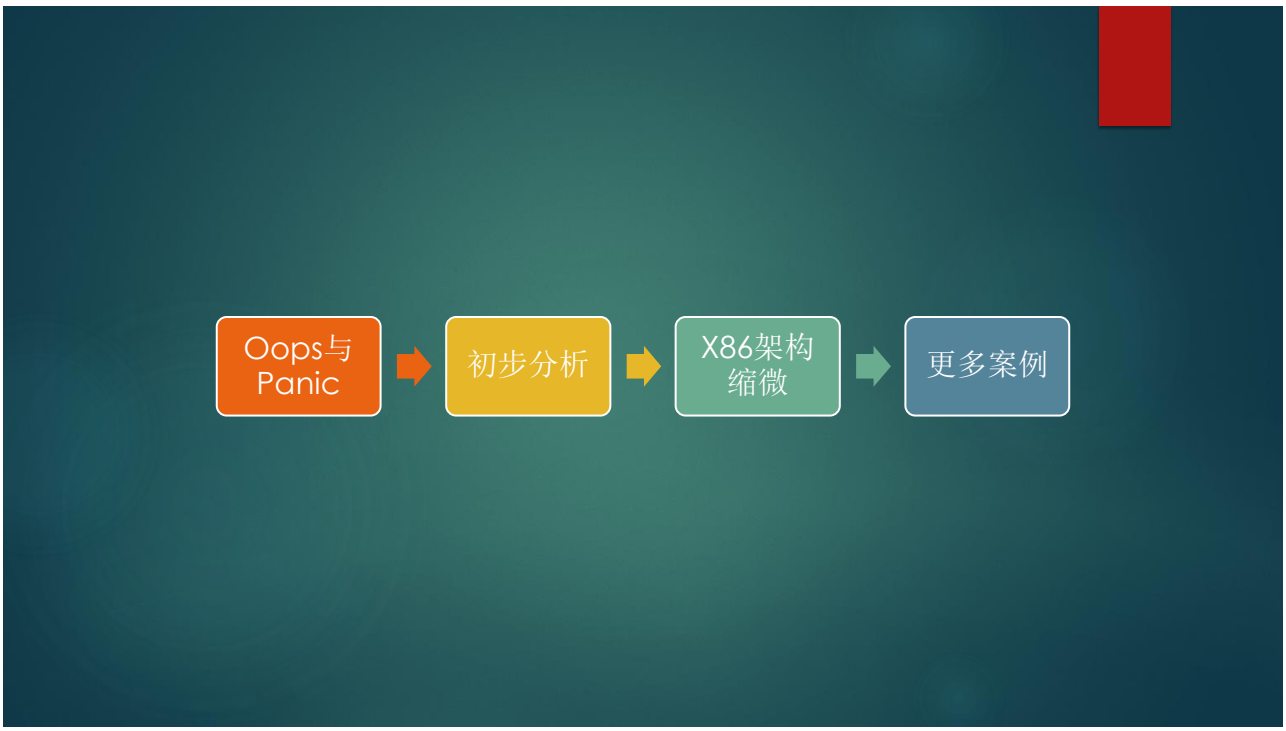
LINUX内核开发与调试 -- Panic

张银奎

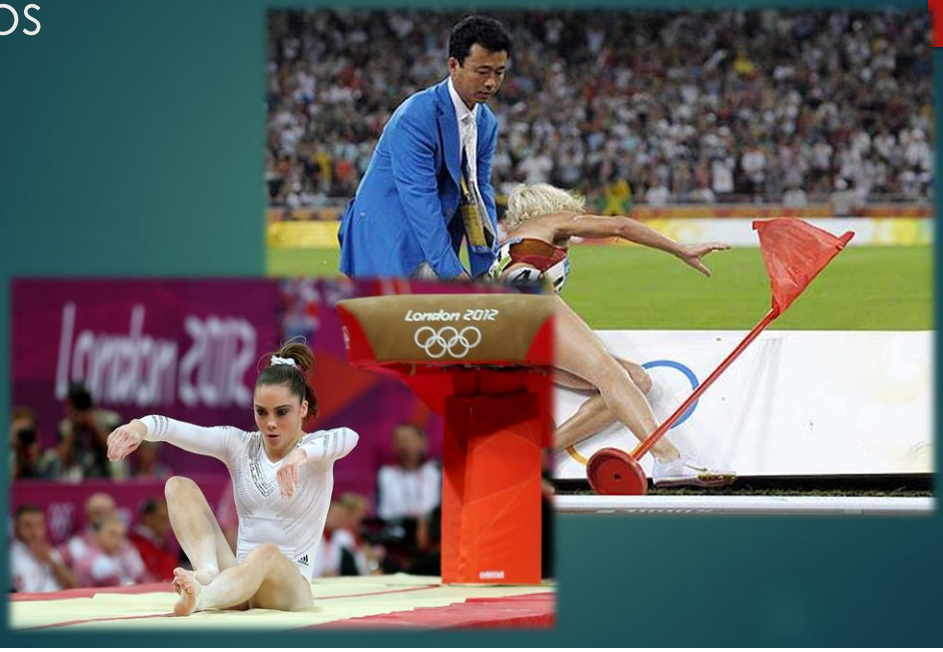
2017/11/22

1



2

oops



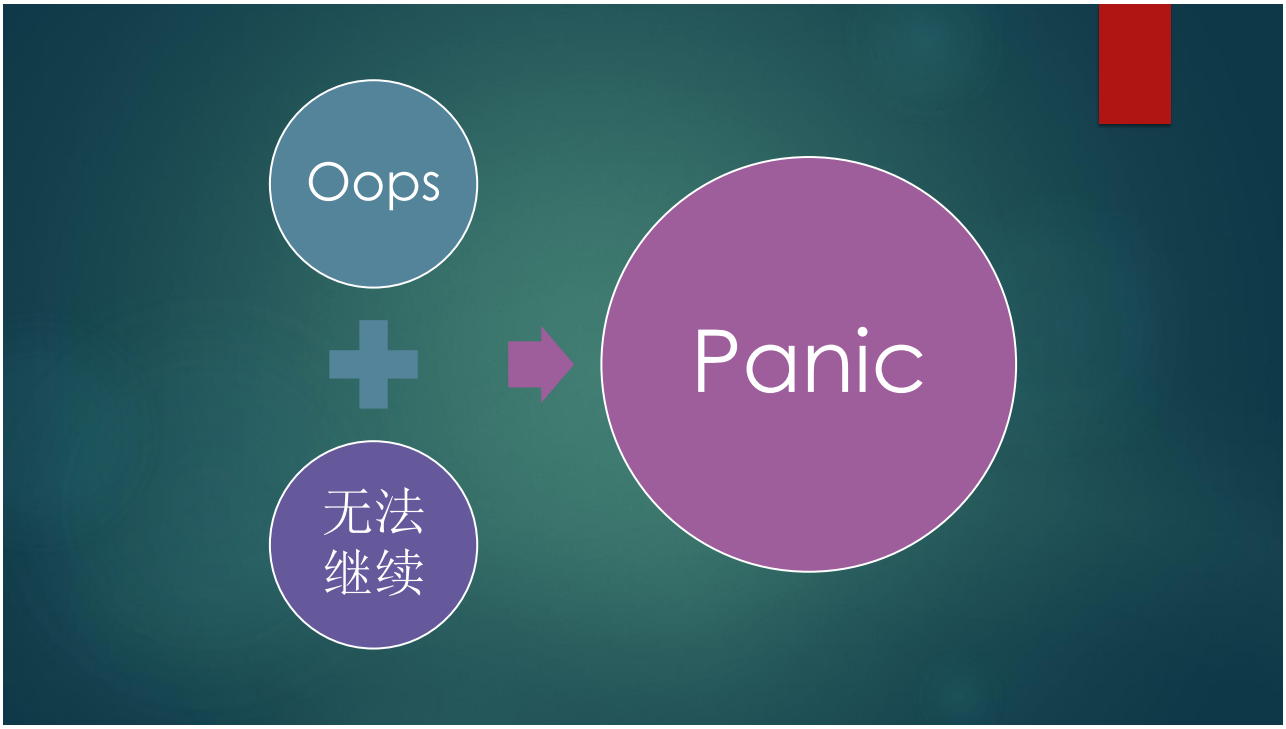
3

Lister - [D:\bench\linux-3.16.3\Documentation\oops-tracing.txt]

File Edit Options Help

```
NOTE: ksymoops is useless on 2.6.  Please use the Oops in its original format
(from dmesg, etc).  Ignore any references in this or other docs to "decoding
the Oops" or "running it through ksymoops".  If you post an Oops from 2.6 that
has been run through ksymoops, people will just tell you to repost it.

Quick Summary
-------------

Find the Oops and send it to the maintainer of the kernel area that seems to be
involved with the problem.  Don't worry too much about getting the wrong person.
If you are unsure send it to the person responsible for the code relevant to
what you were doing.  If it occurs repeatably try and describe how to recreate
it.  That's worth even more than the oops.

If you are totally stumped as to whom to send the report, send it to
linux-kernel@vger.kernel.org. Thanks for your help in making Linux as
stable as humanly possible.
```

4



5

古老的机制

- The OpenSolaris version of panic() was released by Sun in 2005. It is fairly elaborate, and its header comments explain a lot about what happens in a panic situation.
- The Unix V4 implementation of panic() was released in 1973. It basically just prints the core state of the kernel to the console and stops the processor.
- That function is substantially unchanged in Unix V3 according to Amit Singh, who famously dissected an older version of Mac OS X and explained it. That first link takes you to a lovely article explaining macOS's approach to the implementation of panic(), which starts off with a relevant historical discussion.
- The "unix-jun72" project to resurrect Unix V1 from scanned source code printouts shows a very early PDP-11 assembly version of this function, written sometime before June 1972, before Unix was fully rewritten in C. By this point, its implementation is whittled down to a 6-instruction routine that does little more than restart the PDP-11.

