调试

重打包rootfs

```
find . | cpio -o --format=newc > ../rootfs.img

cpio -idmv < rootfs.cpio 解包
```

启动

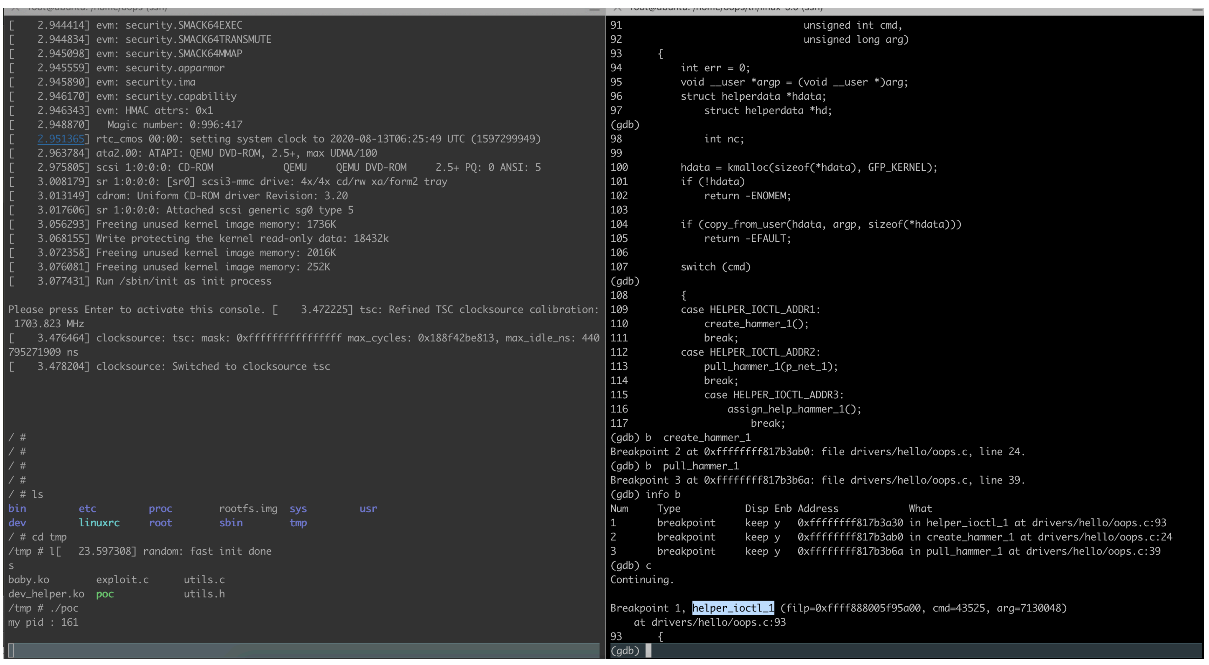
```
qemu-system-x86_64 \
-kernel /home/oops/th/linux-5.0/arch/x86_64/boot/bzImage \
-initrd /home/oops/th/busybox-1.31.0/rootfs.img \
-append "console=ttyS0 nokaslr root=/dev/ram rdinit=/sbin/init" \
-cpu kvm64,+smep,+smap \
-nographic \
-gdb tcp::1234
```

编译

```
gcc exploit.c -o poc -static -w

更新gdb
sudo add-apt-repository ppa:ubuntu-toolchain-r/test
sudo apt-get update
sudo apt-get -y --force-yes install gdb
gdb -v
sudo add-apt-repository --remove ppa:ubuntu-toolchain-r/test
sudo apt-get update
```

