编号

****

**毕业设计（论文）**

**（ 2017 届）**

**题 目**  基于自动测试的安卓应用程序行为抽取

**学生姓名**  李妍娜

**学 号** 1308010302

**专业班级** 软件工程1303班

**指导教师** 刘晓建

**所在学院** 计算机科学与技术

**日 期** 2017 /6/8

# 

# 摘 要

从Android系统产生至今，已经占领了移动设备操作系统约75%的市场，但同时由于Android系统巨大的用户量和开源的特性，Android系统的安全性受到了极大威胁。数字营销网站MarketingLand报道，在过去的12个月里Google Play应用程序下载量已经上升到500亿次，大量的应用在丰富用户体验的同时也带来了日益凸显的安全问题。因此研究基于Android操作系统的安全机制对于防御恶意软件攻击有着重要的作用。

大部分恶意软件实现某些功能时需要调用特定API来完成，通常通过静态分析和动态分析这两种方式来对恶意apk进行检测，动态分析则是指通过运行该软件并且跟踪和监控其运行时的行为来分析程序特征，以便更准确地捕捉到软件的恶意行为。在Android平台中，为了抽取应用程序运行时的行为特征，获取软件运行时调用敏感API的信息，本文设计并完成了对apk反编译后的smali文件中的敏感API注入日志的系统，可以自动地在apk中注入日志代码，在软件运行的时候记录该软件调用的敏感API信息。本文的工作内容如下：

①设计和实现了对smali文件中植入监控代码的算法，并通过若干实例验证了算法的正确性和有效性；

②用Monkey自动化测试方法对软件进行了自动运行和测试，避免了手工测试的费时费力；

③开发了基于Java swing的自动化工具，实现了对apk的反编译、注入代码、重新打包和签名、并用Monkey进行自动运行和测试这一完整过程的封装。通过若干实例测试验证了本工具的正确性。

**关键词**：敏感API调用；动态分析；Android；smali

# 

# 

# **ABSTRACT**

From the Android system has been created, has occupied the mobile device operating system about 75% of the market, but at the same time due to the huge amount of Android system and open source features, Android system security has been a great threat. Digital marketing website MarketingLand reports that Google Play app downloads have risen to 50 billion times over the past 12 months, and a large number of applications have experienced a growing user experience and a growing security issue. So the study based on the Android operating system security mechanism for the defense of malicious software attacks have an important role.

Most malware to achieve certain functions need to call a specific API to complete, usually through static analysis and dynamic analysis of these two ways to detect malicious apk, dynamic analysis is through the operation of the software and track and monitor its run time Behavior to analyze program features in order to more accurately capture the malicious behavior of the software. In the Android platform, in order to extract the behavior of the application runtime behavior, access to the software runtime call sensitive API information, this paper designed and completed the anti-compiled ap51 ap51 file in the sensitive API injection log system, In the apk into the log code, the software is running when the software records the sensitive API information. The work of this paper is as follows:

①The algorithm of implanting the monitoring code in smali file is designed and implemented, and the correctness and validity of the algorithm are verified by several examples.

②with the Monkey automated test method of the software for automatic operation and testing, to avoid the manual test time and effort;

③Developed a Java swing-based automation tool that implements the decompilation, injection code, repackage, and signature of apk, and automates and tests this complete process with Monkey. The correctness of this tool is verified by several examples.

Keywords: sensitive API call; dynamic analysis; Android; smali

# 目 录

**[1 绪论](#_Toc26196)** [1](#_Toc26196)

[1.1 研究背景及意义 1](#_Toc32758)

[1.2 研究现状 3](#_Toc5219)

[1.3 论文结构 4](#_Toc9490)

**[2 Android操作系统介绍](#_Toc29526)** [5](#_Toc29526)

[2.1 应用程序包结构 5](#_Toc2310)

[2.2 Dalvik虚拟机与Java虚拟机的比较 6](#_Toc564)

[2.3 Android应用程序四大组件 7](#_Toc29918)

[2.4 Android系统安全机制 9](#_Toc7025)

[2.5 Android系统安全机制的缺陷 11](#_Toc8305)

**[3 技术背景](#_Toc2551)** [12](#_Toc2551)

[3.1 反编译工具Apktool 12](#_Toc31684)

[3.2 smali语法介绍 13](#_Toc4974)

[3.3 apk签名机制 14](#_Toc2508)

[3.4 Monkey随机测试 15](#_Toc31415)

[3.5 批处理 16](#_Toc18046)

[3.6 ADB调试桥 16](#_Toc12801)

**[4 基于自动测试的应用程序行为抽取系统的设计实现](#_Toc4162)** [18](#_Toc4162)

[4.1 系统总体框架 18](#_Toc27670)

[4.2 反编译模块 20](#_Toc211)

[4.3 日志代码植入模块 22](#_Toc16799)

[4.4 回编译模块 24](#_Toc23334)

[4.5 签名模块 26](#_Toc12285)

[4.6 安装、运行和显示敏感API调用模块 27](#_Toc28064)

[4.7 测试与数据统计 32](#_Toc24305)

**[5 结束语](#_Toc12357)** [37](#_Toc12357)

**[致谢](#_Toc8356)** [38](#_Toc8356)

**[参考文献](#_Toc26088)** [39](#_Toc26088)

**1 绪 论**

## 1.1 研究背景及意义

近几年，Android系统的发展非常迅速，在全球智能手机市场的占有份额已经超过50%。由于Android系统具有开源性、易操作性、界面美观、允许多个应用程序同时运行等特点，所以很受用户喜欢。并且Android系统为程序开发人员提供了大量的功能性接口，方便了开发人员的工作，同时也使Android系统更易扩展和操作。但Android系统的开放性及其便利性也给用户带来了安全隐患，滋生了大量恶意软件，使得Android系统受到极大威胁。

截止2016年，中国智能手机的用户规模已达到6.31亿，相比2015年增长了4.9%[3]。图1-1为15-16年中国智能手机用户规模的柱状图。

#### 图1-1 15-16年中国智能手机用户规模

图1-2为2016年手机恶意软件感染用户系统分布图，从图可以看出，Android系统占了大部分，占比91.45%；其次是苹果IOS系统，占比4.33%。

#### 图1-2 手机恶意软件感染用户系统分布图

图1-3为按伪装应用维度统计的恶意程序分布图：色情播放类占比70%，伪装成系统程序占比15%，伪装成实用工具的占比3%，伪装成支付类的占比1%。

图1-4为按危害类型统计的恶意程序分布图：资费消耗最高占比44%，恶意扣费占比37%，隐私窃取占比13%，恶意广告占比5%。

图1-3 伪装应用饼状图 图1-4 危害类型饼状图

但是当前应用市场上的手机安全软件大多都只针对手机病毒、系统优化加速、短信拦截等进行控制，没有直观地反映手机中有哪些应用程序涉及到了敏感权限访问，也没有告诉用户哪些应用程序可能会泄漏个人隐私或具有泄漏个人隐私的意图。而这一问题可通过动态分析方法得以解决，因此对Andriod应用程序调用敏感API的动态分析是一个非常重要的研究课题，具有实用价值和指导意义。

