Linux任督二脉之内存管理(六)

讲解时间：3月28日晚9点

宋宝华 <21cnbao@gmail.com>

微信群直播：
http://mp.weixin.qq.com/s/6zY7B9vxzDwniYM-woQRRA

扫描二维码报名

麦当劳喜欢您来，喜欢您再来

扫描关注

Linuxer

大纲

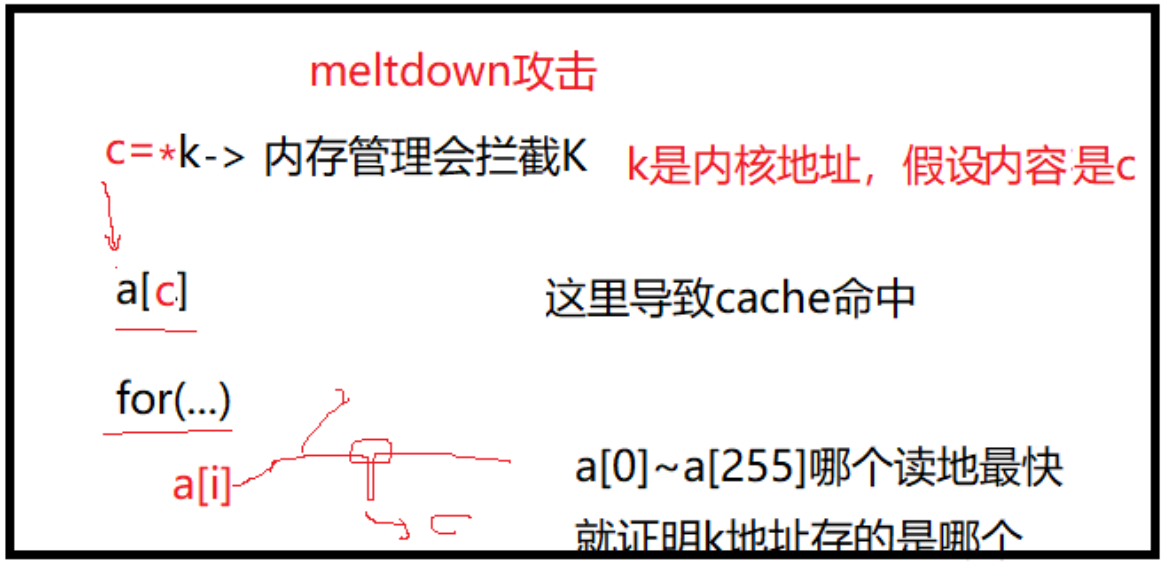
- meltdown的补丁：KPTI(X86和AARCH64)
  KPTI情况下，页表会变成怎样？
- 内核与用户交界点的安全性问题
  为什么要检查地址范围？access_ok?
- copy_from/to_user等API
- 阻止内核访问用户的PAN和SMAP
  内核访问user能力的启停
- 内存碎片避免

