Debugging Enterprise Kernels: A Real World Example

Vlastimil Babka
Linux Kernel Developer, SUSE Labs
vbabka@suse.cz

Enterprise Linux Distro (including Kernel)

- The software installation (e.g. ISO) itself is free (and open source, obviously)
- Customers pay for **support subscription**
  - Delivery of (tested by QA) package updates – fixing known bugs, CVE's… but not more!
  - **Getting reported bugs fixed**
    - Bugs specific to customer's workload, hardware, "luck"…
    - Upstream will also fix reported bugs, but only with latest kernel, no guarantees...
- Dealing with kernel bugs, e.g. crashes
  - Find out the root cause (buggy code) with limited possibilities (compared to development)
    - Typically no direct access to customer's system or workload
    - Long turnarounds for providing a modified kernel and reproducing the bug
  - Write and deliver a fix (upstream first!) or workaround fix goes to next update.
    - Possibly a livepatch in some cases
  - Is a lot of fun ;-)

The Real World Example

- In September 2017, a customer reported a kernel crash
  - Kernel detects an unexpected situation and reports it on the console
  - Lots of information which may or may not be enough to find the root cause
  - Kernel might survive and kill just a single process (Oops), but leave the system in a weird state, so it might be better to reboot immediately (Panic) or collect crash dump first

The Real World Example – Kernel Oops

```
kernel BUG at /usr/src/packages/BUILD/kernel-default-3.0.101/linux-3.0/ipc/shm.c:205!
invalid opcode: 0000 [#1] SMP
CPU 1
Modules linked in: lp parport_pc af_packet st ide_cd_mod ide_core bridge stp llc joydev ext2 des_generic ecb md4
nls_utf8 cifs(X) nfs fscache nfsd lockd nfs_acl auth_rpcgss sunrpc autofs4 binfmt_misc mperf vsock(EX) <...>
[last unloaded: ppa]
Supported: Yes, External

Pid: 26341, comm: <redacted> Tainted: G           E X 3.0.101-84-default #1 VMware, Inc. VMware Virtual
Platform/440BX Desktop Reference Platform
RIP: 0010:[<ffffffff811e466e>]  [<ffffffff811e466e>] shm_close+0x3e/0xb0
RSP: 0018:ffff880211337d88  EFLAGS: 00010202
RAX: fffffffffffffea RBX: fffffffffffffea RCX: 0000000000000006
RDX: 0000000000000000 RSI: 000000000000005c RDI: 0000000000000006
RBP: ffffffff81a46920 R08: 0000000000000002 R09: ffff8804256a84d0
R10: ffff880192cecc00 R11: ffffffff81215a80 R12: ffffffff81a469c0
R13: ffff88008cecac80 R14: ffff880423c33740 R15: 0000000000000001
FS:  00007f2945893760(0000) GS:ffff88043fd00000(0000) knlGS:0000000000000000
CS:  0010 DS: 0000 ES: 0000 CR0: 000000008005003b
CR2: 00007f6941216960 CR3: 00000003f3bea000 CR4: 00000000001407e0
DR0: 0000000000000000 DR1: 0000000000000000 DR2: 0000000000000000
DR3: 0000000000000000 DR6: 00000000ffff0ff0 DR7: 0000000000000400
Process <redacted> (pid: 26341, threadinfo ffff880211336000, task ffff8801b49c8040)
```

The Real World Example – Kernel Oops (cont.)

```
Stack:
 ffff88016d2319e0 ffff880158fea140 00007ffce0bf0000 ffffffff81127fa4
 ffff88008cecac80 ffff880211337dd8 ffff88016d2319e0 ffffffff811280f8
 0000000000000001 ffff88014c826088 ffff88008cecac80 0000000100000028
Call Trace:
 [<ffffffff81127fa4>] remove_vma+0x24/0x80
 [<ffffffff811280f8>] exit_mmap+0xf8/0x120
 [<ffffffff810602d9>] mmput+0x49/0x100
 [<ffffffff81065192>] exit_mm+0x122/0x160
 [<ffffffff81066f39>] do_exit+0x189/0x470
 [<ffffffff8106725d>] do_group_exit+0x3d/0xb0
 [<ffffffff810672e2>] sys_exit_group+0x12/0x20
 [<ffffffff81471df2>] system_call_fastpath+0x16/0x1b
 [<00007f29408be998>] 0x7f29408be997
Code: 8b 6b 08 4c 8d a5 a0 00 00 00 4c 89 e7 e8 0b 49 28 00 8b 33 48 8d bd 98 00 00 00 e8
7d ba ff ff 48 3d 00 f0 ff ff 48 89 c3 76 0a <0f> 0b eb fe 66 0f 1f 44 00 00 65 48 8b 04 25
00 a6 00 00 48 8b
RIP  [<ffffffff811e466e>] shm_close+0x3e/0xb0
 RSP <ffff880211337d88>
```

Kernel Oops – Basic Kernel Info

```
kernel BUG at /usr/src/packages/BUILD/kernel-default-3.0.101/linux-3.0/ipc/shm.c:205!
invalid opcode: 0000 [#1] SMP
CPU 1
Modules linked in: lp parport_pc af_packet st ide_cd_mod ide_core bridge stp llc joydev ext2 des_generic ecb md4
nls_utf8 cifs(X) nfs fscache nfsd lockd nfs_acl auth_rpcgss sunrpc autofs4 binfmt_misc mperf vsock(EX) <...>
[last unloaded: ppa]
Supported: Yes, External

Pid: 26341, comm: <redacted> Tainted: G           E X 3.0.101-84-default #1 VMware, Inc. VMware Virtual
Platform/440BX Desktop Reference Platform
RIP: 0010:[<ffffffff811e466e>]  [<ffffffff811e466e>] shm_close+0x3e/0xb0
RSP: 0018:ffff880211337d88  EFLAGS: 00010202
RAX: fffffffffffffea RBX: fffffffffffffea RCX: 0000000000000006
RDX: 0000000000000000 RSI: 000000000000005c RDI: 0000000000000006
RBP: ffffffff81a46920 R08: 0000000000000002 R09: ffff8804256a84d0
R10: ffff880192cecc00 R11: ffffffff81215a80 R12: ffffffff81a469c0
R13: ffff88008cecac80 R14: ffff880423c33740 R15: 0000000000000001
FS:  00007f2945893760(0000) GS:ffff88043fd00000(0000) knlGS:0000000000000000
CS:  0010 DS: 0000 ES: 0000 CR0: 000000008005003b
CR2: 00007f6941216960 CR3: 00000003f3bea000 CR4: 00000000001407e0
DR0: 0000000000000000 DR1: 0000000000000000 DR2: 0000000000000000
DR3: 0000000000000000 DR6: 00000000ffff0ff0 DR7: 0000000000000400
Process <redacted> (pid: 26341, threadinfo ffff880211336000, task ffff8801b49c8040)
```

Kernel Oops – Basic Kernel Info

```
kernel BUG at /usr/src/packages/BUILD/kernel-default-3.0.101/linux-3.0/ipc/shm.c:205!
invalid opcode: 0000 [#1] SMP
CPU 1
Modules linked in: lp parport_pc af_packet st ide_cd_mod ide_core bridge stp llc joydev ext2 des_generic ecb md4
nls_utf8 cifs(X) nfs fscache nfsd lockd nfs_acl auth_rpcgss sunrpc autofs4 binfmt_misc mperf vsock(EX) <...>
[last unloaded: ppa]
Supported: Yes, External

Pid: 26341, comm: <redacted> Tainted: G           E X 3.0.101-84-default #1 VMware, Inc. VMware Virtual
Platform/440BX Desktop Reference Platform
RIP: 0010:[<ffffffff811e466e>]  [<ffffffff811e466e>] shm_clos       0xb0
RSP: 0018:ffff880211337d88  EFLAGS: 00010202
RAX: fffffffffffffea RBX: fffffffffffffea RCX: 0
RDX: 0000000000000000 RSI: 000000000000005c RDI: 0
RBP: ffffffff81a46920 R08: 0000000000000002 R09: f
R10: ffff880192cecc00 R11: ffffffff81215a80 R12: f
R13: ffff88008cecac80 R14: ffff880423c33740 R15: 0
FS:  00007f2945893760(0000) GS:ffff88043fd00000(00
CS:  0010 DS: 0000 ES: 0000 CR0: 000000008005003b
CR2: 00007f6941216960 CR3: 00000003f3bea000 CR4: 0
DR0: 0000000000000000 DR1: 0000000000000000 DR2: 0
DR3: 0000000000000000 DR6: 00000000ffff0ff0 DR7: 0
Process <redacted> (pid: 26341, threadinfo ffff880
```

Exact kernel version (to check out the proper sources)
Upstream plus thousands of patches

