openEuler

Crash工具基本使用及实战分享

目录

CONTENT

Crash工具基本使用

Crash是一个用于分析内核转储文件的分析工具

https://crash-utility.github.io/crash_whitepaper.html

启动命令：

```
crash  vmlinux  vmcore
```

vmlinux : 未压缩的内核映像文件vmlinux ,包含调试信息

vmcore：kdump生成的内核转储文件，如果此项不指定，默认分析实时系统内存

Crash工具基本使用

Crash基本输出

DTAE表示panic的时间

UPTIME是panic前已运行时间

TASKS表示发生panic时系统中进程总数

PANIC表示系统发生panic的原因及相应信息

PID是发生panic的任务号

COMMAND表示发生panic的任务正在执行的操作

TASK与THREAD_INFO表示发生panic的任务描述符及thread_info结构体的地址

CPU是发生panic的核

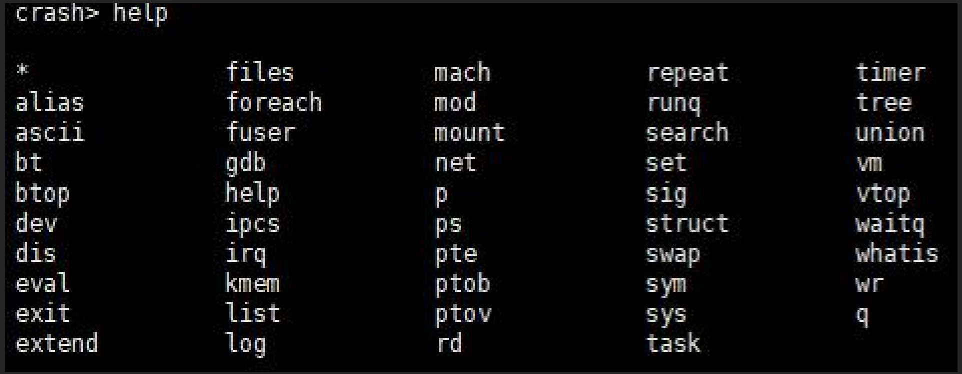
STATE表示发生panic的任务当时的状态

```
[root@localhost 127.0.0.1-2021-04-14-22:17:25]# crash  vmlinux vmcore
      KERNEL: /home/linux-rh-3-10/vmlinux
    DUMPFILE: vmcore  [PARTIAL DUMP]
        CPUS: 16
        DATE: Wed Apr 14 22:17:20 2021
      UPTIME: 12:00:53
LOAD AVERAGE: 0.11, 0.06, 0.04
       TASKS: 283
    NODENAME: localhost.localdomain
     RELEASE: 3.10.0-327.77.60.61.x86_64+
     VERSION: #65 SMP Wed Jan 20 20:46:06 EST 2021
     MACHINE: x86_64  (2593 Mhz)
      MEMORY: 8 GB
       PANIC: "BUG: unable to handle kernel NULL pointer dereference at 0000000000000804"
         PID: 3754
     COMMAND: "systemd-udevd"
        TASK: ffff880034ee8b90  [THREAD_INFO: ffff8800b90c0000]
         CPU: 3
       STATE: TASK_RUNNING (PANIC)

crash>
```

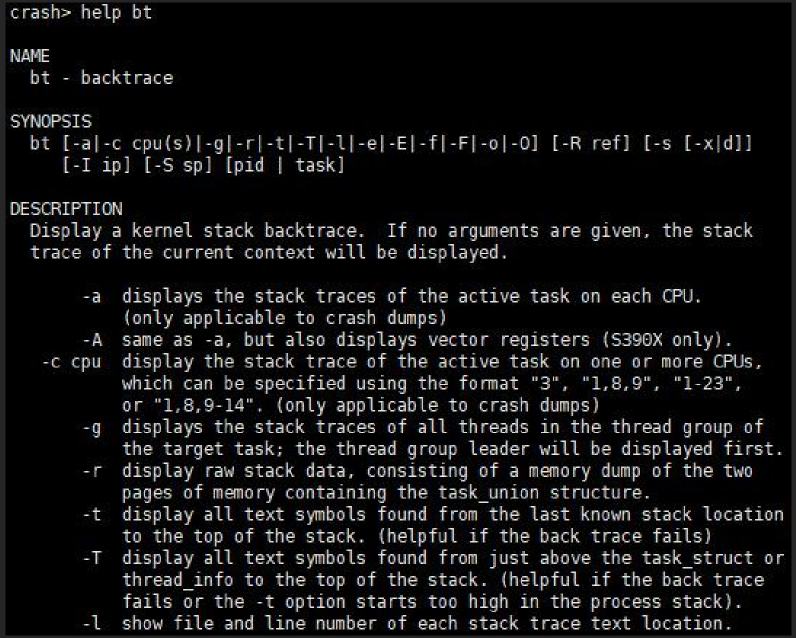
Crash工具基本使用

Crash常用命令

help 获取帮助信息，help 具体命令，如help bt, 可以查看详细使用帮助



```
crash> help

*               files           mach            repeat          timer
alias           foreach         mod             runq            tree
ascii           fuser           mount           search          union
bt              gdb             net             set             vm
btop            help            p               sig             vtop
dev             ipcs            ps              struct          waitq
dis             irq             pte             swap            whatis
eval            kmem            ptob            sym             wr
exit            list            ptov            sys             q
extend          log             rd              task
```



```
crash> help bt

NAME
  bt - backtrace

SYNOPSIS
  bt [-a|-c cpu(s)|-g|-r|-t|-T|-l|-e|-E|-f|-F|-o|-O] [-R ref] [-s [-x|d]]
     [-I ip] [-S sp] [pid | task]

DESCRIPTION
  Display a kernel stack backtrace.  If no arguments are given, the stack
  trace of the current context will be displayed.

       -a  displays the stack traces of the active task on each CPU.
           (only applicable to crash dumps)
       -A  same as -a, but also displays vector registers (S390X only).
   -c cpu  display the stack trace of the active task on one or more CPUs,
           which can be specified using the format "3", "1,8,9", "1-23",
           or "1,8,9-14". (only applicable to crash dumps)
       -g  displays the stack traces of all threads in the thread group of
           the target task; the thread group leader will be displayed first.
       -r  display raw stack data, consisting of a memory dump of the two
           pages of memory containing the task_union structure.
       -t  display all text symbols found from the last known stack location
           to the top of the stack. (helpful if the back trace fails)
       -T  display all text symbols found from just above the task_struct or
           thread_info to the top of the stack. (helpful if the back trace
           fails or the -t option starts too high in the process stack).
       -l  show file and line number of each stack trace text location.
```

Crash工具基本使用

Crash常用命令

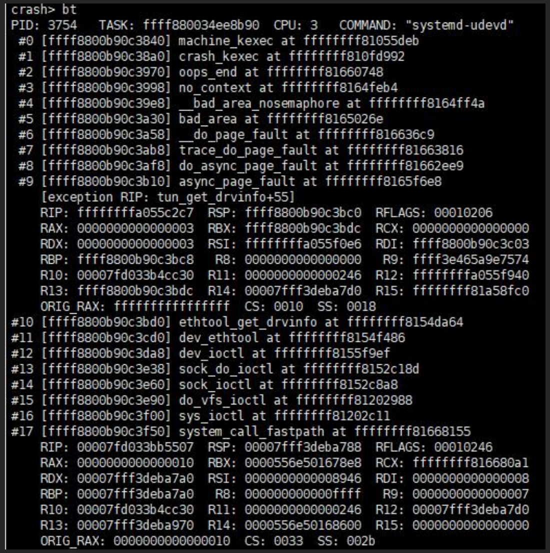
bt 显示函数调用栈， 可以显示所有CPU或指定CPU的栈，或者指定pid

bt　　显示当前cpu栈

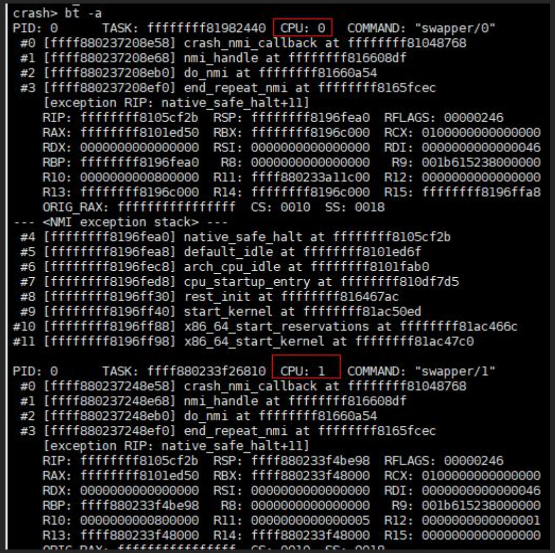
bt -a 显示所有cpu栈

bt -f 显示所有堆栈

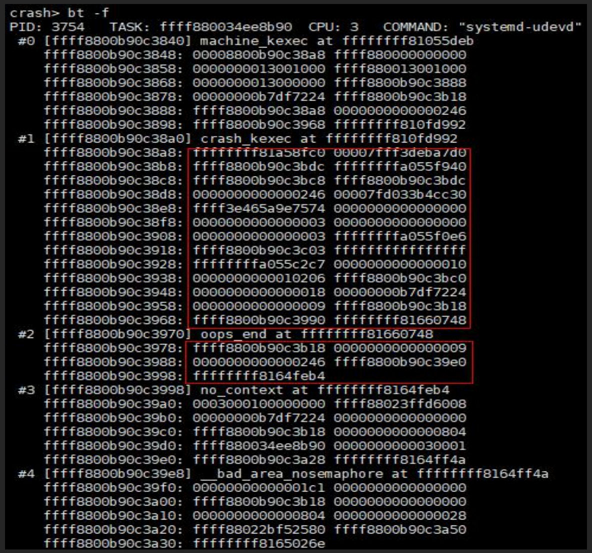
bt -l 显示堆栈tarce的文件与行号



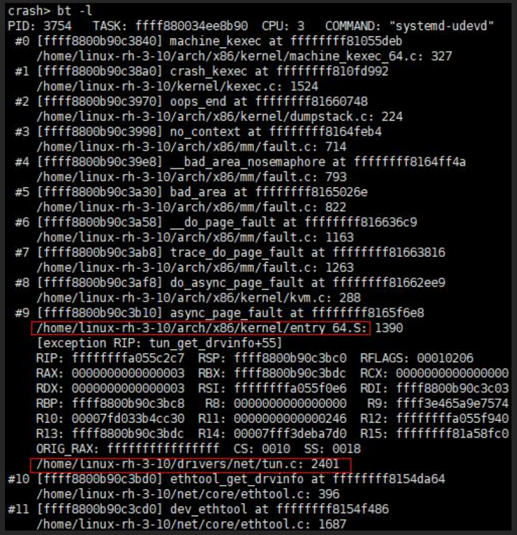
```
crash> bt
PID: 3754   TASK: ffff880034ee8b90  CPU: 3   COMMAND: "systemd-udevd"
 #0 [ffff8800b90c3840] machine_kexec at ffffffff81055deb
 #1 [ffff8800b90c38a0] crash_kexec at ffffffff810fd992
 #2 [ffff8800b90c3970] oops_end at ffffffff81660748
 #3 [ffff8800b90c3998] no_context at ffffffff8164feb4
 #4 [ffff8800b90c39e8] __bad_area_nosemaphore at ffffffff8164ff4a
 #5 [ffff8800b90c3a30] bad_area at ffffffff8165026e
 #6 [ffff8800b90c3a58] __do_page_fault at ffffffff816636c9
 #7 [ffff8800b90c3ab8] trace_do_page_fault at ffffffff81663816
 #8 [ffff8800b90c3af8] do_async_page_fault at ffffffff81662ee9
 #9 [ffff8800b90c3b10] async_page_fault at ffffffff8165f6e8
    [exception RIP: tun_get_drvinfo+55]
    RIP: ffffffffa055c2c7  RSP: ffff8800b90c3bc0  RFLAGS: 00010206
    RAX: 0000000000000003  RBX: ffff8800b90c3bdc  RCX: 0000000000000000
    RDX: 0000000000000003  RSI: ffffffffa055f0e6  RDI: ffff8800b90c3c03
    RBP: ffff8800b90c3bc8   R8: 0000000000000000   R9: ffff3e465a9e7574
    R10: 00007fd033b4cc30  R11: 0000000000000246  R12: ffffffffa055f940
    R13: ffff8800b90c3bdc  R14: 00007fff3deba7d0  R15: ffffffff81a58fc0
    ORIG_RAX: ffffffffffffffff  CS: 0010  SS: 0018
#10 [ffff8800b90c3bd0] ethtool_get_drvinfo at ffffffff8154da64
#11 [ffff8800b90c3cd0] dev_ethtool at ffffffff8154f486
#12 [ffff8800b90c3da8] dev_ioctl at ffffffff8155f9ef
#13 [ffff8800b90c3e38] sock_do_ioctl at ffffffff8152c18d
#14 [ffff8800b90c3e60] sock_ioctl at ffffffff8152c8a8
#15 [ffff8800b90c3e90] do_vfs_ioctl at ffffffff81202988
#16 [ffff8800b90c3f00] sys_ioctl at ffffffff81202c11
#17 [ffff8800b90c3f50] system_call_fastpath at ffffffff81668155
    RIP: 00007fd033bb5507  RSP: 00007fff3deba788  RFLAGS: 00010246
    RAX: 0000000000000010  RBX: 0000556e501678e8  RCX: ffffffff816680a1
    RDX: 00007fff3deba7a0  RSI: 0000000000008946  RDI: 0000000000000008
    RBP: 00007fff3deba7a0   R8: 000000000000ffff   R9: 0000000000000007
    R10: 00007fd033b4cc30  R11: 0000000000000246  R12: 00007fff3deba7d0
    R13: 00007fff3deba970  R14: 0000556e50168600  R15: 0000000000000000
    ORIG_RAX: 0000000000000010  CS: 0033  SS: 002b
```



```
crash> bt -a
PID: 0      TASK: ffffffff81982440  CPU: 0   COMMAND: "swapper/0"
 #0 [ffff880237208e58] crash_nmi_callback at ffffffff81048768
 #1 [ffff880237208e68] nmi_handle at ffffffff816608df
 #2 [ffff880237208eb0] do_nmi at ffffffff81660a54
 #3 [ffff880237208ef0] end_repeat_nmi at ffffffff8165fcec
    [exception RIP: native_safe_halt+11]
    RIP: ffffffff8105cf2b  RSP: ffffffff8196fea0  RFLAGS: 00000246
    RAX: ffffffff8101ed50  RBX: ffffffff8196c000  RCX: 0100000000000000
    RDX: 0000000000000000  RSI: 0000000000000000  RDI: 0000000000000046
    RBP: ffffffff8196fea0   R8: 0000000000000000   R9: 001b615238000000
    R10: 0000000000800000  R11: ffff880233a11c00  R12: 0000000000000000
    R13: ffffffff8196c000  R14: ffffffff8196c000  R15: ffffffff8196ffa8
    ORIG_RAX: ffffffffffffffff  CS: 0010  SS: 0018
--- <NMI exception stack> ---
 #4 [ffffffff8196fea0] native_safe_halt at ffffffff8105cf2b
 #5 [ffffffff8196fea8] default_idle at ffffffff8101ed6f
 #6 [ffffffff8196fec8] arch_cpu_idle at ffffffff8101fab0
 #7 [ffffffff8196fed8] cpu_startup_entry at ffffffff810df7d5
 #8 [ffffffff8196ff30] rest_init at ffffffff816467ac
 #9 [ffffffff8196ff40] start_kernel at ffffffff81ac50ed
#10 [ffffffff8196ff88] x86_64_start_reservations at ffffffff81ac466c
#11 [ffffffff8196ff98] x86_64_start_kernel at ffffffff81ac47c0

PID: 0      TASK: ffff880233f26810  CPU: 1   COMMAND: "swapper/1"
 #0 [ffff880237248e58] crash_nmi_callback at ffffffff81048768
 #1 [ffff880237248e68] nmi_handle at ffffffff816608df
 #2 [ffff880237248eb0] do_nmi at ffffffff81660a54
 #3 [ffff880237248ef0] end_repeat_nmi at ffffffff8165fcec
    [exception RIP: native_safe_halt+11]
    RIP: ffffffff8105cf2b  RSP: ffff880233f4be98  RFLAGS: 00000246
    RAX: ffffffff8101ed50  RBX: ffff880233f48000  RCX: 0100000000000000
    RDX: 0000000000000000  RSI: 0000000000000000  RDI: 0000000000000046
    RBP: ffff880233f4be98   R8: 0000000000000000   R9: 001b615238000000
    R10: 0000000000800000  R11: 0000000000000005  R12: 0000000000000001
    R13: ffff880233f48000  R14: ffff880233f48000  R15: 0000000000000000
    ORIG_RAX: ffffffffffffffff  CS: 0010  SS: 0018
```



```
crash> bt -f
PID: 3754   TASK: ffff880034ee8b90  CPU: 3   COMMAND: "systemd-udevd"
 #0 [ffff8800b90c3840] machine_kexec at ffffffff81055deb
    ffff8800b90c3848: 00008800b90c38a8 ffff880000000000
    ffff8800b90c3858: 0000000013001000 ffff880013001000
    ffff8800b90c3868: 0000000013000000 ffff8800b90c3888
    ffff8800b90c3878: 00000000b7df7224 ffff8800b90c3b18
    ffff8800b90c3888: ffff8800b90c38a8 0000000000000246
    ffff8800b90c3898: ffff8800b90c3968 ffffffff810fd992
 #1 [ffff8800b90c38a0] crash_kexec at ffffffff810fd992
    ffff8800b90c38a8: ffffffff81a58fc0 00007fff3deba7d0
    ffff8800b90c38b8: ffff8800b90c3bdc ffffffffa055f940
    ffff8800b90c38c8: ffff8800b90c3bc8 ffff8800b90c3bdc
    ffff8800b90c38d8: 0000000000000246 00007fd033b4cc30
    ffff8800b90c38e8: ffff3e465a9e7574 0000000000000000
    ffff8800b90c38f8: 0000000000000003 0000000000000000
    ffff8800b90c3908: 0000000000000003 ffffffffa055f0e6
    ffff8800b90c3918: ffff8800b90c3c03 ffffffffffffffff
    ffff8800b90c3928: ffffffffa055c2c7 0000000000000010
    ffff8800b90c3938: 0000000000010206 ffff8800b90c3bc0
    ffff8800b90c3948: 0000000000000018 00000000b7df7224
    ffff8800b90c3958: 0000000000000009 ffff8800b90c3b18
    ffff8800b90c3968: ffff8800b90c3990 ffffffff81660748
 #2 [ffff8800b90c3970] oops_end at ffffffff81660748
    ffff8800b90c3978: ffff8800b90c3b18 0000000000000009
    ffff8800b90c3988: 0000000000000246 ffff8800b90c39e0
    ffff8800b90c3998: ffffffff8164feb4
 #3 [ffff8800b90c3998] no_context at ffffffff8164feb4
    ffff8800b90c39a0: 0003000100000000 ffff88023ffd6008
    ffff8800b90c39b0: 00000000b7df7224 0000000000000000
    ffff8800b90c39c0: ffff8800b90c3b18 0000000000000804
    ffff8800b90c39d0: ffff880034ee8b90 0000000000030001
    ffff8800b90c39e0: ffff8800b90c3a28 ffffffff8164ff4a
 #4 [ffff8800b90c39e8] __bad_area_nosemaphore at ffffffff8164ff4a
    ffff8800b90c39f0: 00000000000001c1 0000000000000000
    ffff8800b90c3a00: ffff8800b90c3b18 0000000000000000
    ffff8800b90c3a10: 0000000000000804 0000000000000028
    ffff8800b90c3a20: ffff88022bf52580 ffff8800b90c3a50
    ffff8800b90c3a30: ffffffff8165026e
```



```
crash> bt -l
PID: 3754   TASK: ffff880034ee8b90  CPU: 3   COMMAND: "systemd-udevd"
 #0 [ffff8800b90c3840] machine_kexec at ffffffff81055deb
    /home/linux-rh-3-10/arch/x86/kernel/machine_kexec_64.c: 327
 #1 [ffff8800b90c38a0] crash_kexec at ffffffff810fd992
    /home/linux-rh-3-10/kernel/kexec.c: 1524
 #2 [ffff8800b90c3970] oops_end at ffffffff81660748
    /home/linux-rh-3-10/arch/x86/kernel/dumpstack.c: 224
 #3 [ffff8800b90c3998] no_context at ffffffff8164feb4
    /home/linux-rh-3-10/arch/x86/mm/fault.c: 714
 #4 [ffff8800b90c39e8] __bad_area_nosemaphore at ffffffff8164ff4a
    /home/linux-rh-3-10/arch/x86/mm/fault.c: 793
 #5 [ffff8800b90c3a30] bad_area at ffffffff8165026e
    /home/linux-rh-3-10/arch/x86/mm/fault.c: 822
 #6 [ffff8800b90c3a58] __do_page_fault at ffffffff816636c9
    /home/linux-rh-3-10/arch/x86/mm/fault.c: 1163
 #7 [ffff8800b90c3ab8] trace_do_page_fault at ffffffff81663816
    /home/linux-rh-3-10/arch/x86/mm/fault.c: 1263
 #8 [ffff8800b90c3af8] do_async_page_fault at ffffffff81662ee9
    /home/linux-rh-3-10/arch/x86/kernel/kvm.c: 288
 #9 [ffff8800b90c3b10] async_page_fault at ffffffff8165f6e8
    /home/linux-rh-3-10/arch/x86/kernel/entry_64.S: 1390
    [exception RIP: tun_get_drvinfo+55]
    RIP: ffffffffa055c2c7  RSP: ffff8800b90c3bc0  RFLAGS: 00010206
    RAX: 0000000000000003  RBX: ffff8800b90c3bdc  RCX: 0000000000000000
    RDX: 0000000000000003  RSI: ffffffffa055f0e6  RDI: ffff8800b90c3c03
    RBP: ffff8800b90c3bc8   R8: 0000000000000000   R9: ffff3e465a9e7574
    R10: 00007fd033b4cc30  R11: 0000000000000246  R12: ffffffffa055f940
    R13: ffff8800b90c3bdc  R14: 00007fff3deba7d0  R15: ffffffff81a58fc0
    ORIG_RAX: ffffffffffffffff  CS: 0010  SS: 0018
    /home/linux-rh-3-10/drivers/net/tun.c: 2401
#10 [ffff8800b90c3bd0] ethtool_get_drvinfo at ffffffff8154da64
    /home/linux-rh-3-10/net/core/ethtool.c: 396
#11 [ffff8800b90c3cd0] dev_ethtool at ffffffff8154f486
    /home/linux-rh-3-10/net/core/ethtool.c: 1687
```

