Introdution to Kernel Crash Debugging

김동현**(Austin Kim)**

austindh.kim@gmail.com

Personal Blog: http://rousalome.egloos.com/

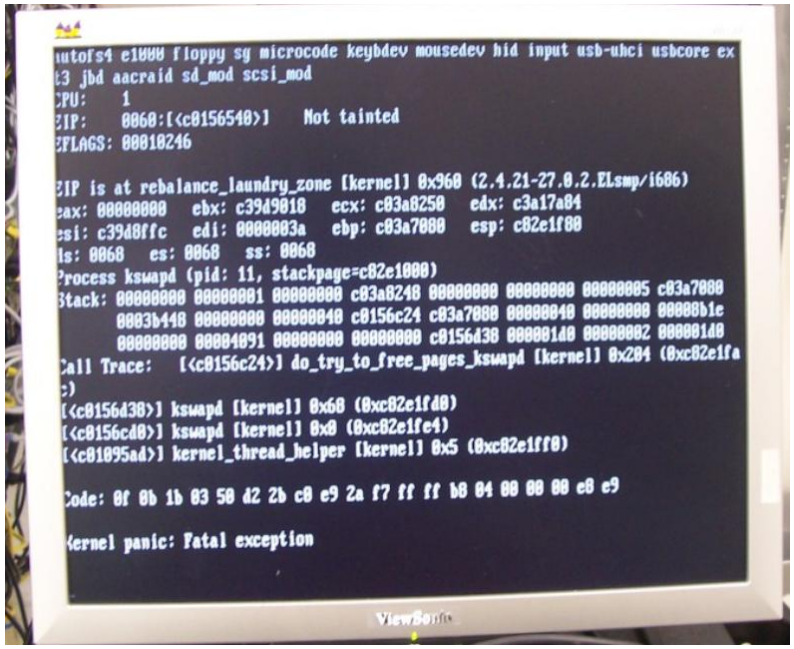
About Presenter

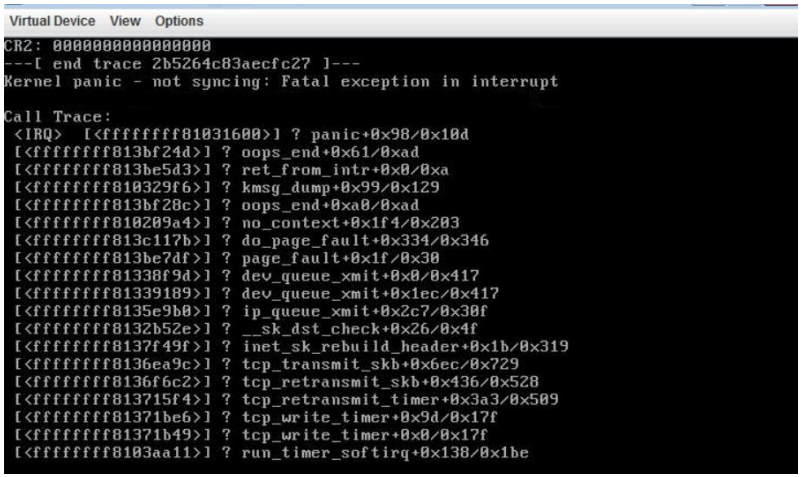
- 11+ years of Linux Kernel Stability debugging and troubleshooting;
- Fix bugs in low-layer software;
- Android BSP (Device Drivers, HAL);
- RTOS;
- Linux Kernel Contribution(as hobby)
  - https://git.kernel.org/pub/scm/linux/kernel/git/next/linux-next.git/log/?qt=author&q=Austin+Kim
- Technical writing
  - Personal Blog: http://rousalome.egloos.com

Austin Kim(austindh.kim@gmail.com)

커널 크래시란 무엇인가?

- A kernel panic is a safety measure taken by an operating system's kernel upon detecting an internal fatal error in which it either is unable to safely recover or cannot have the system continue to run without having a much higher risk of major data loss.