Meltdown 漏洞

页表里面可以表明：kernel/User+kernel权限

Meltdown则从用户空间
偷取了内核空间数据

a[256]; //每个成员4096



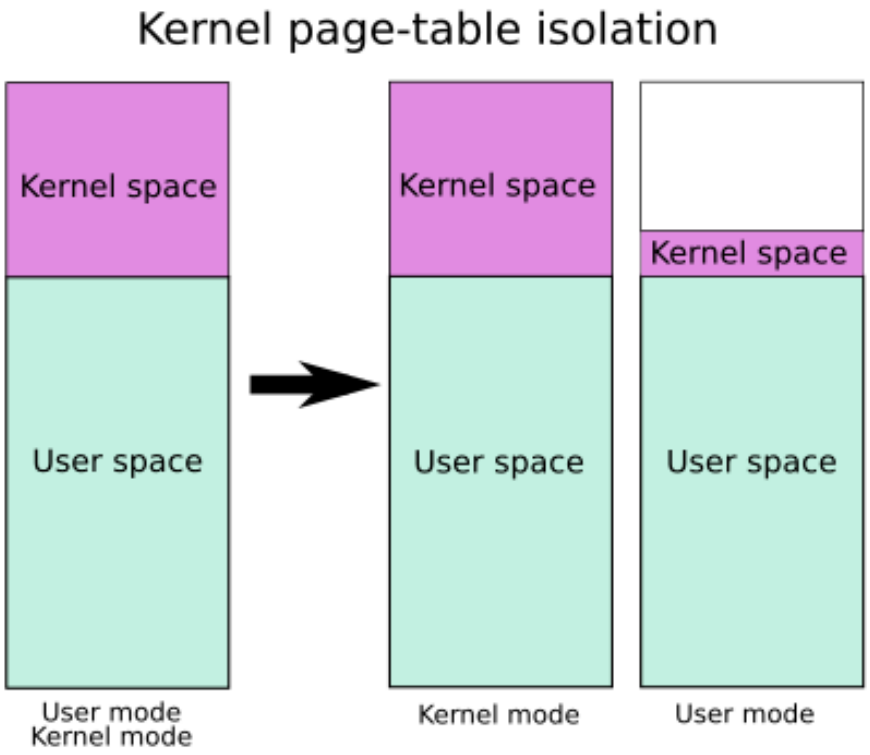
KPTI

KAISER(Kernel Address Isolation to have Side-channels Efficiently Removed): hiding the kernel from user space

https://lwn.net/Articles/738975/

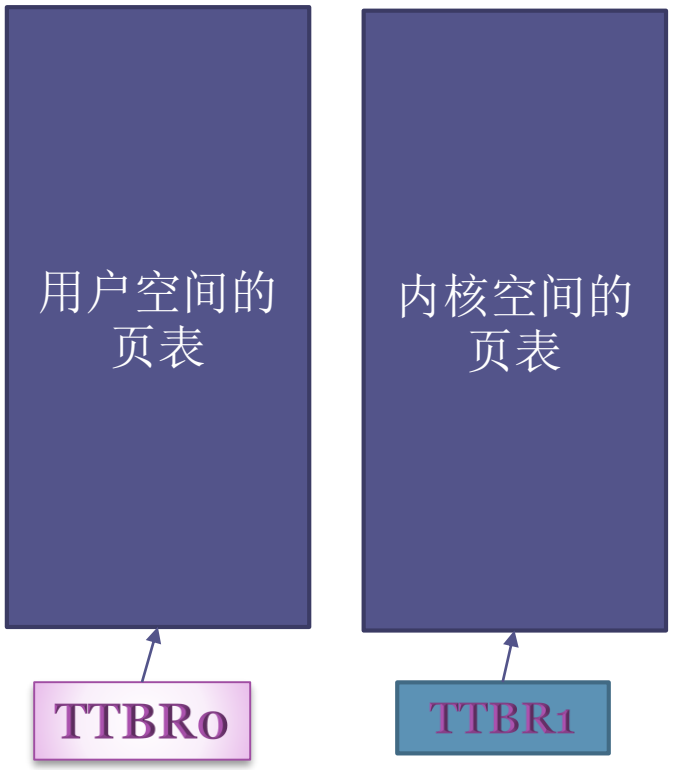
Kernel page-table isolation (KPTI or PTI, previously called KAISER)