Crash工具基本使用

Crash常用命令

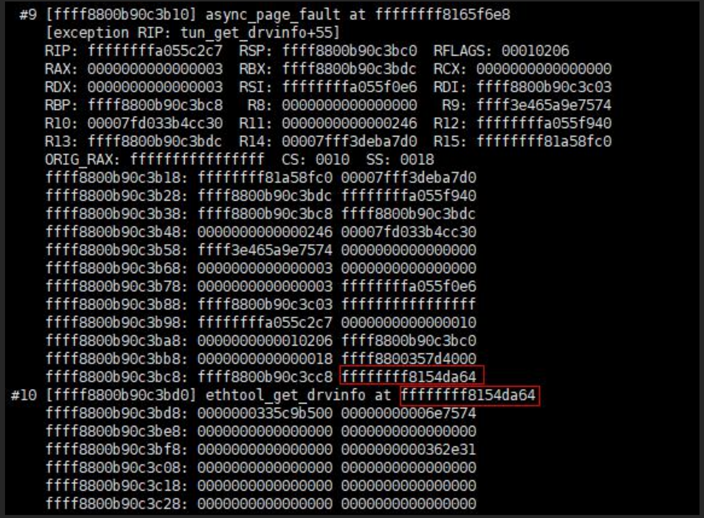
#0 代表callTrace层级

exception RIP 异常指令

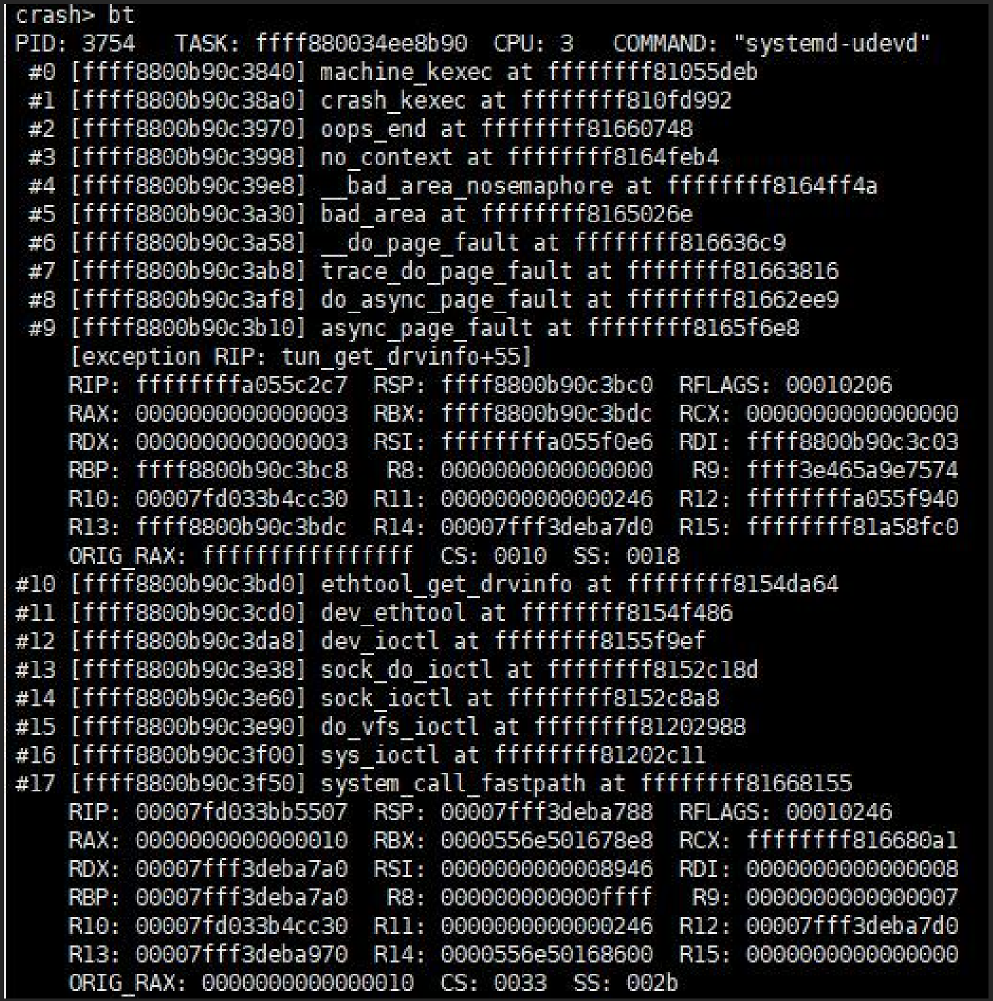
RAX RBX RCX .....   寄存器值

#10 [ffff8800b90c3bd0] ethtool_get_drvinfo at ffffffff8154da64

系统在使用线性地址为ffff8800b90c3bd0处的空间时，在ethtool_get_drvinfo函数中运行，从ffffffff8154da64处开始调用下一层函数tun_get_drvinfo



```
 #9 [ffff8800b90c3b10] async_page_fault at ffffffff8165f6e8
    [exception RIP: tun_get_drvinfo+55]
    RIP: ffffffffa055c2c7  RSP: ffff8800b90c3bc0  RFLAGS: 00010206
    RAX: 0000000000000003  RBX: ffff8800b90c3bdc  RCX: 0000000000000000
    RDX: 0000000000000003  RSI: ffffffffa055f0e6  RDI: ffff8800b90c3c03
    RBP: ffff8800b90c3bc8   R8: 0000000000000000   R9: ffff3e465a9e7574
    R10: 00007fd033b4cc30  R11: 0000000000000246  R12: ffffffffa055f940
    R13: ffff8800b90c3bdc  R14: 00007fff3deba7d0  R15: ffffffff81a58fc0
    ORIG_RAX: ffffffffffffffff  CS: 0010  SS: 0018
    ffff8800b90c3b18: ffffffff81a58fc0 00007fff3deba7d0
    ffff8800b90c3b28: ffff8800b90c3bdc ffffffffa055f940
    ffff8800b90c3b38: ffff8800b90c3bc8 ffff8800b90c3bdc
    ffff8800b90c3b48: 0000000000000246 00007fd033b4cc30
    ffff8800b90c3b58: ffff3e465a9e7574 0000000000000000
    ffff8800b90c3b68: 0000000000000003 0000000000000000
    ffff8800b90c3b78: 0000000000000003 ffffffffa055f0e6
    ffff8800b90c3b88: ffff8800b90c3c03 ffffffffffffffff
    ffff8800b90c3b98: ffffffffa055c2c7 0000000000000010
    ffff8800b90c3ba8: 0000000000010206 ffff8800b90c3bc0
    ffff8800b90c3bb8: 0000000000000018 ffff8800357d4000
    ffff8800b90c3bc8: ffff8800b90c3cc8 ffffffff8154da64
#10 [ffff8800b90c3bd0] ethtool_get_drvinfo at ffffffff8154da64
    ffff8800b90c3bd8: 0000000335c9b500 000000000006e7574
    ffff8800b90c3be8: 0000000000000000 0000000000000000
    ffff8800b90c3bf8: 0000000000000000 0000000000362e31
    ffff8800b90c3c08: 0000000000000000 0000000000000000
    ffff8800b90c3c18: 0000000000000000 0000000000000000
    ffff8800b90c3c28: 0000000000000000 0000000000000000
```



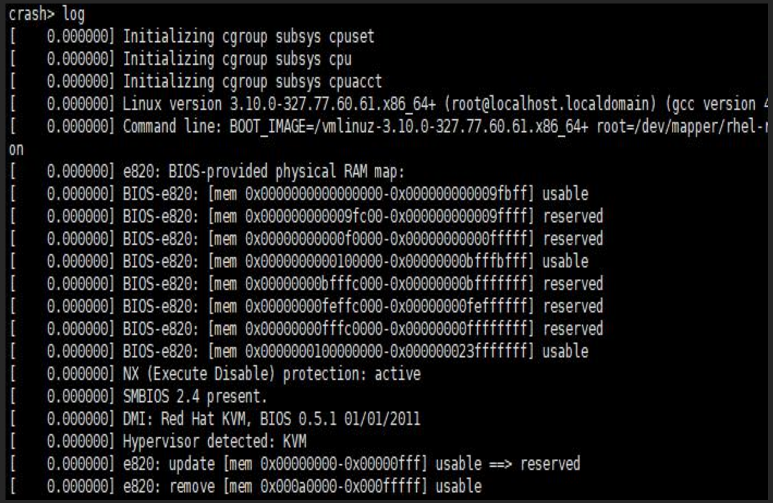
```
crash> bt
PID: 3754   TASK: ffff880034ee8b90  CPU: 3   COMMAND: "systemd-udevd"
 #0 [ffff8800b90c3840] machine_kexec at ffffffff81055deb
 #1 [ffff8800b90c38a0] crash_kexec at ffffffff810fd992
 #2 [ffff8800b90c3970] oops_end at ffffffff81660748
 #3 [ffff8800b90c3998] no_context at ffffffff8164feb4
 #4 [ffff8800b90c39e8] __bad_area_nosemaphore at ffffffff8164ff4a
 #5 [ffff8800b90c3a30] bad_area at ffffffff8165026e
 #6 [ffff8800b90c3a58] __do_page_fault at ffffffff816636c9
 #7 [ffff8800b90c3ab8] trace_do_page_fault at ffffffff81663816
 #8 [ffff8800b90c3af8] do_async_page_fault at ffffffff81662ee9
 #9 [ffff8800b90c3b10] async_page_fault at ffffffff8165f6e8
    [exception RIP: tun_get_drvinfo+55]
    RIP: ffffffffa055c2c7  RSP: ffff8800b90c3bc0  RFLAGS: 00010206
    RAX: 0000000000000003  RBX: ffff8800b90c3bdc  RCX: 0000000000000000
    RDX: 0000000000000003  RSI: ffffffffa055f0e6  RDI: ffff8800b90c3c03
    RBP: ffff8800b90c3bc8   R8: 0000000000000000   R9: ffff3e465a9e7574
    R10: 00007fd033b4cc30  R11: 0000000000000246  R12: ffffffffa055f940
    R13: ffff8800b90c3bdc  R14: 00007fff3deba7d0  R15: ffffffff81a58fc0
    ORIG_RAX: ffffffffffffffff  CS: 0010  SS: 0018
#10 [ffff8800b90c3bd0] ethtool_get_drvinfo at ffffffff8154da64
#11 [ffff8800b90c3cd0] dev_ethtool at ffffffff8154f486
#12 [ffff8800b90c3da8] dev_ioctl at ffffffff8155f9ef
#13 [ffff8800b90c3e38] sock_do_ioctl at ffffffff8152c18d
#14 [ffff8800b90c3e60] sock_ioctl at ffffffff8152c8a8
#15 [ffff8800b90c3e90] do_vfs_ioctl at ffffffff81202988
#16 [ffff8800b90c3f00] sys_ioctl at ffffffff81202c11
#17 [ffff8800b90c3f50] system_call_fastpath at ffffffff81668155
    RIP: 00007fd033bb5507  RSP: 00007fff3deba788  RFLAGS: 00010246
    RAX: 0000000000000010  RBX: 0000556e501678e8  RCX: ffffffff816680a1
    RDX: 00007fff3deba7a0  RSI: 0000000000008946  RDI: 0000000000000008
    RBP: 00007fff3deba7a0   R8: 000000000000ffff   R9: 0000000000000007
    R10: 00007fd033b4cc30  R11: 0000000000000246  R12: 00007fff3deba7d0
    R13: 00007fff3deba970  R14: 0000556e50168600  R15: 0000000000000000
    ORIG_RAX: 0000000000000010  CS: 0033  SS: 002b
```