6

是否把事件升级

```
void oops_end(unsigned long flags, struct pt_regs *regs, int signr)
{
        if (regs && kexec_should_crash(current))
                crash_kexec(regs);
```

```
 if (in_interrupt() || !p->pid || is_global_init(p) || panic_on_oops)
        return 1;
```

7

可配置panic_on_oops

- /proc/sys/kernel/panic_on_oops
- CentOS上默认是1

8

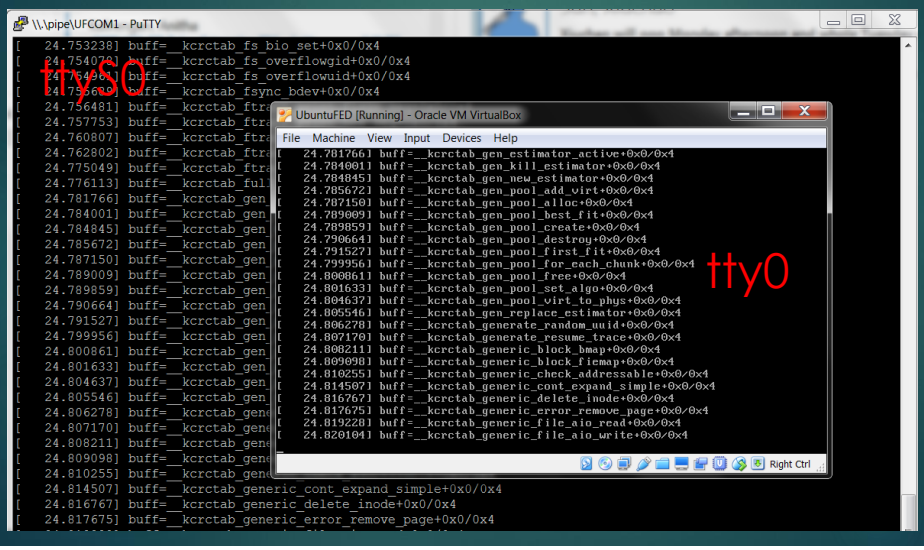
延时后重启

```
if (panic_timeout > 0) {
        /*
         * Delay timeout seconds before rebooting the machine.
         * We can't use the "normal" timers since we just panicked.
         */
        pr_emerg("Rebooting in %d seconds..", panic_timeout);


int panic_timeout = CONFIG_PANIC_TIMEOUT;
```

9

同时输出

console=tty0 console=ttyS0,9600n8



10

Lister - [c:\bench\linux-4.4.14\Documentation\kdump\kdump.txt]

File Edit Options Help

```
==========================================================
Documentation for Kdump - The kexec-based Crash Dumping Solution
==========================================================

This document includes overview, setup and installation, and analysis
information.

Overview
========

Kdump uses kexec to quickly boot to a dump-capture kernel whenever a
dump of the system kernel's memory needs to be taken (for example, when
the system panics). The system kernel's memory image is preserved across
the reboot and is accessible to the dump-capture kernel.

You can use common commands, such as cp and scp, to copy the
memory image to a dump file on the local disk, or across the network to
a remote system.

Kdump and kexec are currently supported on the x86, x86_64, ppc64, ia64,
s390x and arm architectures.

When the system kernel boots, it reserves a small section of memory for
the dump-capture kernel. This ensures that ongoing Direct Memory Access
(DMA) from the system kernel does not corrupt the dump-capture kernel.
The kexec -p command loads the dump-capture kernel into this reserved
memory.

On x86 machines, the first 640 KB of physical memory is needed to boot,
regardless of where the kernel loads. Therefore, kexec backs up this
region just before rebooting into the dump-capture kernel.
```

kdump

11

Ubuntu 服务器指南

https://help.ubuntu.com/lts/serverguide/kernel-crash-dump.html

12

32.3. ANALYZING THE CORE DUMP

To determine the cause of the system crash, you can use the **crash** utility, which provides an interactive prompt very similar to the GNU Debugger (GDB). This utility allows you to interactively analyze a running Linux system as well as a core dump created by `netdump`, `diskdump`, `xendump`, or `kdump`.

Important

To analyze the `vmcore` dump file, you must have the crash and kernel-debuginfo packages installed. To install the crash package in your system, type the following at a shell prompt as `root`:

```
yum install crash
```

To install the kernel-debuginfo package, make sure that you have the yum-utils package installed and run the following command as `root`:

```
debuginfo-install kernel
```

https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/6/html/Deployment_Guide/s1-kdump-crash.html

13

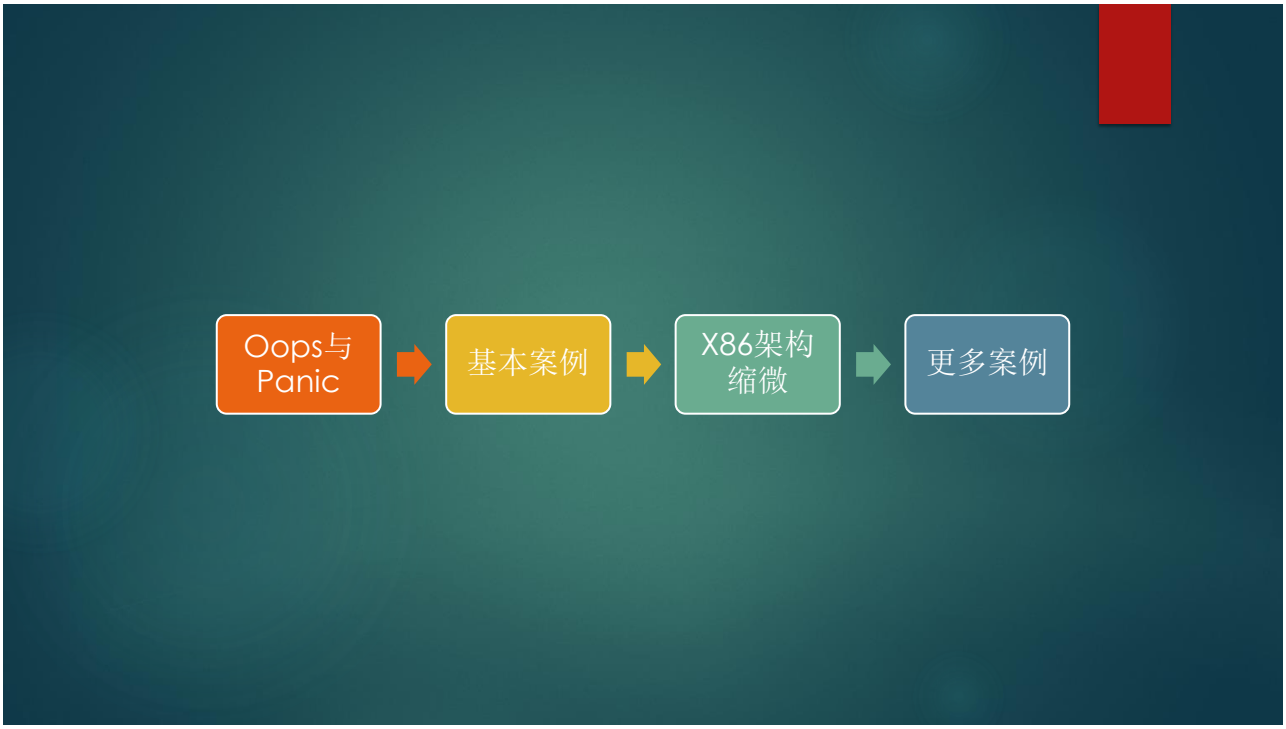
Lister - [c:\bench\linux-4.4.14\Documentation\kdump\gdbmacros.txt]

File Edit Options Help

```
        printf "Trapno %ld, cr2 0x%lx, error_code %ld\n", $pid_task.thread.trap_no, \
                                $pid_task.thread.cr2, $pid_task.thread.error_code

end
document trapinfo
        Run info threads and lookup pid of thread #1
        'trapinfo <pid>' will tell you by which trap & possibly
        address the kernel panicked.
end


define dmesg
        set $i = 0
        set $end_idx = (log_end - 1) & (log_buf_len - 1)

        while ($i < logged_chars)
                set $idx = (log_end - 1 - logged_chars + $i) & (log_buf_len - 1)

                if ($idx + 100 <= $end_idx) || \
                   ($end_idx <= $idx && $idx + 100 < log_buf_len)
                        printf "%.100s", &log_buf[$idx]
                        set $i = $i + 100
                else
                        printf "%c", log_buf[$idx]
                        set $i = $i + 1
                end
        end
end
document dmesg
        print the kernel ring buffer
end
```

GDB macros

14



15

真实案例

```
[    0.000000] tsc: Fast TSC calibration failed
[    0.878127] BUG: unable to handle kernel NULL pointer dereference at 00000008
[    0.881181] IP: [<c155b954>] kallsym_init+0x134/0x260
[    0.882117] *pdpt = 0000000000000000 *pde = f000ff53f000ff53
[    0.883045] Oops: 0000 [#1] SMP
[    0.885233] Modules linked in:
[    0.885703] CPU: 0 PID: 1 Comm: swapper/0 Not tainted 3.11.0gedu #9
[    0.890017] Hardware name: innotek GmbH VirtualBox/VirtualBox, BIOS VirtualBox 12/01/2006
[    0.894979] task: df480000 ti: df44a000 task.ti: df44a000
[    0.898298] EIP: 0060:[<c155b954>] EFLAGS: 00010246 CPU: 0
[    0.903263] EIP is at kallsym_init+0x134/0x260
[    0.909127] EAX: 00000000 EBX: c17f6ab8 ECX: 00000009 EDX: 00000000
[    0.911950] ESI: c17f626c EDI: c17f672c EBP: df44beac ESP: df44be90
[    0.916195]  DS: 007b ES: 007b FS: 00d8 GS: 00e0 SS: 0068
[    0.916973] CR0: 8005003b CR2: 00000008 CR3: 01a62000 CR4: 000406f0
[    0.917816] DR0: 00000000 DR1: 00000000 DR2: 00000000 DR3: 00000000
[    0.920026] DR6: fffe0ff0 DR7: 00000400
```

