Android安全加固：演进和对抗

方家弘
腾讯科恩实验室

提纲

- 概述
- 案例分析
  1. CVE-2013-6282
  2. CVE-2015-3636
  3. CVE-2015-0570
  4. CVE-2015-6637
  5. 其它
- 总结

概述

安卓面临的安全威胁

权限泄漏；静态密钥；敏感信息加密缺失
例子：GeekPwn上被攻破的各类app

文件解析；签名校验；应用权限管理
例子：MasterKeys, FakeID

Java代码和Native代码皆有，内存破坏漏洞居多
例子：Stagefright系列

内存破坏漏洞为主

Linux内核安全

- 提供移动计算需要的关键安全特性
  - 进程隔离
  - 基于用户的权限模型
  - 安全IPC
- 高度可扩展性
  - 裁剪不必要的模块
  - 定制安全增强特性
  - Linux Security Module (LSM) Framework

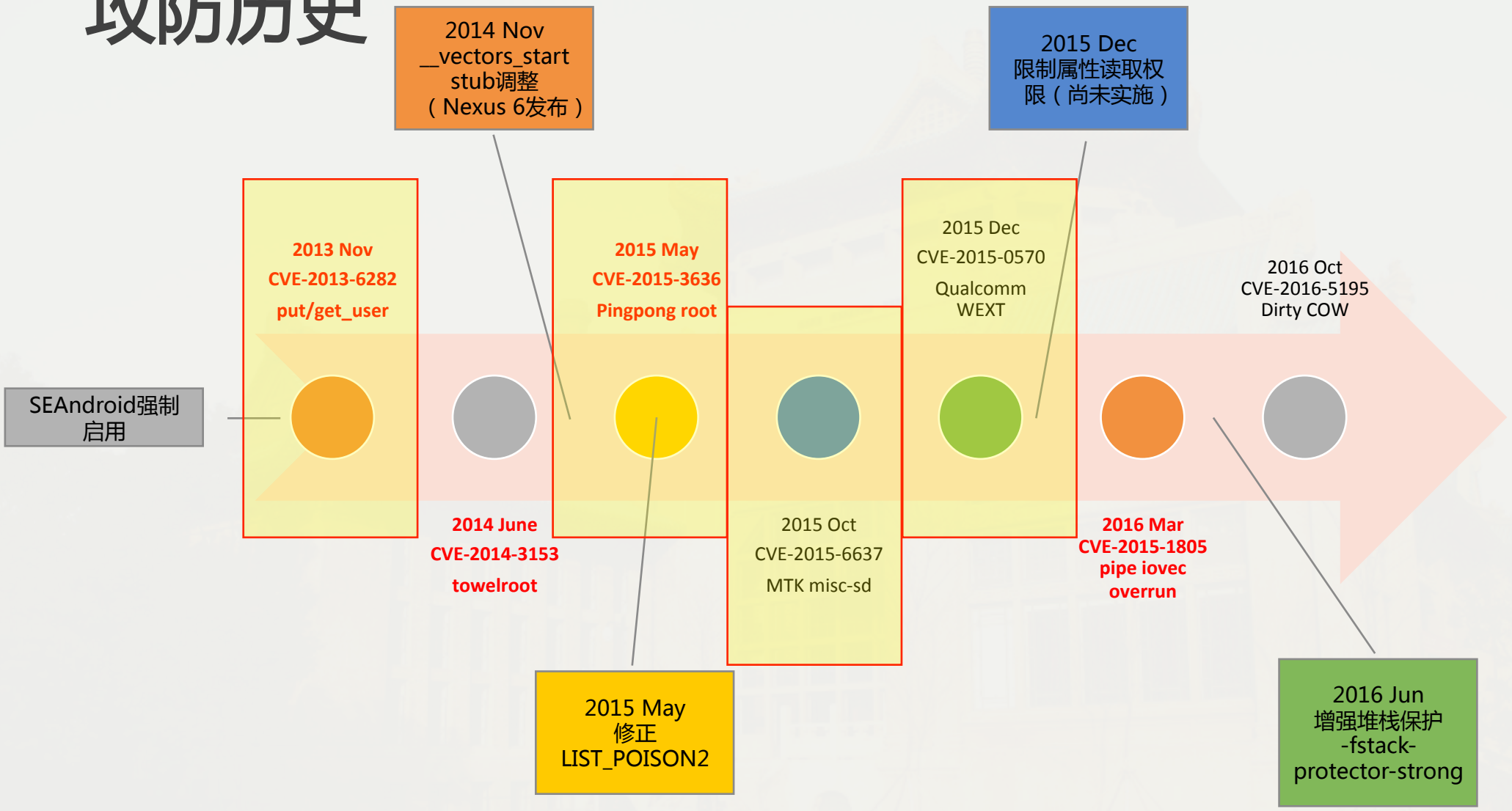
安卓使用Linux内核的策略

- 成熟稳定的内核版本
  - 2.6 → 3.4 → 3.10 → 3.18 → 4.1(?)
  - 新特性 ≈ 新漏洞
- 定制新的安全特性