Taint flags
G – all modules GPL
E – some modules unsigned
X – some modules externally supported

Kernel Oops – What and Where Sh*t Happened?

```
kernel BUG at /usr/src/packages/BUILD/kernel-default-3.0.101/linux-3.0/ipc/shm.c:205!
invalid opcode: 0000 [#1] SMP
CPU 1
Modules linked in: lp parport_pc af_packet st ide_cd_mod ide_core bridge stp llc joydev ext2 des_generic ecb md4
nls_utf8 cifs(X) nfs fscache nfsd lockd nfs_acl auth_rpcgss sunrpc autofs4 binfmt_misc mperf vsock(EX) <...>
[last unloaded: ppa]
Supported: Yes, External

Pid: 26341, comm: <redacted> Tainted: G           E X 3.0.101-84-default #1 VMware, Inc. VMware Virtual
Platform/440BX Desktop Reference Platform
RIP: 0010:[<ffffffff811e466e>]  [<ffffffff811e466e>] shm_close+0x3e/0xb0
RSP: 0018:ffff880211337d88  EFLAGS: 00010202
RAX: ffffffffffffffea RBX: ffffffffffffffea RCX: 0000000000000006
RDX: 0000000000000000 RSI: 000000000000005c RDI: 0000000000000006
RBP: ffffffff81a46920 R08: 0000000000000002 R09: ffff8804256a84d0
R10: ffff880192cecc00 R11: ffffffff81215a80 R12: ffffffff81a469c0
R13: ffff88008cecac80 R14: ffff880423c33740 R15: 0000000000000001
FS:  00007f2945893760(0000) GS:ffff88043fd00000(0000) knlGS:0000000000000000
CS:  0010 DS: 0000 ES: 0000 CR0: 000000008005003b
CR2: 00007f6941216960 CR3: 00000003f3bea000 CR4: 00000000001407e0
DR0: 0000000000000000 DR1: 0000000000000000 DR2: 0000000000000000
DR3: 0000000000000000 DR6: 00000000ffff0ff0 DR7: 0000000000000400
Process <redacted> (pid: 26341, threadinfo ffff880211336000, task ffff8801b49c8040)
```

Kernel Oops – What and Where Sh*t Happened?

```
kernel BUG at /usr/src/packages/BUILD/kernel-default-3.0.101/linux-3.0/ipc/shm.c:205!
invalid opcode: 0000 [#1] SMP
CPU 1
Modules linked in: lp parport_pc af_packet st ide_cd_mod ide_core bridge stp                xt2 des_generic ecb md4
nls_utf8 cifs(X) nfs fscache nfsd lockd nfs_acl auth_rpcgss sunrpc
[last unloaded: ppa]
Supported: Yes, External

Pid: 26341, comm: <redacted> Tainted: G           E X 3.0.101-84-de
Platform/440BX Desktop Reference Platform
RIP: 0010:[<ffffffff811e466e>]  [<ffffffff811e466e>] shm_close+0x3e
RSP: 0018:ffff880211337d88  EFLAGS: 00010202
RAX: ffffffffffffffea RBX: ffffffffffffffea RCX: 0000000000000006
RDX: 0000000000000000 RSI: 000000000000005c RDI: 0000000000000006
RBP: ffffffff81a46920 R08: 0000000000000002 R09: ffff8804256a84d0
R10: ffff880192cecc00 R11: ffffffff81215a80 R12: ffffffff81a469c0
R13: ffff88008cecac80 R14: ffff880423c33740 R15: 0000000000000001
FS:  00007f2945893760(0000) GS:ffff88043fd00000(0000) knlGS:0000000
CS:  0010 DS: 0000 ES: 0000 CR0: 000000008005003b
CR2: 00007f6941216960 CR3: 00000003f3bea000 CR4: 00000000001407e0
DR0: 0000000000000000 DR1: 0000000000000000 DR2: 0000000000000000
DR3: 0000000000000000 DR6: 00000000ffff0ff0 DR7: 0000000000000400
Process <redacted> (pid: 26341, threadinfo ffff880211336000, task ffff8801b49c8040)
```

The indicated line contains:

```
struct shmid_kernel *shp;
...
shp = shm_lock(ns, sfd->id);
BUG_ON(IS_ERR(shp));
```

This is essentially a hard assertion:
`if (<condition>) BUG()`

Kernel Oops – What and Where Sh*t Happened?

```
kernel BUG at /usr/src/packages/BUILD/kernel-default-3.0.101/linux-3.0/ipc/shm.c:205!
invalid opcode: 0000 [#1] SMP
CPU 1
Modules l       in: lp parport_pc af_packet st ide_cd_mod ide_core bridge stp       xt2 des_generic ecb md4
nls_utf         nfs fscache nfsd lockd nfs_acl auth_rpcgss sunrpc
[last
                                                X 3.0.101-84-de
                                                ] shm_close+0x3e
                                                0000000000006
                                                0000000000006
                                                ff8804256a84d0
                                                ffffff81a469c0
                                                0000000000001
                                                0) knlGS:0000000

                                                000000001407e0
                                                0000000000000
                                                0000000000400
                                                11336000, task ffff8801b49c8040)
```

On x86_64, `BUG()` emits a standardized invalid opcode `UD2` (`0F 0B`) triggering the exception.

The exception handler checks for `UD2` opcode and searches the `__bug_table` for details.

This reduces instruction cache footprint compared to `BUG()` being a call. Also prevents speculation into `BUG()` path.

The indicated line contains:

```
struct shmid_kernel *shp;
...
shp = shm_lock(ns, sfd->id);
BUG_ON(IS_ERR(shp));
```

This is essentially a hard assertion:
`if (<condition>) BUG()`

Kernel Oops – Which Code has Crashed?

```
Stack:
 ffff88016d2319e0 ffff880158fea140 00007ffce0bf0000 ffffffff81127fa4
 ffff88008cecac80 ffff880211337dd8 ffff88016d2319e0 ffffffff811280f8
 0000000000000001 ffff88014c826088 ffff88008cecac80 0000000100000028
Call Trace:
 [<ffffffff81127fa4>] remove_vma+0x24/0x80
 [<ffffffff811280f8>] exit_mmap+0xf8/0x120
 [<ffffffff810602d9>] mmput+0x49/0x100
 [<ffffffff81065192>] exit_mm+0x122/0x160
 [<ffffffff81066f39>] do_exit+0x189/0x470
 [<ffffffff8106725d>] do_group_exit+0x3d/0xb0
 [<ffffffff810672e2>] sys_exit_group+0x12/0x20
 [<ffffffff81471df2>] system_call_fastpath+0x16/0x1b
 [<00007f29408be998>] 0x7f29408be997
Code: 8b 6b 08 4c 8d a5 a0 00 00 00 4c 89 e7 e8 0b 49 28 00 8b 33 48 8d bd 98 00 00 00 e8
7d ba ff ff 48 3d 00 f0 ff ff 48 89 c3 76 0a <0f> 0b eb fe 66 0f 1f 44 00 00 65 48 8b 04 25
00 a6 00 00 48 8b
RIP  [<ffffffff811e466e>] shm_close+0x3e/0xb0
 RSP <ffff880211337d88>
```

Kernel Oops – Which Co

```
Stack:
 ffff88016d2319e0 ffff880158fea140 00007ffce0
 ffff88008cecac80 ffff880211337dd8 ffff88016d
 0000000000000001 ffff88014c826088 ffff88008c
Call Trace:
 [<ffffffff81127fa4>] remove_vma+0x24/0x80
 [<ffffffff811280f8>] exit_mmap+0xf8/0x120
 [<ffffffff810602d9>] mmput+0x49/0x100
 [<ffffffff81065192>] exit_mm+0x122/0x160
 [<ffffffff81066f39>] do_exit+0x189/0x470
 [<ffffffff8106725d>] do_group_exit+0x3d/0xb0
 [<ffffffff810672e2>] sys_exit_group+0x12/0x20
 [<ffffffff81471df2>] system_call_fastpath+0x16/0x1b
 [<00007f29408be998>] 0x7f29408be997
Code: 8b 6b 08 4c 8d a5 a0 00 00 00 4c 89 e7 e8 0b 49 28 00 8b 33 48 8d bd 98 00 00 00 e8
7d ba ff ff 48 3d 00 f0 ff ff 48 89 c3 76 0a <0f> 0b eb fe 66 0f 1f 44 00 00 65 48 8b 04 25
00 a6 00 00 48 8b
RIP  [<ffffffff811e466e>] shm_close+0x3e/0xb0
 RSP <ffff880211337d88>
```

A bunch of instructions around the RIP.
RIP position denoted by < >

Recall that `0F 0B` is opcode for UD2

We can disassemble the code listing by piping the oops into `./scripts/decodecode` in the kernel source tree.