16

第二部分

```
[    0.924014] Stack:
[    0.926206]  c18655ce 00012fa6 c16c22d8 c170e174 00000000 00000105 c1a39cd4 df44bef8
[    0.929478]  c19ceb1e c1864976 00000060 000080d0 00000301 de3a15a0 de3a15a0 00000000
[    0.935932]  df44bee8 c13fd5bb c1b3b23c 00000000 00000105 c1a39cd0 df44bef0 c1551769
[    0.940598] Call Trace:
[    0.945476]  [<c19ceb1e>] fedcore_init+0x15/0x350
[    0.946233]  [<c13fd5bb>] ? __class_create+0x4b/0x70
[    0.946993]  [<c1551769>] ? create_extcon_class.part.2+0x19/0x30
[    0.949370]  [<c19ceb07>] ? extcon_class_init+0x13/0x15
[    0.950123]  [<c10020fc>] do_one_initcall+0xdc/0x1b0
[    0.956222]  [<c11c5583>] ? __proc_create+0xa3/0xe0
[    0.956953]  [<c19ceb09>] ? extcon_class_init+0x15/0x15
[    0.957707]  [<c1071243>] ? parse_args+0x283/0x480
[    0.960365]  [<c197dbe3>] kernel_init_freeable+0x11c/0x1b9
[    0.961339]  [<c197d514>] ? do_early_param+0x74/0x74
[    0.962062]  [<c1638eb0>] kernel_init+0x10/0xd0
[    0.963534]  [<c164f0f7>] ret_from_kernel_thread+0x1b/0x28
[    0.967251]  [<c1638ea0>] ? rest_init+0x70/0x70
```

17

问号之含义

```
void printk_address(unsigned long address, int reliable)
{
        pr_cont(" [<%p>] %s%pB\n",
                (void *)address, reliable ? "" : "? ", (void *)address);
}
```

- 不确定

18

第三部分

[0.969798] Code: be 03 e8 10 f6 ff ff 85 c0 74 14 83 c3 01 0f be 03 e8 01 f6 ff ff 85 c0 75 f1 83 c3 01 eb e0 89 d8 83 e0 03 74 05 29 c3 83 c3 04 <a1> 08 00 00 00 89 1d 3c e5 b4 c1 89 5c 24 0c 31 db 89 7c 24 08
[0.984078] EIP: [<c155b954>] kallsym_init+0x134/0x260 SS:ESP 0068:df44be90
[0.987013] CR2: 0000000000000008
[0.988500] ---[end trace 27c8b114d5190fc5]---
[0.989411] ata2.00: ATAPI: VBOX CD-ROM, 1.0, max UDMA/133
[0.990510] ata2.00: configured for UDMA/33
[0.992419] Kernel panic - not syncing: Attempted to kill init! exitcode=0x00 000009
[0.992419]
[0.999444] atkbd serio0: Spurious ACK on isa0060/serio0. Some program might be trying to access hardware directly.

19

CR2: 00000004

- 访问地址变了
- 启用了DBG_SYMTAB

```
#ifdef DBG_SYMTAB
    if(addr >= (unsigned long)0xffffffff8191df90)//0xc19ca000)
        printk("buff=%s\n",buff);
#endif

    //if is a function addr ,it's %pS print format must be: ...+0x0/...
    if((buff[0]=='0')&&(buff[1]=='x'))
        return 0;
    while(buff[i++]){
        if((buff[i] == '+')&&(buff[i+1]=='0')&&(buff[i+2]=='x')&&(buff[i+3]=='0')&&(buff[i+4]=='/'))
            return 1;
    }
    return 0;
```

20

Oops结束标志

```
void print_oops_end_marker(void)
{
      init_oops_id();
      pr_warn("---[ end trace %016llx ]---\n", (unsigned long long)oops_id);
}
```

```
[  221.528822] RIP  [<ffffffffc04b2050>] ll_timer_callback+0x20/0x30 [llaolao]
[  221.531636]  RSP <ffff8a91ffc03e50>
[  221.539468] CR2: 0000000000000bad
[  221.541840] ---[ end trace 6482fe703e5df570 ]---
[  221.547676] Kernel panic - not syncing: Fatal exception in interrupt
[  221.576064] Kernel Offset: 0x3aa00000 from 0xffffffff81000000 (relocation ra
ge: 0xffffffff80000000-0xffffffffbfffffff)
[  223.967344] ---[ end Kernel panic - not syncing: Fatal exception in interrup
```

21

改掉小BUG

```
[   30.527581] buff=.brk.early_pgt_alloc+0x0/0x6000
[   30.528101] buff=__brk_limit+0x0/0x0
[   30.529567] sym found the end c170e170
[   30.529959] end addr of symtable c170e170
[   30.530370] sym num=77734, addr=c16c22d8, name=c170e174
[   30.531141] BUG: unable to handle kernel NULL pointer dereference at 00000008
[   30.531878] IP: [<c155b660>] kallsym_init+0x140/0x260
[   30.532390] *pdpt = 0000000000000000 *pde = f000ff53f000ff53
[   30.533011] Oops: 0000 [#1] SMP
[   30.533356] Modules linked in:
[   30.533669] CPU: 0 PID: 1 Comm: swapper/0 Not tainted 3.11.0gedu #11
[   30.534311] Hardware name: innotek GmbH VirtualBox/VirtualBox, BIOS VirtualBox 12/01/2006
[   30.535124] task: df480000 ti: df44a000 task.ti: df44a000
[   30.535717] EIP: 0060:[<c155b660>] EFLAGS: 00010246 CPU: 0
[   30.539482] EIP is at kallsym_init+0x140/0x260
[   30.539929] EAX: 00000000 EBX: c17f6ab8 ECX: 00000009 EDX: 00000100
[   30.540559] ESI: c17f626c EDI: c17f672c EBP: df44beac ESP: df44be90
[   30.541190]  DS: 007b ES: 007b FS: 00d8 GS: 00e0 SS: 0068
[   30.541725] CR0: 8005003b CR2: 00000008 CR3: 01a62000 CR4: 000406f0
[   30.542349] DR0: 00000000 DR1: 00000000 DR2: 00000000 DR3: 00000000
[   30.542978] DR6: fffe0ff0 DR7: 00000400
[   30.543361] Stack:
[   30.543568]  c1896f18 00012fa6 c16c22d8 c170e174 00000000 00000105 c1a39cd4 df44bef8
[   30.546307]  c19ceb1e c1864976 00000060 000080d0 00000301 de3a15a0 de3a15a0 00000000
[   30.547282]  df44bee8 c13fd5bb c1b3b23c 00000000 00000105 c1a39cd0 df44bef0 c1551769
[   30.548199] Call Trace:
[   30.548454]  [<c19ceb1e>] fedcore_init+0x15/0x350
[   30.548951]  [<c13fd5bb>] ? __class_create+0x4b/0x70
[   30.549474]  [<c1551769>] ? create_extcon_class.part.2+0x19/0x30
[   30.550111]  [<c19ceb07>] ? extcon_class_init+0x13/0x15
```

22

真的BUG现身了

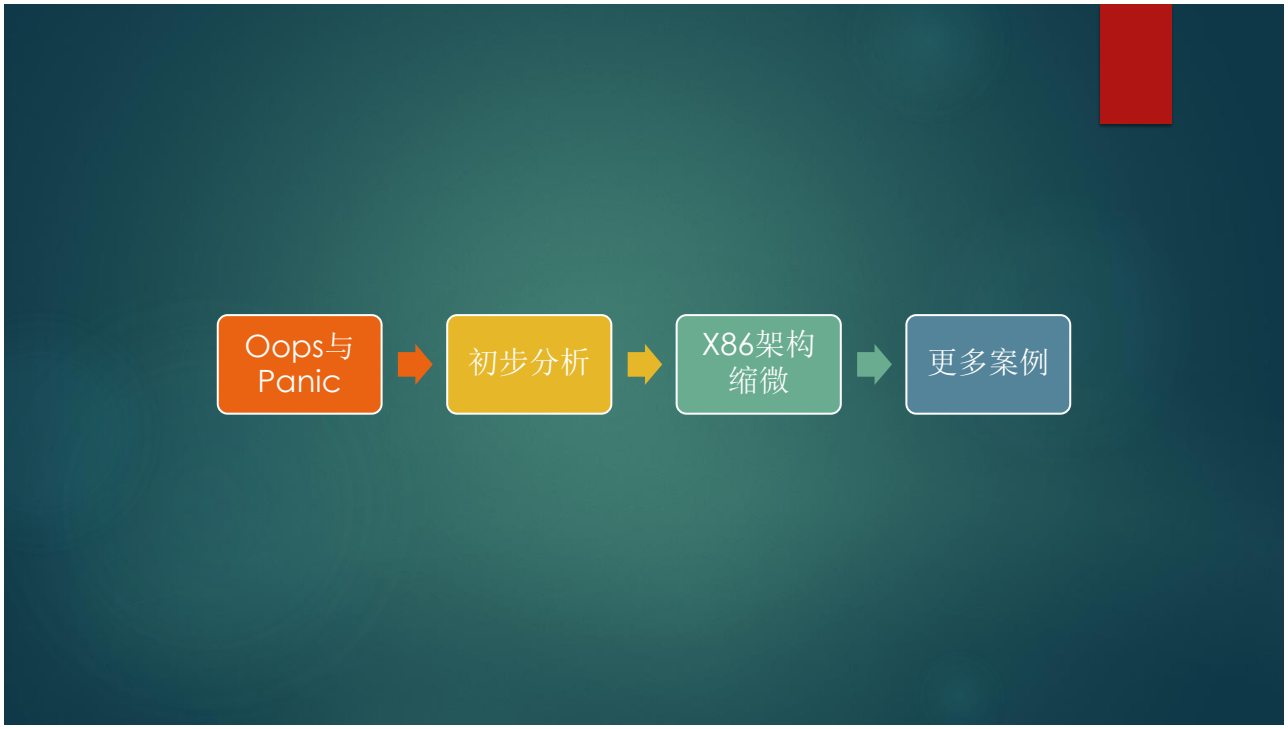
```
//initialize the main structrue eg. kallsyms_addr, kallsyms_name and so on
funaddr_endaddr = find_funadd_endaddr(printk_addr_addr);
kallsyms_num = *((unsigned long *)funaddr_endaddr + 1);
kallsyms_addr = (unsigned long *)funaddr_endaddr - kallsyms_num + 1;
kallsyms_name =  (void *)((unsigned long *)funaddr_endaddr + 2);

printk("sym num=%d, addr=%p, name=%p\n", kallsyms_num, kallsyms_addr, kallsyms_name);

kallsyms_mark =  get_marker_addr(kallsyms_name );

mark_num = kallsyms_num%256 ? kallsyms_num/256 + 1:kallsyms_num/256;
kallsyms_token_tab = (void *)((unsigned long *)kallsyms_mark + mark_num);

kallsyms_token_indx = get_index_addr(kallsyms_token_tab);
mod_header = THIS_MODULE->list.prev;

printk("sym mark=%p, token=%p, index=%p\n", kallsyms_mark,
    kallsyms_token_tab, kallsyms_token_indx);
```