  - 充分利用ARM架构特性
  - 深入定制并整合SELinux
  - 额外的访问控制加固
- 利用工具链提供的攻击缓解措施

攻防循环

- 减少攻击面
- 封堵利用技巧
- 修正实现错误

攻防历史

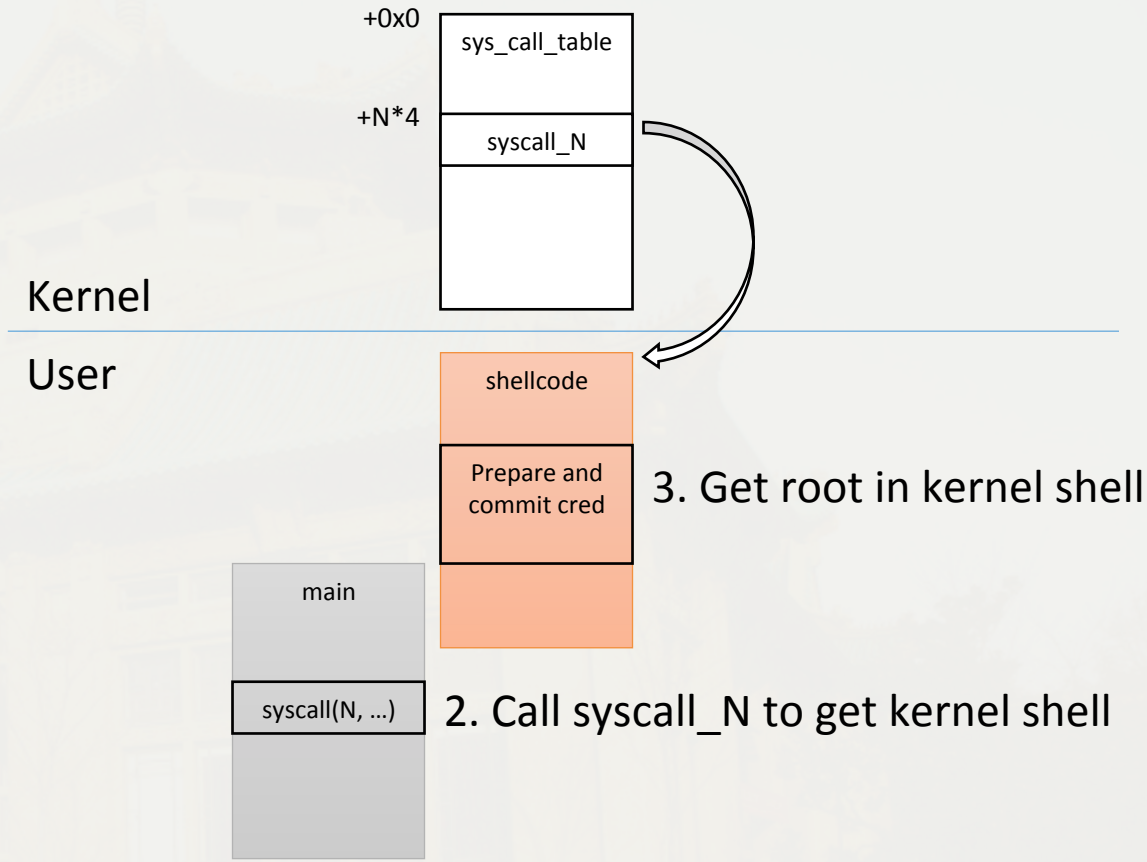

2014 Nov
__vectors_start
stub调整
（Nexus 6发布）
2015 Dec
限制属性读取权
限（尚未实施）
SEAndroid强制
启用
2013 Nov
CVE-2013-6282
put/get_user
2014 June
CVE-2014-3153
towelroot
2015 May
CVE-2015-3636
Pingpong root
2015 May
修正
LIST_POISON2
2015 Oct
CVE-2015-6637
MTK misc-sd
2015 Dec
CVE-2015-0570
Qualcomm
WEXT
2016 Mar
CVE-2015-1805
pipe iovec
overrun
2016 Oct
CVE-2016-5195
Dirty COW
2016 Jun
增强堆栈保护
-fstack-
protector-strong

