# TODO

**Android系统安全和反编译实战**

[**https://download.csdn.net/download/wutianxu123/10227428**](https://download.csdn.net/download/wutianxu123/10227428)

# 系统安全架构

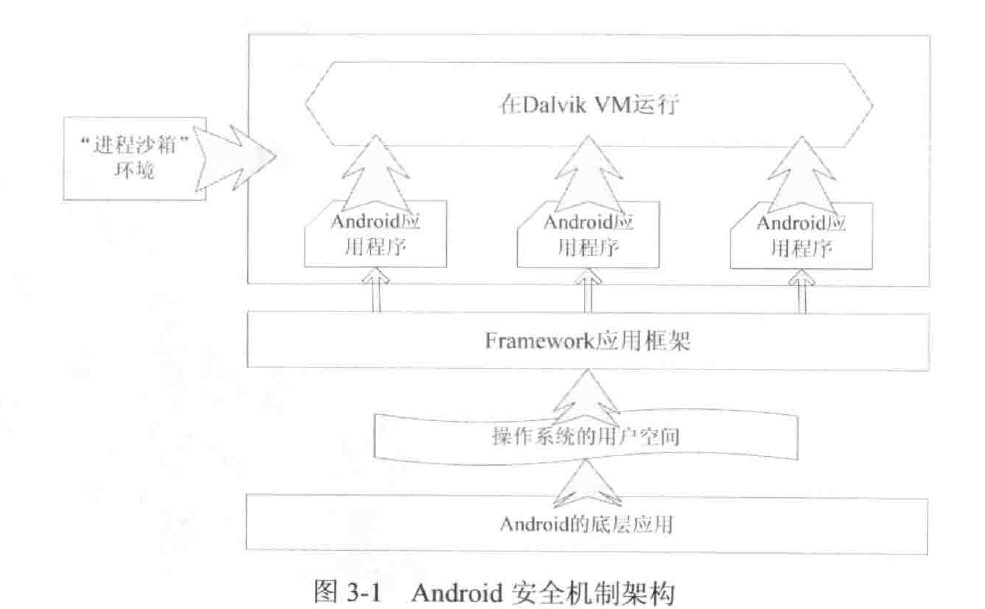
# Android系统的安全机制

## 4.1 Android安全机制概述 61

Android安全机制概述

使用进程沙箱机制来隔离进程资源

通过android系统独有的内存管理技术，安全高效地实现进程之间的通信处理



## 4.1.1 Android的安全机制模型 61

在应用层，提供了如下安全机制的模型

1. 运行时权限控制
2. 签名机制，实现应用程序之间的信息信任和资源共享

机制特点

不同的层次架构机制来保护用户信息安全，保持灵活性

引导用户注意安全

时刻防范

尽快止损

内存管理

权限声明

App签名

访问控制

进程沙箱隔离

进程通信

4.1.2 Android的安全框架概述 62

4.2 分析Linux系统的安全机制 63

4.2.1 Linux用户权限基础 63

4.2.2 进程 68

4.3 分析Android系统的安全机制 70

4.3.1 沙箱模型介绍 70

4.3.2 应用程序的安全机制 72

4.3.3 分区加载机制 73

# 第5章 内存安全机制——匿名共享

内存系统 74

5.1 分析Ashmem驱动程序 74

5.1.1 基础数据结构 74

5.1.2 初始化处理 75

5.1.3 打开匿名共享内存

设备文件 76

5.1.4 内存映射 78

5.1.5 实现读写操作 79

5.1.6 锁定和解锁 81

5.1.7 回收内存块 86

5.2 分析C++访问接口层 87

5.2.1 接口MemoryBase 87

5.2.2 客户端实现 90

5.2.3 接口MemoryBase 94

5.3 分析Java访问接口层 97

5.4 内存优化机制 100

5.4.1 sp和wp简析 100

5.4.2 详解智能指针 102

5.5 Android内存系统的安全

机制分析 118

5.5.1 Ashmem匿名共享内存

的机理 119

5.5.2 使用Low Memory Killer

机制实现安全和高效 119

第6章 Binder通信安全机制（上） 121

6.1 Binder机制基础 121

6.1.1 选择Binder机制的

原因——简洁快速、

低耗内存、更加安全 121

6.1.2 Binder安全机制的

必要性 122

6.1.3 Android的进程间通信

（IPC）机制Binder 122

6.1.4 Service Manager是Binder

机制的上下文管理者 123

6.1.5 Service Manager服务 137

6.2 分析Binder驱动程序 140

6.2.1 分析数据结构 140

6.2.2 分析设备初始化 150

6.2.3 打开Binder设备文件 151

6.2.4 内存映射 152

6.2.5 释放物理页面 156

6.2.6 分配内核缓冲区 157

6.2.7 释放内核缓冲区 158

6.2.8 查询内核缓冲区 160

第7章 Binder通信安全机制（下） 162

7.1 Binder封装库 162

7.1.1 类BBinder 163

7.1.2 类BpRefBase 165

7.1.3 类IPCThreadState 166

