GIAC | 全球互联网架构大会
GLOBAL INTERNET ARCHITECTURE CONFERENCE

Android框架虚拟化实战

董福源
360手机卫士

2017.thegiac.com

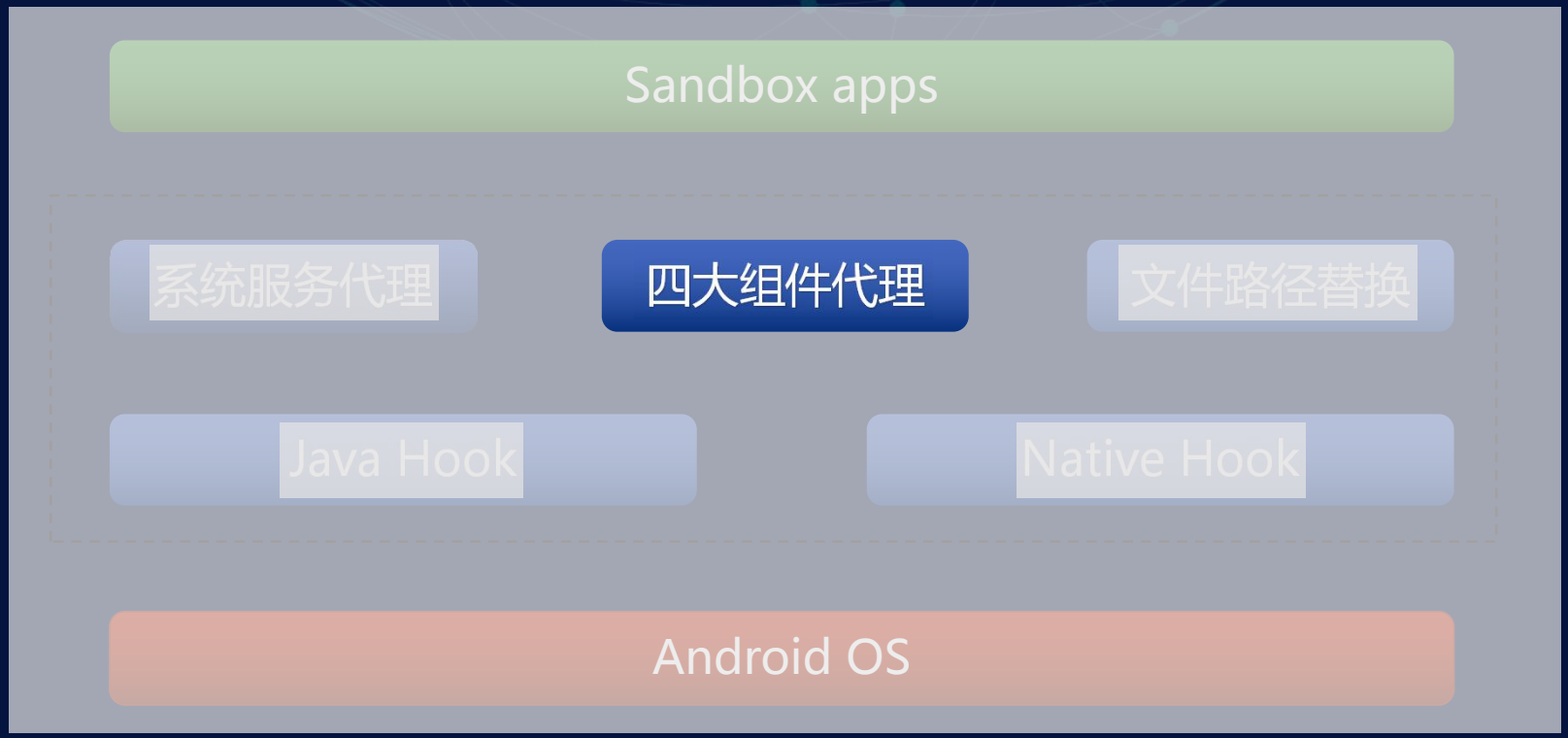
什么是**虚拟化**

原生apk　　在**封闭系统**内　　**免安装**运行