openEuler

Crash工具基本使用

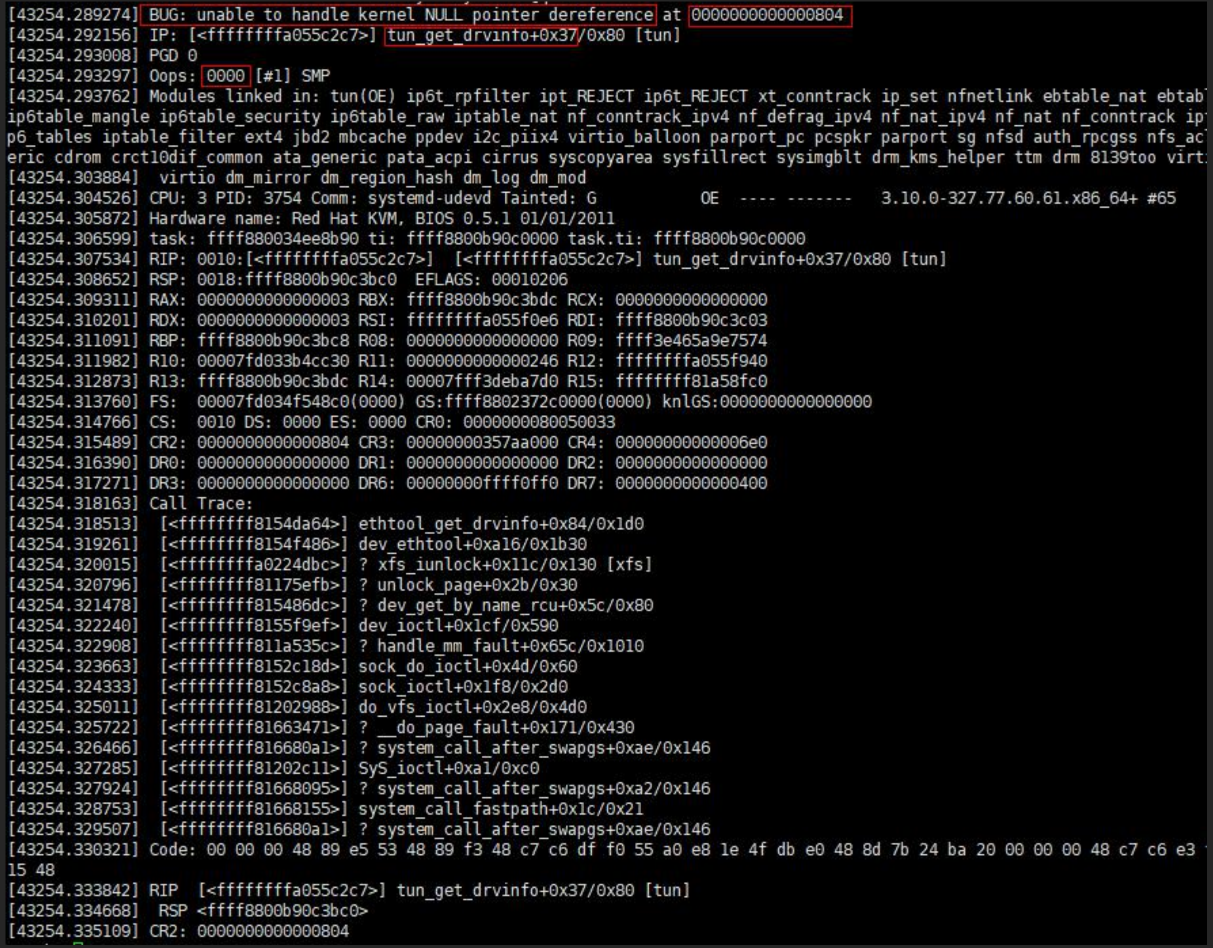
Crash常用命令

log 显示系统消息缓存区



```
crash> log
[    0.000000] Initializing cgroup subsys cpuset
[    0.000000] Initializing cgroup subsys cpu
[    0.000000] Initializing cgroup subsys cpuacct
[    0.000000] Linux version 3.10.0-327.77.60.61.x86_64+ (root@localhost.localdomain) (gcc version 4
[    0.000000] Command line: BOOT_IMAGE=/vmlinuz-3.10.0-327.77.60.61.x86_64+ root=/dev/mapper/rhel-r
on
[    0.000000] e820: BIOS-provided physical RAM map:
[    0.000000] BIOS-e820: [mem 0x0000000000000000-0x000000000009fbff] usable
[    0.000000] BIOS-e820: [mem 0x000000000009fc00-0x000000000009ffff] reserved
[    0.000000] BIOS-e820: [mem 0x00000000000f0000-0x00000000000fffff] reserved
[    0.000000] BIOS-e820: [mem 0x0000000000100000-0x00000000bfffbfff] usable
[    0.000000] BIOS-e820: [mem 0x00000000bfffc000-0x00000000bfffffff] reserved
[    0.000000] BIOS-e820: [mem 0x00000000feffc000-0x00000000feffffff] reserved
[    0.000000] BIOS-e820: [mem 0x00000000fffc0000-0x00000000ffffffff] reserved
[    0.000000] BIOS-e820: [mem 0x0000000100000000-0x000000023fffffff] usable
[    0.000000] NX (Execute Disable) protection: active
[    0.000000] SMBIOS 2.4 present.
[    0.000000] DMI: Red Hat KVM, BIOS 0.5.1 01/01/2011
[    0.000000] Hypervisor detected: KVM
[    0.000000] e820: update [mem 0x00000000-0x00000fff] usable ==> reserved
[    0.000000] e820: remove [mem 0x000a0000-0x000fffff] usable
```

	Value	
Bit	0	1
0	No page found	Invalid access
1	Read or Execute	Write
2	Kernel mode	User mode
3	Not instruction fetch	Instruction fetch

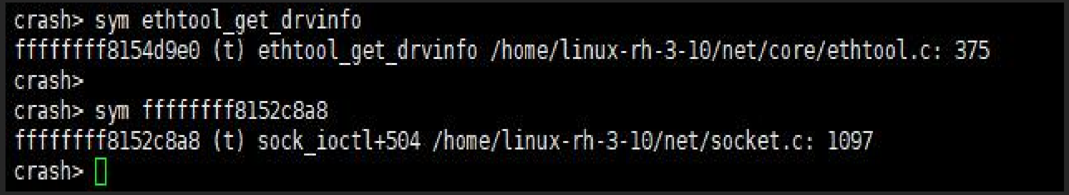


```
[43254.289274] BUG: unable to handle kernel NULL pointer dereference at 0000000000000804
[43254.292156] IP: [<ffffffffa055c2c7>] tun_get_drvinfo+0x37/0x80 [tun]
[43254.293008] PGD 0
[43254.293297] Oops: 0000 [#1] SMP
[43254.293762] Modules linked in: tun(OE) ip6t_rpfilter ipt_REJECT ip6t_REJECT xt_conntrack ip_set nfnetlink ebtable_nat ebtab
ip6table_mangle ip6table_security ip6table_raw iptable_nat nf_conntrack_ipv4 nf_defrag_ipv4 nf_nat_ipv4 nf_nat nf_conntrack ip
p6_tables iptable_filter ext4 jbd2 mbcache ppdev i2c_piix4 virtio_balloon parport_pc pcspkr parport sg nfsd auth_rpcgss nfs_ac
eric cdrom crct10dif_common ata_generic pata_acpi cirrus syscopyarea sysfillrect sysimgblt drm_kms_helper ttm drm 8139too virt
[43254.303884]  virtio dm_mirror dm_region_hash dm_log dm_mod
[43254.304526] CPU: 3 PID: 3754 Comm: systemd-udevd Tainted: G           OE  ---- -------   3.10.0-327.77.60.61.x86_64+ #65
[43254.305872] Hardware name: Red Hat KVM, BIOS 0.5.1 01/01/2011
[43254.306599] task: ffff880034ee8b90 ti: ffff8800b90c0000 task.ti: ffff8800b90c0000
[43254.307534] RIP: 0010:[<ffffffffa055c2c7>]  [<ffffffffa055c2c7>] tun_get_drvinfo+0x37/0x80 [tun]
[43254.308652] RSP: 0018:ffff8800b90c3bc0  EFLAGS: 00010206
[43254.309311] RAX: 0000000000000003 RBX: ffff8800b90c3bdc RCX: 0000000000000000
[43254.310201] RDX: 0000000000000003 RSI: ffffffffa055f0e6 RDI: ffff8800b90c3c03
[43254.311091] RBP: ffff8800b90c3bc8 R08: 0000000000000000 R09: ffff3e465a9e7574
[43254.311982] R10: 00007fd033b4cc30 R11: 0000000000000246 R12: ffffffffa055f940
[43254.312873] R13: ffff8800b90c3bdc R14: 00007fff3deba7d0 R15: ffffffff81a58fc0
[43254.313760] FS:  00007fd034f548c0(0000) GS:ffff8802372c0000(0000) knlGS:0000000000000000
[43254.314766] CS:  0010 DS: 0000 ES: 0000 CR0: 0000000080050033
[43254.315489] CR2: 0000000000000804 CR3: 00000000357aa000 CR4: 00000000000006e0
[43254.316390] DR0: 0000000000000000 DR1: 0000000000000000 DR2: 0000000000000000
[43254.317271] DR3: 0000000000000000 DR6: 00000000ffff0ff0 DR7: 0000000000000400
[43254.318163] Call Trace:
[43254.318513]  [<ffffffff8154da64>] ethtool_get_drvinfo+0x84/0x1d0
[43254.319261]  [<ffffffff8154f486>] dev_ethtool+0xa16/0x1b30
[43254.320015]  [<ffffffffa0224dbc>] ? xfs_iunlock+0x11c/0x130 [xfs]
[43254.320796]  [<ffffffff81175efb>] ? unlock_page+0x2b/0x30
[43254.321478]  [<ffffffff815486dc>] ? dev_get_by_name_rcu+0x5c/0x80
[43254.322240]  [<ffffffff8155f9ef>] dev_ioctl+0x1cf/0x590
[43254.322908]  [<ffffffff811a535c>] ? handle_mm_fault+0x65c/0x1010
[43254.323663]  [<ffffffff8152c18d>] sock_do_ioctl+0x4d/0x60
[43254.324333]  [<ffffffff8152c8a8>] sock_ioctl+0x1f8/0x2d0
[43254.325011]  [<ffffffff81202988>] do_vfs_ioctl+0x2e8/0x4d0
[43254.325722]  [<ffffffff81663471>] ? __do_page_fault+0x171/0x430
[43254.326466]  [<ffffffff816680a1>] ? system_call_after_swapgs+0xae/0x146
[43254.327285]  [<ffffffff81202c11>] SyS_ioctl+0xa1/0xc0
[43254.327924]  [<ffffffff81668095>] ? system_call_after_swapgs+0xa2/0x146
[43254.328753]  [<ffffffff81668155>] system_call_fastpath+0x1c/0x21
[43254.329507]  [<ffffffff816680a1>] ? system_call_after_swapgs+0xae/0x146
[43254.330321] Code: 00 00 00 48 89 e5 53 48 89 f3 48 c7 c6 df f0 55 a0 e8 1e 4f db e0 48 8d 7b 24 ba 20 00 00 00 48 c7 c6 e3
15 48
[43254.333842] RIP  [<ffffffffa055c2c7>] tun_get_drvinfo+0x37/0x80 [tun]
[43254.334668]  RSP <ffff8800b90c3bc0>
[43254.335109] CR2: 0000000000000804
```

Crash工具基本使用

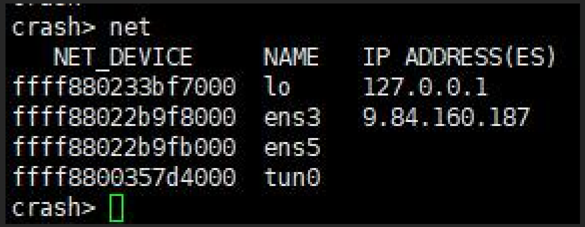
Crash常用命令

sym symbol与虚拟地址转换



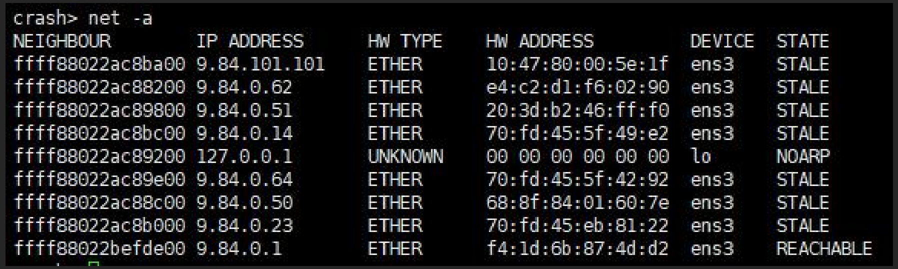
```
crash> sym ethtool_get_drvinfo
ffffffff8154d9e0 (t) ethtool_get_drvinfo /home/linux-rh-3-10/net/core/ethtool.c: 375
crash>
crash> sym ffffffff8152c8a8
ffffffff8152c8a8 (t) sock_ioctl+504 /home/linux-rh-3-10/net/socket.c: 1097
crash> 
```

net 列出网络设备



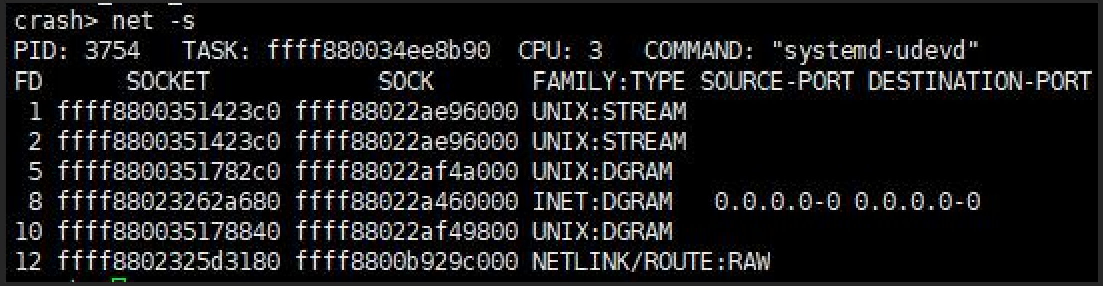
```
crash> net
   NET_DEVICE     NAME   IP ADDRESS(ES)
ffff880233bf7000  lo     127.0.0.1
ffff88022b9f8000  ens3   9.84.160.187
ffff88022b9fb000  ens5
ffff8800357d4000  tun0
crash> 
```

net 显示ARP cache



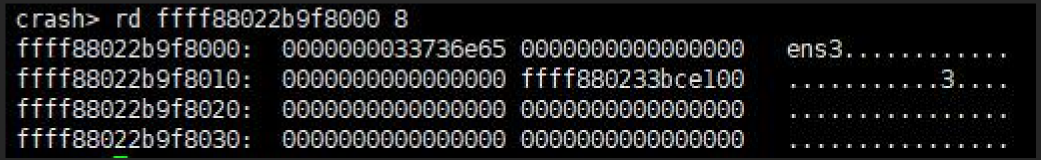
```
crash> net -a
NEIGHBOUR        IP ADDRESS      HW TYPE    HW ADDRESS         DEVICE  STATE
ffff88022ac8ba00 9.84.101.101    ETHER      10:47:80:00:5e:1f  ens3    STALE
ffff88022ac88200 9.84.0.62       ETHER      e4:c2:d1:f6:02:90  ens3    STALE
ffff88022ac89800 9.84.0.51       ETHER      20:3d:b2:46:ff:f0  ens3    STALE
ffff88022ac8bc00 9.84.0.14       ETHER      70:fd:45:5f:49:e2  ens3    STALE
ffff88022ac89200 127.0.0.1       UNKNOWN    00 00 00 00 00 00  lo      NOARP
ffff88022ac89e00 9.84.0.64       ETHER      70:fd:45:5f:42:92  ens3    STALE
ffff88022ac88c00 9.84.0.50       ETHER      68:8f:84:01:60:7e  ens3    STALE
ffff88022ac8b000 9.84.0.23       ETHER      70:fd:45:eb:81:22  ens3    STALE
ffff88022befde00 9.84.0.1        ETHER      f4:1d:6b:87:4d:d2  ens3    REACHABLE
```

net 列出所有sock



```
crash> net -s
PID: 3754   TASK: ffff880034ee8b90  CPU: 3   COMMAND: "systemd-udevd"
FD      SOCKET              SOCK        FAMILY:TYPE SOURCE-PORT DESTINATION-PORT
 1 ffff8800351423c0 ffff88022ae96000 UNIX:STREAM
 2 ffff8800351423c0 ffff88022ae96000 UNIX:STREAM
 5 ffff8800351782c0 ffff88022af4a000 UNIX:DGRAM
 8 ffff88023262a680 ffff88022a460000 INET:DGRAM   0.0.0.0-0 0.0.0.0-0
10 ffff880035178840 ffff88022af49800 UNIX:DGRAM
12 ffff8802325d3180 ffff8800b929c000 NETLINK/ROUTE:RAW
```

rd 直接读内存

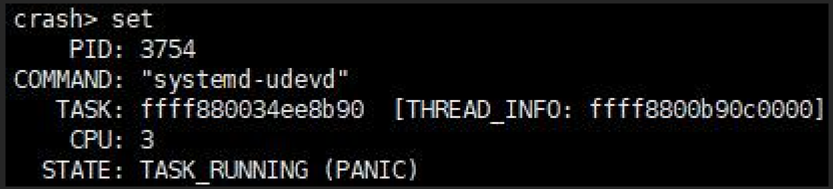


```
crash> rd ffff88022b9f8000 8
ffff88022b9f8000:  0000000033736e65 0000000000000000   ens3............
ffff88022b9f8010:  0000000000000000 ffff880233bce100   ...........3....
ffff88022b9f8020:  0000000000000000 0000000000000000   ................
ffff88022b9f8030:  0000000000000000 0000000000000000   ................
```

Crash工具基本使用

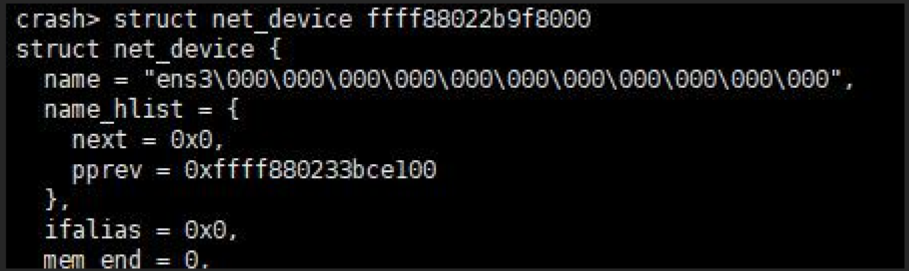
Crash常用命令

set 获取crash的线程号

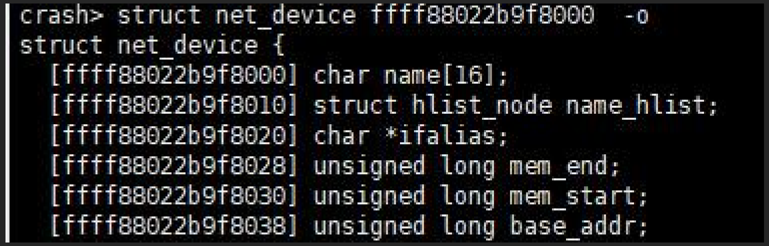


```
crash> set
    PID: 3754
COMMAND: "systemd-udevd"
   TASK: ffff880034ee8b90  [THREAD_INFO: ffff8800b90c0000]
    CPU: 3
  STATE: TASK_RUNNING (PANIC)
```

struct 解析结构体

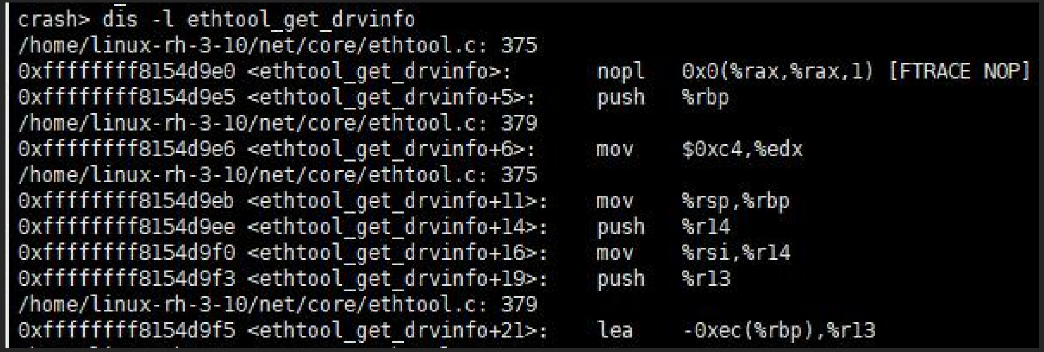


```
crash> struct net_device ffff88022b9f8000
struct net_device {
  name = "ens3\000\000\000\000\000\000\000\000\000\000\000\000",
  name_hlist = {
    next = 0x0,
    pprev = 0xffff880233bce100
  },
  ifalias = 0x0,
  mem_end = 0,
```



