```
Date: Tue, 1 Jun 2021 09:17:12 +0800
From: Mart1l1n <mmmart1lnnnnn@...il.com>
To: oss-security@...ts.openwall.com
Subject: Re: CVE-2021-3564 Linux Bluetooth device initialization
 implementation bug

Because it's been a week since I reported this issue, it's time to post the
POC to the public and the POC is in attachment.

Thanks,

Mart1l1n

On Tue, 25 May 2021 at 15:18, Mart1l1n <mmmart1lnnnnn@...il.com> wrote:

> Hello there,
>
> Our team (BlockSec) found an implementation bug that resides in the kernel
> BlueTooth subsystem when the HCI device initialization fails. It can lead
> to unexpected results, like double-free memory corruption vulnerability.
>
> =*=*=*=*=*=*=*=*=  BUG DETAILS  =*=*=*=*=*=*=*=*=
>
> This implementation bug is inside hci_dev_do_open() function.
>
> static int hci_dev_do_open(struct hci_dev *hdev)
> {
> ...
>     } else {
>         /* Init failed, cleanup */
>         flush_work(&hdev->tx_work);
>         flush_work(&hdev->cmd_work);  // {1}
>         flush_work(&hdev->rx_work);   // {2}
>
>         skb_queue_purge(&hdev->cmd_q);
>         skb_queue_purge(&hdev->rx_q);
>
>         if (hdev->flush)
>           hdev->flush(hdev);
>
>         if (hdev->sent_cmd) {
>           kfree_skb(hdev->sent_cmd);
>           hdev->sent_cmd = NULL;
>         }
> ...
> }
>
> The purpose of flush_work(struct work_struct *work) is to wait for the
> accomplishment of the work_struct. Hence, the accomplishment of the code
> flush_work(&hdev->cmd_work) {1} means the cmd_work is finished. However, we
> discover an implementation bug that can result in activating hci_cmd_work()
> even the hdev->cmd_work has already been flushed {2}.
>
> The process is as follows:
> hci_rx_work() -> hci_event_packet() -> hci_event_packet() ->
> hci_cmd_complete_evt() -> queue_work(hdev->workqueue, &hdev->cmd_work)
>
> We found this implementation bug can lead to double-free memory
> corruption, which resulted from a data race of the hdev->sent_cmd. Here is
> the code snippet for this race.
>
> static void hci_cmd_work(struct work_struct *work)
> {
> ...
>     if (atomic_read(&hdev->cmd_cnt)) {
>         skb = skb_dequeue(&hdev->cmd_q);
>         if (!skb)
>             return;
>
>         kfree_skb(hdev->sent_cmd);
>
>         hdev->sent_cmd = skb_clone(skb, GFP_KERNEL);
> ...
> }
>
>
> We use thread-A to represent hci_dev_do_open() function and the thread-B
> for hci_cmd_work().
> The normal sequence should be like this:
>
>
> ----------------------------------------------------------------------------------------------------
> thread-A                                |  thread-B
>                                         |  kfree_skb(hdev->sent_cmd); (FREE)
>                                         |
>                                         |  hdev->sent_cmd = skb_clone(skb,
> GFP_KERNEL); (WRITE)
> if (hdev->sent_cmd) { (READ)            |
>                                         |
> kfree_skb(hdev->sent_cmd); (FREE)       |
>                                         |
> hdev->sent_cmd = NULL; (WRITE)          |
>                                         |
>
> ----------------------------------------------------------------------------------------------------
>
> However, if the sequence is like this:
>
>
> ----------------------------------------------------------------------------------------------------
> thread-A                                |  thread-B
>                                         |  kfree_skb(hdev->sent_cmd); (FREE)
> if (hdev->sent_cmd) { (READ)            |
>                                         |
> kfree_skb(hdev->sent_cmd); (FREE)       |
>                                         |  hdev->sent_cmd = skb_clone(skb,
> GFP_KERNEL); (WRITE)
>                                         |
> hdev->sent_cmd = NULL; (WRITE)          |
>                                         |
>
> ----------------------------------------------------------------------------------------------------
>
> If the FREE operation in thread-A is before WRITE operation in thread-B,
> it can lead to double-free memory corruption in the kernel.