https://en.wikipedia.org/wiki/Kernel_page-table_isolation

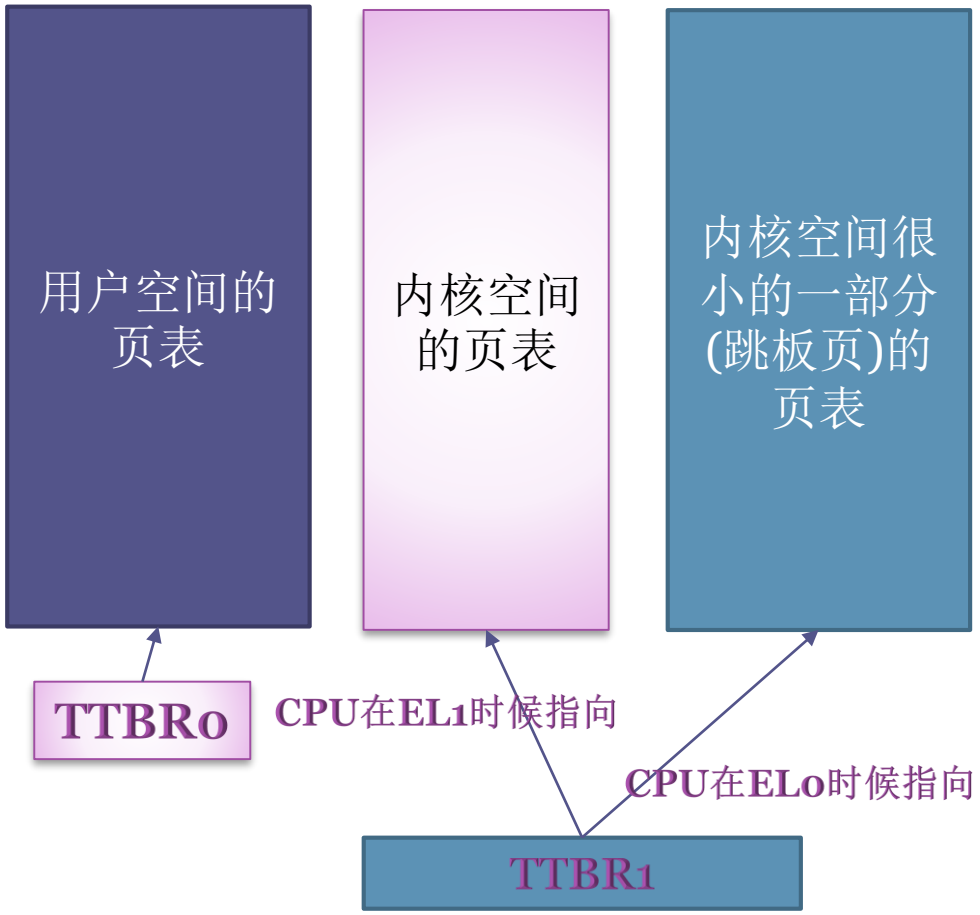


AARCH64 KPTI

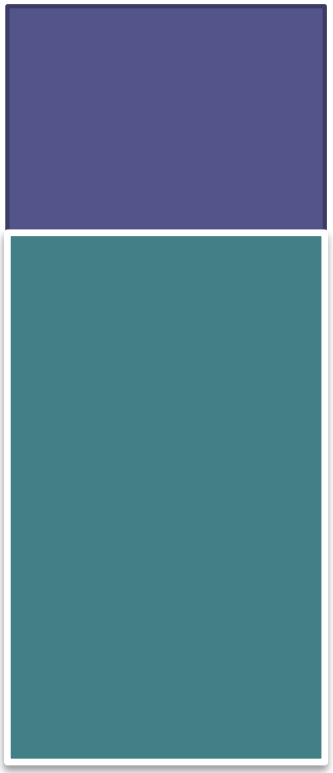
KPTI之前



KPTI之后



Kernel和user的交界点



在特权模式下，CPU可以访问user+kernel的内存；
在user模式下,CPU只能访问0-3GB的内存.
那么为什么不让kernel直接访问user，而是每次要做copy_from/to_user?

为了安全!
用户可以伪造指针，明明系统调用的参数应该是指向user buffer的，伪造一个kernel的地址

一个例子：CVE-2017-5123漏洞

Commit: 4c48abe91be

```
diff --git a/kernel/exit.c b/kernel/exit.c
index 97db9ee0..f3b8c3a 100644
--- a/kernel/exit.c
+++ b/kernel/exit.c
@@ -1625,15 +1625,18 @@ SYSCALL_DEFINE5(waitid, int, which, pid_t, upid, struct siginfo __user *,
        if (!infop)
                return err;

-       if (put_user(err ? 0 : SIGCHLD, &infop->si_signo) ||
-           put_user(0, &infop->si_errno) ||
-           put_user((short)info.cause, &infop->si_code) ||
-           put_user(info.pid, &infop->si_pid) ||
-           put_user(info.uid, &infop->si_uid) ||
-           put_user(info.status, &infop->si_status))
-               err = -EFAULT;
-
+       user_access_begin();
+       unsafe_put_user(err ? 0 : SIGCHLD, &infop->si_signo, Efault);
+       unsafe_put_user(0, &infop->si_errno, Efault);
+       unsafe_put_user((short)info.cause, &infop->si_code, Efault);
+       unsafe_put_user(info.pid, &infop->si_pid, Efault);
+       unsafe_put_user(info.uid, &infop->si_uid, Efault);
+       unsafe_put_user(info.status, &infop->si_status, Efault);
+       user_access_end();
        return err;
+Efault:
```

