ROAD TO THE KERNEL

r2con 2017

Jaime Peñalba Estébanez
@NighterMan

About

JAIME PEÑALBA ESTÉBANEZ

@NighterMan

jpenalba@member.fsf.org

Linux kernel researcher at COSEINC
WhiskeyCON Superstar
The most useless RadareCore member

Abstract

This was intended to be a short talk about race conditions but ended up being a mix. (shit happens)

Besides races I will also cover some other things/tools which are helpful during the debugging/diagnosis and exploitation phases.

IMHO a good set of “helpers” proves to be worth in the long run.

Welcome To...

LINUX KERNEL RACING

Races in the kernel

The Linux kernel is prone to races by nature due to multiple reasons.

- SMP
- Preemption
- Process time quantum expiration
- IRQs
- Relinquish the CPU

Synchronization mechanisms

Technique	Description	Scope	Traceable
Per-CPU Vars	Duplicate a structures among CPUs	All CPUs	---
Atomic operations	Atomic read-modify-write instruction	All CPUs	---
Memory barriers	Avoid instruction reordering	Local/All	---
Spin locks	Lock with busy wait	All CPUs	inline
Semaphore	Lock with blocking wait (sleep)	All CPUs	Yes
Mutex	Mutual exclusive semaphore (might not sleep)	All CPUs	Yes
Seqlocks	Lock based on access counter	All CPUs	inline
Disable IRQ	Forbid interrupt handling on one CPU	Local CPU	inline
Disable Soft IRQ	Forbid deferrable function handling on one CPU	Local CPU	inline
RCU	Lock-free access to shared structures using pointers	All CPUs	inline

Initial source: Understanding Linux Kernel 3rd ed

One Example

```
static int counter;

void get_obj() {
   counter++;
}
```

Could translate to:

```
inc dword [rax]
```

```
add dword [obj.counter], 1
```

```
mov eax, dword [0x0020102c]
add eax, 1
mov dword [0x0020102c], eax
```

One Example

That operation its not atomic, effectively its a read-modify-write operation, so its susceptible to race.

Kernel would use the `atomic_t` type and the `atomic_*()` family functions.

Code should be:

```
lock inc dword [rax]
```

In case of a refcounter for example, we could use this to trigger an UAF.

Races in the kernel

IMHO, in conjunction with UAF, followed by out of bounds and maybe a few type confusions, race conditions are right now the most common bugs in the Linux kernel. (At least on the core components and LTS versions)

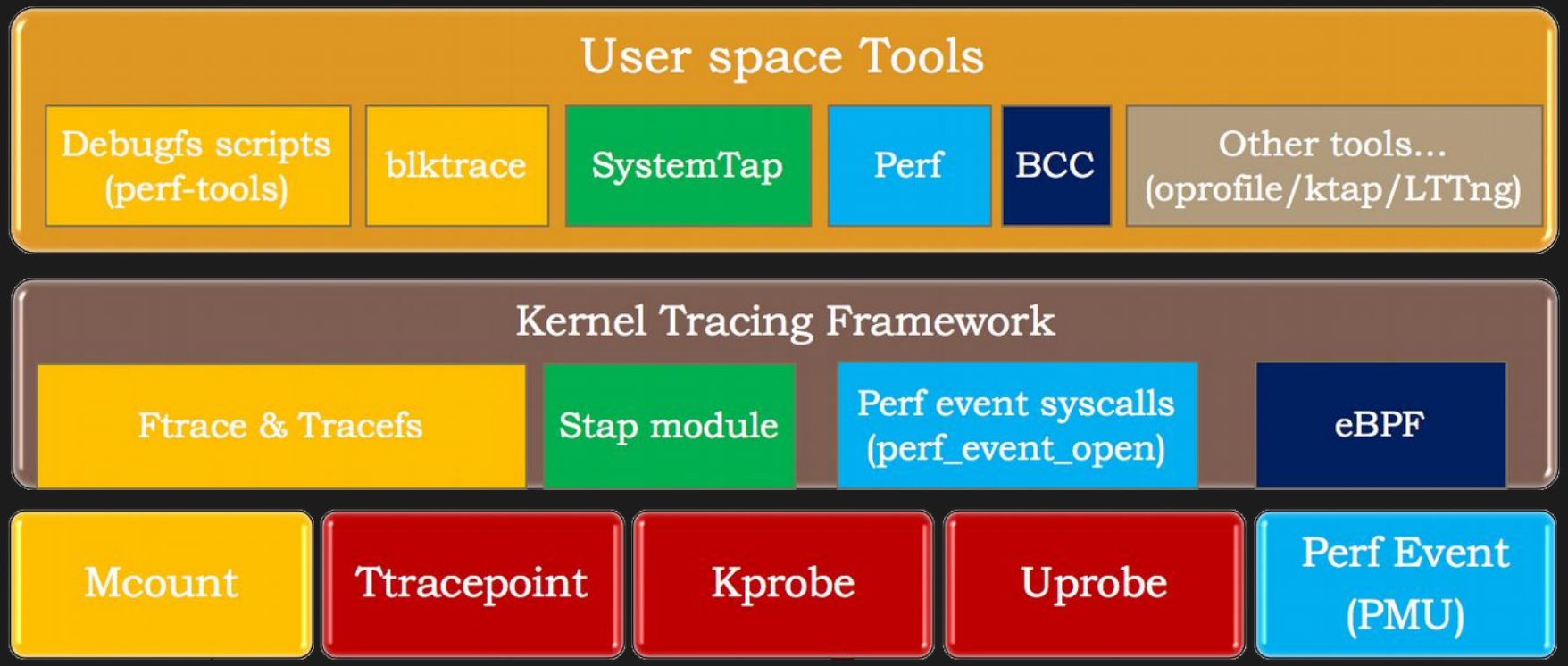
Races are sometimes hard to spot and even harder to reproduce from fuzzer crashes. We still have to see if KTSAN will changes this....

Although sometimes unreliable to trigger, when triggered they use to be very reliable in terms of system stability when exploting them.

Races in the kernel

Tracing on the Kernel

Linux kernel provides several ways for tracing (maybe too many):



For a more detailed description see:
http://www.brendangregg.com/blog/2015-07-08/choosing-a-linux-tracer.html

SystemTap

Although eBFP seems promising, its availability is limited to more recent >4.9 kernels. SystemTap works on almost any version.

SystemTap is the most powerful tracer. It can do everything: profiling, tracepoints, kprobes, uprobes, in-kernel programming...

It uses its own C-like language but guru mode `-g` also allows pure C or even asm.

Works by parsing its own language, generating C code, building a module and loading it.

Check this book:
http://myaut.github.io/dtrace-stap-book/index.html

Tracing Synchronization

Spinlocks, seqlocks and RCU use to be inlined, so there is not easy way to trace them.

Anyway they are easily spotted on the code, so it should not be a problem to deal with them.

Of course you could use source code static analysis to find occurrences in a code path between two functions.

Spinlock Example

```
185  static ssize_t
186  proc_file_read(struct file *file, char __user *buf, size_t nbytes,
187                 loff_t *ppos)
188  {
189      struct proc_dir_entry *pde = PDE(file->f_path.dentry->d_inode);
190      ssize_t rv = -EIO;
191
192      spin_lock(&pde->pde_unload_lock);
193      if (!pde->proc_fops) {
194          spin_unlock(&pde->pde_unload_lock);
195          return rv;
196      }
197      pde->pde_users++;
198      spin_unlock(&pde->pde_unload_lock);
199
200      rv = __proc_file_read(file, buf, nbytes, ppos);
201
202      pde_users_dec(pde);
203      return rv;
204  }
```

Locked for short amount of time and easily spotted

Spinlock and SystemTap

```
%{
#include <linux/preempt.h>
%}

function check_preempt:long () %{
        STAP_RETVALUE = preempt_count();
%}

probe kprobe.statement($1).absolute {
        printf("Preempt is: %d\n", check_preempt());
}
```

```
root@squeeze:/home/jaime/stap# stap -g preempt.stp 0xffffffff813b5cd8
Preempt is: 0
Preempt is: 0
```

Requires CONFIG_PREEMPT enabled

Tracing mutexes

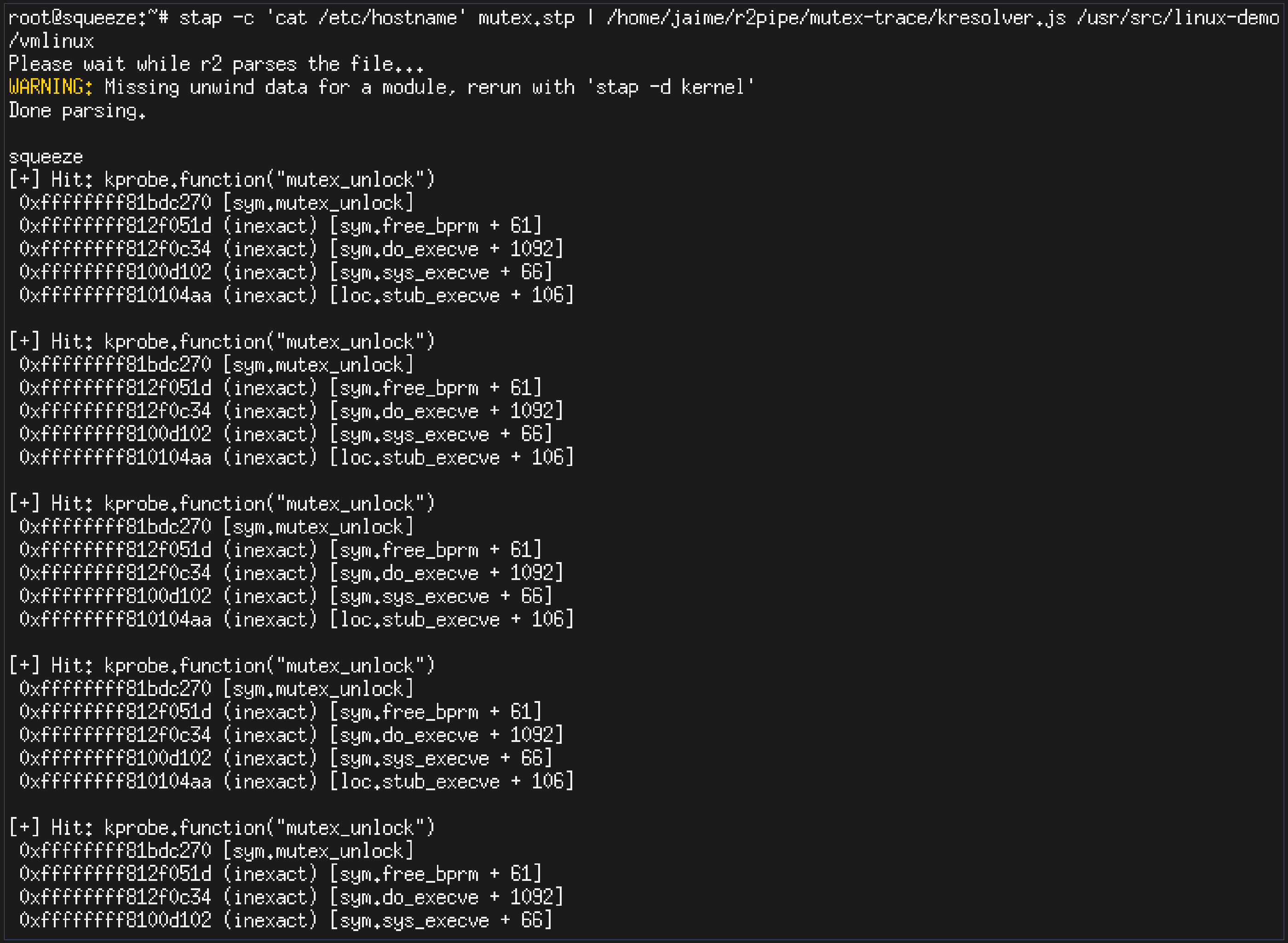
Mutexes and semaphores are much more easy to trace by registering a kprobe for their functions and record their status.

Then we could also insert a probe into a specific code position and check which mutexes or semaphores were held at that point.

