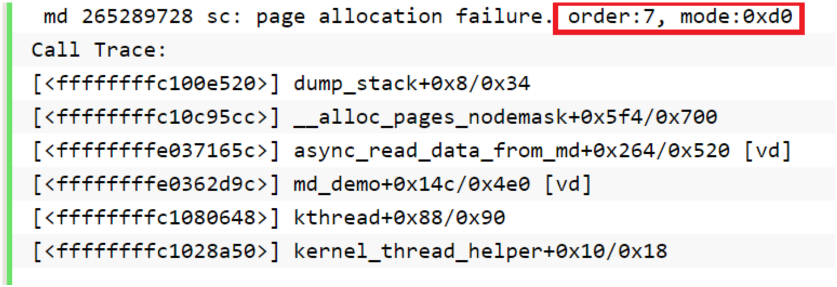
Linux Memory In Practice

May 13, 2016

Low on Memory

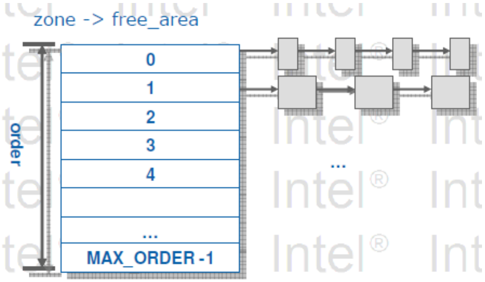
Something like:
char *p = malloc(100);
may fail...ENOMEM
.
.
.
(/proc/sys/vm/overcommit_memory)

Even worse, Out of Memory



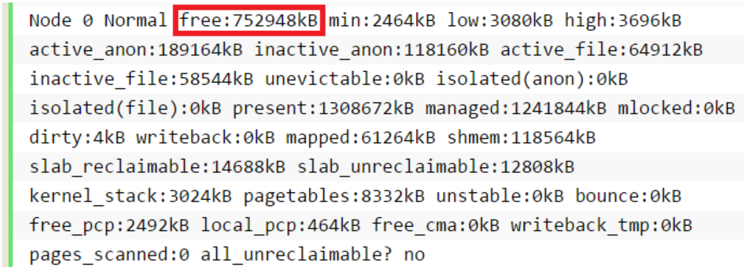
```
 md 265289728 sc: page allocation failure. order:7, mode:0xd0
Call Trace:
[<ffffffffc100e520>] dump_stack+0x8/0x34
[<ffffffffc10c95cc>] __alloc_pages_nodemask+0x5f4/0x700
[<ffffffffe037165c>] async_read_data_from_md+0x264/0x520 [vd]
[<ffffffffe0362d9c>] md_demo+0x14c/0x4e0 [vd]
[<ffffffffc1080648>] kthread+0x88/0x90
[<ffffffffc1028a50>] kernel_thread_helper+0x10/0x18
```

Even worse, Out of Memory



memory usage from sysrq

Note: This is a similar output to OOM..



```
Node 0 Normal free:752948kB min:2464kB low:3080kB high:3696kB
active_anon:189164kB inactive_anon:118160kB active_file:64912kB
inactive_file:58544kB unevictable:0kB isolated(anon):0kB
isolated(file):0kB present:1308672kB managed:1241844kB mlocked:0kB
dirty:4kB writeback:0kB mapped:61264kB shmem:118564kB
slab_reclaimable:14688kB slab_unreclaimable:12808kB
kernel_stack:3024kB pagetables:8332kB unstable:0kB bounce:0kB
free_pcp:2492kB local_pcp:464kB free_cma:0kB writeback_tmp:0kB
pages_scanned:0 all_unreclaimable? no
```

Find the worst offenders

Who was blamed for it?

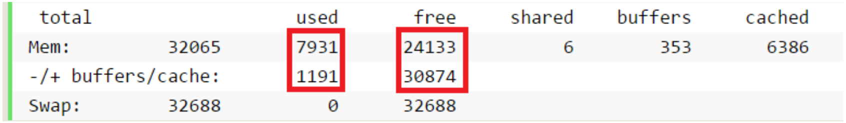
1. How much RAM has kernel used?
2. How much RAM has user space used?