Android系统的一种**沙箱技术**

技术架构

目录

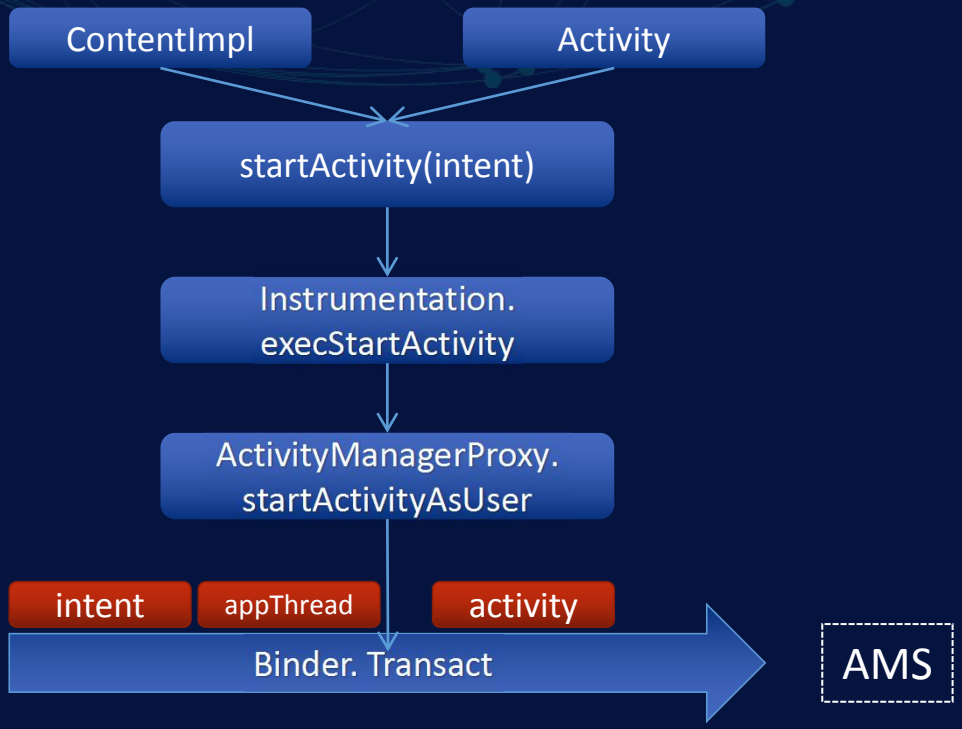


Activity插件化

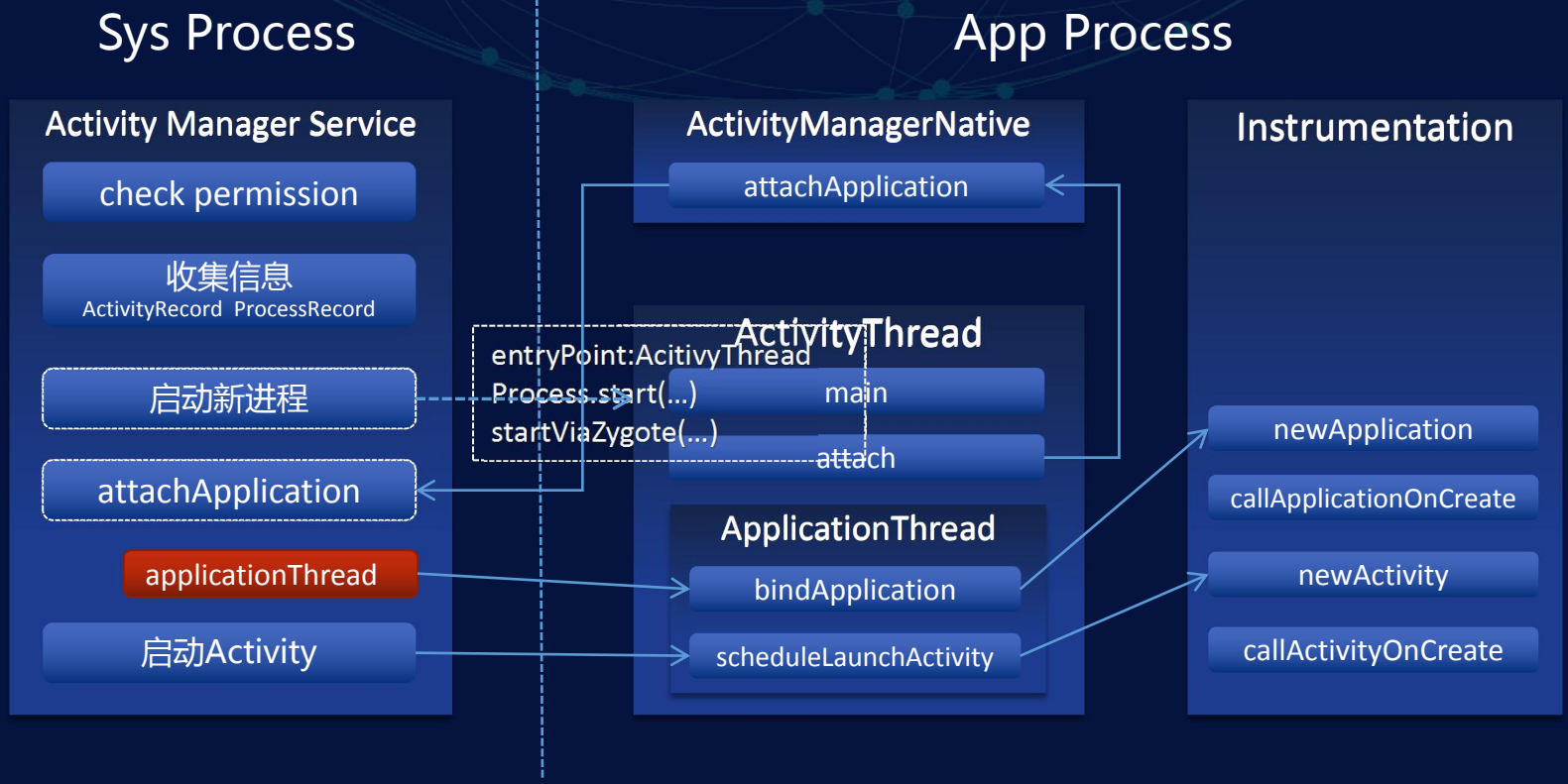
只能启动**Manifest**中声明的activity

占坑方案 → 欺骗AMS