23

改正

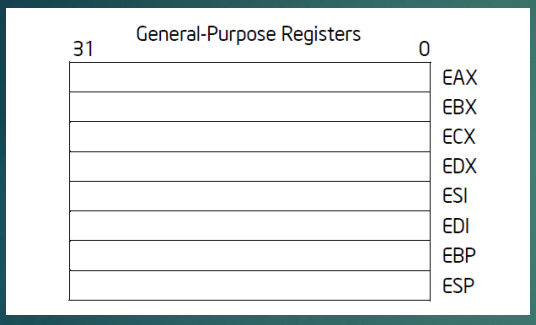
```
//initialize the main structrue eg. kallsyms_addr, kallsyms_name and so on
funaddr_endaddr = find_funadd_endaddr(printk_addr_addr);
kallsyms_num = *((unsigned long *)funaddr_endaddr + 1);
kallsyms_addr = (unsigned long *)funaddr_endaddr - kallsyms_num + 1;
kallsyms_name =  (void *)((unsigned long *)funaddr_endaddr + 2);

printk(KERN_ERR "sym num=%d, addr=%p, name=%p\n", kallsyms_num, kallsyms_addr, kallsyms_name);

kallsyms_mark =  get_marker_addr(kallsyms_name );

mark_num = kallsyms_num%256 ? kallsyms_num/256 + 1:kallsyms_num/256;
kallsyms_token_tab = (void *)((unsigned long *)kallsyms_mark + mark_num);

kallsyms_token_indx = get_index_addr(kallsyms_token_tab);
if(THIS_MODULE != NULL)
    mod_header = THIS_MODULE->list.prev;

printk(KERN_ERR "sym mark=%p, token=%p, index=%p\n", kallsyms_mark,
    kallsyms_token_tab, kallsyms_token_indx);
```

24



25

8个通用寄存器



- EAX函数返回值
- ECX循环次数
- ESI，EDI串操作的源和目标
- ESP栈顶
- EBP栈帧基地址
- 长模式下都为64位，RAX…

26

ntdll!memcpy

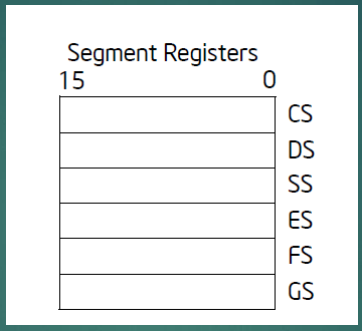

```
76f62345 8b750c          mov     esi,dword ptr [ebp+0Ch]
76f62348 8b4d10          mov     ecx,dword ptr [ebp+10h]
76f6234b 8b7d08          mov     edi,dword ptr [ebp+8]
76f62373 f3a5            rep movs dword ptr es:[edi],dword ptr [esi]
```

- 为性能考虑按DWORD操作，函数中要处理“零头”字节

Memset:
76f6df85 f3ab rep stos dword ptr es:[edi]

27

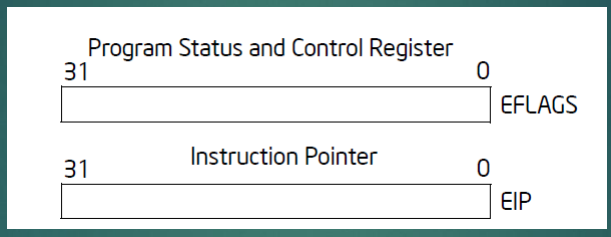
6个段寄存器



- 所有模式下都为16位长

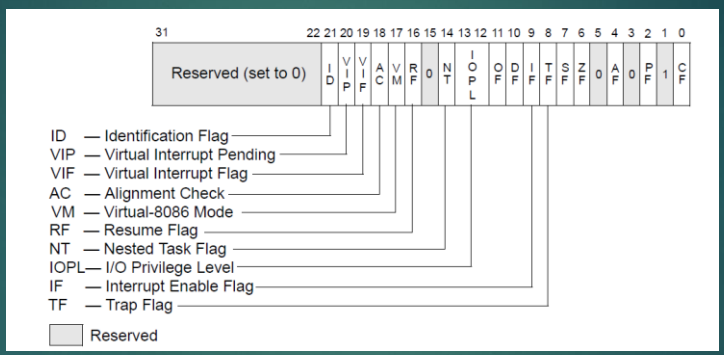
28

1个标志寄存器+1个IP



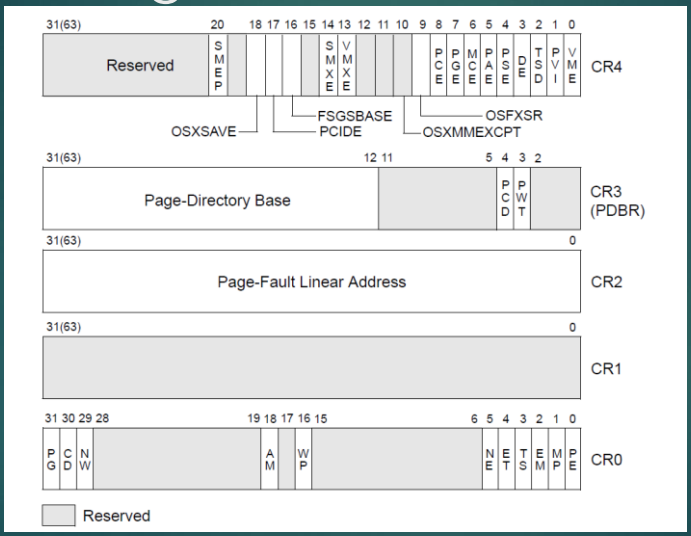
29

EFlags



30

Control Registers



22条常用x86指令 (1/2)

指令	机器码	说明
INT 3	0xCC	软件断点
NOP	0x90	空操作
PUSH	* 0x50/51 …(通用寄存器)	压入栈
POP	* 0x58/59…(通用寄存器)	从栈中弹出
ADD	* 0x80/81/83	加法
SUB	* 0x80/81/83	减法
IDIV	0xF6/0xF7	整数除法
RET	* 0xC3	函数返回
CALL	* 0xE8XXXXXXXX	调用函数
INC/DEC	* 0xFF	递增/递减
MOV	* 0x88/89/8A/8B/8C/8E	赋值

* 部分情况的机器码

22条常用x86指令 (2/2)

指令	机器码	说明
JMP	* 0xE9/EA/EB	绝对跳转
JZ/JNZ	0x74/75	条件跳转
JB/JNB	0x72/73	条件跳转
JA/JBE	0x77/76	条件跳转
JL/JGE	0x7C/7D	条件跳转
JG/JLE	0x7F/7E	条件跳转
TEST	* 0x85XX	逻辑比较
CMP	* 0x38/39/3A/3B	数学比较
XOR	* 0x30/31/32/33	异或
LEA	* 0x8DxxXXXXXXXXXX	取有效地址
MOVS	* 0xA4/A5	串赋值