Mutexes and SystemTap

```
probe kprobe.function("mutex_lock") {
    if (pid() == target()) {
        printf("[+] Hit: %s\n", pp());
        print_backtrace();
        printf("\n");
    }
}


probe kprobe.function("mutex_unlock") {
    if (pid() == target()) {
        printf("[+] Hit: %s\n", pp());
        print_backtrace();
        printf("\n");
    }
}
```

Mutexes and SystemTap



```
root@squeeze:~# stap -c 'cat /etc/hostname' mutex.stp | /home/jaime/r2pipe/mutex-trace/kresolver.js /usr/src/linux-demo
/vmlinux
Please wait while r2 parses the file...
WARNING: Missing unwind data for a module, rerun with 'stap -d kernel'
Done parsing.

squeeze
[+] Hit: kprobe.function("mutex_unlock")
 0xffffffff81bdc270 [sym.mutex_unlock]
 0xffffffff812f051d (inexact) [sym.free_bprm + 61]
 0xffffffff812f0c34 (inexact) [sym.do_execve + 1092]
 0xffffffff8100d102 (inexact) [sym.sys_execve + 66]
 0xffffffff810104aa (inexact) [loc.stub_execve + 106]

[+] Hit: kprobe.function("mutex_unlock")
 0xffffffff81bdc270 [sym.mutex_unlock]
 0xffffffff812f051d (inexact) [sym.free_bprm + 61]
 0xffffffff812f0c34 (inexact) [sym.do_execve + 1092]
 0xffffffff8100d102 (inexact) [sym.sys_execve + 66]
 0xffffffff810104aa (inexact) [loc.stub_execve + 106]

[+] Hit: kprobe.function("mutex_unlock")
 0xffffffff81bdc270 [sym.mutex_unlock]
 0xffffffff812f051d (inexact) [sym.free_bprm + 61]
 0xffffffff812f0c34 (inexact) [sym.do_execve + 1092]
 0xffffffff8100d102 (inexact) [sym.sys_execve + 66]
 0xffffffff810104aa (inexact) [loc.stub_execve + 106]

[+] Hit: kprobe.function("mutex_unlock")
 0xffffffff81bdc270 [sym.mutex_unlock]
 0xffffffff812f051d (inexact) [sym.free_bprm + 61]
 0xffffffff812f0c34 (inexact) [sym.do_execve + 1092]
 0xffffffff8100d102 (inexact) [sym.sys_execve + 66]
 0xffffffff810104aa (inexact) [loc.stub_execve + 106]

[+] Hit: kprobe.function("mutex_unlock")
 0xffffffff81bdc270 [sym.mutex_unlock]
 0xffffffff812f051d (inexact) [sym.free_bprm + 61]
 0xffffffff812f0c34 (inexact) [sym.do_execve + 1092]
 0xffffffff8100d102 (inexact) [sym.sys_execve + 66]
```

Excuse Me Sir, Do You Have a Moment to Talk About the r2 church?

R2 Book

WHAT THE ~~BIBLE~~ SAYS R2

"All Scripture is inspired of ~~God~~."
–2 Timothy 3:16, *New World Translation*.

WHAT THAT CAN MEAN FOR YOU

- Satisfying answers to life's big questions.–Proverbs 2:1-5.
- Reliable guidance for day-to-day living.–Psalm 119:105.
- Real hope for the future.–Romans 15:4.
- SCRIPTABLE RE WORK
- LGPL LICENSE

CAN WE REALLY BELIEVE WHAT THE ~~BIBLE~~ SAYS?
R2 Book

Yes, for at least three reasons:

- *Amazing harmony.* The ~~Bible~~ R2 was written over a period of 10[redacted] years by some 40 different people. Most of them never met one another. Yet, the entire ~~book~~ R2 is harmonious, with one central theme!
- *Honest history.* Secular historians seem quick to cover over the defeats of their people. In contrast, ~~Bible~~ R2 writers candidly recorded both their personal failings and those of their nation.–2 Chronicles 36:15, 16; Psalm 51:1-4.
- *Reliable prophecy.* The ~~Bible~~ R2 foretold the destruction of the ancient city of ~~Babylon~~ iD4 some 200 years in advance. (Isaiah 13:17-22) It revealed not only the manner in which ~~Babylon~~ iD4 would fall but even the name of its conqueror!–Isaiah 45:1-3.

Numerous other ~~Bible~~ R2 prophecies were also fulfilled down to the smallest detail. But is that not what we would expect of the Word of ~~God~~? –2 Peter 1:21. PAUCAKE

If you do not have a ~~Bible~~ R2, you can read it online at ~~www.jw.org/en~~
(Go to PUBLICATIONS > BIBLE)

HTTP://RADA.RE

In r2 we trust

Radare2 r2k for Linux

Developed during RSoC 2016 by:

RAKHOLIYA JENISH
@P4N74

OSCAR SALVADOR
@leberus

RSoC is funded by altruistic donors
So any help is welcome ☺

Radare2 r2k for Linux

R2k for Linux its composed of:

- Simple kernel module
- R2 I/O plugin (running in userspace)

The kernel module creates a character device and r2 interfaces with it through ioctls.

R2k Linux Supports

Kernel versions:

- From 2.6.X
- Up to 4.X

Architectures:

- x86/x86_64
- ARM

Platforms:

- Linux
- Android ?

R2k Installation

Build r2 from git as normally:

```
$ git clone https://github.com/radare/radare2.git
$ cd radare2/
$ sys/install.sh /usr/local
```

Install r2k using r2pm

```
$ r2pm -gi r2k-linux-unstable
$ sudo modprobe r2kmod
$ ls -al /dev/r2k
crw------- 1 root root 252, 0 May 31 02:59 /dev/r2k
```

Open r2 with r2k target

```
$ r2 r2k://
 -- In r2land usability is treated as a bug
[0x00000000]>
```

R2k Linux Features

- Read / Write from memory
- Supported memory maps
  - Kernel Linear address space
  - Selected process address space
  - Physical memory map
- Display information for a given proc
- Print kernel memory map (TODO)
- R/W control registers (partial)
- Honor WP bit when writing (patch)

Not a debugger / Somewhat limited

R2k Linux Features

Use `\?` for current I/O plugin help

```
jaime@squeeze:~$ r2 r2k://
 -- Warning, your trial license is about to expire.
[0x00000000]> \?
Usage:   \[MprRw][lpP] [args...]
\M                      Print kernel memory map
\b      beid [pid]      Change r2k backend. pid is required when beid is 1. Possible beid 0: linear address; 1: process
 address; 2: physical address
\p      pid             Print process information
\rl     addr len        Read from linear address
\rp     pid addr len    Read from process address
\rP     addr len        Read physical address
\R[p]                   Print control registers. Use =!Rp for detailed description
\wl[x]  addr input      Write at linear address. Use =!wlx for input in hex
\wp[x]  pid addr input  Write at process address. Use =!wpx for input in hex
\wP[x]  addr input      Write at physical address. Use =!wPx for input in hex
\W      1|0             Honor arch write protect (1 enable WP, 0 disable WP)
[0x00000000]>
```

Linear Address Space

```
jaime@squeeze:~$ r2 r2k://
 -- Reduce the delta where flag resolving by address is used with cfg.delta
[0x00000000]> #!pipe node /home/jaime/r2pipe/kallsyms-loader/ksymload.js /proc/kallsyms
[0x00000000]> s sym.sys_clone
[0xffffffff8100f4ca]> ps 30
H\x85\xf6H\x89\xd0L\x89\xc2u\x07I\x8b\xb0\x98\x00\x00\x00I\x89\xc9I\x89\xc01\xc9\xe9C\xe1\x03
[0xffffffff8100f4ca]> pd 30
            ;-- sym.sys_clone:
            0xffffffff8100f4ca      4885f6         test rsi, rsi
            0xffffffff8100f4cd      4889d0         mov rax, rdx
            0xffffffff8100f4d0      4c89c2         mov rdx, r8
        ,=< 0xffffffff8100f4d3      7507           jne 0xffffffff8100f4dc
        |   0xffffffff8100f4d5      498bb0980000.  mov rsi, qword [r8 + 0x98] ; [0x98:8]=-1 ; 152
        `-> 0xffffffff8100f4dc      4989c9         mov r9, rcx
            0xffffffff8100f4df      4989c0         mov r8, rax
            0xffffffff8100f4e2      31c9           xor ecx, ecx
        ,=< 0xffffffff8100f4e4      e943e10300     jmp sym.do_fork
        |   ;-- sym.sys_execve:
        |   0xffffffff8100f4e9      4156           push r14
        |   0xffffffff8100f4eb      4989f6         mov r14, rsi
        |   0xffffffff8100f4ee      4155           push r13
        |   0xffffffff8100f4f0      4989d5         mov r13, rdx
        |   0xffffffff8100f4f3      4154           push r12
        |   0xffffffff8100f4f5      4989cc         mov r12, rcx
        |   0xffffffff8100f4f8      55             push rbp
        |   0xffffffff8100f4f9      53             push rbx
        |   0xffffffff8100f4fa      e8f2a90e00     call sym.getname
        |   0xffffffff8100f4ff      483d00f0ffff   cmp rax, 0xfffffffffffff000
        |   0xffffffff8100f505      4889c3         mov rbx, rax
        |   0xffffffff8100f508      4889c5         mov rbp, rax
       ,==< 0xffffffff8100f50b      771c           ja 0xffffffff8100f529
       ||   0xffffffff8100f50d      4889c7         mov rdi, rax
       ||   0xffffffff8100f510      4c89e1         mov rcx, r12
       ||   0xffffffff8100f513      4c89ea         mov rdx, r13
       ||   0xffffffff8100f516      4c89f6         mov rsi, r14
       ||   0xffffffff8100f519      e8245d0e00     call sym.do_execve
       ||   0xffffffff8100f51e      4889df         mov rdi, rbx
       ||   0xffffffff8100f521      4863e8         movsxd rbp, eax
       ||   0xffffffff8100f524      e897a90e00     call sym.putname
[0xffffffff8100f4ca]>
```

Some Syscalls

Physical Address Space

```
jaime@squeeze:~$ r2 r2k://
 -- Learn pancake as if you were radare!
[0x00000000]> \b 0
[0x00000000]> s 0x000C0000
[0x000c0000]> pd 2
            0x000c0000      ff              invalid
            0x000c0001      ff              invalid
[0x000c0000]> \b 2
[0x000c0000]> pd 30
            0x000c0000      55              push rbp
            0x000c0001      aa              stosb byte [rdi], al
        ,=< 0x000c0002      4de97657a400    jmp 0xb0577e
        |   0x000c0008      0000            add byte [rax], al
        |   0x000c000a      0000            add byte [rax], al
        |   0x000c000c      0000            add byte [rax], al
        |   0x000c000e      0000            add byte [rax], al
        |   0x000c0010      0000            add byte [rax], al
        |   0x000c0012      0000            add byte [rax], al
        |   0x000c0014      0000            add byte [rax], al
        |   0x000c0016      0000            add byte [rax], al
        |   0x000c0018      1c91            sbb al, 0x91
        |   0x000c001a      0000            add byte [rax], al
        |   0x000c001c      0000            add byte [rax], al
        |   0x000c001e      49424d006655    add byte [r14 + 0x55], r12b
        |   0x000c0024      6689e5          mov bp, sp
        |   0x000c0027      6653            push bx
        |   0x000c0029      84c0            test al, al
       ,==< 0x000c002b      742c            je 0xc0059
       ||   0x000c002d      66bb0040        mov bx, 0x4000
       ||   0x000c0031      0000            add byte [rax], al
       ||   0x000c0033      fec8            dec al
      ,===< 0x000c0035      7438            je 0xc006f
      |||   0x000c0037      660fb7d2        movzx dx, dx
      |||   0x000c003b      660fb7c9        movzx cx, cx
      |||   0x000c003f      660fafd1        imul dx, cx
      |||   0x000c0043      66c1fa03        sar dx, 3
      |||   0x000c0047      6689d3          mov bx, dx
      |||   0x000c004a      9c              pushfq
      |||   0x000c004b      6681c3ff1f      add bx, 0x1fff
[0x000c0000]>
```

