

单位代码: 10293 密 级: 公 开

南京邮电大学
硕 士 学 位 论 文



论文题目 安卓系统恶意软件动态监测方法的研究

学 号 1015041206
姓 名 王倩文
导 师 沈苏彬 研究员
学 科 专 业 计算机应用技术
研 究 方 向 计算机网络
申请学位类别 工学硕士
论文提交日期 二零一八年六月

Research on the Method of Dynamic Monitoring the Malware on Android system

Thesis Submitted to Nanjing University of Posts and
Telecommunications for the Degree of
Master of Engineering



By

WANG Qian-wen

Supervisor: Prof. SHEN Su-bin

June 2018

南京邮电大学学位论文原创性声明

本人声明所呈交的学位论文是我个人在导师指导下进行的研究工作及取得的研究成果。尽我所知，除了文中特别加以标注和致谢的地方外，论文中不包含其他人已经发表或撰写过的研究成果，也不包含为获得南京邮电大学或其它教育机构的学位或证书而使用过的材料。与我一同工作的同志对本研究所做的任何贡献均已在论文中作了明确的说明并表示了谢意。

本人学位论文及涉及相关资料若有不实，愿意承担一切相关的法律责任。

研究生学号：_____ 研究生签名：_____ 日期：_____

南京邮电大学学位论文使用授权声明

本人承诺所呈交的学位论文不涉及任何国家秘密，本人及导师为本论文的涉密责任并列第一责任人。

本人授权南京邮电大学可以保留并向国家有关部门或机构送交论文的复印件和电子文档；允许论文被查阅和借阅；可以将学位论文的全部或部分内容编入有关数据库进行检索；可以采用影印、缩印或扫描等复制手段保存、汇编本学位论文。本文电子文档的内容和纸质论文的内容相一致。论文的公布（包括刊登）授权南京邮电大学研究生院办理。

非国家秘密类涉密学位论文在解密后适用本授权书。

研究生签名：_____ 导师签名：_____ 日期：_____

摘要

随着智能手机的普及，手机恶意软件的数量急速增加，尤其近几年，基于安卓系统的手机在智能手机市场占据主导地位，针对安卓系统的恶意软件数量呈快速上升趋势。手机恶意软件主要收集手机用户地理位置、通讯录、短信等个人隐私信息，严重威胁了手机用户的个人隐私。有效监测恶意软件并还原行为信息可以为后续清除恶意软件提供有力依据。

论文研究了安卓系统架构与安卓签名机制，分析了安卓系统的安全问题；分别研究了应用程序层监测机制、应用程序框架层监测机制、内核层监测机制的原理和优缺点；选定基于内核层的行为监控方法监控应用程序行为。着重研究了基于内核的层动态行为监控方法和应用程序安全评估策略。

设计了一个基于安卓系统的恶意软件的动态监测方案。通过分析应用执行过程的系统调用来重构出软件的动态行为。结合应用程序系统调用行为和参数信息，提出了基于权限列表的静态安全防御策略和基于上下文信息的动态安全评估策略。论文重点研究了动态安全评估策略，并针对权限泄露、合谋攻击、财务攻击、隐私数据窃取四种恶意软件不同攻击行为分别设计了详细的安全评估策略。

为减轻手机端负荷，与 PC 端结合，利用 MonkeyRunner 工具实现 apk 自动安装和卸载。为验证提出方案的可行性，论文选取 Malgenome Project 数据集中最具代表性的恶意软件家族在安卓模拟器上进行了仿真实验的验证和测试。实验结果表明，该方案能够有效地监测恶意软件行为，并向安卓手机用户发出警告。

关键词：安卓，恶意软件，系统调用，动态监测，安全评估策略

Abstract

With the popularity of smart phones, the number of mobile malware has rapidly increased, especially in recent years, mobile phones based on Android have dominated the smart phone market, and the number of malicious software for Android systems has been rising rapidly. Mobile phone malware mainly collects personal privacy information such as geographical location, contact, and SMS of mobile phone users, which seriously threatens the personal privacy of mobile phone users. Effective monitoring of malicious software and restoration of behavioral information can provide a strong basis for subsequent removal of malicious software.

This thesis studies the Android system architecture and Android signature mechanism, and analyzes the security issues of Android system, and studies principles and advantages and disadvantages of the application layer monitoring mechanism, the application framework layer monitoring mechanism, the kernel layer monitoring mechanism, this thesis uses the behavior monitoring method based on the kernel layer to monitor the behavior of the application. It focuses on studying dynamic behavior monitoring method based on the kernel layer and application security evaluation policy.

A dynamic monitoring approach for malware based on Android is designed. The dynamic behavior of the software is reconstructed by analyzing the system call of the application execution process. Combined with application system call behavior and parameter information, a static security defense policy based on permission list and a dynamic security evaluation policy based on context information are proposed. The thesis focuses on dynamic security evaluation policy and designs specific security evaluation strategies for the four different malicious behaviors of malicious software: rights leakage, collusion attack, financial attack, and privacy data stealing.

In order to reduce the mobile terminal load, the MonkeyRunner tool is adopted to automatically installing and uninstalling the APK combining with the PC terminal. Experiments which select the most representative malware class in Malgenome Project data set for validation and testing have been conducted on the AVD to verify the proposed scheme. The results show that the approach can effectively monitor malware behavior and warn Android mobile users.

Key words: Android, Malware, System Call, Dynamic Monitoring, safety assessment strategy

目录

第一章 绪论	1
1.1 研究背景与意义	1
1.2 国内外研究现状	2
1.3 论文主要工作	4
1.4 论文的组织结构	5
第二章 相关技术分析	6
2.1 安卓系统介绍	6
2.1.1 安卓系统架构	6
2.1.2 安卓基本组件	8
2.1.3 安卓安全机制	9
2.2 安卓系统安全问题分析	12
2.2.1 重打包技术	12
2.2.2 串谋权限攻击	14
2.2.3 四大组件安全	14
2.3 自动化测试技术	16
2.3.1 Monkey	16
2.3.2 Monkeyrunner	16
2.3.3 HierarchyViewer	17
2.4 安卓系统各安全架构层检测机制	17
2.4.1 应用程序层库替换	17
2.4.2 应用程序框架层挂钩 API	18
2.4.3 内核层修改系统调用表	19
2.5 本章小结	20
第三章 安卓恶意软件动态监测方法研究	21
3.1 基于内核的动态行为监控方法	21
3.1.1 系统调用截获功能	22
3.1.2 系统调用解析功能	23
3.2 应用程序安全评估策略	24
3.2.1 恶意应用程序分类	24
3.2.2 静态安全防御策略	25
3.2.3 动态安全评估策略	26
3.3 本章小结	28
第四章 安卓恶意软件动态监测方案设计与实现	29
4.1 安卓恶意软件动态监测方案设计目标	29
4.2 安卓恶意软件动态监测系统的框架	29
4.3 行为触发模块的设计与实现	30
4.4 行为监控模块的设计与实现	31
4.4.1 行为监控模块框架	31
4.4.2 行为监控模块的具体实现	32
4.5 行为分析模块的设计与实现	35
4.5.1 系统调用数据抽取与存储	35
4.5.2 静态安全防御策略的具体实现	36
4.5.3 动态安全评估策略的具体实现	37

4.6 本章小结 40

第五章 实验与结果分析 41

5.1 实验总体方案 41

5.2 实验环境配置 41

5.3 实验步骤 42

5.4 系统测试与结果分析 45

第六章 总结与展望 47

6.1 论文工作总结 47

6.2 进一步工作和展望 47

参考文献 49

附录 1 攻读硕士学位期间撰写的论文 52

附录 2 攻读硕士学位期间参加的科研项目 53

致谢 54

专用术语注释表

缩略词说明:

PC	personal computer	个人计算机
API	Application Programming Interface	应用编程接口
SDK	Software Development Kit	软件开发工具包
APK	AndroidPackage	安卓安装包
MD5	Message Digest Algorithm MD5	报文摘要算法
SVM	Support Vector Machine	支持向量机
UID	UserId	用户标识
GID	GroupId	用户组标识
ICC	Inter Component Communication	组件间通信
APP	Application	应用程序
ADB	Android Debug Bridge	调试桥
JNI	Java Native Interface	Java 本地接口
WAP	Wireless Application Protocol	无线应用协议
AV	Audio and Video	音频视频
IMEI	International Mobile Equipment Identity	国际移动设备身份码
IMSI	International Mobile Subscriber Identification Number	国际移动用户识别码
SMS	Short Message Service	短信服务
MMS	Multimedia Messaging Service	彩信服务

第一章 绪论

1.1 研究背景与意义

如今，国内移动互联网迅速发展，移动设备的数量不断提升，功能也愈发完善，移动设备的系统平台也逐渐火热起来。现阶段常见的移动设备系统平台有 iOS，安卓，Windows Phone。其中，根据市场调研机构 Gartner 公布的 2017 年第一季度智能手机市场报告显示，安卓的市场份额已经达到了 86.1%。安卓应用程序采用 Java 语言编写，是用户和设备的人机接口，为用户提供各种应用功能^[1]。安卓系统是由 Google 以及开放手机联盟共同开发的，安卓系统自身也具备了很多优势，展现了真正的开放，安卓系统包含了底层的操作系统以及应用程序交互界面的全部软件，这致使安卓系统的一些商户不单单可以不需要去支付任何的相关费用，另外能够按需修改以及扩展安卓系统。同时，安卓系统里包含的应用程序之间是平等的，应用程序之间可以任意沟通，这样一来在安卓系统下开发应用程序开发时能够很方便地共享所使用的数据，应用程序想要调用其他应用程序的功能只需要做出简洁地声明。安卓的这些诸多优势进一步解释它能够成为当下热门操作系统的重要原因。

现阶段，智能手机的用户数量越来越多，同时移动终端也出现大量恶意软件，特别是最近几年，基于安卓移动开发平台的手机在整个智能手机终端市场处于主要地位，这使得攻击安卓系统的恶意软件数量越来越庞大。

目前移动互联网发展飞速，出现了种类繁多的恶意代码，对手机用户造成了极大困扰，对互联网行业来说，急需对恶意代码的发展及传播进行严格管控，确保该行业能够稳定、健康地发展，以及确保手机用户的切身利益不受侵害。

ANVA（Anti Network-Virus Alliance of China，中国互联网协会反网络病毒联盟）发布了《移动互联网恶意代码描述规范》，其中将移动终端的恶意软件定义为：“在没有得到用户许可或未作出明确安装提示的情况下，私自在用户手机上安装和运行，对用户的合法权益造成极大侵害的软件”^[2]。

《移动互联网恶意代码描述规范》将恶意软件分为以下八类^[2]：

表 1.1 恶意软件分类

排序	属性分类	特征
1	恶意扣费	在用户毫不知情或未得到用户授权的情况下，通过骗取点击来暗自发送扣费短信或拨打电话等导致用户被扣费用。

2	隐私窃取	利用后台运行来盗取用户隐私信息，如短信内容等，将这些信息私自上传到远程服务器。
3	远程控制	在未得到用户授权或用户毫不知情的情况下，私自联通 C&C 服务器，并按照相关指令执行恶意行为。
4	恶意传播	利用无线网络技术将带着恶意链接的邮件、SMS/MMS 发送给其他手机用户，把恶意代码存储到用户手机上，最终恶意攻击其他移动终端。
5	资费消耗	在未提示用户的情况下私自连接网络，导致资费被扣等。
6	系统破坏	对移动网络和业务运行产生破坏或恶劣影响，导致用户的手机或其他应用程序无法正常使用。
7	诱骗欺诈	通过电话记录、邮箱等诸多方法来对手机用户进行欺骗，对其进行恶意攻击。
8	流氓行为	在用户不了解或者没有授权情况下，操作不能直接破坏系统，同时也不威胁手机用户敏感信息以及恶意扣费等一些恶意攻击行为。

其中一些主要的传播方式有互联网、WAP 或者应用中心等方式下载。也可以使用彩信存储卡来传播恶意代码。

伴随着移动互联网技术的快速发展，移动恶意软件也将进一步传播、泛滥，变得更具智能化。移动恶意软件结合了诸多社交媒体，将各种各样的手段融合在一起使用，这给广大手机用户带来了更大的威胁。因此，有效监测恶意软件并且还原应用程序行为信息可以有效地遏制恶意软件攻击泛滥现象。

1.2 国内外研究现状

相比 PC 端的安全防御，移动平台对于安全体系的研究时间较短，所以安卓系统在检测恶意代码方面还比较落后，随着智能手机不断普及，安卓应用程序愈发普遍化，各式各样的安卓系统恶意代码不断泛滥，更多的安全工作相关人员着手研究安卓系统恶意代码检测技术。

目前针对安卓恶意软件的监测方法主要包括静态检测和动态监测两种方式^[3]。静态检测方案^[4]是在不执行应用程序的情况下，采用逆向分析技术，通过抽取静态特征来判别应用程序是否包含恶意行为^[5]。静态检测方案尽管可以识别出应用程序的部分敏感行为，但是静态检测方案依赖于已知的恶意软件特征库，而特征库中仅仅存储的是已知的恶意软件中带有攻击性的一些特征码。所以，静态检测无法有效检测出未知的恶意攻击行为^[6]。

安卓应用程序的动态检测方案是在模拟器等安卓运行环境中实际运行应用程序，触发

其潜在的恶意行为，动态监测其运行过程，从而判断该应用程序是否包含恶意行为^[7]。相对而言，动态检测方案对安卓应用软件行为的监测效率更高。Xu^[8]等通过程序插桩的方式对应用程序的行为进行监测，并针对不同的攻击配置相应的安全策略。Enck^[9]等通过修改安卓系统的中间层代码，以添加标签的方式来监测数据流向，从而监测应用程序是否包含隐私窃取等敏感恶意行为。Burguera 等通过修改安卓系统内核层代码实现对应用程序的监测，采用 `strace` 获取安卓应用程序的系统调用信息，并对获取到的信息进行聚类分析，从而判断该应用程序是否存在恶意行为^[10]。

静态检测技术前提是不实际执行待测应用程序，采用浏览源程序的代码对应用程序的一些行为进行分析，例如对源代码或二进制程序进行分析。在分析二进制程序时，需要分析文件中有没有包含恶意的行为特征码，这是确定程序性质的重要方法，不过该手段只适用于已经面世的恶意代码，不能检测出未知的木马或病毒，因此必须对病毒库进行持续更新^[11]。对应用程序的源代码进行分析，主要采用对该应用程序进行反汇编的方式，读取其代码清单，然后将该清单和具有恶意行为的 API 调用序列放在一起进行比较，以此确定该程序是否属于恶意程序。通常来说，应用程序要想实施恶意行为，必须对特定的 API 进行调用，这一操作不会随着程序的改变而发生变化，从中可以看出，采用静态检车的方式，能够有效提高检测的准确性，而且省去了更新病毒库的麻烦。

在开展动态检测工作之前，必须先打开待测应用程序进行运行，以便检测工具实现对程序的运行情况和特点进行监控和记录，看看运行过程中有没有产生信息窃取等操作，并根据记录的行为来确定该应用程序是否存在恶意攻击的情况。这项技术的优势在于可以实现对未知恶意程序的区分，但由于缺乏遍历全部执行路径的能力，因此需要提高其准确性，同时由于无法实施拦截正在运行中的恶意程序，因此无法帮助用时及时止损^[12]。