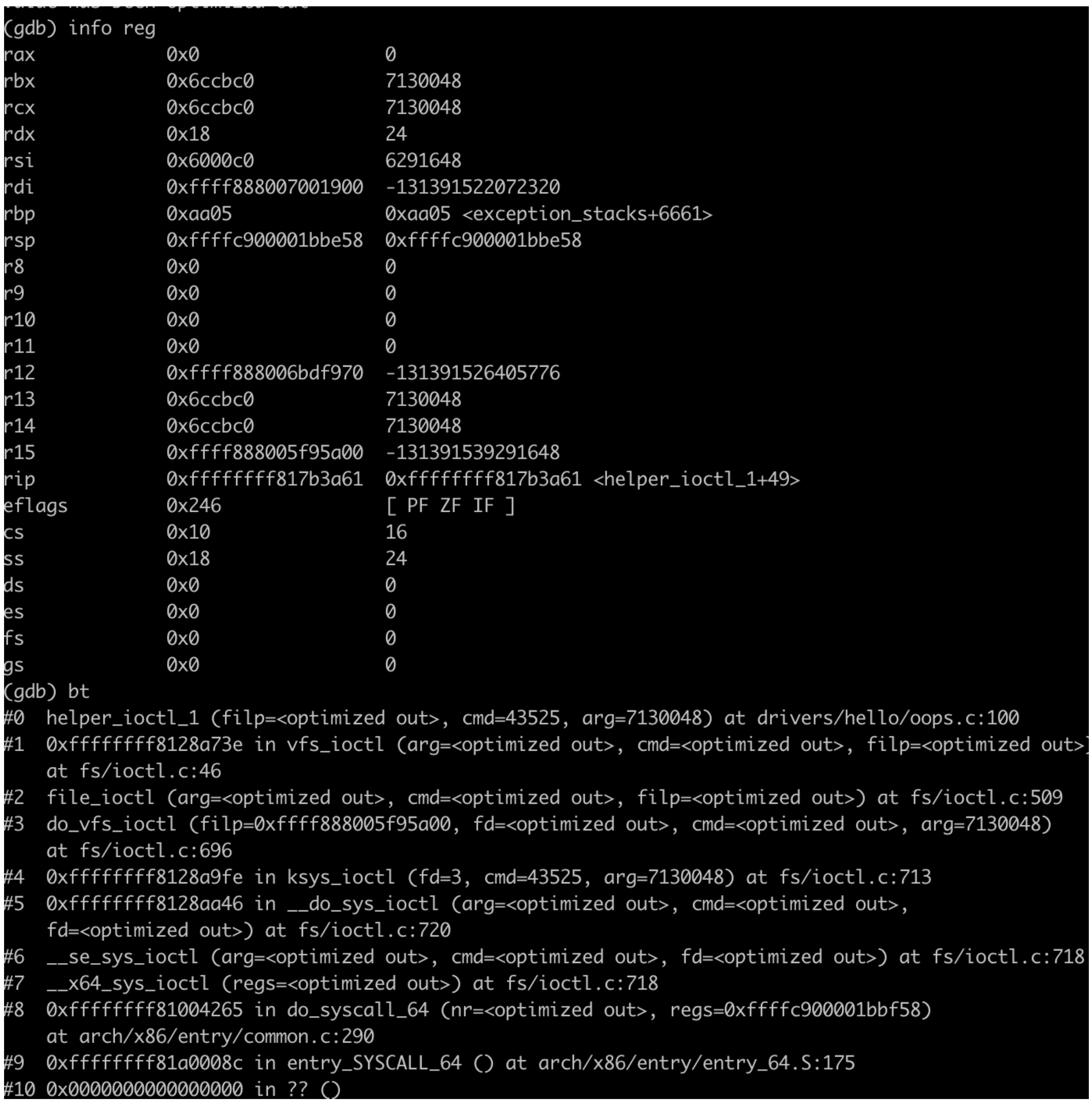
调试

```
gdb vmlinux
target remote :1234

b helper_ioctl_1
```



```
[    2.944414] evm: security.SMACK64EXEC
[    2.944834] evm: security.SMACK64TRANSMUTE
[    2.945098] evm: security.SMACK64MMAP
[    2.945559] evm: security.apparmor
[    2.945890] evm: security.ima
[    2.946170] evm: security.capability
[    2.946343] evm: HMAC attrs: 0x1
[    2.948870]   Magic number: 0:996:417
[    2.951365] rtc_cmos 00:00: setting system clock to 2020-08-13T06:25:49 UTC (1597299949)
[    2.963784] ata2.00: ATAPI: QEMU DVD-ROM, 2.5+, max UDMA/100
[    2.975805] scsi 1:0:0:0: CD-ROM            QEMU     QEMU DVD-ROM     2.5+ PQ: 0 ANSI: 5
[    3.008179] sr 1:0:0:0: [sr0] scsi3-mmc drive: 4x/4x cd/rw xa/form2 tray
[    3.013149] cdrom: Uniform CD-ROM driver Revision: 3.20
[    3.017606] sr 1:0:0:0: Attached scsi generic sg0 type 5
[    3.056293] Freeing unused kernel image memory: 1736K
[    3.068155] Write protecting the kernel read-only data: 18432k
[    3.072358] Freeing unused kernel image memory: 2016K
[    3.076081] Freeing unused kernel image memory: 252K
[    3.077431] Run /sbin/init as init process

Please press Enter to activate this console. [    3.472225] tsc: Refined TSC clocksource calibration:
 1703.823 MHz
[    3.476464] clocksource: tsc: mask: 0xffffffffffffffff max_cycles: 0x188f42be813, max_idle_ns: 440
795271909 ns
[    3.478204] clocksource: Switched to clocksource tsc




/ #
/ #
/ #
/ #
/ # ls
bin         etc         proc        rootfs.img  sys         usr
dev         linuxrc     root        sbin        tmp
/ # cd tmp
/tmp # l[   23.597308] random: fast init done
s
baby.ko        exploit.c      utils.c
dev_helper.ko  poc            utils.h
/tmp # ./poc
my pid : 161
```

```
91                                  unsigned int cmd,
92                                  unsigned long arg)
93      {
94          int err = 0;
95          void __user *argp = (void __user *)arg;
96          struct helperdata *hdata;
97              struct helperdata *hd;
(gdb)
98              int nc;
99
100         hdata = kmalloc(sizeof(*hdata), GFP_KERNEL);
101         if (!hdata)
102             return -ENOMEM;
103
104         if (copy_from_user(hdata, argp, sizeof(*hdata)))
105             return -EFAULT;
106
107         switch (cmd)
(gdb)
108         {
109         case HELPER_IOCTL_ADDR1:
110             create_hammer_1();
111             break;
112         case HELPER_IOCTL_ADDR2:
113             pull_hammer_1(p_net_1);
114             break;
115             case HELPER_IOCTL_ADDR3:
116                 assign_help_hammer_1();
117                     break;
(gdb) b  create_hammer_1
Breakpoint 2 at 0xffffffff817b3ab0: file drivers/hello/oops.c, line 24.