案例分析

有效的防御措施

1) CVE-2013-6282

- 内核任意写入漏洞
- 2013年底出现通用利用
  - 360一键root
- 两个关键技巧
  - 通过ARM中断向量页（0xffff0000）获取sys_call_table的位置
  - 覆盖sys_call_table获得代码执行



1) CVE-2013-6282 利用方法

- 获取sys_call_table的方法：
  - 通过向量表第三项获得存放vector_swi地址的位置
  - 通过读取该位置的内容获得vector_swi的地址
  - 由于sys_call_table和vector_swi的相对位置固定，进而计算得到sys_call_table的地址

```
loc_vector_swi = *(0xffff0008 + 8 + ((*0xffff0008) & 0xfff)))
```

1) CVE-2013-6282 应对措施

- 修补：大幅修改ARM版本的uaccess.h，加入地址范围检查
- 防御措施：针对利用技巧
  - 封堵获得sys_call_table地址的方法
  - 引入代码和数据段只读保护（CONFIG_DEBUG_RODATA）

1) CVE-2013-6282 应对措施

- https://android.googlesource.com/kernel/msm/+/d303fa12f220d5e313f968d395556d8c8d3a1582%5E%21/#F1

```
diff --git a/arch/arm/kernel/entry-armv.S b/arch/arm/kernel/entry-armv.S
index c42a30a..2bf14b2 100644
--- a/arch/arm/kernel/entry-armv.S
+++ b/arch/arm/kernel/entry-armv.S
@@ -1035,9 +1035,9 @@
 /*
  * Vector stubs.
  *
- * This code is copied to 0xffff0200 so we can use branches in the
- * vectors, rather than ldr's.  Note that this code must not
- * exceed 0x300 bytes.
+ * This code is copied to 0xffff1000 so we can use branches in the
+ * vectors, rather than ldr's.  Note that this code must not exceed
+ * a page size.
...
__vectors_start:
…
-       W(ldr)  pc, .LCvswi + stubs_offset
…
+       W(ldr)  pc, __vectors_start + 0x1000
```

- 从高版本内核backport进安卓

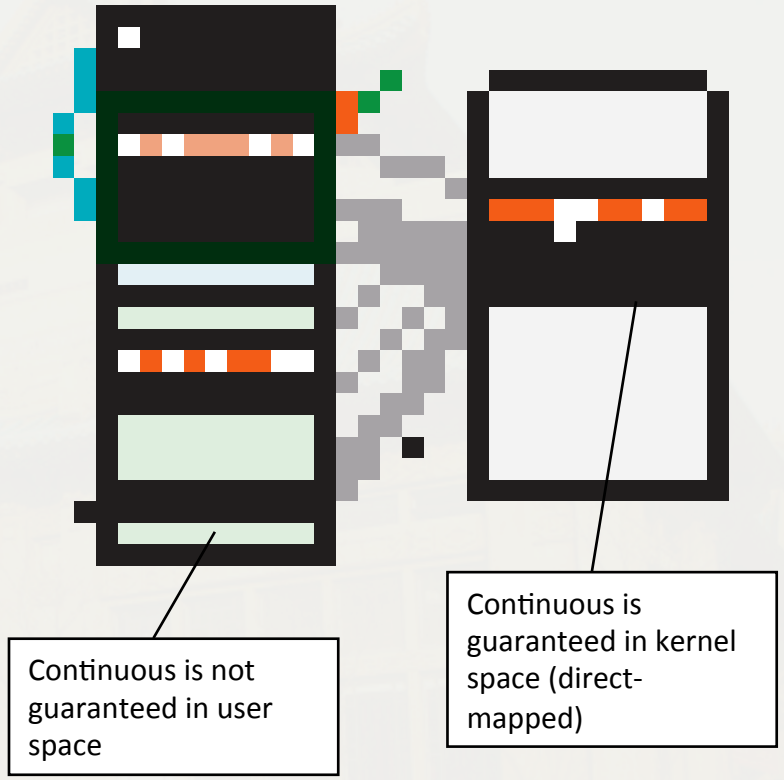
```
#ifdef CONFIG_DEBUG_RODATA
void mark_rodata_ro(void)
{
        set_section_perms(ro_perms, prot);
}

void set_kernel_text_rw(void)
{
        set_section_perms(ro_perms, clear);
}

void set_kernel_text_ro(void)
{
        set_section_perms(ro_perms, prot);
}
#endif /* CONFIG_DEBUG_RODATA */
```

