Date: Thu, 22 Jul 2021 14:28:05 +0800
From: Lin Horse <kylin.formalin@...il.com>
To: oss-security@...ts.openwall.com
Subject: CVE-2021-3640: Linux kernel: UAF in sco_send_frame function

Hello there,

Just like the previous, tedious race condition vulnerability caused by the
unexpected locking behavior (CVE-2021-3573), a similar one is found this
time.

=*=*=*=*=*=*=*=*= BUG DETAILS =*=*=*=*=*=*=*=*=

We can find another place that uses bh_lock_sock() in the Linux Bluetooth
stacks.

```
static void sco_conn_del(struct hci_conn *hcon, int err)
{
    ...
    if (sk) {
        sock_hold(sk);
        bh_lock_sock(sk);   // {1} LOCK
        sco_sock_clear_timer(sk);
        sco_chan_del(sk, err);
        bh_unlock_sock(sk); // {2} UNLOCK
        sco_sock_kill(sk);
        sock_put(sk);
    }
    ...
    hcon->sco_data = NULL;
    kfree(conn);
}
```

Between these lock pairs, sco_chan_del() is called, which will delete the
channel associated with this sk.
At the end of this function, the conn will be released by kfree().

Similar to the CVE-2021-3573, there is another thread that can be
controlled by the attacker. It will wait for the kfree() and thereafter,
race to cause UAF.

For example, the sco_sock_sendmsg() function.

```
static int sco_sock_sendmsg(struct socket *sock, struct msghdr *msg,
                size_t len)
{
    ...

    lock_sock(sk);

    if (sk->sk_state == BT_CONNECTED)
        err = sco_send_frame(sk, msg, len);
    else
        err = -ENOTCONN;

    release_sock(sk);
    return err;
}

static int sco_send_frame(struct sock *sk, struct msghdr *msg, int len)
{
    ...

    skb = bt_skb_send_alloc(sk, len, msg->msg_flags & MSG_DONTWAIT, &err);
    if (!skb)
        return err;

    if (memcpy_from_msg(skb_put(skb, len), msg, len)) { // {3}
        kfree_skb(skb);
        return -EFAULT;
    }

    hci_send_sco(conn->hcon, skb);

    ...
}
```

As you can see, the attacker can adopt userfaultfd technique to stop the
thread at {3} point.

Because the sco_send_frame() is protected by the lock_sock() and
release_sock(), which will not block the sco_conn_del() to release the conn.

One vulnerable race window is shown below:

```
sco_sock_sendmsg thread         |    sco_conn_del thread
                                |
                                |
lock_sock(sk);                  |
                                |
...                             |    bh_lock_sock(sk);
                                |    ...
                                |    bh_unlock_sock(sk);
                                |    ...
                                |    kfree(conn);
// UAF          |
hci_send_sco(conn->hcon, skb); |
                                |
                                |
```

=*=*=*=*=*=*=*=*= BUG EFFECTS =*=*=*=*=*=*=*=*=

Similar to CVE-2021-3573, the attacker may stably cause the UAF and do
further exploitation.

As the sco_conn struct is pretty juicy (two previous data pointers inside)

```
struct sco_conn {
    struct hci_conn    *hcon;

    spinlock_t    lock;
    struct sock    *sk;

    unsigned int    mtu;
};
```

The attacker can easily spray these kmalloc-32 objects with the malicious
payload, with CAP_NET_ADMIN privilege.