(gdb) b  pull_hammer_1
Breakpoint 3 at 0xffffffff817b3b6a: file drivers/hello/oops.c, line 39.
(gdb) info b
Num     Type           Disp Enb Address            What
1       breakpoint     keep y   0xffffffff817b3a30 in helper_ioctl_1 at drivers/hello/oops.c:93
2       breakpoint     keep y   0xffffffff817b3ab0 in create_hammer_1 at drivers/hello/oops.c:24
3       breakpoint     keep y   0xffffffff817b3b6a in pull_hammer_1 at drivers/hello/oops.c:39
(gdb) c
Continuing.

Breakpoint 1, helper_ioctl_1 (filp=0xffff888005f95a00, cmd=43525, arg=7130048)
    at drivers/hello/oops.c:93
93      {
(gdb)
```



```
(gdb) info reg
rax            0x0                 0
rbx            0x6ccbc0            7130048
rcx            0x6ccbc0            7130048
rdx            0x18                24
rsi            0x6000c0            6291648
rdi            0xffff888007001900  -131391522072320
rbp            0xaa05              0xaa05 <exception_stacks+6661>
rsp            0xffffc900001bbe58  0xffffc900001bbe58
r8             0x0                 0
r9             0x0                 0
r10            0x0                 0
r11            0x0                 0
r12            0xffff888006bdf970  -131391526405776
r13            0x6ccbc0            7130048
r14            0x6ccbc0            7130048
r15            0xffff888005f95a00  -131391539291648
rip            0xffffffff817b3a61  0xffffffff817b3a61 <helper_ioctl_1+49>
eflags         0x246               [ PF ZF IF ]
cs             0x10                16
ss             0x18                24
ds             0x0                 0
es             0x0                 0
fs             0x0                 0
gs             0x0                 0
(gdb) bt
#0  helper_ioctl_1 (filp=<optimized out>, cmd=43525, arg=7130048) at drivers/hello/oops.c:100
#1  0xffffffff8128a73e in vfs_ioctl (arg=<optimized out>, cmd=<optimized out>, filp=<optimized out>)
    at fs/ioctl.c:46
#2  file_ioctl (arg=<optimized out>, cmd=<optimized out>, filp=<optimized out>) at fs/ioctl.c:509
#3  do_vfs_ioctl (filp=0xffff888005f95a00, fd=<optimized out>, cmd=<optimized out>, arg=7130048)
    at fs/ioctl.c:696
#4  0xffffffff8128a9fe in ksys_ioctl (fd=3, cmd=43525, arg=7130048) at fs/ioctl.c:713
#5  0xffffffff8128aa46 in __do_sys_ioctl (arg=<optimized out>, cmd=<optimized out>,
    fd=<optimized out>) at fs/ioctl.c:720
#6  __se_sys_ioctl (arg=<optimized out>, cmd=<optimized out>, fd=<optimized out>) at fs/ioctl.c:718
#7  __x64_sys_ioctl (regs=<optimized out>) at fs/ioctl.c:718
#8  0xffffffff81004265 in do_syscall_64 (nr=<optimized out>, regs=0xffffc900001bbf58)
    at arch/x86/entry/common.c:290
#9  0xffffffff81a0008c in entry_SYSCALL_64 () at arch/x86/entry/entry_64.S:175
#10 0x0000000000000000 in ?? ()
```