* 部分情况的机器码

33

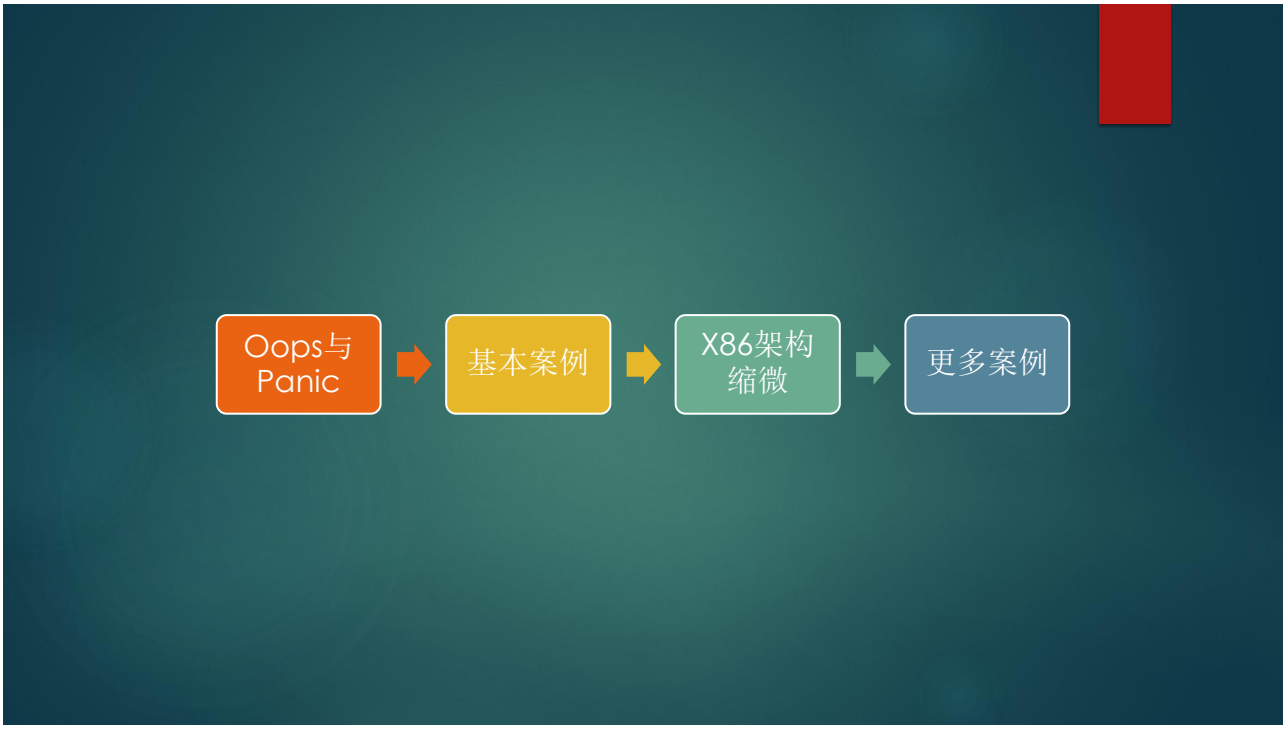
强大的x86指令

1020f789 c78491c400000000000000 mov dword ptr [ecx+edx*4+0C4h],0

```
for(i=0;i<ARRAY_LENGTH;i==)
{
    pArray[i]->m_nField = 0;
}
```

- 11字节机器码
- 循环处理一个数组，ECX指向数组的基地址，EDX做循环变量，索引数组的元素
- http://advdbg.org/blogs/advdbg_system/articles/6213.aspx

34



35

车载系统的案例

[413571.591498] Unable to handle kernel NULL pointer dereference at virtual address 0000002c
[413571.606530] pgd = ed5bc480
[413571.613725] [0000002c] *pgd=00000000
[413571.621839] Internal error: Oops: a05 [#1] PREEMPT SMP ARM
[413571.627435] Modules linked in: bcmdhd exfat(O) cmemk(O) pvrsrvkm(O) [last unloaded: bcmdhd]
[413571.635963] CPU: 1 PID: 188 Comm: mediaserver Tainted: G O 4.4.45-g66454cd #66
[413571.644436] Hardware name: Generic DRA74X (Flattened Device Tree)
[413571.650642] task: ec95a700 ti: ec98c000 task.ti: ec98c000
[413571.656157] PC is at rpmsg_sock_release+0xd8/0x114
[413571.661055] LR is at 0xfffffffa
[413571.664296] pc : [<c0c3ac54>] lr : [<fffffffa>] psr: 80010013
[413571.664296] sp : ec98dee8 ip : ed1f9290 fp : ec98defc
[413571.675997] r10: ea895308 r9 : 00000008 r8 : ed323cc0
[413571.681331] r7 : ee8b9b50 r6 : 00000000 r5 : ed2996c0 r4 : eb477000
[413571.687971] r3 : 00000000 r2 : 00000000 r1 : 00000002 r0 : c14bdc0c
[413571.694613] Flags: Nzcv IRQs on FIQs on Mode SVC_32 ISA ARM Segment user
[413571.701864] Control: 30c5387d Table: ad5bc480 DAC: 55555555

36

MMU信息

[413571.701864] Control: 30c5387d Table: ad5bc480 DAC: 55555555

```
#ifdef CONFIG_CPU_CP15
{
unsigned int ctrl;

buf[0] = '\0';
#ifdef CONFIG_CPU_CP15_MMU
{
unsigned int transbase;
asm("mrc p15, 0, %0, c2, c0\n\t"
    : "=r" (transbase));
snprintf(buf, sizeof(buf), "  Table: %08x  DAC: %08x",
transbase, domain);
}
#endif
asm("mrc p15, 0, %0, c1, c0\n" : "=r" (ctrl));

printk("Control: %08x%s\n", ctrl, buf);
}
#endif
```

void __show_regs(struct pt_regs *regs)

Arch/arm/kernel/process.c

控制寄存器，页表基地址，domain

37

Show me your code!

```
static int rpmsg_sock_release(struct socket *sock)
{
	struct sock *sk = sock->sk;
	struct rpmsg_socket *rpsk = container_of(sk, struct rpmsg_socket, sk);

	pr_debug("sk %p\n", sk);

	if (!sk)
		return 0;

	if (rpsk->unregister_rpdev)
		device_unregister(&rpsk->rpdev->dev);

	sock_put(sock->sk);

	return 0;
}
```

38

```c
static int rpmsg_sock_release(struct socket *sock)
{
    struct sock *sk = sock->sk;
    struct rpmsg_socket *rpsk = container_of(sk, struct rpmsg_socket, sk);
    struct virtproc_info *vrp = NULL;
    int ret;

    if (!sk)
        return 0;

    mutex_lock(&rpmsg_channels_lock);
    if (rpsk->unregister_rpdev) { /* Rx (bound) sockets */
        /* The bound socket's rpmsg device will be removed by rpmsg bus
         * core during recovery, but only after the published rpmsg
         * channel is removed (device registration order). The check for
         * valid vrp will ensure that rpmsg_destroy_channel will not be
         * called if the release from userspace occurs first. However,
         * the socket can be released much later than the recreated vrp
         * as well, so an additional check for a sane socket state is
         * also needed.
         */
        vrp = radix_tree_lookup(&rpmsg_vprocs, rpsk->rproc_id);
        if (vrp && sk->sk_state != RPMSG_ERROR) {
            rpsk->rpdev->ept->priv = NULL;
            mutex_unlock(&rpmsg_channels_lock);
            ret = rpmsg_destroy_channel(rpsk->rpdev);
            if (ret) {
                pr_err("rpmsg_destroy_channel failed for sk %p\n",
                       sk);
            }
            goto release;
        }
    } else { /* Tx (connected) sockets */
        if (sk->sk_state != RPMSG_ERROR)
            list_del(&rpsk->elem);
    }
    mutex_unlock(&rpmsg_channels_lock);

release:
    sock_put(sock->sk);

    return 0;
}
```

TI改过的代码

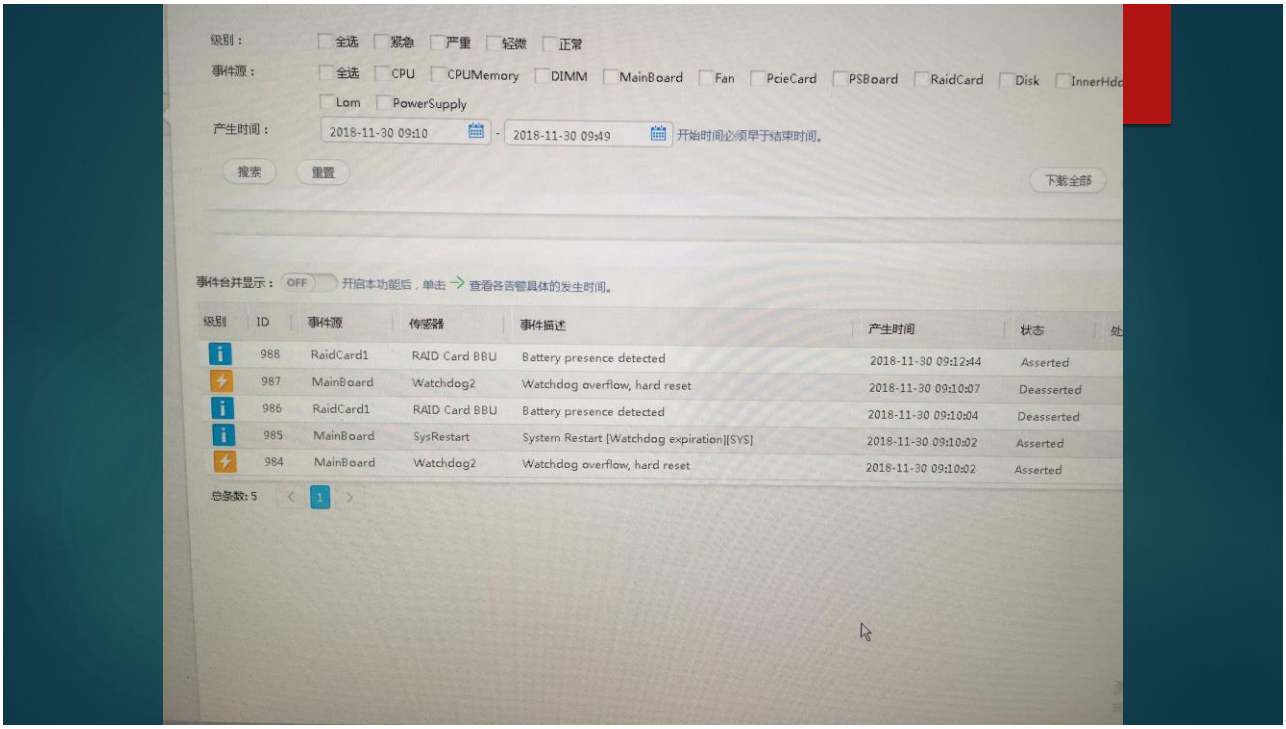
39

代码

[413571.591498] Unable to handle kernel NULL pointer dereference at virtual address 0000002c