## 1.2 研究现状

国外学者对于Android恶意应用检测做过很多研究，例如Androidguard工具，它是基于签名原理的，用来检测恶意代码，但是这种方法存在不足，它不能对未知的恶意样本进行分析检测。David Kleidermacher[4]提出了ARMTtustZone技术来提升Android移动设备的安全，从系统底层检测系统运行行为，它像软件提供了两个运行区域，用户经常使用的普通软件放在普通区中运行，安全关键软件运行在安全区中，安全区里的软件有权访问普通区，而普通区的软件则不允许访问安全区。XManDroid[5]是防止权限提升攻击的安全框架，可以使由Android System service或Content Provider建立的（隐蔽）信道得到有效的检测。

国内很多学者和教授也对恶意应用的预防做了很多研究，路程等[6]人提出通过系统网络日志和系统广播监控恶意应用行为的方法，彭智俊等人提出利用soot工具来检测系统的恶意行为，腾讯实现了动静结合的MalDroidAnylyzer[7]分析沙盘，可以对行为数据进行提取生成分析报告。

对于恶意软件的研究大体可以分为静态分析和动态分析，静态分析[8]是指通过Android逆向编译工程得到程序的smali语言代码，通过对反编译出来的smali代码进行分析，从而在代码层次上实现对软件恶意行为的准确定位，但是由于很多恶意软件采用了加密、代码混淆等方式，很难对其进行逆向编译，因此静态分析无法对其进行检测；动态分析技术有两种方法：一种是动态调试技术，另一种是动态行为分析技术[9]。动态调试是指使用IDA等反编译工具得到软件的C语言伪代码，然后对程序的C语言伪代码加断点进行调试，深入理解代码呈现出来的算法核心和逻辑结构，动态行为分析是在运行应用程序时，借助Android的Logcat系统对Android程序运行时的行为进行记录，然后分析它的行为日志，从而进行研究。

动态行为分析时常采用三种方式获取行为信息：第一种是对敏感API进行HOOK，它的原理是在程序运行的过程中动态地修改目标函数地址的数据，然后使用jmp跳转至该函数的地址，等执行完成之后再恢复目标函数的内存数据，在代理函数中获取到API的调用信息；第二种是在smali代码中对敏感API注入日志的方法，它需要先获取apk的smali代码，然后遍历所有的smali代码，找到有敏感API调用的代码段进行日志代码的植入[10]，这种方法能够更加准确的定位到apk在运行时的各种行为信息；第三种是捕获进程提取关键的行为特征，对此，Xu等人[11]提出Aurasium，在应用程序中嵌入跟踪程序，截获应用程序的系统调用，Burguera等[12]人提出Crowdroid，在Android端使用追踪器strace采集行为数据，然后传到服务器进行分析。

## 1.3 论文结构

本文的主要内容是设计一个基于Monkey自动化测试的应用程序行为抽取系统，可以对APK运行中调用的敏感API进行输出，本文共有五章，结构如下：

第一章，绪论。本章介绍了研究背景、意义以及研究现状，主要描述了当前阶段Android市场的发展状况和面临的安全问题，要处理恶意APK对Android用户造成的威胁必不可少的步骤就是分析恶意APK调用的敏感API，在此基础上抑制恶意APK的增长。

第二章，Android操作系统的介绍。在本章中着重介绍了Android系统的架构，包括应用层、框架层、系统运行层、Linux内核层这四层各自的功能，然后介绍了Android的四大组件以及它们之间的通信和协作，还介绍了Android中的Dalvik虚拟机与Java虚拟机的不同，紧接着分析了Android安全机制及安全机制的不足之处。

第三章，技术背景。本章对这次课题涉及到的相关技术做了简明扼要的介绍，

包括Google的反编译工具apktool、反编译之后的smali语法、apk签名机制、monkey随机测试、批处理、adb调试桥等等。

第四章，系统的详细设计。本章对此系统的各模块的原理、设计思路、流程等进行了分析和介绍，并用实例对本系统的有效性进行了测试，同时统计了结果。

第五章，总结和展望。本章对毕设和论文的工作内容进行了总结同时对以后的工作进行了规划。

# **2 Android操作系统介绍**

## 2.1 应用程序包结构

一个Android程序由多个文件以及文件夹组成，这些文件分别用于不同的功能，常用文件和文件夹如图2-1所示。



#### **图2-1 Android程序的包结构**

·src：该目录下放置的是Android程序的所有包及Java代码，开发者可以在src目录下创建包，在包中创建Java源文件。

·gen：该目录是Android程序中的特殊目录，它是自动生成的，该目录有一个R.java文件，这个文件是用来定义Android程序中所有资源的索引，开发者在编写代码时可以通过这个索引对各种资源进行访问。

·assets：该目录中存放的是一些随程序一起打包的文件，通常是一些多媒体文件资源。当Android程序打包时会将其一起打包，安装时会解压到对应的assets目录下。

·bin：该目录主要包含了一些编译时自动产生的文件，其中包括一个当前项目的apk安装包。

·libs：如果项目中用到了第三方的Jar包，就要把这些jar包都放在libs目录下。

·res：该目录下放的是Android要用到的各种程序资源，如图片、布局、字符串等。图片放在drawable目录下，布局放在layout目录下，字符串放在values目录下。

·AndroidManifest.xml：该文件相当于清单文件，在程序中定义的Activity、Service、Receiver、Provider都要在这个文件里注册，另外应用程序的权限声明也在该文件中。

·proguard-project.txt：该文件是Android提供的混淆代码工具Proguard的配置文件，通过该文件可以混淆应用程序中的代码，防止应用程序被反编译出源码。

## 2.2 Dalvik虚拟机与Java虚拟机的比较

Dalvik虚拟机是由Google公司开发的用于执行Android程序的虚拟机，它可以简单高效地实现线程管理和进程隔离，还大大地提高了内存的使用效率。每一个Android的应用程序都有一个Dalvik虚拟机实例与其对应，程序的代码在虚拟机解析后方能执行。

但是Dalvik虚拟机不是根据Java虚拟机的规范实现的，两者有很多不同之处，而且也不兼容。如图2-3所示[14]。



#### 图2-2 Java虚拟机和Dalvik虚拟机

从图中可以看出，Java虚拟机和Dalvik虚拟机主要有两大区别：一是它们编译后的文件不同；二是它们基于的架构不同。具体如下：

①编译后的文件不同

Java虚拟机上运行的文件格式是.class文件，而在Dalvik虚拟机上运行的是文件格式是.dex文件。Java程序中的Java类会被翻译成为一个或多个字节码文件（.class）然后把它们打包到.jar文件，之后Java虚拟机会从相应的.jar文件和.class文件中获取相应的字节码。而Android程序虽然是用Java语言编写的，但是在翻译成.class文件后还会将所有的.class文件转换成一个.dex文件，然后Dalvik虚拟机会从其中读出数据和指令，最后的.odex文件是为了在运行过程中提高性能而对.dex文件进行的优化，能够加快软件的加载速度和开启速度。

②基于的架构不同

Java虚拟机是基于栈结构的，栈是连续的内存空间，存入和取出的速度都比较慢；而Dalvik虚拟机则是基于寄存器架构的，寄存器是CPU上的一块缓存，它的存取速度比从内存中存取要快得多，这样就能根据硬件最大限度地优化设备，更适合移动设备的使用。

## 2.3 Android应用程序四大组件

Android应用由Activity、Broadcast Receiver、Content Provider、Service四大组件构[6]。它们的交互模型如图2-3所示[15]。