命令	功能
finish	运行程序，直到当前函数完成返回。并打印函数返回时的堆栈地址和返回值及参数值等信息。
u	当你厌倦了在一个循环体内单步跟踪时，这个命令可以运行程序直到退出循环体。
bt	打印当前的函数调用栈的所有信息
info frame info f	这个命令会打印出更为详细的当前栈层的信息，只不过，大多数都是运行时的内内地址。

info args	打印出当前函数的参数名及其值。
info locals	打印出当前函数中所有局部变量及其值
show convenience	该命令查看当前所设置的所有的环境变量，环境变量可以通过set命令设置。

- list

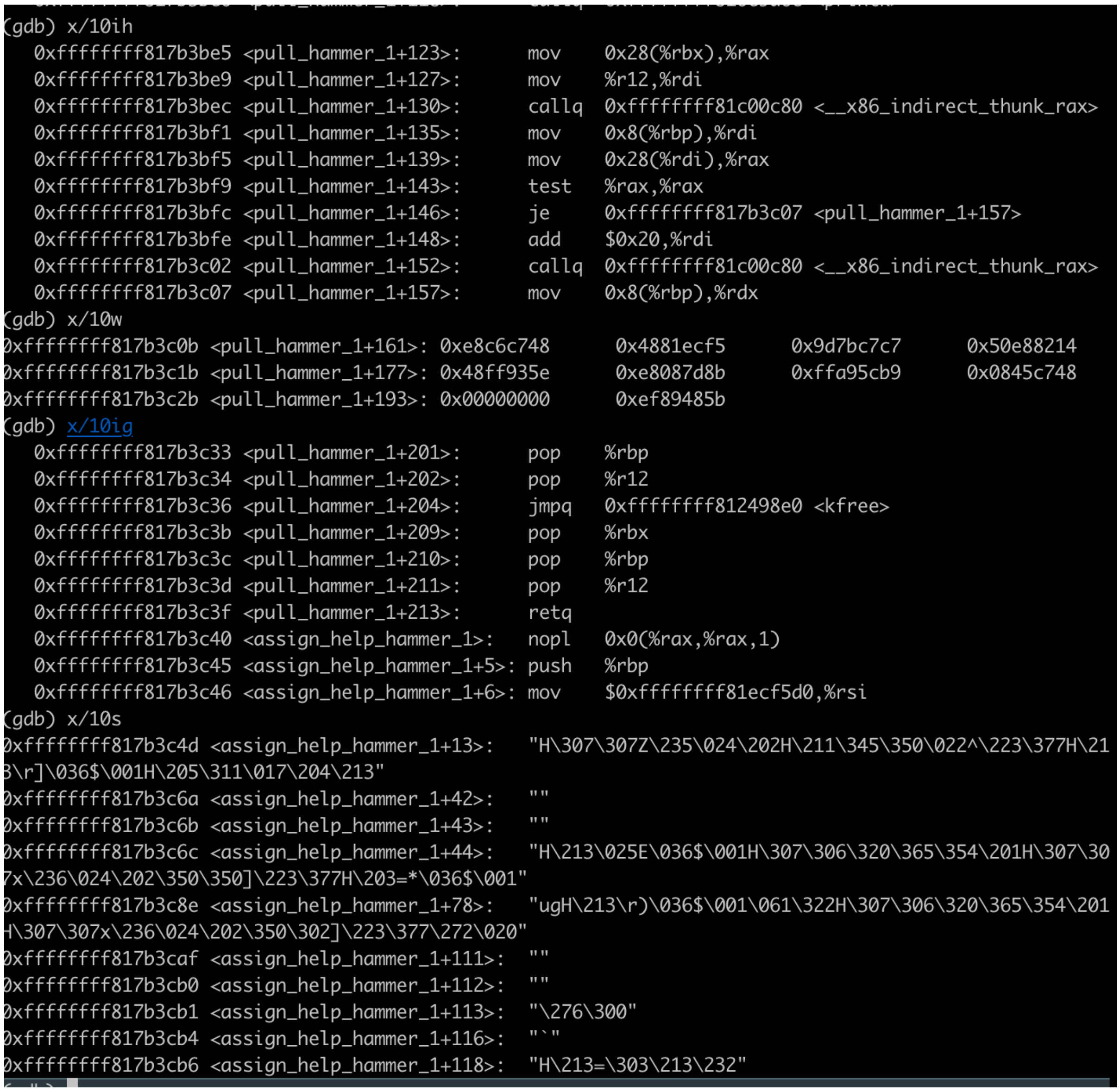
```
list <linenum>
list <first>, <last>
list , <last>
list <function>
set listsize <count>

显示程序第linenum行的周围的源程序。
显示从first行到last行之间的源代码。
set命令设置一次显示源代码的行数
```