VGA BIOS

Pid info

```
jaime@squeeze:~$ r2 r2k://
 -- In Soviet Russia, radare2 has documentation.
[0x00000000]> \p 1
pid = 1
process name = init
00400000-00409000 r-xp 00000000 08:01 171372    init
00608000-00609000 rw-p 00008000 08:01 171372    init
020e8000-02109000 rw-p 00000000 00:00 0         [heap]
7f250b53d000-7f250b53f000 r-xp 00000000 08:01 82064     libdl-2.11.3.so
7f250b53f000-7f250b73f000 ---p 00002000 08:01 82064     libdl-2.11.3.so
7f250b73f000-7f250b740000 r--p 00002000 08:01 82064     libdl-2.11.3.so
7f250b740000-7f250b741000 rw-p 00003000 08:01 82064     libdl-2.11.3.so
7f250b741000-7f250b8a3000 r-xp 00000000 08:01 82058     libc-2.11.3.so
7f250b8a3000-7f250baa3000 ---p 00162000 08:01 82058     libc-2.11.3.so
7f250baa3000-7f250baa7000 r--p 00162000 08:01 82058     libc-2.11.3.so
7f250baa7000-7f250baa8000 rw-p 00166000 08:01 82058     libc-2.11.3.so
7f250baa8000-7f250baad000 rw-p 00000000 00:00 0
7f250baad000-7f250bac9000 r-xp 00000000 08:01 81631     libselinux.so.1
7f250bac9000-7f250bcc8000 ---p 0001c000 08:01 81631     libselinux.so.1
7f250bcc8000-7f250bcc9000 r--p 0001b000 08:01 81631     libselinux.so.1
7f250bcc9000-7f250bcca000 rw-p 0001c000 08:01 81631     libselinux.so.1
7f250bcca000-7f250bccb000 rw-p 00000000 00:00 0
7f250bccb000-7f250bce9000 r-xp 00000000 08:01 82052     ld-2.11.3.so
7f250bea2000-7f250bea6000 rw-p 00000000 00:00 0
7f250bea6000-7f250bee1000 r-xp 00000000 08:01 81678     libsepol.so.1
7f250bee1000-7f250bee2000 rw-p 0003a000 08:01 81678     libsepol.so.1
7f250bee6000-7f250bee8000 rw-p 00000000 00:00 0
7f250bee8000-7f250bee9000 r--p 0001d000 08:01 82052     ld-2.11.3.so
7f250bee9000-7f250beea000 rw-p 0001e000 08:01 82052     ld-2.11.3.so
7f250beea000-7f250beeb000 rw-p 00000000 00:00 0
7ffcb3b79000-7ffcb3b8e000 rw-p 00000000 00:00 0         [stack]
7ffcb3bee000-7ffcb3bef000 r-xp 00000000 00:00 0
STACK BASE ADDRESS = 0xffff88013fa4dff8
[0x00000000]>
```

sysvinit pid info

Process Address Space

```
7f250bee8000-7f250bee9000 r--p 0001d000 08:01 82052    ld-2.11.3.so
7f250bee9000-7f250beea000 rw-p 0001e000 08:01 82052    ld-2.11.3.so
7f250beea000-7f250beeb000 rw-p 00000000 00:00 0
7ffcb3b79000-7ffcb3b8e000 rw-p 00000000 00:00 0        [stack]
7ffcb3bee000-7ffcb3bef000 r-xp 00000000 00:00 0
STACK BASE ADDRESS = 0xffff88013fa4dff8
[0x00000000]> \b 1 1
[0x00000000]> s 0x004024a0
[0x004024a0]> pd 20
            0x004024a0      31ed           xor ebp, ebp
            0x004024a2      4989d1         mov r9, rdx
            0x004024a5      5e             pop rsi
            0x004024a6      4889e2         mov rdx, rsp
            0x004024a9      4883e4f0       and rsp, 0xfffffffffffffff0
            0x004024ad      50             push rax
            0x004024ae      54             push rsp
            0x004024af      49c7c0a06e40.  mov r8, 0x406ea0
            0x004024b6      48c7c1b06e40.  mov rcx, 0x406eb0
            0x004024bd      48c7c7605e40.  mov rdi, 0x405e60          ; "AWAVAUATUH..S..H...." 0x00405e60
            0x004024c4      e8affbffff     call 0x402078
            0x004024c9      f4             hlt
            0x004024ca      90             nop
            0x004024cb      90             nop
            0x004024cc      4883ec08       sub rsp, 8
            0x004024d0      488b05495d20.  mov rax, qword [0x00608220] ; [0x608220:8]=0
            0x004024d7      4885c0         test rax, rax
        ,=< 0x004024da      7402           je 0x4024de
        |   0x004024dc      ffd0           call rax
        `-> 0x004024de      4883c408       add rsp, 8
[0x004024a0]>
```

sysvinit .text section

Process Address Space

```
root@squeeze:~# r2 /bin/init
Cannot open '/bin/init'
root@squeeze:~# r2 /sbin/init [redacted]
 -- I love the smell of bugs in the morning.
[0x004024a0]> pd 20
            ;-- entry0:
            ;-- section..text: [redacted]
            0x004024a0      31ed           xor ebp, ebp                ; section 14 va=0x004024a0 pa=0x000024a0 sz=1916
0 vsz=19160 rwx=--r-x .text
            0x004024a2      4989d1         mov r9, rdx
            0x004024a5      5e             pop rsi
            0x004024a6      4889e2         mov rdx, rsp
            0x004024a9      4883e4f0       and rsp, 0xfffffffffffffff0
            0x004024ad      50             push rax
            0x004024ae      54             push rsp
            0x004024af      49c7c0a06e40.  mov r8, 0x406ea0
            0x004024b6      48c7c1b06e40.  mov rcx, 0x406eb0
            0x004024bd      48c7c7605e40.  mov rdi, 0x405e60           ; main ; "AWAVAUATUH\x89\xf55\x89\xfbH\x81\xec\x
88\x03"
            0x004024c4      e8affbffff     call sym.imp.__libc_start_main
            0x004024c9      f4             hlt
            0x004024ca      90             nop
            0x004024cb      90             nop
            0x004024cc      4883ec08       sub rsp, 8
            0x004024d0      488b05495d20.  mov rax, qword [reloc.__gmon_start___32] ; [0x608220:8]=0
            0x004024d7      4885c0         test rax, rax
        ,=< 0x004024da      7402           je 0x4024de
        |   0x004024dc      ffd0           call rax
        `-> 0x004024de      4883c408       add rsp, 8
[0x004024a0]>
```

```
7f250bee8000-7f250bee9000 r--p 0001d000 08:01 82052    ld-2.11.3.so
7f250bee9000-7f250beea000 rw-p 0001e000 08:01 82052    ld-2.11.3.so
7f250beea000-7f250beeb000 rw-p 00000000 00:00 0
7ffcb3b79000-7ffcb3b8e000 rw-p 00000000 00:00 0        [stack]
7ffcb3bee000-7ffcb3bef000 r-xp 00000000 00:00 0
STACK BASE ADDRESS = 0xffff88013fa4dff8
[0x00000000]> \b 1 1
[0x00000000]> s 0x004024a0 [redacted]
[0x004024a0]> pd 20
            0x004024a0      31ed           xor ebp, ebp
            0x004024a2      4989d1         mov r9, rdx
            0x004024a5      5e             pop rsi
            0x004024a6      4889e2         mov rdx, rsp
            0x004024a9      4883e4f0       and rsp, 0xfffffffffffffff0
            0x004024ad      50             push rax
            0x004024ae      54             push rsp
            0x004024af      49c7c0a06e40.  mov r8, 0x406ea0
            0x004024b6      48c7c1b06e40.  mov rcx, 0x406eb0
            0x004024bd      48c7c7605e40.  mov rdi, 0x405e60           ; "AWAVAUATUH..S..H...." 0x00405e60
            0x004024c4      e8affbffff     call 0x402078
            0x004024c9      f4             hlt
            0x004024ca      90             nop
            0x004024cb      90             nop
            0x004024cc      4883ec08       sub rsp, 8
            0x004024d0      488b05495d20.  mov rax, qword [0x00608220] ; [0x608220:8]=0
            0x004024d7      4885c0         test rax, rax
        ,=< 0x004024da      7402           je 0x4024de
        |   0x004024dc      ffd0           call rax
        `-> 0x004024de      4883c408       add rsp, 8
[0x004024a0]>
```

sysvinit .text section

Compare original init binary with it

from r2k using proc address space

Control Registers

```
jaime@squeeze:~$ r2 r2k://                                                    [13/1816]
 -- Almost 5am, maybe you should go to bed.
[0x00000000]> \R
cr0 = 0x80050033
cr1 = 0x0
cr2 = 0x7f1272d9d401
cr3 = 0x100214000
cr4 = 0x6e0
cr8 = 0x0
[0x00000000]> \Rp
CR0: 0x80050033
 [*] PG:    1
 [*] CD:    0
 [*] NW:    0
 [*] AM:    1
 [*] WP:    1
 [*] NE:    1
 [*] ET:    1
 [*] TS:    0
 [*] EM:    0
 [*] MP:    1
 [*] PE:    1

CR2: 0x7f1272d9d401
Page-Fault Linear Address: 0x7f1272d9d401

CR3: 0x100214000
 [*] Page-Directory Base:    0x100214000
 [*] PCD:                    0
 [*] PWT:                    0

CR4: 0x6e0
 [*] PKE:        0
 [*] SMAP:       0
 [*] SMEP:       0
 [*] OSXSAVE:    0
 [*] PCIDE:      0
 [*] FSGSBASE:   0
 [*] SMXE:       0
 [*] VMXE:       0
 [*] UMIP:       0
 [*] OSXMMEXCPT: 1
 [*] OSFXSR:     1
```

Limitations 1

As everything in this world, r2k is far from perfect:

- R2k is not a debugger
- Very few code runs in kernel space
- When reading/writing the kernel is executing, so expect inconsistencies
- No symbols for now ☹

Limitations 2

Developing a kernel module compatible with multiple archs and making it compatible with releases from 14 years ago up to nowadays its not an easy task and gets messy quickly...

The Linux kernel and its APIs evolve constantly and are sometimes inconsistent between releases/archs/platforms.

Some core components such as MM are not always accessible/exported to modules requiring hacks and duplicating code.

Security also changes and example could be HARDENED_USERCOPY

Some good news

- You get all the features from r2 (except dbg)
    - Emulation (ESIL)
    - Disassemblers
    - Memory inspection/searches
    - Easy scripting on lots of languages
    - Lots of other features provided by r2

- Its good to have some code loaded in the kernel:
    - Install kprobes
    - Hot memory patching
    - Manipulate control registers

Roadmap

R2 is not a source level debugger anyway there are some things we can do to make it more suitable for kernel, bootloaders and electronics

- Improve DWARF support (In progress)
- Improve GDB client
    - kvm/qemu/vmware gdb stub
    - JTAG for embeded hardware (OpenOCD...)
- Add support to install `kprobes`, let the kernel handle the software interrupt and dispatch it to our custom handler.
- Allocate some RWX pages for custom code

Other implementations

Windows

- r2k module for windows
- Also has some bochs r2 plugin for debugging windows bootloader and kernel

Developed by @SkUaTeR he is around here, so feel free to ask him about it in “engrish”.

Alternative: Lars Haukli @zutle zdbg plugin

macOS

Pending to do, only protype.
No ETA, ask @pancake

Scripting with r2pipe

R2pipe is a simple r2 API oriented to do quick scripts. It communicates with r2 using a stdio/stdout or a pipe descriptor.

Its works using normal r2 commands. Those commands can be executed normally or appending a `j` to them for json output.

```
[0x00005430]> pd 1
                ;-- entry0:
                0x00005430      31ed           xor ebp, ebp
[0x00005430]> pdj 1
[{"offset":21552,"esil":"ebp,ebp,^=,$z,zf,=,$p,pf,=,$s,sf,=,$0,cf,=,$0,of,=,0xffffffff,rbp,&=","refptr":false,"f
cn_addr":0,"fcn_last":0,"size":2,"opcode":"xor ebp, ebp","bytes":"31ed","family":"cpu","type":"xor","type_num":2
8,"type2_num":0,"flags":["entry0"]}]

[0x00005430]>
```

Scripting with r2pipe

Just a couple of functions are enough to use r2pipe.