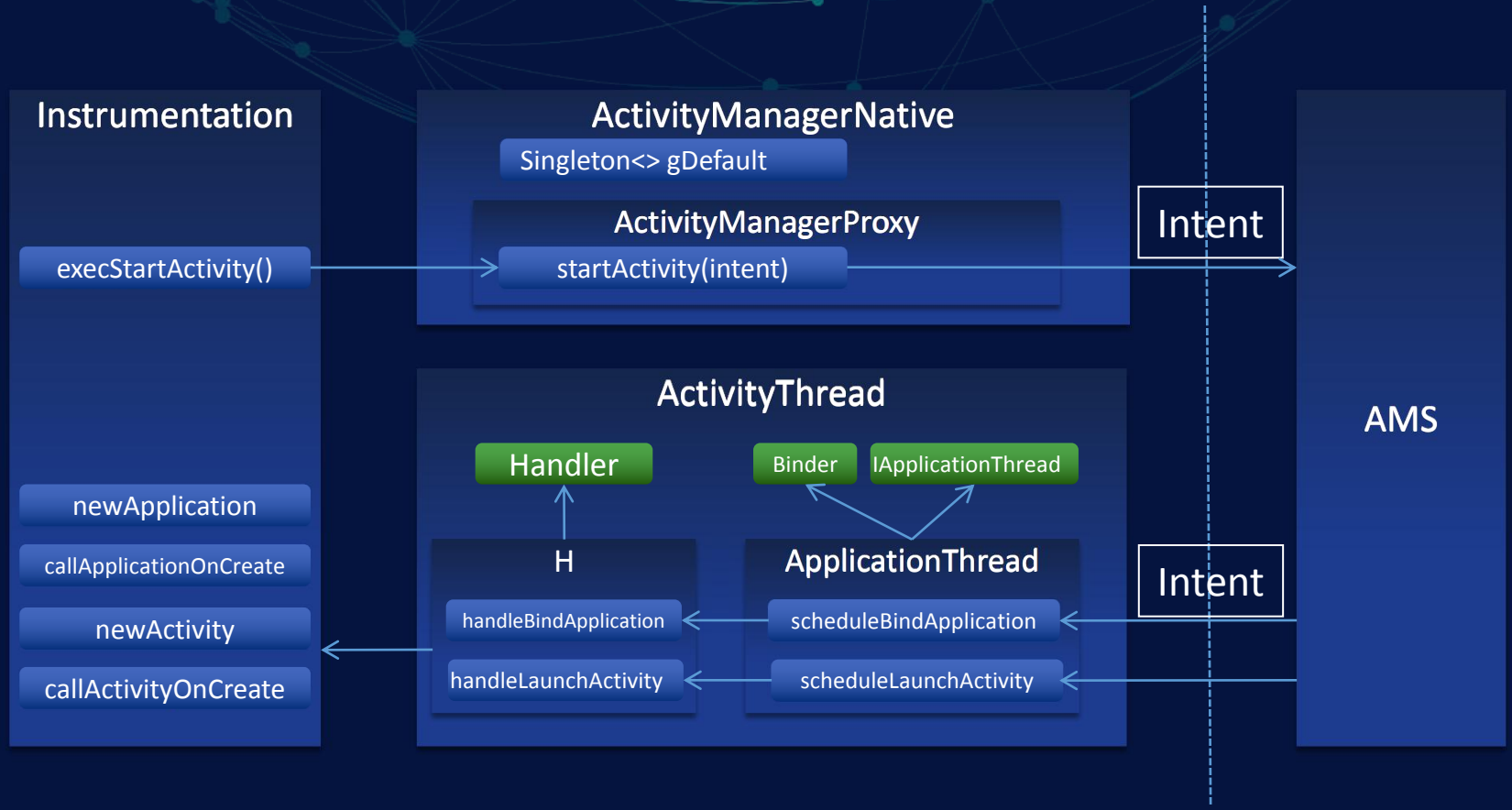
Activity启动过程



Activity启动过程



关键类结构



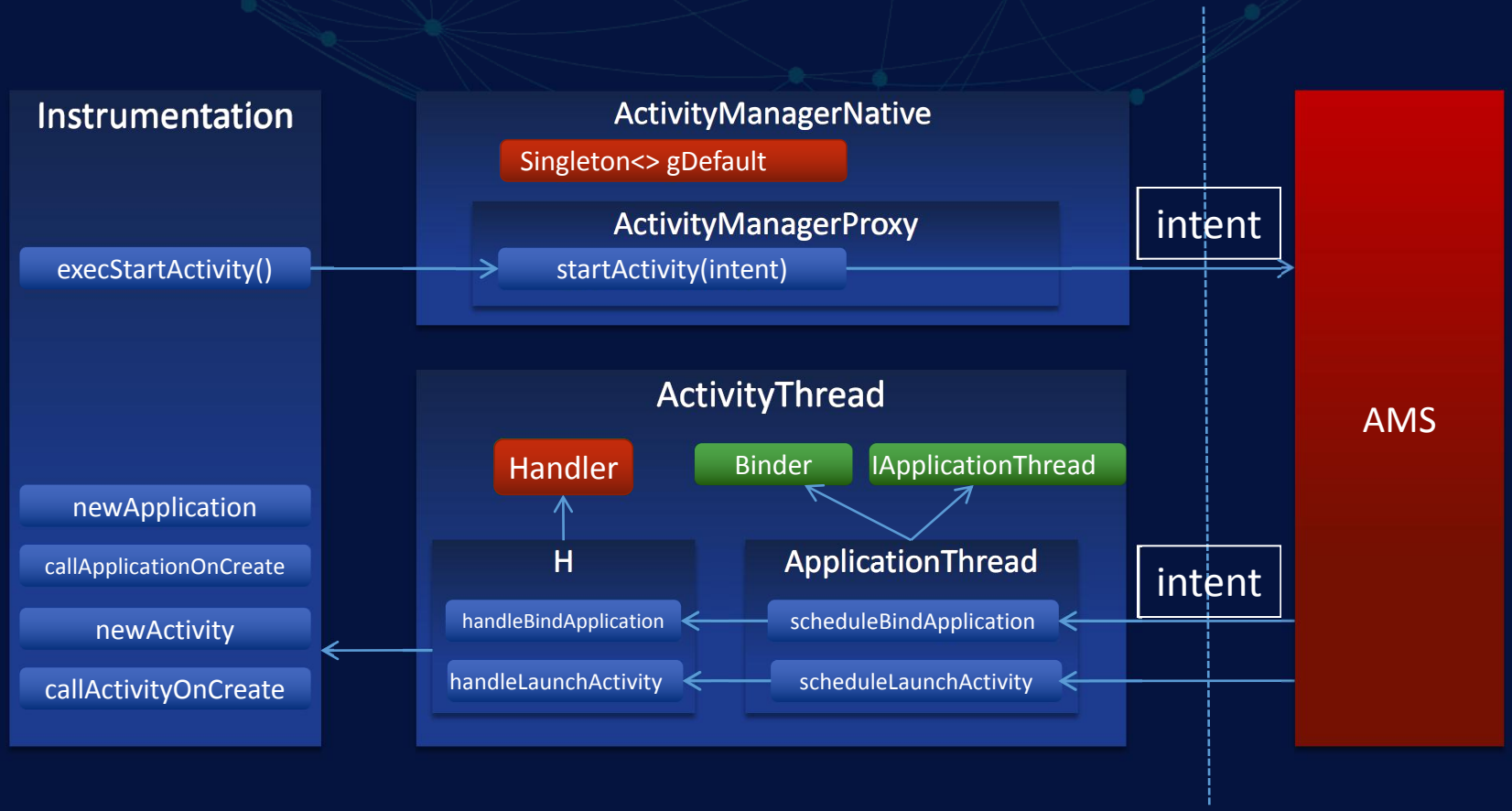
GIAC | 全球互联网架构大会
GLOBAL INTERNET ARCHITECTURE CONFERENCE

Java Hook

如何欺骗AMS?