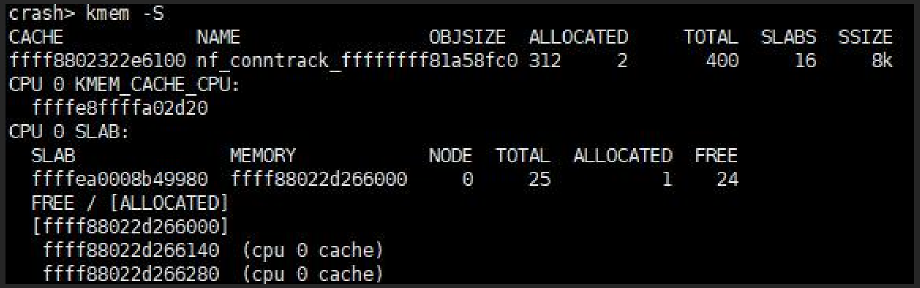
```
crash> struct net_device ffff88022b9f8000  -o
struct net_device {
  [ffff88022b9f8000] char name[16];
  [ffff88022b9f8010] struct hlist_node name_hlist;
  [ffff88022b9f8020] char *ifalias;
  [ffff88022b9f8028] unsigned long mem_end;
  [ffff88022b9f8030] unsigned long mem_start;
  [ffff88022b9f8038] unsigned long base_addr;
```

dis 反汇编地址



```
crash> dis -l ethtool_get_drvinfo
/home/linux-rh-3-10/net/core/ethtool.c: 375
0xffffffff8154d9e0 <ethtool_get_drvinfo>:       nopl   0x0(%rax,%rax,1) [FTRACE NOP]
0xffffffff8154d9e5 <ethtool_get_drvinfo+5>:     push   %rbp
/home/linux-rh-3-10/net/core/ethtool.c: 379
0xffffffff8154d9e6 <ethtool_get_drvinfo+6>:     mov    $0xc4,%edx
/home/linux-rh-3-10/net/core/ethtool.c: 375
0xffffffff8154d9eb <ethtool_get_drvinfo+11>:    mov    %rsp,%rbp
0xffffffff8154d9ee <ethtool_get_drvinfo+14>:    push   %r14
0xffffffff8154d9f0 <ethtool_get_drvinfo+16>:    mov    %rsi,%r14
0xffffffff8154d9f3 <ethtool_get_drvinfo+19>:    push   %r13
/home/linux-rh-3-10/net/core/ethtool.c: 379
0xffffffff8154d9f5 <ethtool_get_drvinfo+21>:    lea    -0xec(%rbp),%r13
```

kmem -S 显示slab对象信息

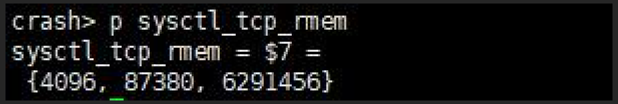


```
crash> kmem -S
CACHE            NAME                 OBJSIZE  ALLOCATED     TOTAL  SLABS  SSIZE
ffff8802322e6100 nf_conntrack_ffffffff81a58fc0 312      2       400     16     8k
CPU 0 KMEM_CACHE_CPU:
  ffffe8ffffa02d20
CPU 0 SLAB:
  SLAB              MEMORY            NODE  TOTAL  ALLOCATED  FREE
  ffffea0008b49980  ffff88022d266000     0     25          1    24
  FREE / [ALLOCATED]
  [ffff88022d266000]
   ffff88022d266140  (cpu 0 cache)
   ffff88022d266280  (cpu 0 cache)
```

Crash工具基本使用

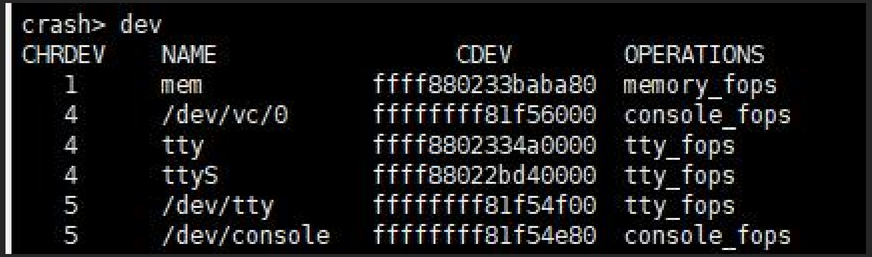
Crash常用命令

p 查看全局变量



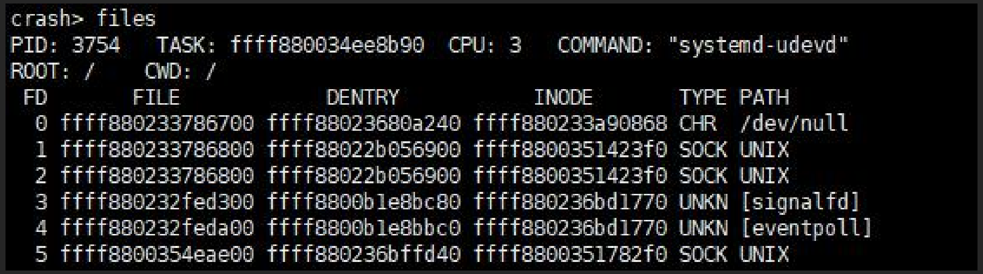
```
crash> p sysctl_tcp_rmem
sysctl_tcp_rmem = $7 =
 {4096, 87380, 6291456}
```

dev命令可以显示系统中块设备和字符设备的信息



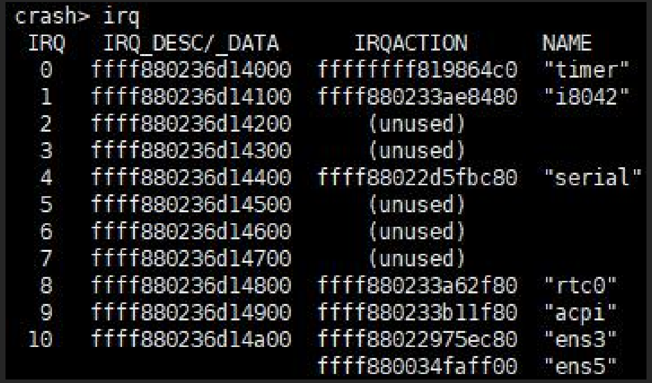
```
crash> dev
CHRDEV    NAME              CDEV           OPERATIONS
   1      mem           ffff880233baba80  memory_fops
   4      /dev/vc/0     ffffffff81f56000  console_fops
   4      tty           ffff8802334a0000  tty_fops
   4      ttyS          ffff88022bd40000  tty_fops
   5      /dev/tty      ffffffff81f54f00  tty_fops
   5      /dev/console  ffffffff81f54e80  console_fops
```

files命令可以显示发生panic的任务所打开的所有文件的信息



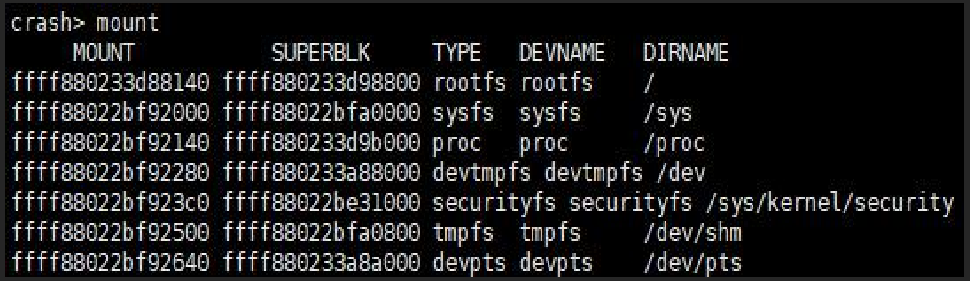
```
crash> files
PID: 3754   TASK: ffff880034ee8b90  CPU: 3   COMMAND: "systemd-udevd"
ROOT: /    CWD: /
 FD       FILE            DENTRY           INODE       TYPE PATH
  0 ffff880233786700 ffff88023680a240 ffff880233a90868 CHR  /dev/null
  1 ffff880233786800 ffff88022b056900 ffff8800351423f0 SOCK UNIX
  2 ffff880233786800 ffff88022b056900 ffff8800351423f0 SOCK UNIX
  3 ffff880232fed300 ffff8800b1e8bc80 ffff880236bd1770 UNKN [signalfd]
  4 ffff880232feda00 ffff8800b1e8bbc0 ffff880236bd1770 UNKN [eventpoll]
  5 ffff8800354eae00 ffff880236bffd40 ffff8800351782f0 SOCK UNIX
```

Irq命令可以显示相应中断信息



```
crash> irq
 IRQ   IRQ_DESC/_DATA      IRQACTION      NAME
  0   ffff880236d14000  ffffffff819864c0  "timer"
  1   ffff880236d14100  ffff880233ae8480  "i8042"
  2   ffff880236d14200      (unused)
  3   ffff880236d14300      (unused)
  4   ffff880236d14400  ffff88022d5fbc80  "serial"
  5   ffff880236d14500      (unused)
  6   ffff880236d14600      (unused)
  7   ffff880236d14700      (unused)
  8   ffff880236d14800  ffff880233a62f80  "rtc0"
  9   ffff880236d14900  ffff880233b11f80  "acpi"
 10   ffff880236d14a00  ffff88022975ec80  "ens3"
                        ffff880034faff00  "ens5"
```

mount 用于显示挂载的文件系统信息



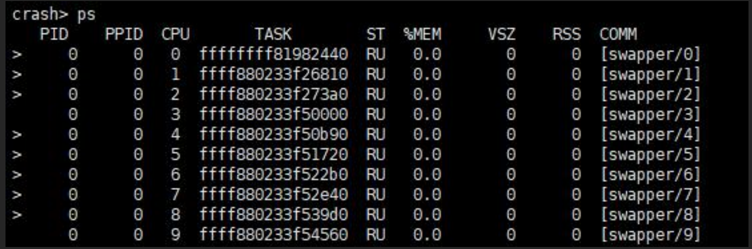
```
crash> mount
     MOUNT           SUPERBLK      TYPE   DEVNAME   DIRNAME
ffff880233d88140 ffff880233d98800 rootfs rootfs    /
ffff88022bf92000 ffff88022bfa0000 sysfs  sysfs     /sys
ffff88022bf92140 ffff880233d9b000 proc   proc      /proc
ffff88022bf92280 ffff880233a88000 devtmpfs devtmpfs /dev
ffff88022bf923c0 ffff88022be31000 securityfs securityfs /sys/kernel/security
ffff88022bf92500 ffff88022bfa0800 tmpfs  tmpfs     /dev/shm
ffff88022bf92640 ffff880233a8a000 devpts devpts    /dev/pts
```

openEuler

Crash工具基本使用

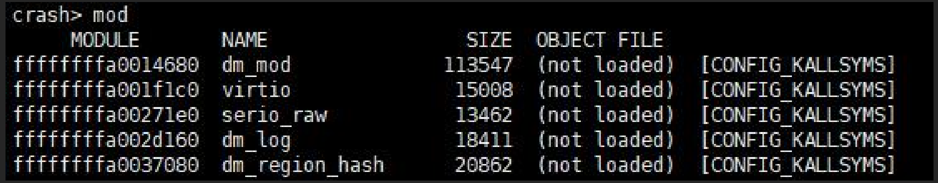
Crash常用命令

ps 查看系统中进程的信息

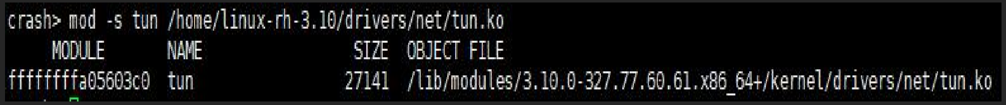


```
crash> ps
   PID    PPID  CPU       TASK        ST  %MEM     VSZ    RSS  COMM
>     0      0   0  ffffffff81982440  RU   0.0       0      0  [swapper/0]
>     0      0   1  ffff880233f26810  RU   0.0       0      0  [swapper/1]
>     0      0   2  ffff880233f273a0  RU   0.0       0      0  [swapper/2]
      0      0   3  ffff880233f50000  RU   0.0       0      0  [swapper/3]
>     0      0   4  ffff880233f50b90  RU   0.0       0      0  [swapper/4]
>     0      0   5  ffff880233f51720  RU   0.0       0      0  [swapper/5]
>     0      0   6  ffff880233f522b0  RU   0.0       0      0  [swapper/6]
>     0      0   7  ffff880233f52e40  RU   0.0       0      0  [swapper/7]
>     0      0   8  ffff880233f539d0  RU   0.0       0      0  [swapper/8]
      0      0   9  ffff880233f54560  RU   0.0       0      0  [swapper/9]
```

mod命令可以显示或者加载内核模块

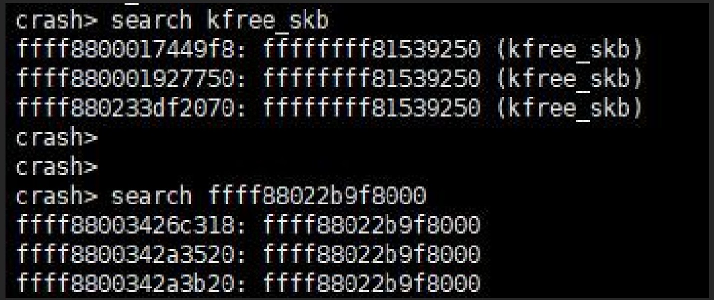


```
crash> mod
     MODULE       NAME                   SIZE  OBJECT FILE
ffffffffa0014680  dm_mod               113547  (not loaded)  [CONFIG_KALLSYMS]
ffffffffa001f1c0  virtio                15008  (not loaded)  [CONFIG_KALLSYMS]
ffffffffa00271e0  serio_raw             13462  (not loaded)  [CONFIG_KALLSYMS]
ffffffffa002d160  dm_log                18411  (not loaded)  [CONFIG_KALLSYMS]
ffffffffa0037080  dm_region_hash        20862  (not loaded)  [CONFIG_KALLSYMS]
```



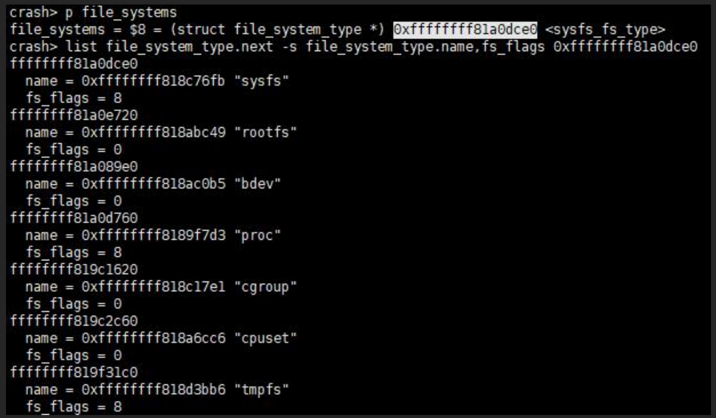
```
crash> mod -s tun /home/linux-rh-3.10/drivers/net/tun.ko
     MODULE       NAME                   SIZE  OBJECT FILE
ffffffffa05603c0  tun                   27141  /lib/modules/3.10.0-327.77.60.61.x86_64+/kernel/drivers/net/tun.ko
```

search命令可以在内存中寻找所有存放目标数据的地址



```
crash> search kfree_skb
ffff8800017449f8: ffffffff81539250 (kfree_skb)
ffff880001927750: ffffffff81539250 (kfree_skb)
ffff880233df2070: ffffffff81539250 (kfree_skb)
crash>
crash>
crash> search ffff88022b9f8000
ffff88003426c318: ffff88022b9f8000
ffff8800342a3520: ffff88022b9f8000
ffff8800342a3b20: ffff88022b9f8000
```

list命令用于显示链表的内容



```
crash> p file_systems
file_systems = $8 = (struct file_system_type *) 0xffffffff81a0dce0 <sysfs_fs_type>
crash> list file_system_type.next -s file_system_type.name,fs_flags 0xffffffff81a0dce0
ffffffff81a0dce0
  name = 0xffffffff818c76fb "sysfs"
  fs_flags = 8
ffffffff81a0e720
  name = 0xffffffff818abc49 "rootfs"
  fs_flags = 0
ffffffff81a089e0
  name = 0xffffffff818ac0b5 "bdev"
  fs_flags = 0
ffffffff81a0d760
  name = 0xffffffff8189f7d3 "proc"
  fs_flags = 8
ffffffff819c1620
  name = 0xffffffff818c17e1 "cgroup"
  fs_flags = 0
ffffffff819c2c60
  name = 0xffffffff818a6cc6 "cpuset"
  fs_flags = 0
ffffffff819f31c0
  name = 0xffffffff818d3bb6 "tmpfs"
  fs_flags = 8
```