Kernel Oops - ./scripts/decodecode output

```
Code: 8b 6b 08 4c 8d a5 a0 00 00 00 4c 89 e7 e8 0b 49 28 00 8b 33 48 8d bd 98 00 00 00 e8 7d ba ff
ff 48 3d 00 f0 ff ff 48 89 c3 76 0a <0f> 0b eb fe 66 0f 1f 44 00 00 65 48 8b 04 25 00 a6 00 00 48 8b
All code
========
   0:   8b 6b 08                mov    0x8(%rbx),%ebp
   3:   4c 8d a5 a0 00 00 00    lea    0xa0(%rbp),%r12
   a:   4c 89 e7                mov    %r12,%rdi
   d:   e8 0b 49 28 00          callq  0x28491d
  12:   8b 33                   mov    (%rbx),%esi
  14:   48 8d bd 98 00 00 00    lea    0x98(%rbp),%rdi
  1b:   e8 7d ba ff ff          callq  0xffffffffffffba9d
  20:   48 3d 00 f0 ff ff       cmp    $0xfffffffffffff000,%rax
  26:   48 89 c3                mov    %rax,%rbx
  29:   76 0a                   jbe    0x35
  2b:*  0f 0b                   ud2            <-- trapping instruction
  2d:   eb fe                   jmp    0x2d
  2f:   66 0f 1f 44 00 00       nopw   0x0(%rax,%rax,1)
  35:   65 48 8b 04 25 00 a6    mov    %gs:0xa600,%rax
  3c:   00 00
  3e:   48                      rex.W
  3f:   8b                      .byte 0x8b
```

Kernel Oops - ./scripts/decodecode output

```
Code: 8b 6b 08 4c 8d a5 a0 00 00 00 4c 89 e7 e8 0b 49 28 00 8
ff 48 3d 00 f0 ff ff 48 89 c3 76 0a <0f> 0b eb fe 66 0f 1f 44
All code
========
   0:   8b 6b 08                mov    0x8(%rbx),%ebp
   3:   4c 8d a5 a0 00 00 00    lea    0xa0(%rbp),%r12
   a:   4c 89 e7                mov    %r12,%rdi
   d:   e8 0b 49 28 00          callq  0x28491d
  12:   8b 33                   mov    (%rbx),%esi
  14:   48 8d bd 98 00 00 00    lea    0x98(%rbp),%rdi
  1b:   e8 7d ba ff ff          callq  0xffffffffffffba9d
  20:   48 3d 00 f0 ff ff       cmp    $0xfffffffffffff000,%r
  26:   48 89 c3                mov    %rax,%rbx
  29:   76 0a                   jbe    0x35
  2b:*  0f 0b                   ud2            <-- trapping instruction
  2d:   eb fe                   jmp    0x2d
  2f:   66 0f 1f 44 00 00       nopw   0x0(%rax,%rax,1)
  35:   65 48 8b 04 25 00 a6    mov    %gs:0xa600,%rax
  3c:   00 00
  3e:   48                      rex.W
  3f:   8b                      .byte 0x8b
```

```
struct shmid_kernel *shp;
...
shp = shm_lock(ns, sfd->id);
BUG_ON(IS_ERR(shp));
```

`rdi` contained value of `ns`
`rsi` contained value of `sfd→id`
both might be lost now

Kernel Oops - ./scripts/decodecode output

```
Code: 8b 6b 08 4c 8d a5 a0 00 00 00 4c 89 e7 e8 0b 49 28 00 8
ff 48 3d 00 f0 ff ff 48 89 c3 76 0a <0f> 0b eb fe 66 0f 1f 44
All code
========
   0:   8b 6b 08                mov    0x8(%rbx),%ebp
   3:   4c 8d a5 a0 00 00 00    lea    0xa0(%rbp),%r12
   a:   4c 89 e7                mov    %r12,%rdi
   d:   e8 0b 49 28 00          callq  0x28491d
  12:   8b 33                   mov    (%rbx),%esi
  14:   48 8d bd 98 00 00 00    lea    0x98(%rbp),%rdi
  1b:   e8 7d ba ff ff          callq  0xffffffffffffba9d
  20:   48 3d 00 f0 ff ff       cmp    $0xfffffffffffff000,%rax
  26:   48 89 c3                mov    %rax,%rbx
  29:   76 0a                   jbe    0x35
  2b:*  0f 0b                   ud2              <-- trapping instruction
  2d:   eb fe                   jmp    0x2d
  2f:   66 0f 1f 44 00 00       nopw   0x0(%rax,%rax,1)
  35:   65 48 8b 04 25 00 a6    mov    %gs:0xa600,%rax
  3c:   00 00
  3e:   48                      rex.W
  3f:   8b                      .byte 0x8b
```

```
struct shmid_kernel *shp;
...
shp = shm_lock(ns, sfd->id);
BUG_ON(IS_ERR(shp));
```

rax contains value of shp current, not lost

Kernel Oops - ./scripts/decodecode output

```
Code: 8b 6b 08 4c 8d a5 a0 00 00 00 4c 89 e7 e8 0b 49 28 00 8
ff 48 3d 00 f0 ff ff 48 89 c3 76 0a <0f> 0b eb fe 66 0f 1f 44
All code
========
   0:   8b 6b 08                mov    0x8(%rbx),%ebp
   3:   4c 8d a5 a0 00 00 00    lea    0xa0(%rbp),%r12
   a:   4c 89 e7                mov    %r12,%rdi
   d:   e8 0b 49 28 00          callq  0x28491d
  12:   8b 33                   mov    (%rbx),%esi
  14:   48 8d bd 98 00 00 00    lea    0x98(%rbp),%rdi
  1b:   e8 7d ba ff ff          callq  0xffffffffffffba9d
  20:   48 3d 00 f0 ff ff       cmp    $0xfffffffffffff000,%rax
  26:   48 89 c3                mov    %rax,%rbx
  29:   76 0a                   jbe    0x35
  2b:*  0f 0b                   ud2            <-- trapping instruction
  2d:   eb fe                   jmp    0x2d
  2f:   66 0f 1f 44 00 00       nopw   0x0(%rax,%rax,1)
  35:   65 48 8b 04 25 00 a6    mov    %gs:0xa600,%rax
  3c:   00 00
  3e:   48                      rex.W
  3f:   8b                      .byte 0x8b
```

```
struct shmid_kernel *shp;
...
shp = shm_lock(ns, sfd->id);
BUG_ON(IS_ERR(shp));
```

Kernel Oops – Register Contents

```
kernel BUG at /usr/src/packages/BUILD/kernel-default-3.0.101/linu
invalid opcode: 0000 [#1] SMP
CPU 1
Modules linked in: lp parport_pc af_packet st ide_cd_mod ide_core
nls_utf8 cifs(X) nfs fscache nfsd lockd nfs_acl auth_rpcgss sunrp
[last unloaded: ppa]
Supported: Yes, External

Pid: 26341, comm: <redacted> Tainted: G           E X 3.0.101-8
Platform/440BX Desktop Reference Platform
RIP: 0010:[<ffffffff811e466e>]  [<ffffffff811e466e>] shm_clos  0x
RSP: 0018:ffff880211337d88  EFLAGS: 00010202
RAX: fffffffffffffea RBX: fffffffffffffea RCX: 0000000000000006
RDX: 0000000000000000 RSI: 000000000000005c RDI: 0000000000000006
RBP: ffffffff81a46920 R08: 0000000000000002 R09: ffff8804256a84d0
R10: ffff880192cecc00 R11: ffffffff81215a80 R12: ffffffff81a469c0
R13: ffff88008cecac80 R14: ffff880423c33740 R15: 0000000000000001
FS:  00007f2945893760(0000) GS:ffff88043fd00000(0000) knlGS:0000000000000000
CS:  0010 DS: 0000 ES: 0000 CR0: 000000008005003b
CR2: 00007f6941216960 CR3: 00000003f3bea000 CR4: 00000000001407e0
DR0: 0000000000000000 DR1: 0000000000000000 DR2: 0000000000000000
DR3: 0000000000000000 DR6: 00000000ffff0ff0 DR7: 0000000000000400
Process <redacted> (pid: 26341, threadinfo ffff880211336000, task ffff8801b49c8040)
```

`rax` (and `rbx`) is the error value
-22 = -EINVAL
`rdi` is a `ns` pointer? definitely not
`rsi` is `shm->id`? could be?
`rbp`, `r11`, `r12` – kernel code/static data
`r09`, `r10`, `r13`, `r14` – kernel data

Crash Reference Card (64-bit)

Reg	Usage	Saved
RAX	Return value	no
RBX	Local variable	yes
RCX	Argument #4	no
RDX	Argument #3	no
RSI	Argument #2	no
RDI	Argument #1	no
RBP	(Stack base pointer)	yes
RSP	Stack pointer	yes