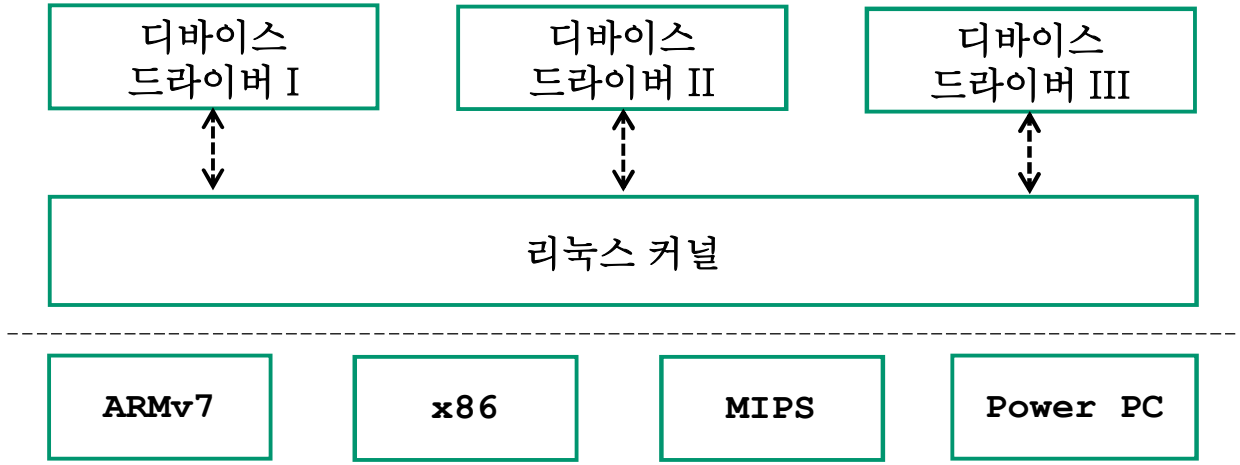
커널 크래시에 대한 다양한 생각

- 커널 크래시는 절대 발생하지 않는다! 리눅스 커널은 안정된 운영체제이기 때문이다.
- 디바이스 설정을 제대로 못해 커널 크래시가 발생한다.
- 커널 크래시는 하드웨어적인 문제로 인해 발생한다.
- 커널 크래시가 발생하면 심각한 오류가 있는 상태다.

Austin Kim(austindh.kim@gmail.com)

리눅스 커널 개발 과정에 따른 커널 크래시 유형[1/3]

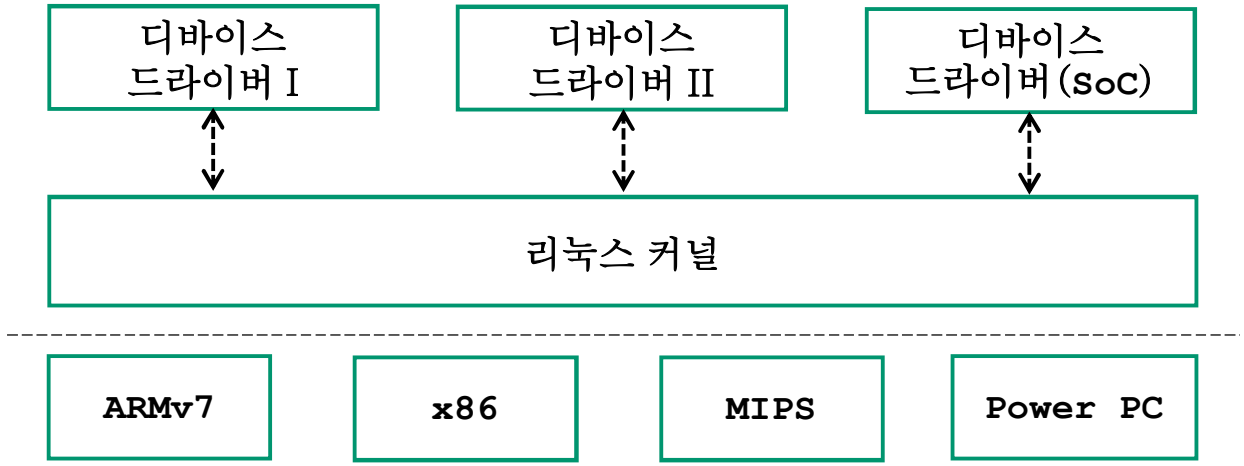
- 리눅스 커널의 심장인 리눅스 커널 커뮤니티
  - 메일링 리스트에서 논의
  - syzbot 구글 그룹 메일링 리스트로 버그 공유
  - 디버깅 방법 논의 과정 공개됨