RAM used by kernel - free

```
$ free
              total        used        free      shared  buff/cache
Mem:        3950132      626020     2081976      428552     1242136
Swap:       3906556           0     3906556
```

total = used + free + buff/cache

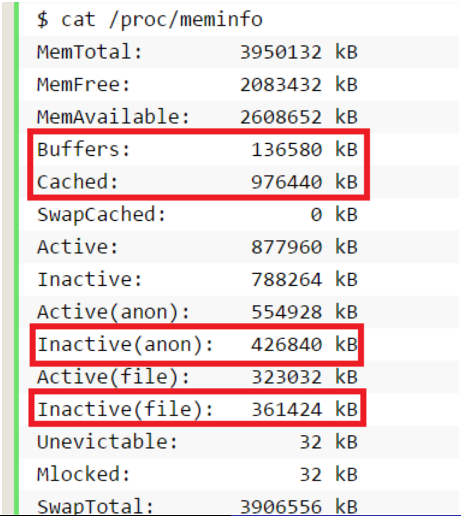
RAM used by kernel - free



```
  total                    used       free     shared    buffers     cached
Mem:         32065         7931      24133          6        353       6386
-/+ buffers/cache:         1191      30874
Swap:        32688            0      32688
```

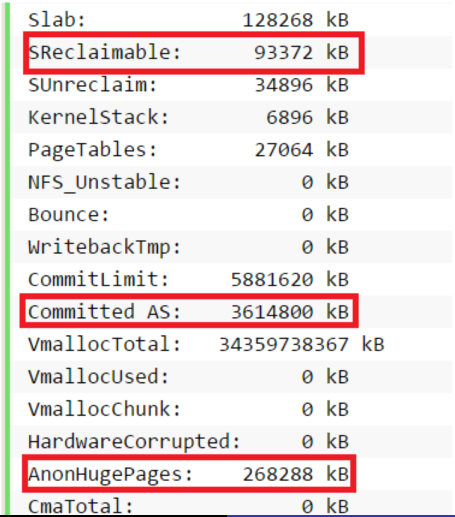
approximately
used - buff/cache: 7931 - 6386 -353
free + buff/cache: 24133 + 6386 + 353

RAM used by kernel - /proc/meminfo



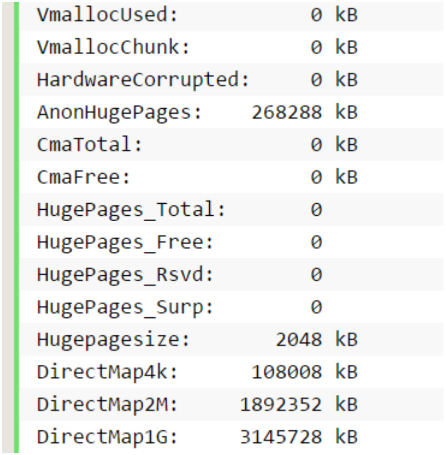
```
$ cat /proc/meminfo
MemTotal:        3950132 kB
MemFree:         2083432 kB
MemAvailable:    2608652 kB
Buffers:          136580 kB
Cached:           976440 kB
SwapCached:            0 kB
Active:           877960 kB
Inactive:         788264 kB
Active(anon):     554928 kB
Inactive(anon):   426840 kB
Active(file):     323032 kB
Inactive(file):   361424 kB
Unevictable:          32 kB
Mlocked:              32 kB
SwapTotal:       3906556 kB
```

RAM used by kernel - /proc/meminfo



```
Slab:             128268 kB
SReclaimable:      93372 kB
SUnreclaim:        34896 kB
KernelStack:        6896 kB
PageTables:        27064 kB
NFS_Unstable:          0 kB
Bounce:                0 kB
WritebackTmp:          0 kB
CommitLimit:     5881620 kB
Committed_AS:    3614800 kB
VmallocTotal:   34359738367 kB
VmallocUsed:           0 kB
VmallocChunk:          0 kB
HardwareCorrupted:     0 kB
AnonHugePages:    268288 kB
CmaTotal:              0 kB
```