#### **图2-3 四大组件交互模型**

①Activity

Activity是Android应用程序中最基本的组件，每个应用程序中，可以有一个或多个Activity，但是必须有一个入口的Activity当程序打开时首先显示给用户。每一个activity被给予一个默认窗口，每一个activity作为一个独立的类被实现，并且继承了Activity这个基类。Activity类是由一些View视图组成的用户接口，并对事件作出响应。Activity的生命周期如图2-4所示。



#### **图2-4 Activity的生命周期**

②Broadcast Receiver

Broadcase Receiver翻译过来就是是广播消息接收器，它不负责执行某个任务。它其实相当于一个全局的监听器。其他组件是Broadcast Receiver的事件源，这一点和普通事件中的监听器很类似，但是普通监听器是监听组件中的对象，而Broadcast Receiver监听其他组件。

开发者创建自己的类继承Broadcast Receiver这个类，然后重写onReceive()方法，当其他组件发送广播消息时，如果IntentFilter中配置的消息与其相同，则onReceiver()方法被触发。Broadcast Receiver通过Context.registReceiver()方法注册，然后再在AndroidMenifest.xml中用<receiver>元素注册。

③Service

service没有用户界面，它同样是一个Android组件，它通常不用和用户进行交互，也没有图形用户界面，它会一直运行在后台，它具有自己的生命周期。每个service都扩展自类Serivce。Service组件监控组件的运行并且为其他应用提供后台的服务。

④Content Provider

Android应用运行在各自的进程中，它们相互之间是独立的，当应用程序之间需要进行数据交互时就需要用到ContentProvider,用户要实现自定义的Content Provider时需要实现insert(Uri,ContentValues)、delete(Uri,ContentValues)、update(Uri,ContentValues,String,String[])、query(Uri,String[],String,String[],String)这些抽象方法。

## 2.4 Android系统安全机制

在Android中，安全设计覆盖贯穿在系统架构的每个层面，包括Linux内核层、应用程序框架层、运行库层、应用程序层等。Android力求在开源开放的同时保障系统的安全，保护用户设备和程序以及数据信息等安全，Android的安全模型主要包括以下机制[16]：

·进程沙箱隔离机制  
 ·应用程序签名机制  
 ·权限声明机制  
 ·访问控制机制  
 ·进程通信机制  
 ·内存管理机制

①进程沙箱隔离机制，Android程序在其安装的时候被赋予了独特且固定的用户标识（UID），系统为每个程序分配一个沙箱，每个应用程序在其各自的Dalvik虚拟机中运行，拥有独立的运行空间和资源，互不干扰，默认情况下不能使用系统资源或其他应用的资源，而签名相同的两个应用程序之间可以通过在manifest文件中添加sharedUserId属性来共享数据，使得Android应用程序在安装时被赋予独特的用户标识（UID），并永久保持。应用程序及其运行的Dalvik虚拟机运行在独立的Linux进程空间，与其它应用程序完全隔离。如图2-5所示。



#### 图2-5 Android沙箱隔离机制

#### ②应用程序签名机制，Android规定了开发者必须对开发出的APK文件进行数字签名，通过签名来表示一个应用程序和它的开发者之间具有某种信任关系。系统在安装应用程序的时候，首先会检查该apk是否具有签名，如果没有将不允许安装，有签名才会安装成功。并且当该应用程序有更新的版本时，系统会检查其新版本的签名是否与之前的相同，如果相同则替换之前的版本，如果不同，则当成一个新的应用程序安装。如果Android开发者开发了几个具有相同名字的安装包，这时则可以通过签名来区分，如果签名不相同，则不会被替换掉，也防止了恶意的软件替换该应用程序。 ③权限声明机制，权限分为系统权限、Android ROOT权限、Android应用程序权限。对于应用程序权限，这个权限是由开发者在Manifest文件中声明的，然后安装的时候由用户授权。判断UID的值来确认属于哪种进程从而判断所具有的权限，如果UID为0，则为root权限，对权限不做控制，如果UID是system sever的uid，说明是system server也不做控制，否则的话通过PackageManagerService.checkUidPermission()在mGrantedUri-Permissions 这个表中查找并判断是否具有相应的权限。 ④访问控制机制，这一机制是为了防止系统文件和用户的数据安全被非法访问。Linux内核包含强访问控制机制和自主访问控制机制，但是由于某些原因谷歌并没有将强访问控制机制加入Android系统中去，文件访问权限控制实现了自主访问控制机制，文件的权限通过user、group、other与读、写、执行组合起来实现的，一般只有root和system server才有系统文件访问的权，其他应用程序若要访问文件必须要向系统申请权限。 ⑤内存管理机制，Linux的进程管理机制是在进程活动停止后结束进程，但是在Android平台上，则使尽可能把进程留在运存中，一直到内存不够才会根据“oom\_adj”的值决定结束哪个进程。Android将进程分为前台进程、可见进程、次要服务、后台进程、内容供应节点、空进程6种进程，其中前台进程、可见进程、次要进程不会被终止。内容供应节点在由于内存原因需要终止时具有较高的优先权。空进程是程序结束后在进程中存留的没有数据的空进程，系统的内存低于某个界限时会自动关闭空进程。

## 2.5 Android系统安全机制的缺陷

Android操作系统存在以下漏洞和问题：

①签名机制管理松散。为了提高Android系统的开放性，Google采用了数字自签名的方法，这种宽松的签名证书意味着不用数字证书认证机构授权，可以随意发布应用程序[17]，意味着恶意程序也能随意发布至网上，因此导致恶意程序泛滥。

②应用程序权限控制只针对单个应用程序。应用程序权限控制针对单个应用程序能有效地防止权限越级，但是多个应用在组合以后，能轻易提升权限，达到隐式权限提升[18]，使得恶意代码获得高级权限，导致信息泄露、系统受到破坏，这也是目前很多恶意代码获得运行的主要途径。

③缺乏对权限的动态运行时监测[19]。权限一经被授权给应用程序后，在应用程序的生命期间，它将不会被移除，即使声明此权限源程序被删除。

# 

# 3 技术背景

## 3.1 反编译工具Apktool

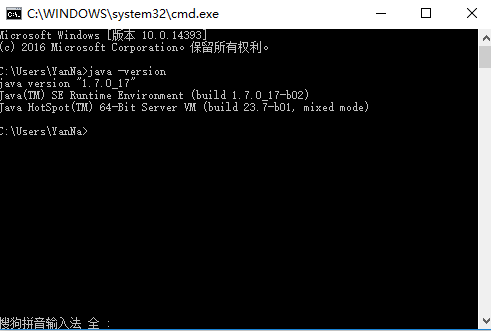
APKTool是GOOGLE提供的APK反编译工具，能够[反编译](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=2232234&ss_c=ssc.citiao.link" \t "http://baike.sogou.com/_blank)及回编译apk，同时安装反编译系统apk所需要的framework-res框架，清理上次反编译文件夹等功能。

①环境配置

·安装JAVA。

·配置JDK和JRE。

·测试；运行CMD（开始，运行，输入cmd，回车），输入java -version，回车，如出现图3-1所示的JDK版本，那就是已经安装成功了。



#### 图3-1 查看JDK版本

②命令

·decode

该命令用于进行[反编译](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=2232234&ss_c=ssc.citiao.link" \t "http://baike.sogou.com/_blank)apk文件，一般用法为apktool d D:\apkset\my.apk D:\apktoolresult，D:\apkset\my.apk代表了要反编译的apk文件的路径。D:\apktoolresult代表了反编译后的文件的存储位置。