1) CVE-2013-6282 应对措施

- 有效性：
  - 封堵了通用的获得sys_call_table的方法
  - 提高了覆盖sys_call_table作为shellcode入口的难度
- 缺点：
  - 仅仅针对了这个特定的利用，对于其他的信息泄露问题并没有作用
  - 内核页表依然可以被操作，将只读属性修改掉
  - 内核中依然存在很多可以用作入口的非只读数据，比如文件操作入口

2) CVE-2015-3636

- Ping socket UAF漏洞
- 主要难点在于如何填补free之后的ping_sock对象
- 利用direct mapped memory
  - 在用户态填补内核态的空闲页



2) CVE-2015-3636 利用方法

- 依赖于错误的LIST_POISON2
  - 0x200200任然属于有效地址，没有能够成功“下毒”
- 盲目填充并不能100%稳定利用，结合另外三个问题：
  - /proc/iomem → 提供物理内存布局
  - /proc/self/pagemap → 获得虚址和物理内存页的关系
  - CVE-2016-3809 → 泄露sock对象的内核地址

可以准确获知需要填充的sock对象对应的物理页，进而借此精确判断是否填充成功

2) CVE-2015-3636 应对措施

- 将LIST_POISON2指向0地址页

```
diff --git a/include/linux/poison.h b/include/linux/poison.h
index 2110a81..253c9b4 100644
--- a/include/linux/poison.h
+++ b/include/linux/poison.h
@@ -19,8 +19,8 @@
  * under normal circumstances, used to verify that nobody uses
  * non-initialized list entries.
  */
-#define LIST_POISON1  ((void *) 0x00100100 + POISON_POINTER_DELTA)
-#define LIST_POISON2  ((void *) 0x00200200 + POISON_POINTER_DELTA)
+#define LIST_POISON1  ((void *) 0x100 + POISON_POINTER_DELTA)
+#define LIST_POISON2  ((void *) 0x200 + POISON_POINTER_DELTA)
```

- 对非SYS_ADMIN用户禁用了/proc/self/pagemap

```
diff --git a/fs/proc/task_mmu.c b/fs/proc/task_mmu.c
index 956b75d..6dee68d 100644
--- a/fs/proc/task_mmu.c
+++ b/fs/proc/task_mmu.c
@@ -1325,6 +1325,9 @@ out:

 static int pagemap_open(struct inode *inode, struct file *file)
 {
+   /* do not disclose physical addresses: attack vector */
+   if (!capable(CAP_SYS_ADMIN))
+       return -EPERM;
    pr_warn_once("Bits 55-60 of /proc/PID/pagemap entries are about "
            "to stop being page-shift some time soon. See the "
            "linux/Documentation/vm/pagemap.txt for details.\n");
```

2) CVE-2015-3636 应对措施

- 有效性：
  - 修复了错误的LIST_POISON2，使得在触发路径上对列表做dereference操作的漏洞无法触发。
  - 禁用pagemap大大限制了direct mapped内存在漏洞利用中的效用

案例分析

被突破的防护措施

3) CVE-2015-0570

- 存在于高通WLAN驱动的WEXI API实现中
- 明显的堆栈溢出漏洞
- 编译器防护失效的案例

```
int wlan_hdd_set_filter(hdd_context_t *pHddCtx, tpPacketFilterCfg pRequest, tANI_U8 sessionId)
{
    tSirRcvPktFilterCfgType    packetFilterSetReq = {0};
    ...

    switch (pRequest->filterAction)
    {
        case HDD_RCV_FILTER_SET:
            ...
            for (i=0; i < pRequest->numParams; i++)
            {
                ...
                packetFilterSetReq.paramsData[i].dataLength = pRequest->paramsData[i].dataLength;
                ...
                memcpy(&packetFilterSetReq.paramsData[i].compareData,
                        pRequest->paramsData[i].compareData, pRequest->paramsData[i].dataLength);
                memcpy(&packetFilterSetReq.paramsData[i].dataMask,
                        pRequest->paramsData[i].dataMask, pRequest->paramsData[i].dataLength);
                ...
            }
        ...
    }
    return 0;
}
```