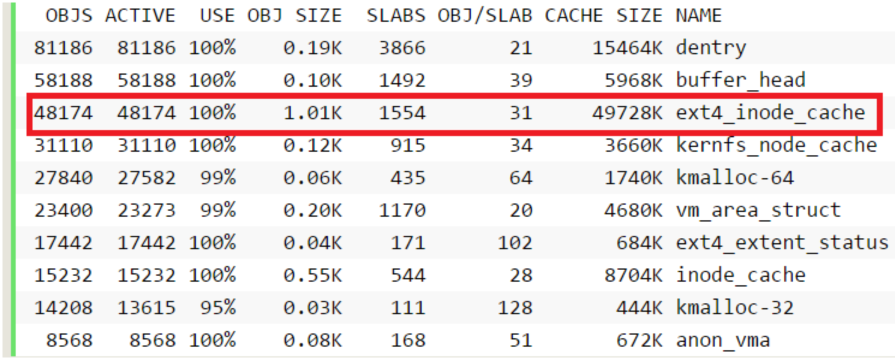
RAM used by kernel - /proc/meminfo



```
VmallocUsed:            0 kB
VmallocChunk:           0 kB
HardwareCorrupted:      0 kB
AnonHugePages:     268288 kB
CmaTotal:               0 kB
CmaFree:                0 kB
HugePages_Total:        0
HugePages_Free:         0
HugePages_Rsvd:         0
HugePages_Surp:         0
Hugepagesize:        2048 kB
DirectMap4k:       108008 kB
DirectMap2M:      1892352 kB
DirectMap1G:      3145728 kB
```

RAM used by kernel - slab?

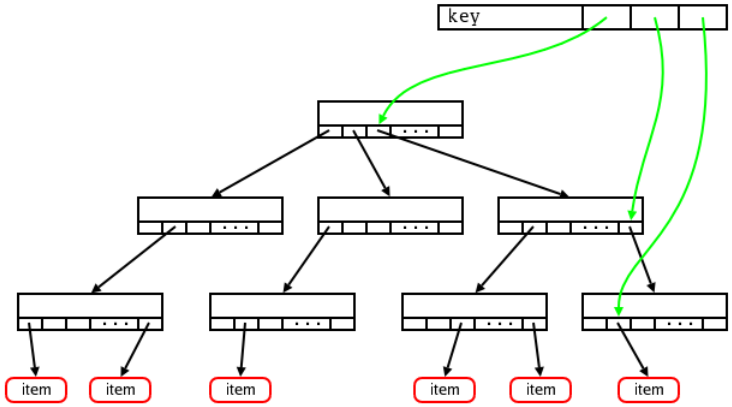
slabtop



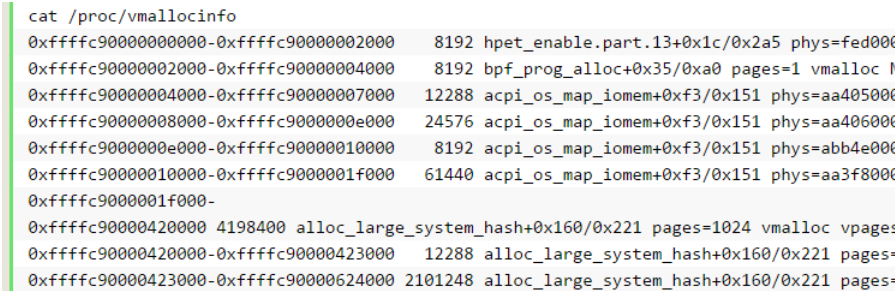
```
  OBJS ACTIVE  USE OBJ SIZE  SLABS OBJ/SLAB CACHE SIZE NAME
 81186  81186 100%    0.19K   3866       21     15464K dentry
 58188  58188 100%    0.10K   1492       39      5968K buffer_head
 48174  48174 100%    1.01K   1554       31     49728K ext4_inode_cache
 31110  31110 100%    0.12K    915       34      3660K kernfs_node_cache
 27840  27582  99%    0.06K    435       64      1740K kmalloc-64
 23400  23273  99%    0.20K   1170       20      4680K vm_area_struct
 17442  17442 100%    0.04K    171      102       684K ext4_extent_status
 15232  15232 100%    0.55K    544       28      8704K inode_cache
 14208  13615  95%    0.03K    111      128       444K kmalloc-32
  8568   8568 100%    0.08K    168       51       672K anon_vma
```

