

操作系统与虚拟化安全课题研究

|  |  |
| --- | --- |
| 题目： | Android安全体制实现分析 |

|  |  |
| --- | --- |
| 成 员： | 龙东恒 王帅 宋文浩 |
| 院 系： | 软件与微电子学院 |
| 导师姓名： | 沈晴霓 |

二〇一七年 十二月

# **目录**

[**目录** 1](#_Toc502827091)

[**一、选题背景** 2](#_Toc502827092)

[**二、AVD安装** 3](#_Toc502827093)

[2.1 Windows 4](#_Toc502827094)

[2.2 Mac 4](#_Toc502827095)

[**三、Android安全机制** 5](#_Toc502827096)

[3.1安全机制简述 5](#_Toc502827097)

[3.2 安全机制迭代更新过程 5](#_Toc502827098)

[3.3 权限管理——核心机制 5](#_Toc502827099)

[**四、权限管理** 6](#_Toc502827100)

[4.1 权限管理介绍 6](#_Toc502827101)

[4.2 权限管理版本迭代 6](#_Toc502827102)

[4.3 权限管理现有问题 6](#_Toc502827103)

[4.4 权限管理改进建议 6](#_Toc502827104)

[五、实验总结 7](#_Toc502827105)

# **一、选题背景**

简要讲述Android近来的市场规模，以及2016年Android安全报告简述。

# **二、AVD安装**

Android虚拟设备（AVD）是一种仿真器配置，允许开发人员通过模拟真实设备功能来测试应用程序，开发人员无需使用装有Android操作系统的手机、平板电脑及相关电子设备，在AVD中开发人员可以完成绝大部分的开发、测试工作。AVD允许开发人员通过硬件和软件设备对其参数（如API版本、底层Android系统等）进行配置，以满足开发需求。同时，AVD提供用户友好的人机交互图形界面来帮助开发人员轻松地创建和管理AVD。

目前，AVD支持在多种IDE中安装，常见的安装的方式有如下两种：

1. Ecplise + Android Plugin，即选择Ecplise作为开发环境，通过安装Android插件和依赖包完成Android开发环境的配置。
2. Android Studio，直接选择Android专用开发环境。

对于选择何种安装方式，是开发者个人的开发习惯。在Android Studio流行前，Ecplise + Android Plugin基本是安卓开发从业人员的首选，但是在多年的使用过程中，IDE外加插件的方式逐渐暴露出其无法解决的弊病，插件下载安装缓慢、IDE容易崩溃、不齐全的开发工具、无版本更新提示、不友好的用户界面等等，阻碍开发进度。为此，Google出于各种目的，推出自家的Android开发环境——Android Studio，其具有以下强大优势：

* Ecplise中所有功能；
* 基于Gradle的构建支持；
* 直接支持应用签名功能；
* Android特定重构和快速修复；
* 内置Google云、svn、git工具支持；
* 自带布局编辑器，允许用户拖放组件并在多屏幕配置上预览布局；
* 提示工具更好地对程序性能、可用性、版本兼容和其他问题进行控制捕捉；
* 支持NDK等框架。

除此之外，Android Studio的定期更新使得开发者能够第一时间使用到最新特性（目前最新版本为Android Studio 3），大大提高应用开发效率，并一步步取代Ecplise固有的市场地位。

综上，组内选择Android Studio作为实验的开发环境，并介绍在Windows和Mac中如何安装配置。此外，对于Android开发而言，另一个必备工具是adb，即安卓开发调试器，实验中也会对其进行安装配置介绍。

## 2.1 Windows

[Android Studio + adb] 图文并茂，尽量减少使用我/我们等第一人称的使用，尽量减少生活化的语言，允许吹逼。

## 2.2 Mac

[Android Studio + adb] 图文并茂，尽量减少使用我/我们等第一人称的使用，尽量减少生活化的语言，允许吹逼。

# **三、Android安全机制**

在移动设备持有量持续增长的时代，用户对隐私保护的需求愈来愈强烈，Android作为移动设备市场中市场份额最大的移动端操作系统，其不同于IOS封闭的开源性，一方面为它带来了IOS所不能具备的社区活力，但另一方面也为它带来了潜在的安全风险，熟悉Android其底层所具有的安全机制能够使我们更好了解手中的设备，更加清楚如何去保护自己的信息，以防泄露。本章将从Android的安全机制开始，讲解Android的基本安全机制，同时对其版本迭代过程中新加或删减的安全特性做简要的介绍，而实验的核心在于Android中的权限管理机制，一个让用户、开发者既爱又恨的安全机制。

