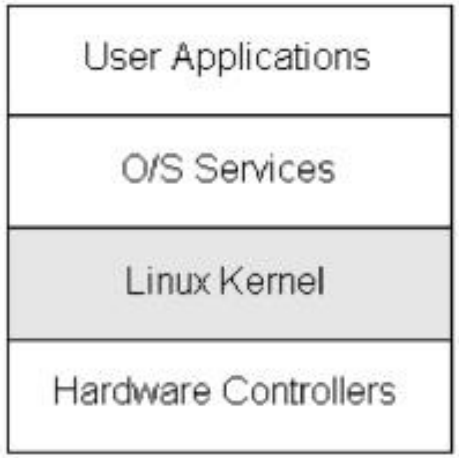
# 第0019讲 20内存与Kasan工具源码详解68





零声学院讲师：Vico老师

## 一、内存检测基础

## 二、Kasan工具源码详解

# 一、内存检测基础

Linux内核与驱动几乎所有代码都是C语言干出来的，C它具备强大的功能，特别是指针非常灵活及访问内存。但也存在一些问题，如果编写代码刚好引用空指针，内核的虚拟内存机制可以捕捉到，直接产生一个oops错误警告。

**常见内存访问错误：**

- 越界访问（out-of-bounds）
- 访问已经被释放的内存（use after free）
- 重复释放（double free）
- 内存泄漏（memory leak）
- 栈溢出（stack overflow）

# 二、Kasan内核检测工具



内核地址消毒剂（KASAN，Kernel Address SANitizer）是一个动态的内存错误检查工具，为发现“释放后使用”和“越界访问”这两类缺陷提供快速和综合的解决方案。KASAN使用影子（shadow memory）内存记录内存的每个字节是否可以安全访问，使用编译时插桩在每次访问内存时检查影子内存。

mm > kasan > C kasan.h > KASAN_STACK_PARTIAL

```
14
15  #ifdef CONFIG_KASAN_GENERIC
16  #define KASAN_FREE_PAGE         0xFF  /* page was freed */
17  #define KASAN_PAGE_REDZONE      0xFE  /* redzone for kmalloc_large allocations */
18  #define KASAN_KMALLOC_REDZONE   0xFC  /* redzone inside slub object */
19  #define KASAN_KMALLOC_FREE      0xFB  /* object was freed (kmem_cache_free/kfree) */
20  #else
21  #define KASAN_FREE_PAGE         KASAN_TAG_INVALID
22  #define KASAN_PAGE_REDZONE      KASAN_TAG_INVALID
23  #define KASAN_KMALLOC_REDZONE   KASAN_TAG_INVALID
24  #define KASAN_KMALLOC_FREE      KASAN_TAG_INVALID
25  #endif
26
```

在Linux系统启动初始化时，分配shadow需要的物理内存并将它们直接映射到对应的虚拟内存地址区域：

arch > x86 > mm > C kasan_init_64.c > kasan_init(void)

```
293
294 void __init kasan_init(void)
295 {
296     int i;
297     void *shadow_cpu_entry_begin, *shadow_cpu_entry_end;
298
299 #ifdef CONFIG_KASAN_INLINE
300     register_die_notifier(&kasan_die_notifier);
301 #endif
302
303     memcpy(early_top_pgt, init_top_pgt, sizeof(early_top_pgt));
```

将内存的内核虚拟地址转换成影子地址方法操作如下：

include > linux > C kasan.h > kasan_populate_early_shadow(const void *, const void *)

```
29  static inline void *kasan_mem_to_shadow(const void *addr)
30  {
31      return (void *)((unsigned long)addr >> KASAN_SHADOW_SCALE_SHIFT)
32          + KASAN_SHADOW_OFFSET;
33  }
34
```

KASAN模块提供测试程序，具体内核源码如下：

lib > C test_kasan.c > kmalloc_tests_init(void)

```
649  static int __init kmalloc_tests_init(void)
650  {
651      /*
652       * Temporarily enable multi-shot mode. Otherwise, we'd only get a
653       * report for the first case.
654       */
655      bool multishot = kasan_save_enable_multi_shot();
656
657      kmalloc_oob_right();
```


办学宗旨：一切只为渴望更优秀的你

办学愿景：让技术简单易懂