Java Hook

Java Hook



Java Hook

Singleton

```
private static final Singleton<IActivityManager> gDefault = new Singleton<IActivityManager>() {
    protected IActivityManager create() {
        IBinder b = ServiceManager.getService("activity");
        if (false) {
            Log.v("ActivityManager", "default service binder = " + b);
        }
        IActivityManager am = asInterface(b);
        if (false) {
            Log.v("ActivityManager", "default service = " + am);
        }
        return am;
    }
};
```

Handler.mCallback

```
public void dispatchMessage(Message msg) {
    if (msg.callback != null) {
        handleCallback(msg);
    } else {
        if (mCallback != null) {
            if (mCallback.handleMessage(msg)) {
                return;
            }
        }
        handleMessage(msg);
    }
}
```

Activity插件化

只有1个桩

standard模式

唯一的taskAffinity

```
<activity android:name="${applicationId}.stub.ActivityProxy$P0" android:exported="false"
    android:process=":Plugin0" android:taskAffinity="${applicationId}.activity.stack0"
    android:configChanges="keyboard|keyboardHidden|navigation|orientation|screenSize" android:hardwareAccelerated="true" />
```

Activity插件化

多个Activity？

多实例的桩代替

多种模式？

用standard的桩模拟

维护ActivityStack

Activity管理

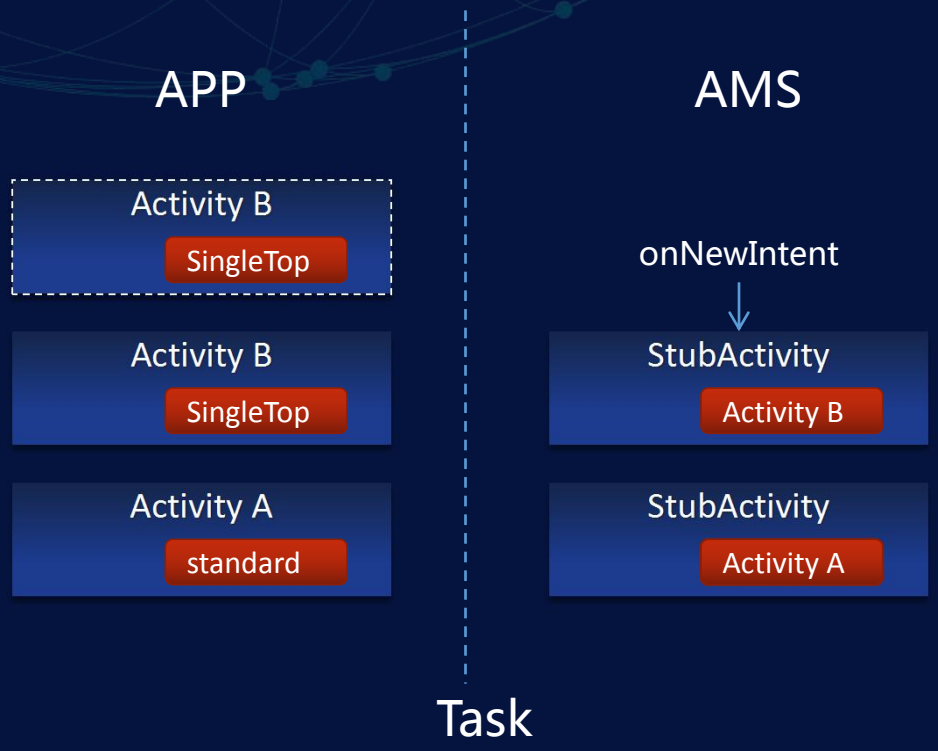
桩管理

Activity插件化

模拟SingleTop

检测栈顶

FLAG_ACTIVITY_SINGLE_TOP



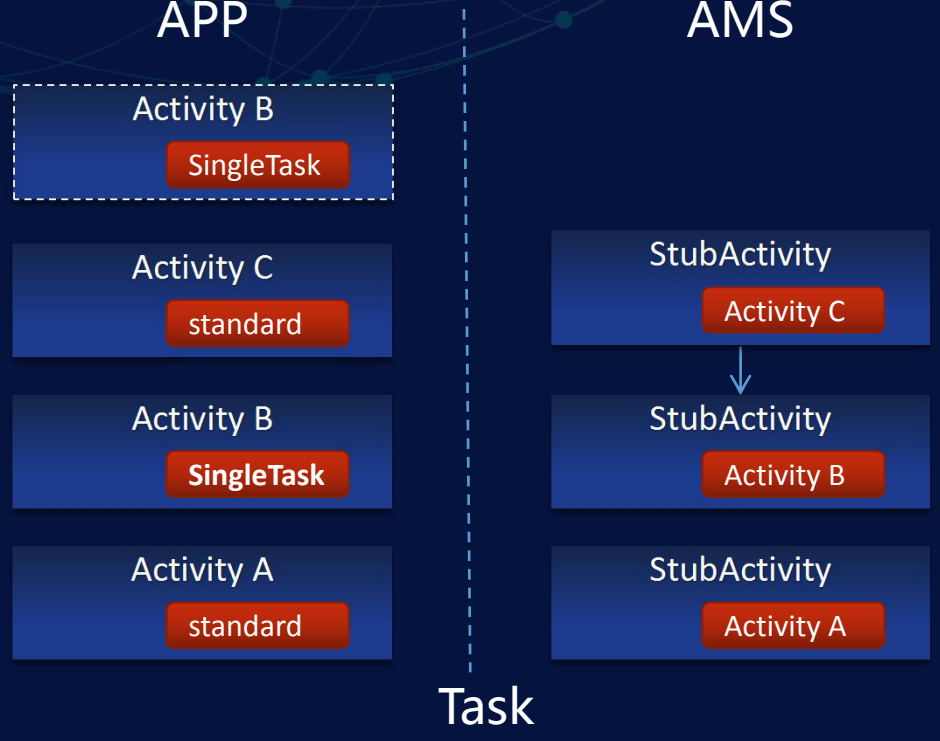
Activity插件化

模拟SingleTask