## 3.1安全机制简述

Android提供了一个适用于移动设备的开放源代码平台和应用环境，其软件栈由七层组成，如图3-1所示。自顶向下的各层软件栈，亦称组件，都假定其底层实现的各栈均已采取适当的安全措施。值得注意的是除了作为Root代码运行的少量Android操作系统代码外，Linux内核上方的所有代码都受到应用沙盒的限制。

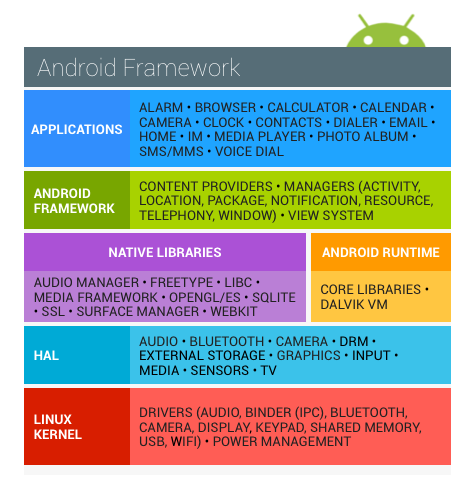


图3-1 Android七层软件栈

在七层软件栈中，其主要构造有以三块：

* HAL：设备硬件，Android能够在多种硬件设备配置中运行，其中包括智能手机、平板电脑、手表、汽车等。Android独立于处理器，但它确实利用了一些针对硬件的安全功能，例如ARM eXecute-Never。
* Linux Kernel：Android核心操作系统是在Linux内核之上构建的。所有设备资源（例如摄像头、蓝牙等）都需要通过操作系统的访问控制才能使用。
* Android Runtime：Android应用通常都是使用Java进行编写，并在ART中运行。不过，仍有许多应用（包裹核心Android服务和应用）实在本机应用或包含本机库。ART和本机应用在相同的安全环境中运行。应用在文件系统中有一个专用部分，他们可以在其中写入私密数据，包括数据库和原始文件。

除此之外，Application层对Android核心操作系统进行了扩展，其应用有两个主要来源，一是预安装应用，由开放源代码Android平台或设备制造商开发的一套预先安装的应用，二是用户自主安装应用，由任何第三方应用开放式开发环境提供。

针对以上的七层软件栈，Android在各层或多层中建立其成熟的多层安全模型，如表3-1所示，其中Linux Kernel层实现的是系统和内核安全性，指依赖于底层Linux所提供的相关特性，自底部保证上层的安全，Android运行时和本地库层实现的是实现安全性，通常指在开发流程、产品实现中常用的为了保障安全的实施方案，Android应用层实现的是应用安全性，指应用使用过程的安全保障。

表3-1 安全机制表

|  |  |
| --- | --- |
| 安全模型组成部分 | 主要安全机制 |
| Linux kernel | 基于POSIX的应用沙盒  基于DAC实现的文件访问控制  基于SElinux实现的强制访问控制  用于文件系统、私有密钥的加密功能  验证启动（类TPM） |
| Android Runtime & Native Libraries | 内存管理单元（MMU）  基于编程语言的强制类型安全  移动设备安全 |
| Android Application | 权限机制  应用签名机制  组件封装  Dalvik虚拟机 |

实验的重点在于上表中的应用层的权限机制的管理，因此，上表中的绝大部分内容将不会做介绍，若是想了解相关的概念，请参考附录中的网址。

## 3.2 安全机制迭代更新过程

了解Android版本更新过程中安全特性的增强，有助于了解设备安全的发展趋势，下图展示了历代版本更新中安全特性的增强项。可以看到，（补充一些特点）

## 3.3 权限管理——核心机制

# **四、权限管理**

## 4.1 权限管理介绍

## 4.2 权限管理版本迭代

### 4.3 权限管理现有问题

### 4.4 权限管理改进建议

# **五、实验总结**