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

[<prev] [next>] [day] [month] [year] [list]

Date: Sat, 2 Apr 2022 14:53:10 +0800 (GMT+08:00)
From: 周多明 <duoming@....edu.cn>
To: oss-security@...ts.openwall.com
Subject: CVE-2022-1204: Linux kernel: UAF caused by binding operation when
 ax25 device is detaching

Hello there,

There are use-after-free vulnerabilities in net/ax25/af_ax25.c of linux that allow
attacker to crash linux kernel by simulating ax25 device from user space.

=*=*=*=*=*=*=*=*= Bug Details =*=*=*=*=*=*=*=*=

The resources such as ax25_dev and net_device will be freed in ax25_dev_device_down(),
if we call ax25_bind() between ax25_kill_by_device() and kfree() in ax25_dev_device_down(),
we could use ax25_dev in functions such as ax25_bind() ax25_release(), ax25_connect(),
ax25_ioctl(), ax25_getname(), ax25_sendmsg(), ax25_getsockopt() and ax25_info_show() after
ax25_dev has been deallocated, and use net_device in functions such as ax25_release(),
ax25_sendmsg(), ax25_getsockopt(), ax25_getname() and ax25_info_show() after net_device
has been deallocated.

One of the concurrency UAF related with ax25_dev in ax25_bind() can be shown as below:

```
       (USE)                  |     (FREE)
                              |  ax25_device_event
                              |    ax25_dev_device_down
ax25_bind                     |    ...
  ...                         |      kfree(ax25_dev)
  ax25_fillin_cb()            |    ...
    ax25_fillin_cb_from_dev() |
  ...                         |
```

One of the concurrency UAF related with net_device in ax25_release() can be shown as below:

```
          (USE)                   |      (FREE)
                                  |  ax25_kill_by_device()
    ax25_bind()                   |
    ax25_connect()                |    ...
                                  |  ax25_dev_device_down()
                                  |    ...
                                  |    dev_put_track(dev, ...) //FREE
    ax25_release()                |    ...
      ax25_send_control()         |
        alloc_skb()      //USE    |
```

=*=*=*=*=*=*=*=*= Bug Effects =*=*=*=*=*=*=*=*=

We can successfully trigger the vulnerabilities to crash the linux kernel.