Bose 等在塞班平台基础上设计了一个检测系统^[13]。该系统内包含大量的恶意信息和良性信息，在训练分类模型时选取了比较经典的支持向量机的方式，在对待测应用程序内的因果时序进行区别表示时，选择了 Temporal Logic of causal knowledge 的方法，此方法可以体现出数据之间的时序关联，同时其概念和拓扑结构相同。

JACOB 等^[14]对以特征行为为基础的检测方式进行了归纳，将有关的检测器分成了形式化的检测器和基于模拟的检测器两种，同时还从数据提取的解释机制和时机对上述两种检测器进行了更为细致的分类，并对不同训练模型的形成方案进行了分析。

Dai 等^[15]在 Windows Mobile 操作平台的基础上，设计出了以特征行为为基础的检测方案，这个方案的特征序列来源于动态捕获程序对 API 敏感序列的调用，输出后变为有穷状

态机的形式，最终借助模型校验的手段将该特征序列与恶意代码特征库进行比对。

Michael 等^[16]采用的是静态分析的方式，使用工具为反编译工具 baksmali，这种方式具有开源特性，能够有效开展数据流解析的工作，以便查看应用程序中是否存在敏感数据查询等越权操作，结果显示，权限机制中包含着某些漏洞，尽管一些恶意软件并不具备权限，却依然可以执行一些越权的操作。论文提出的应用程序安全评估策略，一方面在基于权限列表的安全防御策略中运用了权限机制，另一方面在行为分析过程中针对权限泄露给出了具体的安全策略，这样双重检查防止了权限机制中漏洞的出现，体现了方案的先进性。

1.3 论文主要工作

针对如何对安卓恶意软件进行有效监控这一问题，论文提出了一个基于安卓系统的恶意软件的动态监测方案。该方案通过计算文件 MD5 值判断应用程序是否存在重打包迹象，进而实现样本过滤。选定内核层监测方法替换系统调用表还原应用程序行为，并针对权限泄露、合谋攻击、财务攻击、隐私数据窃取四种恶意软件不同攻击行为给出相应安全策略。内核层行为监控是通过替换系统调用表的方法进行的，应用程序执行自定义函数，当正在执行的操作能够被该进程访问，则调用系统原有的函数；否则报错，并不执行原系统调用，实现对应用程序的行为控制。系统对运行在自定义内核情况下的应用程序的行为有控制功能。根据安全评估策略能够测试出恶意软件的攻击类型，达到对应用程序行为的监测功能。为减轻手机端负荷，与 PC 端结合，利用 MonkeyRunner 工具实现 apk 自动安装和卸载。实验表明，该方案能够有效地监测恶意软件行为，并向安卓手机用户发出警告。具体研究内容如下：

(1) 讨论了安卓恶意软件检测技术的研究背景和意义，分析了国内外学术界关于安卓恶意软件检测的发展现状。

(2) 研究了安卓系统架构及四大组件，研究了安卓安全机制中数字签名、沙箱机制以及权限控制机制，讨论了目前安卓系统存在的安全问题。研究了安卓系统各安全架构层监测机制。分析了应用程序层监测机制、应用程序框架层监测机制、内核层监测机制的原理和优缺点。

(3) 着重研究了基于内核的动态行为监控方法和应用程序安全评估方法。主要解决了基于内核的行为监控中内核的初始化问题以及如何使用 Binder 解析系统调用这一问题。针对应用程序安全评估，提出了基于权限列表的静态安全防御策略和基于上下文信息的动态安全评估策略。只有通过静态安全防御策略的应用程序才能继续进行动态安全评估。论文重

点研究了动态安全评估策略，并针对权限泄露、合谋攻击、财务攻击、隐私数据窃取这四种恶意攻击类型分别制定了详细的安全评估策略。

(4) 提出了一个基于安卓系统的恶意软件的动态监测方案，并通过实验验证方案的可行性。选取了 Malgenome Project 数据集最具代表性的恶意软件类族在安卓模拟器上进行实验验证和测试。

论文所提方案的创新性主要体现在两个方面。第一，目前针对基于权限机制的静态分析方法存在漏洞，仍然会出现某些不具备权限的恶意软件执行越权操作。论文所提方案在此基础上，进一步针对权限泄露设计了动态安全评估策略。保证了在应用程序运行前和应用程序运行中对权限列表进行双重检查，解决了权限机制漏洞问题。第二，在动态安全评估策略中不单单根据应用程序动态行为，还考虑了操作频率和前后台等上下文信息，提高了恶意软件的监测效率。

1.4 论文的组织结构

本篇论文总共有六章，具体组织结构如下：

第一章：绪论。介绍了课题的研究背景、国内外的研究现状，介绍了恶意软件攻击类型以及恶意软件检测方法。最后介绍了论文的主要工作和组织结构。

第二章：相关技术分析。介绍了安卓系统架构及四大组件，介绍了安卓安全机制中数字签名、沙箱机制以及权限控制机制，分析了安卓系统的安全问题，其中包括重打包、串谋权限攻击、四大组件安全。分析了应用程序层监测机制、应用程序框架层监测机制、内核层监测机制的原理和优缺点。

第三章：安卓恶意软件动态监测方法研究。研究了基于内核的行为监控方法和应用程序安全评估方法。主要研讨了基于内核的行为监控中内核的初始化问题以及如何使用 Binder 解析系统调用这一问题。提出了基于上下文信息的安全评估策略。

第四章：安卓恶意软件动态监测方案设计与实现。以前面的理论及技术为基础，对安卓恶意软件动态监测方案的总体设计进行详尽的阐述，主要包括总体设计、关键功能模块的设计及其交互，以及各功能模块的具体实现。

第五章：实验结果与分析。本章对所提出的安卓恶意软件监测方法的效果进行验证，同时对整个系统进行测试，分析最终的监测结果，验证论文所提出方案的适用性和有效性。

第六章：总结与展望。总结了论文相关工作，对进一步工作和展望做出了阐述。

第二章 相关技术分析

本章首先研究了安卓系统架构、安卓系统四大组件以及安卓安全机制中数字签名、沙箱机制以及权限控制机制，然后分析了安卓系统的安全问题，其中包括重打包、串谋权限攻击、四大组件安全。研究了自动化测试技术以及安卓系统各安全架构层监测机制。分析了应用程序层监测机制、应用程序框架层监测机制、内核层监测机制的原理和优缺点，着重分析了应用程序层库替换、应用程序框架层挂钩 API，内核层修改系统调用表三个行为监测方法。

2.1 安卓系统介绍

2.1.1 安卓系统架构

安卓的系统安全架构采用分层模式，如图 2.1 安卓系统架构分为四层，从低层到高层分别是内核层、系统运行库层、应用程序框架层、应用程序层^[17]。



图 2.1 安卓系统架构

(1) 应用程序层

系统以及核心应用程序包一同发布，其中包括主屏、浏览器和通讯录等程序，帮助用户实现人机交互。其中的每一项应用程序的编写语言都是 JAVA 语言，通过调用相应的 API 来完成。

(2) 应用程序框架层

应用程序框架层包含程序开发所需要的一系列类库，这使得程序开发者能够很方便的使用 API 框架。应用程序框架层的设计模式对组件的重用进行了简化，应用程序仅仅需要

遵守框架所要求的安全性限制，即能够发布自己定义的功能块，同时能够使用其他应用程序发布的功能块。活动管理器的功能是对程序的生命周期进行监管，还具备包括导航在内的基础功能；窗口管理器的作用是对开启中的窗口进行管理；内容提供器的作用是共享应用程序间的数据，并实现不同程序间的彼此访问；视觉系统管理的范围是所有应用程序中包含的基本组件。

（3）系统运行库层

系统运行时库被划分成两个部分：**Dalvik** 虚拟机与核心库。**Dalvik** 虚拟机设计目的是使手机这种低内存平台上能够适用安卓系统，每个程序都具有独立的 **Dalvik** 虚拟机；核心库包含了 **JAVA** 语言所提供的核心库的大部分功能，同时为应用程序开发者提供了安卓的核心 **API**。

（4）内核层

安卓系统一方面依赖于 **Linux** 内核，另一方面又不是完全依赖于 **Linux** 内核的。安卓系统一方面借助 **Linux** 内核来实现进程管理等核心的功能，另一方面增加了自身独有的功能，如低内存管理等。这样一来，内核就可以在智能手机终端的环境中顺利运行，有效提高了智能手机终端的相关性能。

2.1.2 安卓基本组件

安卓应用程序中包含了很多的应用程序组件，这些组件经过有机结合形成了整体。安卓系统对组件进行了分类和定义，任何种类的组件都具有指定的用途以及固定的生命周期，组件的创建以及组件毁灭的方法都是由生命周期决定的。以下分别介绍安卓系统的四种基本应用组件^[18]。

（1）活动

一个活动就可以指代一个用户交互界面。例如，一个微博客户端应用中包含了进行用户信息显示、用户微博新建和微博阅读在内的三个活动，不同活动之间彼此独立，互不干扰。因此，所有的应用程序都具备启动其他的应用程序的任意一个活动。例如，在微博客户端的应用程序中，照相机应用也可以启动，用来进行照片的拍摄，以使用户编辑微博。

（2）服务

服务是运行于后台的组件类别，主要去开展一些损耗时间操作以及一些进程任务。服务是不具有用户交互的接口的。比如当用户进行电子书阅读时，也可以通过服务来播放后台音乐。或者用户进行微博浏览时，可以借助服务来加载服务端的新内容，而不需要锁定

所呈现的微博内容的相关活动。其它的组件可以启动一个服务，并与之进行交互。

（3） 内容提供者

对于不同的应用程序而言，内容提供者具有数据共享的重要作用。在安卓平台上，SQLite 数据库、文件、web 等，都是应用程序数据的存放点，内容提供者应用程序可以把储存在各个位置的数据共享出来。比如，系统中包含一个对通讯录信息进行管理的内容提供者，此时，其他具有相应权限的程序都可以对该内容提供者进行访问，进一步读取并且修改联系人信息。

（4） 广播接收器

广播接收器组件能够对系统范围中的广播通知进行响应。系统可以进行大量的广播通知发布，例如对手机屏幕进行关闭，或手机电量不足等。应用程序也可以进行不同广播通知的发送，例如在下载应用程序时，数据下载结束，可以访问相应程序。广播接收器不包含交互接口，因此用户无法与程序进行交互，只能经过状态栏的通知来知晓事件已经发生。广播接收器处理的任务一般规模较小，比如打开服务处理某些事务，广播接收器以广播接收器的子类的方式完成，广播的传递形式为统一的 Intent 对象形式。

这四大组件中，活动和服务的类定义在 `android.app`^[19]中，另外两种在 `Android.content`^[20]中定义。

2.1.3 安卓安全机制

（1） 数字签名

在发布安卓应用之前，都是.apk 的形式存在的，开发人员需要数字签名才可以发布。安卓系统对数字签名进行了严格的规定，安装到系统中的所有应用程序都应该遵守此原则，私钥掌握在程序开发人员自己手中^[21]。数字证书的作用显而易见，具有非常高的安全性，可以标识出开发人员和程序间的一种信赖关系，只有完成数字签名的安装，并且验证无误，应用才能安装成功。安卓的数字签名主要有如下作用：第一，避免同名的其他软件对程序进行恶意替换，只要数字签名完成后，就可以明确开发人员的具体身份，开发人员独自掌握私钥，如此即便遇到同名的恶意软件，也无法进行替换；第二，数字签名可以促使软件包更加完整。开发人员记录下签名的具体数据信息，避免遭受反编译行为，避免其他用户对其重新打包；第三，针对程序升级有利。只有相同的签名，应用才可以更好的升级为最新版。在应用升级时，系统会对签名信息进行检测，只有签名信息和包名两者都相同的情况下，系统才会判断应用属于已知应用，对其进行升级，反之则不会升级；第四，对开发

应用程序以及模块化设计都具有一定帮助。在安卓系统中，相同进程可以准许若干应用程序一起使用，但这是有前提条件的，即这些应用程序必须具有相同的证书签名，并且系统将这些应用看作是彼此独立的若干个应用。此种情况下，可以将程序看作是模块来完成部署，系统能够升级其中任何一个模块；第五，对数据或者代码共享有利。安卓系统以签名为基础设定了一种权限机制，在此权限机制下，程序能够将自身功能公开给另外一个签名相同的应用。同样的，包含相同签名的几个应用程序是能够通过采用基于数字签名的权限机制这一方式实现代码的安全共享。

Apk 签名机制包括下述两个阶段：第一，包扫描阶段；第二，权限创建阶段。在第一个阶段要对包的数字签名证书以及完整性进行验证。系统包与普通包存在一定差异，后者检查的对象为 manifest 文件夹，是对其中子文件进行检查，检查的内容有 jar 包中的所有文件。然而前者检查的内容为 AndroidManifest.xml 配置文件，只要该文件签名信息完成，且提取了验证信息就可以了。

（2）沙箱机制

安卓系统是基于 Linux 内核的移动智能平台操作系统。Linux 操作系统可以多用户一同使用，其中有关的访问文件的具体权限，一起通过用户 ID(UID)以及用户组的 ID(GID)来一起控制。也就是说，用户 ID 与用户组的 ID 共同决定了 Linux 的安全机制。在安卓系统中 UID 与 GID 的实现存在差异，因为 Linux 属于多用户的一种操作系统，不同用户对应不同的 UID；然而安卓通常应用在移动操作系统上，一般情况下只有一个用户，因此此处 UID 指的不是使用操作系统的用户，而是所有应用都有属于自己的 UID，即所有应用，也就是说在 Linux 系统中所有进程都扮演一个用户的角色，从而达到隔离的效果^[22]。

在 Linux 当中，以 GID 与 UID 为基础的安全机制中包含了如下角色：第一，用户；第二，进程；第三，文件。所有用户都有与之对应的 ID，按照小组的方式对所有用户进行了划分，每个用户都有与之对应的 GID，且都属于一对一的关系。某个用户是能够同时归属于好几个用户组，换言之，每一个 UID 能够相应地得到一个 GID。任何一个用户所属用户组都不唯一，即 UID 与 GID 的关系为一对多的关系。用户所属用户组，均称作主用户组，其他的属于补充用户组。在 Linux 当中，所有文件都具有如下权限：第一，Write；第二，Execute；第三，Read。然后根据用户的不同属性进行了划分，与三种权限并且对应的组分别为如下三个：第一，Other；第二，Group；第三，Owner。