3) CVE-2015-0570 堆栈保护失效

- 依赖于GCC的-fstack-protector开关
- 默认情况下保护栈上大于等于8字节的“串”
  - 性能 vs. 安全
- 算法存在问题，无法识别以数组形式嵌套在内的结构体

```
struct PacketFilterParamsCfg
{
    uint8_t              protocolLayer;
    uint8_t              cmpFlag;
    uint8_t              dataOffset;
    uint8_t              dataLength;
    uint8_t              compareData[8];
    uint8_t              dataMask[8];
}



typedef struct
{
    uint8_t            filterAction;
    uint8_t            filterId;
    uint8_t            numParams;
    struct PacketFilterParamsCfg paramsData [5];
}tPacketFilterCfg, *tpPacketFilterCfg;
```

3)后续改进措施

- -fstack-protector-strong vs. -fstack-protector vs. **-fstack-protector-all**
- 高通和Google均建议使用新的-fstack-protector-strong代替-fstack-protector，在性能和安全之间获取更好的平衡
  - ~2% → ~20%
  - http://android-developers.blogspot.com/2016/07/protecting-android-with-more-linux.html
- 需要GCC 4.9或更高版本
  - https://gcc.gnu.org/ml/gcc-patches/2012-06/msg00974.html
- 定义了CONFIG_CC_STACKPROTECTOR_STRONG编译选项

4) CVE-2015-6637

- GeekPwn 2015演示项目
- 存在于MTK片上系统（SoC）的MMC驱动中，驱动本身被SELinux策略保护
- 典型的越界访问导致的代码执行

```
if(msdc_ctl->total_size <= 0)
        return -EINVAL;

    host_ctl = mtk_msdc_host[msdc_ctl->host_num];  <== Bug here
    BUG_ON(!host_ctl);
    BUG_ON(!host_ctl->mmc);
```

```
        if (host->ops->enable && !stop && host->claim_cnt == 1)
                host->ops->enable(host);    <== Code execution
```

4) CVE-2015-6637

- 触发漏洞需要的条件
  - root/system uid
  - 具有访问misc_sd_device字符设备的权限
- 可能有权限的上下文

```
# cat qiku_av.txt | grep "ALLOW " | grep ">misc_sd_device" | grep "ioctl"
[AV] 4378: ALLOW factory-->misc_sd_device (chr_file) [ioctl read open]
[AV] 7554: ALLOW em_svr-->misc_sd_device (chr_file) [ioctl read open]
[AV] 10552: ALLOW unconfineddomain-->misc_sd_device (file) [append create write ...
[AV] 10556: ALLOW recovery-->misc_sd_device (chr_file) [append create execute ...
[AV] 10559: ALLOW unconfineddomain-->misc_sd_device (chr_file) [append create ...
[AV] 10562: ALLOW vold-->misc_sd_device (chr_file) [ioctl read open]
[AV] 12202: ALLOW mmc_ffu-->misc_sd_device (chr_file) [ioctl read open]
```