如果指定的路径中文件夹已经存在，那么输入完该命令后会有提示信息，并且无法执行，可以修改命令加入-f指令apktool d –f这样就会强行覆盖已经存在的文件。

·build

该命令用于编译修改好的文件，一般用法为apktool b D:\apktoolresult，D:\apktoolresult代表修改后的文件路径，可以是之前反编译的文件存放路径，输入这行命令后，如果一切正常，D:\apktoolresult内会出现dist文件夹，里边存放已打包未签名的apk。

## 3.2 smali语法介绍

Smali是[Dalvik虚拟机指令语言](http://blog.csdn.net/xiaotian15/article/details/8530279)，在使用Apktool反编译apk文件成功后，会在输出目录下自动生成一个smali文件夹，其中存放着所有反编译出的smali文件，这些文件会根据程序包的层次结构生成相应的目录，程序中所有的类都会在相应的目录下生成独立的smali文件。Smali语言相当于一种中间语言，介于java源代码与dex可执行文件中间，由于dex可执行文件很难阅读，所以反编译生成smali语言，可以有助于更好的理解源代码。Smali的数据类型如表3-1所示。

#### 表3-1 smali数据类型

|  |  |  |
| --- | --- | --- |
| 基本数据  类型 | 语法 | 含义 |
| V | Void,无返回值类型 |
| Z | boolean |
| B | byte |
| S | short |
| C | char |
| I | int |
| J | long |
| F | float |
| D | double |
| L | Java类类型 |
| [ | 数组类型 |

续表3-1

|  |  |  |
| --- | --- | --- |
| 对象 | Lpackage/name/ObjectName; | 相当于java中的package.name.ObjectName。L表示这是对象类型，package/name该对象所在的包，;表示对象名称的结束 |
| 数组 | 单维整形数组用[I来表示,每增加一个维度就增加一个[ | [I表示int[],[[C表示char[][] |
| 方法 | 采用类似于  Lpackage\name\ObjectName;->MethodName(III)Z的形式 | 对象调用了boolean类型的MethodName(int,int,int) |
| 变量 | 采用类似于  Lpackage\name\ObjectName;->FiledName:Ljava\lang\String; | 表示对象类型为string类型，名称为FiledName |
| 寄存器 | V命名方式和P命名方式，P表示参数寄存器。V表示本地寄存器 | 指令.registers指定方法中寄存器的总数，指令.locals表明方法中非参数寄存器的数量，当一个方法被调用时，方法的K个参数被置于最后K个寄存器中，而非静态方法中的第一个参数总是调用该方法的对象 |

## 3.3 apk签名机制

所有Android应用在发布时都要被打包成.apk文件，APK文件在发布时都必须进行签名，这与在信息安全领域中所使用的数字证书的用途有所不同，它是应用包用来进行自我认证的一种机制。它不需要权威的数字证书签名机构的认证，而是由开发者自己来进行数字证书的使用和控制。Android系统利用数字签名来对应用程序的作者进行标识，并在应用程序间建立一种相互的信任关系，而不是用来判定应用程序是否应该被安装。

对Android应用签名可以采用调试模式和发布模式：使用Android开发工具(命令行和Eclipse等)开发的应用是用一个调试私有密钥自动签名的，这些应用被称为调试模式应用，这些应用只能用于开发者自行测试，不能够发布到Google Play官方应用市场上。当开发者需要在Google Play市场上将自己的应用程序发布时，就必须用安卓的签名机制使用其私有密钥签署应用，在应用安装时对其进行验证，用以实现对应用程序安装时的来源鉴定。

为了便于安装，安装时需要将应用程序打包成.apk文件。.apk 文件中包含了应用程序的.dex文件以及其它的非代码资源。Android要求所有的应用程序(代码和非代码资源)都进行数字签名，从而使应用程序的作者对该应用负责。只有在证书有效的情况下，且签名通过了公钥的验证，那么签名后的.apk 文件才是有效的。

## 3.4 Monkey随机测试

Monkey顾名思义就是像猴子一样任意对设备输入事件，它是由Java语言编写的并由“monkey”来启动执行，通过Monkey测试可以模拟用户滑动轨迹球、敲击键盘、触摸屏幕等动作对设备上的程序进行随机和压力测试。在使用Monkey测试之前必须搭建好SDK的环境和JDK1.5以上版本，然后在环境变量中把adbtools的目录添加到path里，就可以在命令行里对adb当前接入的Android设备进行测试了。

在PC机上打开CMD命令提示符，输入adb shell monkey+{命令参数}即可开始测试。常用命令参数如表3-2所示。

#### 表3-2 monkey命令常用参数

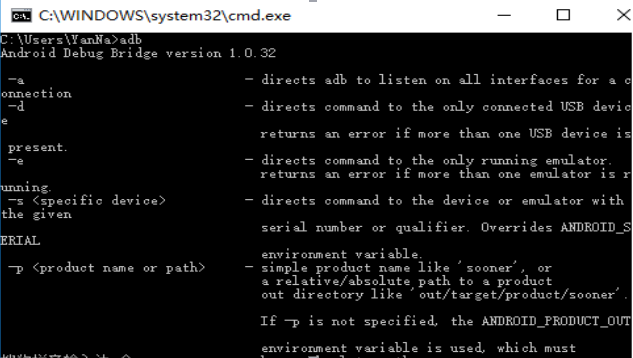
|  |  |  |
| --- | --- | --- |
| 基  础  参  数 | -v | 用于指定反馈信息级别，**level 0：-v，level 1：-v-v，level 2：-v-v-v** |
| -s | 随机数种子，指定伪随机生成器的seed值，若seed值相同则两次测试的事件序列也相同 |
| -throttle | 用于指定用户操作的时延，单位是毫秒 |
| -p | 指定要测试的一个或多个包名 |
| 发送的事件类型 | --pct-touch | 调整触摸事件的百分比 |
| --pct-motion | 调整动作事件的百分比 |
| --pct-trackball | 调整轨迹事件的百分比 |
| --pct-nav | 调整“基本”导航事件的百分比 |
| --pct-syskeys | 调整“系统”按键事件的百分比 |
| --pct-anyevent | 调整其它类型事件的百分比 |
| 调  试  选  项 | --ignore-crashes | 用于指定当应用程序崩溃时，Monkey是否停止运行 |
| --ignore-timeouts | 用于指定当应用程序发生ANR错误时，Monkey是否停止运行 |
| --ignnore-security-exceptions | 用于指定当应用程序发生许可（如证书许可，网络许可等）错误时，Monkey是否停止运行 |
| --kill-process-after-error | 用于指定发生错误时，停止运行程序 |
| --monitor-native-crashes | 用于指定是否监视并报告应用程序发生崩溃的本地代码 |