- open()/connect()
- cmd()
- cmdj()

Scripting with r2pipe

The most supported languages are:

- NodeJS
- Python
- Swift
- C/Nim/Vala/C++

Scripting with r2pipe

But there is also r2pipe for:

	pipe	spawn	async	http	tcp	rap	json	plug	lib	buff
C	X	X	-	X	X	X	X	X	X	X
C++/Qt	X	X	-	-	-	-	X	-	X	-
C# / F#	X	X	X	X	-	-	-	-	X	-
D	X	-	-	-	-	-	X	-	-	-
Erlang	X	X	-	-	-	-	-	-	-	-
Go	X	X	-	-	-	-	X	-	-	-
Haskell	X	X	-	X	-	-	X	-	-	-
Java/Groovy	-	X	-	X	-	-	-	-	X	-
Lisp	-	X	-	-	-	-	X	-	-	-
NewLisp	X	X	-	X	-	-	X	-	X	-
Nim	-	-	-	X	-	-	X	-	X	-
NodeJS	X	X	X	X	X	-	X	X	-	X
Ocaml	-	X	-	-	-	-	X	-	-	-
Perl	X	X	-	X	X	-	X	-	-	-
PHP	-	X	-	-	-	-	-	-	-	-
Python	X	X	X	X	X	X	X	X	X	-
Ruby	X	X	-	-	-	-	X	-	-	-
Rust	X	X	-	-	-	-	X	-	-	-
Swift	X	X	X	X	-	-	X	-	X	-
Vala	-	X	X	-	-	-	-	-	-	-
Clojure	X	X	-	-	-	-	-	-	-	-

Scripting with r2pipe

Installation

```
$ pip install r2pipe
```

or

```
$ pip3 install r2pipe
```

Usage example:

```
import r2pipe

r2 = r2pipe.open("/bin/ls")
r2.cmd('aa')
print(r2.cmd("afl"))
print(r2.cmdj("aflj"))  # evaluates JSONs and returns an object
r2.quit()
```

Injecting artifical sleeps

A good way to quickly test if a piece of code is prone to race conditions is by injecting an sleep somewhere in the code.

This is useful to test paths that you suspect might be racy, but you don’t want to spend time reading code or tracing the execution.

Small Demo

```
root@squeeze:~# node /home/jaime/r2pipe/kinject-sleep/kinject-sleep.js -p 0xffffffff814f1310 -d 0xffffffffa0209340 -f /
tmp/patch.undo
[*] Looking for opcodes to overwrite with jump...
   [-] Need to overwrite: push rbp (size 1)
   [-] Need to overwrite: mov rbp, rsp (size 3)
   [-] Need to overwrite: push r15 (size 2)
   [-] Need to overwrite: push r14 (size 2)
[*] Jump back addr will be: 0xffffffff814f1318
[*] Searching for msleep...
   [-] msleep is at: 0xffffffff81108c70
[*] Injecting sleep code at 0xffffffffa0209340...
   [-] Sleep time will be 1000 msecs
[*] Patching function with jmp at addr: 0xffffffff814f1310
[*] Done. Lets hope this wont crash....
root@squeeze:~# time ipcs

------ Shared Memory Segments --------
key        shmid      owner      perms      bytes      nattch     status

------ Semaphore Arrays --------
key        semid      owner      perms      nsems

------ Message Queues --------
key        msqid      owner      perms      used-bytes   messages


real    0m2.005s
user    0m0.000s
sys     0m0.003s
root@squeeze:~# node /home/jaime/r2pipe/kinject-sleep/kinject-sleep.js -u /tmp/patch.undo
[*] Undo patch 0xffffffff814f1310: 554889e541574156
[*] Undo patch 0xffffffffa0209340: 554489c04889e54157415641554154489d7534889c34881eca80000004889bd78ffffff48897598488
cf4889c648894d8889459444894d9048894580e89e0b3de14839c30f85c9030000488b45884c897db048c745b828
root@squeeze:~# time ipcs

------ Shared Memory Segments --------
key        shmid      owner      perms      bytes      nattch     status

------ Semaphore Arrays --------
key        semid      owner      perms      nsems

------ Message Queues --------
key        msqid      owner      perms      used-bytes   messages


real    0m0.003s
user    0m0.000s
sys     0m0.004s
root@squeeze:~#
```

patch semctl syscall

Small Demo

```
%{
#include <linux/delay.h>
%}

function do_sleep() %{
        msleep(1000);
%}

probe kprobe.statement($1).absolute {
        do_sleep();
}
```

Using SystemTap

Small Demo

```
root@squeeze:~# time ipcs

------ Shared Memory Segments --------
key        shmid      owner      perms      bytes      nattch     status

------ Semaphore Arrays --------
key        semid      owner      perms      nsems

------ Message Queues --------
key        msqid      owner      perms      used-bytes   messages


real    0m0.003s
user    0m0.000s
sys     0m0.003s
root@squeeze:~# stap -g --suppress-time-limits sleep.stp 0xffffffff814f1310
^Z
[1]+  Stopped                 stap -g --suppress-time-limits sleep.stp 0xffffffff814f1310
root@squeeze:~# bg
[1]+ stap -g --suppress-time-limits sleep.stp 0xffffffff814f1310 &
root@squeeze:~# time ipcs

------ Shared Memory Segments --------
key        shmid      owner      perms      bytes      nattch     status

------ Semaphore Arrays --------
key        semid      owner      perms      nsems

------ Message Queues --------
key        msqid      owner      perms      used-bytes   messages


real    0m1.004s
user    0m0.001s
sys     0m0.002s
root@squeeze:~# fg
stap -g --suppress-time-limits sleep.stp 0xffffffff814f1310
^Croot@squeeze:~#
```

Using SystemTap

Dwarf Parser

Dwarf parser was implemented hopping to improve structure support when debug symbols are available.

There are already other means to handle structures and also to handle DWARF in r2, but this is r2land, we love duplicities...

Implemented as plugin.

Dwarf Installation

Build r2 from git as normally:

```
$ git clone https://github.com/radare/radare2.git
$ cd radare2/
$ sys/install.sh /usr/local
```

Install dwarf-parser plugin with r2pm

```
$ r2pm -i libfwarf
$ r2pm -i dwarf-parser
```

Dwarf Help

Dwarf parser cmds start with `idd`

```
[0xffffffffff600000]> idd?
|Usage: idd commands for dwarf plugin. replacing idd can be replaced with "?:dwarf"
| iddi <path to dwarf file>                initialise sdb and dbg entries for dwarf_info
| idd structname                           print struct data for their respective members
| iddv structname[.member[.member[...]]]   print value for specific member of structure if "structname.member.." else
| idda structname[.member[.member[...]]]   print address for requested  member of structure or structure itself
| iddlg                                    print flags in r2 format for all global variables
| iddlf                                    print flags in r2 format for all functions
| iddd structname[.member[.member[...]]]   print c type declaration
| idddl structname[.member[.member[...]]]  print c type declaration including members of structure/union type
| iddt structname[.member[.member[...]]]   print type and size
[0xffffffffff600000]>
```

Dwarf Cmds

You can print the structure declaration in C format

```
[0xffffffffff600000]> iddd key
struct key {
  atomic_t usage;
  key_serial_t serial;
  union (null) {
    struct list_head graveyard_link;
    struct rb_node serial_node;
  } ;
  struct key_type *type;
  struct rw_semaphore sem;
  struct key_user *user;
  void *security;
  union (null) {
    time_t expiry;
    time_t revoked_at;
  } ;
  uid_t uid;
  gid_t gid;
  key_perm_t perm;
```

```
  gid_t gid;
  key_perm_t perm;
  short unsigned int quotalen;
  short unsigned int datalen;
  long unsigned int flags;
  char *description;
  union (null) {
    struct list_head link;
    long unsigned int x[2];
    void *p[2];
  } type_data;
  union (null) {
    long unsigned int value;
    void *data;
    struct keyring_list *subscriptions;
  } payload;
  time_t last_used_at;
};
```

Dwarf Cmds

You can print the structure declaration in C format. Also in JSON

```
[0xffffffffff600000]> idddj key ~{}
{
  "name": "key",
  "size": 160,
  "inbits": false,
  "struct": true,
  "members": [
    {
      "name": "usage",
      "type": "atomic_t ",
      "typedef": true,
      "struct": false,
      "union": false,
      "pointer": 0,
      "const": false,
      "volatile": false,
      "function": false,
      "enum": false,
      "array": false,
      "size": 4,
      "inbits": false,
      "offset": 0,
      "members": [
        {
          "name": "counter",
          "type": "volatile int ",
          "typedef": false,
          "struct": false,
          "union": false,
          "pointer": 0,
```

```
    {
      "name": "sem",
      "type": "struct rw_semaphore ",
      "typedef": false,
      "struct": true,
      "union": false,
      "pointer": 0,
      "const": false,
      "volatile": false,
      "function": false,
      "enum": false,
      "array": false,
      "size": 32,
      "inbits": false,
      "offset": 40,
      "members": [
        {
          "name": "count",
          "type": "rwsem_count_t ",
          "typedef": true,
          "struct": false,
          "union": false,
          "pointer": 0,
          "const": false,
          "volatile": false,
          "function": false,
          "enum": false,
          "array": false,
          "size": 8,
          "inbits": false,
          "offset": 40
        },
```

Dwarf Cmds

Print size and type
(JSON oh rly?)

```
[0xffffffff81046016]>  iddt key_type.name
type : const char *
size : 8
[0xffffffff81046016]>  iddt key_type.link
type : struct list_head
size : 16
[0xffffffff81046016]>  iddtj key_type.link
{"type":"struct list_head ","size":"16"}
[0xffffffff81046016]>
```

Radare2 gdbserver

Part of GSoC 2017:

SRIMANTA BARUA

Srimanta has been working on improving previous gdbserver implementation

Radare2 gdbserver

gdbserver custom cmds

```
[0xffffffff81046016]> \?
Usage: =!cmd args
 =!pid              - show targeted pid
 =!pkt s            - send packet 's'
 =!monitor cmd      - hex-encode monitor command and pass to target interpreter
 =!detach [pid]     - detach from remote/detach specific pid
 =!inv.reg          - invalidate reg cache
 =!pktsz            - get max packet size used
 =!pktsz bytes      - set max. packet size as 'bytes' bytes
 =!exec_file [pid] - get file which was executed for current/specified pid
[0xffffffff81046016]>
```

Radare2 gdbserver

Connecting to a qemu/kvm instance

```
jaime@Infinity:~/research/kernels/demo$ r2 -i load,r2 -d gdb://127.0.0.1:41806
= attach 1 0
 -- This is just an existentialist experiment.
[0xffffffff81046016]> iddv key_type @ sym.key_type_logon
name = 0xffffffff817db0a8,
def_datalen = 0x0,
instantiate = 0xffffffff81238ea0,
update = 0xffffffff81238dd0,
match = 0xffffffff81238db0,
revoke = 0xffffffff81238d50,
destroy = 0xffffffff81238d20,
describe = 0xffffffff81238cc0,
read = 0x0,
request_key = 0x0,
link = {
  next = 0xffffffff81af06c0,
  prev = 0xffffffff81af0c70
}
[0xffffffff81046016]>
```

Radare2 gdbserver

Monitor cmd

```
[0xffffffff81046016]> \monitor help
acl_add aclname match allow|deny [index] -- add a match rule to the access control list
acl_policy aclname allow|deny -- set default access control list policy
acl_remove aclname match -- remove a match rule from the access control list
acl_reset aclname -- reset the access control list
acl_show aclname -- list rules in the access control list
balloon target -- request VM to change its memory allocation (in MB)
block_job_cancel [-f] device -- stop an active background block operation (use -f
                        if the operation is currently paused)
block_job_complete device -- stop an active background block operation
block_job_pause device -- pause an active background block operation
block_job_resume device -- resume a paused background block operation
block_job_set_speed device speed -- set maximum speed for a background block operation
block_passwd block_passwd device password -- set the password of encrypted block devices
block_resize device size -- resize a block image
block_set_io_throttle device bps bps_rd bps_wr iops iops_rd iops_wr -- change I/O throttle limits for a block drive
block_stream device [speed [base]] -- copy data from a backing file into a block device
boot_set bootdevice -- define new values for the boot device list
change device filename [format [read-only-mode]] -- change a removable medium, optional format
chardev-add args -- add chardev
chardev-remove id -- remove chardev
client_migrate_info protocol hostname port tls-port cert-subject -- set migration information for remote display
closefd closefd name -- close a file descriptor previously passed via SCM rights
commit device|all -- commit changes to the disk images (if -snapshot is used) or backing files
cpu index -- set the default CPU
```