Austin Kim(austindh.kim@gmail.com)

리눅스 커널 개발 과정에 따른 커널 크래시 유형[2/3]

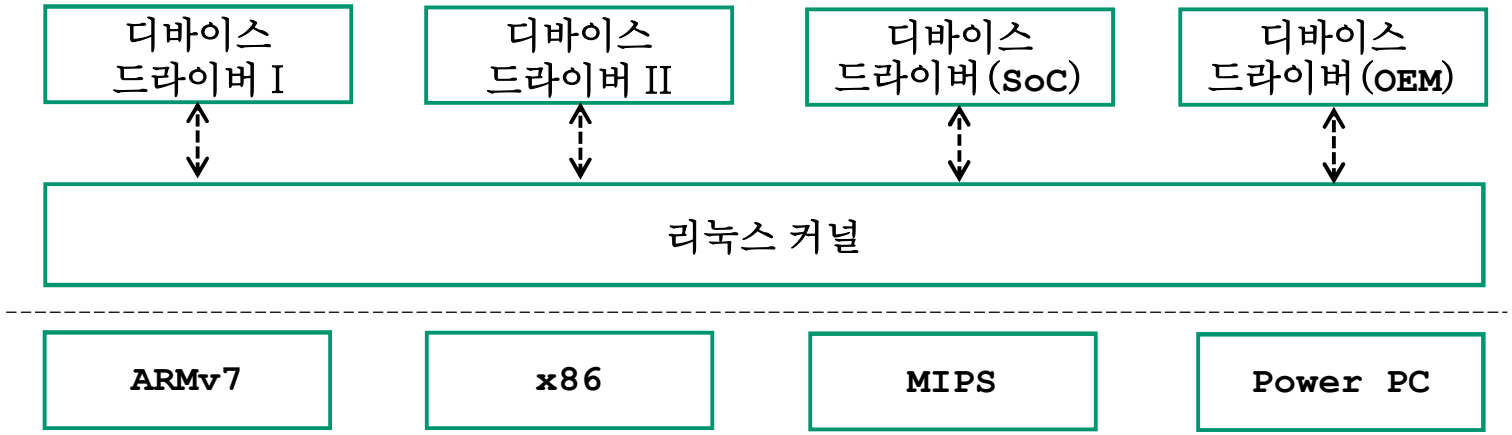
- SoC(System-on-Chip) 벤더
  - SoC 스팩에 맞게 드라이버를 올림
  - 버그 수정 및 디버깅 과정 공유 안됨



Austin Kim(austindh.kim@gmail.com)

리눅스 커널 개발 과정에 따른 커널 크래시 유형[3/3]

- OEM 벤더
  - 실제 사용화 제품 개발
  - Peripheral(RAM, Sensor, Network) 디바이스 탑재
  - 버그 수정 및 디버깅 과정 공유 안됨



Austin Kim(austindh.kim@gmail.com)

커널 크래시의 유형

- 논리적인 오류
  - BUG, Panic
  - Fault, Exception
  - Watchdog Reset
  - Race Condition