一个简单例子

查看panic核上的栈

```
crash> bt
PID: 3754   TASK: ffff880034ee8b90  CPU: 3   COMMAND: "systemd-udevd"
 #0 [ffff8800b90c3840] machine_kexec at ffffffff81055deb
 #1 [ffff8800b90c38a0] crash_kexec at ffffffff810fd992
 #2 [ffff8800b90c3970] oops_end at ffffffff81660748
 #3 [ffff8800b90c3998] no_context at ffffffff8164feb4
 #4 [ffff8800b90c39e8] __bad_area_nosemaphore at ffffffff8164ff4a
 #5 [ffff8800b90c3a30] bad_area at ffffffff8165026e
 #6 [ffff8800b90c3a58] __do_page_fault at ffffffff816636c9
 #7 [ffff8800b90c3ab8] trace_do_page_fault at ffffffff81663816
 #8 [ffff8800b90c3af8] do_async_page_fault at ffffffff81662ee9
 #9 [ffff8800b90c3b10] async_page_fault at ffffffff8165f6e8
    [exception RIP: tun_get_drvinfo+55]
    RIP: ffffffffa055c2c7  RSP: ffff8800b90c3bc0  RFLAGS: 00010206
    RAX: 0000000000000003  RBX: ffff8800b90c3bdc  RCX: 0000000000000000
    RDX: 0000000000000003  RSI: ffffffffa055f0e6  RDI: ffff8800b90c3c03
    RBP: ffff8800b90c3bc8   R8: 0000000000000000   R9: ffff3e465a9e7574
    R10: 00007fd033b4cc30  R11: 0000000000000246  R12: ffffffffa055f940
    R13: ffff8800b90c3bdc  R14: 00007fff3deba7d0  R15: ffffffff81a58fc0
    ORIG_RAX: ffffffffffffffff  CS: 0010  SS: 0018
#10 [ffff8800b90c3bd0] ethtool_get_drvinfo at ffffffff8154da64
#11 [ffff8800b90c3cd0] dev_ethtool at ffffffff8154f486
#12 [ffff8800b90c3da8] dev_ioctl at ffffffff8155f9ef
#13 [ffff8800b90c3e38] sock_do_ioctl at ffffffff8152c18d
#14 [ffff8800b90c3e60] sock_ioctl at ffffffff8152c8a8
#15 [ffff8800b90c3e90] do_vfs_ioctl at ffffffff81202988
#16 [ffff8800b90c3f00] sys_ioctl at ffffffff81202c11
#17 [ffff8800b90c3f50] system_call_fastpath at ffffffff81668155
    RIP: 00007fd033bb5507  RSP: 00007fff3deba788  RFLAGS: 00010246
    RAX: 0000000000000010  RBX: 0000556e501678e8  RCX: ffffffff816680a1
    RDX: 00007fff3deba7a0  RSI: 0000000000008946  RDI: 0000000000000008
    RBP: 00007fff3deba7a0   R8: 000000000000ffff   R9: 0000000000000007
    R10: 00007fd033b4cc30  R11: 0000000000000246  R12: 00007fff3deba7d0
    R13: 00007fff3deba970  R14: 0000556e50168600  R15: 0000000000000000
    ORIG_RAX: 0000000000000010  CS: 0033  SS: 002b
```

```
crash> log
[43254.289274] BUG: unable to handle kernel NULL pointer dereference at 0000000000000804
[43254.292156] IP: [<ffffffffa055c2c7>] tun_get_drvinfo+0x37/0x80 [tun]
[43254.293008] PGD 0
[43254.293297] Oops: 0000 [#1] SMP
[43254.293762] Modules linked in: tun(OE) ip6t_rpfilter ipt_REJECT ip6t_REJECT [43254.304526] CPU: 3 PID: 3754 Comm: systemd-udevd Tainted: G
OE  ---- -------   3.10.0-327.77.60.61.x86_64+ #65
[43254.305872] Hardware name: Red Hat KVM, BIOS 0.5.1 01/01/2011
[43254.306599] task: ffff880034ee8b90 ti: ffff8800b90c0000 task.ti: ffff8800b90c0000
[43254.307534] RIP: 0010:[<ffffffffa055c2c7>]  [<ffffffffa055c2c7>] tun_get_drvinfo+0x37/0x80 [tun]
[43254.308652] RSP: 0018:ffff8800b90c3bc0  EFLAGS: 00010206
[43254.309311] RAX: 0000000000000003 RBX: ffff8800b90c3bdc RCX: 0000000000000000
[43254.310201] RDX: 0000000000000003 RSI: ffffffffa055f0e6 RDI: ffff8800b90c3c03
[43254.311091] RBP: ffff8800b90c3bc8 R08: 0000000000000000 R09: ffff3e465a9e7574
[43254.311982] R10: 00007fd033b4cc30 R11: 0000000000000246 R12: ffffffffa055f940
[43254.312873] R13: ffff8800b90c3bdc R14: 00007fff3deba7d0 R15: ffffffff81a58fc0
[43254.313760] FS:  00007fd034f548c0(0000) GS:ffff8802372c0000(0000) knlGS:0000000000000000
[43254.314766] CS:  0010 DS: 0000 ES: 0000 CR0: 0000000080050033
[43254.315489] CR2: 0000000000000804 CR3: 00000000357aa000 CR4: 00000000000006e0
[43254.316390] DR0: 0000000000000000 DR1: 0000000000000000 DR2: 0000000000000000
[43254.317271] DR3: 0000000000000000 DR6: 00000000ffff0ff0 DR7: 0000000000000400
[43254.318163] Call Trace:
[43254.318513]  [<ffffffff8154da64>] ethtool_get_drvinfo+0x84/0x1d0
[43254.319261]  [<ffffffff8154f486>] dev_ethtool+0xa16/0x1b30
[43254.320015]  [<ffffffffa0224dbc>] ? xfs_iunlock+0x11c/0x130 [xfs]
[43254.320796]  [<ffffffff81175efb>] ? unlock_page+0x2b/0x30
[43254.321478]  [<ffffffff815486dc>] ? dev_get_by_name_rcu+0x5c/0x80
[43254.322240]  [<ffffffff8155f9ef>] dev_ioctl+0x1cf/0x590
[43254.322908]  [<ffffffff811a535c>] ? handle_mm_fault+0x65c/0x1010
[43254.323663]  [<ffffffff8152c18d>] sock_do_ioctl+0x4d/0x60
[43254.324333]  [<ffffffff8152c8a8>] sock_ioctl+0x1f8/0x2d0
[43254.325011]  [<ffffffff81202988>] do_vfs_ioctl+0x2e8/0x4d0
[43254.325722]  [<ffffffff81663471>] ? __do_page_fault+0x171/0x430
[43254.326466]  [<ffffffff816680a1>] ? system_call_after_swapgs+0xae/0x146
[43254.327285]  [<ffffffff81202c11>] SyS_ioctl+0xa1/0xc0
[43254.327924]  [<ffffffff81668095>] ? system_call_after_swapgs+0xa2/0x146
[43254.328753]  [<ffffffff81668155>] system_call_fastpath+0x1c/0x21
[43254.329507]  [<ffffffff816680a1>] ? system_call_after_swapgs+0xae/0x146
```

一个简单例子

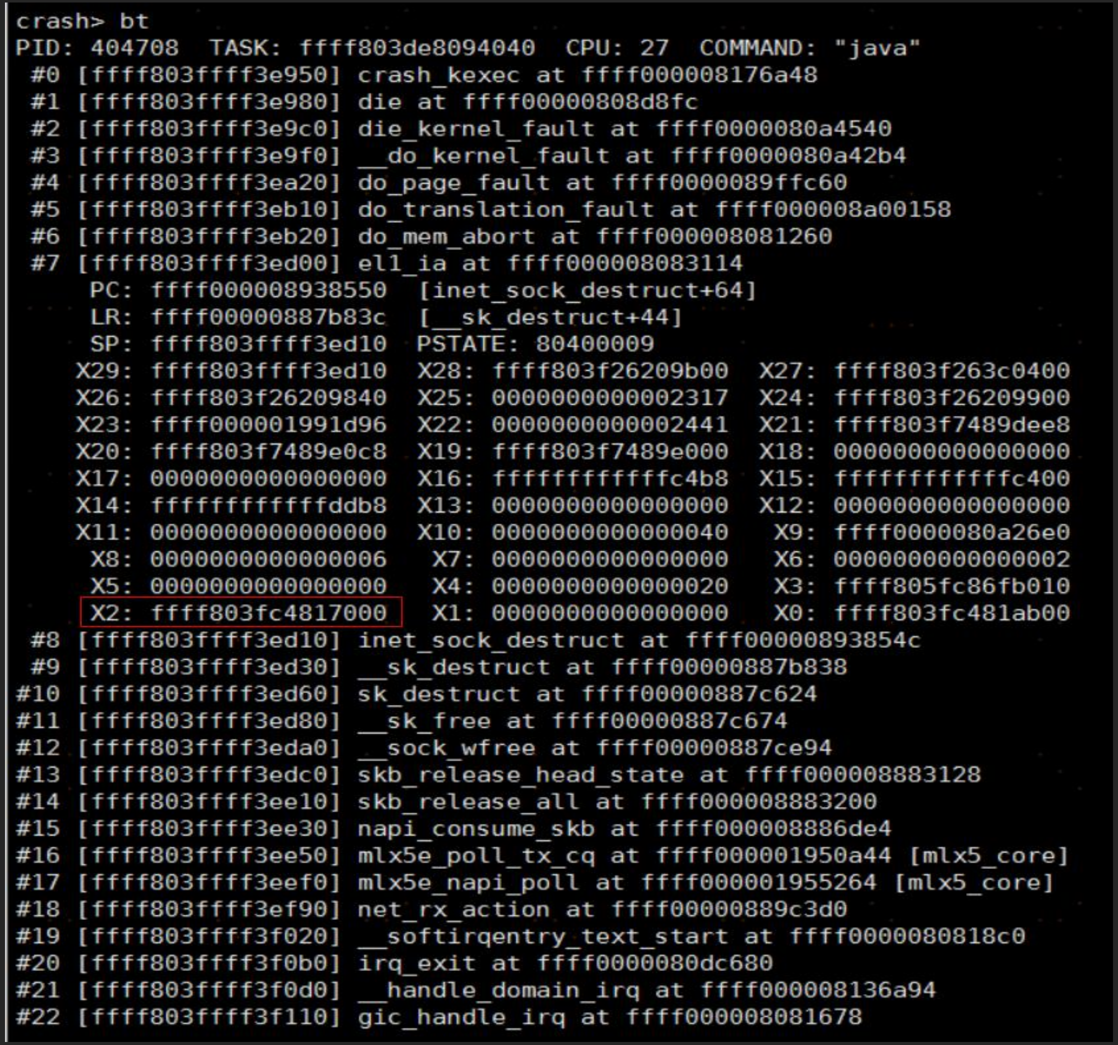
查看panic核上的栈

```
crash> mod  -s  tun  /home/linux-rh-3.10/drivers/net/tun.ko  //加载模块信息
    MODULE       NAME                  SIZE  OBJECT FILE
ffffffffa05603c0  tun                27141  /lib/modules/3.10.0-
327.77.60.61.x86_64+/kernel/drivers/net/tun.ko
crash>
crash> dis -l tun_get_drvinfo+0x37 4
/home/linux-rh-3-10/drivers/net/tun.c: 2401
0xffffffffa055c2c7 <tun_get_drvinfo+55>:       mov    0x804,%eax  //tun->flags
/home/linux-rh-3-10/drivers/net/tun.c: 2403
0xffffffffa055c2ce <tun_get_drvinfo+62>:       and    $0xf,%eax
0xffffffffa055c2d1 <tun_get_drvinfo+65>:       cmp    $0x1,%eax
0xffffffffa055c2d4 <tun_get_drvinfo+68>:       je     0xffffffffa055c2f8
<tun_get_drvinfo+104>

2396 static void tun_get_drvinfo(struct net_device *dev, struct ethtool_drvinfo *info)
2397 {
2398        struct tun_struct *tun = netdev_priv(dev);
2399
2400        strlcpy(info->driver, DRV_NAME, sizeof(info->driver));
2401        strlcpy(info->version, DRV_VERSION, sizeof(info->version));
2402
2403        tun = NULL;
2404        switch (tun->flags & TUN_TYPE_MASK) {
2405        case IFF_TUN:
2406                strlcpy(info->bus_info, "tun", sizeof(info->bus_info));
2407                break;

crash> struct tun_struct -ox
struct tun_struct {
    [0x0] struct tun_file *tfiles[256];
  [0x800] unsigned int numqueues;
  [0x804] unsigned int flags;
  [0x808] kuid_t owner;
  [0x80c] kgid_t group;
  [0x810] struct net_device *dev;
```

```
crash> log
[43254.289274] BUG: unable to handle kernel NULL pointer dereference at 0000000000000804
[43254.292156] IP: [<ffffffffa055c2c7>] tun_get_drvinfo+0x37/0x80 [tun]
[43254.293008] PGD 0
[43254.293297] Oops: 0000 [#1] SMP
[43254.293762] Modules linked in: tun(OE) ip6t_rpfilter ipt_REJECT ip6t_REJECT [43254.304526] CPU: 3 PID: 3754 Comm: systemd-udevd Tainted: G
OE  ---- -------   3.10.0-327.77.60.61.x86_64+ #65
[43254.305872] Hardware name: Red Hat KVM, BIOS 0.5.1 01/01/2011
[43254.306599] task: ffff880034ee8b90 ti: ffff8800b90c0000 task.ti: ffff8800b90c0000
[43254.307534] RIP: 0010:[<ffffffffa055c2c7>]  [<ffffffffa055c2c7>] tun_get_drvinfo+0x37/0x80 [tun]
[43254.308652] RSP: 0018:ffff8800b90c3bc0  EFLAGS: 00010206
[43254.309311] RAX: 0000000000000003 RBX: ffff8800b90c3bdc RCX: 0000000000000000
[43254.310201] RDX: 0000000000000003 RSI: ffffffffa055f0e6 RDI: ffff8800b90c3c03
[43254.311091] RBP: ffff8800b90c3bc8 R08: 0000000000000000 R09: ffff3e465a9e7574
[43254.311982] R10: 00007fd033b4cc30 R11: 0000000000000246 R12: ffffffffa055f940
[43254.312873] R13: ffff8800b90c3bdc R14: 00007fff3deba7d0 R15: ffffffff81a58fc0
[43254.313760] FS:  00007fd034f548c0(0000) GS:ffff8802372c0000(0000) knlGS:0000000000000000
[43254.314766] CS:  0010 DS: 0000 ES: 0000 CR0: 0000000080050033
[43254.315489] CR2: 0000000000000804 CR3: 00000000357aa000 CR4: 00000000000006e0
[43254.316390] DR0: 0000000000000000 DR1: 0000000000000000 DR2: 0000000000000000
[43254.317271] DR3: 0000000000000000 DR6: 00000000ffff0ff0 DR7: 0000000000000400
[43254.318163] Call Trace:
[43254.318513]  [<ffffffff8154da64>] ethtool_get_drvinfo+0x84/0x1d0
[43254.319261]  [<ffffffff8154f486>] dev_ethtool+0xa16/0x1b30
[43254.320015]  [<ffffffffa0224dbc>] ? xfs_iunlock+0x11c/0x130 [xfs]
[43254.320796]  [<ffffffff81175efb>] ? unlock_page+0x2b/0x30
[43254.321478]  [<ffffffff815486dc>] ? dev_get_by_name_rcu+0x5c/0x80
[43254.322240]  [<ffffffff8155f9ef>] dev_ioctl+0x1cf/0x590
[43254.322908]  [<ffffffff811a535c>] ? handle_mm_fault+0x65c/0x1010
[43254.323663]  [<ffffffff8152c18d>] sock_do_ioctl+0x4d/0x60
[43254.324333]  [<ffffffff8152c8a8>] sock_ioctl+0x1f8/0x2d0
[43254.325011]  [<ffffffff81202988>] do_vfs_ioctl+0x2e8/0x4d0
[43254.325722]  [<ffffffff81663471>] ? __do_page_fault+0x171/0x430
[43254.326466]  [<ffffffff816680a1>] ? system_call_after_swapgs+0xae/0x146
[43254.327285]  [<ffffffff81202c11>] SyS_ioctl+0xa1/0xc0
[43254.327924]  [<ffffffff81668095>] ? system_call_after_swapgs+0xa2/0x146
[43254.328753]  [<ffffffff81668155>] system_call_fastpath+0x1c/0x21
[43254.329507]  [<ffffffff816680a1>] ? system_call_after_swapgs+0xae/0x146
```