(1) One of the use-after-free bug backtraces related with ax25_dev is shown below.

```
[  208.136725] BUG: KASAN: use-after-free in ax25_send_control+0x3c/0x210
[  208.136725] Read of size 8 at addr ffff888007c4ad08 by task ax25_co_rel/3072
[  208.136725] Call Trace:
[  208.136725]  dump_stack+0x7d/0xa3
[  208.136725]  print_address_description.constprop.0+0x18/0x130
[  208.136725]  ? ax25_send_control+0x3c/0x210
[  208.136725]  ? ax25_send_control+0x3c/0x210
[  208.136725]  kasan_report.cold+0x7f/0x10e
[  208.136725]  ? _raw_write_lock_bh+0x80/0xd0
[  208.136725]  ? ax25_send_control+0x3c/0x210
[  208.136725]  ax25_send_control+0x3c/0x210
[  208.136725]  ax25_release+0x2db/0x3b0
[  208.136725]  __sock_release+0x6d/0x120
[  208.136725]  sock_close+0xc/0x10
[  208.136725]  __fput+0x104/0x3b0
[  208.136725]  task_work_run+0x8f/0xd0
[  208.136725]  get_signal+0xbae/0xc00
[  208.136725]  ? ax25_connect+0x3c1/0x800
[  208.136725]  arch_do_signal_or_restart+0x1d9/0xc70
[  208.136725]  ? wait_woken+0x110/0x110
[  208.136725]  ? selinux_netlbl_socket_connect+0x26/0x30
[  208.136725]  ? kick_process+0x12/0x80
[  208.136725]  ? task_work_add+0xcd/0xe0
[  208.136725]  ? restore_sigcontext+0x320/0x320
[  208.136725]  ? __sys_connect+0x108/0x120
[  208.136725]  ? __sys_connect_file+0xc0/0xc0
[  208.136725]  ? common_nsleep+0x5a/0x70
[  208.136725]  ? copy_init_fpstate_to_fpregs+0x60/0x60
[  208.136725]  ? __ia32_sys_clock_adjtime+0x30/0x30
[  208.136725]  exit_to_user_mode_prepare+0xaa/0x120
[  208.136725]  syscall_exit_to_user_mode+0x1d/0x40
[  208.136725]  entry_SYSCALL_64_after_hwframe+0x44/0xa9
[  208.136725] RIP: 0033:0x7fa754c17d2b
[  208.136725] Code: 83 ec 18 89 54 24 0c 48 89 34 24 89 7c 24 08 e8 fb fa ff ff 8b 54 24 0c 48 8b 34 24 41
89 c0 8b 7c 24 08 b8 2a 00 00 00 0f 05 <48> 3d 00 f0 ff ff 77 2f 44 89 c7 89 44 24 08 e8 31 fb ff ff 8b 44
[  208.136725] RSP: 002b:00007fa5dfd26ee0 EFLAGS: 00000293 ORIG_RAX: 000000000000002a
[  208.136725] RAX: fffffffffffffe00 RBX: 0000000000000000 RCX: 00007fa754c17d2b
[  208.136725] RDX: 0000000000000010 RSI: 00000000006021c0 RDI: 0000000000000005
[  208.136725] RBP: 00007fa5dfd26f00 R08: 0000000000000000 R09: 00007fa5dfd27700
[  208.136725] R10: 0000000000000000 R11: 0000000000000293 R12: 00007ffc91618f7e
[  208.136725] R13: 00007ffc91618f7f R14: 00007fa5dfd26fc0 R15: 00007fa5dfd27700
[  208.136725]
[  208.136725] Allocated by task 3070:
[  208.136725]  kasan_save_stack+0x1b/0x40
[  208.136725]  ____kasan_kmalloc.constprop.0+0x84/0xa0
[  208.136725]  ax25_dev_device_up+0x27/0x1a0
[  208.136725]  ax25_device_event+0x12d/0x160
[  208.136725]  raw_notifier_call_chain+0x5e/0x70
[  208.136725]  __dev_notify_flags+0xbf/0x180
[  208.136725]  dev_change_flags+0x92/0xb0
[  208.136725]  devinet_ioctl+0x92f/0xbd0
[  208.136725]  inet_ioctl+0x259/0x290
[  208.136725]  sock_do_ioctl+0xa8/0x1e0
[  208.136725]  sock_ioctl+0x2ee/0x3f0
[  208.136725]  __x64_sys_ioctl+0xb4/0xf0
[  208.136725]  do_syscall_64+0x33/0x40
[  208.136725]  entry_SYSCALL_64_after_hwframe+0x44/0xa9
[  208.136725]
[  208.136725] Freed by task 3071:
[  208.136725]  kasan_save_stack+0x1b/0x40
[  208.136725]  kasan_set_track+0x1c/0x30
[  208.136725]  kasan_set_free_info+0x20/0x30
[  208.136725]  ____kasan_slab_free+0xec/0x120
[  208.136725]  kfree+0x8f/0x210
[  208.136725]  ax25_device_event+0x14e/0x160
[  208.136725]  raw_notifier_call_chain+0x5e/0x70
[  208.136725]  dev_close_many+0x17d/0x230
[  208.136725]  rollback_registered_many+0x1f1/0x950
[  208.136725]  unregister_netdevice_queue+0x133/0x200
[  208.136725]  unregister_netdev+0x13/0x20
[  208.136725]  mkiss_close+0xc4/0x120
[  208.136725]  tty_ldisc_hangup+0x1ab/0x2d0
[  208.136725]  __tty_hangup.part.0+0x306/0x510
[  208.136725]  tty_release+0x200/0x670
[  208.136725]  __fput+0x104/0x3b0
[  208.136725]  task_work_run+0x8f/0xd0
[  208.136725]  exit_to_user_mode_prepare+0x114/0x120
[  208.136725]  syscall_exit_to_user_mode+0x1d/0x40
[  208.136725]  entry_SYSCALL_64_after_hwframe+0x44/0xa9
```