## 3.5 批处理

批处理是应用于DOS和Windows系统中的一种类似于Unix的[shell脚本](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=8961472&ss_c=ssc.citiao.link" \t "http://baike.sogou.com/_blank)的简化脚本语言，它通常是由COMMAND.COM或者CMD.EXE解释运行的。批处理文件的扩展名为.bat或者.cmd，批处理文件也叫[批处理程序](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=126951&ss_c=ssc.citiao.link" \t "http://baike.sogou.com/_blank)，是由一条条的DOS命令组成的普通文本文件，可以用DOS命令创建，可以用记事本直接编辑，也可以用EDIT.EXE来编辑。在“命令提示符”下输入bat文件的名称，也可以双击某个批处理文件，系统就会调用CMD.EXE运行这个批处理程序。系统要解析并运行某个批处理文件时，首先会扫描完整个批处理文件，如果没有遇见exit命令或者发生错误而意外退出，就会从第一行命令开始往下依次执行，一直到程序的结尾。

## 3.6 ADB调试桥

Android调试桥是指adb.exe工具（Android Debug Bridge，ADB），存在于SDK的platform-tools目录中，允许开发人员与模拟器或者连接的Android设备进行通信。为了使用该指令快速完成某项操作需要将adb.exe所在的目录配置到path环境变量中，之后就可以在命令行窗口中使用了，如图3-2所示。



#### 图3-2 adb命令

ADB的常见指令介绍如下：

·adb start-server：开启adb服务。

·adb devices：列出所有设备。

·adb logcat：查看日志。

·adb kill-server：关闭adb服务。

·adb shell：挂载到Linux的空间。

·adb -s<模拟器名称>install<应用程序（加扩展名）>：安装应用程序到指定模拟器。

# 

# 4 基于自动测试的应用程序行为抽取系统的设计实现

## 4.1 系统总体框架

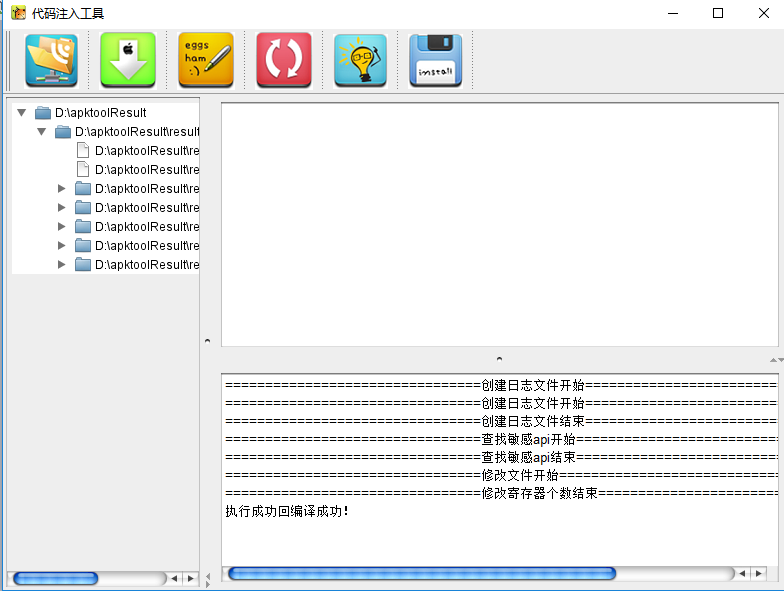
本系统的实现思路是：首先利用APKTOOL工具对需要分析的apk包进行反编译，得到其smali代码，然后对smali代码进行日志插桩，即遍历程序中所有的smali文件，定位到敏感API，在其有敏感API调用的代码的下一行注入smali语法格式的日志代码段，然后利用APKTOOL工具回编译，得到新的未签名的apk，未签名的apk不能运行在安卓设备上，所以还需要利用signAPK这个工具进行签名，签名完成之后在命令提示符中用命令的方式将签名之后的apk安装在安卓模拟器上，利用Android ADB自带的随机测试工具MONKEY进行测试，这时就可以得到需要分析的apk在运行过程中调用了哪些敏感API。具体过程如图4-1所示。



#### 图4-1 系统总体架构图

**图4-2 系统序列图**

本系统用Java语言开发，本系统的界面展示如图4-3所示，本系统将apk的日志插桩过程封装成了一个简易的小工具，用户只需要按步骤执行即可实现查看apk中调用的敏感API，工具栏按钮依次是打开文件、反编译、植入日志代码、回编译、签名、运行并显示数据结果。



#### 图4-3 系统界面图

本工具使用说明：本工具是基于Java swing设计开发的，用户在使用过程中必须严格按照顺序执行，即在本代码注入工具中从工具栏中最左边的打开文件按钮开始到最右边的安装运行按钮结束，执行过程中必须等上一步执行结束才能开始下一步，具体过程为：点击打开按钮选择感兴趣的apk->点击反编译按钮将该apk中的dex文件反编译为smali文件->点击植入代码按钮在smali文件中插入监听的日志代码->点击回编译按钮将植入后的文件打包成apk->点击签名按钮对通过回编译得到的apk进行签名->点击安装运行按钮将该apk安装到海马玩模拟器中然后运行该apk，将调用的敏感API的调用信息打印在命令行中。

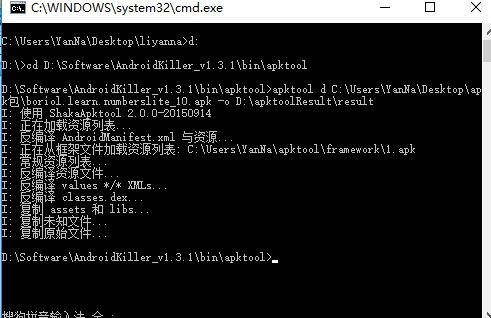
## 4.2 反编译模块

该模块将用户输入的apk包中的dex文件反编译成smali文件，此模块使用到了Google公司的apktool工具，在这里将命令封装成一个批处理文件，用Java中的方法获取用户输入的apk包的名称，以参数形式传递给批处理文件，此模算法块伪代码如表4-1所示。

#### 表4-1 反编译模块算法伪代码

|  |
| --- |
| Procedure backcompile( )  apkpath=要反编译的apk存放路径  file=反编译结果的存放路径  my.bat=apktool d apkpath -o file  exec(my.bat)  Output(反编译over!) |

如图4-4为反编译模块的批处理命令执行过程。



#### 图4-4 反编译命令

执行完批处理程序后会在指定的目录下生成一系列文件和文件夹，如图4-5所示。



#### 图4-5 反编译生成文件示意图

## 4.3 日志代码植入模块

该模块在反编译产生的smali文件夹中通过如图4-6的方法创建一个logutil文件夹，在此文件夹中生成Logutil文件，文件中定义了一个log()方法，将smali文件逐一遍历，定位敏感API，如图4-7为判断是否含有敏感API的方法，然后判断其所在的方法体的第一行.locals后面的int型数是否为0，若为0则给它加1，否则直接将敏感API的名字存入V0寄存器中，再插入Logutil类中的log()方法，并将V0中的数据作为实参传递给V()方法，这样就可以通过Android中的logcat进行查看，本模块算法伪码如表4-2所示，

原文件及插入代码后的文件如图4-8所示。

#### 表4-2 注入代码模块算法伪码

|  |
| --- |
| //创建日志类  Logpath=反编译的smali文件路径\\logutil\\Logutil.smali  p=敏感API  begin  Procedure createlogfile( logpath)  If logpath not exist  Call mkdir() //创建父目录  Call createNewFile() //创建logpath文件  Else call write(log()) //调用File类中的write()方法  Procedure smali格式语言log(p) |