现网问题实战

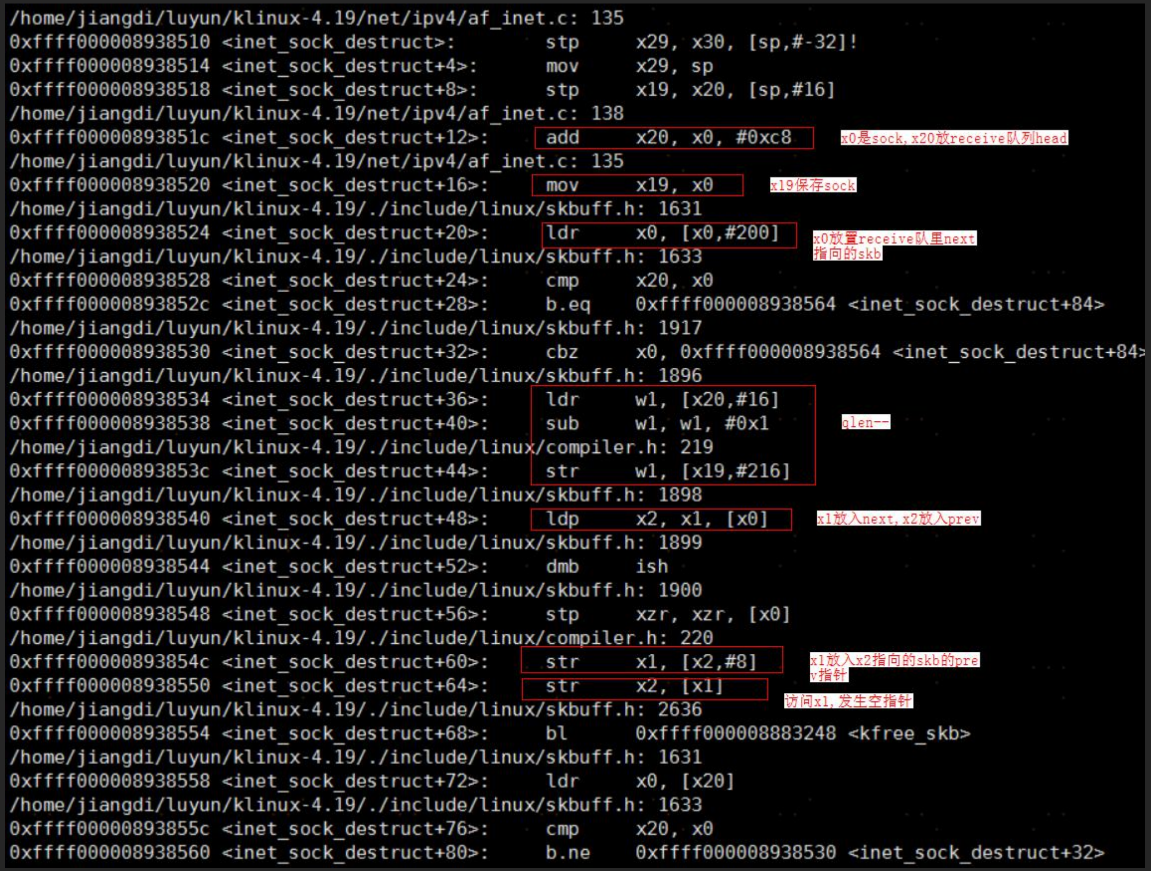
查看panic核上的栈，然后反汇编找出问题点



```
crash> bt
PID: 404708  TASK: ffff803de8094040  CPU: 27  COMMAND: "java"
 #0 [ffff803ffff3e950] crash_kexec at ffff000008176a48
 #1 [ffff803ffff3e980] die at ffff00000808d8fc
 #2 [ffff803ffff3e9c0] die_kernel_fault at ffff0000080a4540
 #3 [ffff803ffff3e9f0] __do_kernel_fault at ffff0000080a42b4
 #4 [ffff803ffff3ea20] do_page_fault at ffff0000089ffc60
 #5 [ffff803ffff3eb10] do_translation_fault at ffff000008a00158
 #6 [ffff803ffff3eb20] do_mem_abort at ffff000008081260
 #7 [ffff803ffff3ed00] el1_ia at ffff000008083114
     PC: ffff000008938550  [inet_sock_destruct+64]
     LR: ffff00000887b83c  [__sk_destruct+44]
     SP: ffff803ffff3ed10  PSTATE: 80400009
    X29: ffff803ffff3ed10  X28: ffff803f26209b00  X27: ffff803f263c0400
    X26: ffff803f26209840  X25: 0000000000002317  X24: ffff803f26209900
    X23: ffff000001991d96  X22: 0000000000002441  X21: ffff803f7489dee8
    X20: ffff803f7489e0c8  X19: ffff803f7489e000  X18: 0000000000000000
    X17: 0000000000000000  X16: ffffffffffffc4b8  X15: ffffffffffffc400
    X14: ffffffffffffddb8  X13: 0000000000000000  X12: 0000000000000000
    X11: 0000000000000000  X10: 0000000000000040   X9: ffff0000080a26e0
     X8: 0000000000000006   X7: 0000000000000000   X6: 0000000000000002
     X5: 0000000000000000   X4: 0000000000000020   X3: ffff805fc86fb010
     X2: ffff803fc4817000   X1: 0000000000000000   X0: ffff803fc481ab00
 #8 [ffff803ffff3ed10] inet_sock_destruct at ffff00000893854c
 #9 [ffff803ffff3ed30] __sk_destruct at ffff00000887b838
#10 [ffff803ffff3ed60] sk_destruct at ffff00000887c624
#11 [ffff803ffff3ed80] __sk_free at ffff00000887c674
#12 [ffff803ffff3eda0] __sock_wfree at ffff00000887ce94
#13 [ffff803ffff3edc0] skb_release_head_state at ffff000008883128
#14 [ffff803ffff3ee10] skb_release_all at ffff000008883200
#15 [ffff803ffff3ee30] napi_consume_skb at ffff000008886de4
#16 [ffff803ffff3ee50] mlx5e_poll_tx_cq at ffff000001950a44 [mlx5_core]
#17 [ffff803ffff3eef0] mlx5e_napi_poll at ffff000001955264 [mlx5_core]
#18 [ffff803ffff3ef90] net_rx_action at ffff0000889c3d0
#19 [ffff803ffff3f020] __softirqentry_text_start at ffff0000080818c0
#20 [ffff803ffff3f0b0] irq_exit at ffff0000080dc680
#21 [ffff803ffff3f0d0] __handle_domain_irq at ffff000008136a94
#22 [ffff803ffff3f110] gic_handle_irq at ffff000008081678
```

现网问题实战

反汇编出问题的函数inet_sock_destruct，分析代码流程

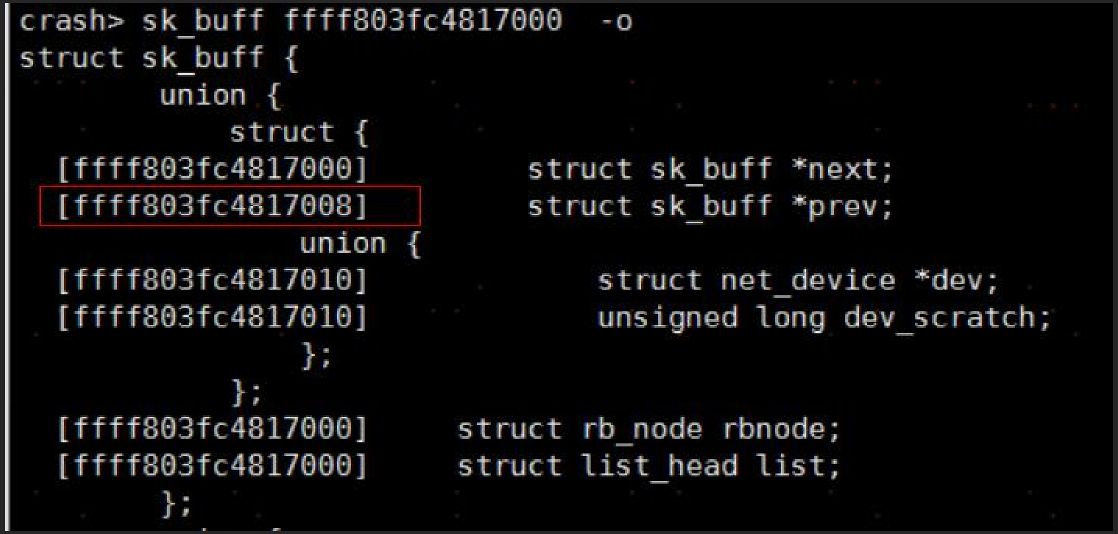


```
/home/jiangdi/luyun/klinux-4.19/net/ipv4/af_inet.c: 135
0xffff000008938510 <inet_sock_destruct>:       stp     x29, x30, [sp,#-32]!
0xffff000008938514 <inet_sock_destruct+4>:     mov     x29, sp
0xffff000008938518 <inet_sock_destruct+8>:     stp     x19, x20, [sp,#16]
/home/jiangdi/luyun/klinux-4.19/net/ipv4/af_inet.c: 138
0xffff00000893851c <inet_sock_destruct+12>:    add     x20, x0, #0xc8      x0是sock,x20放receive队列head
/home/jiangdi/luyun/klinux-4.19/net/ipv4/af_inet.c: 135
0xffff000008938520 <inet_sock_destruct+16>:    mov     x19, x0      x19保存sock
/home/jiangdi/luyun/klinux-4.19/./include/linux/skbuff.h: 1631
0xffff000008938524 <inet_sock_destruct+20>:    ldr     x0, [x0,#200]     x0放置receive队里next
/home/jiangdi/luyun/klinux-4.19/./include/linux/skbuff.h: 1633                  指向的skb
0xffff000008938528 <inet_sock_destruct+24>:    cmp     x20, x0
0xffff00000893852c <inet_sock_destruct+28>:    b.eq    0xffff000008938564 <inet_sock_destruct+84>
/home/jiangdi/luyun/klinux-4.19/./include/linux/skbuff.h: 1917
0xffff000008938530 <inet_sock_destruct+32>:    cbz     x0, 0xffff000008938564 <inet_sock_destruct+84>
/home/jiangdi/luyun/klinux-4.19/./include/linux/skbuff.h: 1896
0xffff000008938534 <inet_sock_destruct+36>:    ldr     w1, [x20,#16]
0xffff000008938538 <inet_sock_destruct+40>:    sub     w1, w1, #0x1      qlen--
/home/jiangdi/luyun/klinux-4.19/./include/linux/compiler.h: 219
0xffff00000893853c <inet_sock_destruct+44>:    str     w1, [x19,#216]
/home/jiangdi/luyun/klinux-4.19/./include/linux/skbuff.h: 1898
0xffff000008938540 <inet_sock_destruct+48>:    ldp     x2, x1, [x0]      x1放入next,x2放入prev
/home/jiangdi/luyun/klinux-4.19/./include/linux/skbuff.h: 1899
0xffff000008938544 <inet_sock_destruct+52>:    dmb     ish
/home/jiangdi/luyun/klinux-4.19/./include/linux/skbuff.h: 1900
0xffff000008938548 <inet_sock_destruct+56>:    stp     xzr, xzr, [x0]
/home/jiangdi/luyun/klinux-4.19/./include/linux/compiler.h: 220
0xffff00000893854c <inet_sock_destruct+60>:    str     x1, [x2,#8]      x1放入x2指向的skb的pre
0xffff000008938550 <inet_sock_destruct+64>:    str     x2, [x1]         v指针
/home/jiangdi/luyun/klinux-4.19/./include/linux/skbuff.h: 2636     访问x1,发生空指针
0xffff000008938554 <inet_sock_destruct+68>:    bl      0xffff000008883248 <kfree_skb>
/home/jiangdi/luyun/klinux-4.19/./include/linux/skbuff.h: 1631
0xffff000008938558 <inet_sock_destruct+72>:    ldr     x0, [x20]
/home/jiangdi/luyun/klinux-4.19/./include/linux/skbuff.h: 1633
0xffff00000893855c <inet_sock_destruct+76>:    cmp     x20, x0
0xffff000008938560 <inet_sock_destruct+80>:    b.ne    0xffff000008938530 <inet_sock_destruct+32>
```

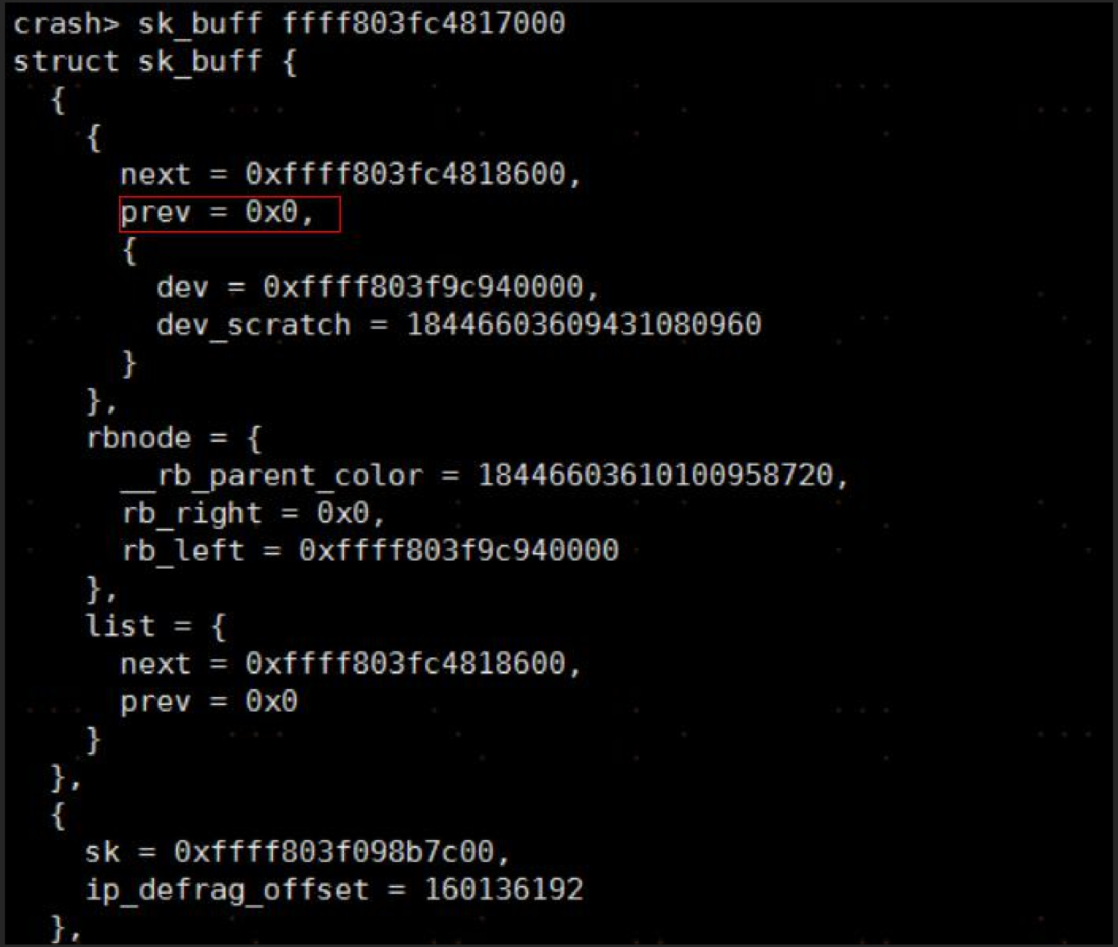
现网问题实战

可以看到问题是在访问x2寄存器保存的skb地址

偏移8地址时发生了空指针引用



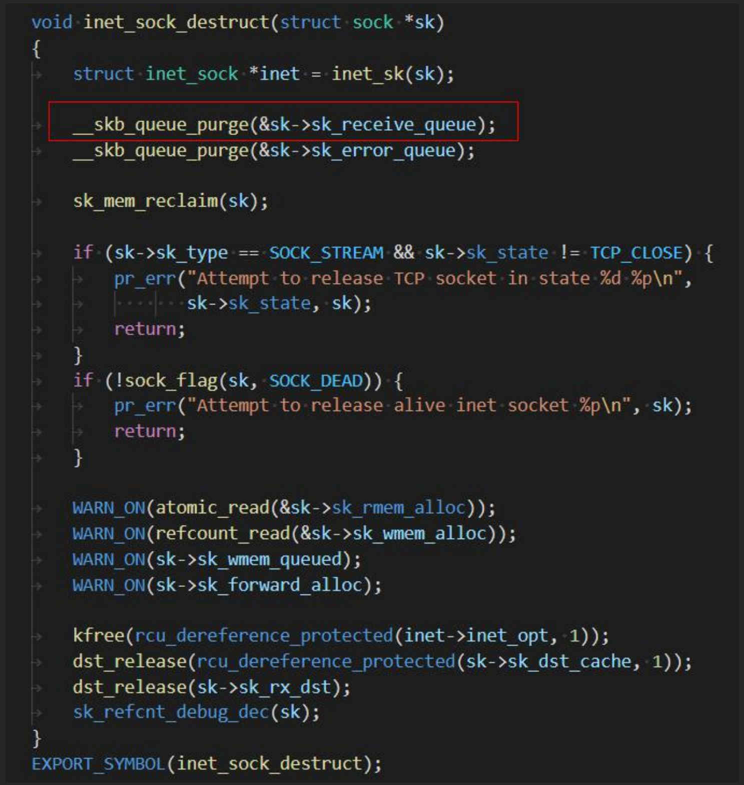
```
crash> sk_buff ffff803fc4817000  -o
struct sk_buff {
        union {
            struct {
  [ffff803fc4817000]         struct sk_buff *next;
  [ffff803fc4817008]         struct sk_buff *prev;
                union {
  [ffff803fc4817010]             struct net_device *dev;
  [ffff803fc4817010]             unsigned long dev_scratch;
                };
            };
  [ffff803fc4817000]     struct rb_node rbnode;
  [ffff803fc4817000]     struct list_head list;
        };
```



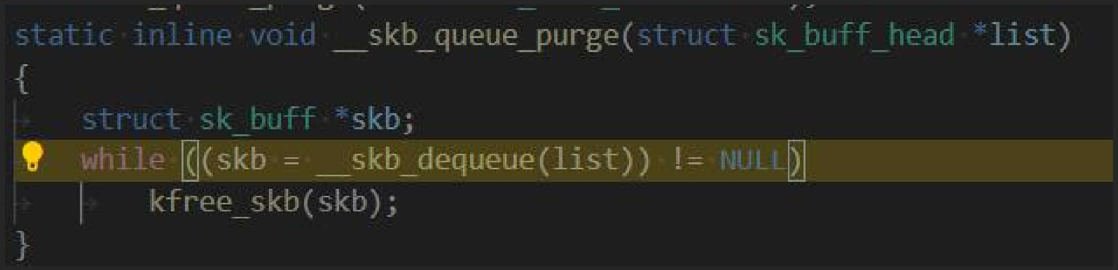
```
crash> sk_buff ffff803fc4817000
struct sk_buff {
  {
    {
      next = 0xffff803fc4818600,
      prev = 0x0,
      {
        dev = 0xffff803f9c940000,
        dev_scratch = 18446603609431080960
      }
    },
    rbnode = {
      __rb_parent_color = 18446603610100958720,
      rb_right = 0x0,
      rb_left = 0xffff803f9c940000
    },
    list = {
      next = 0xffff803fc4818600,
      prev = 0x0
    }
  },
  {
    sk = 0xffff803f098b7c00,
    ip_defrag_offset = 160136192
  },
```