RAM used by kernel - page cache?

page cache: radix tree



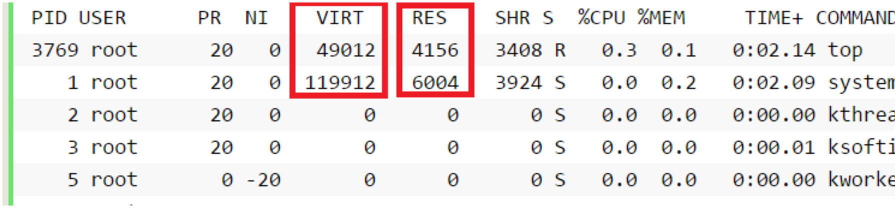
RAM used by kernel - vmalloc?



```
cat /proc/vmallocinfo
0xffffc90000000000-0xffffc90000002000     8192 hpet_enable.part.13+0x1c/0x2a5 phys=fed00
0xffffc90000002000-0xffffc90000004000     8192 bpf_prog_alloc+0x35/0xa0 pages=1 vmalloc
0xffffc90000004000-0xffffc90000007000    12288 acpi_os_map_iomem+0xf3/0x151 phys=aa40500
0xffffc90000008000-0xffffc9000000e000    24576 acpi_os_map_iomem+0xf3/0x151 phys=aa40600
0xffffc9000000e000-0xffffc90000010000     8192 acpi_os_map_iomem+0xf3/0x151 phys=abb4e00
0xffffc90000010000-0xffffc9000001f000    61440 acpi_os_map_iomem+0xf3/0x151 phys=aa3f800
0xffffc9000001f000-
0xffffc90000420000 4198400 alloc_large_system_hash+0x160/0x221 pages=1024 vmalloc vpage
0xffffc90000420000-0xffffc90000423000    12288 alloc_large_system_hash+0x160/0x221 pages=
0xffffc90000423000-0xffffc90000624000 2101248 alloc_large_system_hash+0x160/0x221 pages=
```

cat /proc/vmallocinfo

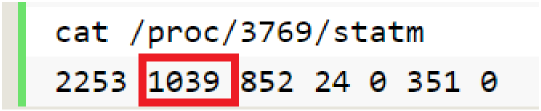
RAM used by process - top



```
  PID USER      PR  NI    VIRT   RES   SHR S  %CPU %MEM     TIME+ COMMAND
 3769 root      20   0   49012  4156  3408 R   0.3  0.1   0:02.14 top
    1 root      20   0  119912  6004  3924 S   0.0  0.2   0:02.09 system
    2 root      20   0       0     0     0 S   0.0  0.0   0:00.00 kthrea
    3 root      20   0       0     0     0 S   0.0  0.0   0:00.01 ksofti
    5 root       0 -20       0     0     0 S   0.0  0.0   0:00.00 kworke
```

top + shift + m: Sort by memory usage

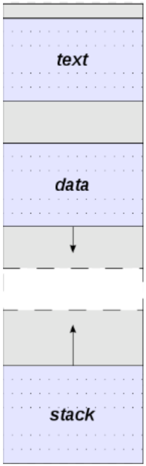
RAM used by process - /proc/pid/statm



```
cat /proc/3769/statm
2253 1039 852 24 0 351 0
```