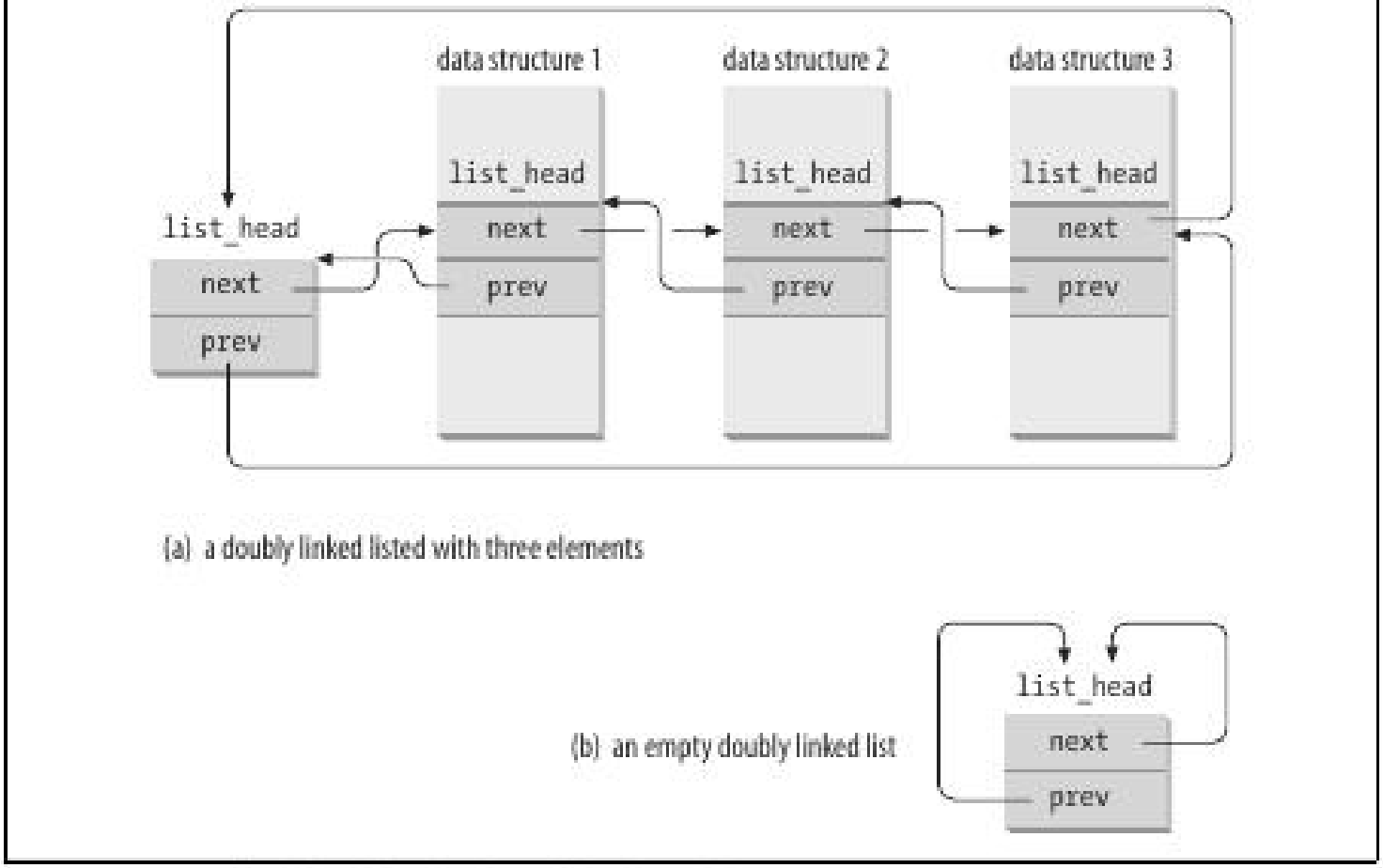
Demo time (or not)

HI, MY NAME IS JAIME

```
jaime@Infinity:~/research/kernels/demo$ r2 -i load.r2 -d gdb://127.0.0.1:41806
= attach 1 0
 -- Control the height of the terminal on serial consoles with e scr.height
[0xffffffff81046016]> iddv key_type.link.next @ sym.key_types_list
0xffffffff81233910
[0xffffffff81046016]> iddv key_type.link.next.next @ sym.key_types_list
0x54415541e5894855
[0xffffffff81046016]> iddv key_type.link.next.next.next @ sym.key_types_list
0xffffffffffffffff
[0xffffffff81046016]> iddv key_type.link.next.next.next.next @ sym.key_types_list
Segmentation fault
jaime@Infinity:~/research/kernels/demo$
```

AND WELCOME TO JACKASS

Listing processes



Source: Understanding Linux Kernel 3rd ed

Listing processes

```
#define next_task(p) \
    list_entry_rcu((p)->tasks.next, struct task_struct, tasks)

#define for_each_process(p) \
    for (p = &init_task ; (p = next_task(p)) != &init_task ; )
```

```
#define list_entry_rcu(ptr, type, member) \
    container_of(rcu_dereference(ptr), type, member)
```

```
#define container_of(ptr, type, member) ({          \
    const typeof( ((type *)0)->member ) *__mptr = (ptr);    \
    (type *)( (char *)__mptr - offsetof(type,member) );})
```

Listing processes

Example listing processes using r2pipe and dwarf plugin

Example traversing key_types list using r2pipe

Listing processes

```
[0xffffffff81046016]> #!pipe /home/jaime/research/r2-scripts/list-processes.py
PID: 1 - init - loginuid: -1
PID: 2 - kthreadd - loginuid: -1
PID: 3 - migration/0 - loginuid: -1
PID: 4 - ksoftirqd/0 - loginuid: -1
PID: 5 - stopper/0 - loginuid: -1
PID: 6 - watchdog/0 - loginuid: -1
PID: 7 - migration/1 - loginuid: -1
PID: 8 - stopper/1 - loginuid: -1
PID: 9 - ksoftirqd/1 - loginuid: -1
PID: 10 - watchdog/1 - loginuid: -1
PID: 11 - migration/2 - loginuid: -1
PID: 12 - stopper/2 - loginuid: -1
PID: 13 - ksoftirqd/2 - loginuid: -1
PID: 14 - watchdog/2 - loginuid: -1
PID: 15 - migration/3 - loginuid: -1
PID: 16 - stopper/3 - loginuid: -1
PID: 17 - ksoftirqd/3 - loginuid: -1
PID: 18 - watchdog/3 - loginuid: -1
PID: 19 - events/0 - loginuid: -1
PID: 20 - events/1 - loginuid: -1
PID: 21 - events/2 - loginuid: -1
PID: 22 - events/3 - loginuid: -1
PID: 23 - events/0 - loginuid: -1
PID: 24 - events/1 - loginuid: -1
PID: 25 - events/2 - loginuid: -1
PID: 26 - events/3 - loginuid: -1
PID: 27 - events_long/0 - loginuid: -1
PID: 28 - events_long/1 - loginuid: -1
PID: 29 - events_long/2 - loginuid: -1
PID: 30 - events_long/3 - loginuid: -1
PID: 31 - events_power_ef - loginuid: -1
PID: 32 - events_power_ef - loginuid: -1
PID: 33 - events_power_ef - loginuid: -1
PID: 34 - events_power_ef - loginuid: -1
```

Random /b/

4CHAN PARTY VAN

Somehow not as funny when it's parked in your driveway.

Sanitizers

Sanitizers are easy and common on user space, but not so common in kernel space.

KASAN
- gcc >= 4.9.2
- Introduced in ~4.0

UBSAN
- gcc >= 5.0
- Introduced in 4.5

KTSAN
Out of tree. Recently released. Only works with clang IIRC.

Your target might not support sanitizers...

Sanitizers

```
Sanitizers
Arrow keys navigate the menu.  <Enter> selects submenus --->.  Highlighted letters are hotkeys.  Pressing <Y>
includes, <N> excludes, <M> modularizes features.  Press <Esc><Esc> to exit, <?> for Help, </> for Search.  Legend:
[*] built-in  [ ] excluded  <M> module  < > module capable

[*] AddressSanitizer: dynamic memory error detector
(128) Quarantine size in MB
(0x38ff0000) Offset of shadow memory
[*]   Check double for double free
[ ]   Intel performance counters workaround
[ ]   Enable tests on boot
[*] UndefinedBehaviorSanitizer: undefined behavior detector
[ ]   Check shift operations
[*]   Detect division by zero
[*]   Unreachable code
[*]   Check variable length array size
[*]     Ignore zero array size
[*]   NULL Pointer checking
[*]   Return statement checking
[*]   Signed integer overflow checking
[*]   Array bounds checking
[ ]     Strict array bounds checking
[ ]   Checking of alignment of pointers
[*]   Check memory references to __builtin_object_size
[*]   Floating point div by zero checking
[*]   Floating point to int cast
[*]   Chech non-null args on calls
[*]   Chech return non-null
[*]   Chech bool values
[*]   Chech enum range
[*]   Chech C++ members
[ ]   Enable tests on boot

<Select>   < Exit >   < Help >
```

Dwarf information

```
./kdwarf2db: option requires an argument -- 'h'
dwarf2db 0.1 [load dwarf information into mongodb]
Jaime Peñalba <jpenalbae@gmail.com>

Usage: ./kdwarf2db [-m] [-h host] [-p port] [-d database] dwarf_file
  -m:           mark as module
  -h host:      mongodb host
  -p port:      mongodb port
  -d database:  mongodb database
```

Structures Search

```
kstruct.js 0.1 [search matching structures into the database]
Jaime Peñalba <jpenalbae@gmail.com>

Usage: ./kstruct.js [options]

Options for structure members search:
  -t type: is type
  -T typedef: is typedef
  -N name: is named
  -b size: has byte size of
  -o offset: is at offset N
  -p pointer: integer. 0 for no pointer, 1 pointer, 2 double pointer...
  -A size: is an array of the given size
  -e bool: is an enum
  -d bool: is a typedef
  -f bool: is a function
  -u bool: is an union
  -r bool: is a structure
  -a bool: is an array
  -v bool: is volatile
  -c bool: is a const

General search options:
  -s size: search structures which would be allocate on the same cache pool
  -S size: search structures matching exactly the given size
  -n name: search structure named "name"

  -l level: how many levels of nested structures to search
  -m bool: search structures declared into modules or not

Output options:
  -L: print Long description
  -X: print eXtra long description
  -B: print structure Bytes size next to the name
  -C: print Comma separated list of results
```

Structures Search

```
~/security-CVS/kaf$ ./kstruct.js -n thread_info -X
[*] Found 1 matches

struct thread_info {
    struct task_struct *task;                                   /* offset: 0 (size: 8) */
    struct exec_domain *exec_domain;                            /* offset: 8 (size: 8) */
    (typedef __u32) unsigned int flags;                         /* offset: 16 (size: 4) */
    (typedef __u32) unsigned int status;                        /* offset: 20 (size: 4) */
    (typedef __u32) unsigned int cpu;                           /* offset: 24 (size: 4) */
    int preempt_count;                                          /* offset: 28 (size: 4) */
    (typedef mm_segment_t) struct mm_segment_t {
        long unsigned int seg;                                  /* offset: 32 (size: 8) */
    } addr_limit;                                               /* offset: 32 (size: 8) */
    struct restart_block {
        func (*fn)();                                           /* offset: 40 (size: 8) */
        union {
            struct {
                long unsigned int arg0;                         /* offset: 48 (size: 8) */
                long unsigned int arg1;                         /* offset: 56 (size: 8) */
                long unsigned int arg2;                         /* offset: 64 (size: 8) */
                long unsigned int arg3;                         /* offset: 72 (size: 8) */
            };                                                  /* offset: 48 (size: 32) */
            struct {
                (typedef u32) unsigned int *uaddr;              /* offset: 48 (size: 8) */
                (typedef u32) unsigned int val;                 /* offset: 56 (size: 4) */
                (typedef u32) unsigned int flags;               /* offset: 60 (size: 4) */
                (typedef u32) unsigned int bitset;              /* offset: 64 (size: 4) */
                (typedef u64) long long unsigned int time;      /* offset: 72 (size: 8) */
                (typedef u32) unsigned int *uaddr2;             /* offset: 80 (size: 8) */
            } futex;                                            /* offset: 48 (size: 40) */
            struct {
                (typedef __kernel_clockid_t) int index;         /* offset: 48 (size: 4) */
                struct timespec *rmtp;                          /* offset: 56 (size: 8) */
                struct compat_timespec *compat_rmtp;            /* offset: 64 (size: 8) */
                (typedef u64) long long unsigned int expires;   /* offset: 72 (size: 8) */
            } nanosleep;                                        /* offset: 48 (size: 32) */
            struct {
                struct pollfd *ufds;                            /* offset: 48 (size: 8) */
                int nfds;                                       /* offset: 56 (size: 4) */
                int has_timeout;                                /* offset: 60 (size: 4) */
                long unsigned int tv_sec;                       /* offset: 64 (size: 8) */
```