现网问题实战

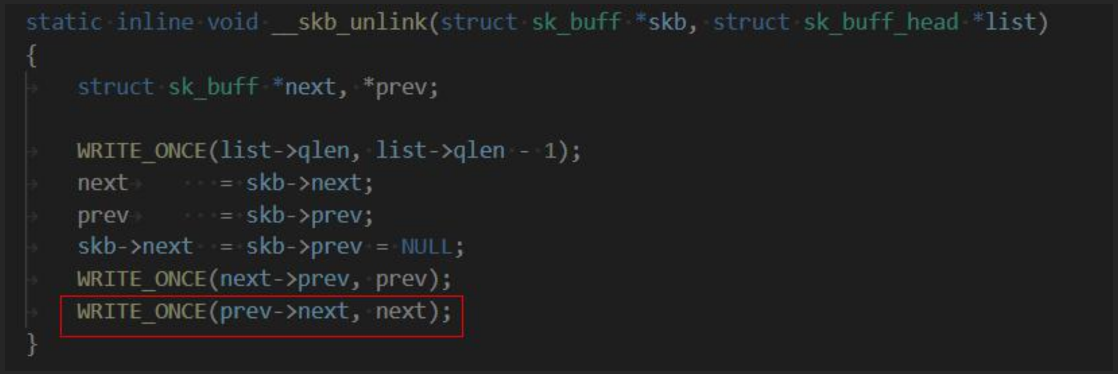
查看对应的C代码



```c
void inet_sock_destruct(struct sock *sk)
{
	struct inet_sock *inet = inet_sk(sk);

	__skb_queue_purge(&sk->sk_receive_queue);
	__skb_queue_purge(&sk->sk_error_queue);

	sk_mem_reclaim(sk);

	if (sk->sk_type == SOCK_STREAM && sk->sk_state != TCP_CLOSE) {
		pr_err("Attempt to release TCP socket in state %d %p\n",
		       sk->sk_state, sk);
		return;
	}
	if (!sock_flag(sk, SOCK_DEAD)) {
		pr_err("Attempt to release alive inet socket %p\n", sk);
		return;
	}

	WARN_ON(atomic_read(&sk->sk_rmem_alloc));
	WARN_ON(refcount_read(&sk->sk_wmem_alloc));
	WARN_ON(sk->sk_wmem_queued);
	WARN_ON(sk->sk_forward_alloc);

	kfree(rcu_dereference_protected(inet->inet_opt, 1));
	dst_release(rcu_dereference_protected(sk->sk_dst_cache, 1));
	dst_release(sk->sk_rx_dst);
	sk_refcnt_debug_dec(sk);
}
EXPORT_SYMBOL(inet_sock_destruct);
```



```c
static inline void __skb_queue_purge(struct sk_buff_head *list)
{
	struct sk_buff *skb;
	while ((skb = __skb_dequeue(list)) != NULL)
		kfree_skb(skb);
}
```

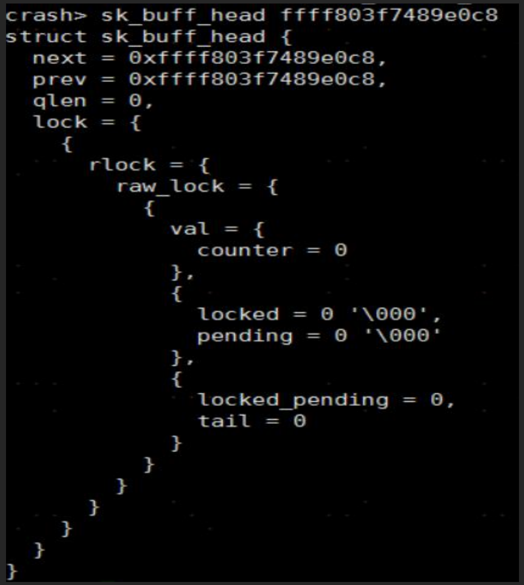


```c
static inline void __skb_unlink(struct sk_buff *skb, struct sk_buff_head *list)
{
	struct sk_buff *next, *prev;

	WRITE_ONCE(list->qlen, list->qlen - 1);
	next	   = skb->next;
	prev	   = skb->prev;
	skb->next  = skb->prev = NULL;
	WRITE_ONCE(next->prev, prev);
	WRITE_ONCE(prev->next, next);
}
```

现网问题实战

从上面反汇编代码可以知道，X20寄存器存放的是sk->sk_receive_queue,
解析可以看到qlen为0，说明receive队列的报文已经释放过

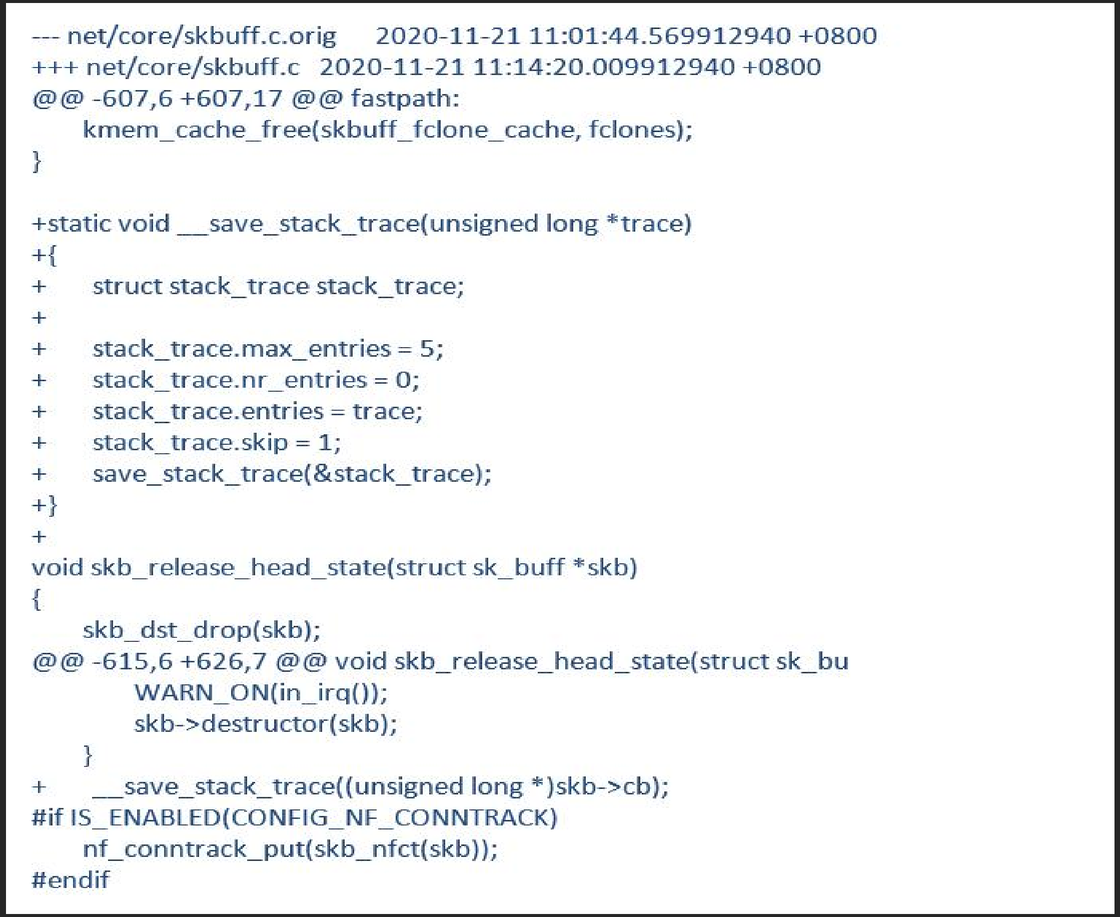
再查看x2寄存器保存的skb发现是其他连接的报文，与当前释放的sock不同，
所以怀疑原来的报文已经被释放过一次



```
crash> sk_buff_head ffff803f7489e0c8
struct sk_buff_head {
  next = 0xffff803f7489e0c8,
  prev = 0xffff803f7489e0c8,
  qlen = 0,
  lock = {
    {
      rlock = {
        raw_lock = {
          {
            val = {
              counter = 0
            },
            {
              locked = 0 '\000',
              pending = 0 '\000'
            },
            {
              locked_pending = 0,
              tail = 0
            }
          }
        }
      }
    }
  }
}
```

现网问题实战

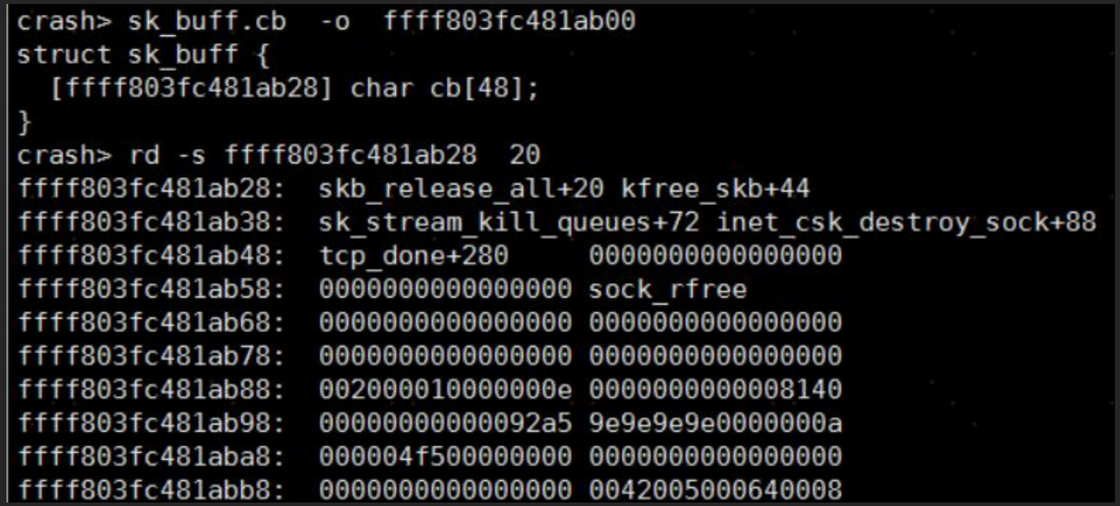
打上一个调测补丁，在skb报文释放时保存栈到skb->cb区域，然后尝试复现



```
--- net/core/skbuff.c.orig     2020-11-21 11:01:44.569912940 +0800
+++ net/core/skbuff.c   2020-11-21 11:14:20.009912940 +0800
@@ -607,6 +607,17 @@ fastpath:
      kmem_cache_free(skbuff_fclone_cache, fclones);
}

+static void __save_stack_trace(unsigned long *trace)
+{
+       struct stack_trace stack_trace;
+
+       stack_trace.max_entries = 5;
+       stack_trace.nr_entries = 0;
+       stack_trace.entries = trace;
+       stack_trace.skip = 1;
+       save_stack_trace(&stack_trace);
+}
+
void skb_release_head_state(struct sk_buff *skb)
{
      skb_dst_drop(skb);
@@ -615,6 +626,7 @@ void skb_release_head_state(struct sk_bu
            WARN_ON(in_irq());
            skb->destructor(skb);
      }
+       __save_stack_trace((unsigned long *)skb->cb);
#if IS_ENABLED(CONFIG_NF_CONNTRACK)
      nf_conntrack_put(skb_nfct(skb));
#endif
```

现网问题实战

问题出现时读取receive队列的skb(即x0)的cb区域，
从中可以看到保存的报文之前释放的路径:
tcp_done –>inet_csk_destroy_sock->skb_release_all->skb_release_all->kfree_skb

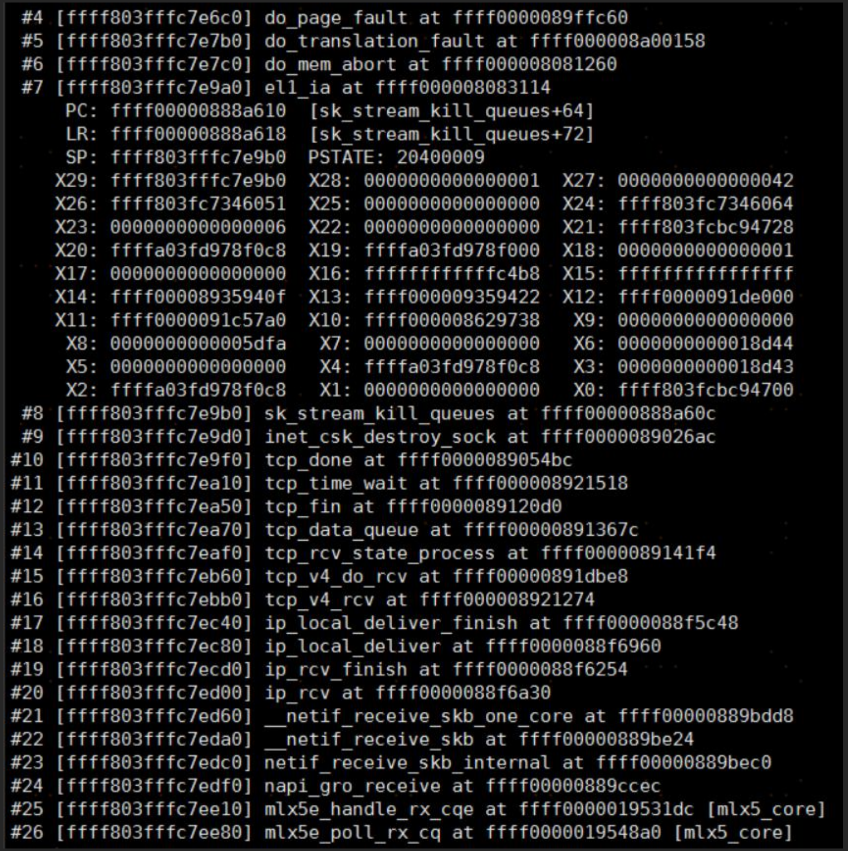


```
crash> sk_buff.cb  -o  ffff803fc481ab00
struct sk_buff {
  [ffff803fc481ab28] char cb[48];
}
crash> rd -s ffff803fc481ab28  20
ffff803fc481ab28:  skb_release_all+20 kfree_skb+44
ffff803fc481ab38:  sk_stream_kill_queues+72 inet_csk_destroy_sock+88
ffff803fc481ab48:  tcp_done+280      0000000000000000
ffff803fc481ab58:  0000000000000000 sock_rfree
ffff803fc481ab68:  0000000000000000 0000000000000000
ffff803fc481ab78:  0000000000000000 0000000000000000
ffff803fc481ab88:  002000010000000e 0000000000008140
ffff803fc481ab98:  00000000000092a5 9e9e9e9e0000000a
ffff803fc481aba8:  000004f500000000 0000000000000000
ffff803fc481abb8:  0000000000000000 0042005000640008
```