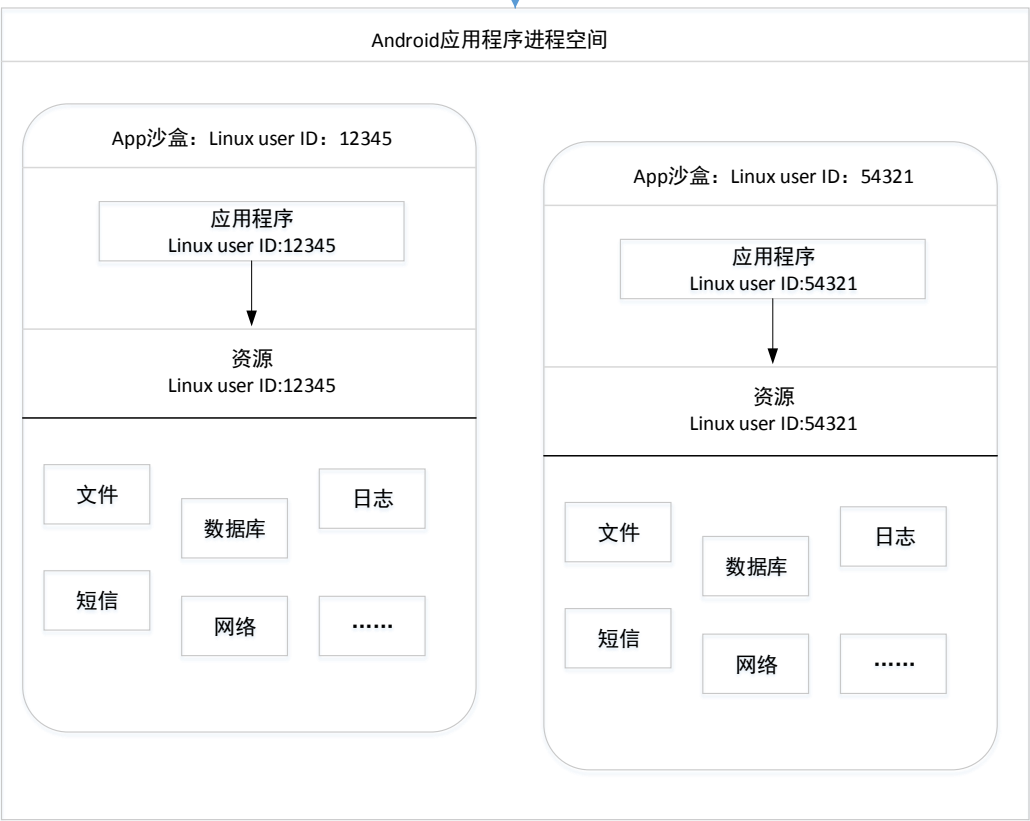


图 2.2 系统程序沙盒状态箱

因为安卓系统的操作基础为 Linux 操作系统内核，其安全访问控制策略同样适用于安卓系统。基于 Linux 文件的访问控制，创建出一种完善的沙箱机制，从而确保整个环节的安全可靠性，另外也具有一定的独立性。实现的过程具体为：在安卓系统中，所有应用中都设有 dalvik 虚拟机实例。也就是说所有应用彼此独立，所有应用程序都与系统用户相对应，即应用都具有与之对应的用户 ID，从而达到良好的隔离效果。具体见上图 2.2。

（3） 权限控制机制

针对系统权限的限制，Android 系统应用了权限机制，以避免非权限用户应用不相关数据信息、滥用查访权限以至于用户体验差，同时能够有效避免有目的的应用敏感权限盗取系统用户的私人信息等。Android 系统对用户的查访给出了一定的权限^[23]。若需对部分敏感权限进行查访时，必须在 Androidmanifest.xml 文件中进行自主申请，并且在程序运行阶段系统会自动检查其申请权限范围。程序安装至移动设备前，系统将对用户显示其申请的权限，若用户允许授权则可实现安装下载。程序下载完成后请求访问敏感权限时，安卓系统会再一次查询其请求的权限，如某一程序请求查访个人通讯录，则需在配置文件中写入以下代码：

```
<manifest xmlns: android="http://schemas.android.com/apk/res/android"
package="test.myTestApp">
<uses-permission android: name="android. permission. READ CONTACTS"/>
...
</manifest>
```

安卓系统在装载应用时同样需对应用程序所请求的相关权限进行检查，同时需将其反馈至用户。用户可自行选择接受或拒绝其所申请的权限，只要用户不同意访问权限申请，应用系统则不再装载；相反用户同意权限后则可继续安装，完成装载后用户使用程序时程序可直接应用权限而不需再次对用户发出权限申请。安卓系统安装 app 时，用户不可选择程序所申请的权限，仅有全部允许与放弃安装两个选项可供选择，虽能够对应用系统在安装时申请的权限进行随时查访，且同意操作是不可逆的，也无法动态授予。App 在权限未经得用户允许时，点击界面图标通常会显示异常窗口且自动关闭程序，用户能够由系统日志中对该类异常数据进行查访。

2.2 安卓系统安全问题分析

2.2.1 重打包技术

盗版行为即利用重打包技术对原版 APP 进行盗窃，属于十分严重的恶意行为。恶意编程人员能够由 Android app 商城随意下载应用系统压缩包文件，而后对其进行解压、篡改、重打包，然后更改名称上传至 Google Play 或者别的软件市场^[24]。该行为严重损害了用户权益，并严重威胁了其个人信息安全。另外，该行为严重损害了最初研发者的利益。

Wu zhou 将 22906 个非 Google Play 中的 APP 与 68187 个 Google Play 应用商店中的 APP 进行对比分析，结果表明非 Google Play 软件市场中属于盗版重打包程序的数量占其总数的 5%~13%^[25]。

关于此方面，Yajin Zhou 等研究人员进行了分析，最后提出 Android Malware Genome Project。对如何识别安卓系统恶意软件进行分享，以求研发出性能更优的检测恶意软件的工具。截止到目前，共采集了 1260 多个恶意应用程序样本，可将其划分成四十九个不同的种类，而大部分的恶意程序特征现已可以识别检测。在这 1260 个盗版程序中，绝大部分(1083) 恶意程序是首先对原始合法压缩包文件进行解压反编译操作，再插入有着恶意行为的编码，

不论是哪种恶意软件，都是将能够执行恶意行为的代码加入到已有应用程序中来实现恶意行为的。首先对正常应用程序执行反编译处理操作，捕获当前应用程序的源代码，紧接着需要植入恶意代码到最初的源代码文件中，把越权信息等加入到 apk 配置文件中，最后执行重打包操作并发布到第三方市场，如图 2.3 所示。

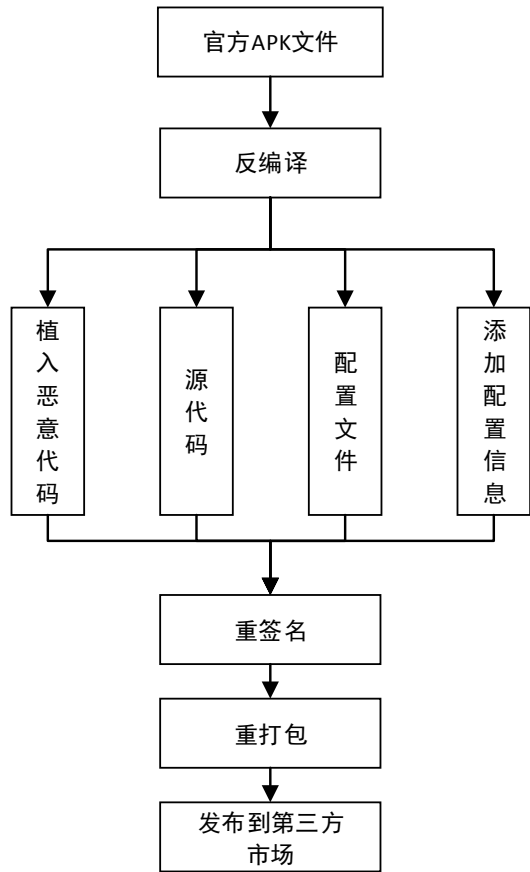


图 2.3 恶意代码植入过程

判断应用程序是否经过重打包处理，安卓系统安全机制中的签名机制给出了解决方案。安卓系统的签名机制可以有效地审核开发者的身份。一方面能够避免出现修改程序包的情况，另一方面可以帮助程序确认信赖关系，实现具有相同私钥的应用程序之间代码和数据的共享^[27]。APK 签名后会生成一个 META-INF 文件目录，其中包括 MAINFEST.MF, CERT.SF 和 CERT.RSA，它们均为签名类文件。在装载 APP 时，安卓系统会在第一时间对待安装的 APK 包进行检查，看其是否包含签名文件，只有有着签名文件的 APP 系统才会同意安装。同样地，只有在数字签名相同的情况下才会进行应用程序的更新和升级，否则当作新程序处理。

2.2.2 串谋权限攻击

安卓应用程序对资源的访问分为两种方式：一种方式通过访问其他应用程序的组件实现，另外一种方式是使用 Framework 提供的功能。一种方法是最为普遍的自定义的权限机制控制，另外一种系统是系统层次的权限控制。通常，无权限说明的 APP 无法查访目标资源。但是能够通过访问其他应用程序组件的方法实现权限的提升，如图 2.4 所示。由于组件 A2 没有连接网络下载文件同时将文件保存至存储卡这一权限，因此 A 程序是不能够进行联网下载文件的一系列操作的。但是，B 程序下组件 B1 拥有此权限。这时，A 程序下组件 A2 能够采用访问 B 程序下组件 B1 的方法完成文件的下载。这种方法巧妙地挣脱了安卓系统的权限机制的约束，也最终对安卓系统的安全造成了威胁。

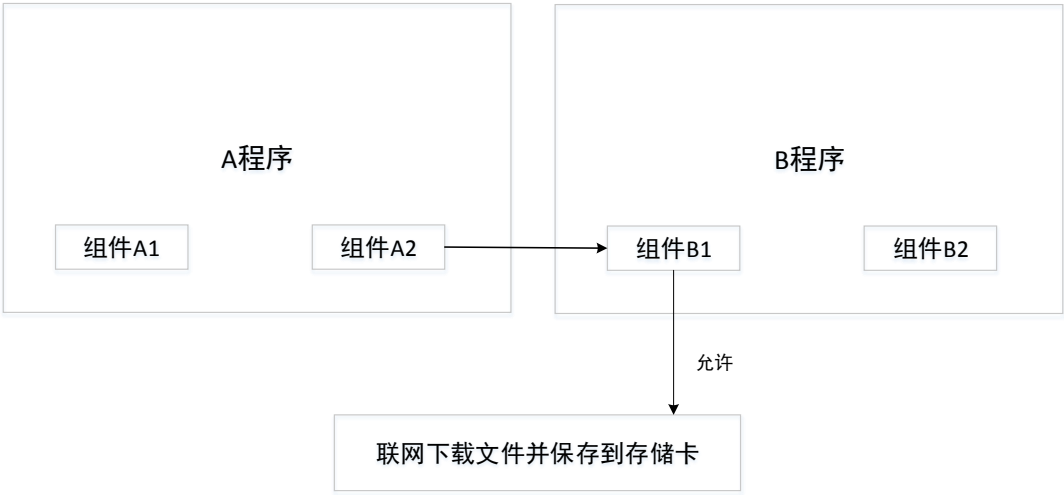


图 2.4 权限提升原理

2.2.3 四大组件安全

安卓框架为开发者提供了四大组件^[28]，分别是：活动、服务、广播接收器以及内容提供者。安卓系统中任意应用程序都包含四大组件中的一种或者多种。组件与组件之间，包括组件与应用程序之间都是依靠 ICC 机制来实现通信的。

(1) 活动组件安全

活动是其中唯一用户可见的组件，主要用于信息的显示或者输入。活动作为应用程序显示功能的载体，其安全性至关重要。对于活动的安全问题，至关重要的是访问权限控制机制。由于安卓系统的四大组件经过设定的 Intent 过滤装置过滤后，通常情况下其他外部 APP 能够对其进行查访。这就意味着此应用程序或许会面临其它外部程序的恶意攻击。

除此之外，活动组件的另外一个安全问题是活动劫持。2011 年，活动劫持技术经由 SpiderLabs 安全问题分析小组公布。该技术的原理是：有着活动劫持技术的恶意 APP 被装载至用户移动设备后，恶意应用系统会查访系统中的全部进程，一旦遍历到劫持目标的活动(如银行登录界面)，则启动带有独特标志的活动覆盖正常的活动。当用户结束输入时，恶意程序将得到的用户信息发送到指定的服务器，然后切换回正常的活动。活动劫持技术不用得到任何的权限许可，这进一步加重了对它的检测难度。

(2) 服务组件安全

服务组件是不包含任何可视化的界面的。它长期运行在安卓系统的后台，主要进行一些长期操作或为别的组件提供支持服务。比如某一服务能够在不影响用户使用手机程序的前提下可在后台下载电视、播放音乐、进行计算等。服务组件与活动组件类似，当服务组件指定了 Intent 过滤器之后，它默认是可以被其它应用程序访问的。因此，恶意程序可以通过启动服务、绑定服务、停止服务等操作对应用程序进行破坏。应用程序若想杜绝此现象的出现，可以在配置文件中设置 `android:exported=false`。

(3) 广播接收器组件安全

广播机制是一种基于消息发布和消息订阅的事件驱动模型，即广播发送者发送消息，广播接收者需要订阅消息才可以接收消息。在安卓系统架构中，广播主要包括以下两种：一，APP 自定义广播；二，系统广播。后者主要职能为对系统信息的变化进行显示与通知，如电池用量、短信息提示、来电提示灯等。应用程序自定义的广播主要功能是显现应用程序自身的一些变化。安卓系统具有普通广播以及有序广播这两种广播方式。普通广播能够发送无序的广播，还能够被全部接收器接听；第二种方式能够发送的是有序的广播，该广播会被优先级较高的接收器优先接收并依次往下传递。

(4) 内容提供者组件安全

由于 Linux 用户的 ID 限制机制，不同的应用程序之间无法进行数据的直接交互。因此，安卓系统提供了内容提供者组件来实现数据的共享。应用程序只有将数据保存在内容提供者中，才可以使其被其它申请了相应权限的应用程序访问。比如：一般手机存放的联系人相关信息对于手机用户来说是非常重要的，当某一个应用程序成功获取到相关操作权限，这时该应用程序就能够任意地借助内容提供者组件带来的便利去操作联系人等隐私数据。所以，应用程序开发者有必要提前声明内容提供者组件的读权限与写权限。否则，恶意程序不需要任何权限便可以获得组件中的数据，这使得用户的手机暴露在危险之中。

2.3 自动化测试技术

安卓系统动态测试是指在安卓终端设备或者安卓模拟装置对用户执行程序的启动、使用等一系列操作进行模拟，同时在应用程序运行的过程当中实现对敏感行为的监控，以便进一步检测该应用程序是否安全以及质量相关的问题。较之于人工执行，数字化动态测试大幅提升了测试的准确率与测试效率，极大的缩短了测试时间，节省了人力。安卓系统提供了 **Monkey** 等自动化测试工具，能够有效触发 **APP** 中的敏感行为，同时能够对应用系统在运行时的稳定性与安全性进行检测。具体可见表 2.1。

表 2.1 自动化测试应用工具列表

应用工具	主要适用场景
Monkey	通过传送伪随机事件流（触屏等）实现恶意行为触发
MonkeyRunner	利用脚本 Python 及 API 接口自定义事件流，对用户行为进行模拟以实行恶性行为触发
HierarchyViewer	针对 UI 界面实施检视，基于控件布局及属性等信息以完成触发