Reg	Usage	Saved
R8	Argument #5	no
R9	Argument #6	no
R10	Scratch registers	no
R11		no
R12	Local variables	yes
R13		yes
R14		yes
R15		yes

Virtual Memory Layout

0000000000000000 - 00007FFFFFFFFFFF	user space
DEAD0000xxxxxxxx	pointer poisons
FFFF800000000000 - FFFF87FFFFFFFFFF	hypervisor area
FFFF880000000000 - FFFFC7FFFFFFFFFF	direct mapping
FFFFC90000000000 - FFFFE8FFFFFFFFFF	vmalloc/ioremap
FFFFEA0000000000 - FFFFEAFFFFFFFFFF	vmemmap
FFFFFFFF80000000 - FFFFFFFF9FFFFFFF	kernel text+data
FFFFFFFFA0000000 - FFFFFFFFFFxxxxxx	kernel modules
FFFFFFFFFFxxxxxx - FFFFFFFFFF5FFFFF	permanent fixmaps
FFFFFFFFFF600000 - FFFFFFFFFFDFFFFF	vsyscalls (deprecated)

Calling Conventions

	Function	Syscall
1	RDI	RDI
2	RSI	RSI
3	RDX	RDX
4	RCX	R10
5	R8	R8
6	R9	R9
Return	RAX	RAX
RetHi	RDX	–
Syscall number:		RAX

Kernel Oops – Stack Trace

```
Stack:
 ffff88016d2319e0 ffff880158fea140 00007ffce0bf0000 ffffffff81127fa4
 ffff88008cecac80 ffff880211337dd8 ffff88016d2319e0 ffffffff811280f8
 0000000000000001 ffff88014c826088 ffff88008cecac80 0000000100000028
Call Trace:
 [<ffffffff81127fa4>] remove_vma+0x24/0x80
 [<ffffffff811280f8>] exit_mmap+0xf8/0x120
 [<ffffffff810602d9>] mmput+0x49/0x100
 [<ffffffff81065192>] exit_mm+0x122/0x160
 [<ffffffff81066f39>] do_exit+0x189/0x470
 [<ffffffff8106725d>] do_group_exit+0x3d/0xb0
 [<ffffffff810672e2>] sys_exit_group+0x12/0x20
 [<ffffffff81471df2>] system_call_fastpath+0x16/0x1b
 [<00007f29408be998>] 0x7f29408be997
Code: 8b 6b 08 4c 8d a5 a0 00 00 00 4c 89 e7 e8 0b 49 28 00 8b 33 48 8d bd 98 00 00 00 e8
7d ba ff ff 48 3d 00 f0 ff ff 48 89 c3 76 0a <0f> 0b eb fe 66 0f 1f 44 00 00 65 48 8b 04 25
00 a6 00 00 48 8b
RIP  [<ffffffff811e466e>] shm_close+0x3e/0xb0
 RSP <ffff880211337d88>
```

Kernel Oops – Stack Trac

```
Stack:
 ffff88016d2319e0 ffff880158fea140 00007ffce0bf
 ffff88008cecac80 ffff880211337dd8 ffff88016d23
 0000000000000001 ffff88014c826088 ffff88008ced
Call Trace:
 [<ffffffff81127fa4>] remove_vma+0x24/0x80
 [<ffffffff811280f8>] exit_mmap+0xf8/0x120
 [<ffffffff810602d9>] mmput+0x49/0x100
 [<ffffffff81065192>] exit_mm+0x122/0x160
 [<ffffffff81066f39>] do_exit+0x189/0x470
 [<ffffffff8106725d>] do_group_exit+0x3d/0xb0
 [<ffffffff810672e2>] sys_exit_group+0x12/0x20
 [<ffffffff81471df2>] system_call_fastpath+0x16/0x1b
 [<00007f29408be998>] 0x7f29408be997
Code: 8b 6b 08 4c 8d a5 a0 00 00 00 4c 89 e7 e8 0b 49 28 00 8b 33 48 8d bd 98 00 00 00 e8
7d ba ff ff 48 3d 00 f0 ff ff 48 89 c3 76 0a <0f> 0b eb fe 66 0f 1f 44 00 00 65 48 8b 04 25
00 a6 00 00 48 8b
RIP  [<ffffffff811e466e>] shm_close+0x3e/0xb0
 RSP <ffff880211337d88>
```

Backtrace reconstructed by unwinding the stack, showing the return addresses from individual call frames.

Context matters! `shm_close()` is unlikely to be buggy by itself.

Here, task was exiting and cleaning up its memory layout – a list of vma's.
(see /proc/pid/maps for an example)
A vma was backed by shared memory segment, unregistering its usage led to BUG.

Kernel Debugging – General Approach

- First, understand the immediate cause
  - Typically some unexpected/wrong value somewhere in memory
    - NULL pointer access, because certain structure's field was NULL/bogus
    - Page table corruption, SLAB corruption, strange lock value…
  - Here we know `shm_lock()` returned `-EINVAL`, but we don't know yet why
- Second, try to determine what could cause the value
  - Single bit flip? RAM error (yes, they do happen without ECC)
    - Often manifests as multiple different bugs from the same machine
  - Wrong use by upper layers? For example, SLAB corruption is almost never a bug in SLAB code, but e.g. result of double-free of a kernel object allocated from SLAB
- Note: no general and complete recipe
  - Mostly from own experience, or learn from others' analyses
  - Knowing the subsystem helps, still lots of staring into source code of the exact version

Why does shm_lock() return -EINVAL?

```
/*
 * shm_lock_(check_) routines are called in the paths where the rw_mutex
 * is not necessarily held.
 */
static inline struct shmid_kernel *shm_lock(struct ipc_namespace *ns, int id)
{
        struct kern_ipc_perm *ipcp = ipc_lock(&shm_ids(ns), id);

        if (IS_ERR(ipcp))
                return (struct shmid_kernel *)ipcp;

        return container_of(ipcp, struct shmid_kernel, shm_perm);
}
```

Why does `ipc_lock()` return `-EINVAL`?

```
/**
 * ipc_lock - Lock an ipc structure without rw_mutex held
 * @ids: IPC identifier set
 * @id: ipc id to look for
 *
 * Look for an id in the ipc ids idr and lock the associated ipc object.
 *
 * The ipc object is locked on exit.
 */

struct kern_ipc_perm *ipc_lock(struct ipc_ids *ids, int id)
{
        struct kern_ipc_perm *out;
        int lid = ipcid_to_idx(id);

        rcu_read_lock();
        out = idr_find(&ids->ipcs_idr, lid);
        if (out == NULL) {
                rcu_read_unlock();
                return ERR_PTR(-EINVAL);
        }
```

Why does `ipc_lock()` return `-EINVAL`?

```
        spin_lock(&out->lock);

        /* ipc_rmid() may have already freed the ID while ipc_lock
         * was spinning: here verify that the structure is still valid
         */
        if (out->deleted) {
                spin_unlock(&out->lock);
                rcu_read_unlock();
                return ERR_PTR(-EINVAL);
        }

        return out;
}
```

Why does `ipc_lock()` return `-EINVAL`?

- Either `idr_find()` didn’t find the `shmid`, or it was already deleted
- The oops report can’t help anymore, need to inspect memory
- For that, customers enable collecting of **kernel crash dump (vmcore)**
  - kdump mechanism based on kexec – separate crash kernel is preloaded with own reserved memory, takes over after panic, captures RAM and saves it
- **crash** tool for inspection of vmcore (with vmlinux and debuginfo)
  - Created by David Anderson from Red Hat
  - Understands some kernel internals of a large range of kernel version
    - Memory mappings, SLAB allocator, tasks, linked lists, trees…
    - See backtraces (no variable values via debuginfo, though), dmesg, memory info
    - Interpret memory as structures, search memory, determine what kind of object is at given address ...