Structures Search

```
~/security-CVS/kaf$ ./kstruct.js -o 168 -p 1 -r true -t list_head
device
dev_pm_info
file
file_lock
sb_writers
netns_ipv6
cgroup
vfsmount
request
irq_desc
pci_dev
pnp_card
pnp_dev
rq
vmap_block
timerfd_ctx
pci_host_bridge
xenbus_device
xs_handle
agp_bridge_data
tpm_vendor_specific
tpm_chip
bus_type_private
root_device
Scsi_Host
pcmcia_driver
usb_device
usb_hcd
rtc_device
powercap_control_type
dma_chan_dev
in_device
virtio_device
port
nozomi
ipw_hardware
drm_file
drm_mode_config
drm_dp_mst_port
```

Funcions data usage

```
kfunction.js 0.1 [search matching functions into the database]
Jaime Peñalba <jpenalbae@gmail.com>

Usage: ./kfunction.js [options]

Options for local variables search:
  -t type: is type
  -T typedef: is typedef
  -N name: is named
  -b size: has byte size of
  -o offset: is at offset N
  -p pointer: integer. 0 for no pointer, 1 pointer, 2 double pointer...
  -A size: is an array of the given size
  -e bool: is an enum
  -d bool: is a typedef
  -f bool: is a function
  -u bool: is an union
  -r bool: is a structure
  -a bool: is an array
  -v bool: is volatile
  -c bool: is a const

General search options:
  -n name: search functions named "name"

  -l level: how many levels of nested structures to search
  -m bool: search structures declared into modules or not

Output options:
  -L: print Long description
  -D: print declaration file
  -C: print Comma separated list of results
```

Search for code paths

```
~$ ktrace.js -h
Usage: node trace.js to_func [from_func] [options]

Where options are:
-l                Print long description
-c maxcalls       Maximun function calls
-p maxpath        Maximum path length
-d file           Cscope database file
-x func1,func2    Exclude paths containning the given functions
```

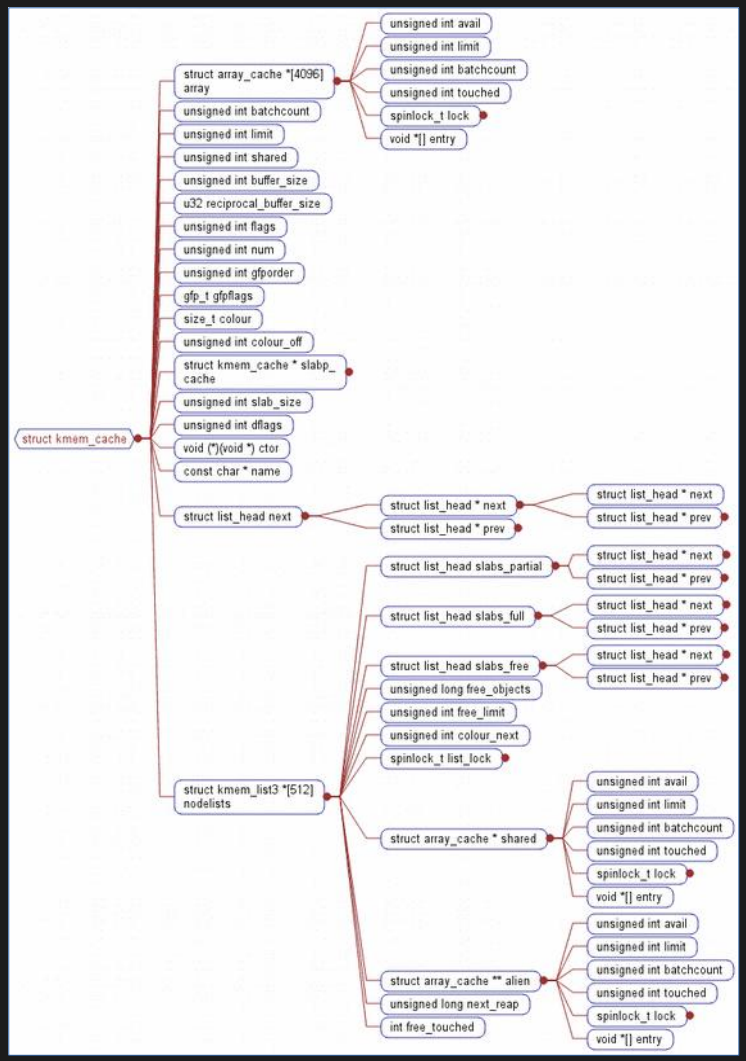
Search for code paths

```
^$ ktrace.js con_release_unimap
[✖] fbcon_init() -> con_copy_unimap() -> con_free_unimap() -> con_release_unimap()
[✖] fbcon_event_notify() -> fbcon_fb_unbind() -> set_con2fb_map() -> con2fb_init_display() -> fbcon_set_disp() -> con_copy_unimap() -> con_free_unimap() -> con_release_unimap()
[✖] fbcon_event_notify() -> fbcon_remap_all() -> set_con2fb_map() -> con2fb_init_display() -> fbcon_set_disp() -> con_copy_unimap() -> con_free_unimap() -> con_release_unimap()
[✖] fbcon_event_notify() -> fbcon_fb_registered() -> set_con2fb_map() -> con2fb_init_display() -> fbcon_set_disp() -> con_copy_unimap() -> con_free_unimap() -> con_release_unimap()
[✖] fbcon_event_notify() -> set_con2fb_map() -> con2fb_init_display() -> fbcon_set_disp() -> con_copy_unimap() -> con_free_unimap() -> con_release_unimap()
[✖] fbcon_event_notify() -> fbcon_new_modelist() -> fbcon_set_disp() -> con_copy_unimap() -> con_free_unimap() -> con_release_unimap()
[✖] vgacon_init() -> con_free_unimap() -> con_release_unimap()
[✖] vgacon_deinit() -> con_free_unimap() -> con_release_unimap()
[✖] vt_compat_ioctl() -> vt_ioctl() -> do_unimap_ioctl() -> con_set_unimap() -> con_unify_unimap() -> con_release_unimap()
[✖] vt_compat_ioctl() -> compat_unimap_ioctl() -> con_set_unimap() -> con_unify_unimap() -> con_release_unimap()
[✖] tty_init() -> vty_init() -> console_map_init() -> con_set_default_unimap() -> con_unify_unimap() -> con_release_unimap()
[✖] con_open() -> vc_allocate() -> con_set_default_unimap() -> con_unify_unimap() -> con_release_unimap()
[✖] vt_compat_ioctl() -> vt_ioctl() -> vc_allocate() -> con_set_default_unimap() -> con_unify_unimap() -> con_release_unimap()
[✖] SYSCALL_DEFINE4() -> kernel_kexec() -> pm_prepare_console() -> vt_move_to_console() -> vc_allocate() -> con_set_default_unimap() -> con_unify_unimap() -> con_release_unimap()
[✖] resume_store() -> software_resume() -> hibernation_restore() -> pm_prepare_console() -> vt_move_to_console() -> vc_allocate() -> con_set_default_unimap() -> con_unify_unimap() -> con_release_unimap()
[✖] snapshot_ioctl() -> hibernation_restore() -> pm_prepare_console() -> vt_move_to_console() -> vc_allocate() -> con_set_default_unimap() -> con_unify_unimap() -> con_release_unimap()
[✖] acpi_system_write_sleep() -> hibernate() -> pm_prepare_console() -> vt_move_to_console() -> vc_allocate() -> con_set_default_unimap() -> con_unify_unimap() -> con_release_unimap()
[✖] acpi_system_write_sleep() -> acpi_suspend() -> hibernate() -> pm_prepare_console() -> vt_move_to_console() -> vc_allocate() -> con_set_default_unimap() -> con_unify_unimap() -> con_release_unimap()
[✖] tps65010_probe() -> tps65010_work() -> tps65010_interrupt() -> hibernate() -> pm_prepare_console() -> vt_move_to_console() -> vc_allocate() -> con_set_default_unimap() -> con_unify_unimap() -> con_release_unimap()
[✖] state_store() -> hibernate() -> pm_prepare_console() -> vt_move_to_console() -> vc_allocate() -> con_set_default_unimap() -> con_unify_unimap() -> con_release_unimap()
[✖] SYSCALL_DEFINE4() -> hibernate() -> pm_prepare_console() -> vt_move_to_console() -> vc_allocate() -> con_set_default_unimap() -> con_unify_unimap() -> con_release_unimap()
[✖] resume_store() -> software_resume() -> pm_prepare_console() -> vt_move_to_console() -> vc_allocate() -> con_set_default_unimap() -> con_unify_unimap() -> con_release_unimap()
[-] MAX_CALLS: regulator_register() -> set_machine_constraints() -> suspend_prepare() -> pm_prepare_console() -> vt_move_to_console() -> vc_allocate() -> con_set_default_unimap() -> con_unify_unimap() -> con_release_unimap()
[✖] regulator_suspend_prepare() -> suspend_prepare() -> pm_prepare_console() -> vt_move_to_console() -> vc_allocate() -> con_set_default_unimap() -> con_unify_unimap() -> con_release_unimap()
[✖] state_store() -> enter_state() -> suspend_prepare() -> pm_prepare_console() -> vt_move_to_console() -> vc_allocate() -> con_set_default_unimap() -> con_unify_unimap() -> con_release_unimap()
[-] MAX_CALLS: pm_suspend() -> enter_state() -> suspend_prepare() -> pm_prepare_console() -> vt_move_to_console() -> vc_allocate() -> con_set_default_unimap() -> con_unify_unimap() -> con_release_unimap()
[✖] SYSCALL_DEFINE4() -> kernel_kexec() -> pm_restore_console() -> vt_move_to_console() -> vc_allocate() -> con_set_default_unimap() -> con_unify_unimap() -> con_release_unimap()
[✖] resume_store() -> software_resume() -> hibernation_restore() -> pm_restore_console() -> vt_move_to_console() -> vc_allocate() -> con_set_default_unimap() -> con_unify_unimap() -> con_release_unimap()
[✖] snapshot_ioctl() -> hibernation_restore() -> pm_restore_console() -> vt_move_to_console() -> vc_allocate() -> con_set_default_unimap() -> con_unify_unimap() -> con_release_unimap()
[✖] acpi_system_write_sleep() -> hibernate() -> pm_restore_console() -> vt_move_to_console() -> vc_allocate() -> con_set_default_unimap() -> con_unify_unimap() -> con_release_unimap()
[✖] acpi_system_write_sleep() -> acpi_suspend() -> hibernate() -> pm_restore_console() -> vt_move_to_console() -> vc_allocate() -> con_set_default_unimap() -> con_unify_unimap() -> con_release_unimap()
[✖] tps65010_probe() -> tps65010_work() -> tps65010_interrupt() -> hibernate() -> pm_restore_console() -> vt_move_to_console() -> vc_allocate() -> con_set_default_unimap() -> con_unify_unimap() -> con_release_unimap()
[✖] state_store() -> hibernate() -> pm_restore_console() -> vt_move_to_console() -> vc_allocate() -> con_set_default_unimap() -> con_unify_unimap() -> con_release_unimap()
[✖] SYSCALL_DEFINE4() -> hibernate() -> pm_restore_console() -> vt_move_to_console() -> vc_allocate() -> con_set_default_unimap() -> con_unify_unimap() -> con_release_unimap()
[✖] resume_store() -> software_resume() -> pm_restore_console() -> vt_move_to_console() -> vc_allocate() -> con_set_default_unimap() -> con_unify_unimap() -> con_release_unimap()
[-] MAX_CALLS: regulator_register() -> set_machine_constraints() -> suspend_prepare() -> pm_restore_console() -> vt_move_to_console() -> vc_allocate() -> con_set_default_unimap() -> con_unify_unimap() -> con_release_unimap()
[✖] regulator_suspend_prepare() -> suspend_prepare() -> pm_restore_console() -> vt_move_to_console() -> vc_allocate() -> con_set_default_unimap() -> con_unify_unimap() -> con_release_unimap()
[✖] state_store() -> enter_state() -> suspend_prepare() -> pm_restore_console() -> vt_move_to_console() -> vc_allocate() -> con_set_default_unimap() -> con_unify_unimap() -> con_release_unimap()
[-] MAX_CALLS: pm_suspend() -> enter_state() -> suspend_prepare() -> pm_restore_console() -> vt_move_to_console() -> vc_allocate() -> con_set_default_unimap() -> con_unify_unimap() -> con_release_unimap()
[✖] state_store() -> enter_state() -> suspend_finish() -> pm_restore_console() -> vt_move_to_console() -> vc_allocate() -> con_set_default_unimap() -> con_unify_unimap() -> con_release_unimap()
```