2.3.1 Monkey

Monkey 属于自动化检测工具的一种，不仅能够在模拟器中运行，也可在现实应用终端运行，对 **APP** 能够进行随时随机触发敏感行为，以传送伪随机用户事件流的形式来检测安卓 **APP** 系统。上述随机事件流通常指触屏、击键等^[29]。编写插入命令指令，可对目标 **APP** 进行指定，也可设定伪随机时间的数量，在现实终端或模拟装置中触发恶意行为。目标 **APP** 在运行中执行该操作，整个触发仅在目标 **APP** 崩溃或者触发时间结束后才可停止，由此可对 **APP** 系统的安合性与稳定性进行检测。尽管 **Monkey** 检测工具可模仿用户操作，可其传送的事件流为随机的、无指定性的，因此仍有待完善。

2.3.2 Monkeyrunner

Monkeyrunner 是^[30]一种可扩展的自动化测试工作，同时能够实现对多个设备节点进行管控。根据预设的屏幕、按键事件的参数值，**Monkeyrunner** 可参照触发步骤自主进行事件触发，从而完成复杂逻辑的事件触发操作^[31]。与此同时，它有着许多 **API** 接口。不仅可利用 **API** 进行程序编写，同时能够对若干个模拟器或现实终端设备上的 **APP** 进行恶行行为触发操作，该过程中所执行的触发事件为相同的，以完成对若干装置的检测与控制；而且 **Python** 脚本和触发系统的研发能够对触发过程与触发事件流进行管控，以完成自动化可延

展的恶意程序触发操作^[32]。Monkeyrunner 主要包括 MonkeyDevice 等工具。此外，Monkeyrunner 允许程序研发者对类进行设置，以完善自动化恶意行为触发的延展^[33]。

2.3.3 HierarchyViewer

相较于前两种自动化检测工具，HierarchyViewer^[34]的不同之处在于它用于检测安卓系统 SDK 中的 UI 界面。其主要作用为对 APP 界面的信息列表、控件属性信息与控件布局信息进行检查。并基于各控件元素间的逻辑关系组建控件结构树，以对控件间的树形关系进行阐述。HierarchyViewer 与移动设备的连接方式主要采用 Socket 通信方式，即传送 DUMP 命令及端口监听数据流以采集控件信息与窗口界面信息。这里，数据流即为格式一定的字符串，而 HierarchyViewer 可对其进行分析、处理，从而得到 UI 界面的信息列表、控件属性信息与控件布局信息^[35]。

安卓 APP 是在事件驱动基础上实现的 UI 应用程序，采用 HierarchyViewer 自动化检测工作可获取用户（UI）控件信息。HierarchyViewer 结合 Monkey 或者 Monkeyrunner 实现的自动化检测工具，基于控件实现行为触发是现今安卓程序恶意行为触发的核心方法。论文采用 Monkeyrunner 实现安卓恶意软件行为的触发操作。

2.4 安卓系统各安全架构层检测机制

2.4.1 应用程序层库替换

安卓系统各个组件之间数据的传递运用广播机制，从应用程序层面来看，若要得到广播，一个重要条件是完成了广播接收器注册工作。广播根据优先级别，由高至低进行传播，在完成高等级数据广播之后才向下进行广播。如果高等级接收器选择了 abortBroadcast() 方法，此时无法正常进行广播接收，对于低等级接收器来说，无法有效接收相应的广播。值得注意的是，从安卓系统来看，其自身无法对广播优先级别予以设定，而通过应用程序对有关广播接收器进行注册后，才进行设定，基于此，能够方便地进行应用程序层的应用程序行为的实时监控。现如今，绝大多数的安卓应用程序都是选择该种模式对骚扰电话、垃圾短信进行拦截的。但其无法在防范恶意软件上广泛应用，原因包括以下三点：首先，不管是安全的应用程序还是恶意的应用程序都是运行在应用程序层的，安全的应用程序不具备对恶意软件的优先级别进行控制的权限；其次，该监控模式有待完善，监控面往往受系

统广播影响比较大，如接到新来电，或者接收新消息时，系统足够可以进行广播，不过对联系人与短信等信息进行访问的过程中，系统无法自动进行广播。再次，这一种监控方式不能够屏蔽安全凭证低的应用程序，设定只能够将广播发送给安全系数高的应用程序。

替换 `libc.so` 库，如此替换从用户角度而言是公开透明的，不会对用户正常使用产生干扰。系统调用时，将对需通过 `Auranium` 进行替换的库中重新进行定向，同时在库内完成安全检查工作。若符合安全标准，则转向目标的系统调用，从而开始进行系统调用。如果不符合安全标准，则返回上层操作，同时向用户发出警示信息。

针对库替换而言，其最为有利的条件就是最大限度防止系统代码被修改。若要达到上述目的，只要重新对目标 APP 对应的代码进行打包操作处理。该方法可操作性较强。不过也存在诸多问题：首先，对原 APP 完整性产生影响。重新进行打包操作，也许会造成应用软件无法正常执行。就 `APKTOOL` 反汇编程序来说，其可以把安卓系统内的 `dex` 文件进行反汇编处理，得到 `smali` 应用程序。从 `smali` 来看，和原生 `dex` 文件进行对比，其可读性、可更改性要好得多。完成修改之后，运用 `APKTOOL` 反汇编程序进行重新打包处理。如此便实现了更改目标应用的目的。通过 `APKTOOL` 反汇编程序进行转化，会发生损耗。换言之，重新进行打包处理之后，该程序将无法正常运行，或存在 `Bug`。尤其是当下安卓程序使用面变得日益广阔，在加固程序时采取了不少混淆式的手段。综上可知，目前找到了成熟的手段对该程序应用进行加固处理了。与此同时，从目标软件来看，其签名发生了改变。值得注意的是，要对软件版权问题予以重视。程序开发人员不希望其程序出现修改，同时也不愿意承担，因为重新打包的操作，造成应用程序不够稳定，会造成经济损失。所以，这并非有效处理方案。就安全层面而言，如此操作安全隐患巨大。第一，依托 `libc.so` 库而进行的替换，无法对未使用 `libc.so` 发起的攻击进行有效防御。换言之，若恶意应用程序开发人员没有通过 `libc.so` 库调用系统，则便无法进行库替换。恶意应用程序开发人员对 `Native Code` 进行访问时，可利用 `JNI` 来完成。依托 `Native code` 调用系统，在绕过 `libc.so` 库之后便能够避免被攻击。

2.4.2 应用程序框架层挂钩 API

应用程序框架层为应用程序提供了一系列的 `API` 接口。对应用程序而言，其能够对该接口进行调用，然后达到访问信息、通信录等目的。若要在软件框架层监控应用程序行为，就必须在安卓框架的源码中植入对应的监控代码段。这样的监控方式能够全面地实时地监控应用程序的行为，但由于安卓版本多种多样，目前尚未找到不同版本有效兼容的处理方

法。就安卓版本来说，其更新速度快，不同版本的框架不完全一样，因此，该方案不具有良好的通用性，必须根据版本更新情况，持续地对代码进行修改，同时要对系统进行重新编译。就该监控模式来主说，其有着一个不足之处，即会对系统性能产生严重耗损，并导致应用软件无法长期保持高效运行，一些高端设备能够应对这一问题，但是某些低端的设备就能够明显暴露应用程序执行延迟这一现象，最终用户体验受到严重影响。

安卓系统为研发人员配置了大量 API 函数，安卓软件在对该函数进行调用后，实现诸多功能^[36]。因此，通过监测应用程序调用的 API 函数，便能够全面分析应用程序的相关行为，辨别其是否具有恶意倾向^[37]。

采用 Hook 技术截获 API 调用的基本原理是先将代码注入到目标进程中，然后使用 ptrace^[38]函数对目标进程 attach，接着跳转到 mmap 函数来分配一小段内存空间，然后将机器码拷贝到刚分配的内存中去，接下来执行注入的代码。Hook 常用工具为 Xposed 框架，获得 root 权限后，安卓的 Xposed 框架能够对在应用程序的运行产生影响并且不修改 APK。使用替换/system/bin/app_process 的方式对 zygote 进程进行有效掌控，从而使得 app_process 可在启动时对 XposedBridge.jar 包进行及时加载，进而截获 zygote 的进程。

2.4.3 内核层修改系统调用表

内核层的监控机制采用的是捕获系统调用来对应用软件的各种行为进行动态监控。和前述监控模式进行对比，就内核层监控而言，其优点如下：第一，内核层的权限比其他安卓架构层要高，这使得它能够对用户层实际操作时的应用软件的各种行为产生影响；第二，内核层监控能够得到更丰富的信息，不管是什么文件，对操作进行访问时，均离不开内核层支持；第三，从内核层监控来看，其不会发生严重的系统性能耗损，同时，其与系统调用进行连接的接口，与框架层进行对比数目要少得多，所以该监控模式与框架层监控模式进行对比，其对系统的影响相对要小得多；第四，内核层监控有较好的安卓版本兼容性，适用的范围更广。与不稳定的应用程序框架层进行对比，其稳定性较强，尤其是在进行系统调用时，其所提供的接口极少发生变化。捕获应用程序调用的系统调用接口是内核层监控的重要组成部分，Linux 提供了多种能够便捷地跟踪以及截获系统调用的方法。

内核层监测机制采用的是替换系统调用表中的系统调用函数的模式。该模式对系统调用表进行更改时，是利用内核模块来完成的。使用自身模块内的函数去替换系统原有的调用函数，从而实现对目标进程行为的监测。监测具有强制性特点，是该模式的优势之所在。针对安卓系统而言，因为其 root 权限受控十分严格，如果将内核植到内核模块当中，则取

出的时候会比较困难。如此一来，就不需要有些行为绕过监控的范围了。在内核层监测安卓系统应用程序时覆盖全面。不管是否为安卓进程，不管是否为第三方应用，均能够被监测。

2.5 本章小结

本章介绍了安卓系统架构及四大组件，阐述了安卓安全机制中数字签名、沙箱机制以及权限控制机制，讨论了目前安卓系统存在的安全问题，其中设计重打包技术、串谋权限以及四大组件安全。讨论了应用程序层监测机制、应用程序框架层监测机制、内核层监测机制的原理和优缺点，着重分析了应用程序层库替换、应用程序框架层挂钩 API，内核层修改系统调用表三个行为检测方法，最终选定采用内核层修改系统调用表这一监控技术。

第三章 安卓恶意软件动态监测方法研究

本章研究了基于内核的动态行为监控方法和应用程序安全评估策略。分别介绍了内核层的两个功能点：系统调用截获功能、系统调用解析功能。系统调用截获功能研究了系统调用流程以及系统调用截获的原理。系统调用解析功能通过研究 **Binder** 解析分析应用程序行为。另外，研究了恶意应用程序分类，研究了应用程序安全评估策略方法，制定了应用程序安装前静态的安全防御策略和应用程序运行时动态的安全评估策略。为第四章针对权限泄露、合谋攻击、财务攻击、隐私数据窃取这四种恶意攻击类型设计和描述详细安全评估策略打下基础。

3.1 基于内核的动态行为监控方法

Linux 处在安卓系统平台的最底层，其包含的 **Java** 代码和本地代码在 **Linux** 中是按照进程追个执行的^[39]。现阶段有一些包含恶意攻击的应用程序，在执行恶意攻击行为前会检测是否有监控软件等存在于系统中，如果有则躲避这些程序的监控，继续执行恶意程序。一方面，**Linux** 操作系统拥有安卓平台最高的权限，如果将动态行为分析程序存放到内核中以实现行为监测对于拥有一般权限的应用程序来说是难以检测的，这样就避免出现应用程序恶意规避监测这一现象。另一方面，**Linux** 系统是安卓系统平台的一个重要组成部分，是实际存在于智能移动相关设备当中的，能够很方便地去监测并且分析应用软件的行为^[40]。基于内核的行为监控方法虽然能够克服上面提到的困难点，但在实际执行过程当中还需要认真考虑一下两个相关问题：同样是 **Linux** 操作系统，但是安卓平台下的 **Linux** 系统和传统的 **PC** 端的 **Linux** 系统是不一样的，那么怎样去设计安卓平台下 **Linux** 系统中的自定义内核监测模块？设计完成后怎么去部署和实际运行这个监控模块呢？

论文研究了内核监控中最重要的两个功能点：系统调用截获功能、系统调用解析功能。系统调用截获功能在内核初始化之后启动，执行安卓应用软件行为监控，通过截获应用的系统调用函数，监控应用程序的行为；系统调用解析功能主要采用 **Binder** 解析技术，解析系统调用参数信息，获取安卓应用的行为和数据参数信息。有效的内核行为监控方法才能还原应用程序行为，继而根据行为特征给出相应的安全评估策略。

3.1.1 系统调用截获功能

基于内核层的动态行为监控中，内核层行为监控模块采用的是用户自定义的内核监控模块，编写完成后再使用可加载内核模块技术将自定义内核监测模块动态地加载到内核当中去。待测应用程序安装运行后，自定义的内核监控模块就能够对应用程序运行过程中的系统调用进行实时捕获，要想实现这一功能，需要将系统调用表 `sys_call_table` 中的系统调用函数替换为用户自定义的函数。

(1) 系统调用流程

通过描述 `open` 系统调用分析系统调用的实际过程，如图 3.1 所示：应用程序发出 `open` 系统调用执行请求。这时函数就会首先产生一个软中断命令，这个命令使得 CPU 陷入到内核的一个状态。紧接着，CPU 去响应这个软中断，通过查找中断向量表得到中断入口，根据中断入口区获取 `open` 系统调用号，进而获取系统调用表的基地址。接下来需要根据获取到的系统调用表的基地址以及系统调用号去计算出 `open` 系统调用函数的入口地址，最后只需要将算出的地址传给程序计数器，就能够顺利执行 `open` 系统调用函数了。

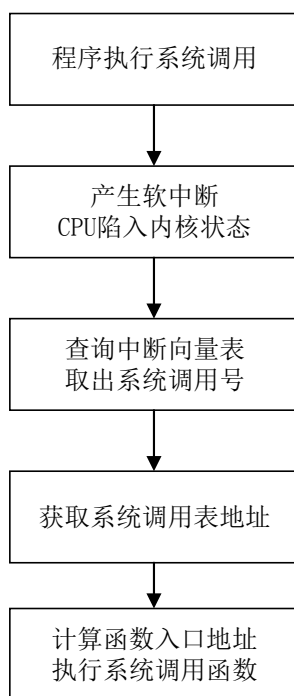


图 3.1 系统调用步骤

(2) 系统调用截获原理

系统调用截获的关键在于系统调用表地址的获取，也就是 `sys_call_table` 基地址的获取，Linux 在解决系统调用相关处理问题时，从用户发出请求到真正执行相关的系统调用，前后