- 하드웨어 문제
  - 메모리 비트 플립
  - 전원 불안정
  - 클락/파워 게이팅
  - 프로세서 내 버그

Austin Kim(austindh.kim@gmail.com)

커널 크래시 디버깅을 위해 갖추어야 하는 스킬

- CPU 아키텍처
  - 어셈블리 언어
  - 익셉션
  - 인터럽트 처리 방식
  - 레지스터 세트

- 커널 주요 서브 시스템
  - 프로세스, 인터럽트 처리 방식, 시스템 콜, 시그널 메모리 관리, 동기화
  - 워크큐, Soft IRQ

- Debug Feature
  - strace, ftrace, 커널 로그, vmcore

Austin Kim(austindh.kim@gmail.com)

커널 크래시 디버깅 프로그램 – Crash Utility

- 오픈 소스 기반 커널 크래시 디버깅 유틸리티
- 메인테이너: David Anderson(레드햇)
- 주요 자료
  - https://people.redhat.com/anderson/crash_whitepaper/

```
crash> bt e5752c00
PID: 1787   TASK: e5752c00  CPU: 4   COMMAND: "net_socket"
bt: WARNING:  stack address:0xe853fa38, program
counter:0xc0ee5b60
 #0 [<c0ed8b64>] (panic) from [<c0125038>]
 #1 [<c0125038>] (__stack_chk_fail) from [<c032b6cc>]
 #2 [<c032b6cc>] (sock_has_perm) from [<c0327d00>]
 #3 [<c0327d00>] (security_socket_recvmsg) from
[<c0ceb1c8>]
 #4 [<c0ceb1c8>] (sock_recvmsg) from [<c0cec474>]
 #5 [<c0cec474>] (___sys_recvmsg) from [<c0ced5b4>]
 #6 [<c0ced5b4>] (__sys_recvmsg) from [<c0106820>]
```


Austin Kim(austindh.kim@gmail.com)

커널 크래시 디버깅 프로그램 – GDB

- GNU 소프트웨어 시스템을 위한 기본 디버거

```
(gdb) bt
#0  0x000055555572de2b in arm64_is_kernel_exception_frame (bt=0x7fffffffd640,
stkptr=18446743644915693792) at arm64.c:1785
#1  0x000055555572ffaf in arm64_back_trace_cmd (bt=0x7fffffffd640) at arm64.c:2594
#2  0x00005555556ef0c4 in back_trace (bt=0x7fffffffd640) at kernel.c:3164
#3  0x00005555556ed624 in cmd_bt () at kernel.c:2833
#4  0x000055555564a73b in exec_command () at main.c:879
#5  0x000055555564a515 in main_loop () at main.c:826
#6  0x00005555558b5b43 in captured_command_loop (data=data@entry=0x0) at main.c:258
#7  0x00005555558b46ca in catch_errors (func=func@entry=0x5555558b5b30 <captured_command_loop>,
func_args=func_args@entry=0x0,
    errstring=errstring@entry=0x555555b0b728 "", mask=mask@entry=6) at exceptions.c:557
#8  0x00005555558b6c42 in captured_main (data=data@entry=0x7fffffffdfb0) at main.c:1064
#9  0x00005555558b46ca in catch_errors (func=func@entry=0x5555558b5e70 <captured_main>,
func_args=func_args@entry=0x7fffffffdfb0,
    errstring=errstring@entry=0x555555b0b728 "", mask=mask@entry=6) at exceptions.c:557
#10 0x00005555558b702e in gdb_main (args=0x7fffffffdfb0) at main.c:1079
#11 gdb_main_entry (argc=<optimized out>, argv=<optimized out>) at main.c:1099
#12 0x000055555570fc53 in gdb_main_loop (argc=2, argv=0x7fffffffe148) at gdb_interface.c:76
#13 0x000055555564a1e0 in main (argc=4, argv=0x7fffffffe148) at main.c:707
```