续表4-2

|  |
| --- |
| Call android.util.log.d(“liyanna”,p)  End if  //定位敏感API  for temp=smali文件第一行to最后一行  line++  If (isHasSAPI(temp)=true)  Then if (.locals 0)  Then .locals 0←.locals 1  End if  将line,敏感API存入集合list  End if  //重写smali文件  Procedure writeContent()  Sort(line)//将集合中的line按从大到小排序  ReadLine()  If (line=集合中的line)  Then write(log(API)) //在对应的行写入log方法，将API字符串作为参数传递给log  End if |

对atticlab.bodyscanner.apk进行注入，其中WakefulBroadcastReceiver.smali文件中的startWakefulService()方法注入日志代码前后对比表4-3和4-4所示。

**表4-3 注入代码前的startWakefulService()方法**

|  |
| --- |
| .method public static  (Landroid/content/Context;Landroid/content/Intent;)Landroid/content/ComponentName;  .locals 8  .param p0, "context" # Landroid/content/Context;  .param p1, "intent" # Landroid/content/Intent;  .prologue  .line 81  sget-object v5, Landroid/support/v4/content/WakefulBroadcastReceiver;->mActiveWakeLocks:Landroid/util/SparseArray;  monitor-enter v5 |

续表4-3

|  |
| --- |
| .line 82  :try\_start\_0  sget v1, Landroid/support/v4/content/WakefulBroadcastReceiver;->mNextId:I  .end method |

**表4-4 注入代码后的startWakefulService()方法**

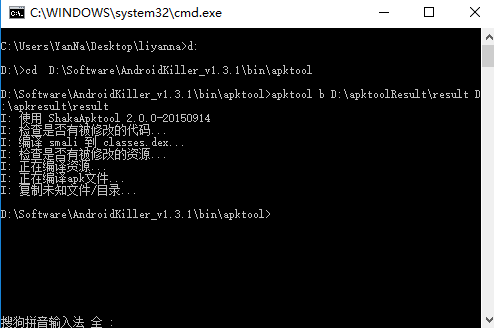
|  |
| --- |
| .method public static(Landroid/content/Context;Landroid/content/Intent;)Landroid/content/ComponentName;  .locals 8  **const-string v0,"Landroid/content/Context;**  **->getSystemService(Ljava/lang/String;)Ljava/lang/Object;"**  **invoke-static {v0},Llogutil/LogUtil;->log(Ljava/lang/String;)V**  .param p0, "context" # Landroid/content/Context;  .param p1, "intent" # Landroid/content/Intent;  .prologue  .line 81  sget-object v5, Landroid/support/v4/content/WakefulBroadcastReceiver;->mActiveWakeLocks:Landroid/util/SparseArray;  monitor-enter v5  .line 82  :try\_start\_0  sget v1, Landroid/support/v4/content/WakefulBroadcastReceiver;->mNextId:I  .end method |

## 4.4 回编译模块

该模块将已经注入了日志代码的smali文件回编译成dex文件并重新打包成APK，但值得注意的是此时的APk未经签名，不被允许安装在Android平台。利用GOOGLE公司的apktool工具进行回编译的执行过程如图4-6所示，本模块算法伪代码如表4-5所示。

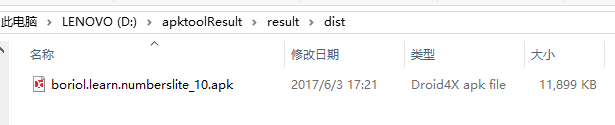
**表4-5 回编译模块算法伪码**

|  |
| --- |
| Procedure recompile( )  file=上一模块反编译结果的存放路径  outfile=回编译结果要保存的路径  huibian.bat=apktool b file outfile  exec(huibian.bat)  Output(回编译 over!) |



#### 图4-6 回编译执行过程

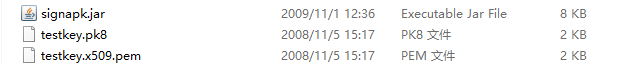
回编译之后会在dist目录下生成一个未签名的apk包，如图4-7所示。



#### 图4-7 未签名apk生成目录

## 4.5 签名模块

该模块对回编译模块得到的未签名的apk进行签名，使用Android平台自带的signAPK.jar工具，并使用Openssl来创建私钥/公钥对，如图4-8所示，然后用命令java -jar signapk.jar certificate.pem key.pk8 your-app.apk your-signed-app.apk即可进行签名，本模块算法伪码如表4-6所示。

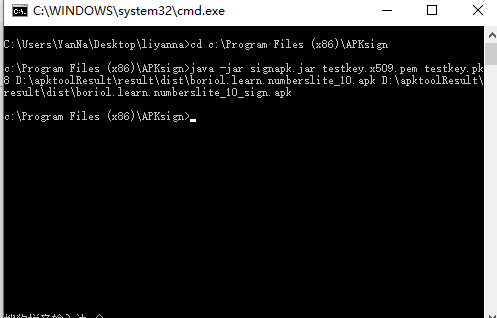


#### 图4-8 签名工具和秘钥

**表4-6 签名模块算法伪码**

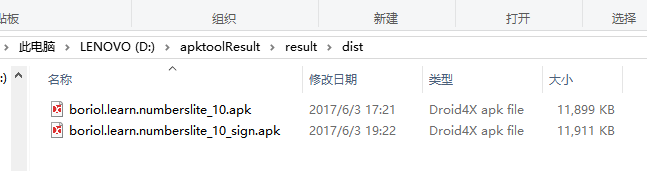
|  |
| --- |
| Procedure signAPK( )  dist=回编译的apk存放路径  signapkname=未签名的apkname+"\_sign.apk"  qmapkpath=签名后的apk存放路径+"\\"+signapkname  qianming.bat=java -jar signapk.jar testkey.x509.pem testkey.pk8 dist qmapkpath  exec(qianming.bat)  Output(签名 over!) |

给APK签名的执行过程如图4-9所示。



#### 图4-9 签名的执行过程

在dist目录下生成签名后的APK，如图4-10所示。



#### 图4-10 生成签名APK文件

## 4.6 安装、运行和显示敏感API调用模块

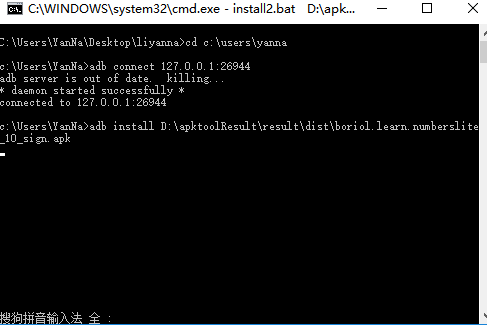
将签名后的APK安装在海马玩模拟器上，然后将该APK的包名传递给monkey，用adb shell monkey+包名即可启动该APK进行随机测试。当软件运行后会在Android的日志系统里打印出当前软件调用的敏感API信息。