一共需要执行两次查询表的操作。一次是按照指定索引根据中断描述符表进行中断处理函数地址的查找，一次是根据系统调用号通过查询系统调用表地址获取相对应的系统调用服务例程地址。结合中断描述符表以及系统调用表表项可以看出，任意一个表项都有与之相对应的处理函数的地址，需要用自定义函数地址去替换系统原有的处理函数地址，最终达到系统调用截获的目的。接下来需要通过可装载内核模块技术动态地将自定义模块接入到内核系统中，这样在应用程序安装启动后，就可以实现系统调用的实时截获相关功能。

3. 1. 2 系统调用解析功能

Binder 技术是系统调用解析功能实现的核心，通过采用 Binder 技术解析捕获到的应用程序系统调用，得到应用程序的相关行为参数信息。在安卓平台下，任意一个应用程序都是由若干个活动以及若干个服务组合而成的，一方面，这些活动和服务有可能是在同一个进程中运行的，另一方面也可能是在不同的进程中运行的。那么运行在不同进程中的活动和服务之间是如何实现通信的呢？安卓系统为了实现这一过程引入了远程过程调用。Binder 框架作为安卓系统中标准的进程间通信机制，是在 Open Binder 的基础上进行改编的，而 Open Binder 是由 PalmSource 公司开发，它能够实现大多数的操作系统中组件之间的相互通信^[52]。

与 COM 以及 CORBA 分布式组件架构相类似，按照通俗的方法来说就是能够实现远程过程的调用。在安卓系统平台下，Binder 进程间通信机制由若干个组件组合而成，分别是 Binder 驱动程序、客户端、服务器以及服务器管理程序，如图 3.2 所示为这些组件的关系。

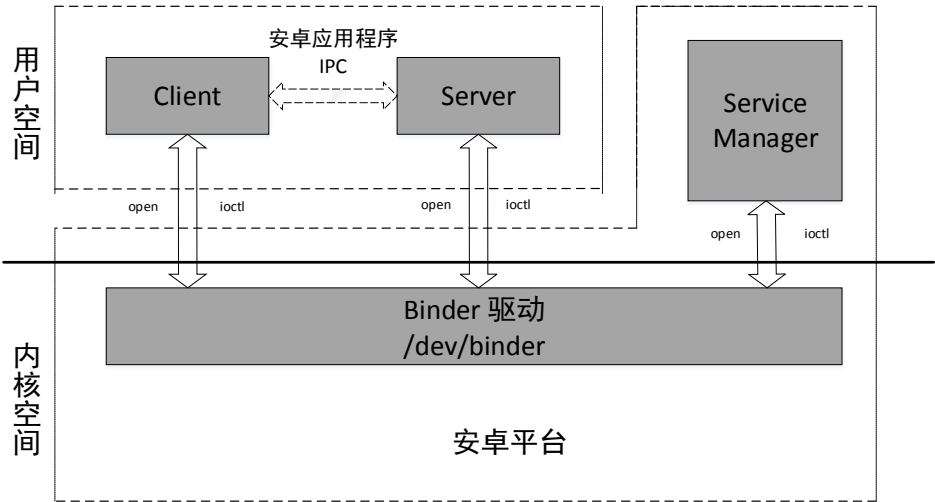


图 3.2 Binder 组件间关系

在用户空间中运行的有客户端、服务器以及服务器管理程序。在内核空间中运行的有 Binder 驱动程序。其中，安卓系统负责实现服务器管理程序以及 Binder 驱动程序的一些功

能。而服务器以及客户端不属于安卓系统负责的范围，它们需要相应的应用程序去实现。用户空间与系统设备文件之间的通信依托于 **Binder** 驱动程序，而服务器管理程序要想实现与 **Binder** 驱动程序的交互，需要借助于 **open** 函数以及 **ioctl** 函数，服务器以及客户端同样需要借助于这两个函数实现与 **Binder** 驱动程序的交互。**Binder** 驱动程序还有一个重要的作用就是能够实现服务器与客户端的交互。服务器管理程序除了进行各个服务器的管理工作，还需要能够实现服务器查询等相关功能。

当客户端需要请求服务器的某一个方法时，表面上看起来是请求了本地的一个方法，实质上真正请求的是本地的某个代理接口。客户端如果想要使用这个接口，首先需要提前将相关参数载入到某个程序中，进一步去发送这个程序，**Binder** 驱动进行接收，然后将接收到的程序发送到指定的服务器中。后面要进行的就是对这个程序的相关解析工作，处理完成后返回所得的结果给客户端。整个过程最重要的部分是服务器端方法和客户端接口的相对应。

3.2 应用程序安全评估策略

应用程序安全评估策略可分为应用程序运行前静态的安全防御策略和应用程序运行时动态的安全评估策略两个部分。用户下载应用程序准备安装时，对待安装的应用程序进行检测，论文基于权限列表设定了 7 条静态主动防御策略，只有满足静态主动防御策略的应用程序才能够进一步安装，并继续进行接下来的动态安全评估，否则不允许安装并发出危险警告。动态安全评估策略基于内核行为监控模块所截获的系统调用，论文基于上下文信息制定了一种安全评估策略。

3.2.1 恶意应用程序分类

恶意软件的攻击类型主要有：权限泄露、财务攻击、合谋攻击以及隐私数据窃取。另外还包含一些新型的恶意软件攻击类型。论文针对不同的恶意软件攻击行为给出相应的安全策略。

权限泄露在安卓移动平台中普遍存在。比如 **RATC** 与 **EXPLOID** 两者表现出来的危险性都非常高。原因来自 **Linux** 内核中的漏洞较多，可以通过不正常手段得到 **Root** 权限。若得到 **Root** 权限，则表示安卓系统已完全被恶意软件掌控^[45]。

合谋攻击^[46,47,48]属于不易查觉到的权限泄露攻击。合谋攻击的工作原因是通过至少两个以上的软件应用达到攻击的目的。其中参与攻击的 APP 申请的权限不多，其目的是远离 AV 引擎的数据检测。等在完成若干个 APP 安排以后，APP 与 APP 完成的信息传递就会通过原来设定好的模式完成。

在恶意行为中还有一个重要的攻击手段就是对财务数据进行攻击。此类攻击的操作方法是利用特定的数据传递进入到系统服务器，在用户毫不知情的状态下完成付费项目的订购。此类攻击表现出来的行为主要有两类，一类是用户资金受损；另一类则是垃圾邮件和短信大量增加。

除了以上几种攻击类型外，还有一类恶意软件的存在，它的主要目的是完成用户非公开数据的获取。这些非公开数据涉及到用户的信息配置、用户的通信簿、用户的短信等。这些都属于隐私信息。如果恶意软件得到了用户的这些非公开信息以后，会按照事先制定的模式完成远程操作。

3.2.2 静态安全防御策略

静态安全评估策略基于配置文件中的权限申请，论文从单个权限和权限组合两个角度分析应用程序是否具有恶意倾向，进而组织恶意应用程序的安装，达到主动防御的效果。但是权限机制中包含着某些漏洞，一些恶意软件不具备权限却可以执行越权操作。所以，论文在后面的动态安全评估策略中，针对权限泄露这类问题，给出了具体的安全策略，保证了在应用程序运行前和应用程序运行中对权限列表进行双重检查。解决了权限机制漏洞问题。

单个权限以设置应用程序信息为例，申请了修改列表参数权限的应用程序，可以执行电话接管等操作，直接导致系统权限泄露，需要阻值此类应用程序的安装。

权限组合以语音记录权限、电话状态权限和网络连接权限为例，同时申请了语音记录权限、使用电话状态权限和连接互联网权限，此类应用程序可以通过网络窃取并实时发送手机的通话内容，直接导致用户隐私泄露，需要阻值此类恶性应用程序的安装。

下表为安全防御策略表：

表 3.1 安全防御策略表

序号	权限声明	实现功能	恶意程序类型
1	android.permission.WRITE_SMS android.permission.SEND_SMS	防止恶意应用程序随意编辑收费短信并发送	财务攻击
2	android.permission.SET_PREFERRED_APPLICATION	防止应用程序获取电话功能管理等操作	权限泄露
3	android.permission.RECEIVE_SMS android.permission.WRITE_SMS	防止恶意应用程序随意接收资费短信发送短信	财务攻击
4	android.permission.RECOED_AUDIO android.permission.PHONE_STATE android.permission.INTERNET	防止恶性应用软件通过网络窃取并实时发送手机的通话内容	隐私泄露
5	android.permission.INTERNET android.permission.ACCESS_FINE_LOCATION android.permission.RECEIVE_BOOT_COMPLETE	防止应用程序获取手机用户的位置信息，并将其通过网络进行发送	隐私泄露
6	android.permission.INTERNET android.permission.RECEIVE_BOOT_COMPLETE android.permission.ACCESS_COARSE_LOCATION	防止应用程序获取大概的位置信息并发送	隐私泄露
7	android.permission.INSTALL_SHORTCUT android.permission.UNINSTALL_SHORTCUT	防止应用程序修改快捷方式的操作。	隐私泄露

3. 2. 3 动态安全评估策略

为使针对恶意软件的检测率不断得到提升，必须通过上下文特殊行为的辅助来完成对应判定。第一步必须判定上下文信息的定义，从实际出发，在较短时间内判定出前后台信息与恶意软件之间的关系是十分必要的，常规的观察行为都是恶意行为处于系统操作过程中出现的，而这一系列的过程对于用户来讲并不知情，因此首要工作就是判定该行为发生在前后还是后台。精准的判定在后台发生是不太容易实现。论文运用有效的方法完成判定，判断以安卓系统的另一个操作步骤为基础。若系统显示出来的地址是前台，对应的 UI 操作也是必须出现的，Binder 通信中会产生地址访问数据。对应的 Ioct 系统调用数据中也会产生对应的数据。进程数据与线程数据结合在一起，能够进一步完成后台是否出现目标应用的判定。线程中如果出现，则可确定该行为产生的地址必定是后台。若在进程中出现，必须进行上下文 UI 操作的探求与分析。如产生 UI 行为，则能够确定用户已了解到该行为的出现。不论用户是自主点击亦或是观察到行为的出现。都是将手机用户融入到应用程序运

行过程中的。虽然这种判别不是很精确，但可以真实地将问题表述出来。还有一个较为关键的数据则是目标行为出现的频率。通常来讲，敏感行为出现的频率相对较高，一般表现出来的现象是目标性更强。如用户在应用短信程序，短信也是发给用户熟人的。这个情况下不管用户使用数据的频率多高都不用视为威胁。行为上下文也是一个非常关键的环境因素。该阶段数据表现出来的主要作用就是完成信息的记录，如直接进入互联网平台进行操作，若用户在发生这些行为之前取得联系人或者获得非公开数据，进行网络连接产生的威胁系统数要高于正常情况。所以说对用户行为进行系统分析是十分重要的程序。

论文基于上下文信息定义了一种安全评估策略，用于判别应用程序是否具有恶意行为。动态安全评估策略基于内核行为监控模块所截获的系统调用相关数据。

在图 3.3 所示的安全评估策略的表示模板中，定义如下：

requester 是与 **operation** 等对象关联的变量，是发起操作请求的应用程序用户标识(**UID**, **User Identifier**)；**UID** 标识了哪个用户运用了这个程序。

operation 参数数据可从应用程序行为信息表中获得，表示程序在目标设备上执行的具体操作。

data 是发起 **operation** 传递的数据信息；**data** 这个对象是可有可无的，所以用[]表示。

target 是操作执行的目标设备，论文中默认本地安卓系统设备，当需要使用系统服务时，可定义 **target** 为所请求的系统服务。

context 表示操作执行的上下文环境，属于一个对象，包含 **isBackground** 等属性区分上下文中具体信息，它提供与用户交互的信息，操作频率，运行状态等上下文信息。

condition 表示应用需要满足的执行条件。**record** 表示将满足的条件 **condition** 记录到日志的语句。例如当条件满足 **operation == network connect** 时，记录“网络连接”到日志中。

通过安全策略后执行 **security-enforcement** 语句。**security-enforcement** 是一系列的安全操作集合，包括 **frequency>limit** 判断操作频率是否超过限制，超限则报错；**warn user Dangerous** 向用户发出危险警告等操作。

record 与 **security-enforcement** 都是可选项，所以用[]表示。“;”表示语句之间的分隔符。

ON、**GO ON** 属于关键字，**IF**、**THEN**、**ELSE** 属于条件表达式的关键字。

其中“[]”表示里面的对象或参数是可选项。

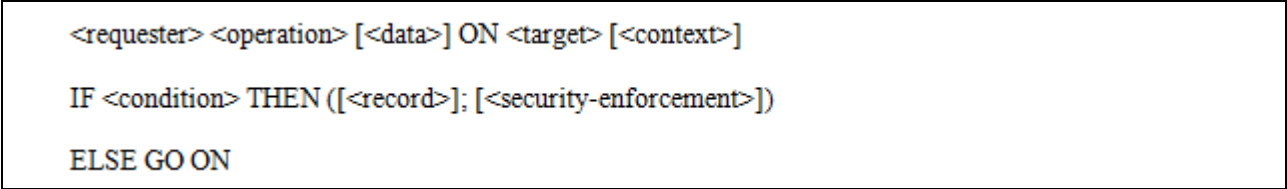


图 3.3 安全评估策略模板

3.3 本章小结

本章研究了基于内核的动态行为监控方法和应用程序安全评估策略。分别介绍了内核层的两个功能点：系统调用截获功能、系统调用解析功能。系统调用截获功能研究了系统调用流程以及系统调用截获的原理。系统调用解析功能通过研究 **Binder** 解析分析应用程序行为。另外，研究了恶意应用程序分类，研究了应用程序安全评估策略方法，制定了应用程序安装前静态的安全防御策略和应用程序运行时动态的安全评估策略。

第四章 安卓恶意软件动态监测方案设计与实现

本章首先介绍了安卓恶意软件动态检测方案的设计目标，将整个监测系统分为 3 个模块，分别是行为触发模块、行为监控模块以及行为分析模块。介绍了行为监控模块中内核初始化和系统调用捕获的实现过程，详细阐述了行为分析模块的系统调用数据的抽取。针对应用程序权限列表给出了静态安全防御策略，针对行为分析给出了应用程序动态安全评估策略。

4.1 安卓恶意软件动态监测方案设计目标

安卓恶意软件动态检测方案的设计目标：通过对应用程序行为的监控，判定恶意应用程序的攻击类型。一方面，针对应用程序动态行为监控，需要选择最全面的监测方法，从而能够充分获取应用程序的行为信息。论文选定了监测最全面的基于内核层的替换系统调用表的方法。另一方面，需要准确判定恶意应用程序的攻击类型。论文针对不同的恶意攻击类型设计了详细的安全评估策略。

论文主要研究的是安卓系统应用程序行为动态监控相关技术以及应用程序安全评估策略，提出了一个基于安卓平台的恶意软件动态监测模型。这个模型可分为三个模块：行为触发模块、行为监控模块、行为分析模块。行为触发模块设计目的是为了减少人为操作，采用 MonkeyRunner 实现应用程序的自动安装、运行和卸载。对待安装的应用程序进行检测，论文基于权限列表设定了 7 条静态主动防御策略，只有满足静态主动防御策略的应用程序才能够进一步安装，并继续进行接下来的动态安全评估，否则不允许安装并发出危险警告。应用程序启动完成后，进入行为监控模块。在对应用程序行为监控之前，要首先完成内核的编译，将内核中系统调用表的系统调用函数替换成自定义的函数，加载编译后的内核，实现系统调用的截获，并采用 Binder 技术解析系统调用。将获取到的系统调用行为和数据信息，存储到 SQLite 数据库，依次进行权限泄露策略、财务攻击策略、合谋攻击策略以及隐私数据窃取策略判断，判断应用程序的恶意攻击类型。