Austin Kim(austindh.kim@gmail.com)

커널 크래시 디버깅 프로그램 – Trace32

- 임베디드 개발에서 가장 널리 쓰이는 디버깅 프로그램
- 주요 자료
  - https://www.trace32.com/

```
-000|__ipv6_dev_ac_inc()
-001|addrconf_join_anycast()
-002|__ipv6_ifa_notify()
-003|local_bh_enable(inline)
-003|rcu_read_unlock_bh(inline)
-003|ipv6_ifa_notify()
-004|addrconf_dad_begin(inline)
-004|addrconf_dad_work()
-005|static_key_count(inline)
-005|static_key_false(inline)
-005|trace_workqueue_execute_end(inline)
-005|process_one_work()
-006|list_empty(inline)
-006|worker_thread()
-007|kthread()
-008|ret_from_fork(asm)
-009|ret_fast_syscall(asm)
```


Austin Kim(austindh.kim@gmail.com)

주요 Signature – BUG

- 커널에서 치명적인 논리적인 오류가 있을 때 발생

```
Hello,

syzbot found the following crash on:
...
IMPORTANT: if you fix the bug, please add the following tag to the commit:
Reported-by: syzbot+221cc24572a2fed23b6b@syzkaller.appspotmail.com

BUG: unable to handle page fault for address: fffffc0000000000
#PF: supervisor read access in kernel mode
#PF: error_code(0x0000) - not-present page
...
Call Trace:
  __kasan_check_write+0x14/0x20 mm/kasan/common.c:98
  set_bit include/asm-generic/bitops-instrumented.h:28 [inline]
  io_wq_cancel_all+0x28/0x2a0 fs/io-wq.c:617
  io_uring_flush+0x35a/0x4e0 fs/io_uring.c:3936
  filp_close+0xbd/0x170 fs/open.c:1174
  close_files fs/file.c:388 [inline]
  put_files_struct fs/file.c:416 [inline]
  put_files_struct+0x1d7/0x2f0 fs/file.c:413
  exit_files+0x83/0xb0 fs/file.c:445
```

주요 Signature – Data Abort

- MMU가 처리 못하는 메모리

```
[287229.435076] <c6>mmc0: mmc_start_bkops: raw_bkops_status=0x2, from_exception=1
[287230.328287] <c6>mmc0: mmc_start_bkops: Starting bkops
[287231.319886] <26>Unable to handle kernel NULL pointer dereference at virtual address
000000fc
[287231.319920] <26>pgd = c0004000
[287231.319936] <22>[000000fc] *pgd=00000000
[287231.319957] <6>Internal error: Oops: 5 [#1] PREEMPT SMP ARM
[287231.319974] <c2>Modules linked in: core_ctl(PO)
[287231.320000] <c6>CPU: 2 PID: 13 Comm: kworker/1:0 Tainted: P        W  O 3.10.49-gca5eb64 #1
[287231.320028] <c2>Workqueue: events irq_affinity_notify
[287231.320046] <c6>task: e0063fc0 ti: e01f2000 task.ti: e01f2000
[287231.320065] <c2>PC is at __blocking_notifier_call_chain+0x4c/0xf8
[287231.320083] <c2>LR is at pm_qos_update_target+0x20c/0x240
[287231.320102] <c2>pc : [<c006c194>]    lr : [<c0068ce0>]    psr: 00000013
```


Austin Kim(austindh.kim@gmail.com)

주요 Signature – Slub Object Corruption [1/2]