>
>
> =*=*=*=*=*=*=*=*=  BUG EFFECTS  =*=*=*=*=*=*=*=*=
>
> For now, we can successfully trigger the vulnerability to corrupt the
> kernel memory and thus crash the kernel. Although this bug is related to
> Bluetooth device initialization, the attacker can trigger it without extra
> privileges.
>
> That is because the Linux kernel does not ask for the privilege when
> attaching the HCI device as the attached device is default set to
> HCI_AUTO_OFF state. This bug is inside in the very first attaching
> procedure and requires no syscalls.
>
```

```
> The crash log is presented below.
>
> ==================================================================
> [  500.906562] hci0 type 1 len 3
> [  500.904986] BUG: KASAN: use-after-free in kfree_skb+0x33/0x1c0
> [  500.904986] Read of size 4 at addr ffff888009d3599c by task
> kworker/u5:0/54
> [  500.904986]
> [  500.909997] CPU: 0 PID: 54 Comm: kworker/u5:0 Not tainted 5.11.11+ #16
> [  500.909997] Hardware name: QEMU Standard PC (i440FX + PIIX, 1996), BIOS
> 1.13.0-1ubuntu1.1 04/01/2014
> [  500.909997] Workqueue: hci0 hci_power_on
> [  500.909997] Call Trace:
> [  500.909997]  dump_stack+0x16c/0x1be
> [  500.924511]  print_address_description+0x7b/0x3a0
> [  500.924511]  __kasan_report+0x14e/0x200
> [  500.924511]  ? kfree_skb+0x33/0x1c0
> [  500.924511]  ? skb_queue_purge+0x193/0x1c0
> [  500.924511]  kasan_report+0x47/0x60
> [  500.924511]  ? skb_queue_purge+0x193/0x1c0
> [  500.924511]  check_memory_region+0x2e2/0x330
> [  500.924511]  kfree_skb+0x33/0x1c0
> [  500.924511]  hci_dev_do_open+0x1008/0x1570
> [  500.924511]  ? printk+0x62/0x83
> [  500.924511]  hci_power_on+0x183/0x580
> [  500.924511]  ? strscpy+0x7f/0x240
> [  500.924511]  process_one_work+0x722/0x1150
> [  500.924511]  worker_thread+0xb5c/0x17d0
> [  500.924511]  ? process_one_work+0x1150/0x1150
> [  500.924511]  kthread+0x2fc/0x320
> [  500.924511]  ? process_one_work+0x1150/0x1150
> [  500.924511]  ? kthread_unuse_mm+0x1d0/0x1d0
> [  500.924511]  ret_from_fork+0x22/0x30
> [  500.924511]
> [  500.924511] Allocated by task 273:
> [  500.924511]  ____kasan_kmalloc+0xc6/0x100
> [  500.924511]  kmem_cache_alloc+0xfe/0x1f0
> [  500.924511]  skb_clone+0x1b5/0x360
> [  500.924511]  hci_cmd_work+0x15d/0x350
> [  500.924511]  process_one_work+0x722/0x1150
> [  500.924511]  worker_thread+0xb5c/0x17d0
> [  500.924511]  kthread+0x2fc/0x320
> [  500.924511]  ret_from_fork+0x22/0x30
> [  500.924511]
> [  500.924511] Freed by task 273:
> [  500.924511]  kasan_set_track+0x3d/0x70
> [  500.924511]  kasan_set_free_info+0x1f/0x40
> [  500.924511]  ____kasan_slab_free+0x10e/0x140
> [  500.924511]  kmem_cache_free+0xca/0x210
> [  500.924511]  hci_cmd_work+0x150/0x350
> [  500.924511]  process_one_work+0x722/0x1150
> [  500.924511]  worker_thread+0xb5c/0x17d0
> [  500.924511]  kthread+0x2fc/0x320
> [  500.924511]  ret_from_fork+0x22/0x30
> [  500.924511]