7.2 初始化Java层Binder框架 168

7.3 分析MediaServer的通信机制 170

7.3.1 MediaServer的入口函数 170

7.3.2 ProcessState 171

7.3.3 defaultServiceManager 173

7.3.4 注册MediaPlayerService 179

7.3.5 分析StartThread Pool和

join Thread Pool 188

7.4 总结进程通信机制的安全性 190

7.4.1 进程先线程安全 190

7.4.2 远程过程调用机制

（RPC） 191

7.4.3 实现线程安全方法 192

7.4.4 Binder中的安全策略 193

第8章 Android虚拟机基础 194

8.1 Dalvik VM和JVM的差异 194

8.2 Dalvik虚拟机的主要特征 195

8.3 Dalvik VM架构 196

8.3.1 Dalvik虚拟机的代码结构 196

8.3.2 dx工具 198

8.3.3 Dalvik VM的进程管理 198

8.3.4 Android的初始化流程 198

8.4 Dalvik VM控制VM命令详解 199

8.4.1 基本命令 199

8.4.2 扩展的JNI检测 199

8.4.3 断言 200

8.4.4 字节码校验和优化 200

8.4.5 Dalvik VM的运行模式 201

8.4.6 死锁预测 201

8.4.7 dump堆栈追踪 202

8.4.8 dex文件和校验 202

8.4.9 产生标志位 202

8.5 Dalvik VM进程管理 202

8.5.1 Zygote基础 202

8.5.2 Dalvik的进程模型 211

8.5.3 Dalvik的进程通信 215

8.6 Zygote（孕育）进程 218

8.6.1 Zygote基础 218

8.6.2 分析Zygote的启动过程 219

第9章 Dalvik VM的运作流程 233

9.1 Dalvik VM相关的可执行程序 233

9.1.1 dalvikvm、dvz和

app\_process简介 233

9.1.2 对比app\_process和

dalvikvm的执行过程 234

9.2 初始化Dalvik VM 236

9.2.1 开始虚拟机的准备工作 236

9.2.2 初始化跟踪显示系统 237

9.2.3 初始化垃圾回收器 237

9.2.4 初始化线程列表和

主线程环境参数 237

9.2.5 分配内部操作方法的

表格内存 238

9.2.6 初始化虚拟机的指令码

相关的内容 238

9.2.7 分配指令寄存器状态

的内存 239

9.2.8 分配指令寄存器状态

的内存和最基本用的

Java库 239

9.2.9 初始化使用的Java类库

线程类 240

9.2.10 初始化虚拟机使用的

异常Java类库 241

9.2.11 初始化其他对象 242

9.3 启动Zygote 250

9.3.1 在init.rc中配置zygote

启动参数 250

9.3.2 启动Socket服务端口 250

9.3.3 加载preload-classes 251

9.3.4 加载preload-resources 252

9.4 启动

---------------------

作者：weixin\_34025151

来源：CSDN

原文：https://blog.csdn.net/weixin\_34025151/article/details/87182992

版权声明：本文为博主原创文章，转载请附上博文链接！

# REF