4) CVE-2015-6637 攻击链？

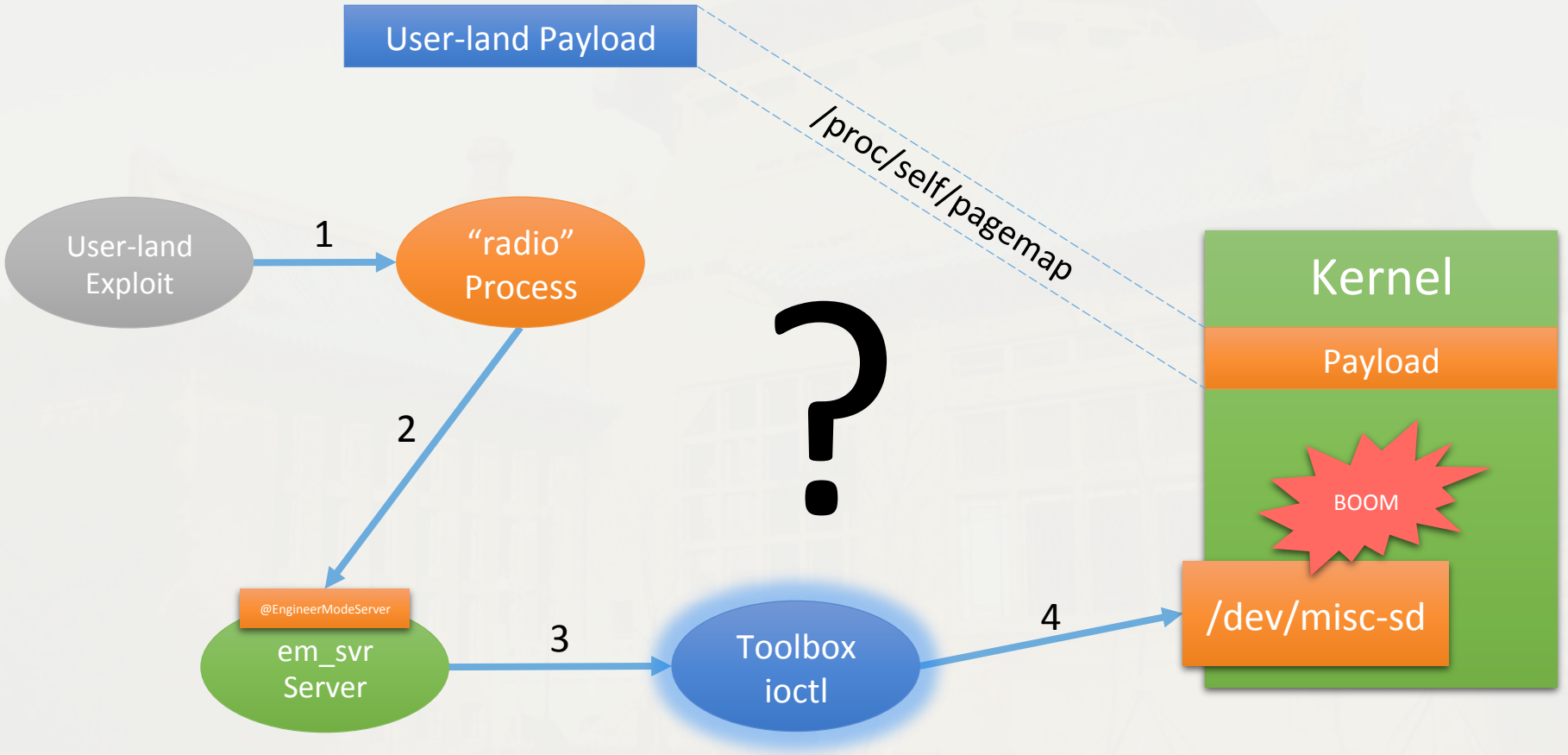
4) CVE-2015-6637

- 以em_svr为突破口，获取em_svr上下文的execve权限

```
# cat qiku_av.txt | grep "ALLOW " | grep "em_svr-->" | grep "execute"
[AV] 4418: ALLOW em_svr-->system_file (file) [execute_no_trans]
[AV] 5393: ALLOW em_svr-->shell_exec (file) [execute read execute_no_trans open]
[AV] 6076: ALLOW em_svr-->thermal_manager_exec (file) [execute getattr read ...
[AV] 7135: ALLOW em_svr-->em_svr_exec (file) [execute getattr read entrypoint open]
```