The provided POC code can cause the crash report below:

```
[   62.856933]
==================================================================
[   62.857336] BUG: KASAN: use-after-free in sco_sock_sendmsg+0x1d6/0x2c0
[   62.858202] Read of size 8 at addr ffff888002478540 by task
poc.sco.new/120
[   62.858663]
[   62.859014] CPU: 0 PID: 120 Comm: poc.sco.new Not tainted 5.13.0+ #1
[   62.859405] Hardware name: QEMU Standard PC (i440FX + PIIX, 1996), BIOS
```

```
1.10.2-1ubuntu1 04/01/2014
[   62.859884] Call Trace:
[   62.860168]  dump_stack_lvl+0x73/0x9e
[   62.860525]  print_address_description+0x82/0x3a0
[   62.860879]  __kasan_report+0x154/0x240
[   62.861115]  ? lock_sock_nested+0x100/0x140
[   62.861446]  ? sco_sock_sendmsg+0x1d6/0x2c0
[   62.861811]  kasan_report+0x45/0x60
[   62.862133]  sco_sock_sendmsg+0x1d6/0x2c0
[   62.862461]  ? sco_sock_getsockopt+0x410/0x410
[   62.862748]  ? inet_send_prepare+0x190/0x190
[   62.863000]  sock_write_iter+0x21b/0x230
[   62.863232]  vfs_write+0x53a/0x5c0
[   62.863479]  ksys_write+0x8b/0x100
[   62.863723]  ? __fpregs_load_activate+0xc2/0x150
[   62.864017]  do_syscall_64+0x43/0x90
[   62.864287]  entry_SYSCALL_64_after_hwframe+0x44/0xae
[   62.864615] RIP: 0033:0x7f9b6c8d4abf
[   62.865073] Code: 89 54 24 18 48 89 74 24 10 89 7c 24 08 e8 69 fd ff ff
48 8b 54 24 18 48 8b 74 24 10 41 89 c0 8b 7c 24 08 b8 01 00 00 00 0f 05
<48> 3d 00 f0 ff ff 77 2d 44 89 c7 48 89 44 24 08 e8 9c fd ff ff 48
[   62.865843] RSP: 002b:00007ffd6b0133a0 EFLAGS: 00000293 ORIG_RAX:
0000000000000001
[   62.866304] RAX: ffffffffffffffda RBX: 000055be494024e0 RCX:
00007f9b6c8d4abf
[   62.866660] RDX: 0000000000000010 RSI: 00007f9b6c90e000 RDI:
0000000000000005
[   62.866992] RBP: 00007ffd6b013480 R08: 0000000000000000 R09:
00007f9b6c703700
[   62.867293] R10: 00007f9b6c7039d0 R11: 0000000000000293 R12:
000055be49400d10
[   62.867576] R13: 00007ffd6b013570 R14: 0000000000000000 R15:
0000000000000000
[   62.868106]
[   62.868302] Allocated by task 120:
[   62.868586]  ____kasan_kmalloc+0xb5/0xe0
[   62.868999]  kmem_cache_alloc_trace+0x12d/0x210
[   62.869349]  sco_sock_connect+0x1f7/0x4a0
[   62.869647]  __sys_connect+0x16f/0x1a0
[   62.869944]  __x64_sys_connect+0x38/0x40
[   62.870243]  do_syscall_64+0x43/0x90
[   62.870556]  entry_SYSCALL_64_after_hwframe+0x44/0xae
[   62.870883]
[   62.871020] Freed by task 125:
[   62.871192]  kasan_set_track+0x3d/0x70
[   62.871432]  kasan_set_free_info+0x1f/0x40
[   62.871708]  ____kasan_slab_free+0x111/0x150
[   62.871956]  kfree+0xf3/0x2d0
[   62.872208]  hci_conn_hash_flush+0xbf/0x120
[   62.872529]  hci_dev_do_close+0x51a/0x870
[   62.872789]  hci_unregister_dev+0x23a/0xb70
[   62.873054]  vhci_release+0x3f/0x70
[   62.873334]  __fput+0x197/0x360
[   62.873598]  task_work_run+0xc0/0xe0
[   62.873919]  exit_to_user_mode_prepare+0xf0/0x130
[   62.874253]  syscall_exit_to_user_mode+0x20/0x40
[   62.874511]  do_syscall_64+0x52/0x90
[   62.874768]  entry_SYSCALL_64_after_hwframe+0x44/0xae
[   62.875160]
[   62.875352] The buggy address belongs to the object at ffff888002478540
[   62.875352]  which belongs to the cache kmalloc-32 of size 32
[   62.875900] The buggy address is located 0 bytes inside of