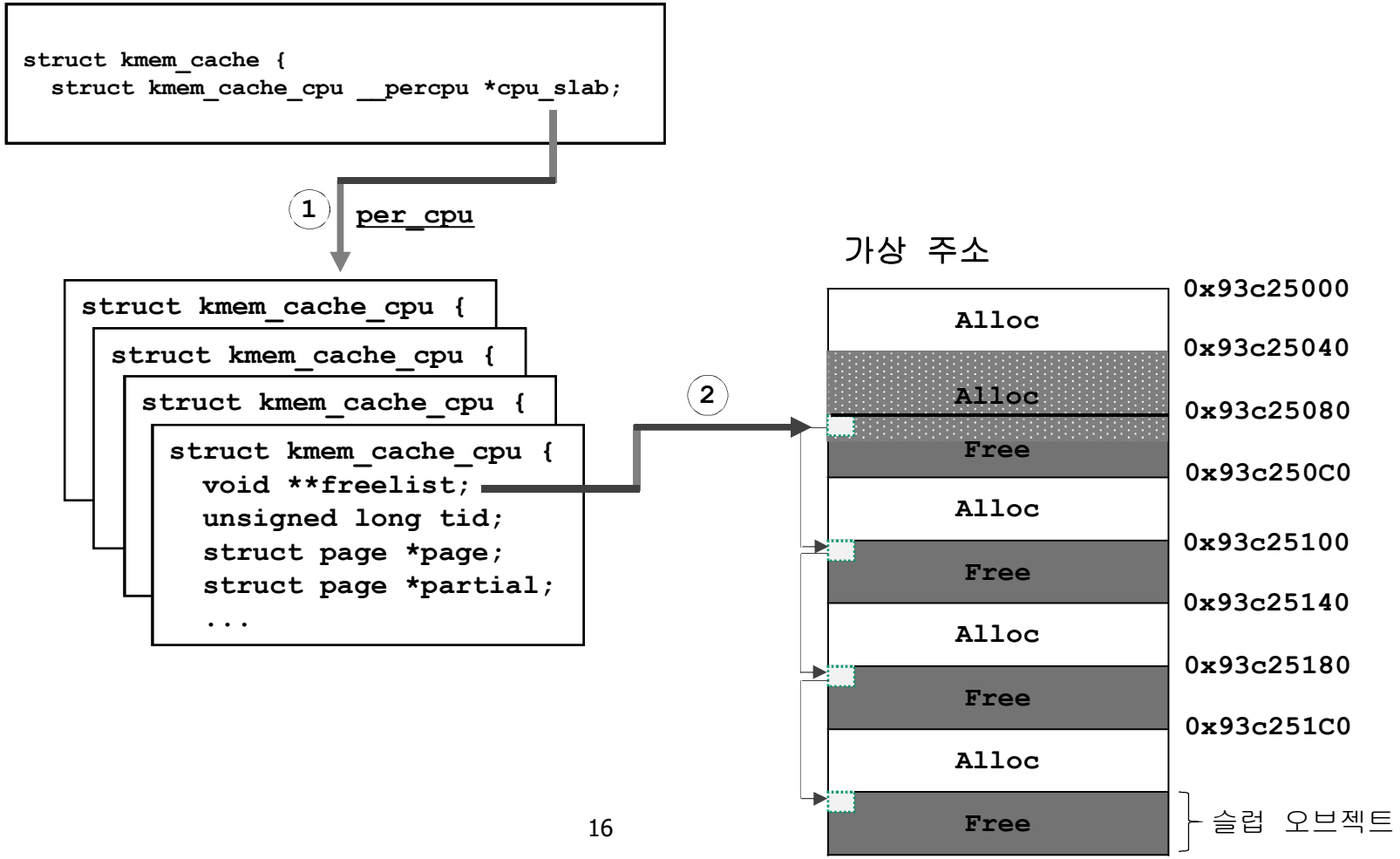
- Slub Object를 침범함 Corruption

```
[  202.292579] lkdtm: Performing direct entry WRITE_AFTER_FREE
[  202.293587] BUG kmalloc-1024 (Not tainted): Poison overwritten
[  202.293620] Disabling lock debugging due to kernel taint
[  202.293634] INFO: 0xd5885a40-0xd5885e3e. First byte 0x78 instead of 0x6b
[  202.293659] INFO: Allocated in lkdtm_do_action+0xd0/0x1a8 age=0 cpu=0 pid=7421
[  202.293674]          lkdtm_do_action+0xd0/0x1a8
[  202.293689]          direct_entry+0xe4/0x110
[  202.293705]          vfs_write+0xd0/0x180
[  202.293719]          SyS_write+0x38/0x68
[  202.293735]          __sys_trace_return+0x0/0x18
[  202.293752] INFO: Freed in lkdtm_do_action+0xd8/0x1a8 age=0 cpu=0 pid=7421
[  202.293766]          direct_entry+0xe4/0x110
[  202.293779]          vfs_write+0xd0/0x180
[  202.293793]          SyS_write+0x38/0x68
[  202.293808]          __sys_trace_return+0x0/0x18
[  202.293823] INFO: Slab 0xc3d07200 objects=26 used=26 fp=0x  (null) flags=0x4080
[  202.293837] INFO: Object 0xd5885a40 @offset=23104 fp=0xd5882f80
[  202.293837]
[  202.293858] Bytes b4 d5885a30: 5a 5a 5a 5a 5a 5a 5a 5a 5a 5a 5a 5a 5a 5a 5a 5a  ZZZZZZZZZZZZZZZZ
[  202.293874] Object d5885a40: 78 78 78 78 78 78 78 78 78 78 78 78 78 78 78 78  xxxxxxxxxxxxxxxx
[  202.293888] Object d5885a50: 78 78 78 78 78 78 78 78 78 78 78 78 78 78 78 78  xxxxxxxxxxxxxxxx
```