1039 * 4 = 4156k

RAM used by process - process address space

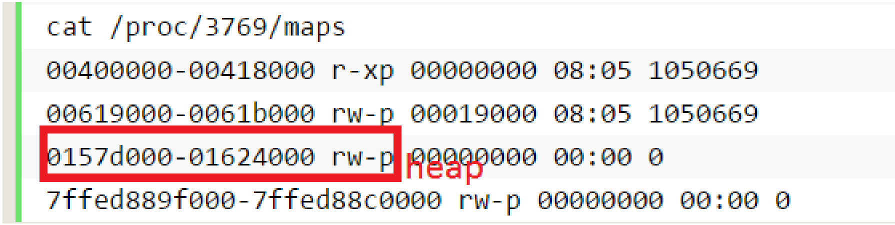


RAM used by process - memory layout

```
0000000000000000 - 00007fffffffffff (=47 bits) user space
ffff880000000000 - ffffc7ffffffffff (=64 TB) direct mapping of all mem
ffffc90000000000 - ffffe8ffffffffff (=45 bits) vmalloc/ioremap space
ffffea0000000000 - ffffeaffffffffff (=40 bits) virtual memory map (1TB)
ffffffff80000000 - ffffffffa0000000 (=512 MB)  kernel text mapping
ffffffffa0000000 - ffffffffff5fffff (=1526 MB) module mapping space
```

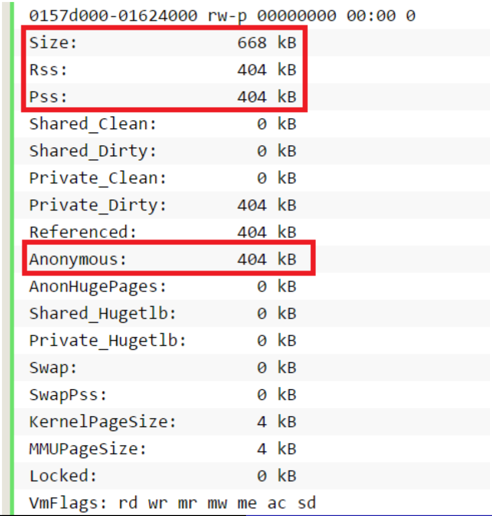
RAM used by process - /proc/pid/maps

Typical process addresss space:



```
cat /proc/3769/maps
00400000-00418000 r-xp 00000000 08:05 1050669
00619000-0061b000 rw-p 00019000 08:05 1050669
0157d000-01624000 rw-p 00000000 00:00 0
7ffed889f000-7ffed88c0000 rw-p 00000000 00:00 0
```

RAM used by process - /proc/pid/smaps

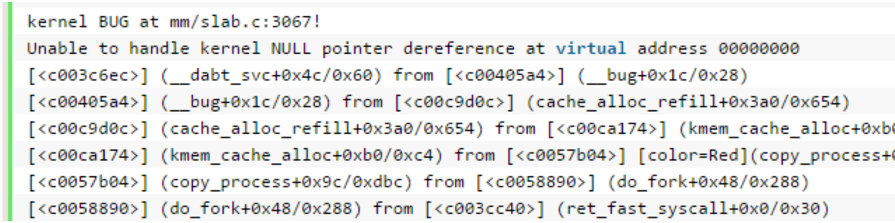


```
0157d000-01624000 rw-p 00000000 00:00 0
Size:                668 kB
Rss:                 404 kB
Pss:                 404 kB
Shared_Clean:          0 kB
Shared_Dirty:          0 kB
Private_Clean:         0 kB
Private_Dirty:       404 kB
Referenced:          404 kB
Anonymous:           404 kB
AnonHugePages:         0 kB
Shared_Hugetlb:        0 kB
Private_Hugetlb:       0 kB
Swap:                  0 kB
SwapPss:               0 kB
KernelPageSize:        4 kB
MMUPageSize:           4 kB
Locked:                0 kB
VmFlags: rd wr mr mw me ac sd
```