4.2 安卓恶意软件动态监测系统的框架

安卓恶意软件动态监测系统将整个监测系统分为 3 个模块，分别是行为触发模块、行为监控模块、行为分析模块。监测框架如图 4.1 所示。

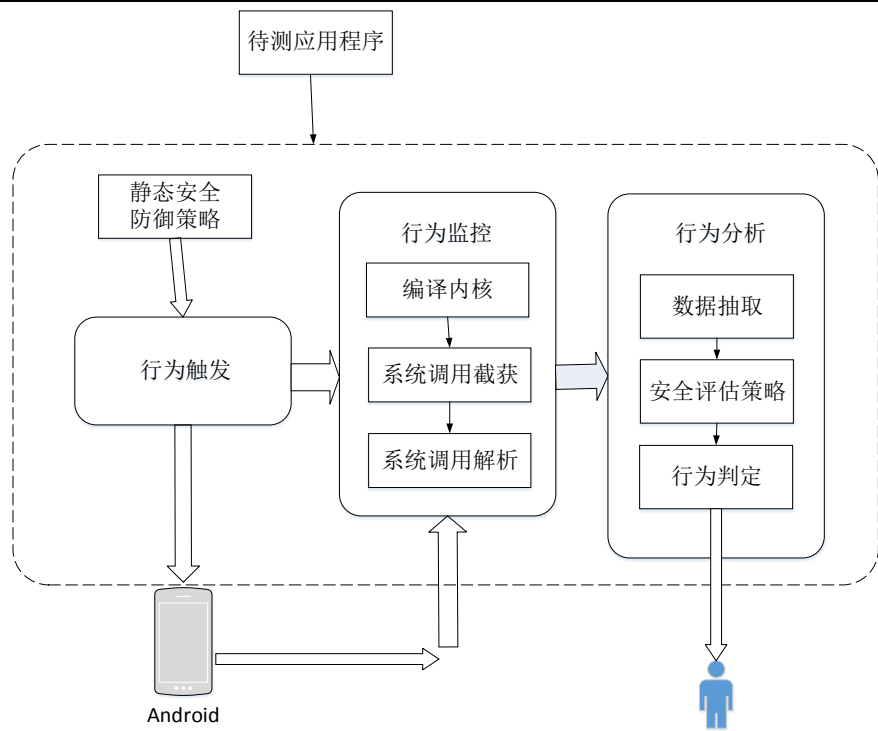


图 4.1 安卓恶意软件动态监测框架

应用程序启动之前需要检查权限列表，只有通过静态安全防御策略检测的应用程序才可以进行安装。

行为触发模块是为了实现应用程序的自动安装、运行和卸载。

行为监控模块是论文监测系统的关键模块，负责对应用程序的行为进行监控。其主要任务包括：编译内核和系统调用解析。编译内核是在对应用程序监控之前必须完成的类似于环境搭建工作，将监控程序载入到内核中去。系统调用解析方面采用 **Binder** 机制。

行为分析模块负责对恶意攻击行为进行判别，结合系统调用行为参数信息，定义上下文状态信息，依次进行权限泄露策略、合谋攻击策略、财务攻击策略、隐私数据窃取策略判断，判断应用程序的恶意性。

4.3 行为触发模块的设计与实现

在运用界面解析技术的基础上，得到了全部控件对应的中心坐标值，以及应用的窗口信息。利用脚本生成技术，把所有控件的中心坐标进行处理，得到相应的脚本，同时对 **Monkeyrunner** 进行调用执行上述脚本完成对应用的自动化安装、启动以及卸载。作如下详细阐述：

安装脚本。安装安卓应用软件，通过 **Monkeyrunner installPackage API** 进行安装，如果测试用例为 **test.apk**。相应的安装脚本具体如下。

```
from com.android.monkeyrunner
import MonkeyRunner,MonkeyDevice,MonkeyImage
device = Monkeyrunner.waitForConnection()
device.installPackage(compont = 'test.apk')
```

启动脚本。主要用来启动应用软件，这里需要通过 Monkeyrunner startActivity API 获得相应的启动脚本。当应用程序进入运行状态后，依据前面界面解析得到的控件的坐标相关信息，实现控件的遍历点击。具体的启动脚本如下所示：

```
from com.android.monkeyrunner
import MonkeyRunner,MonkeyDevice,MonkeyImage
def doPress(x,y,type)
device = Monkeyrunner.waitForConnection()
```

卸载脚本。在对脚本遍历之后，软件测试结束，需对完成测试的安卓软件进行卸载处理。通过 Monkeyrunner removePackage API 获得相应的卸载脚本。从所需卸载的程序看，如果测试用例为 test.apk，则所得到的脚本具体如下：

```
from com.android.monkeyrunner
import MonkeyRunner,MonkeyDevice,MonkeyImage
def doPress(x,y,type)
device = Monkeyrunner.waitForConnection()
device.removePackage('test.apk')
```

4.4 行为监控模块的设计与实现

4.4.1 行为监控模块框架

行为监控模块首先要完成内核的编译工作，采用替换系统调用表的方式实现自定义内核监测模块，也就是需要将系统调用表中的地址替换为自定义的调用函数地址，编译完成后，采用可加载内核模块技术将自定义模块动态地链入到内核系统中，这样就完成了行为监控的准备工作，应用程序运行后，自定义内核监测模块就能够实时监测应用程序的动态行为，达到系统调用捕获的目的。接下来就需要根据截获到的系统调用相关数据去分析应用程序行为，这里采用 Binder 分析机制。行为监控框图如图 4.2 所示。

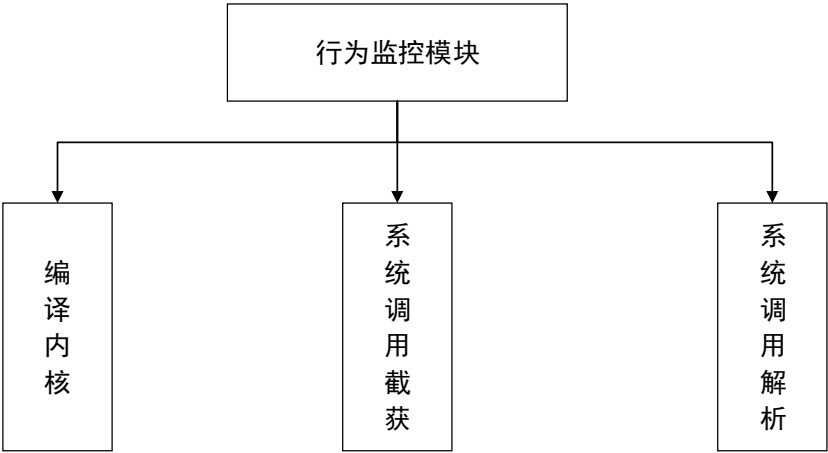


图 4.2 行为监控框图

4. 4. 2 行为监控模块的具体实现

(1) 内核初始化

论文采用了基于内核层的替换系统调用表的行为监控方法，在完成自定义模块编写工作后，如何将这一模块嵌入到 Linux 内核中呢？这里需要采用可加载内核模块技术。安卓系统平台的 Linux 内核不同于传统 PC 端的 Linux 内核，其默认情况下是不能够支持可加载内核模块这一功能的。但是可以通过设置实现这一功能，只需要修改 Linux 内核的相关配置文件，将这一功能开启即可。开启这个功能很简单，只需要将系统原本设置的此功能关闭的开关打开，就能够像正常的 PC 段 Linux 内核一样使用此功能。

第一步是 Linux 内核的重新编译。一般情况下是需要将自定义的内核行为监测模块编译成为一个镜像类的文件，紧接着需要进入到 shell 模式下，这里涉及到安卓调试桥相关内容，在进入 shell 后，使用 insmod 实现自定义行为监测模块的加载，亦或者是使用 rmmod 实现自定义模块的卸载。成功编译好内核监控环境后方可实现应用程序系统调用的截获。

Linux 内核所有的系统调用函数地址都是存放在 sys_call_table 当中的，想要实现应用程序系统调用的截获，需要将原本指向系统调用函数的地址替换为自定义的处理函数地址。那么第一步就是系统调用表地址的获取。在 Liunx 2.4.18 之前的内核版本中，sys_call_table 是能够直接被导出以供开发者使用的，但是这种操作给系统安全带来了很大的威胁，所以之后的版本是不能够实现这一操作的。那么如何获取到系统调用表的基地址呢？目前有一种通用的方法，应用程序发生异常时，Linux 内核会自动地跳转到异常向量表继续处理相关进程任务。从 0xffff0000 开始存放着全部的中断例程，0xffff0000 这个起始地址本身指向一个异常进程拷贝指令。在整个操作过程中，需要事先将系统原本的 sys_call_table 地址存放

到寄存器当中，处理过程当中搜索特定的系统调用表指令，从而找到系统调用表基地址的位置。接下来只要将其替换为自定义的函数地址即可。系统调用表地址的获取的核心代码如下：

```
/*Get the address of sys_call_table*/
long * get_sys_call_table(void){
    struct _idt_descriptor * idt;
    struct _idtr idtr;
    unsigned int sys_call_off;
    int sys_call_table=0;
    unsigned char* p;
    int i;
    asm("sidt %0":"=m"(idtr));
    kprintk("    address of idtr: 0x%x\n", (unsigned int)&idtr);
    idt=(struct _idt_descriptor*)(idtr.base+8*0x80);
    sys_call_off=((unsigned int)(idt->offset_high<<16)|(unsigned int)idt->offset_low);
    kprintk("    address of idt 0x80: 0x%x\n", sys_call_off);
    p=(unsigned char *)sys_call_off;
    for(i=0; i<100; i++){
        if(p[i]==0xff&& p[i+1]==0x14&& p[i+2]==0x85){
            sys_call_table=((int*)p)[i+3];
            kprintk("    address of sys_call_table: 0x%x\n", sys_call_table);
            return (long*)sys_call_table;
        }
    }
    return 0;
}
```

(2) 系统调用截获

定位到系统调用表的基地址后，系统调用截获工作即可开展，截获的具体流程如 4.3 所示。第一步是要将原有的系统调用地址保存起来，然后再将这个系统调用地址替换为自定义的地址，执行对应的系统调用，并且做好数据记录工作。在完成系统调用操作之后，再返回控制权给系统。

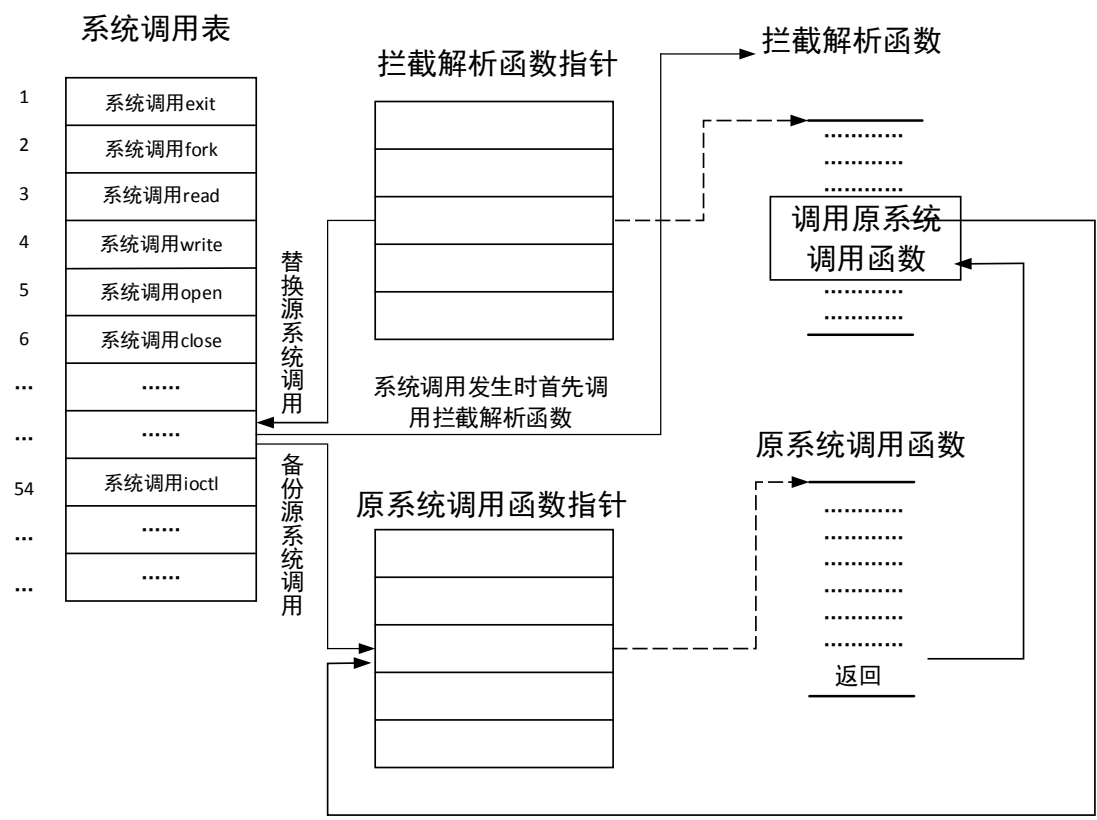


图 4.3 系统调用截获过程

安卓操作系统内核与 Linux 关联，并处于 Linux 操作系统下，应用程序表现出来的形态有两类，一类是用户态，另一类则是核心态^[41]，核心态与用户态之间的差别主要来自于前者运行级别高于后者，处于操作系统的执行层次也不同；用户态是指常态用户的使用程序，不具备较高的级别，无法进入到其他程序的操作环境中^[42]。系统调用接口的位置处于核心态与用户态的中间位置，对于核心态操作系统来讲它的主要作用是为用户态反馈更多的系统服务^[43]。利用完成的请求程序，用户态能够在某种状态下得到对应的服务内容，系统得到控制权后发出请求指令完成函数的数据反馈。

系统调用处于逻辑使用状态下的表示主要是利用系统调用号完成，实际地址的执行还需要利用系统调用表完成。sys_call_table 通过地址完成内容的索取。其中 sys_call_table 被看作是单个数组，它的主要作用是完成详细地址的保存。系统调用号工作程序是利用对应的系统服务完成下标号的注释，利用 sys_call_table 数组地址的一系列访问，直接向对应程序跳转，最终达到系统调用的目的。系统调用截获的核心代码如下：

```
//My own sys_open
asm linkage long my_sys_open(char * filename, int flags, int mode){
    kprintk("The process is \"%s\"(pid is %i)\n",current->comm,current->pid);
    kprintk("The file is being accessed is \"%s\"\n",filename);
    return orig_open(filename,flags,mode);
}
```

```
}  
void start_hook(void){  
    g_sys_call_table = get_sys_call_table();  
    if(!g_sys_call_table){  
        kprintk("Get sys_call_table error!\n");  
        return;  
    }  
    if(g_sys_call_table[__NR_close] != (unsigned long)sys_close){  
        kprintk("Incorrect sys_call_table address!\n");  
        return;  
    }  
    g_old_sys_open = g_sys_call_table[__NR_open];  
    orig_open = (long (*)(char *, int, int))g_sys_call_table[__NR_open];  
    g_oldcr0=close_cr();  
    g_sys_call_table[__NR_open] = my_sys_open;  
    open_cr(g_oldcr0);  
}
```