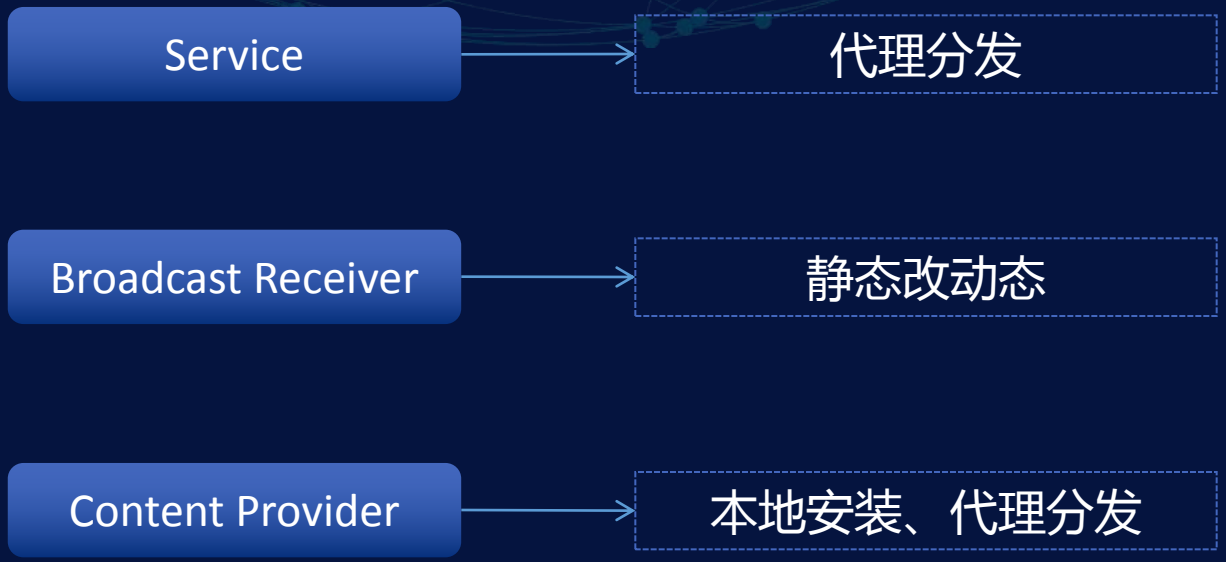
检测栈

清除Activity C

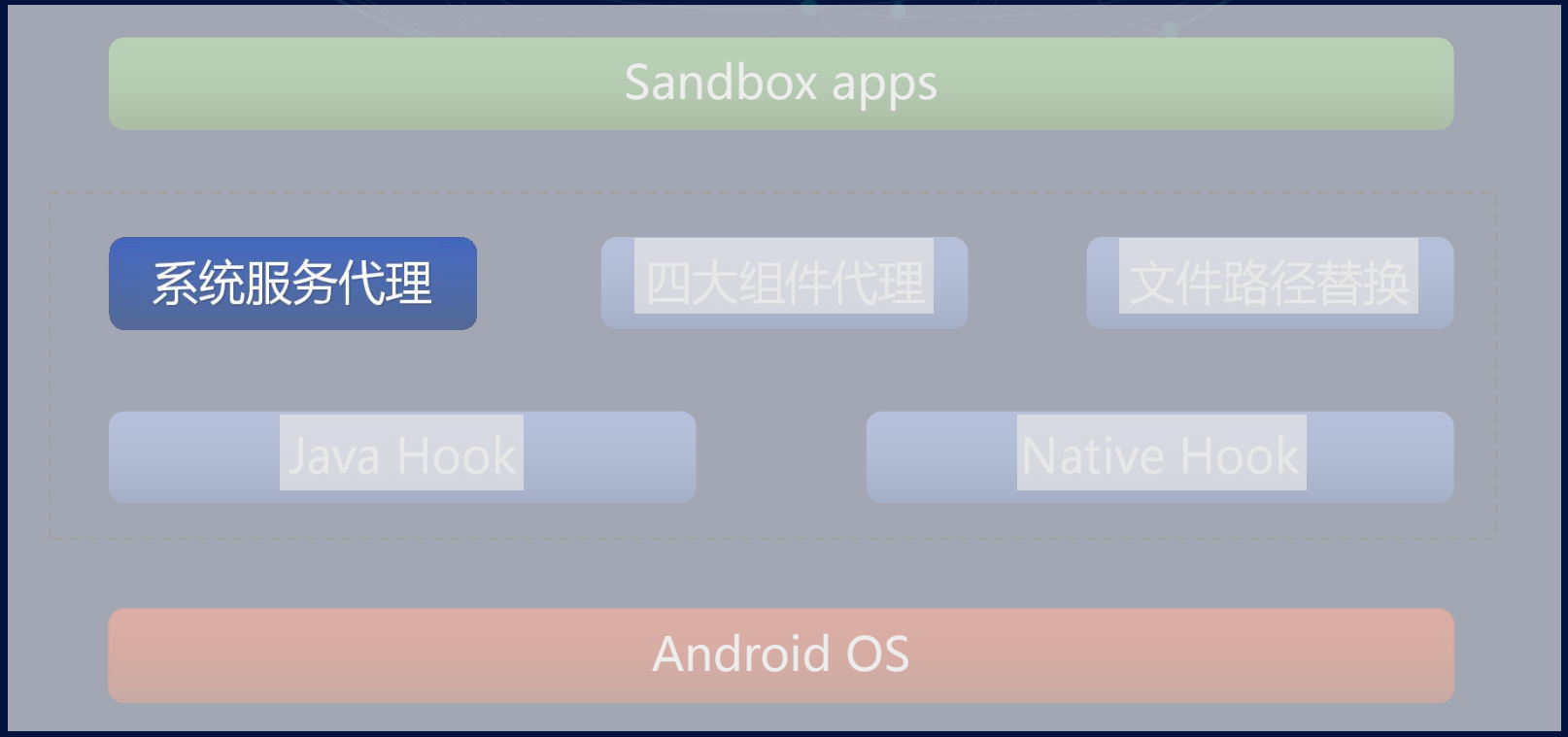
FLAG_ACTIVITY_SINGLE_TOP



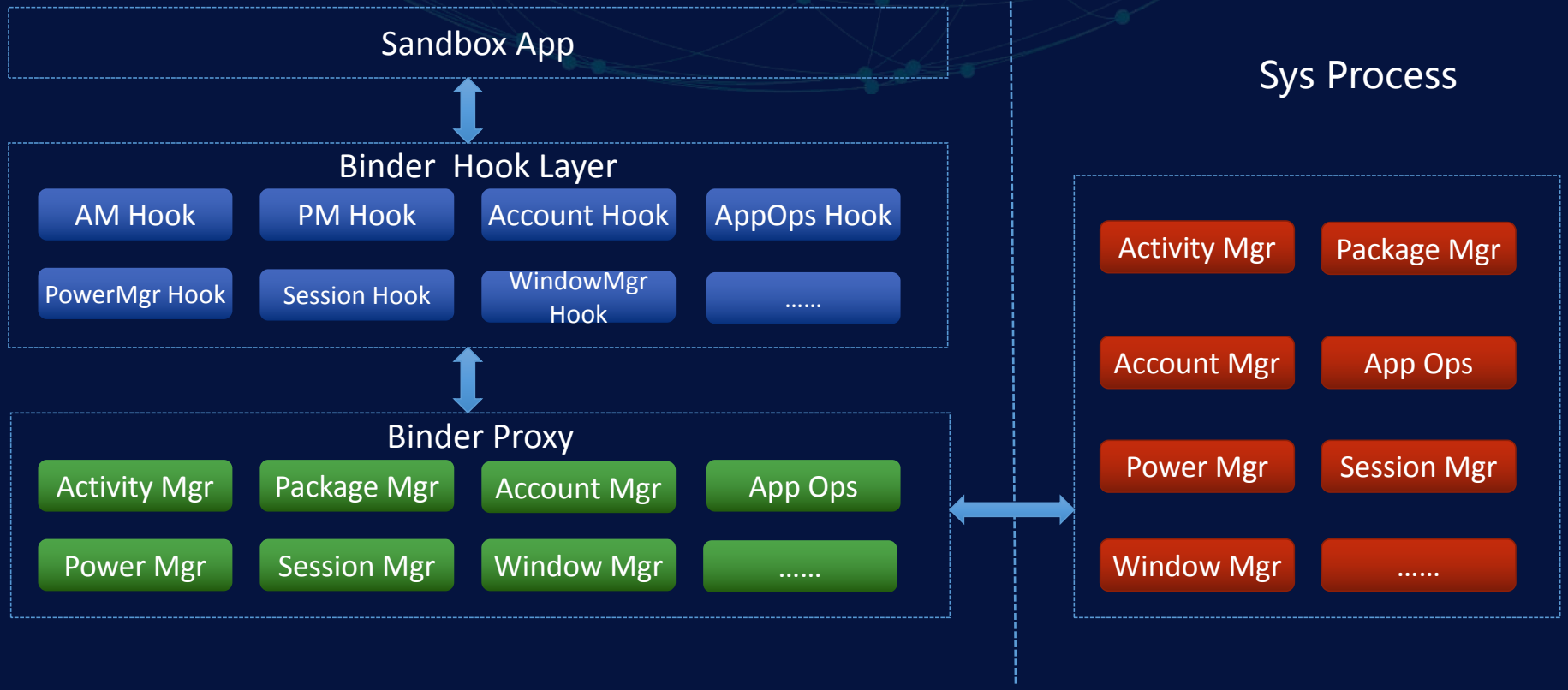
其它组件



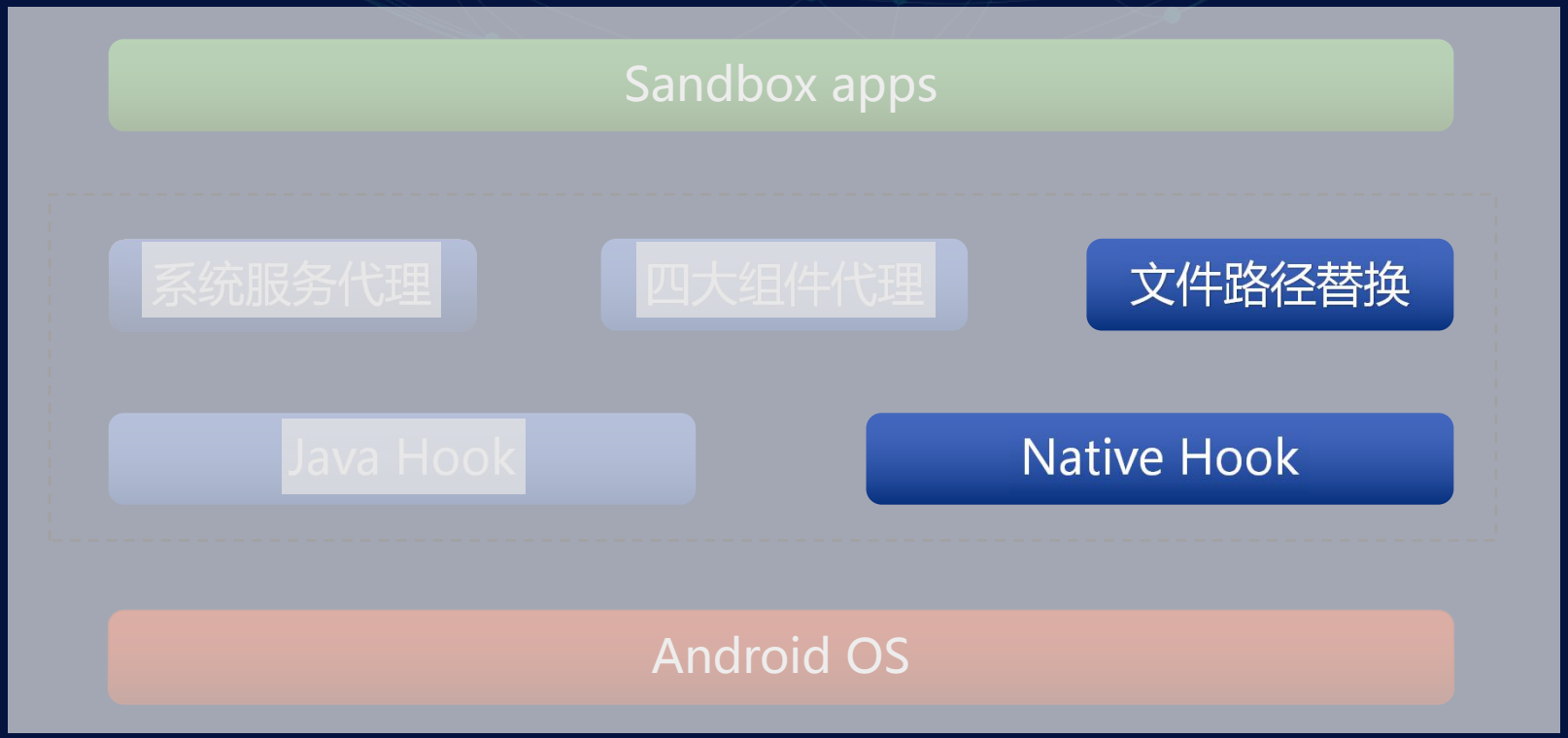
目录



系统服务



目录



文件重定向

路径替换

SD卡目录隔离

目录重定向

Native IO Hook 运行时替换

Inline hook

分配空间存放前2条指令 - orig

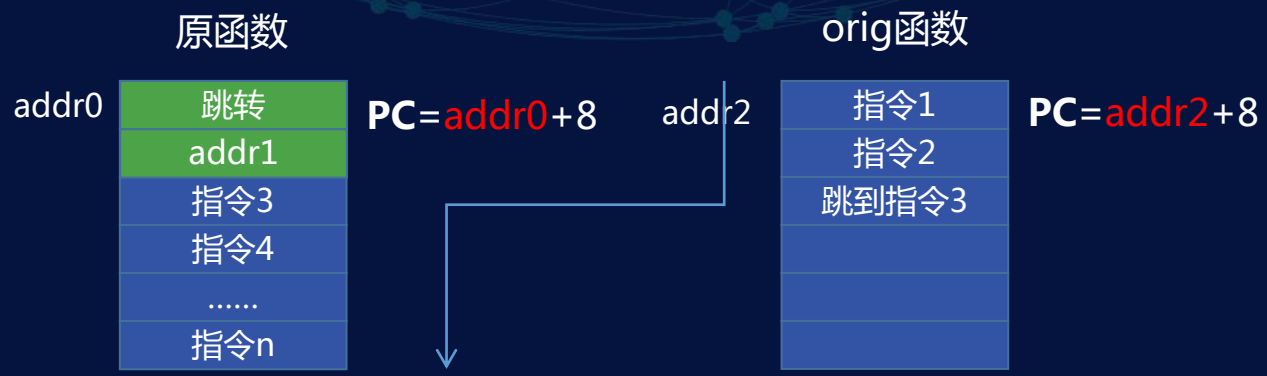
修改原函数前2条指令，跳转到hook函数

执行orig函数

核心问题

跨地址空间执行 → 指令转译

核心问题



与PC相关的指令需要转译

指令转译

跳转：

ARM指令：

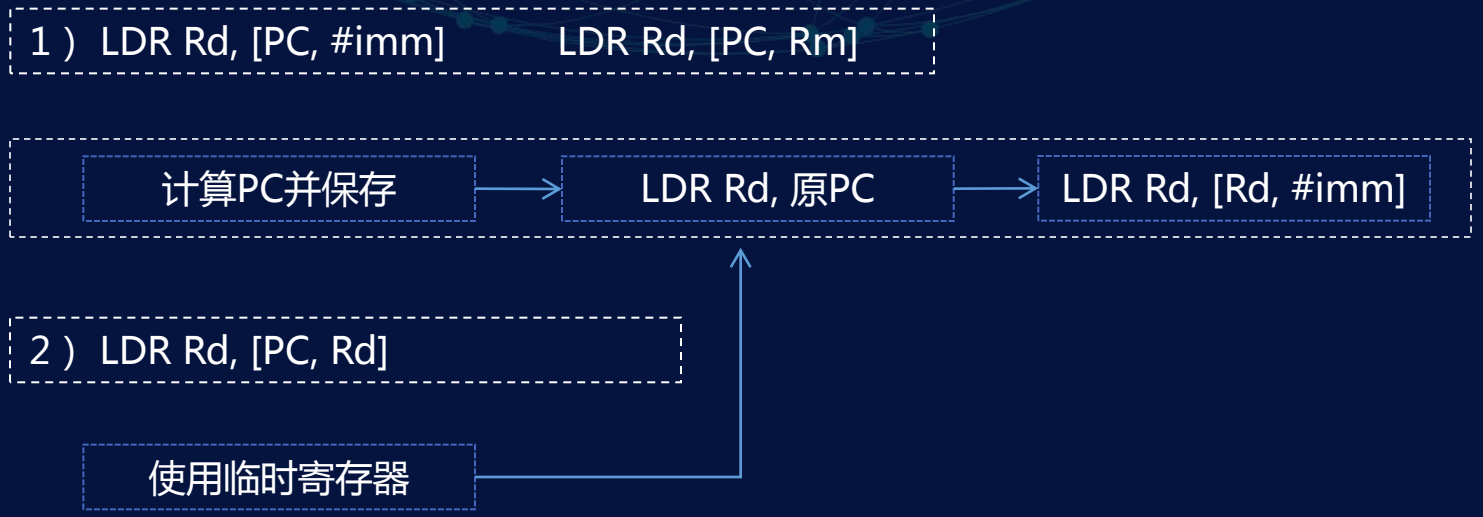
LDR, PC, PC+1　　e51ff004

addr1　　只需要替换原函数的前2个指令

在前2条出现的只有**1**种指令

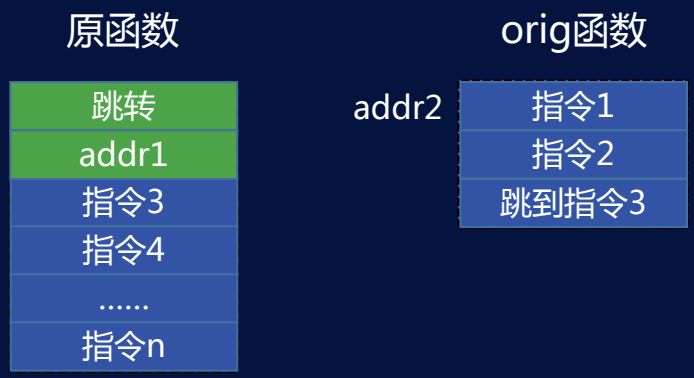
Single Data Transfer类指令　　Rn=PC

指令转译



Inline Hook

- 分配orig空间
- 将orig映射到内存空间
- 复制指令(转译)
- 将orig改为可执行
- 将原函数改为可写
- 替换指令



Inline Hook

1 分配orig空间