[   62.875900]  32-byte region [ffff888002478540, ffff888002478560)
[   62.876472] The buggy address belongs to the page:
[   62.876885] page:00000000db13206d refcount:1 mapcount:0
mapping:0000000000000000 index:0x0 pfn:0x2478
[   62.877481] flags: 0x100000000000200(slab|node=0|zone=1)
[   62.878361] raw: 0100000000000200 ffffea0000078d00 0000000e0000000e
ffff888001041500
[   62.878901] raw: 0000000000000000 0000000080400040 00000001ffffffff
0000000000000000
[   62.879313] page dumped because: kasan: bad access detected
[   62.879588]
[   62.879704] Memory state around the buggy address:
[   62.880003]  ffff888002478400: fb fb fb fb fc fc fc fc fb fb fb fb fc fc
fc fc
[   62.880286]  ffff888002478480: fb fb fb fb fc fc fc fc fb fb fb fb fc fc
fc fc
[   62.880532] >ffff888002478500: fa fb fb fb fc fc fc fc fa fb fb fb fc fc
fc fc
[   62.880785]                                            ^
[   62.881199]  ffff888002478580: 00 00 00 00 fc fc fc fc 00 00 00 fc fc fc
fc fc
[   62.881457]  ffff888002478600: 00 00 00 fc fc fc fc fc 00 00 00 fc fc fc
fc fc
[   62.881716]
==================================================================
[   62.881991] Disabling lock debugging due to kernel taint
[   62.883072] BUG: unable to handle page fault for address:
fffffbfff22fa79f
[   62.883427] #PF: supervisor read access in kernel mode
[   62.883774] #PF: error_code(0x0000) - not-present page
[   62.884165] PGD 36fd0067 P4D 36fd0067 PUD 36df4067 PMD 0
[   62.884827] Oops: 0000 [#1] SMP KASAN NOPTI
[   62.885132] CPU: 0 PID: 120 Comm: poc.sco.new Tainted: G    B
  5.13.0+ #1
[   62.885528] Hardware name: QEMU Standard PC (i440FX + PIIX, 1996), BIOS
1.10.2-1ubuntu1 04/01/2014
[   62.885901] RIP: 0010:__asan_store8+0x6c/0xb0
[   62.886184] Code: be 00 00 00 00 00 fc ff df 0f be 14 32 85 d2 74 07 83
e0 07 39 d0 7d 29 c3 48 89 fe 48 c1 ee 03 48 ba 00 00 00 00 00 fc ff df
<80> 3c 16 00 75 11 48 89 c6 48 c1 ee 03 0f be 14 16 85 d2 75 d2 eb
[   62.886853] RSP: 0018:ffff8880030ffbf8 EFLAGS: 00000006
[   62.887244] RAX: ffffffff917d3d02 RBX: 0000000000040000 RCX:
fffffffba337d86
[   62.887524] RDX: dffffc0000000000 RSI: 1fffffff22fa79f RDI:
ffffffff917d3cfb
[   62.887855] RBP: 0000000000000030 R08: dffffc0000000000 R09:
0000000000000007
[   62.888162] R10: ffffed100035159c R11: 00000000000000fb R12:
ffffffff917a1b4b
[   62.888476] R13: fffffffffffffff8 R14: ffff888001a8acdc R15:
ffff888036432188
[   62.888838] FS:  00007f9b6c704740(0000) GS:ffff888036400000(0000)
knlGS:0000000000000000
[   62.889331] CS:  0010 DS: 0000 ES: 0000 CR0: 0000000080050033
[   62.889908] CR2: fffffbfff22fa79f CR3: 00000000011c0000 CR4:
00000000003006f0
[   62.890341] Call Trace:
[   62.890617]  queued_spin_lock_slowpath+0x286/0x410
[   62.890915]  _raw_spin_lock_irqsave+0x9f/0xb0
[   62.891201]  skb_queue_tail+0x1c/0x90
[   62.891548]  hci_send_sco+0xd6/0x110
[   62.891871]  sco_sock_sendmsg+0x1e1/0x2c0
[   62.892170]  ? sco_sock_getsockopt+0x410/0x410
[   62.892511]  ? inet_send_prepare+0x190/0x190
[   62.892796]  sock_write_iter+0x21b/0x230
[   62.893156]  vfs_write+0x53a/0x5c0
[   62.893533]  ksys_write+0x8b/0x100
[   62.893870]  ? __fpregs_load_activate+0xc2/0x150