- print
- x/nfu addr

常用：x /64 addr

使用x命令可以按格式查看绝对地址的内存信息，内存信息按NFU格式打印到控制台。

nfu 是格式表达式:
n：查看内存的个数
f：显示格式，显示格式可以是i（instruction），x（16进制）。
u：按什么数据类型显示：
u：可以是下面几种类型：
b：1字节显示（Bytes）。
h：2字节显示（Halfwords）。
w：4字节显示（Words），默认显示类型为4字节显示。
g：8字节显示（Giant words）。



```
(gdb) x/10ih
   0xffffffff817b3be5 <pull_hammer_1+123>:      mov    0x28(%rbx),%rax
   0xffffffff817b3be9 <pull_hammer_1+127>:      mov    %r12,%rdi
   0xffffffff817b3bec <pull_hammer_1+130>:      callq  0xffffffff81c00c80 <__x86_indirect_thunk_rax>
   0xffffffff817b3bf1 <pull_hammer_1+135>:      mov    0x8(%rbp),%rdi
   0xffffffff817b3bf5 <pull_hammer_1+139>:      mov    0x28(%rdi),%rax
   0xffffffff817b3bf9 <pull_hammer_1+143>:      test   %rax,%rax
   0xffffffff817b3bfc <pull_hammer_1+146>:      je     0xffffffff817b3c07 <pull_hammer_1+157>
   0xffffffff817b3bfe <pull_hammer_1+148>:      add    $0x20,%rdi
   0xffffffff817b3c02 <pull_hammer_1+152>:      callq  0xffffffff81c00c80 <__x86_indirect_thunk_rax>
   0xffffffff817b3c07 <pull_hammer_1+157>:      mov    0x8(%rbp),%rdx
(gdb) x/10w
0xffffffff817b3c0b <pull_hammer_1+161>: 0xe8c6c748      0x4881ecf5      0x9d7bc7c7      0x50e88214
0xffffffff817b3c1b <pull_hammer_1+177>: 0x48ff935e      0xe8087d8b      0xffa95cb9      0x0845c748
0xffffffff817b3c2b <pull_hammer_1+193>: 0x00000000      0xef89485b
(gdb) x/10ig
   0xffffffff817b3c33 <pull_hammer_1+201>:      pop    %rbp
   0xffffffff817b3c34 <pull_hammer_1+202>:      pop    %r12
   0xffffffff817b3c36 <pull_hammer_1+204>:      jmpq   0xffffffff812498e0 <kfree>
   0xffffffff817b3c3b <pull_hammer_1+209>:      pop    %rbx
   0xffffffff817b3c3c <pull_hammer_1+210>:      pop    %rbp
   0xffffffff817b3c3d <pull_hammer_1+211>:      pop    %r12
   0xffffffff817b3c3f <pull_hammer_1+213>:      retq
   0xffffffff817b3c40 <assign_help_hammer_1>:   nopl   0x0(%rax,%rax,1)
   0xffffffff817b3c45 <assign_help_hammer_1+5>: push   %rbp
   0xffffffff817b3c46 <assign_help_hammer_1+6>: mov    $0xffffffff81ecf5d0,%rsi
(gdb) x/10s
0xffffffff817b3c4d <assign_help_hammer_1+13>:   "H\307\307Z\235\024\202H\211\345\350\022^\223\377H\21
3\r]\036$\001H\205\311\017\204\213"
0xffffffff817b3c6a <assign_help_hammer_1+42>:   ""
0xffffffff817b3c6b <assign_help_hammer_1+43>:   ""
0xffffffff817b3c6c <assign_help_hammer_1+44>:   "H\213\025E\036$\001H\307\306\320\365\354\201H\307\30
7x\236\024\202\350\350]\223\377H\203=*\036$\001"
0xffffffff817b3c8e <assign_help_hammer_1+78>:   "ugH\213\r)\036$\001\061\322H\307\306\320\365\354\201
H\307\307x\236\024\202\350\302]\223\377\272\020"
0xffffffff817b3caf <assign_help_hammer_1+111>:  ""
0xffffffff817b3cb0 <assign_help_hammer_1+112>:  ""
0xffffffff817b3cb1 <assign_help_hammer_1+113>:  "\276\300"
0xffffffff817b3cb4 <assign_help_hammer_1+116>:  "`"
0xffffffff817b3cb6 <assign_help_hammer_1+118>:  "H\213=\303\213\232"
```

set print pretty on

display/i $pc

display/3i $pc

set disassemble-next on

set print array

set print array on

打开数组显示，打开后当数组显示时，每个元素占一行，假如不打开的话，每个元素则以逗号分隔。这个选项默认是关闭的。

set print array off

p/x $rax

b *main和b main 的区别

p *array@len

@的左边是数组的首地址的值，也就是变量array所指向的内容，右边
则是数据的长度，其保存在变量len中，其输出结果，大约是下面这
个样子的：

```
(gdb) p *array@len
$1 = {2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40}
```

如果是静态数组的话，可以直接用print数组名，就可以显示数组中所有数据的内容了。

```
x 按十六进制格式显示变量。
d 按十进制格式显示变量。
u 按十六进制格式显示无符号整型。
o 按八进制格式显示变量。
t 按二进制格式显示变量。
a 按十六进制格式显示变量。
c 按字符格式显示变量。
f 按浮点数格式显示变量

(gdb) p i
$21 = 101


(gdb) p/a i
$22 = 0x65

(gdb) p/c i
$23 = 101 'e'

(gdb) p/f i
$24 = 1.41531145e-43

(gdb) p/x i
$25 = 0x65

(gdb) p/t i
$26 = 1100101
```

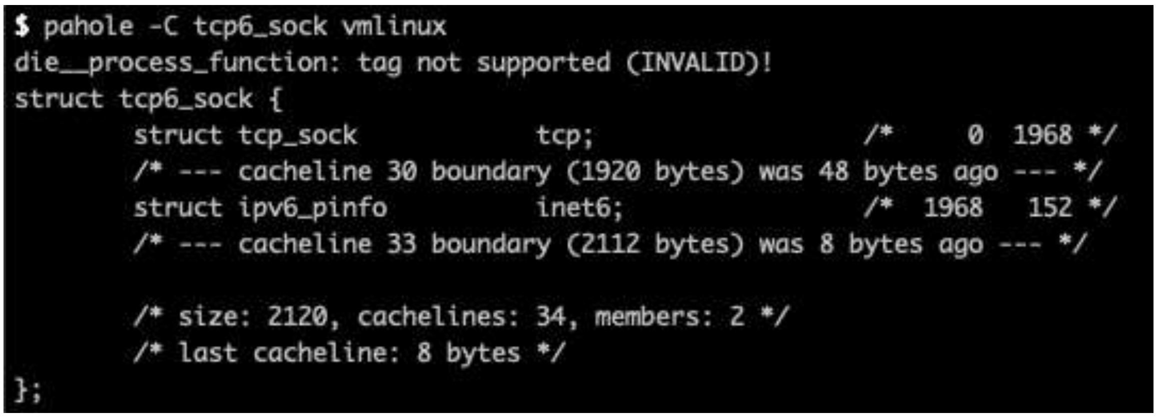
```
(gdb) target remomte :1234
(gdb) file /patch/to/vmlinux
(gdb) info r
(gdb) info all
(gdb) set print pretty on
(gdb) set print array on
(gdb) set print array-indexes on
(gdb) show print array-indexes
(gdb) ptype
(gdb) p *(struct mm_struct*)0xffffff......
(gdb) hb init_mm

打印数组
(gdb) p/x *idt_table@2
$47 =   {[0x0] = {
    offset_low = 0xc30,
    segment = 0x10,
    bits = {
      ist = 0x0,
      zero = 0x0,
      type = 0xe,
      dpl = 0x0,
      p = 0x1
    },
    offset_middle = 0x8300,
    offset_high = 0xffffffff,
    reserved = 0x0
  },
  [0x1] = {
    offset_low = 0x1160,
    segment = 0x10,
    bits = {
      ist = 0x3,
      zero = 0x0,
      type = 0xe,
      dpl = 0x0,
      p = 0x1
    },
    offset_middle = 0x8300,
    offset_high = 0xffffffff,
    reserved = 0x0
  }}
(gdb) print x=5
(gdb) set var x=5
(gdb) print $rsp
```