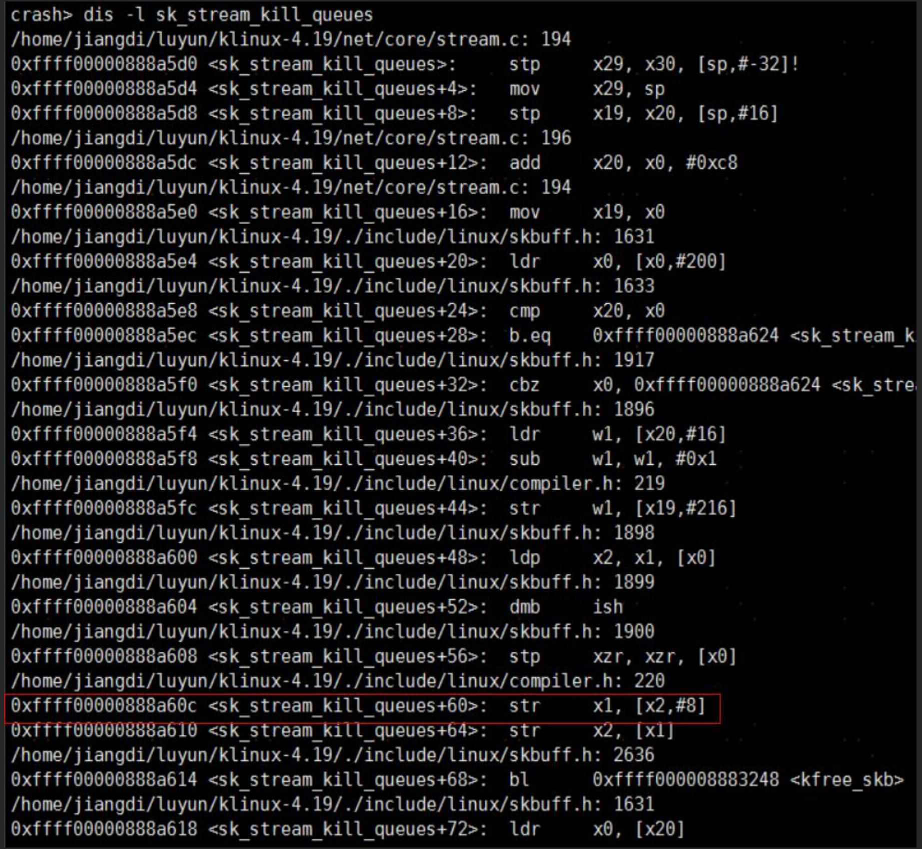
现网问题实战

对照vmcore的栈

问题也发生在对x2寄存器保存skb的访问



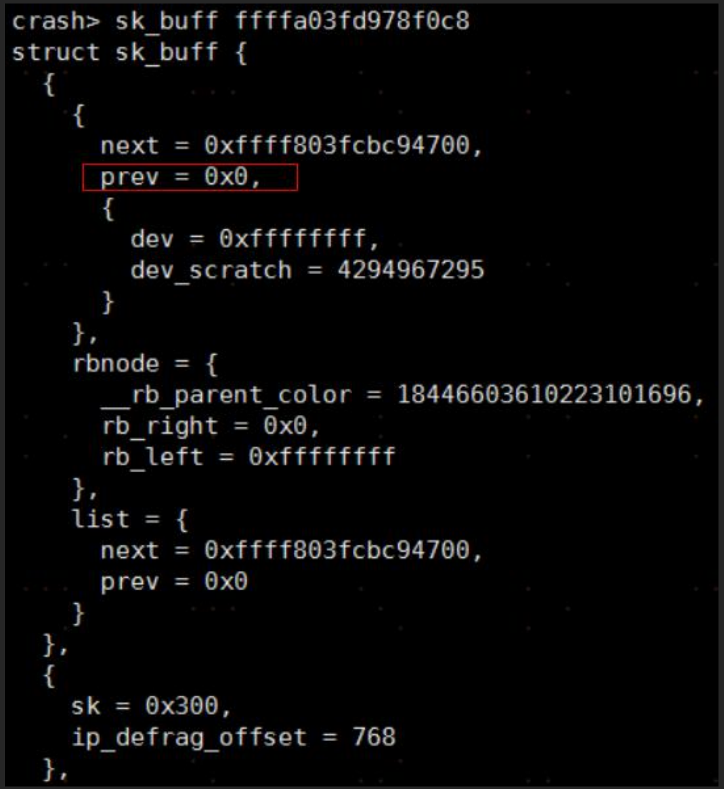
```
 #4 [ffff803fffc7e6c0] do_page_fault at ffff0000089ffc60
 #5 [ffff803fffc7e7b0] do_translation_fault at ffff000008a00158
 #6 [ffff803fffc7e7c0] do_mem_abort at ffff000008081260
 #7 [ffff803fffc7e9a0] el1_ia at ffff000008083114
     PC: ffff00000888a610  [sk_stream_kill_queues+64]
     LR: ffff00000888a618  [sk_stream_kill_queues+72]
     SP: ffff803fffc7e9b0  PSTATE: 20400009
    X29: ffff803fffc7e9b0  X28: 0000000000000001  X27: 0000000000000042
    X26: ffff803fc7346051  X25: 0000000000000000  X24: ffff803fc7346064
    X23: 0000000000000006  X22: 0000000000000000  X21: ffff803fcbc94728
    X20: ffffa03fd978f0c8  X19: ffffa03fd978f000  X18: 0000000000000001
    X17: 0000000000000000  X16: ffffffffffffc4b8  X15: ffffffffffffffff
    X14: ffff00008935940f  X13: ffff000009359422  X12: ffff0000091de000
    X11: ffff0000091c57a0  X10: ffff000008629738   X9: 0000000000000000
     X8: 0000000000005dfa   X7: 0000000000000000   X6: 0000000000018d44
     X5: 0000000000000000   X4: ffffa03fd978f0c8   X3: 0000000000018d43
     X2: ffffa03fd978f0c8   X1: 0000000000000000   X0: ffff803fcbc94700
 #8 [ffff803fffc7e9b0] sk_stream_kill_queues at ffff00000888a60c
 #9 [ffff803fffc7e9d0] inet_csk_destroy_sock at ffff0000089026ac
#10 [ffff803fffc7e9f0] tcp_done at ffff0000089054bc
#11 [ffff803fffc7ea10] tcp_time_wait at ffff000008921518
#12 [ffff803fffc7ea50] tcp_fin at ffff0000089120d0
#13 [ffff803fffc7ea70] tcp_data_queue at ffff00000891367c
#14 [ffff803fffc7eaf0] tcp_rcv_state_process at ffff0000089141f4
#15 [ffff803fffc7eb60] tcp_v4_do_rcv at ffff00000891dbe8
#16 [ffff803fffc7ebb0] tcp_v4_rcv at ffff000008921274
#17 [ffff803fffc7ec40] ip_local_deliver_finish at ffff0000088f5c48
#18 [ffff803fffc7ec80] ip_local_deliver at ffff0000088f6960
#19 [ffff803fffc7ecd0] ip_rcv_finish at ffff0000088f6254
#20 [ffff803fffc7ed00] ip_rcv at ffff0000088f6a30
#21 [ffff803fffc7ed60] __netif_receive_skb_one_core at ffff00000889bdd8
#22 [ffff803fffc7eda0] __netif_receive_skb at ffff00000889be24
#23 [ffff803fffc7edc0] netif_receive_skb_internal at ffff00000889bec0
#24 [ffff803fffc7edf0] napi_gro_receive at ffff00000889ccec
#25 [ffff803fffc7ee10] mlx5e_handle_rx_cqe at ffff0000019531dc [mlx5_core]
#26 [ffff803fffc7ee80] mlx5e_poll_rx_cq at ffff0000019548a0 [mlx5_core]
```



```
crash> dis -l sk_stream_kill_queues
/home/jiangdi/luyun/klinux-4.19/net/core/stream.c: 194
0xffff00000888a5d0 <sk_stream_kill_queues>:     stp     x29, x30, [sp,#-32]!
0xffff00000888a5d4 <sk_stream_kill_queues+4>:   mov     x29, sp
0xffff00000888a5d8 <sk_stream_kill_queues+8>:   stp     x19, x20, [sp,#16]
/home/jiangdi/luyun/klinux-4.19/net/core/stream.c: 196
0xffff00000888a5dc <sk_stream_kill_queues+12>:  add     x20, x0, #0xc8
/home/jiangdi/luyun/klinux-4.19/net/core/stream.c: 194
0xffff00000888a5e0 <sk_stream_kill_queues+16>:  mov     x19, x0
/home/jiangdi/luyun/klinux-4.19/./include/linux/skbuff.h: 1631
0xffff00000888a5e4 <sk_stream_kill_queues+20>:  ldr     x0, [x0,#200]
/home/jiangdi/luyun/klinux-4.19/./include/linux/skbuff.h: 1633
0xffff00000888a5e8 <sk_stream_kill_queues+24>:  cmp     x20, x0
0xffff00000888a5ec <sk_stream_kill_queues+28>:  b.eq    0xffff00000888a624 <sk_stream_k
/home/jiangdi/luyun/klinux-4.19/./include/linux/skbuff.h: 1917
0xffff00000888a5f0 <sk_stream_kill_queues+32>:  cbz     x0, 0xffff00000888a624 <sk_stre
/home/jiangdi/luyun/klinux-4.19/./include/linux/skbuff.h: 1896
0xffff00000888a5f4 <sk_stream_kill_queues+36>:  ldr     w1, [x20,#16]
0xffff00000888a5f8 <sk_stream_kill_queues+40>:  sub     w1, w1, #0x1
/home/jiangdi/luyun/klinux-4.19/./include/linux/compiler.h: 219
0xffff00000888a5fc <sk_stream_kill_queues+44>:  str     w1, [x19,#216]
/home/jiangdi/luyun/klinux-4.19/./include/linux/skbuff.h: 1898
0xffff00000888a600 <sk_stream_kill_queues+48>:  ldp     x2, x1, [x0]
/home/jiangdi/luyun/klinux-4.19/./include/linux/skbuff.h: 1899
0xffff00000888a604 <sk_stream_kill_queues+52>:  dmb     ish
/home/jiangdi/luyun/klinux-4.19/./include/linux/skbuff.h: 1900
0xffff00000888a608 <sk_stream_kill_queues+56>:  stp     xzr, xzr, [x0]
/home/jiangdi/luyun/klinux-4.19/./include/linux/compiler.h: 220
0xffff00000888a60c <sk_stream_kill_queues+60>:  str     x1, [x2,#8]
0xffff00000888a610 <sk_stream_kill_queues+64>:  str     x2, [x1]
/home/jiangdi/luyun/klinux-4.19/./include/linux/skbuff.h: 2636
0xffff00000888a614 <sk_stream_kill_queues+68>:  bl      0xffff000008883248 <kfree_skb>
/home/jiangdi/luyun/klinux-4.19/./include/linux/skbuff.h: 1631
0xffff00000888a618 <sk_stream_kill_queues+72>:  ldr     x0, [x20]
```

现网问题实战

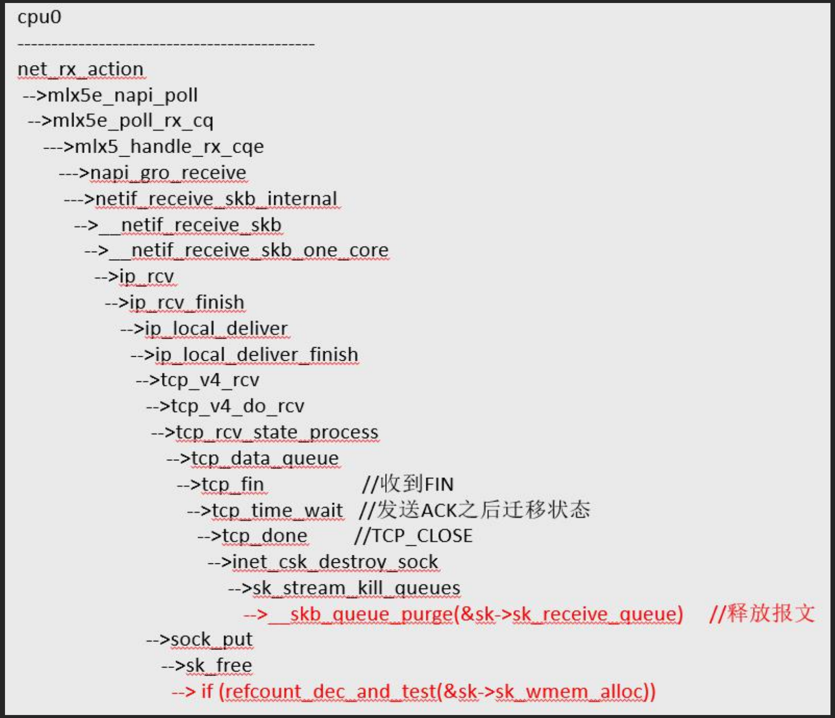
解析x2寄存器中的skb报文，可以看到prev指针也为空



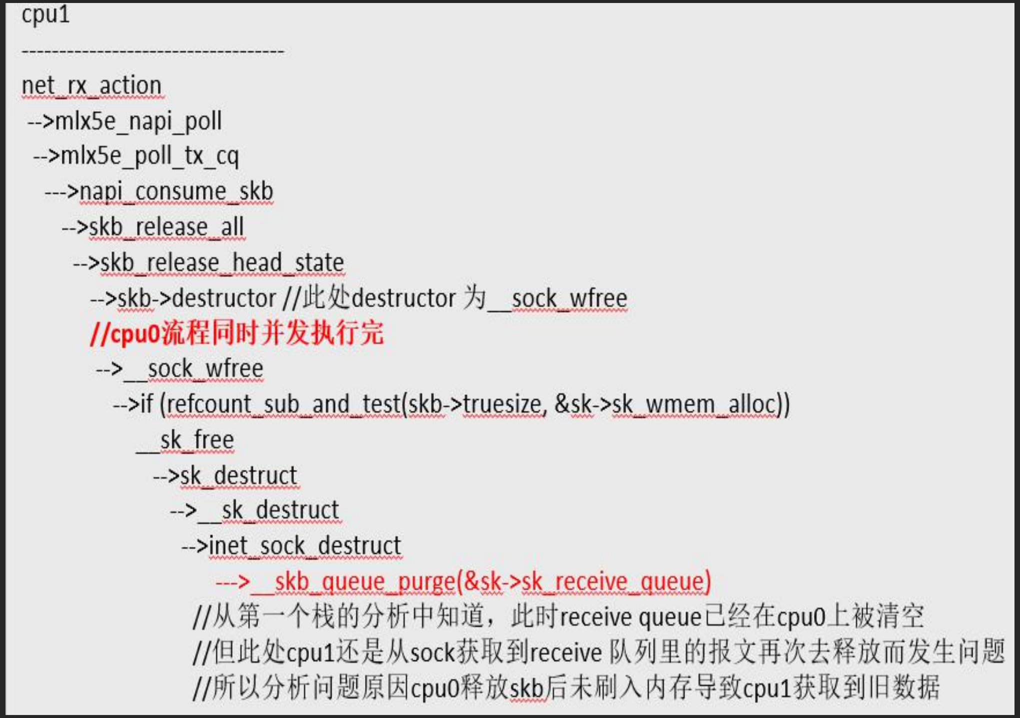
```
crash> sk_buff ffffa03fd978f0c8
struct sk_buff {
  {
    {
      next = 0xffff803fcbc94700,
      prev = 0x0,
      {
        dev = 0xffffffff,
        dev_scratch = 4294967295
      }
    },
    rbnode = {
      __rb_parent_color = 18446603610223101696,
      rb_right = 0x0,
      rb_left = 0xffffffff
    },
    list = {
      next = 0xffff803fcbc94700,
      prev = 0x0
    }
  },
  {
    sk = 0x300,
    ip_defrag_offset = 768
  },
```

现网问题实战

从上述两个vmcore的栈分析， 可以看到存在
两条并发路径操作同一sock的receive队列：



```
cpu0
---------------------------------------------
net_rx_action
 -->mlx5e_napi_poll
  -->mlx5e_poll_rx_cq
    --->mlx5_handle_rx_cqe
      --->napi_gro_receive
       --->netif_receive_skb_internal
         -->__netif_receive_skb
           -->__netif_receive_skb_one_core
             -->ip_rcv
               -->ip_rcv_finish
                 -->ip_local_deliver
                   -->ip_local_deliver_finish
                    -->tcp_v4_rcv
                      -->tcp_v4_do_rcv
                       -->tcp_rcv_state_process
                         -->tcp_data_queue
                           -->tcp_fin              //收到FIN
                             -->tcp_time_wait  //发送ACK之后迁移状态
                               -->tcp_done       //TCP_CLOSE
                                 -->inet_csk_destroy_sock
                                   -->sk_stream_kill_queues
                                     -->__skb_queue_purge(&sk->sk_receive_queue)    //释放报文
                      -->sock_put
                        -->sk_free
                         --> if (refcount_dec_and_test(&sk->sk_wmem_alloc))
```



```
cpu1
-----------------------------------
net_rx_action
 -->mlx5e_napi_poll
  -->mlx5e_poll_tx_cq
    --->napi_consume_skb
      -->skb_release_all
        -->skb_release_head_state
          -->skb->destructor //此处destructor 为__sock_wfree
          //cpu0流程同时并发执行完
           -->__sock_wfree
             -->if (refcount_sub_and_test(skb->truesize, &sk->sk_wmem_alloc))
                __sk_free
                  -->sk_destruct
                    -->__sk_destruct
                      -->inet_sock_destruct
                         --->__skb_queue_purge(&sk->sk_receive_queue)
                     //从第一个栈的分析中知道，此时receive queue已经在cpu0上被清空
                     //但此处cpu1还是从sock获取到receive 队列里的报文再次去释放而发生问题
                     //所以分析问题原因cpu0释放skb后未刷入内存导致cpu1获取到旧数据
```

现网问题实战

根因总结：

网卡在对同一条tcp流做hash时收发分发到了不同的CPU上处理，

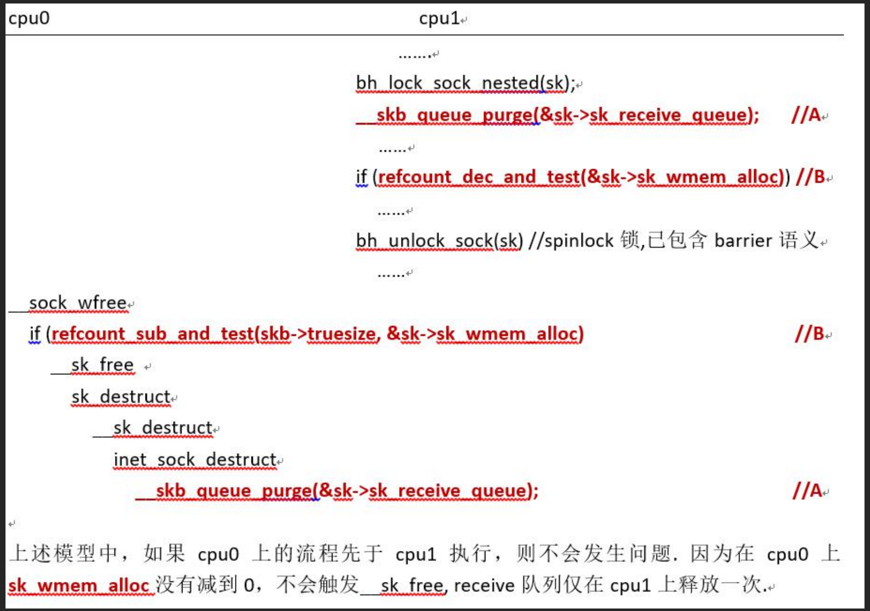
cpu1在收到FIN报文处理流程中会调用__skb_queue_purge清空释放sock的receive队列，

同时cpu0上在释放回收发送的ACK报文时也会调用__skb_queue_purge清空receive队列.

但在cpu0 __sock_wfree之前没有内存屏障，存在内存序问题，

导致cpu0获取到sk_receive_queued的过期数据，从而触发double free.

抽象简单模型如下：



```
cpu0                                        cpu1
                                                 .......
                                            bh_lock_sock_nested(sk);
                                            __skb_queue_purge(&sk->sk_receive_queue);     //A
                                              ......
                                            if (refcount_dec_and_test(&sk->sk_wmem_alloc)) //B
                                              ......
                                            bh_unlock_sock(sk) //spinlock 锁,已包含 barrier 语义
                                              ......
__sock_wfree
  if (refcount_sub_and_test(skb->truesize, &sk->sk_wmem_alloc)                           //B
    __sk_free
      sk_destruct
        __sk_destruct
          inet_sock_destruct
            __skb_queue_purge(&sk->sk_receive_queue);                                     //A
```

上述模型中，如果 cpu0 上的流程先于 cpu1 执行，则不会发生问题. 因为在 cpu0 上 sk_wmem_alloc 没有减到 0，不会触发__sk_free, receive 队列仅在 cpu1 上释放一次.

解决方案：

__sock_wfree在执行sk_free前需要加内存屏障

Thank you

openEuler