没有判决put user目标地址的合法性！

这样你有1000种办法攻击内核！

一个例子：CVE-2017-5123修复

Commit: 96ca579a1

```
diff --git a/kernel/exit.c b/kernel/exit.c
index f2cd53e..cf28528 100644
--- a/kernel/exit.c
+++ b/kernel/exit.c
@@ -1610,6 +1610,9 @@ SYSCALL_DEFINE5(waitid, int, which, pid_t, upid, struct siginfo __user *,
        if (!infop)
                return err;

+       if (!access_ok(VERIFY_WRITE, infop, sizeof(*infop)))
+               goto Efault;
+
        user_access_begin();
        unsafe_put_user(signo, &infop->si_signo, Efault);
        unsafe_put_user(0, &infop->si_errno, Efault);
@@ -1735,6 +1738,9 @@ COMPAT_SYSCALL_DEFINE5(waitid,
        if (!infop)
                return err;

+       if (!access_ok(VERIFY_WRITE, infop, sizeof(*infop)))
+               goto Efault;
+
        user_access_begin();
        unsafe_put_user(signo, &infop->si_signo, Efault);
        unsafe_put_user(0, &infop->si_errno, Efault);
```

交界点API

copy_from_user(void *to, const void __user *from, unsigned long n)
copy_to_user(void __user *to, const void *from, unsigned long n)
put_user(x, ptr)
get_user(x, p)
access_ok(type, addr, size)

确保是合法的user地址！

```
#ifndef INLINE_COPY_FROM_USER
unsigned long _copy_from_user(void *to, const void __user *from, unsigned long n)
{
        unsigned long res = n;
        might_fault();
        if (likely(access_ok(VERIFY_READ, from, n))) {
                kasan_check_write(to, n);
                res = raw_copy_from_user(to, from, n);
        }
        if (unlikely(res))
                memset(to + (n - res), 0, res);
        return res;
}
EXPORT_SYMBOL(_copy_from_user);
#endif

#ifndef INLINE_COPY_TO_USER
unsigned long _copy_to_user(void __user *to, const void *from, unsigned long n)
{
        might_fault();
        if (likely(access_ok(VERIFY_WRITE, to, n))) {
                kasan_check_read(from, n);
                n = raw_copy_to_user(to, from, n);
        }
        return n;
}
EXPORT_SYMBOL(_copy_to_user);
#endif
```

PAN(privileged no-access)

Kernel对userspace的访问，限制在特定的代码区间内，如copy_from/to_user:

```
static inline unsigned long __must_check
raw_copy_from_user(void *to, const void __user *from, unsigned long n)
{
        unsigned int __ua_flags;

        __ua_flags = uaccess_save_and_enable();
        n = arm_copy_from_user(to, from, n);
        uaccess_restore(__ua_flags);
        return n;
}
```

这段区间可以访问userspace

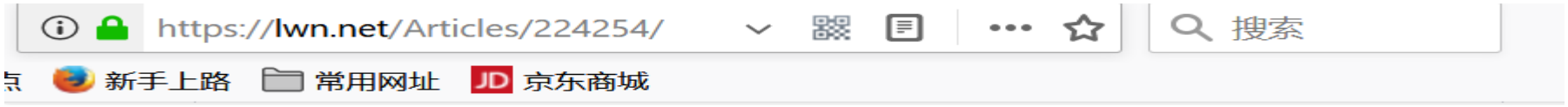
```
static inline unsigned long __must_check
raw_copy_to_user(void __user *to, const void *from, unsigned long n)
{
#ifndef CONFIG_UACCESS_WITH_MEMCPY
        unsigned int __ua_flags;
        __ua_flags = uaccess_save_and_enable();
        n = arm_copy_to_user(to, from, n);
        uaccess_restore(__ua_flags);
        return n;
#else
        return arm_copy_to_user(to, from, n);
#endif
}
```

阅读mainline代码：
arch/arm/include/asm/uaccess.h

什么是碎片?

- Internal fragmentation： 申请32个字节，但是buddy要给1页 -> slab
- External fragmentation : 申请2^n连续页，但是系统尽管空闲内存很多，由于非连续，也无法满足

一个重构28次的patch



Group pages of related mobility together to reduce external fragmentation v28

From: Mel Gorman <mel@csn.ul.ie>
To: akpm@linux-foundation.org
Subject: [PATCH 0/12] Group pages of related mobility together to reduce external fragmentation v28
Date: Thu, 1 Mar 2007 10:02:29 +0000 (GMT)
Cc: Mel Gorman <mel@csn.ul.ie>, linux-kernel@vger.kernel.org, linux-mm@kvack.org
Archive-link: Article, Thread