- gdb打印member offset
  `p &((struct rb_node*)0)->rb_left`

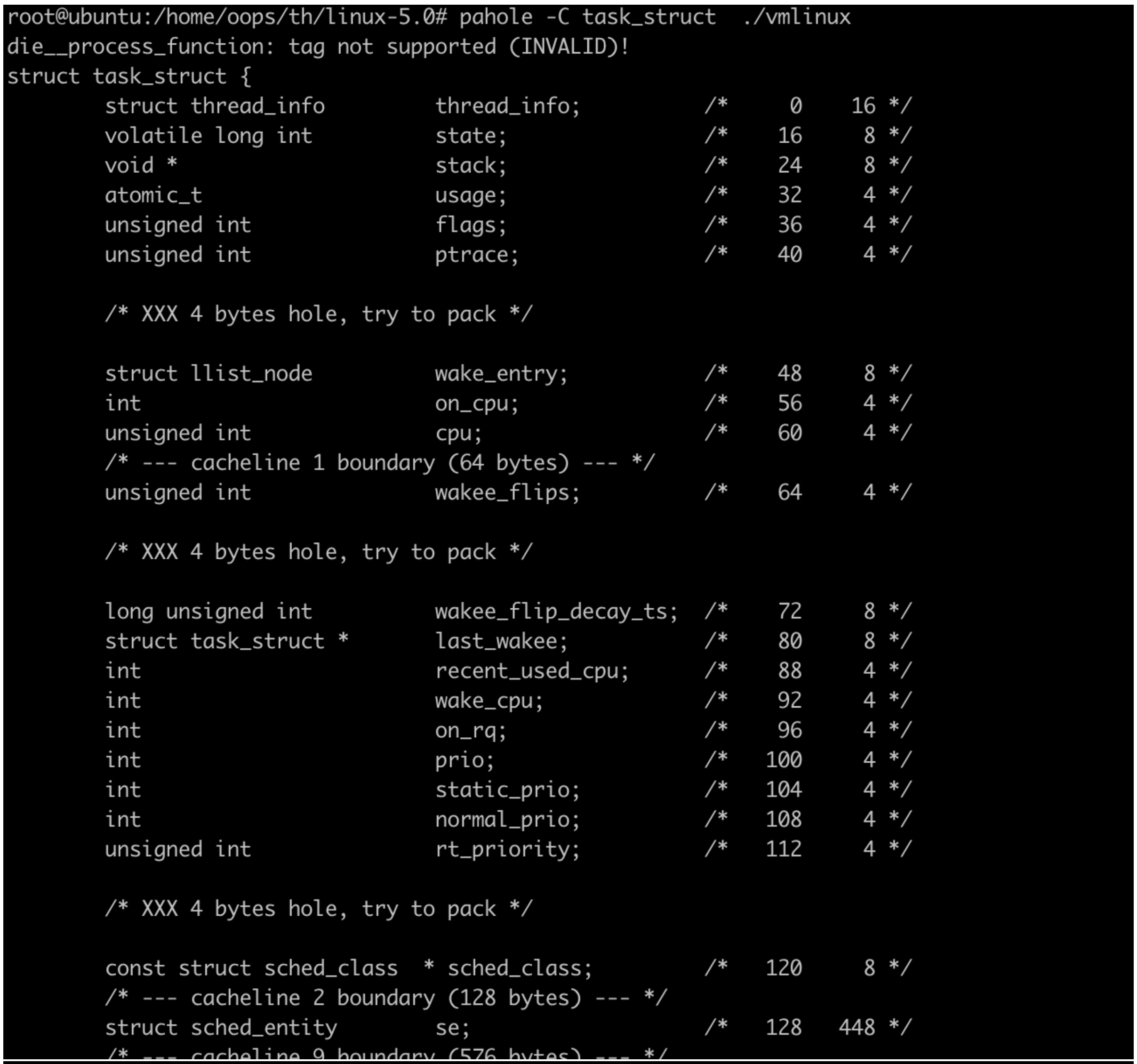
gdb中加载驱动的符号表
add-sysmbol-file *.ko 0x(.text的值)