- SELinux策略限制了em_svr调取/data分区下的payload
- 但是我们可以借助ioctl命令行工具触发漏洞
- 如何给这个命令行传递参数？

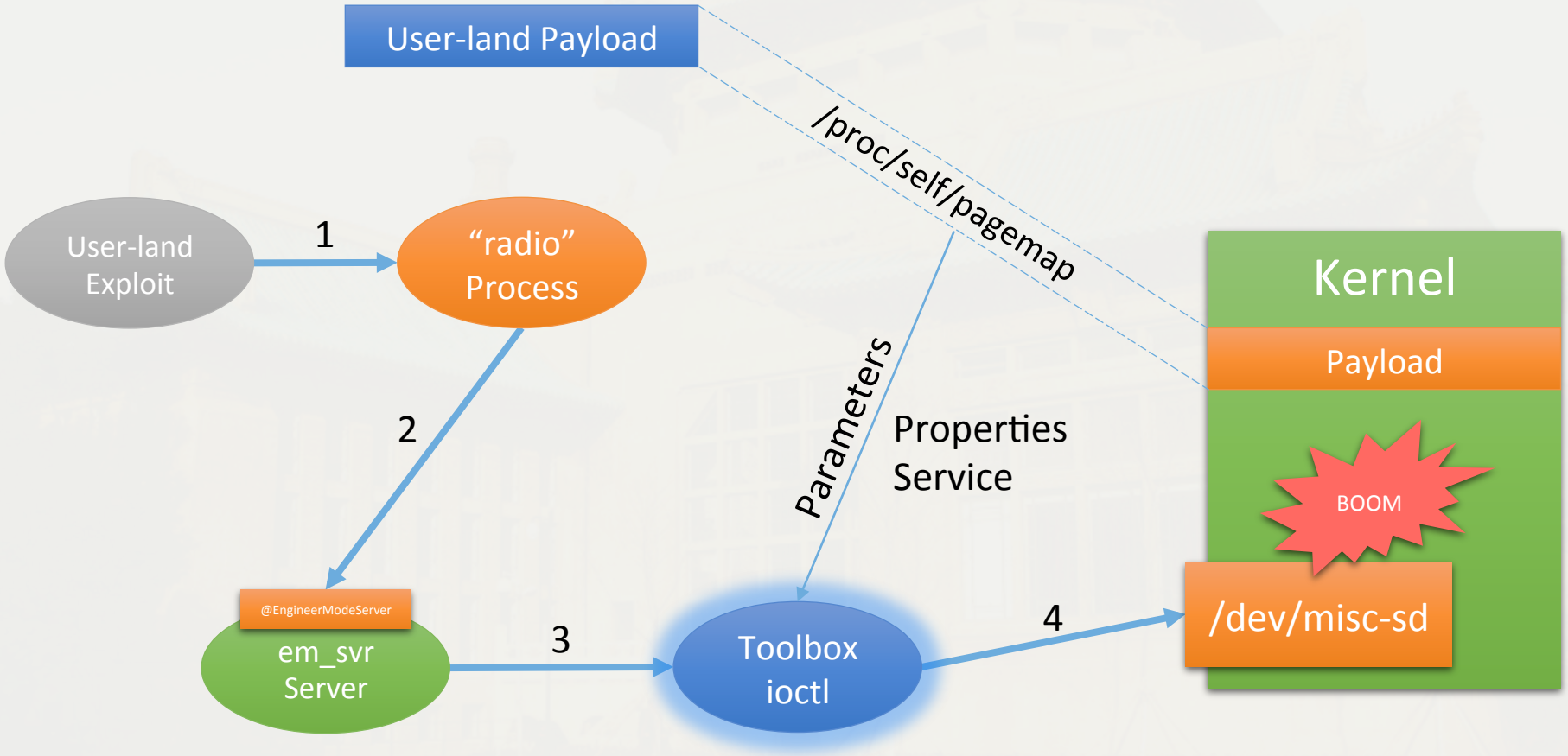
4) CVE-2015-6637 攻击链？



4) CVE-2015-6637

- 传统IPC渠道也被SELinux策略封堵
- 属性服务是个例外
  - 设置属性受SELinux策略限制
  - 读取属性完全不受限
- 利用方法
  - 通过untrusted_app可以设置的属性
  - 在em_svr的上下文中读取并使用

4) CVE-2015-6637 完整攻击链



4) 后续改进措施

- SELinux的策略配置非常完善，依靠ioctl工具和property服务才得以绕过
- Google在2015年12月开始研究加入对property的读权限控制
- https://android.googlesource.com/platform/system/sepolicy/+/949d7cbc29c1a658f00b966a81fd3f710c065fec
- 相关策略依旧在开发中

5) 其它

- 在一些攻击面连续出现较严重安全问题后，Google通常也会考虑限制其访问
- 最近的例子
  - Wireless Extension
  - Performance Counter
- 延伸阅读
  - https://www.blackhat.com/eu-16/briefings.html#rooting-every-android-from-extension-to-exploitation

总结

- 减少攻击面
  - SELinux
  - Capability
  - 直接禁用
- 封堵特定利用技巧
  - 复制PC的老路
- 修正错误实现
  - LIST_POISON2
- 漏洞和攻击手段促进安卓各类防护机制的进步
- 没有一劳永逸的防护方案，对抗会长期存在

感谢您的关注！