crash – shm_close() disassembly

```
crash> dis -lr ffffffff811e466e
/usr/src/debug/kernel-default-3.0.101/linux-3.0/ipc/shm.c: 196
0xffffffff811e4630 <shm_close>:         push   %r12
0xffffffff811e4632 <shm_close+2>:       push   %rbp
0xffffffff811e4633 <shm_close+3>:       push   %rbx
/usr/src/debug/kernel-default-3.0.101/linux-3.0/ipc/shm.c: 198
0xffffffff811e4634 <shm_close+4>:       mov    0x98(%rdi),%rax
0xffffffff811e463b <shm_close+11>:      mov    0xa0(%rax),%rbx
/usr/src/debug/kernel-default-3.0.101/linux-3.0/ipc/shm.c: 200
0xffffffff811e4642 <shm_close+18>:      mov    0x8(%rbx),%rbp
/usr/src/debug/kernel-default-3.0.101/linux-3.0/ipc/shm.c: 202
0xffffffff811e4646 <shm_close+22>:      lea    0xa0(%rbp),%r12
0xffffffff811e464d <shm_close+29>:      mov    %r12,%rdi
0xffffffff811e4650 <shm_close+32>:      callq  0xffffffff81468f60 <down_write>
/usr/src/debug/kernel-default-3.0.101/linux-3.0/ipc/shm.c: 125
0xffffffff811e4655 <shm_close+37>:      mov    (%rbx),%esi
0xffffffff811e4657 <shm_close+39>:      lea    0x98(%rbp),%rdi
0xffffffff811e465e <shm_close+46>:      callq  0xffffffff811e00e0 <ipc_lock>
/usr/src/debug/kernel-default-3.0.101/linux-3.0/ipc/shm.c: 127
0xffffffff811e4663 <shm_close+51>:      cmp    $0xfffffffffffff000,%rax
/usr/src/debug/kernel-default-3.0.101/linux-3.0/ipc/shm.c: 125
0xffffffff811e4669 <shm_close+57>:      mov    %rax,%rbx
/usr/src/debug/kernel-default-3.0.101/linux-3.0/ipc/shm.c: 127
0xffffffff811e466c <shm_close+60>:      jbe    0xffffffff811e4678 <shm_close+72>
/usr/src/debug/kernel-default-3.0.101/linux-3.0/ipc/shm.c: 205
0xffffffff811e466e <shm_close+62>:      ud2
```

crash – shm_close() disas

```
shm_close(struct vm_area_struct *vma)

vma pointer is in rdi
```

```
crash> dis -lr ffffffff811e466e
/usr/src/debug/kernel-default-3.0.101/linux-3.0/ipc/shm.c: 196
0xffffffff811e4630 <shm_close>:         push   %r12
0xffffffff811e4632 <shm_close+2>:       push   %rbp
0xffffffff811e4633 <shm_close+3>:       push   %rbx
/usr/src/debug/kernel-default-3.0.101/linux-3.0/ipc/shm.c: 198
0xffffffff811e4634 <shm_close+4>:       mov    0x98(%rdi),%rax
0xffffffff811e463b <shm_close+11>:      mov    0xa0(%rax),%rbx
/usr/src/debug/kernel-default-3.0.101/linux-3.0/ipc/shm.c: 200
0xffffffff811e4642 <shm_close+18>:      mov    0x8(%rbx),%rbp
/usr/src/debug/kernel-default-3.0.101/linux-3.0/ipc/shm.c: 202
0xffffffff811e4646 <shm_close+22>:      lea    0xa0(%rbp),%r12
0xffffffff811e464d <shm_close+29>:      mov    %r12,%rdi
0xffffffff811e4650 <shm_close+32>:      callq  0xffffffff81468f60 <down_write>
/usr/src/debug/kernel-default-3.0.101/linux-3.0/ipc/shm.c: 125
0xffffffff811e4655 <shm_close+37>:      mov    (%rbx),%esi
0xffffffff811e4657 <shm_close+39>:      lea    0x98(%rbp),%rdi
0xffffffff811e465e <shm_close+46>:      callq  0xffffffff811e00e0 <ipc_lock>
/usr/src/debug/kernel-default-3.0.101/linux-3.0/ipc/shm.c: 127
0xffffffff811e4663 <shm_close+51>:      cmp    $0xfffffffffffff000,%rax
/usr/src/debug/kernel-default-3.0.101/linux-3.0/ipc/shm.c: 125
0xffffffff811e4669 <shm_close+57>:      mov    %rax,%rbx
/usr/src/debug/kernel-default-3.0.101/linux-3.0/ipc/shm.c: 127
0xffffffff811e466c <shm_close+60>:      jbe    0xffffffff811e4678 <shm_close+72>
/usr/src/debug/kernel-default-3.0.101/linux-3.0/ipc/shm.c: 205
0xffffffff811e466e <shm_close+62>:      ud2
```

crash – shm_close() disas

```
crash> dis -lr ffffffff811e466e
/usr/src/debug/kernel-default-3.0.101/linux-3.0/ipc/shm.c: 196
0xffffffff811e4630 <shm_close>:         push   %r12
0xffffffff811e4632 <shm_close+2>:       push   %rbp
0xffffffff811e4633 <shm_close+3>:       push   %rbx
/usr/src/debug/kernel-default-3.0.101/linux-3.0/ipc/shm.c: 198
0xffffffff811e4634 <shm_close+4>:       mov    0x98(%rdi),%rax
0xffffffff811e463b <shm_close+11>:      mov    0xa0(%rax),%rbx
/usr/src/debug/kernel-default-3.0.101/linux-3.0/ipc/shm.c: 200
0xffffffff811e4642 <shm_close+18>:      mov    0x8(%rbx),%rbp
/usr/src/debug/kernel-default-3.0.101/linux-3.0/ipc/shm.c: 202
0xffffffff811e4646 <shm_close+22>:      lea    0xa0(%rbp),%r12
0xffffffff811e464d <shm_close+29>:      mov    %r12,%rdi
0xffffffff811e4650 <shm_close+32>:      callq  0xffffffff81468f60 <down_write>
/usr/src/debug/kernel-default-3.0.101/linux-3.0/ipc/shm.c: 125
0xffffffff811e4655 <shm_close+37>:      mov    (%rbx),%esi
0xffffffff811e4657 <shm_close+39>:      lea    0x98(%rbp),%rdi
0xffffffff811e465e <shm_close+46>:      callq  0xffffffff811e00e0 <ipc_lock>
/usr/src/debug/kernel-default-3.0.101/linux-3.0/ipc/shm.c: 127
0xffffffff811e4663 <shm_close+51>:      cmp    $0xfffffffffffff000,%rax
/usr/src/debug/kernel-default-3.0.101/linux-3.0/ipc/shm.c: 125
0xffffffff811e4669 <shm_close+57>:      mov    %rax,%rbx
/usr/src/debug/kernel-default-3.0.101/linux-3.0/ipc/shm.c: 127
0xffffffff811e466c <shm_close+60>:      jbe    0xffffffff811e4678 <shm_close+72>
/usr/src/debug/kernel-default-3.0.101/linux-3.0/ipc/shm.c: 205
0xffffffff811e466e <shm_close+62>:      ud2
```

```
shm_close(struct vm_area_struct *vma)
```

vma pointer is in `rdi`

```
struct file * file = vma->vm_file;
```

`file` pointer is in `rax`

crash – shm_close() disas

```
crash> dis -lr ffffffff811e466e
/usr/src/debug/kernel-default-3.0.101/linux-3.0/ipc/shm.c: 196
0xffffffff811e4630 <shm_close>:         push   %r12
0xffffffff811e4632 <shm_close+2>:       push   %rbp
0xffffffff811e4633 <shm_close+3>:       push   %rbx
/usr/src/debug/kernel-default-3.0.101/linux-3.0/ipc/shm.c: 198
0xffffffff811e4634 <shm_close+4>:       mov    0x98(%rdi),%rax
0xffffffff811e463b <shm_close+11>:      mov    0xa0(%rax),%rbx
/usr/src/debug/kernel-default-3.0.101/linux-3.0/ipc/shm.c: 200
0xffffffff811e4642 <shm_close+18>:      mov    0x8(%rbx),%rbp
/usr/src/debug/kernel-default-3.0.101/linux-3.0/ipc/shm.c: 202
0xffffffff811e4646 <shm_close+22>:      lea    0xa0(%rbp),%r12
0xffffffff811e464d <shm_close+29>:      mov    %r12,%rdi
0xffffffff811e4650 <shm_close+32>:      callq  0xffffffff81468f60 <down_write>
/usr/src/debug/kernel-default-3.0.101/linux-3.0/ipc/shm.c: 125
0xffffffff811e4655 <shm_close+37>:      mov    (%rbx),%esi
0xffffffff811e4657 <shm_close+39>:      lea    0x98(%rbp),%rdi
0xffffffff811e465e <shm_close+46>:      callq  0xffffffff811e00e0 <ipc_lock>
/usr/src/debug/kernel-default-3.0.101/linux-3.0/ipc/shm.c: 127
0xffffffff811e4663 <shm_close+51>:      cmp    $0xfffffffffffff000,%rax
/usr/src/debug/kernel-default-3.0.101/linux-3.0/ipc/shm.c: 125
0xffffffff811e4669 <shm_close+57>:      mov    %rax,%rbx
/usr/src/debug/kernel-default-3.0.101/linux-3.0/ipc/shm.c: 127
0xffffffff811e466c <shm_close+60>:      jbe    0xffffffff811e4678 <shm_close+72>
/usr/src/debug/kernel-default-3.0.101/linux-3.0/ipc/shm.c: 205
0xffffffff811e466e <shm_close+62>:      ud2
```

```
shm_close(struct vm_area_struct *vma)

vma pointer is in rdi
```

```
struct file * file = vma->vm_file;

file pointer is in rax
```

```
struct shm_file_data *sfd =
                    shm_file_data(file);
sfd pointer is in rbx
```