- pahole安装
  apt-get install dwarves

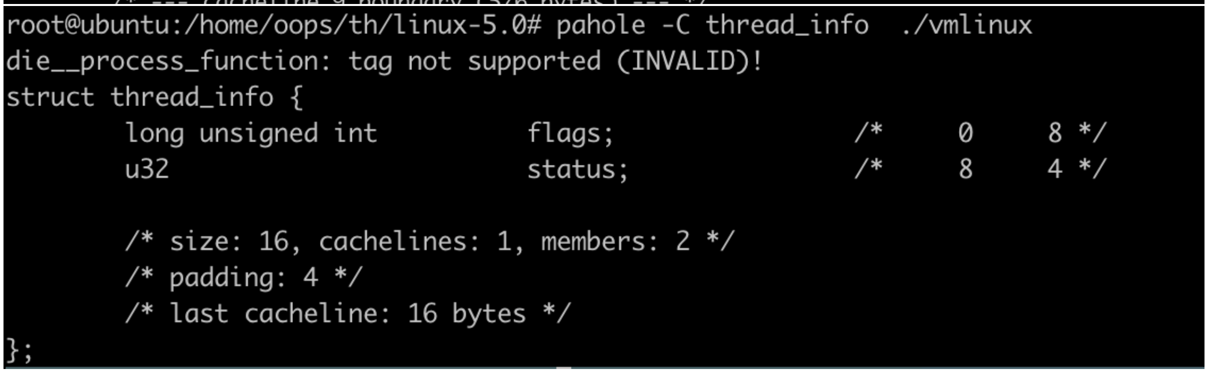
分析结构体可以用pahole工具，或者gdb 8.1之后用ptype /o



```
$ pahole -C tcp6_sock vmlinux
die__process_function: tag not supported (INVALID)!
struct tcp6_sock {
        struct tcp_sock           tcp;                     /*     0  1968 */
        /* --- cacheline 30 boundary (1920 bytes) was 48 bytes ago --- */
        struct ipv6_pinfo         inet6;                   /*  1968   152 */
        /* --- cacheline 33 boundary (2112 bytes) was 8 bytes ago --- */

        /* size: 2120, cachelines: 34, members: 2 */
        /* last cacheline: 8 bytes */
};
```



```
root@ubuntu:/home/oops/th/linux-5.0# pahole -C task_struct  ./vmlinux
die__process_function: tag not supported (INVALID)!
struct task_struct {
        struct thread_info         thread_info;          /*     0    16 */
        volatile long int          state;                /*    16     8 */
        void *                     stack;                /*    24     8 */
        atomic_t                   usage;                /*    32     4 */
        unsigned int               flags;                /*    36     4 */
        unsigned int               ptrace;               /*    40     4 */

        /* XXX 4 bytes hole, try to pack */

        struct llist_node          wake_entry;           /*    48     8 */
        int                        on_cpu;               /*    56     4 */
        unsigned int               cpu;                  /*    60     4 */
        /* --- cacheline 1 boundary (64 bytes) --- */
        unsigned int               wakee_flips;          /*    64     4 */

        /* XXX 4 bytes hole, try to pack */

        long unsigned int          wakee_flip_decay_ts;  /*    72     8 */
        struct task_struct *       last_wakee;           /*    80     8 */
        int                        recent_used_cpu;      /*    88     4 */
        int                        wake_cpu;             /*    92     4 */
        int                        on_rq;                /*    96     4 */
        int                        prio;                 /*   100     4 */
        int                        static_prio;          /*   104     4 */
        int                        normal_prio;          /*   108     4 */
        unsigned int               rt_priority;          /*   112     4 */

        /* XXX 4 bytes hole, try to pack */

        const struct sched_class  * sched_class;         /*   120     8 */
        /* --- cacheline 2 boundary (128 bytes) --- */
        struct sched_entity        se;                   /*   128   448 */
        /* --- cacheline 9 boundary (576 bytes) --- */
```