本模块处理流程图如图4-11所示。



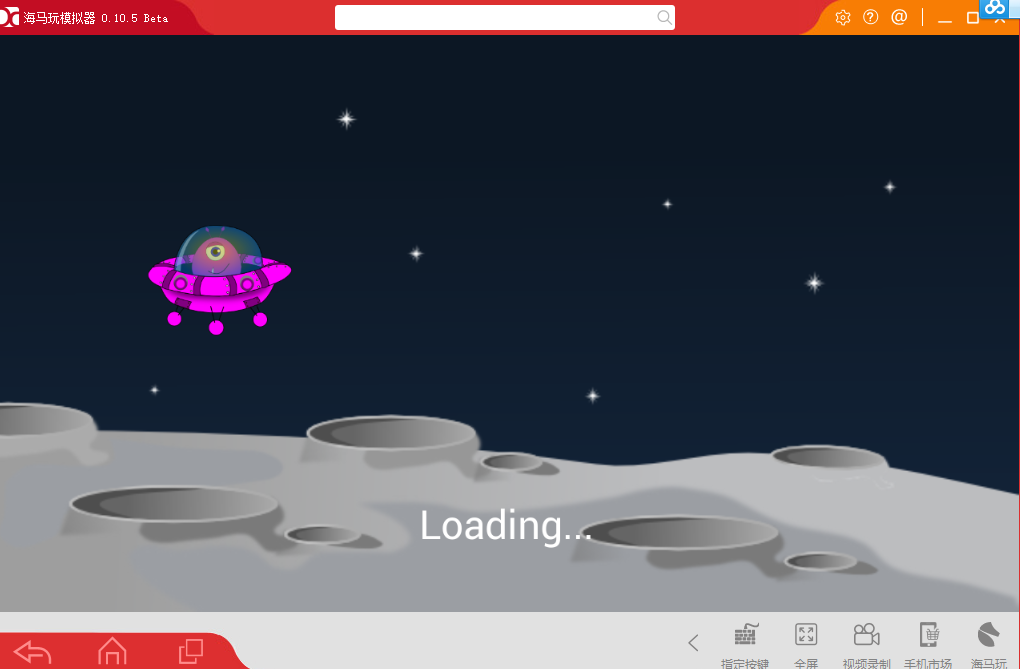
#### 图4-11 安装运行模块流程图

用adb install命令将APK安装到海马玩安卓模拟器中，如图4-12为植入代码之后的numberslite.apk的安装过程。



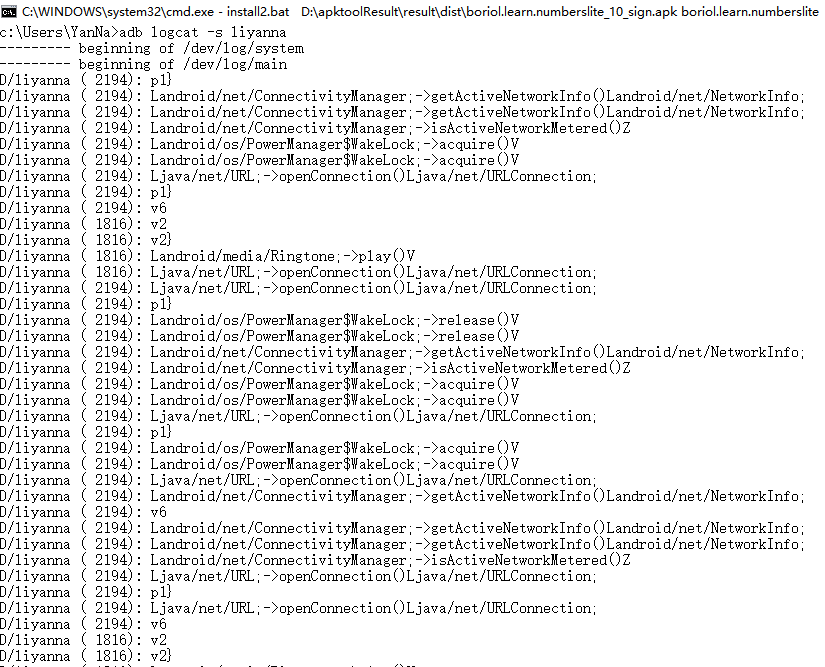
#### 图4-12 连接Android模拟器并安装apk

用monkey命令在海马玩模拟器中启动numberslite.apk,并输入了1000个随机事件，如图4-13所示。



#### 图4-13 海马玩模拟器中启动apk

该APK调用的敏感API打印在命令行中，如图4-14、4-15、4-16、4-17所示。



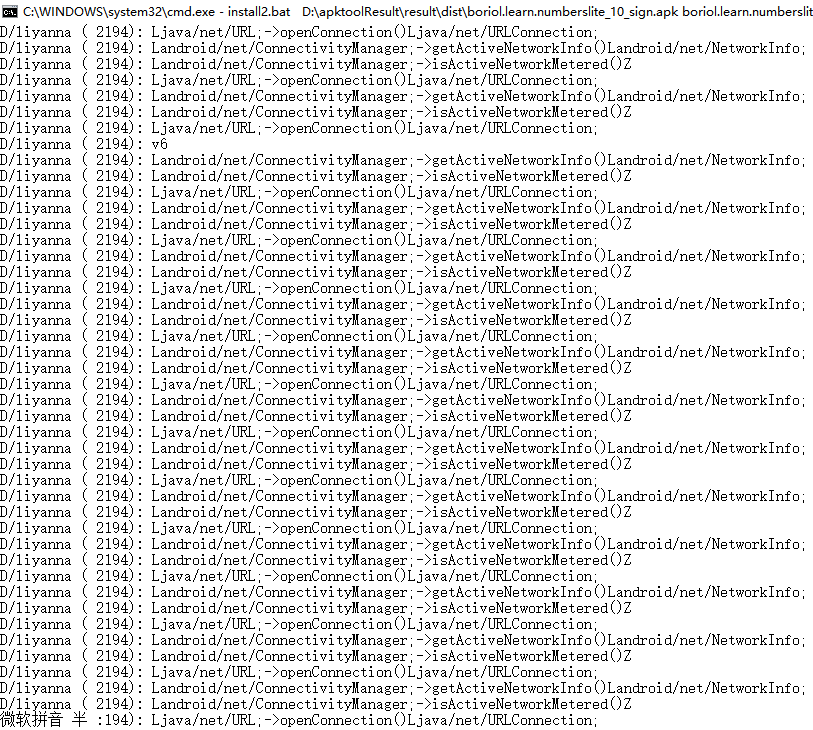
#### 图4-14 敏感API调用结果



#### 图4-15 敏感API调用结果



#### 图4-16 敏感API调用结果



#### 图4-17 敏感API调用结果

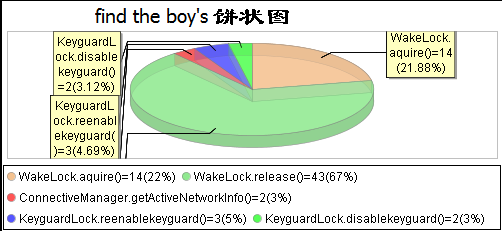
## 4.7 测试与数据统计

通过对8个APK进行了测试，在海马玩上运行这些APK如图4-18所示，将每个APK中调用的敏感API进行统计，在每个饼状图中绘制出每个API所占比例。

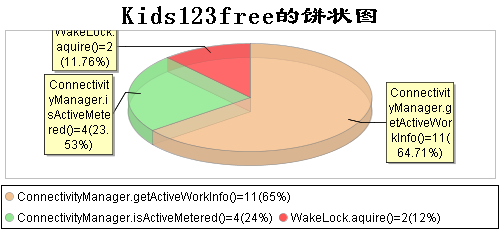


#### 图4-18 海马玩模拟器中安装的已经插桩的apk

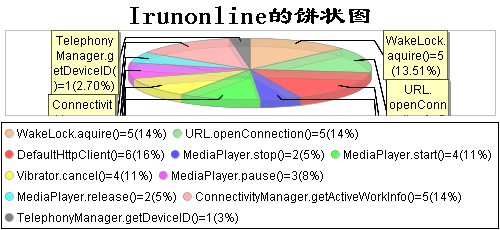
下面为敏感API调用的统计结果，绘制成饼状图如图4-19至4-26下。



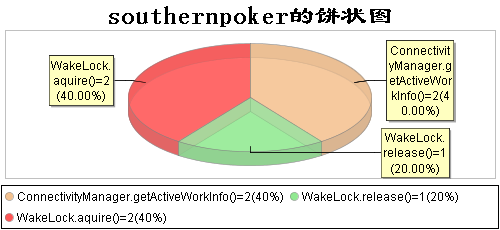
#### 图4-19 findboy’apk中敏感API统计图



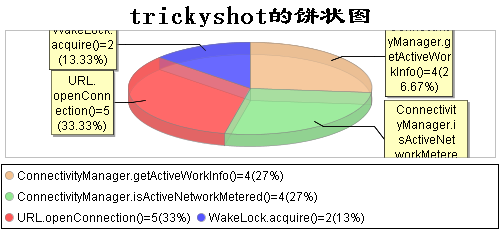
#### 图4-20 kids123free apk中敏感API统计图



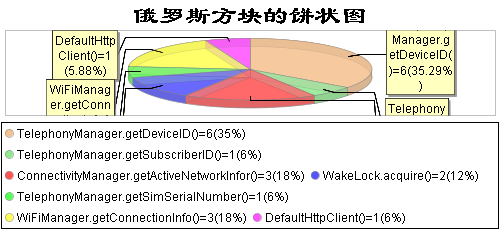
#### 图4-21 Irunonline apk中敏感API统计图



#### 图4-22 southernpoker apk中敏感API统计图



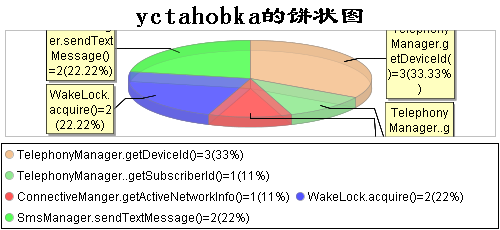
#### 图4-23 trickyshot apk中敏感API统计图



#### 图4-24 俄罗斯方块apk中敏感API统计图

#### 消灭星星饼状图

#### 图4-25 消灭星星apk中敏感API统计图



#### 图4-26 yctahobka apk中敏感API统计图

## 

# 5 结束语

本文首先介绍了Android平台的发展状况以及在发展过程中面临的安全威胁，引用网络资源的数据和图标对恶意APK的种类进行了分析，并对当前国际的研究现状进行了调研，由于恶意APK实现某些功能时往往调用了敏感API，由此引出本课题的研究意义。在进行本课题开始时首先对Android系统的架构和apk包的结构进行了详细的介绍，分析了Android中的Dalvik虚拟机的原理，然后分析了Android系统的安全机制和安全机制存在的不足之处。之后介绍了本次毕业设计中所用到的技术，包括apktool、smali语法、apk签名机制、monkey随机测试、adb调试桥等等。然后根据需求分析设计完成了该基于monkey自动测试的应用程序行为抽取系统，并对各个模块分别进行了详细的介绍，随后用八个实例进行了测试，统计了执行结果。本次毕设开发了基于Java swing的自动化工具，实现了对apk的反编译、注入代码、重新打包和签名、并用Monkey进行自动运行和测试这一完整过程的封装。经过通过实例测试验证了本系统的可行性和正确性。

# 致 谢

美好的大学生涯即将以这最后几个月的毕业设计而宣告结束，这是大学最后的作业。这四年的大学，因为学校，因为老师，我可以容易的学到教程要求的知识，因为学校给予我一个广阔自由的平台，老师的授课欢快又不失严肃，这种环境使我可以在学校安心的充实自己，丰富自己。同学校友间的融洽相处，互帮互助让我心里倍感到温暖。一个离家求学的人仿佛又找到了自己的新家如此的温暖。学会了如何去关心他人，帮助他人，正确的为人处事的方式积极乐观的生活态度。我还学会了学习一切重在自己的学习方法，这会让我在以后的工作中受益匪浅。

