1008 RDI: 00000000ffffff9c
[   66.646795] RBP: 00000000004b1008 R08: 0000000000000000 R09: 00007ffc22c66baf
[   66.646806] R10: 0000000000000000 R11: 0000000000000293 R12: 0000000000000000
[   66.646817] R13: 0000000000000000 R14: 000000000041b2a0 R15: 00007f812f800000
[   66.646834]  </TASK>

[   66.646848] Allocated by task 2115:
[   66.646859]  kasan_save_stack+0x26/0x50
[   66.646875]  kasan_set_track+0x25/0x40
[   66.646888]  kasan_save_alloc_info+0x1e/0x30
[   66.646898]  __kasan_kmalloc+0xb4/0xc0
[   66.646911]  kmalloc_trace+0x4a/0xb0
[   66.646924]  xillybus_init_chrdev+0xf2/0x815 [xillybus_class]
[   66.646944]  xillyusb_probe.cold+0xa61/0xadf [xillyusb]
[   66.646966]  usb_probe_interface+0x266/0x740
[   66.646981]  really_probe+0x1fa/0xa80
[   66.646993]  __driver_probe_device+0x2cb/0x490
[   66.647005]  driver_probe_device+0x4e/0x140
[   66.647017]  __driver_attach+0x1a3/0x520
[   66.647028]  bus_for_each_dev+0x11e/0x1c0
[   66.647039]  driver_attach+0x3d/0x60
[   66.647050]  bus_add_driver+0x449/0x5a0
[   66.647061]  driver_register+0x219/0x390
[   66.647073]  usb_register_driver+0x228/0x400
[   66.647084]  0xffffffffc033202d
[   66.647094]  do_one_initcall+0x97/0x310
[   66.647109]  do_init_module+0x19a/0x630
[   66.647120]  load_module+0x6ca4/0x7d90
[   66.647131]  __do_sys_finit_module+0x134/0x1d0
[   66.647142]  __x64_sys_finit_module+0x72/0xb0
[   66.647153]  do_syscall_64+0x59/0x90
[   66.647164]  entry_SYSCALL_64_after_hwframe+0x63/0xcd

[   66.647182] Freed by task 2089:
[   66.647191]  kasan_save_stack+0x26/0x50
[   66.647205]  kasan_set_track+0x25/0x40
[   66.647218]  kasan_save_free_info+0x2e/0x50
[   66.647228]  ____kasan_slab_free+0x174/0x1e0
[   66.647241]  __kasan_slab_free+0x12/0x20
[   66.647255]  slab_free_freelist_hook+0xd0/0x1a0
[   66.647267]  __kmem_cache_free+0x193/0x2c0
[   66.647280]  kfree+0x79/0x120
[   66.647291]  xillybus_cleanup_chrdev+0x3c1/0x570 [xillybus_class]
[   66.647311]  xillyusb_disconnect+0xfe/0x790 [xillyusb]
[   66.647332]  usb_unbind_interface+0x187/0x7c0
[   66.647345]  device_remove+0x117/0x170
[   66.647357]  device_release_driver_internal+0x418/0x660
[   66.647369]  device_release_driver+0x12/0x20
[   66.647381]  bus_remove_device+0x28f/0x540
[   66.647392]  device_del+0x501/0xc30
[   66.647407]  usb_disable_device+0x2a5/0x660
[   66.647418]  usb_disconnect.cold+0x1f9/0x620
[   66.647431]  hub_event+0x16d3/0x3d20
[   66.647446]  process_one_work+0x778/0x11c0
[   66.647459]  worker_thread+0x544/0x1180
[   66.647471]  kthread+0x280/0x320
[   66.647481]  ret_from_fork+0x1f/0x30