Austin Kim(austindh.kim@gmail.com)

주요 Signature – Slub Object Corruption [2/2]

- Slub Object Corruption 원인



Austin Kim(austindh.kim@gmail.com)

주요 Signature – Memory Corruption

- 코드가 깨져 있음

```
- Vmcore
0xc0174088 <profile_munmap>:    andeq   r0, r0, r0
0xc017408c <profile_munmap+0x4>:        ldmdahi r7!, {r3, r4, r5, r6, r8, r12, sp, lr, pc}^
0xc0174090 <profile_munmap+0x8>:        ldmibhi r4, {sp}^
0xc0174094 <profile_munmap+0xc>:        bhi     0xbec9f25c
0xc0174098 <profile_munmap+0x10>:       bhi     0xbecd33a0
0xc017409c <profile_tick>:                      ; <UNDEFINED> instruction: 0x87dcb159
0xc01740a0 <profile_tick+0x4>:  mrc     15, 0, r2, cr13, cr0, {4}

- Vmlinux
c0174088:       e1a02000        mov     r2, r0
c017408c:       e3a01000        mov     r1, #0
c0174090:       e59f0000        ldr     r0, [pc]        ; c0174098 <profile_munmap+0x10>
c0174094:       eaff3ac3        b       c0142ba8 <blocking_notifier_call_chain>
c0174098:       c132a5c0        .word   0xc132a5c0
c017409c:       e92d4010        push    {r4, lr}
```


Austin Kim(austindh.kim@gmail.com)

주요 Signature – 유효하지 않은 자료구조 필드

- 스핀락

```
crash> arch_spinlock_t 0xb130d348
struct arch_spinlock_t {
  {
    slock = 0x73637461,
    tickets = {
      owner = 0x7461,
      next = 0x7363
    }
  }
}
```

- 태스크 디스크립터 / 스레드 정보

```
crash> task b8d3d6c0
struct task_struct {
  state = 0x14ff97,
  stack = 0x11,
  usage = {
    counter = 0x3
```

```
crash> struct thread_info b313d6d4
struct thread_info {
  flags = 0x32808,
  preempt_count = 0x0,
  addr_limit = 0x0,
  task = 0xff,
  cpu = 0xe5c24400,
```


Austin Kim(austindh.kim@gmail.com)