crash – shm_close() disas

```
crash> dis -lr ffffffff811e466e
/usr/src/debug/kernel-default-3.0.101/linux-3.0/ipc/shm.c: 196
0xffffffff811e4630 <shm_close>:         push   %r12
0xffffffff811e4632 <shm_close+2>:       push   %rbp
0xffffffff811e4633 <shm_close+3>:       push   %rbx
/usr/src/debug/kernel-default-3.0.101/linux-3.0/ipc/shm.c: 198
0xffffffff811e4634 <shm_close+4>:       mov    0x98(%rdi),%rax
0xffffffff811e463b <shm_close+11>:      mov    0xa0(%rax),%rbx
/usr/src/debug/kernel-default-3.0.101/linux-3.0/ipc/shm.c: 200
0xffffffff811e4642 <shm_close+18>:      mov    0x8(%rbx),%rbp
/usr/src/debug/kernel-default-3.0.101/linux-3.0/ipc/shm.c: 202
0xffffffff811e4646 <shm_close+22>:      lea    0xa0(%rbp),%r12
0xffffffff811e464d <shm_close+29>:      mov    %r12,%rdi
0xffffffff811e4650 <shm_close+32>:      callq  0xffffffff8146
/usr/src/debug/kernel-default-3.0.101/linux-3.0/ipc/shm.c: 125
0xffffffff811e4655 <shm_close+37>:      mov    (%rbx),%esi
0xffffffff811e4657 <shm_close+39>:      lea    0x98(%rbp),%rdi
0xffffffff811e465e <shm_close+46>:      callq  0xffffffff811e00e0
/usr/src/debug/kernel-default-3.0.101/linux-3.0/ipc/shm.c: 127
0xffffffff811e4663 <shm_close+51>:      cmp    $0xfffffffffffff000,%rax
/usr/src/debug/kernel-default-3.0.101/linux-3.0/ipc/shm.c: 125
0xffffffff811e4669 <shm_close+57>:      mov    %rax,%rbx
/usr/src/debug/kernel-default-3.0.101/linux-3.0/ipc/shm.c: 127
0xffffffff811e466c <shm_close+60>:      jbe    0xffffffff811e4678 <shm_close+72>
/usr/src/debug/kernel-default-3.0.101/linux-3.0/ipc/shm.c: 205
0xffffffff811e466e <shm_close+62>:      ud2
```

```
shm_close(struct vm_area_struct *vma)

~~vma pointer is in rdi~~
```

```
struct file * file = vma->vm_file;

file pointer is in rax (?)
```

```
struct shm_file_data *sfd =
                    shm_file_data(file);
sfd pointer is in rbx
```

```
struct ipc_namespace *ns = sfd->ns;
down_write(&shm_ids(ns).rw_mutex);
we just lost vma in rdi
we might have lost *file in rax
```

crash – shm_close() disas

```
crash> dis -lr ffffffff811e466e
/usr/src/debug/kernel-default-3.0.101/linux-3.0/ipc/shm.c: 196
0xffffffff811e4630 <shm_close>:         push   %r12
0xffffffff811e4632 <shm_close+2>:       push   %rbp
0xffffffff811e4633 <shm_close+3>:       push   %rbx
/usr/src/debug/kernel-default-3.0.101/linux-3.0/ipc/shm.c: 198
0xffffffff811e4634 <shm_close+4>:       mov    0x98(%rdi),%rax
0xffffffff811e463b <shm_close+11>:      mov    0xa0(%rax),%rbx
/usr/src/debug/kernel-default-3.0.101/linux-3.0/ipc/shm.c: 200
0xffffffff811e4642 <shm_close+18>:      mov    0x8(%rbx),%rbp
/usr/src/debug/kernel-default-3.0.101/linux-3.0/ipc/shm.c: 202
0xffffffff811e4646 <shm_close+22>:      lea    0xa0(%rbp),%r12
0xffffffff811e464d <shm_close+29>:      mov    %r12,%rdi
0xffffffff811e4650 <shm_close+32>:      callq  0xffffffff81468f60 <down
/usr/src/debug/kernel-default-3.0.101/linux-3.0/ipc/shm.c: 125
0xffffffff811e4655 <shm_close+37>:      mov    (%rbx),%esi
0xffffffff811e4657 <shm_close+39>:      lea    0x98(%rbp),%rdi
0xffffffff811e465e <shm_close+46>:      callq  0xffffffff811e00e0 <ipc_
/usr/src/debug/kernel-default-3.0.101/linux-3.0/ipc/shm.c: 127
0xffffffff811e4663 <shm_close+51>:      cmp    $0xfffffffffffff000,%rax
/usr/src/debug/kernel-default-3.0.101/linux-3.0/ipc/shm.c: 125
0xffffffff811e4669 <shm_close+57>:      mov    %rax,%rbx
/usr/src/debug/kernel-default-3.0.101/linux-3.0/ipc/shm.c: 127
0xffffffff811e466c <shm_close+60>:      jbe    0xffffffff811e4678 <shm_close+72>
/usr/src/debug/kernel-default-3.0.101/linux-3.0/ipc/shm.c: 205
0xffffffff811e466e <shm_close+62>:      ud2
```

`shm_close(struct vm_area_struct *vma)`

~~vma pointer is in rdi~~

`struct file * file = vma->vm_file;`

~~file pointer is in rax~~

```
struct shm_file_data *sfd =
                    shm_file_data(file);
```

sfd pointer is in rbx

```
shp = shm_lock(ns, sfd->id);
```

shm_lock(ns, id) is inlined and does:

```
struct kern_ipc_perm *ipcp =
    ipc_lock(&shm_ids(ns), id);
```

rsi has sfd->id which we want to know

rdi has &shm_ids(ns)

we definitely lost rax now

crash – shm_close() disas

```
crash> dis -lr ffffffff811e466e
/usr/src/debug/kernel-default-3.0.101/linux-3.0/ipc/shm.c: 196
0xffffffff811e4630 <shm_close>:         push   %r12
0xffffffff811e4632 <shm_close+2>:       push   %rbp
0xffffffff811e4633 <shm_close+3>:       push   %rbx
/usr/src/debug/kernel-default-3.0.101/linux-3.0/ipc/shm.c: 198
0xffffffff811e4634 <shm_close+4>:       mov    0x98(%rdi),%rax
0xffffffff811e463b <shm_close+11>:      mov    0xa0(%rax),%rbx
/usr/src/debug/kernel-default-3.0.101/linux-3.0/ipc/shm.c: 200
0xffffffff811e4642 <shm_close+18>:      mov    0x8(%rbx),%rbp
/usr/src/debug/kernel-default-3.0.101/linux-3.0/ipc/shm.c: 202
0xffffffff811e4646 <shm_close+22>:      lea    0xa0(%rbp),%r12
0xffffffff811e464d <shm_close+29>:      mov    %r12,%rdi
0xffffffff811e4650 <shm_close+32>:      callq  0xffffffff81468f60 <down_write>
/usr/src/debug/kernel-default-3.0.101/linux-3.0/ipc/shm.c: 125
0xffffffff811e4655 <shm_close+37>:      mov    (%rbx),%esi
0xffffffff811e4657 <shm_close+39>:      lea    0x98(%rbp),%rdi
0xffffffff811e465e <shm_close+46>:      callq  0xffffffff811e00e0 <ipc_lock>
/usr/src/debug/kernel-default-3.0.101/linux-3.0/ipc/shm.c: 127
0xffffffff811e4663 <shm_close+51>:      cmp    $0xfffffffffffff000,%rax
/usr/src/debug/kernel-default-3.0.101/linux-3.0/ipc/shm.c: 125
0xffffffff811e4669 <shm_close+57>:      mov    %rax,%rbx
/usr/src/debug/kernel-default-3.0.101/linux-3.0/ipc/shm.c: 127
0xffffffff811e466c <shm_close+60>:      jbe    0xffffffff811e4678 <shm_close+72>
/usr/src/debug/kernel-default-3.0.101/linux-3.0/ipc/shm.c: 205
0xffffffff811e466e <shm_close+62>:      ud2
```

```
shm_close(struct vm_area_struct *vma)

~~vma pointer is in rdi~~
```

```
struct file * file = vma->vm_file;

~~file pointer is in rax~~
```

```
struct shm_file_data *sfd =
                    shm_file_data(file);
~~sfd pointer is in rbx~~
```