kernel detect critical issue

slab caused kernel panic

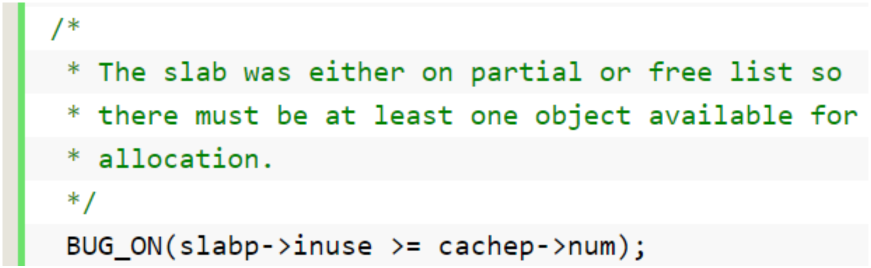
slab bug_on triggered



```
kernel BUG at mm/slab.c:3067!
Unable to handle kernel NULL pointer dereference at virtual address 00000000
[<c003c6ec>] (__dabt_svc+0x4c/0x60) from [<c00405a4>] (__bug+0x1c/0x28)
[<c00405a4>] (__bug+0x1c/0x28) from [<c00c9d0c>] (cache_alloc_refill+0x3a0/0x654)
[<c00c9d0c>] (cache_alloc_refill+0x3a0/0x654) from [<c00ca174>] (kmem_cache_alloc+0xb
[<c00ca174>] (kmem_cache_alloc+0xb0/0xc4) from [<c0057b04>] [color=Red](copy_process+
[<c0057b04>] (copy_process+0x9c/0xdbc) from [<c0058890>] (do_fork+0x48/0x288)
[<c0058890>] (do_fork+0x48/0x288) from [<c003cc40>] (ret_fast_syscall+0x0/0x30)
```

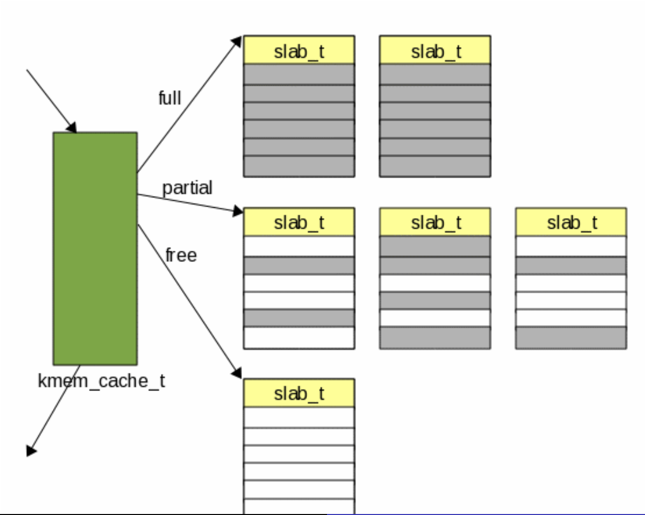
do_fork : copy_process : kmem_cache_alloc : cache_alloc_refill : __bug

slab bug_on triggered

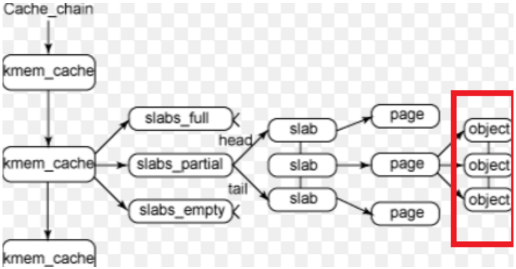


```
/*
 * The slab was either on partial or free list so
 * there must be at least one object available for
 * allocation.
 */
BUG_ON(slabp->inuse >= cachep->num);
```