字段unregister_rpdev在结构体里的偏移应该是2c

40



41

IPMI

Intelligent Platform Management Interface

v1.0 was announced on September 16, 1998: base specification
v1.5, published on March 1, 2001: added features including IPMI over LAN, IPMI over Serial/Modem, and LAN Alerting
v2.0, published on February 14, 2004: added features including Serial over LAN, Group Managed Systems, Enhanced Authentication, Firmware Firewall, and VLAN Support
v2.0 revision 1.1, published on February 11, 2014: amended for errata, clarifications, and addenda, plus addition of support for IPv6 Addressing
v2.0 revision 1.1 Errata 7, published on April 21, 2015: amended for errata, clarifications,addenda

42

IPMI Spec中的BMC

43

[Openipmi-developer] [PATCH] ipmi: fix BT reset for a while when cmd timeout

[Openipmi-developer] [PATCH] ipmi: fix BT reset for a while when cmd timeout
From: Xie XiuQi <xiexiuqi@hu...> - 2014-02-11 10:29:56

```
I fould a problem: when a cmd timeout and just
in that time bt->seq < 2, system will alway keep
retrying and we can't send any cmd to bmc.

the error message is like this:
[  530.908621] IPMI BT: timeout in RD_WAIT [ ] 1 retries left
[  582.661329] IPMI BT: timeout in RD_WAIT [ ]
[  582.661334] failed 2 retries, sending error response
[  582.661337] IPMI: BT reset (takes 5 secs)
[  693.335307] IPMI BT: timeout in RD_WAIT [ ]
[  693.335312] failed 2 retries, sending error response
[  693.335315] IPMI: BT reset (takes 5 secs)
[  804.825161] IPMI BT: timeout in RD_WAIT [ ]
[  804.825166] failed 2 retries, sending error response
[  804.825169] IPMI: BT reset (takes 5 secs)
```

https://sourceforge.net/p/openipmi/mailman/message/31961092/

44

OpenIpmi项目中的讨论

When BT reset, a cmd "warm reset" will be sent to bmc, but this cmd is Optional in spec(refer to ipmi-interface-spec-v2). Some machines don't support this cmd.

So, bt->init is introduced. Only during insmod, we do BT reset when response timeout to avoid system crash.

https://sourceforge.net/p/openipmi/mailman/message/31961092/

45

Xie XiuQi

Subject [PATCH 3.12 44/82] ipmi: fix timeout calculation when bmc is disconnected
Date Mon, 24 Aug 2015 11:09:04 +0200
share
From: Xie XiuQi <xiexiuqi@huawei.com>

3.12-stable review patch. If anyone has any objections, please let me know.

==========

commit e21404dc0ac7ac971c1e36274b48bb460463f4e5 upstream.

Loading ipmi_si module while bmc is disconnected, we found the timeout is longer than 5 secs. Actually it takes about 3 mins and 20 secs.(HZ=250)

https://lkml.org/lkml/2015/8/24/210

46

OS (C:) › bench › linux-4.16 › linux-4.16 › drivers › char › ipmi

名称	修改日期	类型	大小
bt-bmc.c	2018-04-02 5:20	C Source	12 KB
ipmi_bt_sm.c	2018-04-02 5:20	C Source	21 KB
ipmi_devintf.c	2018-04-02 5:20	C Source	21 KB
ipmi_dmi.c	2018-04-02 5:20	C Source	7 KB
ipmi_dmi.h	2018-04-02 5:20	C/C++ Header	1 KB
ipmi_kcs_sm.c	2018-04-02 5:20	C Source	14 KB
ipmi_msghandler.c	2018-04-02 5:20	C Source	132 KB
ipmi_powernv.c	2018-04-02 5:20	C Source	8 KB
ipmi_poweroff.c	2018-04-02 5:20	C Source	20 KB
ipmi_si.h	2018-04-02 5:20	C/C++ Header	2 KB
ipmi_si_hardcode.c	2018-04-02 5:20	C Source	5 KB
ipmi_si_hotmod.c	2018-04-02 5:20	C Source	5 KB
ipmi_si_intf.c	2018-04-02 5:20	C Source	65 KB
ipmi_si_mem_io.c	2018-04-02 5:20	C Source	4 KB
ipmi_si_parisc.c	2018-04-02 5:20	C Source	2 KB
ipmi_si_pci.c	2018-04-02 5:20	C Source	4 KB
ipmi_si_platform.c	2018-04-02 5:20	C Source	14 KB
ipmi_si_port_io.c	2018-04-02 5:20	C Source	3 KB
ipmi_si_sm.h	2018-04-02 5:20	C/C++ Header	6 KB
ipmi_smic_sm.c	2018-04-02 5:20	C Source	17 KB
ipmi_ssif.c	2018-04-02 5:20	C Source	54 KB
ipmi_watchdog.c	2018-04-02 5:20	C Source	35 KB
Kconfig	2018-04-02 5:20	文件	4 KB
Makefile	2018-04-02 5:20	文件	1 KB

47

Lister - [C:\bench\linux-4.16\linux-4.16\drivers\char\ipmi\ipmi_bt_sm.c]

File Edit Options Help

```
static enum si_sm_result error_recovery(struct si_sm_data *bt,
                                        unsigned char status,
                                        unsigned char cCode)
{
        char *reason;

        bt->timeout = bt->BT_CAP_req2rsp;

        switch (cCode) {
        case IPMI_TIMEOUT_ERR:
                reason = "timeout";
                break;
        default:
                reason = "internal error";
                break;
        }

        printk(KERN_WARNING "IPMI BT: %s in %s %s ",    /* open-ended line */
                reason, STATE2TXT, STATUS2TXT);

        /*
         * Per the IPMI spec, retries are based on the sequence number
         * known only to this module, so manage a restart here.
         */
        (bt->error_retries)++;
        if (bt->error_retries < bt->BT_CAP_retries) {
                printk("%d retries left\n",
                       bt->BT_CAP_retries - bt->error_retries);
                bt->state = BT_STATE_RESTART;
                return SI_SM_CALL_WITHOUT_DELAY;
        }

        printk(KERN_WARNING "failed %d retries, sending error response\n",
               bt->BT_CAP_retries);
        if (!bt->nonzero_status)
                printk(KERN_ERR "IPMI BT: stuck, try power cycle\n");

        /* this is most likely during insmod */
        else if (bt->seq <= (unsigned char)(bt->BT_CAP_retries & 0xFF)) {
                printk(KERN_WARNING "IPMI: BT reset (takes 5 secs)\n");
                bt->state = BT_STATE_RESET1;
                return SI_SM_CALL_WITHOUT_DELAY;
        }
```

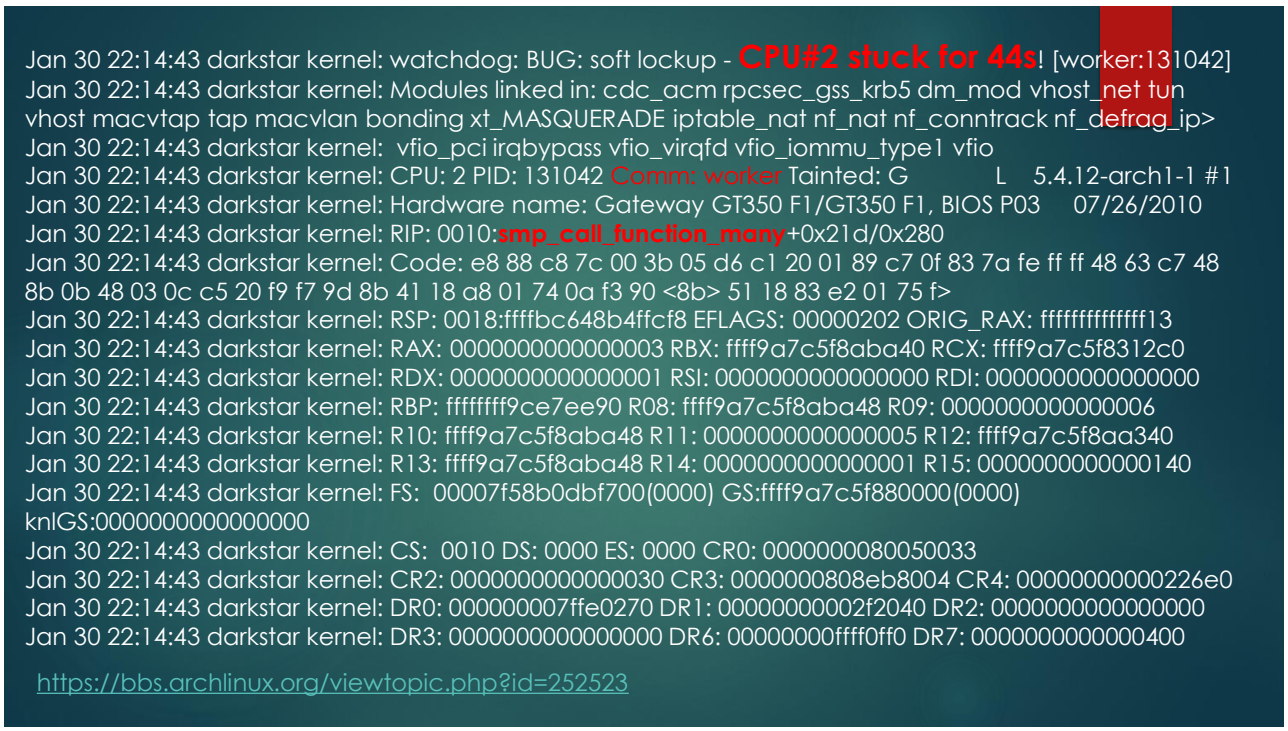
I: IPMI BT: timeout in RD_WAIT [ ] 1 retries left
I: IPMI BT: timeout in RD_WAIT [ ] 1 retries left
I: IPMI BT: timeout in RD_WAIT [ ]
I: failed 2 retries, sending error response
I: IPMI BT: timeout in RD_WAIT [ ] 1 retries left
I: IPMI BT: timeout in RD_WAIT [ ]
I: failed 2 retries, sending error response
bin/bmc-watchdog[4341]: fiid_obj_get: 'timer_state': data