2条备份指令 + 2条指令跳回 + 转码指令

Inline Hook

2 将orig映射到内存空间

```
uint32_t *buffer(reinterpret_cast<uint32_t *>(mmap(
        NULL, length, PROT_READ | PROT_WRITE, MAP_ANON | MAP_PRIVATE, -1, 0
)));
```

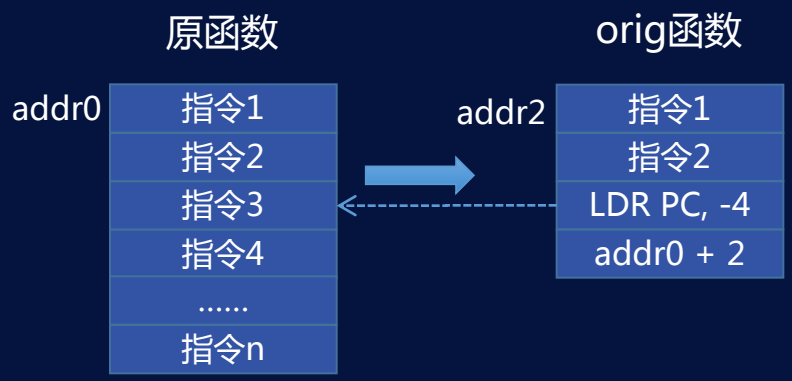
mmap　　创建内存映射

- length为第1步计算出来的大小
- PROT_READ | PROT_WRITE 可读写权限
- MAP_ANON, fd = -1 匿名内存映射。相当于分配一块内存，并映射到内存空间

Inline Hook

3 复制指令

1) 普通指令



Inline Hook

3 复制指令

2) LDR Rd, [PC, #imm] LDR Rd, [PC, Rm]

指令1：LDR R5, [PC, imm]

PC：addr0 + 8

指令a：LDR R5, [PC, #8]

指令b：LDR R5, [R5, imm]

两次赋值Rd 代替 直接使用PC

Inline Hook

3 复制指令

3) LDR Rd, [PC, Rd]

指令1：LDR R5, [**PC**, R5]

Rd=Rm, 不能用Rd临时存储PC

其它过程与2)相同

与2)相同，使用新寄存器，操作前PUSH，操作后POP

Inline Hook

4 将orig改为可执行

```
if (mprotect(buffer, length, PROT_READ | PROT_EXEC) == -1) {
    LOGE("MS:Error:mprotect():%d", errno);
    goto fail;
}
```

mprotect　　将内存改为可执行

5 将原函数改为可写

参数改为：PROT_WRITE

Inline Hook

THUMB指令集

短跳转	无法实现跨越addr0到addr2的跳转
需要转为ARM指令	用ARM指令实现跳转
指令对齐	4字节对齐
比ARM复杂很多	替换6-7条指令，实现十几种指令的转译

Example

App : com.demo1

Write file : /data/data/com.demo1/files/test.txt

Hook

```
void *func = dlsym(lib, "_open");
```

fopen_hook(file)

路径替换

```
fopen_hook(file) {
……
return func_orig(file);
}
```

/data/data/com.demo1/files/test.txt

/data/data/com.docker/Apps/com.demo1/files/test.txt

总结

End thanks !

技术交流：奇卓社(手机卫士技术公众号)

GIAC | 全球互联网架构大会
GLOBAL INTERNET ARCHITECTURE CONFERENCE

GIAC

全球互联网架构大会

GLOBAL INTERNET ARCHITECTURE CONFERENCE

扫码关注GIAC公众号

2017.thegiac.com