> [  500.924511] The buggy address belongs to the object at ffff888009d358c0
> [  500.924511]  which belongs to the cache skbuff_head_cache of size 232
> [  500.924511] The buggy address is located 220 bytes inside of
> [  500.924511]  232-byte region [ffff888009d358c0, ffff888009d359a8)
> [  500.924511] The buggy address belongs to the page:
> [  500.924511] page:00000000b691648a refcount:1 mapcount:0
> mapping:0000000000000000 index:0x0 pfn:0x9d35
> [  500.924511] flags: 0x100000000000200(slab)
> [  500.924511] raw: 0100000000000200 dead000000000100 dead000000000122
> ffff888006d64640
> [  500.924511] raw: 0000000000000000 00000000000c000c 00000001ffffffff
> 0000000000000000
> [  500.924511] page dumped because: kasan: bad access detected
> [  500.924511]
> [  500.924511] Memory state around the buggy address:
> [  500.924511]  ffff888009d35880: fc fc fc fc fc fc fc fc fa fb fb fb fb
> fb fb fb
> [  500.924511]  ffff888009d35900: fb fb fb fb fb fb fb fb fb fb fb fb fb
> fb fb fb
> [  500.924511] >ffff888009d35980: fb fb fb fb fb fc fc fc fc fc fc fc fc
> fc fc fc
> [  500.924511]                             ^
> [  500.924511]  ffff888009d35a00: fa fb fb fb fb fb fb fb fb fb fb fb fb
> fb fb fb
> [  500.924511]  ffff888009d35a80: fb fb fb fb fb fb fb fb fb fb fb fb fb
> fc fc fc
> [  500.924511]
> ==================================================================
> [  500.924511] Disabling lock debugging due to kernel taint
> [  501.014277]
> ==================================================================
> [  501.014929] BUG: KASAN: double-free or invalid-free in
> hci_dev_do_open+0x1008/0x1570
> [  501.014929]
> [  501.014929] CPU: 0 PID: 54 Comm: kworker/u5:0 Tainted: G    B
>   5.11.11+ #16
> [  501.014929] Hardware name: QEMU Standard PC (i440FX + PIIX, 1996), BIOS
> 1.13.0-1ubuntu1.1 04/01/2014
> [  501.014929] Workqueue: hci0 hci_power_on
> [  501.014929] Call Trace:
> [  501.014929]  dump_stack+0x16c/0x1be
> [  501.014929]  ? hci_dev_do_open+0x1008/0x1570
> [  501.014929]  ? hci_dev_do_open+0x1008/0x1570
> [  501.014929]  print_address_description+0x7b/0x3a0
> [  501.014929]  ? hci_dev_do_open+0x1008/0x1570
> [  501.014929]  ? hci_dev_do_open+0x1008/0x1570
> [  501.014929]  kasan_report_invalid_free+0x54/0xd0
> [  501.014929]  ____kasan_slab_free+0xe7/0x140
> [  501.014929]  kmem_cache_free+0xca/0x210
> [  501.014929]  ? hci_dev_do_open+0x1008/0x1570
> [  501.014929]  hci_dev_do_open+0x1008/0x1570
> [  501.014929]  ? printk+0x62/0x83
> [  501.014929]  hci_power_on+0x183/0x580
> [  501.014929]  ? strscpy+0x7f/0x240
> [  501.014929]  process_one_work+0x722/0x1150
> [  501.014929]  worker_thread+0xb5c/0x17d0
> [  501.014929]  ? process_one_work+0x1150/0x1150
> [  501.014929]  kthread+0x2fc/0x320
> [  501.014929]  ? process_one_work+0x1150/0x1150
> [  501.014929]  ? kthread_unuse_mm+0x1d0/0x1d0
> [  501.014929]  ret_from_fork+0x22/0x30
> [  501.014929]
> [  501.014929] Allocated by task 273:
> [  501.014929]  ____kasan_kmalloc+0xc6/0x100
> [  501.014929]  kmem_cache_alloc+0xfe/0x1f0
> [  501.014929]  skb_clone+0x1b5/0x360
> [  501.014929]  hci_cmd_work+0x15d/0x350
> [  501.014929]  process_one_work+0x722/0x1150
> [  501.014929]  worker_thread+0xb5c/0x17d0
> [  501.014929]  kthread+0x2fc/0x320
> [  501.014929]  ret_from_fork+0x22/0x30
> [  501.014929]
> [  501.014929] Freed by task 273:
> [  501.014929]  kasan_set_track+0x3d/0x70
> [  501.066803]  kasan_set_free_info+0x1f/0x40
> [  501.066803]  ____kasan_slab_free+0x10e/0x140
> [  501.066803]  kmem_cache_free+0xca/0x210
> [  501.066803]  hci_cmd_work+0x150/0x350
> [  501.066803]  process_one_work+0x722/0x1150
> [  501.066803]  worker_thread+0xb5c/0x17d0
> [  501.066803]  kthread+0x2fc/0x320
> [  501.066803]  ret_from_fork+0x22/0x30
> [  501.066803]
> [  501.066803] The buggy address belongs to the object at ffff888009d358c0
> [  501.066803]  which belongs to the cache skbuff_head_cache of size 232
> [  501.066803] The buggy address is located 0 bytes inside of
> [  501.066803]  232-byte region [ffff888009d358c0, ffff888009d359a8)
> [  501.066803] The buggy address belongs to the page:
> [  501.066803] page:00000000b691648a refcount:1 mapcount:0
> mapping:0000000000000000 index:0x0 pfn:0x9d35
> [  501.066803] flags: 0x100000000000200(slab)
> [  501.066803] raw: 0100000000000200 dead000000000100 dead000000000122
> ffff888006d64640
> [  501.066803] raw: 0000000000000000 00000000c000c 00000001ffffffff
> 0000000000000000
> [  501.066803] page dumped because: kasan: bad access detected
> [  501.066803]
> [  501.066803] Memory state around the buggy address:
> [  501.066803]  ffff888009d35780: fa fb fb fb fb fb fb fb fb fb fb fb fb
> fb fb fb
> [  501.066803]  ffff888009d35800: fb fb fb fb fb fb fb fb fb fb fb fb fb
> fc fc fc
> [  501.066803] >ffff888009d35880: fc fc fc fc fc fc fc fc fa fb fb fb fb
> fb fb fb
> [  501.066803]                                            ^
> [  501.066803]  ffff888009d35900: fb fb fb fb fb fb fb fb fb fb fb fb fb
> fb fb fb
> [  501.066803]  ffff888009d35980: fb fb fb fb fb fc fc fc fc fc fc fc fc
> fc fc fc
> [  501.066803]
> ==================================================================
>
> =*=*=*=*=*=*=*=*=  Timeline  =*=*=*=*=*=*=*=*=
>
> 2021-05-17: Bug reported to security () kernel org and linux-distros () vs openwall org
>
> 2021-05-25: CVE-2021-3564 assigned
>
> We informed security@...nel.org on May 17, 2021. Now the 7-day embargo period is over, we are being asked to bring the issue to public.
>
> Since our patch has not been applied to upstream yet, we will release the POC later.
>
> =*=*=*=*=*=*=*=*=  Credit  =*=*=*=*=*=*=*=*=
>
> HaoXiong@...ckSec Team
>
> LinMa@...cksec Team
>
> syzkaller
>
>
>
> Best regards.
>
> Mart1ll1n
>
```

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

Download attachment "reproduce.zip**" of type "**application/zip**" (8680 bytes)**