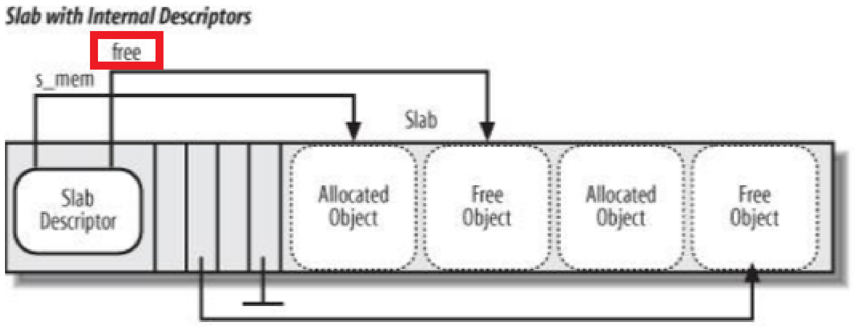
slab structure



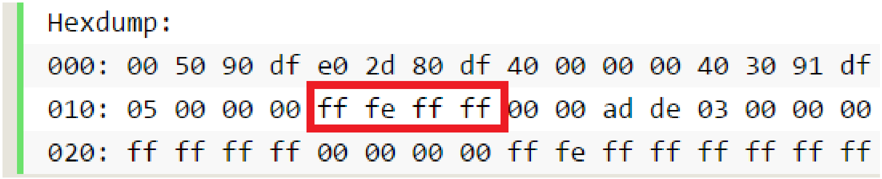
slab objects



slab objects inside



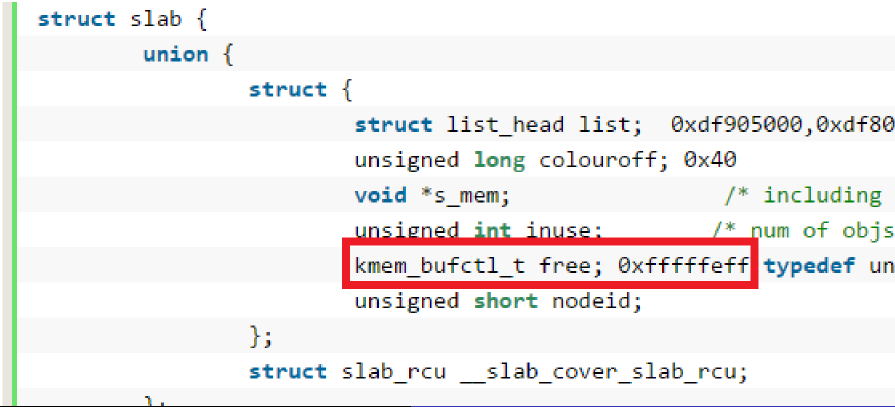
dump of slab



```
Hexdump:
000: 00 50 90 df e0 2d 80 df 40 00 00 00 40 30 91 df
010: 05 00 00 00 ff fe ff ff 00 00 ad de 03 00 00 00
020: ff ff ff ff 00 00 00 00 ff fe ff ff ff ff ff ff
```

slab info

slab.free = ffffffff means there is no free obj in this slab, but fffffeff?



```
struct slab {
        union {
                struct {
                        struct list_head list;  0xdf905000,0xdf80
                        unsigned long colouroff; 0x40
                        void *s_mem;                    /* including
                        unsigned int inuse;             /* num of objs
                        kmem_bufctl_t free; 0xfffffeff typedef un
                        unsigned short nodeid;
                };
                struct slab_rcu __slab_cover_slab_rcu;
        };
```

malloc

What happened when:

```
char *p = malloc(100);//only virtual memory allocated
memset(p, 0, 100);//physical memory allocated
```

Is it a valid kernel address?

How?
virt_addr_valid?

Is it a valid kernel address?

```
0000000000000000 - 00007fffffffffff (=47 bits) user space
ffff880000000000 - ffffc7ffffffffff (=64 TB) direct mapping of all mem
ffffc90000000000 - ffffe8ffffffffff (=45 bits) vmalloc/ioremap space
ffffea0000000000 - ffffeaffffffffff (=40 bits) virtual memory map (1TB)
ffffffff80000000 - ffffffffa0000000 (=512 MB)  kernel text mapping
ffffffffa0000000 - ffffffffff5fffff (=1526 MB) module mapping space
```