Block Transfer (BT) Interface

48


Products & Services › Knowledgebase › RHEL6: System resets after "/usr/sbin/bmc-watchdog: fiid_obj_get: 'timer_state': data not available"...
RHEL6: System resets after "/usr/sbin/bmc-watchdog: fiid_obj_get: 'timer_state': data not available".
SOLUTION VERIFIED - Updated April 9 2018 at 6:54 AM - English
Issue
Following error was logged before system reboots unexpectedly.
Raw
Mar 15 04:16:37 localhost /usr/sbin/bmc-watchdog[5003]: fiid_obj_get: 'timer_state': data not available
https://access.redhat.com/solutions/3403821

49


Jan 30 22:14:43 darkstar kernel: watchdog: BUG: soft lockup - CPU#2 stuck for 44s! [worker:131042]
Jan 30 22:14:43 darkstar kernel: Modules linked in: cdc_acm rpcsec_gss_krb5 dm_mod vhost_net tun vhost macvtap tap macvlan bonding xt_MASQUERADE iptable_nat nf_nat nf_conntrack nf_defrag_ip>
Jan 30 22:14:43 darkstar kernel: vfio_pci irqbypass vfio_virqfd vfio_iommu_type1 vfio
Jan 30 22:14:43 darkstar kernel: CPU: 2 PID: 131042 Comm: worker Tainted: G L 5.4.12-arch1-1 #1
Jan 30 22:14:43 darkstar kernel: Hardware name: Gateway GT350 F1/GT350 F1, BIOS P03 07/26/2010
Jan 30 22:14:43 darkstar kernel: RIP: 0010:smp_call_function_many+0x21d/0x280
Jan 30 22:14:43 darkstar kernel: Code: e8 88 c8 7c 00 3b 05 d6 c1 20 01 89 c7 0f 83 7a fe ff ff 48 63 c7 48 8b 0b 48 03 0c c5 20 f9 f7 9d 8b 41 18 a8 01 74 0a f3 90 <8b> 51 18 83 e2 01 75 f>
Jan 30 22:14:43 darkstar kernel: RSP: 0018:ffffbc648b4ffcf8 EFLAGS: 00000202 ORIG_RAX: ffffffffffffff13
Jan 30 22:14:43 darkstar kernel: RAX: 0000000000000003 RBX: ffff9a7c5f8aba40 RCX: ffff9a7c5f8312c0
Jan 30 22:14:43 darkstar kernel: RDX: 0000000000000001 RSI: 0000000000000000 RDI: 0000000000000000
Jan 30 22:14:43 darkstar kernel: RBP: ffffffff9ce7ee90 R08: ffff9a7c5f8aba48 R09: 0000000000000006
Jan 30 22:14:43 darkstar kernel: R10: ffff9a7c5f8aba48 R11: 0000000000000005 R12: ffff9a7c5f8aa340
Jan 30 22:14:43 darkstar kernel: R13: ffff9a7c5f8aba48 R14: 0000000000000001 R15: 0000000000000140
Jan 30 22:14:43 darkstar kernel: FS: 00007f58b0dbf700(0000) GS:ffff9a7c5f880000(0000) knlGS:0000000000000000
Jan 30 22:14:43 darkstar kernel: CS: 0010 DS: 0000 ES: 0000 CR0: 0000000080050033
Jan 30 22:14:43 darkstar kernel: CR2: 0000000000000030 CR3: 0000000808eb8004 CR4: 00000000000226e0
Jan 30 22:14:43 darkstar kernel: DR0: 000000007ffe0270 DR1: 00000000002f2040 DR2: 0000000000000000
Jan 30 22:14:43 darkstar kernel: DR3: 0000000000000000 DR6: 00000000ffff0ff0 DR7: 0000000000000400
https://bbs.archlinux.org/viewtopic.php?id=252523

50

```
Jan 30 22:14:43 darkstar kernel: Call Trace:
Jan 30 22:14:43 darkstar kernel:  flush_tlb_mm_range+0xed/0x150
Jan 30 22:14:43 darkstar kernel:  tlb_flush_mmu+0xa4/0x160
Jan 30 22:14:43 darkstar kernel:  tlb_finish_mmu+0x3d/0x70
Jan 30 22:14:43 darkstar kernel:  unmap_region+0xf4/0x130
Jan 30 22:14:43 darkstar kernel:  __do_munmap+0x255/0x4c0
Jan 30 22:14:43 darkstar kernel:  __vm_munmap+0x67/0xb0
Jan 30 22:14:43 darkstar kernel:  __x64_sys_munmap+0x28/0x30
Jan 30 22:14:43 darkstar kernel:  do_syscall_64+0x4e/0x140
Jan 30 22:14:43 darkstar kernel:  entry_SYSCALL_64_after_hwframe+0x44/0xa9
Jan 30 22:14:43 darkstar kernel: RIP: 0033:0x7f5c028d10db
Jan 30 22:14:43 darkstar kernel: Code: 8b 15 a9 5d 0c 00 f7 d8 64 89 02 48 c7 c0 ff ff ff ff eb 89 66 2e 0f 1f
84 00 00 00 00 00 90 f3 0f 1e fa b8 0b 00 00 00 0f 05 <48> 3d 01 f0 ff ff 73 0>
Jan 30 22:14:43 darkstar kernel: RSP: 002b:00007f58b0dbd038 EFLAGS: 00000206 ORIG_RAX:
000000000000000b
Jan 30 22:14:43 darkstar kernel: RAX: ffffffffffffffda RBX: 00007f58da31e9c0 RCX: 00007f5c028d10db
Jan 30 22:14:43 darkstar kernel: RDX: 0000000000000000 RSI: 0000000000801000 RDI: 00007f58d9b1e000
Jan 30 22:14:43 darkstar kernel: RBP: 00007f58b4dc79c0 R08: 0000000000000000 R09: 0000000000000001
Jan 30 22:14:43 darkstar kernel: R10: 00007f58d76000c0 R11: 0000000000000206 R12: 00007f5c029bb020
Jan 30 22:14:43 darkstar kernel: R13: 0000000002800000 R14: 00007f58b0dbd140 R15: 00007f58b0dbf700
```

51

smp_call_function_many

```
/**
 * smp_call_function_many(): Run a function on a set of other CPUs.
 * @mask: The set of cpus to run on (only runs on online subset).
 * @func: The function to run. This must be fast and non-blocking.
 * @info: An arbitrary pointer to pass to the function.
 * @wait: If true, wait (atomically) until function has completed
 *        on other CPUs.
 *
 * If @wait is true, then returns once @func has returned.
 *
 * You must not call this function with disabled interrupts or from a
 * hardware interrupt handler or from a bottom half handler. Preemption
 * must be disabled when calling this function.
 */
```

Kernel/smp.c

52

循环等待

```
/* Send a message to all CPUs in the map */
arch_send_call_function_ipi_mask(cfd->cpumask_ipi);

if (wait) {
    for_each_cpu(cpu, cfd->cpumask) {
        call_single_data_t *csd;

        csd = per_cpu_ptr(cfd->csd, cpu);
        csd_lock_wait(csd);
    }
}
```

call_single_data_t

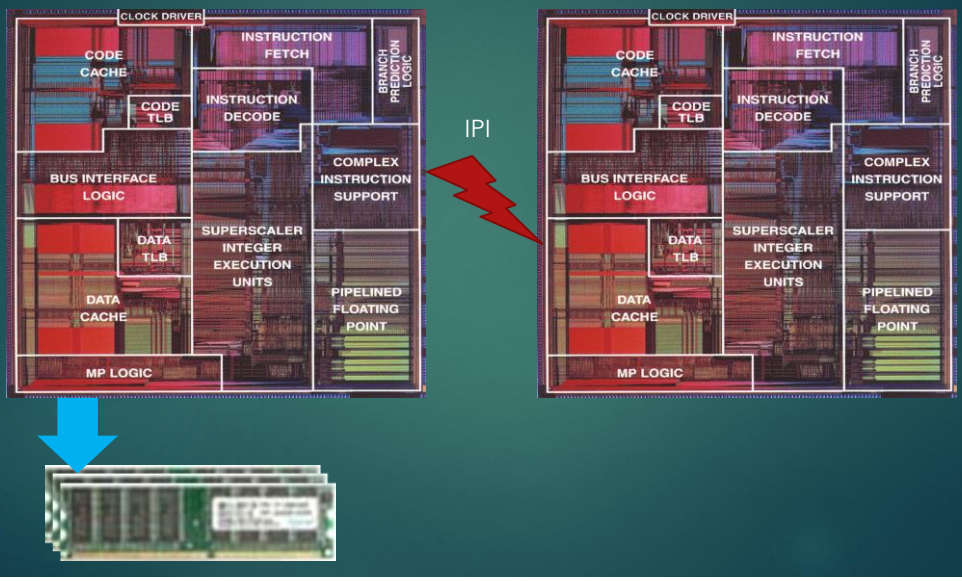
53

```
/**
 * smp_cond_load_relaxed() - (Spin) wait for cond with no ordering guarantees
 * @ptr: pointer to the variable to wait on
 * @cond: boolean expression to wait for
 *
 * Equivalent to using READ_ONCE() on the condition variable.
 *
 * Due to C lacking lambda expressions we load the value of *ptr into a
 * pre-named variable @VAL to be used in @cond.
 */
#ifndef smp_cond_load_relaxed
#define smp_cond_load_relaxed(ptr, cond_expr) ({        \
    typeof(ptr) __PTR = (ptr);              \
    typeof(*ptr) VAL;                   \
    for (;;) {                          \
        VAL = READ_ONCE(*__PTR);            \
        if (cond_expr)                  \
            break;                      \
        cpu_relax();                    \
    }                               \
    VAL;                                \
})
#endif
```