GDB Scripts

```
asan addr -- print ASAN info about a given addr
asanquarantined addr -- Check if addr is in ASAN quarantine
kmemcache addr -- print cache info about a given addr
kmemcachename name -- find kmem cache by name
kmemcacheac name -- Print kmem cache array caches
kmemcachegraph name [-f] -- find kmem cache by name and generate graph
kmemcachelist -- List all kmem caches
slab addr -- Find cache and slab info for the given addr
slabasan addr [-l] -- Print slab for the given addr including asan info
```

Dynamic Memory



ASAN Query

```
>>> asan 0xFFFF8800A3CF70C0
kmem_cache at: 0xffff8800bf0701c0
Shadow at: 0xffff88004d78ee18
Found redzone in shadow at: 0xffff88004d78ee30
Redzone in mem at: 0xffff8800a3cf7180

Kmalloc size: 160
Object size: 192
Cache name: size-192

Allocated by thread 1085:
   0xffffffff814ae53e <start_this_handle+1598>
   0xffffffff814ae6c9 <journal_start+281>
   0xffffffff81408a90 <ext3_journal_start_sb+144>
   0xffffffff813f2634 <ext3_unlink+260>
   0xffffffff812fa56a <vfs_unlink+442>
   0xffffffff81301777 <do_unlinkat+455>
   0xffffffff81303fb1 <sys_unlink+17>
   0xffffffff81010062 <system_call+146>

Freed by thread 285:
   0xffffffff814b5cb6 <__journal_drop_transaction+470>
   0xffffffff814b5f3d <__journal_remove_checkpoint+285>
   0xffffffff814b6db9 <journal_clean_one_cp_list+281>
   0xffffffff814b6f59 <__journal_clean_checkpoint_list+169>
   0xffffffff814b219a <journal_commit_transaction+906>
   0xffffffff814bd79a <kjournald+378>
   0xffffffff8112bfe5 <kthread+245>
   0xffffffff8101120a <child_rip+10>

Address is located 0 bytes inside of 160-byte region. [0xffff8800a3cf70c0, 0xffff8800a3cf7160]
```

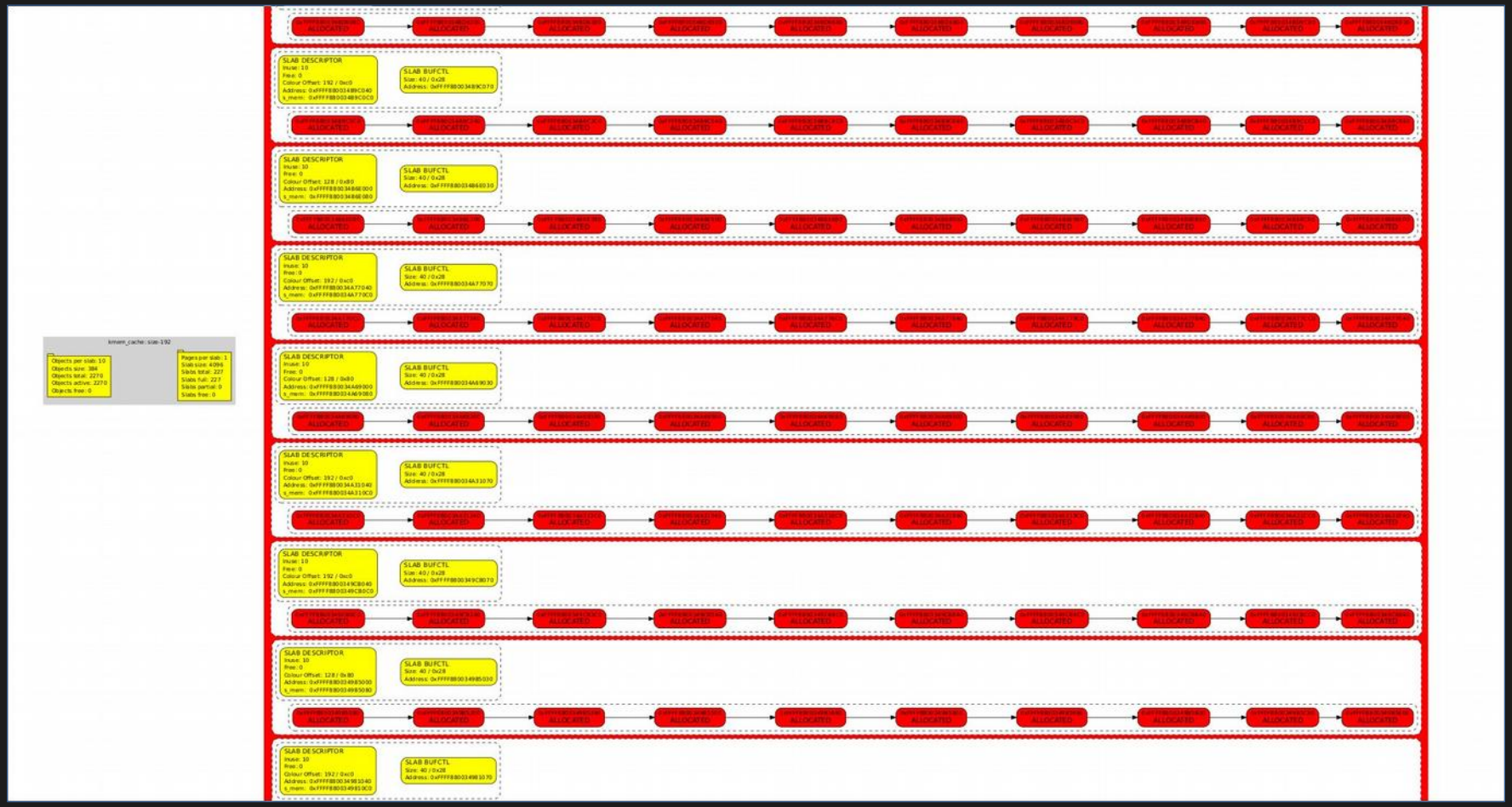
SLAB Cahe Info

```
Address is located 0 bytes inside of 192 byte region, [0xffff8800bf0701c0, 0xffff8800bf0701c0]
>>> kmemcachename size-192
kmem_cache at: 0xffff8800bf0701c0

$12 = {
  array = {[0] = 0xffff8800bf2fb400, [1] = 0xffff8800bc144680, [2] = 0x0 <per_cpu.irq_stack_union> <repeats 4094 times>},
  batchcount = 27,
  limit = 54,
  shared = 8,
  buffer_size = 384,
  reciprocal_buffer_size = 11184811,
  flags = 270336,
  num = 10,
  gfporder = 0,
  gfpflags = 0,
  colour = 2,
  colour_off = 64,
  slabp_cache = 0x0 <per_cpu.irq_stack_union>,
  slab_size = 128,
  dflags = 0,
  ctor = 0x0 <per_cpu.irq_stack_union>,
  name = 0xffffffff81e00bd0 "size-192",
  next = {
    next = 0xffff8800bf0681d8,
    prev = 0xffff8800bf088258
  },
  object_size = 192,
  nodelists = {[0] = 0xffff8800bf021580, [1] = 0x0 <per_cpu.irq_stack_union>, [2] = 0x82a173e3812ba35a, [3] = 0x829d228e829d30e1, [4] = 0xffffffff829d23d9 <x86_64_start_kernel+327>, [5] = 0x0 <per_cpu.irq
_stack_union>, [6] = 0x0 <per_cpu.irq_stack_union>, [7] = 0x0 <per_cpu.irq_stack_union>, [8] = 0x0 <per_cpu.irq_stack_union>, [9] = 0x0 <per_cpu.irq_stack_union>, [10] = 0x1 <per_cpu.irq_stack_union+1>, [
11] = 0x0 <per_cpu.irq_stack_union>, [12] = 0x0 <per_cpu.irq_stack_union>, [13] = 0x0 <per_cpu.irq_stack_union>, [14] = 0x0 <per_cpu.irq_stack_union>, [15] = 0x0 <per_cpu.irq_stack_union>, [16] = 0x0 <per
_cpu.irq_stack_union>, [17] = 0x0 <per_cpu.irq_stack_union>, [18] = 0xffffffff00000000, [19] = 0x0 <per_cpu.irq_stack_union> <repeats 493 times>}
}

0xFFFF8800BF0701C0: size-192
 - Object size/per slab: 384/10
 - Pages per slab/bytes: 1/4096
 - Objects total/active/free: 2270/2270/0
 - SLABS total/full/partial/free: 227/227/0/0
 - Array caches available shared - cpu1/cpu2/...: 0 - 8/0

Array cache shared avail/limit/batchcount/touched: 0/216/3131961357/0


Array cache cpu0 avail/limit/batchcount/touched: 8/54/27/1
0xFFFF880033CA30C0, 0xFFFF880033CA3240, 0xFFFF880033CA33C0, 0xFFFF880033CA3540, 0xFFFF880033CA36C0, 0xFFFF880033CA3840, 0xFFFF880033CA39C0, 0xFFFF880033CA3B40

Array cache cpu1 avail/limit/batchcount/touched: 0/54/27/0
```