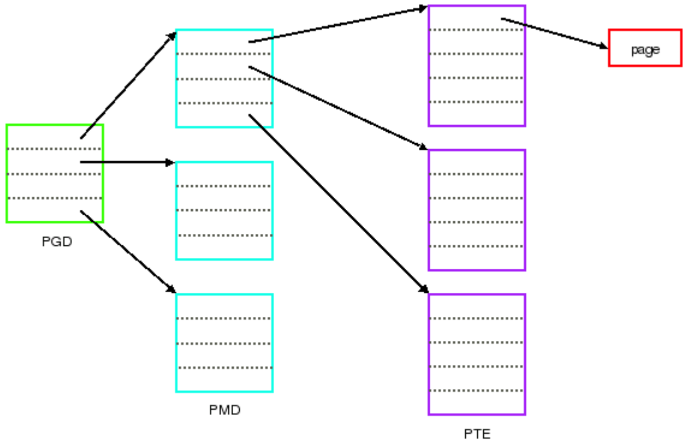
kernel layout

Is it a valid kernel address?

Is it also a safe address to access?
A pagetable walk is required

Is it a valid kernel address?

page walker:



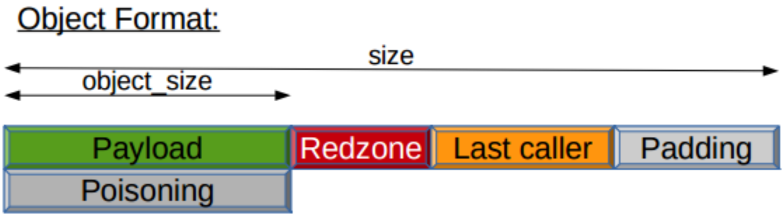
detect memory overflow for page/slab

How to detect memory overflow/overwrite for page/slab?
current implementation in kernel:
1.Post-detection
2.Runtime-detection

Post-detection

How?
SLAB_POISON : avoid double-free
SLAB_RED_ZONE : overflow detection

Post-detection



Runtime-detection

How?
Documentation/kmemcheck.txt
pages = alloc_pages(GFP_KERNEL,1); addr = page_address(pages); if(*addr == 'a') //kmemcheck warn.
kmemcheck is based on page fault and uses extra shadow page
Only support x86

Runtime-detection

Can we write a platform-independent detection method?

Runtime-detection

Let's learn from kmemcheck, we use a rb-tree to maintain all the allocated regions,and add checking-hooks in memcpy/memset/strcpy,etc

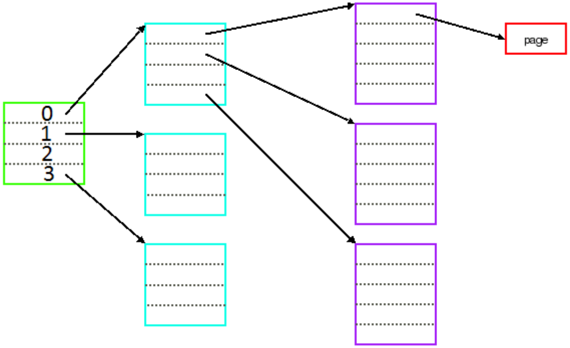
dynamical-size buffer

alloc_page? kmalloc? vmalloc?
ring buffer is fixed-size, and hard to use...
What can we learn from page cache?
convenient read(fd, offset, size)
What can we learn from page table?
only struct page is needed.

dynamical-size buffer

How about mixing the two together?
Let's turn page cache tree node into struct page.

dynamical-size buffer



4 * (4k/sizeof(unsigned long) * (4k/sizeof(unsigned long) * 4k = 4.2G

persistent memory

How can we add a new zone to linux?
Check the git log of ZONE_DEVICE

Reference

- memory layout.
  https://www.kernel.org/doc/Documentation/x86/x86_64/mm.txt
- Understanding the Linux Kernel, 3rd Edition.
  http://gauss.ececs.uc.edu/Courses/c4022/code/memory/understandi
- lwn
  https://lwn.net/Articles
- feel free to contact
  yu.chen.surf@gmail.com

questions

Q/A