(2) One of the use-after-free bug backtraces related with net_device is shown below.

```
[  769.959339] BUG: KASAN: use-after-free in ax25_send_control+0x43/0x210
[  769.959339] Read of size 2 at addr ffff8880092520de by task ax25_co_rel/1970
[  769.966904] Call Trace:
[  769.966904]  <TASK>
[  769.966904]  dump_stack_lvl+0x57/0x7d
[  769.966904]  print_address_description.constprop.0+0x1f/0x150
[  769.966904]  ? ax25_send_control+0x43/0x210
[  769.966904]  ? ax25_send_control+0x43/0x210
[  769.966904]  kasan_report.cold+0x7f/0x11b
[  769.966904]  ? ax25_send_control+0x43/0x210
[  769.966904]  ax25_send_control+0x43/0x210
[  769.966904]  ? trace_hardirqs_on+0x1c/0x110
[  769.966904]  ax25_release+0x2db/0x3b0
[  769.966904]  ? lock_release+0xb2/0x470
[  769.966904]  __sock_release+0x6d/0x120
[  769.966904]  sock_close+0xf/0x20
[  769.966904]  __fput+0x11f/0x420
[  769.966904]  task_work_run+0x86/0xd0
[  769.966904]  get_signal+0x1096/0x1240
[  769.966904]  ? lockdep_hardirqs_on_prepare+0xe/0x230
[  769.966904]  ? __local_bh_enable_ip+0x7e/0xf0
[  769.966904]  ? trace_hardirqs_on+0x1c/0x110
[  769.966904]  ? ax25_connect+0x3c1/0x800
[  769.966904]  ? signal_setup_done+0x2a0/0x2a0
[  769.966904]  arch_do_signal_or_restart+0x1df/0xbf0
[  770.012784]  exit_to_user_mode_prepare+0x143/0x1c0
[  770.012784]  syscall_exit_to_user_mode+0x19/0x50
[  770.012784]  do_syscall_64+0x48/0x90
[  770.012784]  entry_SYSCALL_64_after_hwframe+0x44/0xae
[  770.012784] RIP: 0033:0x7fba03510d2b
[  770.012784] Code: 83 ec 18 89 54 24 0c 48 89 34 24 89 7c 24 08 e8 fb fa ff ff 8b 54 24 0c 48 8b 34 24 41
89 c0 8b 7c 24 08 b8 2a 00 00 00 0f 05 <48> 3d 00 f0 ff ff 77 2f 44
[  770.016580] RSP: 002b:00007fb9a9f5aee0 EFLAGS: 00000293 ORIG_RAX: 000000000000002a
[  770.016580] RAX: fffffffffffffe00 RBX: 0000000000000000 RCX: 00007fba03510d2b
[  770.021380] RDX: 0000000000000010 RSI: 00000000006021c0 RDI: 0000000000000005
[  770.021380] RBP: 00007fb9a9f5af00 R08: 0000000000000000 R09: 00007fb9a9f5b700
[  770.021380] R10: 0000000000000000 R11: 0000000000000293 R12: 00007ffeb426b5ce
[  770.021380] R13: 00007ffeb426b5cf R14: 00007fb9a9f5afc0 R15: 00007fb9a9f5b700
[  770.021380]  </TASK>
[  770.021380]
[  770.021380] Allocated by task 1283:
[  770.025691]  kasan_save_stack+0x1e/0x40
[  770.025691]  __kasan_kmalloc+0x81/0xa0
[  770.025691]  alloc_netdev_mqs+0x5a/0x680
[  770.025691]  mkiss_open+0x6c/0x380
[  770.025691]  tty_ldisc_open+0x55/0x90
[  770.029456]  tty_set_ldisc+0x193/0x2e0
[  770.029456]  tty_ioctl+0x4ae/0xc70
[  770.029456]  __x64_sys_ioctl+0xb4/0xf0
[  770.029456]  do_syscall_64+0x3b/0x90
[  770.029456]  entry_SYSCALL_64_after_hwframe+0x44/0xae
[  770.029456]
[  770.033625] Freed by task 1969:
[  770.033625]  kasan_save_stack+0x1e/0x40
[  770.033625]  kasan_set_track+0x21/0x30
[  770.033625]  kasan_set_free_info+0x20/0x30
[  770.033625]  __kasan_slab_free+0xfa/0x130
[  770.033625]  kfree+0xa3/0x2c0
[  770.033625]  device_release+0x54/0xe0
[  770.037210]  kobject_put+0xa5/0x120
[  770.037210]  tty_ldisc_kill+0x3e/0x80
[  770.037210]  tty_ldisc_hangup+0x1b2/0x2c0
[  770.037210]  __tty_hangup.part.0+0x316/0x520
[  770.037210]  tty_release+0x200/0x670
[  770.037210]  __fput+0x11f/0x420
[  770.037210]  task_work_run+0x86/0xd0
[  770.037210]  exit_to_user_mode_prepare+0x1b2/0x1c0
[  770.037210]  syscall_exit_to_user_mode+0x19/0x50
[  770.041472]  do_syscall_64+0x48/0x90
[  770.041472]  entry_SYSCALL_64_after_hwframe+0x44/0xae
```

=*=*=*=*=*=*=*= Bug Reproduce =*=*=*=*=*=*=*=

We could use pseudoterminal-based device emulation to simulate ax25 device from user space and create a socket for it. Then, we create four threads: the first thread is used to initialize and start ax25 device, the second thread is used to close the pseudoterminal-based device, the third thread is used to execute bind and connect syscalls, the last thread is used to close the socket. Let these four threads to interleave, we could reproduce the bug.

=*=*=*=*=*=*=*=*= Bug Fix =*=*=*=*=*=*=*=*=

The patch that have been applied to mainline Linux kernel is shown below.
https://github.com/torvalds/linux/commit/d01ffb9eee4af165d83b08dd73ebdf9fe94a519b
https://github.com/torvalds/linux/commit/87563a043cef044fed5db7967a75741cc16ad2b1
https://github.com/torvalds/linux/commit/feef318c855a361a1eccd880f33e88c460eb63b4
https://github.com/torvalds/linux/commit/9fd75b66b8f68498454d685dc4ba13192ae069b0
https://github.com/torvalds/linux/commit/5352a761308397a0e6250fdc629bb3f615b94747

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

2022-01-28: commit d01ffb9eee4a accepted to mainline kernel
2022-02-04: commit 87563a043cef accepted to mainline kernel
2022-01-28: commit feef318c855a accepted to mainline kernel
2022-03-21: commit 9fd75b66b8f6 accepted to mainline kernel
2022-03-29: commit 5352a7613083 accepted to mainline kernel
2022-04-02: CVE-2022-1204 is assigned

=*=*=*=*=*=*=*=*= Credit =*=*=*=*=*=*=*=*=
Duoming Zhou <duoming@....edu.cn>

Best Regards,
Duoming Zhou

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.