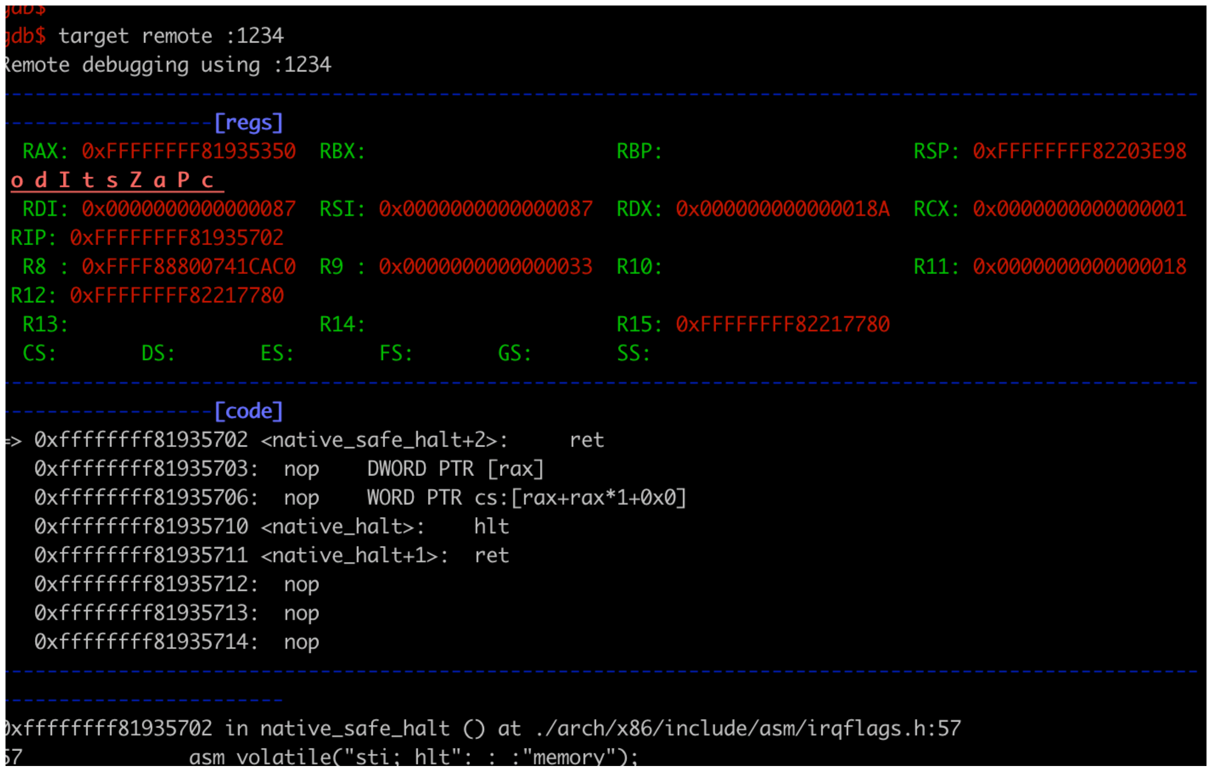


```
root@ubuntu:/home/oops/th/linux-5.0# pahole -C thread_info  ./vmlinux
die__process_function: tag not supported (INVALID)!
struct thread_info {
        long unsigned int          flags;                /*     0     8 */
        u32                        status;               /*     8     4 */

        /* size: 16, cachelines: 1, members: 2 */
        /* padding: 4 */
        /* last cacheline: 16 bytes */
};
```

安装gdb插件

git clone https://github.com/gatieme/GdbPlugins cp -rf ./GdbPlugins ~/.GdbPlugins

```
# 使用 peda
echo "source ~/.GdbPlugins/peda/peda.py" > ~/.gdbinit

# 使用 gef
echo "source ~/.GdbPlugins/gef/gef.py" > ~/.gdbinit

#使用 gdbinit
echo "source ~/.GdbPlugins/gdbinit/gdbinit" > ~/.gdbinit
```



```
db$ target remote :1234
emote debugging using :1234
--------------------------------------------------------------------------------------------------
-----------------[regs]
 RAX: 0xFFFFFFFF81935350  RBX:                     RBP:                     RSP: 0xFFFFFFFF82203E98
o d I t s Z a P c
 RDI: 0x0000000000000087  RSI: 0x0000000000000087  RDX: 0x000000000000018A  RCX: 0x0000000000000001
RIP: 0xFFFFFFFF81935702
 R8 : 0xFFFF88800741CAC0  R9 : 0x0000000000000033  R10:                     R11: 0x0000000000000018
R12: 0xFFFFFFFF82217780
 R13:                     R14:                     R15: 0xFFFFFFFF82217780
 CS:       DS:       ES:       FS:       GS:       SS:
--------------------------------------------------------------------------------------------------
-----------------[code]
> 0xffffffff81935702 <native_safe_halt+2>:     ret
  0xffffffff81935703:  nop    DWORD PTR [rax]
  0xffffffff81935706:  nop    WORD PTR cs:[rax+rax*1+0x0]
  0xffffffff81935710 <native_halt>:    hlt
  0xffffffff81935711 <native_halt+1>:  ret
  0xffffffff81935712:  nop
  0xffffffff81935713:  nop
  0xffffffff81935714:  nop
--------------------------------------------------------------------------------------------------
-----------------------

xffffffff81935702 in native_safe_halt () at ./arch/x86/include/asm/irqflags.h:57
7               asm volatile("sti; hlt": : :"memory");
```

- 加入log

```
         if(strstr(current->comm,"poc")!=0)
             {
                 printk("%s:%d(0x) (pid=%d, comm=%s)\n",
__func__,__LINE__,__LINE__,current->pid, current->comm);
                 dump_stack();

             }
```

参数的传递

```
x86_64: rdi, rsi, rdx ,rcx, r8,r9
x86, stack
arm64: x0-x7
arm: ro -r3
```

linux安装 ROPgadget

```
cd ROPgadget
sudo python setup.py install
git clone https://github.com/JonathanSalwan/ROPgadget.git
ROPgadget --binary level4 --only "pop|ret"

Gadgets information
```

遇到的问题

apt-get update失败

16.04 double free

- 解决
  sudo apt-get purge libappstream3
  或者
  sudo apt-get remove libappstream3
- 各种环境安装：
  apt-get install binutils python2.7 perl socat git build-essential gdb gdbserver