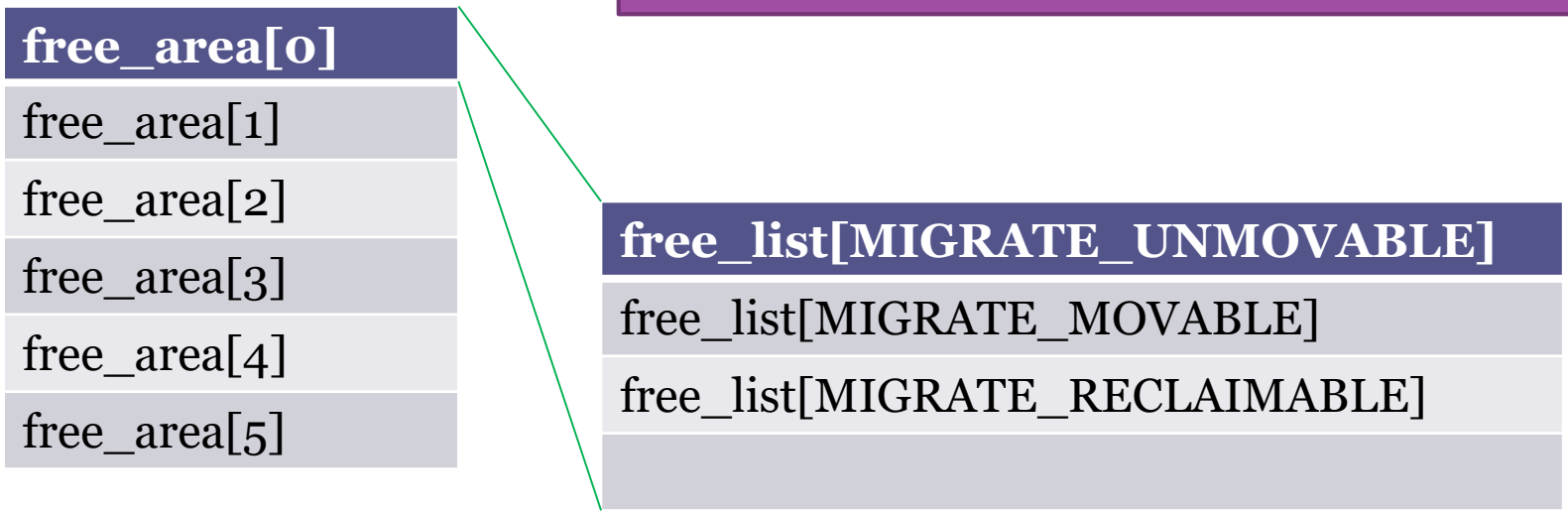
ere is the latest revision of the anti-fragmentation patches. Of
articular note in this version is special treatment of high-order atomic
llocations. Care is taken to group them together and avoid grouping pages
f other types near them. Artifical tests imply that it works. I'm trying to
et the hardware together that would allow setting up of a "real" test. If
nyone already has a setup and test that can trigger the atomic-allocation
roblem, I'd appreciate a test of these patches and a report. The second

基于migration type的free list

```
1.struct zone {
2.   ......
3.   struct free_area   free_area[MAX_ORDER];
4.   ......
5.}_____cacheline_internodealigned_in_smp;
```

```
1.struct free_area {
2.   struct list_head  free_list[MIGRATE_TYPES];

3.   unsigned long     nr_free;
4.};
```