we lost `sfd` in `rbx` :(

Getting the vma pointer

- We lost the `vma` pointer in `shm_close()` and all intermediate pointers leading to `shmid`
  - Checking `ipc_lock()` asm also showed that `shmid` gets quickly lost inside it
- There's a chance some function up the call stack saved the `vma`, but how to find it without following more assembler?
  - `vma`'s (`struct vm_area_struct` objects) have a dedicated SLAB cache, and crash can mark these caches automatically in the backtrace

crash – rich backtrace with local data

```
crash> bt -FF
...
 #4 [ffff880211337cd0] invalid_op at ffffffff81472ddb
    [exception RIP: shm_close+62]
    RIP: ffffffff811e466e  RSP: ffff880211337d88  RFLAGS: 00010202
    RAX: ffffffffffffffea  RBX: ffffffffffffffea  RCX: 0000000000000006
    RDX: 0000000000000000  RSI: 000000000000005c  RDI: 0000000000000006
    RBP: ffffffff81a46920   R8: 0000000000000002   R9: ffff8804256a84d0
    R10: ffff880192cecc00  R11: ffffffff81215a80  R12: ffffffff81a469c0
    R13: ffff88008cecac80  R14: ffff880423c33740  R15: 0000000000000001
    ORIG_RAX: ffffffffffffffff  CS: 0010  SS: 0018
    ffff880211337cd8: 0000000000000001 [ffff880423c33740:signal_cache]
    ffff880211337ce8: [ffff88008cecac80:mm_struct] init_ipc_ns+160
    ffff880211337cf8: init_ipc_ns      ffffffffffffffea
    ffff880211337d08: apparmor_file_free_security [ffff880192cecc00:size-32]
    ffff880211337d18: [ffff8804256a84d0:idr_layer_cache] 0000000000000002
    ffff880211337d28: ffffffffffffffea 0000000000000006
    ffff880211337d38: 0000000000000000 000000000000005c
    ffff880211337d48: 0000000000000006 ffffffffffffffff
    ffff880211337d58: shm_close+62     0000000000000010
    ffff880211337d68: 0000000000010202 ffff880211337d88
    ffff880211337d78: 0000000000000018 shm_close+51
    ffff880211337d88: [ffff88016d2319e0:vm_area_struct] [ffff880158fea140:vm_area_struct]
    ffff880211337d98: 00007ffce0bf0000 remove_vma+36
 #5 [ffff880211337da0] remove_vma at ffffffff81127fa4
    ffff880211337da8: [ffff88008cecac80:mm_struct] ffff880211337dd8
    ffff880211337db8: [ffff88016d2319e0:vm_area_struct] exit_mmap+248
 #6 [ffff880211337dc0] exit_mmap at ffffffff811280f8
```

crash – rich backtrace with lo

```
crash> bt -FF
...
 #4 [ffff880211337cd0] invalid_op at ffffffff81472ddb
    [exception RIP: shm_close+62]
    RIP: ffffffff811e466e  RSP: ffff880211337d88  RFLAGS: 00010202
    RAX: ffffffffffffffea  RBX: ffffffffffffffea  RCX: 0000000000000006
    RDX: 0000000000000000  RSI: 000000000000005c  RDI: 0000000000000006
    RBP: ffffffff81a46920   R8: 0000000000000002   R9: ffff8804256a84d0
    R10: ffff880192cecc00  R11: ffffffff81215a80  R12: ffffffff81a469c0
    R13: ffff88008cecac80  R14: ffff880423c33740  R15: 0000000000000001
    ORIG_RAX: ffffffffffffffff  CS: 0010  SS: 0018
    ffff880211337cd8: 0000000000000001 [ffff880423c33740:signal_cache]
    ffff880211337ce8: [ffff88008cecac80:mm_struct] init_ipc_ns+160
    ffff880211337cf8: init_ipc_ns      ffffffffffffffea
    ffff880211337d08: apparmor_file_free_security [ffff880192cecc00:size-32]
    ffff880211337d18: [ffff8804256a84d0:idr_layer_cache] 0000000000000002
    ffff880211337d28: ffffffffffffffea 0000000000000006
    ffff880211337d38: 0000000000000000 000000000000005c
    ffff880211337d48: 0000000000000006 ffffffffffffffff
    ffff880211337d58: shm_close+62     0000000000000010
    ffff880211337d68: 0000000000010202 ffff880211337d88
    ffff880211337d78: 0000000000000018 shm_close+51
    ffff880211337d88: [ffff88016d2319e0:vm_area_struct] [ffff880158fea140:vm_area_struct]
    ffff880211337d98: 00007ffce0bf0000 remove_vma+36
 #5 [ffff880211337da0] remove_vma at ffffffff81127fa4
    ffff880211337da8: [ffff88008cecac80:mm_struct] ffff880211337dd8
    ffff880211337db8: [ffff88016d2319e0:vm_area_struct] exit_mmap+248
 #6 [ffff880211337dc0] exit_mmap at ffffffff811280f8
```

```
remove_vma(vma) – vma in rdi
<remove_vma>:          push   %rbp
<remove_vma+1>:        push   %rbx
<remove_vma+2>:        mov    %rdi,%rbx
<remove_vma+5>:        sub    $0x8,%rsp
<remove_vma+9>:        mov    0x88(%rdi),%rax
<remove_vma+16>:       mov    0x18(%rdi),%rbp
<remove_vma+20>:       test   %rax,%rax
<remove_vma+23>:       je     <remove_vma+36>
<remove_vma+25>:       mov    0x8(%rax),%rax
<remove_vma+29>:       test   %rax,%rax
<remove_vma+32>:       je     <remove_vma+36>
<remove_vma+34>:       callq  *%rax
<remove_vma+36>:       mov    0x98(%rbx),%rdi
```

We enter shm_close() with vma in rbx, return address to remove_vma+36 on stack:

```
<shm_close>:           push   %r12
<shm_close+2>:         push   %rbp
<shm_close+3>:         push   %rbx
```

Our vma is ffff88016d2319e0, hooray!

crash – finally getting the shmid

```
crash> struct vm_area_struct.vm_file ffff88016d2319e0
  vm_file = 0xffff8800148fcb80

crash> struct file.private_data 0xffff8800148fcb80
  private_data = 0xffff8803f382cc60

crash> struct shm_file_data 0xffff8803f382cc60
struct shm_file_data {
  id = 13008988,
  ns = 0xffffffff81a46920 <init_ipc_ns>,
  file = 0xffff88037a645680,
  vm_ops = 0xffffffff816268a0 <shmem_vm_ops>
}
```

crash – checking existing shmid's with `ipcs`

- This checks the same structure that `ipc_lock()` was searching by `idr_find()`
  - No need to inspect it manually (a colleague did that, anyway)
- Our id 13008988 is indeed not there – was it freed too early, or is there a corruption?
  - The id 13008943 looks pretty close...

```
crash> ipcs -m
SHMID_KERNEL     KEY      SHMID      UID   PERMS BYTES      NATTCH
STATUS
ffff880424c74b90 00004dc4 98304      50016 760   40141728   1
...
ffff8803792ed0d0 000027b4 12976174   28900 740   4194480    24
ffff8803790f5790 000027c5 13008943   28900 740   20672      12
ffff880365da6b90 0000277a 13795376   28900 740   512000000  8519
ffff88039f3169d0 00002792 13828145   28900 740   54060      24
ffff880365d75b90 000027ae 13860914   28900 740   2076       24
ffff880365da6c90 000027ac 13893683   28900 740   535000     11
```

crash – checking suspicious shmid

```
crash> struct shmid_kernel ffff8803790f5790
struct shmid_kernel {
  shm_perm = {
…
    deleted = 0,
    id = 13008943,
    key = 10181,
…
  },
  shm_file = 0xffff88037a645680,
  shm_nattch = 12,
}
```

- Let’s compare it with our shm_file_data – same file pointer! One of the id’s is most likely corrupted, but which?

```
crash> struct shm_file_data 0xffff8803f382cc60
struct shm_file_data {
  id = 13008988,
  file = 0xffff88037a645680,
}
```

crash – which shmid is corrupted?

```
crash> search 0xffff88037a645680 # the shm_file pointer
```

- the `shmid_kernel ffff8803790f5790` with `shm_file` field
- two self-references (empty `list_head`)

crash – which shmid is corrupted?

`crash> search 0xffff88037a645680 # the shm_file pointer`

- the `shmid_kernel ffff8803790f5790` with `shm_file` field
- two self-references (empty `list_head`)
- 12 objects from the `size-32` kmalloc cache
  - one is our `shm_file_data 0xffff8803f382cc60` with id 13008988
  - the other 11 appear to be correct `shm_file_data` structures with id 13008943
  - `shm_nattch = 12` – there should be 12 instances with id 13008943

crash – which shmid is corrupted?