[   66.647501] Last potentially related work creation:
[   66.647510]  kasan_save_stack+0x26/0x50
[   66.647524]  __kasan_record_aux_stack+0xb6/0xd0
[   66.647534]  kasan_record_aux_stack_noalloc+0xb/0x20
[   66.647544]  kvfree_call_rcu+0xaf/0xa50
[   66.647555]  memcg_reparent_list_lrus+0x477/0x790
[   66.647566]  mem_cgroup_css_offline+0x1ea/0x310
[   66.647579]  css_killed_work_fn+0xe4/0x380
[   66.647590]  process_one_work+0x778/0x11c0
[   66.647602]  worker_thread+0x544/0x1180
```

```
[   66.647614]  kthread+0x280/0x320
[   66.647623]  ret_from_fork+0x1f/0x30

[   66.647642] The buggy address belongs to the object at ffff88811eb45f80
                 which belongs to the cache kmalloc-64 of size 64
[   66.647656] The buggy address is located 16 bytes inside of
                 64-byte region [ffff88811eb45f80, ffff88811eb45fc0)

[   66.647677] The buggy address belongs to the physical page:
[   66.647686] page:000000009bdb1cfb refcount:1 mapcount:0 mapping:0000000000000000 index:0xffff88811eb45a00 pfn:0x11eb45
[   66.647701] flags: 0x17ffffc0000200(slab|node=0|zone=2|lastcpupid=0x1fffff)
[   66.647720] raw: 0017ffffc0000200 ffffea000481fec8 ffffea0004544b48 ffff888100042640
[   66.647731] raw: ffff88811eb45a00 000000000020001e 00000001ffffffff 0000000000000000
[   66.647738] page dumped because: kasan: bad access detected

[   66.647751] Memory state around the buggy address:
[   66.647760]  ffff88811eb45e80: 00 00 00 00 00 00 00 00 fc fc fc fc fc fc fc fc
[   66.647771]  ffff88811eb45f00: 00 00 00 00 00 00 fc fc fc fc fc fc fc fc fc fc
[   66.647781] >ffff88811eb45f80: fa fb fb fb fb fb fb fb fc fc fc fc fc fc fc fc
[   66.647790]                    ^
[   66.647800]  ffff88811eb46000: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
[   66.647810]  ffff88811eb46080: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
[   66.647818] ==================================================================
[   66.647826] Disabling lock debugging due to kernel taint
```

Regards,
Hyunwoo Kim.

---

next reply other threads:[~2022-10-22 17:54 UTC|newest]

Thread overview: 4+ messages / expand[flat|nested] mbox.gz Atom feed top
2022-10-22 17:54 Hyunwoo Kim [this message]
2022-10-23 14:19 ` [PATCH] char: xillybus: Fix use-after-free in xillyusb_open() Eli Billauer
2022-10-23 14:26 ` Hyunwoo Kim
2022-10-24 7:10 ` Eli Billauer

find likely ancestor, descendant, or conflicting patches for this message:

dfblob:39bcbfd908b dfblob:d615f74dcf3

search (help)

Reply instructions:

You may reply publicly to this message via plain-text email
using any one of the following methods:

* Save the following mbox file, import it into your mail client,
  and reply-to-all from there: mbox

  Avoid top-posting and favor interleaved quoting:
  https://en.wikipedia.org/wiki/Posting_style#Interleaved_style

* Reply using the **--to, --cc,** and **--in-reply-to**
  switches of git-send-email(1):

```
  git send-email \
    --in-reply-to=20221022175404.GA375335@ubuntu \
    --to=imv4bel@gmail.com \
    --cc=arnd@arndb.de \
    --cc=eli.billauer@gmail.com \
    --cc=gregkh@linuxfoundation.org \
    --cc=linux-kernel@vger.kernel.org \
    /path/to/YOUR_REPLY
```

  https://kernel.org/pub/software/scm/git/docs/git-send-email.html

* If your mail client supports setting the **In-Reply-To** header
  via mailto: links, try the mailto: link

Be sure your reply has a **Subject:** header at the top and a blank line before the message body.