内存申请与fallback

在本migrate type中无内存可分配时，就要去fallback的migrate type列表中找最大块内存，迁移过来

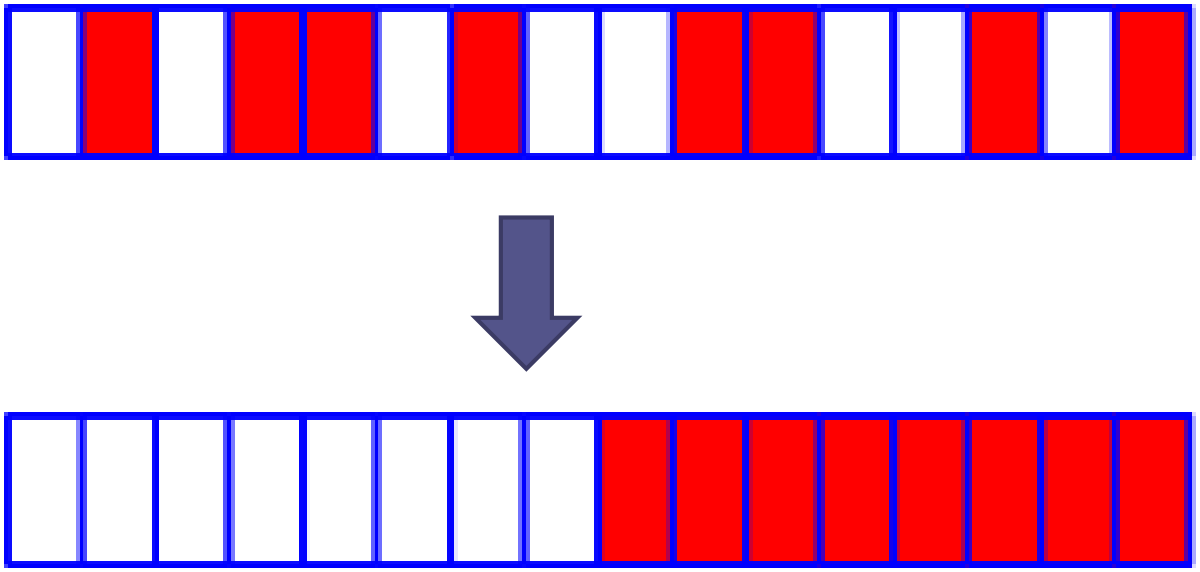
```
/*
 * This array describes the order lists are fallen back to when
 * the free lists for the desirable migrate type are depleted
 */
static int fallbacks[MIGRATE_TYPES][4] = {
        [MIGRATE_UNMOVABLE]   = { MIGRATE_RECLAIMABLE, MIGRATE_MOVABLE,   MIGRATE_TYPES },
        [MIGRATE_RECLAIMABLE] = { MIGRATE_UNMOVABLE,   MIGRATE_MOVABLE,   MIGRATE_TYPES },
        [MIGRATE_MOVABLE]     = { MIGRATE_RECLAIMABLE, MIGRATE_UNMOVABLE, MIGRATE_TYPES },
#ifdef CONFIG_CMA
        [MIGRATE_CMA]         = { MIGRATE_TYPES }, /* Never used */
#endif
#ifdef CONFIG_MEMORY_ISOLATION
        [MIGRATE_ISOLATE]     = { MIGRATE_TYPES }, /* Never used */
#endif
};
```

Memory compaction

- 触发途径：

✓ echo 1 > /proc/sys/vm/compact_memory

✓ higher-order分配失败



Memory compaction的一个例子

- ✓ echo 1 > /proc/sys/vm/compact_memory

```
root@baohua-VirtualBox:/proc/sys/vm# cat /proc/buddyinfo
Node 0, zone      DMA      7     45     17      7      2      3      2      0      1      1      0
Node 0, zone   Normal    984    935    535    188    115     97     32      6      3      3      1
Node 0, zone  HighMem    161    107     69     43     25     13     11      4      1      0      0
root@baohua-VirtualBox:/proc/sys/vm# echo 1 > compact_memory
root@baohua-VirtualBox:/proc/sys/vm# cat /proc/buddyinfo
Node 0, zone      DMA      4     17      3      3      1      2      3      1      1      1      0
Node 0, zone   Normal    653    685    341    117     50     21      9      5      3      7      8
Node 0, zone  HighMem     23     58     39     33     18     16     10      6      2      0      0
```

课后阅读

SMAP
https://en.wikipedia.org/wiki/Supervisor_Mode_Access_Prevention

CVE-2017-5123
https://github.com/nongiach/CVE/tree/master/CVE-2017-5123

宋宝华: ARM64 Linux meltdown修复补丁KPTI的最重要3个patch
http://mp.weixin.qq.com/s/jMp281XDYtBWDAKYwvUnUw

KPTI补丁分析
http://mp.weixin.qq.com/s/PX2VpPO7ms3YhwikQ3Ngpg

早期有录播的课程(非微课)

- 《Linux总线、设备、驱动模型》
http://edu.csdn.net/course/detail/5329

- 深入探究Linux的设备树
http://edu.csdn.net/course/detail/5627

- Linux进程、线程和调度
http://edu.csdn.net/course/detail/5995

- C语言大型软件设计的面向对象
https://edu.csdn.net/course/detail/6496

谢 谢！