`crash> search 0xffff88037a645680 # the shm_file pointer`

- the `shmid_kernel ffff8803790f5790` with `shm_file` field
- two self-references (empty `list_head`)
- 12 objects from the `size-32` kmalloc cache
  - one is our `shm_file_data 0xffff8803f382cc60` with id 13008988
  - the other 11 appear to be correct `shm_file_data` structures with id 13008943
  - `shm_nattch = 12` – there should be 12 instances with id 13008943
- Conclusion: the `shmid_kernel` in the registry with id 13008943 is correct, the single `shm_file_data` with id 13008988 (that got BUG) is wrong
  - Note: it's often possible to cross-check data in kernel like this, as there are redundancies for e.g. better performance

What next?

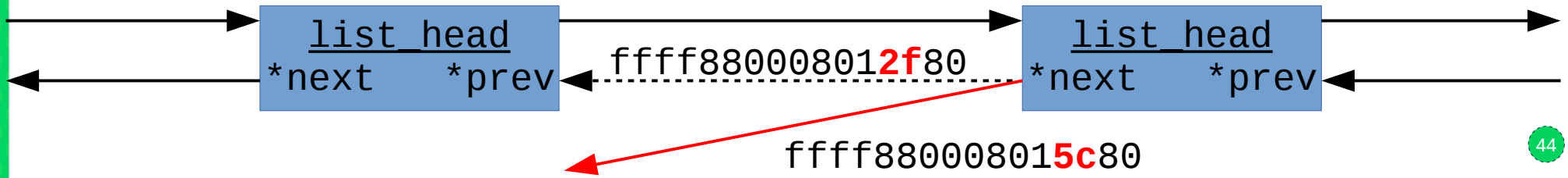
- We found what exactly was wrong (first phase), but not why it was wrong
  - Doesn’t look like a RAM error. And catching stray writes by software is very hard.

What next?

- We found what exactly was wrong (first phase), but not why it was wrong
  - Doesn’t look like a RAM error. And catching stray writes by software is very hard.
  - It only happened once, in a VM. Further occurrences would show if this happens at the same place or randomly, and if there’s a pattern in the corruption itself.
- 6 months later, another bug report – crashes while reading /proc/slabinfo and other operation in SLAB due to broken lists of slab structures
  - Used our in-development tool `crash-python` for more thorough integrity checks of SLAB than `crash` offers
    - In several dumps, the corruption of a doubly-linked list allowed to detect a corrupted pointer and compare it with an expected value
    - Others were FUBAR because they didn’t crash soon enough

What next?

- We found what exactly was wrong (first phase), but not why it was wrong
  - Doesn't look like a RAM error. And catching stray writes by software is very hard.
  - It only happened once, in a VM. Further occurrences would show if this happens at the same place or randomly, and if there's a pattern in the corruption itself.
- 6 months later, another bug report – crashes while reading /proc/slabinfo and other operation in SLAB due to broken lists of slab structures
  - Used our in-development tool `crash-python` for more thorough integrity checks of SLAB than `crash` offers
    - In several dumps, the corruption of a doubly-linked list allowed to detect a corrupted pointer and compare it with an expected value
    - Others were FUBAR because they didn't crash soon enough



The Pattern Emerges...

- Turns out all the “simple” corruptions changed byte `2f` to `5c`, and affected a size-32 or size-64 kmalloc object
- For example, back to the shmid case:

```
crash> eval -b 13008943 # the expected value
hexadecimal: c6802f
    decimal: 13008943
      octal: 61500057
     binary:00000000000000000000000011000110100000000101111
   bits set: 23 22 18 17 15 5 3 2 1 0

crash> eval -b 13008988 # the wrong value
hexadecimal: c6805c
    decimal: 13008988
      octal: 61500134
     binary:00000000000000000000000011000110100000001011100
   bits set: 23 22 18 17 15 6 4 3 2
```

If In Doubt, Try ASCII (kudos to Michal Hocko)

- `2f` is `'/'`, `5c` is `'\'`

If In Doubt, Try ASCII (kudos to Michal Hocko)

- `2f` is `'/'`, `5c` is `'\'`
- Was somebody rewriting Linux filesystem paths to Windows?
  - The CIFS module (Samba client) has a function for that – `convert_delimiter()`, called by e.g. `cifs_build_path_to_root()`

```
static inline void convert_delimiter(char *path, char delim)
{
        int i;
        char old_delim;

        if (path == NULL)
                return;

        if (delim == '/')
                old_delim = '\\';
        else
                old_delim = '/';

        for (i = 0; path[i] != '\0'; i++) {
                if (path[i] == old_delim)
                        path[i] = delim;
        }
}
```

cifs_build_path_to_root() (v3.0.101)

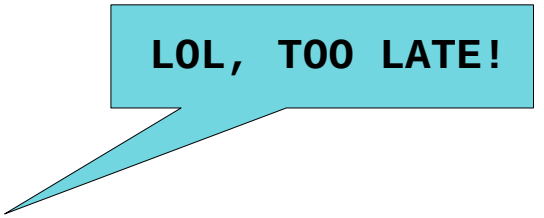
```
char *cifs_build_path_to_root(struct smb_vol *vol, struct cifs_sb_info *cifs_sb,
                              struct cifs_tcon *tcon)
{
        int pplen = vol->prepath ? strlen(vol->prepath) : 0;
        int dfsplen;
        char *full_path = NULL;
...
        dfsplen = strnlen(tcon->treeName, MAX_TREE_SIZE + 1);
...
        full_path = kmalloc(dfsplen + pplen + 1, GFP_KERNEL);
        if (full_path == NULL)
                return full_path;

        if (dfsplen)
                strncpy(full_path, tcon->treeName, dfsplen);
        strncpy(full_path + dfsplen, vol->prepath, pplen);
        convert_delimiter(full_path, CIFS_DIR_SEP(cifs_sb));
        full_path[dfsplen + pplen] = 0; /* add trailing null */
        return full_path;
}
```

cifs_build_path_to_root() (v3.0.101)

```
char *cifs_build_path_to_root(struct smb_vol *vol, struct cifs_sb_info *cifs_sb,
                              struct cifs_tcon *tcon)
{
        int pplen = vol->prepath ? strlen(vol->prepath) : 0;
        int dfsplen;
        char *full_path = NULL;
...
        dfsplen = strnlen(tcon->treeName, MAX_TREE_SIZE + 1);
...
        full_path = kmalloc(dfsplen + pplen + 1, GFP_KERNEL);
        if (full_path == NULL)
                return full_path;

        if (dfsplen)
                strncpy(full_path, tcon->treeName, dfsplen);
        strncpy(full_path + dfsplen, vol->prepath, pplen);
        convert_delimiter(full_path, CIFS_DIR_SEP(cifs_sb));
        full_path[dfsplen + pplen] = 0; /* add trailing null */
        return full_path;
}
```



Epilogue

- The bug was introduced upstream probably in 2011, kernel 3.1 (the commit was then backported to stable 3.0.x)

Epilogue

- The bug was introduced upstream probably in 2011, kernel 3.1 (the commit was then backported to stable 3.0.x)
- The bug was fixed upstream in 2012, kernel 3.8, unknowingly (the commit intends to fix something else, much less critical)
  - This is a very common experience...
  - Upstream stable 3.2 was still maintained and vulnerable back in 2018, but CIFS maintainers didn’t respond to my report…

Epilogue

- The bug was introduced upstream probably in 2011, kernel 3.1 (the commit was then backported to stable 3.0.x)
- The bug was fixed upstream in 2012, kernel 3.8, unknowingly (the commit intends to fix something else, much less critical)
  - This is a very common experience...
  - Upstream stable 3.2 was still maintained and vulnerable back in 2018, but CIFS maintainers didn’t respond to my report…
- Customer reported it only in 2017 – 6 years old bug
  - Needs a specific CIFS client/server settings, including frequent reconnect
  - Maybe it was reported before, but happened just once and we didn’t solve it

Epilogue

- The bug was introduced upstream probably in 2011, kernel 3.1 (the commit was then backported to stable 3.0.x)
- The bug was fixed upstream in 2012, kernel 3.8, unknowingly (the commit intends to fix something else, much less critical)
  - This is a very common experience...
  - Upstream stable 3.2 was still maintained and vulnerable back in 2018, but CIFS maintainers didn’t respond to my report…
- Customer reported it only in 2017 – 6 years old bug
  - Needs a specific CIFS client/server settings, including frequent reconnect
  - Maybe it was reported before, but happened just once and we didn’t solve it
- Soon after we fixed our kernel, a different customer (with unpatched kernel) suddenly reported the same bug
  - Also a very common experience
  - Actually surprised that there was just one...

Thank you.