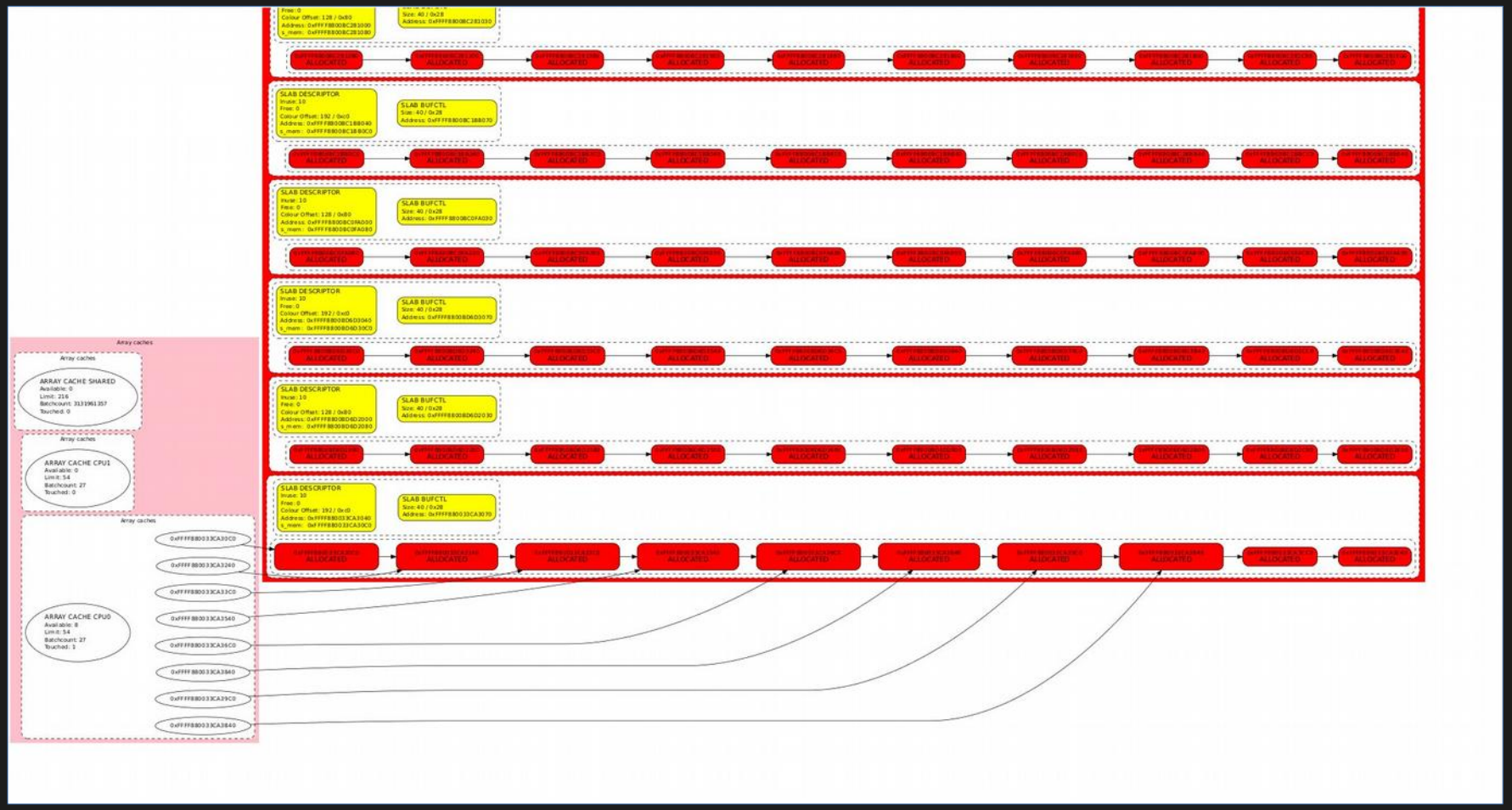
SLAB Inspection

```
>>> slab 0xffff8800a3cf7840
kmem_cache at: 0xFFFF8800BF0701C0
$13 = {
  array = {[0] = 0xffff8800bf2fb400, [1] = 0xffff8800bc144680, [2] = 0x0 <per_cpu.irq_stack_unio
  batchcount = 27,
  limit = 54,
  shared = 8,
  buffer_size = 384,
  reciprocal_buffer_size = 11184811,
  flags = 270336,
  num = 10,
  gfporder = 0,
  gfpflags = 0,
  colour = 2,
  colour_off = 64,
  slabp_cache = 0x0 <per_cpu.irq_stack_union>,
  slab_size = 128,
  dflags = 0,
  ctor = 0x0 <per_cpu.irq_stack_union>,
  name = 0xffffffff81e00bd0 "size-192",
  next = {
    next = 0xffff8800bf0681d8,
    prev = 0xffff8800bf088258
  },
  object_size = 192,
  nodelists = {[0] = 0xffff8800bf021580, [1] = 0x0 <per_cpu.irq_stack_union>, [2] = 0x82a173e381
ck_union>, [9] = 0x0 <per_cpu.irq_stack_union>, [10] = 0x1 <per_cpu.irq_stack_union+1>, [11] = 0
] = 0x0 <per_cpu.irq_stack_union>, [18] = 0xffffffff00000000, [19] = 0x0 <per_cpu.irq_stack_unio
}

0xFFFF8800BF0701C0: size-192
 - Object size/per slab: 384/10
 - Pages per slab/bytes: 1/4096
 - Objects total/active/free: 2270/2270/0
 - SLABS total/full/partial/free: 227/227/0/0
 - Array caches available shared - cpu1/cpu2/...: 0 - 8/0


Found in slab_full at 0xFFFF8800A3CF7040
$14 = {
  list = {
    next = 0xffff8800ba500000,
    prev = 0xffff8800954f6000
  },
  colouroff = 192,
  s_mem = 0xffff8800a3cf70c0,
  inuse = 10,
  free = 4294967295,
  nodeid = 0
}

Slab info:
-----------------------------------------------
| 0xFFFF8800A3CF7040  | SLAB descriptor      |
-----------------------------------------------
| 0xFFFF8800A3CF7070  | kmem_bufctl_t array  |
-----------------------------------------------
|                                            |
-------------------------------------------------------
| 0xFFFF8800A3CF70C0  | allocated            | NON AC |
-------------------------------------------------------
| 0xFFFF8800A3CF7240  | allocated            | NON AC |
-------------------------------------------------------
| 0xFFFF8800A3CF73C0  | allocated            | NON AC |
-------------------------------------------------------
| 0xFFFF8800A3CF7540  | allocated            | NON AC |
-------------------------------------------------------
| 0xFFFF8800A3CF76C0  | allocated            | NON AC |
-------------------------------------------------------
| 0xFFFF8800A3CF7840  | allocated            | NON AC | <----
-------------------------------------------------------
| 0xFFFF8800A3CF79C0  | allocated            | NON AC |
-------------------------------------------------------
| 0xFFFF8800A3CF7B40  | allocated            | NON AC |
-------------------------------------------------------
| 0xFFFF8800A3CF7CC0  | allocated            | NON AC |
-------------------------------------------------------
| 0xFFFF8800A3CF7E40  | allocated            | NON AC |
-------------------------------------------------------
```

SLAB Graphs 1

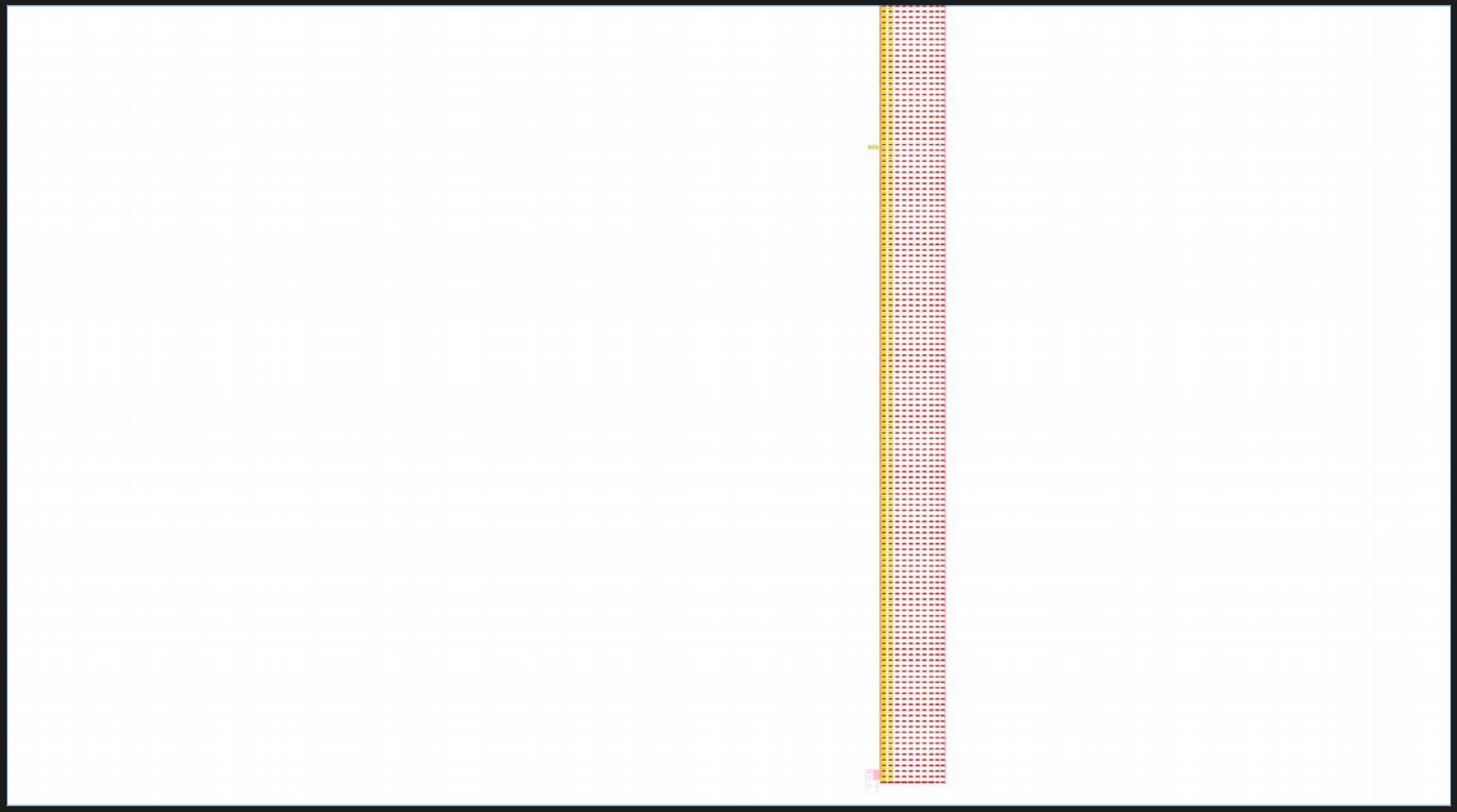


SLAB Graphs 2

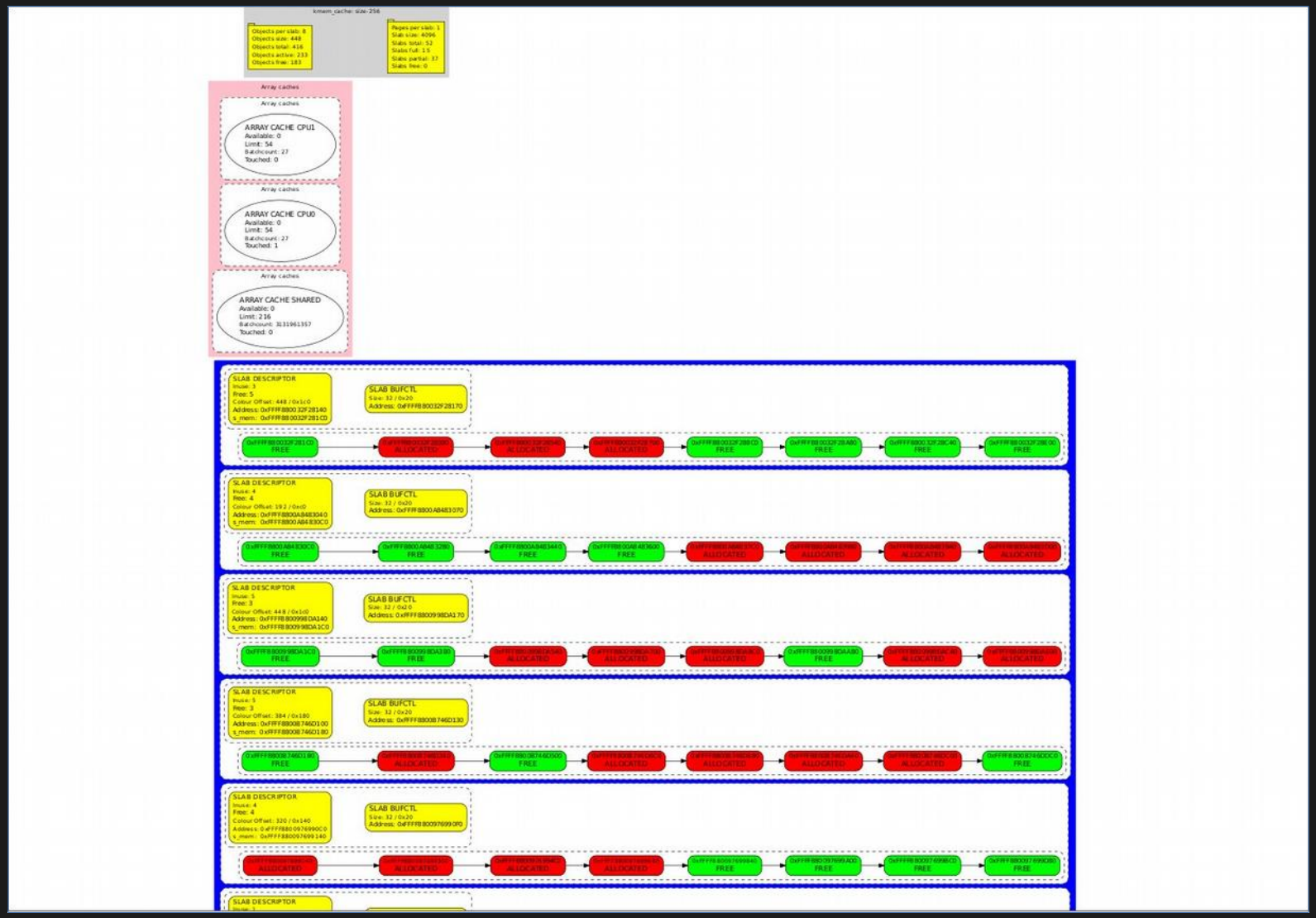


Per CPU Array Caches

SLAB Graphs 3



SLAB Graphs 4



Exploit Primitives

You will end up using same techniques many times when exploiting so creating your own set of libs will also be useful.

Some examples:

- physmap resolver
- UAF allocator/deallocator
- Payloads/Backdoor
- ....

Questions?