忙等

54

TLB同步



55

flush_tlb_mm_range

arch > x86 > mm > C tlb.c > ...

```
728  void flush_tlb_mm_range(struct mm_struct *mm, unsigned long start,
729                          unsigned long end, unsigned int stride_shift,
730                          bool freed_tables)
731  {
732      int cpu;
```

```
754
755      if (mm == this_cpu_read(cpu_tlbstate.loaded_mm)) {
756          VM_WARN_ON(irqs_disabled());
757          local_irq_disable();
758          flush_tlb_func_local(&info, TLB_LOCAL_MM_SHOOTDOWN);
759          local_irq_enable();
760      }
761
762      if (cpumask_any_but(mm_cpumask(mm), cpu) < nr_cpu_ids)
763          flush_tlb_others(mm_cpumask(mm), &info);
764
765      put_cpu();
766  }
```

本地冲洗

通知伙伴

56

2020-01-30 20:26:00
Gruntz
Member From: Haskovo, Bulgaria
Registered: 2007-08-31 Posts: 285
Hello all,

I have a supemicro motherboard and two xeon x5650. I have 64GB of ram and several VMs on it.

I have one windows10, that i use for gaming (with gpu pass-through)
I have another, with archlinux for work (soft dev, testings... i use this one as my main desktop.GPU pass-through here too.)
I have 3-4-5 more, for database server, gitlab server, stuff like that.

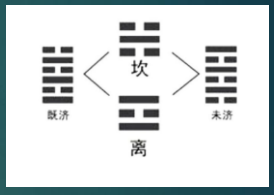
From time to time my host machine crashes. It has a lot of "watchdog: BUG: soft lockup - CPU#4 stuck for 45s! [worker:131043]" messages.
Each time different processor. After it appears, the computer slowly becomes unresponsive, and eventually hangs completely.

Do you have any clue what this could be? I read about the problem. It appears when a cpu is stuck ot task for a long time or something like that.
But that is normal for VMs. Can I work around it?

Best regards.

https://bbs.archlinux.org/viewtopic.php?id=252523

57



未济

58

切问而近思

欢迎关注格友公众号

59

调整printk输出级别

```
[   26.764092] c1707688 c19d77dc
[   26.764384] buff=memblock_isolate_range+0x0/0x21d
[   26.764835] c170768c c19d79f9
[   26.765131] too bad, 40351 not found
[   26.767323] BUG: unable to handle kernel NULL pointer dereference at 00000004
[   26.768016] IP: [<c155b5b7>] kallsym_init+0x97/0x260
[   26.768499] *pdpt = 0000000000000000 *pde = f000ff53f000ff53
[   26.769065] Oops: 0000 [#1] SMP
[   26.769402] Modules linked in:
[   26.769720] CPU: 0 PID: 1 Comm: swapper/0 Not tainted 3.11.0gedu #10
[   26.770343] Hardware name: innotek GmbH VirtualBox/VirtualBox, BIOS VirtualBox 12/01/2006
[   26.771276] task: df480000 ti: df44a000 task.ti: df44a000
[   26.771853] EIP: 0060:[<c155b5b7>] EFLAGS: 00010292 CPU: 0
[   26.772438] EIP is at kallsym_init+0x97/0x260
[   26.772920] EAX: 00000000 EBX: c16e000c ECX: 0000a1f2 EDX: 00000092
[   26.773587] ESI: 00000105 EDI: c1a39cd4 EBP: df44beac ESP: df44be90
[   26.774256]  DS: 007b ES: 007b FS: 00d8 GS: 00e0 SS: 0068
[   26.774803] CR0: 8005003b CR2: 00000004 CR3: 01a62000 CR4: 000406f0
[   26.775436] DR0: 00000000 DR1: 00000000 DR2: 00000000 DR3: 00000000
[   26.776024] DR6: fffe0ff0 DR7: 00000400
```

60

还崩

```
[   34.538600] sym mark=c17f626c, token=c17f672c, index=c17f6ab8
[   34.555984] BUG: unable to handle kernel NULL pointer dereference at   (null)
[   34.557236] IP: [<c1643951>] name_2_addr+0x1b2/0x213
[   34.557779] *pdpt = 0000000000000000 *pde = f000ff53f000ff53
[   34.558381] Oops: 0000 [#1] SMP
[   34.558718] Modules linked in:
[   34.559029] CPU: 0 PID: 1 Comm: swapper/0 Not tainted 3.11.0gedu #12
[   34.559636] Hardware name: innotek GmbH VirtualBox/VirtualBox, BIOS VirtualBox 12/01/2006
[   34.560399] task: df480000 ti: df44a000 task.ti: df44a000
[   34.560914] EIP: 0060:[<c1643951>] EFLAGS: 00010246 CPU: 0
[   34.561433] EIP is at name_2_addr+0x1b2/0x213
[   34.561855] EAX: df44be00 EBX: 00000009 ECX: 00000002 EDX: c186566e
[   34.562545] ESI: 00000000 EDI: c17f6ab8 EBP: df44be88 ESP: df44bd64
[   34.563198]  DS: 007b ES: 007b FS: 00d8 GS: 00e0 SS: 0068
[   34.563712] CR0: 8005003b CR2: 00000000 CR3: 01a62000 CR4: 000406f0
[   34.564308] DR0: 00000000 DR1: 00000000 DR2: 00000000 DR3: 00000000
[   34.564893] DR6: fffe0ff0 DR7: 00000400
[   34.565256] Stack:
[   34.565450]  c186566e c17f6260 00012fa6 c17f672c c17f6269 ffffffff 5f420000 6b72625f
[   34.566276]  6d696c5f 00007469 00000000 00000000 00000000 00000000 00000000 00000000
[   34.570804]  00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
[   34.571763] Call Trace:
[   34.572067]  [<c155b68c>] kallsym_init+0x16c/0x260
[   34.572595]  [<c19ceb1e>] fedcore_init+0x15/0x350
[   34.573137]  [<c13fd5bb>] ? __class_create+0x4b/0x70
[   34.573661]  [<c1551769>] ? create_extcon_class.part.2+0x19/0x30
[   34.574296]  [<c19ceb07>] ? extcon_class_init+0x13/0x15
[   34.574854]  [<c10020fc>] do_one_initcall+0xdc/0x1b0
[   34.575380]  [<c11c5583>] ? __proc_create+0xa3/0xe0
[   34.575843]  [<c19ceb09>] ? extcon_class_init+0x15/0x15
```

61

隐藏的static函数

```
static unsigned int mod_name_2_addr(char *name)
{
    struct module *cur_mod;
    char mod_name[50];
    int size = 0;
    unsigned int ret = 0;

    kmemset(mod_name,0,50);
    if ((size = get_mod_name(name,'!')) ) {
        strncp(mod_name,name,size);
        if ((cur_mod = get_module(mod_name)) != NULL)
            ret = mod_find_symname(cur_mod, name+size+1);

    } else {
        list_for_each_entry_rcu(cur_mod, mod_header, list)
            if ((ret = mod_find_symname(cur_mod, name)) != 0)
                break;
    }

    return ret;
}
```

```
fed@fed-VirtualBox:~/work/src/linux-3.11.5/drivers/fedcore$ objdump -t fedcore.o | grep name
0000000c l     O .rodata        00000008 fedcore_name
00000038 l     F .text.unlikely 00000073 mod_find_symname
000000ab l     F .text.unlikely 00000213 name_2_addr
000002be l     F .text.unlikely 0000003e is_funname_char
00013308 g     O .bss   00000004 kallsyms_name
```

62

去掉static后

[65.512800] sym num=77735, addr=c16c22d8, name=c170e178
[65.514034] sym mark=c17f627c, token=c17f673c, index=c17f6ac8
[65.532323] BUG: unable to handle kernel NULL pointer dereference at (null)
[65.537343] IP: [<c155ad9e>] **mod_name_2_addr**+0xde/0x140
[65.541007] *pdpt = 0000000000000000 *pde = f000ff53f000ff53
[65.542054] Oops: 0000 [#1] SMP
[65.542789] Modules linked in:
[65.543541] CPU: 0 PID: 1 Comm: swapper/0 Not tainted 3.11.0gedu #13
[65.544632] Hardware name: innotek GmbH VirtualBox/VirtualBox, BIOS VirtualBox 12/01/2006
[65.546029] task: df480000 ti: df44a000 task.ti: df44a000
[65.546942] EIP: 0060:[<c155ad9e>] EFLAGS: 00010246 CPU: 0
[65.547866] EIP is at mod_name_2_addr+0xde/0x140
[65.549710] EAX: 00000010 EBX: c186567e ECX: 00000063 EDX: 00000000
[65.550773] ESI: c17f627a EDI: 00000000 EBP: df44bd94 ESP: df44bd48
[65.552132] DS: 007b ES: 007b FS: 00d8 GS: 00e0 SS: 0068
[65.553029] CR0: 8005003b CR2: 00000000 CR3: 01a62000 CR4: 000406f0
[65.554137] DR0: 00000000 DR1: 00000000 DR2: 00000000 DR3: 00000000
[65.555134] DR6: fffe0ff0 DR7: 00000400
[65.555831] Stack:

63