[   62.894258]  do_syscall_64+0x43/0x90
[   62.894523]  entry_SYSCALL_64_after_hwframe+0x44/0xae
[   62.894929] RIP: 0033:0x7f9b6c8d4abf
[   62.895178] Code: 89 54 24 18 48 89 74 24 10 89 7c 24 08 e8 69 fd ff ff
48 8b 54 24 18 48 8b 74 24 10 41 89 c0 8b 7c 24 08 b8 01 00 00 00 0f 05
<48> 3d 00 f0 ff ff 77 2d 44 89 c7 48 89 44 24 08 e8 9c fd ff ff 48
[   62.895930] RSP: 002b:00007ffd6b0133a0 EFLAGS: 00000293 ORIG_RAX:
0000000000000001
[   62.896396] RAX: ffffffffffffffda RBX: 000055be494024e0 RCX:
00007f9b6c8d4abf
```

```
[   62.896749] RDX: 0000000000000010 RSI: 00007f9b6c90e000 RDI:
0000000000000005
[   62.897081] RBP: 00007ffd6b013480 R08: 0000000000000000 R09:
00007f9b6c703700
[   62.897430] R10: 00007f9b6c7039d0 R11: 0000000000000293 R12:
000055be49400d10
[   62.897814] R13: 00007ffd6b013570 R14: 0000000000000000 R15:
0000000000000000
[   62.898239] Modules linked in:
[   62.898623] CR2: fffffbfff22fa79f
[   62.899350] ---[ end trace e705e323d4c8b589 ]---
[   62.899645] RIP: 0010:__asan_store8+0x6c/0xb0
[   62.899918] Code: be 00 00 00 00 00 fc ff df 0f be 14 32 85 d2 74 07 83
e0 07 39 d0 7d 29 c3 48 89 fe 48 c1 ee 03 48 ba 00 00 00 00 00 fc ff df
<80> 3c 16 00 75 11 48 89 c6 48 c1 ee 03 0f be 14 16 85 d2 75 d2 eb
[   62.900625] RSP: 0018:ffff8880030ffbf8 EFLAGS: 00000006
[   62.900997] RAX: ffffffff917d3d02 RBX: 0000000000040000 RCX:
ffffffffba337d86
[   62.901276] RDX: dfffffc0000000000 RSI: 1ffffffff22fa79f RDI:
ffffffff917d3cfb
[   62.901700] RBP: 0000000000000030 R08: dfffffc0000000000 R09:
0000000000000007
[   62.902083] R10: ffffed100035159c R11: 00000000000000fb R12:
ffffffff917a1b4b
[   62.902496] R13: fffffffffffffff8 R14: ffff888001a8acdc R15:
ffff888036432188
[   62.902820] FS:  00007f9b6c704740(0000) GS:ffff888036400000(0000)
knlGS:0000000000000000
[   62.903228] CS:  0010 DS: 0000 ES: 0000 CR0: 0000000080050033
[   62.903566] CR2: fffffbfff22fa79f CR3: 00000000011c0000 CR4:
00000000003006f0

=*=*=*=*=*=*=*=*=  BUG REPRODUCE  =*=*=*=*=*=*=*=*=

As above introduced, this race condition is highly controllable with
userfaultfd techniques.

The attacker has to fake an SCO connection and then calls
sco_sock_sendmsg() with the expected controllable faulting page.
After that, the attacker just needs to detach the controller to call
sco_conn_del().

The calling trace is:

hci_unregister_dev() -> hci_dev_do_close() -> hci_conn_hash_flush() ->
hci_disconn_cfm() ->
sco_disconn_cfm() -> sco_conn_del().

You can refer to the provided POC code for the details.

=*=*=*=*=*=*=*=*=  Timeline  =*=*=*=*=*=*=*=*=

2021-07-08: Bug reported to security@...nel.org and
linux-distros@...openwall.org
2021-07-09: CVE-2021-3640 is assigned
2021-07-22: 14 days of the embargo is over

One sad thing is that the bluez team is currently focused on fixing up the
CVE-2021-3573, which I failed to properly patched, and the patch for this
new is not yet fully discussed.
I hope the patch will be settled down and merged to the mainline in the
near future.

=*=*=*=*=*=*=*=*=  Credt  =*=*=*=*=*=*=*=*=
LinMa@...ckSec Team

Best Regards
```

Content of type "text/html**" skipped**

Download attachment "reproduce.tar**" of type "**application/x-tar**" (51200 bytes)**