在这里感谢我的指导老师刘晓建老师一步一步耐心的指导和讲解，感谢我的同学雷倩在完成本次课题过程中给予我的帮助。同时，我也非常感谢我的室友和同学们，他们给了我很大的关心和支持。陪伴我走过了四年的大学生活。

最后，希望自己能够以这份毕设完美的结束自己的学生生涯，再用激情去迎接更加充满希望而美好的明天。

**参考文献**

[1] 杨云君. Android 的设计与实现[M]. 北京：机械工业出版社，2013:45-49.

[2] 李刚.疯狂 Android 讲义[M]. 北京：电子工业出版社，2013:25-42

[3] 张孝祥.Java就业培训教程[M]. 北京：清华大学出版社，2007

[4] Davi L,et al.Privilege escalation attacks on Android[C]//Information Security Conference-ISC,2010:346-360

[5] Bugiel S,et al.Towards Taming Privilege-Esealation Attacks on Android[C]

//Information Security Conference-ISC,2010:346-360

[6] 路程.Android平台恶意软件检测系统的设计与实现[D].北京:北京邮电大 学.2012,35-36

[7] risksk.Android沙盘原理与实现[OL].

http://security.tencent.com/index.php/blog/msg/7,2012.10.2

[8] 童振飞,杨庚.Android平台恶意软件的静态行为检测[J].江苏通信科学与实 践,2011(12),39-47

[9] 丰生强.Android软件安全与逆向工程[M].北京:人民邮电出版社,2013,2,156-157

[10] 吕晓庆，邹仕洪.基于smali的Android软件敏感API调用日志模块嵌入系统[D]. 北 京:北京邮电大学.2012,1-6

[11] XU R,SAIDI H,ANDERSON R.Aurasium:practical policy enforcement For

Android application[C]//The 21st USENIX Conference on Security Symposiun.

USENIX Association Berkeley.CA,USA,c2012:27

[12] BURGUERA I,ZURATUZA U,et al.Crowdroid:behavior-based malware detection system for Android[C]//The 1st ACM Workshop on security and privacy in Smartphones and Mobile Devices.New York,USA,c2011:15-26

[13] 柯元旦.Android内核剖析[M].北京:电子工业出版社,2012,20-23

[14] 传智播客高教产品研发部.Android移动应用基础教程[M].北京:中国铁道出版社，2014,4-5

[15] 杨丰盛.Android技术内幕系统卷[M].北京:机械工业出社,2011,56-62

[16] 狄婧.Android安全机制分析与解决方案解析[J].硅谷,2011,24:60-62

[17] 张中文,雷灵光,王跃武.Android Permission机制的实现与安全分析[C].第27次全国计算机安全学术交流会.2012.8:3-5

[18] 左玲,基于Android恶意软件检测系统的设计与实现.[D].成都:电子科技大 学,2012,13-18

[19] A.Shabtai.Malware Detection on Mobile Devices[C].Proceeding of the 11th International Conference Monbile Data Management.Kansas City,Mission,USA,

2010:289-290