恶意行为关联的操作程序有很多，主要有文件的创建，文件的删除，文件的读取和关闭及文件的重命名等，以上系统操作的功能实现都是利用系统内核的调用来完成的，该过程统称为系统调用。系统调用捕获技术利用应用系统的调用完成对应的数据截取，达到实施监控的目的，从而不断获取新的数据信息。

4.5 行为分析模块的设计与实现

4.5.1 系统调用数据抽取与存储

应用层数据中心负责存储所有受监控应用程序的静态和动态信息，静态信息包含应用的 UID 和包名标识，应用加入监控列表的最初时间等；动态信息包含应用的运行状态、安全级别，执行某一操作频率、携带数据等信息。针对这些数据信息，结合安卓系统本身的特性，采用安卓系统自带的 SQLite 数据库来存储这些应用数据信息。建立一张 app_information 表用于存储应用的基本信息，包含 UID、包名、应用安全级别、运行状态、监控起始时间和安全评价信息，详细定义见表 4.1。

对于每个受监控应用，我们根据应用的 UID 和包名组合创建一张名为 uid_pname 的表，表名称对于任何已安装的第三应用都是独一无二的，其中包含应用执行的操作，以及和操作相关的数据、频率等上下文信息，详细定义见表 4.2。

表 4.1 app_information 字段信息定义

表字段定义	字段描述	字段类型
ID	表的主键，应用的唯一标识。	integer
UID	应用在安装时系统分配的应用标识。	int
packageName	应用程序的包名， AndroidManifest 文件定义。	varchar
securityLevel	应用程序安全信息级别:分为高、中、低三个级别，对应数字 1,0, -1。	int
isBackground	应用程序运行状态,包含前后和后台两种状态,取值 0 和 1。	int
startTime	应用加入 AppSecure 应用监控列表的时间。	date
safetyEvaluation	针对应用的三种威胁行为给出安全评价信息。	varchar

表 4.2 uid_pname 字段信息定义

表字段定义	字段描述	字段类型
ID	表的主键，应用唯一标识。	integer
operation	应用执行的操作信息。	varchar
frequency	应用执行某一操作的频率。	float
data	应用执行某一操作使用的数据信息。	varchar
getPrivateDate	应用是否获取用户隐私数据，取值 1 和 0。	int

4. 5. 2 静态安全防御策略的具体实现

静态安全防御策略是在应用程序安装之前对权限列表的检查，实现主动防御功能。用户下载应用程序准备安装时，对待安装的应用程序进行检测，论文基于权限列表设定了 7 条静态主动防御策略，只有满足静态主动防御策略的应用程序才能够进一步安装，并进行接下来的动态安全评估，否则不允许安装并发出危险警告。为方便后期拓展，系统增加了添加策略、修改策略、删除策略等操作。核心代码如下：

```
public boolean onOptionsItemSelected(MenuItem item)
{
    intent intent = null;
    switch(item.getItemId())
    {
        case 1://添加策略
            intent = new Intent(this,newForm.class);
```

```
        ID = 0;
        startActivityForResult(intent,REQUEST);
        break;
    case 2://修改策略
        intent = new Intent(this,newForm.class);
        intent.putExtra("ID",ID);//获取当前选择行的值
        startActivityForResult(intent,REQUEST);
        break;
    case 3://删除策略
        bll.delete(ID);
        //刷新列表
        List<newModle> list = bll.selectAll(1);
        contactAdapter(contactAdapter);
        break;
    }
    return false;
}
```

4.5.3 动态安全评估策略的具体实现

(1) 权限攻击安全评估策略

在安卓环境下，用户的所有权限必须在 AndroidManifest 文件完成记录。考虑到这些设定是以记录文件为前提，当行为出现以后必须完成二次验证。例如，当监测到目标应用程序处于短信发送状态时，能够自动完成 AndroidManifest 的检查判定其短信发生权限是否存在。当未查询出指定记录时，则需要立刻拒绝操作请求然后通知手机用户。这样的操作不会因为具有 root 权限就不起作用。这种访问控制是完全强制按照 AndroidManifest 文件的信息来实现的。具体安全策略如下：

```
Translate operation to permissions
IF (permissions isn't in AndroidManifest) THEN (return false, record "权限不足", warn user Dangerous)
ELSE GO ON
```

执行需要权限申请的操作之前，系统会检查配置文件的权限列表中是否存在这一操作的权限声明，如果不存在，则报错并记录到日志；否则，继续执行下一步操作。

(2) 合谋攻击安全评估策略

以 SDCARD 举例进行合谋攻击一系列程序描述。SDCARD 是媒介使用率最高的一类。处于 SDCARD 层直接获取 SDCARD 的写权限就可以完成 SDCARD 对应操作。SDCARD 被看作是各个 APP 之间共享数据最佳选择点。为达到控制此类文件的目的，必须由程序启动开始就完成 SDCARD 文件的整体记录。在出现其他程序进入到系统进行再次访问的时候

可以作可疑标记，同时提醒使用者，具体安全策略如下：

```
UID operation data ON target context
IF (operation == network connect)
THEN (record “网络连接”；
IF (data.destination in Blacklist) THEN (warn user Dangerous)
IF (context.isBackground==true) THEN (warn user Normal)
IF (Gotten_privace_data==true) THEN (warn user Dangerous)
IF (operation==file operation) THEN (
IF (UID is not monitored app&& filename has been accessed by monitored app)
THEN (record “应用程序未被监控，文件被受监控程序访问”；
IF (frequency > limit) THEN (warn user Dangerous)
ELSE (warn user Normal)
IF (UID is monitored app) THEN (record the filename)
))
IF (operation == Binder transaction &&target is third-party app) THEN(
IF (Gotten_privace_data == true)
THEN (warn user Dangerous, record “第三方应用程序异常访问隐私数据”)
ELSE warn user Normal
ELSE GO ON
```

当应用程序执行连接网络操作时，如果同时出现传送数据至黑名单服务器、或者后台执行操作、或者获取隐私数据等行为均向用户发出危险警告。如果操作对象为文件系统，并且出现非处于监测状态的应用程序使用该文件或者对该文件的访问频率超过最高限制，则向用户发出危险警告。当系统异常并且应用程序来自第三方应用市场，这时如果访问隐私数据，也会向用户发出危险警告。否则继续执行下一步操作。

（3） 财务攻击安全评估策略

针对财务攻击，以短信劫持举例说明。短信劫持一方面必须完成 ActivityManagerService 服务的注册，目的是实现信号的接收。另一方面，还需要监控所有向 ActivityManagerService 注册的短信接收事件的行为，达到短信劫持监控的目的。发生注册短信接收事件后应在第一时间提醒手机操作者并警告。具体安全策略如下：

```
UID operation data ON target context
IF (operation ==sendSMS) THEN(
record “发送短信”；
IF (data.smsNUM in Blacklist) THEN (warn user Dangerous)
IF (context.isBackground ==true) THEN (warn user Dangerous)
IF (data.smsNUM not in contact) THEN (warn user Dangerous)
IF (frequency limit) THEN (warn user Dangerous))
IF (UID register SMS_RECEIVED &&target ==ActivityManagerService) THEN (warn
user dangerous)
ELSE GO ON
```


当执行发送短信操作时，首先记录“发送短信”到日志中，紧接着判断是否出现接收短信号码在用户设定的黑名单中、或者后台处理操作、或者执行频率超过限制等情况，若出现以上某一种情况则向用户发出危险警告。当应用程序申请接收短信服务，则立刻向用户发出危险警告，否则继续执行下一步操作。

(4) 隐私数据泄露安全评估策略

若想得到对应的 Android 系统非公开信息，软件软件必须首先申请 `com.Android.internal.telephony.IPhoneSubInfo` 这一请求。但若想得到联系人信息则必须申请 `Android.Content.IcontentProvider` 相关请求，通过解析应用软件的一系列权限申请，获取到其想要获取的具体的信息类型，再向手机用户发出危险警告。具体安全评估策略如下：

```
UID operation data ON target context
IF (target== IPhoneSubInfo) THEN (
record “获取系统隐私数据”;
IF (operation==IMEI) THEN (warn user Normal)
IF (operation==IMSI) THEN (warn user Normal)
IF (operation==phone_number) THEN (warn user Normal)
IF (context.isBackground ==true) THEN (warn user Dangerous)
set Gotten_privace_data = true;
)
IF (target == IContentProvider) THEN (
record “获取联系人信息”;
IF (operation == get_sms) THEN (warn user Normal)
IF (operation ==get_contacts) THEN (warn user Normal)
IF (context.isBackground ==true) THEN (warn user Dangerous)
set Gotten_privace_data =true
)
```

当应用程序想要获取系统隐私数据时，将 `target` 值定义为 `IPhoneSubInfo`，同时记录“获取系统隐私数据”到日志中，应用程序在获取系统隐私数据后继续访问 `IMEI`、`IMSI`、`phone_number` 等隐私数据，只能定义应用程序访问了隐私数据，不属于隐私泄露，所以提示用户运行正常。但如果是在后台操作隐私数据，即 `isBackground` 值为 `true` 时，向用户发出危险警告。当应用程序想要获取联系人信息时，将 `target` 值定义为 `IcontentProvider`，同时记录“获取联系人信息”到日志中，但如果是在后台操作隐私数据，即 `isBackground` 值为 `true` 时，向用户发出危险警告。

4.6 本章小结

本章阐述了安卓恶意软件动态检测方案的设计目标，论文将整个监测系统分为 3 个模块，分别是行为触发模块、行为监控模块、行为分析模块，分别给出了各个模块功能的具体实现方法。其中，重点描述了行为监控模块中内核初始化和系统调用捕获的实现过程。行为分析模块描述了系统调用数据的抽取，以及针对应用程序权限列表给出了静态安全防护策略，针对行为分析给出了应用程序动态安全评估策略。

第五章 实验与结果分析

在上一章节中，论文对安卓恶意软件监测系统框架进行了介绍，并且详细阐述了各功能模块的设计和实现。本章将在上一章的系统设计基础上，对所提出的安卓恶意软件监测方法的效果进行验证，同时对整个系统进行测试，分析最终的监测结果，验证论文所提出方案的适用性和有效性。

5.1 实验总体方案

论文实验在安卓模拟器上进行，采用 Linux 2.6.29 内核和安卓 4.3 系统框架，选取了来自著名研究社区 Malgenome Project^[50]的恶意样本。实验主要由以下几个部分组成：

第一步：PC 端和手机端的环境搭建。PC 端处理行为触发模块相关设置，手机端主要是内核环境的编译。

第二步：静态安全防御策略检测。环境搭建完成后，首先将样本依次进行静态安全防御策略检测，这个检测基于 4.5.2 安全策略编码实现，在 eclipse 环境下进行。

第三步：应用程序自动执行。通过第二步静态安全防御策略的应用程序才能够进行安装。采用 MonkeyRunner 依次安装并触发应用程序执行。

第四步：内核行为监控。应用程序启动后，论文 4.4.2 设计的自定义内核监测模块开始捕获并解析系统调用，并记录到行为日志中。

第五步：动态安全评估策略检测。系统调用解析完成后，依次进行论文 4.5.3 制定的权限攻击、合谋攻击、财务攻击和隐私数据泄露四个安全评估检测，最终判定应用程序攻击类型。

5.2 实验环境配置

实验采用的 PC 端为 windows 7（配置信息表 5.1），手机端为安卓模拟器（Android 4.3、Linux 2.6.29），模拟器配置信息如图 5.1 所示。之所以选择模拟器来实现，主要是因为模拟的源代码比较容易得到。

表 5.1 PC 端配置环境

项目	配置
操作系统	Windows7
运行环境	JDK1.8
CPU	Inter(R)Core(TM)i3 2.27GHZ
内存	4G
硬盘	300G

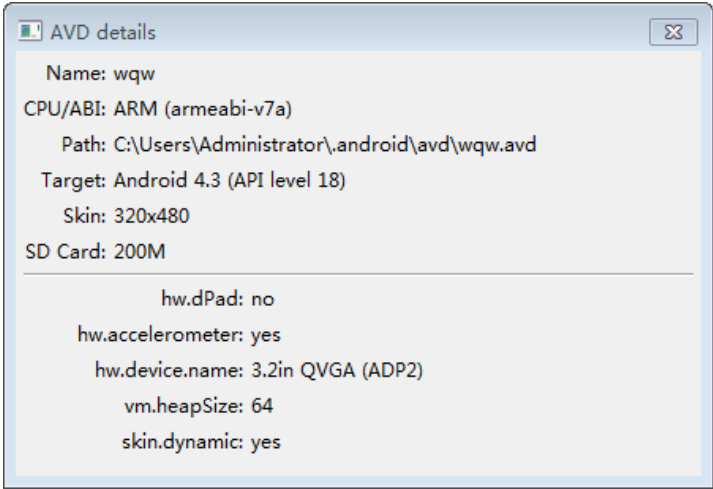


图 5.1 模拟器配置信息

5.3 实验步骤

内核编译环境搭建：

实验环境搭建第一步：编译内核。

在 Linux 内核中，所有的系统调用都存在于 sys_call_table 这样一张表中，为了能够拦截相关的系统调用，需要将这个表中的特定调用地址指向自己设计的函数，或者直接修改此表。

修改系统调用表最核心的部分就是对内核进行编译。编译的步骤如下：

(1) 下载安卓内核源码

执行如下命令，获取内核源码的主分支。

```
git clone git : // android.git.kernel.org /kernel/common.git
```

(2) 设定环境变量

需要使用交叉编译器。修改用户目录下的.bashrc 文件，添加如下代码：

```
export  
PATH=$PATH:~/Android/source/prebuilt/linux-x86/toolchain/arm-eabi-4.4.0/bin  
export ARCH=arm
```

(3) 设定交叉编译参数

打开 makefile 文件, 该文件位于 kernel 目录下。将 CROSS_COMPILE 指向 arm-eabi 编译器。

(4) 配置 config 文件

获取模拟器正在使用的 config 文件。首先要启动 AVD, 执行 `adb pull /proc/config.gz` 命令, 用来获取模拟器中包含的内核配置文件, 接着解压 config.gz 文件。

(5) 编译内核

首先需要定位到内核目录, 然后使用 `make` 命令对内核执行编译操作。编译完成后, 会生成新的内核镜像名为 `zImage`, 该文件位于 `/kernelarch/arm/boot/` 目录下。

(6) 修改安卓的系统调用表

设定新函数地址 `new_openat` 用来替换 `sys_call_table` 中的原始函数 `openat` 的调用地址。

下面只需要在启动模拟器时, 加载新镜像即可。实验过程如下:

安装好 android 的 SDK, 配置好开发环境 Eclipse, 用 eclipse 打开 android 模拟器 (如图 5.2) 或者在 CMD 中用 `emulator` 命令打开模拟器如下图 5.3 所示:

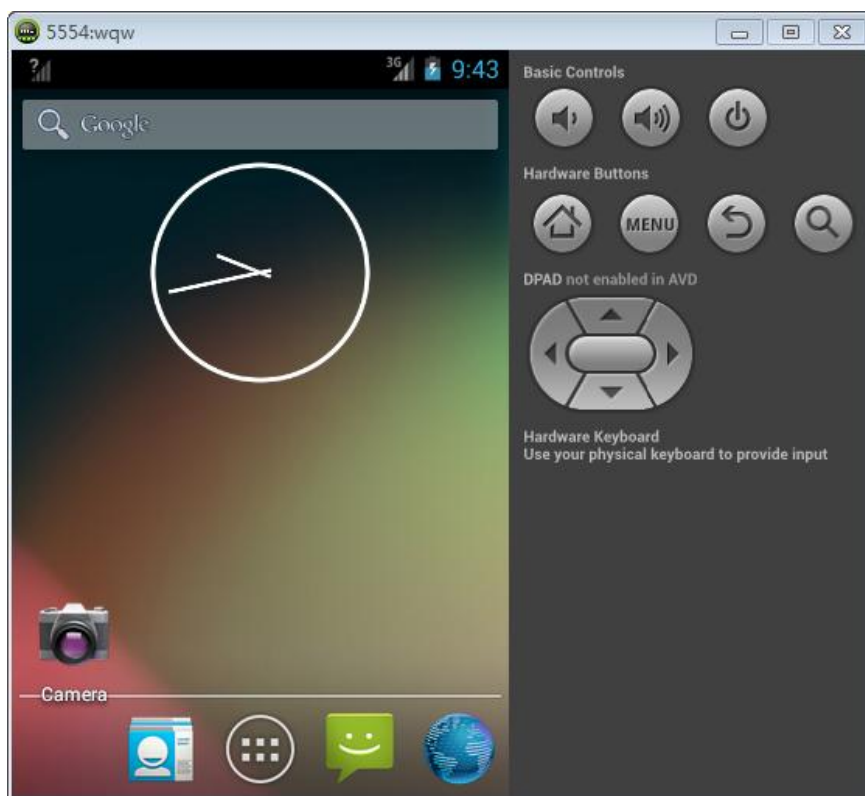


图 5.2 安卓模拟器



图 5.3 emulator 命令打开模拟器

运行 monkeyrunner 之前必须先运行相应的模拟器，不然 monkeyrunner 无法连接设备。打开另一个 CMD 窗口，定位到 tools 目录，输入命令“monkeyrunner”，进入 shell 命令交互模式。出现如下图 5.4 界面说明与模拟器连接成功，自动化测试环境配置完成。



图 5.4 monkeyrunner 与模拟器连接成功

在 Linux 内核 2.6 之后，不能直接导出 sys_call_table 的地址，首先要获取系统调用表的地址（如图 5.5 所示），从而实现系统调用的截获。

```
[ 903.537218] Monitor init
[ 903.537224] address of idtr: 0xdec31e5e
[ 903.537225] address of idt 0x80: 0xc168b328
[ 903.537227] address of sys_call_table:0xc1697140
```

图 5.5 系统调用表地址

获取了系统调用表的地址后，接下来进行系统调用的截获工作。所谓截获就是将系统调用表里的地址指向我们自己写的一个函数，系统调用先执行我们自己写的函数，处理完后，再返回原来系统调用的执行函数。

内核编译完成后，执行 sudo insmod 命令完成内核的加载，加载成功后执行 dmesg 命令查询系统日志，如图 5.6 所示表明已完成内核编译工作，行为监控环境搭建完成。

```
[ 5385.646568] The process is "vminfo"(pid is 1000)
[ 5385.646575] The file is being accessed is "/var/run/utmp"
[ 5385.648792] The process is "vminfo"(pid is 1000)
[ 5385.648799] The file is being accessed is "/etc/passwd"
[ 5386.318082] The process is "dmesg"(pid is 3557)
[ 5386.318087] The file is being accessed is "/etc/ld.so.cache"
[ 5386.318119] The process is "dmesg"(pid is 3557)
[ 5386.318121] The file is being accessed is "/lib/i386-linux-gnu/libc.so.6"
[ 5386.318384] The process is "dmesg"(pid is 3557)
[ 5386.318388] The file is being accessed is "/usr/lib/locale/locale-archive"
[ 5386.318577] The process is "dmesg"(pid id 3557)
[ 5386.318579] The file is being accessed is "/usr/lib/locale/locale-archive"
```

图 5.6 查询系统日志

5.4 系统测试与结果分析

实验中恶意软件来源于 Malgenome Project, 数据集集中的 1200 个恶意样本均已根据它们所产生的恶意行为的不同特点进行了归类。如下图 5.7 所示, 文件名为 APK 的 MD5^[51]值。以测试用例 2da50baf74dc43a5ffd891f202cfd011.apk 和 4ae1dcdbebbbb281cd4ea021d27b72b04.apk 举例分析。

2da50baf74dc43a5ffd891f202cfd011.apk 在实验中未通过静态安全防御检测, 所以安装被阻止。通过分析权限申请列表, 发现其同时申请了语音记录权限、使用电话状态权限和连接互联网权限, 存在监听用户通话内容并实时上传到指定服务器恶意行为, 违反了论文制定的静态安全防御策略第 4 条。

4ae1dcdbebbbb281cd4ea021d27b72b04.apk 在动态安全评估检测中被判定为财务攻击类型的应用程序。通过分析其行为发现, 此应用程序在启动时, 首先执行用户手机号码读取操作, 然后将获得的信息进行记录, 接下来将相关信息发送到指定的服务器上, 在用户不知情的情况下, 订阅一些收费服务。另外, 通过行为分析发现它会对来自 99735、55991 和 96512 等短信进行拦截, 并将内容发送给服务器。当短信来自 36341 的时候, 它会自动回复一条 Yes 的短信。这样, 此应用程序在用户毫不知情的情况下, 成功订阅了收费服务。




















 0be0a939037c373e6576f84d8a979800.apk	2015/5/10 5:23	APK 文件
 0fab7e8605892bcc23abdbce09c6f8a3.apk	2015/5/10 5:23	APK 文件
 0ff7368a21de052a0fd670ffdbbacdbe.apk	2015/5/10 5:02	APK 文件
 1ca231169ab226ceab0f5c71450097df.apk	2015/5/10 5:20	APK 文件
 02a2ba8174893890cf213152a6b190f1.apk	2015/5/10 5:25	APK 文件
 2a047b5b16a9a7e745727aacd5bebc7d45540c05.apk	2015/5/10 5:13	APK 文件
 2da50baf74dc43a5ffd891f202cfd011.apk	2015/5/10 5:08	APK 文件
 3ad372b90759ac77fe9285b53a2119c8.apk	2015/5/10 5:04	APK 文件
 3d84d7e98d5fcd308dcc418bad5b08bf.apk	2015/5/10 5:23	APK 文件
 3e597a070e81d50f92dfa75261f2e9f9.apk	2015/5/10 5:24	APK 文件
 3eea7a9bdeba1c6de34dc79de831784c.apk	2015/5/10 5:20	APK 文件
 4ae1dcdbebbb281cd4ea021d27b72b04.apk	2015/5/10 5:07	APK 文件
 4bf221a78836abfe9f30695f572a4f3a.apk	2015/5/10 5:19	APK 文件
 4f2e52de076cbfe6ca5e6b20df7d5467.apk	2015/5/10 5:23	APK 文件
 5a0cc3307f7cff2852ce20d1d106ca87.apk	2015/5/10 5:17	APK 文件
 5f292bf455633d8c2f30be974a91b328.apk	2015/5/10 4:57	APK 文件
 06e481f10da2a9552dd5c9ee4525f233.apk	2015/5/10 5:26	APK 文件
 6ae7b0d04e2fd64a50703910d0eff9cc.apk	2015/5/10 5:21	APK 文件
 6b0fa323d01fb7c363c9fcb9948812a1.apk	2015/5/10 5:25	APK 文件

图 5.7 恶意软件样本

实验监测得出恶意应用权限泄露 411 个、合谋攻击 8 个、财务攻击 357 个、隐私数据窃取 127 个，其他攻击类型的应用 19 个。从实验结果来看，基于内核层修改系统调用表这一监控方式能够监测出所有的攻击类型，证明这一方法能够全面地监测恶意软件行为。但是有约 1.6% 的恶意样本未能监测到具体执行了哪种恶意行为，这是因为存在新的恶意软件类别，这也是论文安全策略定义模块需要完善的地方。

第六章 总结与展望

6.1 论文工作总结

近年来安卓移动设备取得了较快发展。不管是家居还是手机都与人们的日常生活息息相关。当安卓移动设备越来越接近人们生活的时候，其表现出来的安全模式也渐渐进入人们的视野，不管是官方市场还是第三方市场，恶意软件的足迹不断出现。而出现的这些恶意软件对于使用者来讲带来了很多不便，严重的情况下会导致财产受到侵犯。一些应用软件能够通过系统形成的漏洞直接取得对应的操作权限，达到操控整体系统的目的。还有一些应用软件嵌入恶意攻击代码达到用户隐私数据的窃取等目的。同时某些应用程序通过相互之间的合作共同达到恶意攻击的目的。总的来说，针对安卓系统的恶意攻击是全方位的。

论文主要工作如下：

研究了安卓系统架构、安卓系统四大组件以及安卓安全机制中数字签名、沙箱机制以及权限控制机制，分析了安卓系统的安全问题，研究了安卓系统各安全架构层监测机制。分析了应用程序层监测机制、应用程序框架层监测机制、内核层监测机制的原理和优缺点。

着重研究了基于内核的行为监测方法和应用程序安全评估方法。主要研讨了基于内核的行为监测中内核的初始化问题以及如何使用 **Binder** 解析系统调用这一问题。研究了基于上下文信息的评估方法，提出了一种安全评估策略表示方法，针对权限泄露、合谋攻击、财务攻击、隐私数据窃取这四种恶意攻击类型制定了详细的安全评估策略，用来判定恶意应用程序的攻击类型。

设计了一个基于安卓系统的恶意软件的动态监测方案。通过实验测试对论文所提出的安卓恶意软件动态监测方案，分析最终的监测结果，验证论文所提出方案的适用性和有效性。

6.2 进一步工作和展望

论文提出了一个基于安卓系统的恶意软件的动态监测方案，并对此方案进行设计和实现，测试结果表明该方案能够很好地对安卓恶意软件行为进行监测。但是由于时间关系，上述工作还存在一些不足之处，需要进一步研究和完善。

- (1) 本方案对权限泄露、合谋攻击、财务攻击、隐私数据窃取给出了具体的安全策略，对于其他类型攻击未能给出详细安全策略，需要进一步研究。
- (2) 本方案覆盖的内核版本和系统版本比较有限，目前只在安卓模拟器上做了实验。真机的内核源码获取较困难，有待进一步改善。

参考文献

- [1]林城. Android23 应用开发实战[M].北京:机械工业出版社,2011.
- [2]中国反网络病毒联盟[EB/OL].<http://www.anva.org.cn>,2013-3-25.
- [3] Shabtai A. Malware Detection on Mobile Devices[C]// Eleventh International Conference on Mobile Data Management. IEEE Computer Society, 2010: 289-290.
- [4] Schmidt A D, Bye R, Schmidt H G, et al. Static Analysis of Executables for Collaborative Malware Detection on Android[C]//IEEE International Conference on Communications IEEE, 2009: 1-5.
- [5] 李佳.Android 平台恶意软件检测评估技术研究[D],北京邮电大学,2012.
- [6] 王菲飞.基于 Android 平台的手机恶意代码检测与防护技术研究[D].北京交通大学,2012.
- [7]Hu G, Venugopal D.A Malware Signature Extraction and Detection Method Applied to Mobile Networks[C]//Performance, Computing, and Communications Conference, 2007. IPCCC 2007 IEEE International. IEEE,2007:1926.
- [8] Xu R, Anderson R. Aurasium: practical policy enforcement for Android applications[c]// Usenix Conference on Security Symposium. USENIX Association, 2012: 27-27.
- [9] Enck W, Gilbert P, Chun B G, et al. TaintDroid: an information flow tracking system for real-time privacy monitoring on smartphones[J]. Acn Transactions on Computer Systems, 2014, 32(2): 1-29.
- [10] Burguera I, Zurutuza U, Nadjm-Tehrani S. Crowdroid: behavior-based malware detection system for Android[c]// ACM Workshop on Security and Privacy in Smartphones and Mobile Devices. ACM, 2011: 15-26.
- [11]胡伟. Android 系统架构及其驱动研究[J]. 广州广播电视大学学报,2010,10[4]:96-101.
- [12] Shu X, Du Z, Chen R. Research on mobile location service design based on Android[C]//wireless Communications, Networking and Mobile Computing, 2009. Wicom'09.5th International Conference on. IEEE 2009: 1-4.
- [13]Bose A, Hu X, Shin K G et al. Behavioral detection of malware on mobile handsets[C]. Proceedings of the 6th international conference on Mobile systems, applications, and services. ACM. 2008: 225-238.
- [14] Jacob G Debar H, Filiol E. Behavioral detection of malware: from a survey towards an established taxonomy[J]. Journal in computer Virology. 2008. 4(3): 251-266.
- [15]Dai S, Liu Y, Wang T, et al. Behavior-based malware detection on mobile phone[C]. Wireless Communications Networking and Mobile Computing (WICOM).2010 6th International Conference on. IEEE.2010: 1-4.
- [16]Grace M, Zhou Y, Wang Z, et al. Systematic detection of capability leaks in stock Android smartphones[C]. Proceedings of the 19th Annual Symposium on Network and Distributed System Security. 2012.
- [17]吴俏. Android 安全机制解析与应用实践[M].机械工业出版社,2013.
- [18]N. Idika, A. Mathur. A Survey of Malware Detection Techniques, Tech. Rep. SERC-TR-286[R]. Department of Computer Science of Purdue University, West Lafayette,USA,2007:7-8.
- [19] Google Inc. Android Developer Documents android-app [EB/OL]. [2011-12-30]. <http://developer.android.com/reference/android/app/package-summary.html>.
- [20]Google Inc. Android Developer Documents android -content [EB/OL]. [2011-12-30]. <http://developer.android.com/reference/android/content/package-summary.html>.
- [21]柯元旦. Android 内核剖析[M]// 电子工业出版社,2011.
- [22] android-x86 Project - Run Android on Your PC[EB/OL]. <http://www.android-x86org/>, 2014-05-17.
- [23]Jeff Six. Application Security for the Android Platform. Published by O Reilly Media. 2011. 12
- [24] Min Zheng, Patrick P.C. Lee, John C.S. Lui. ADAM: An Automatic and Extensible Platform to Stress

- Test Android Anti-Virus Systems[C]. Proceedings of the 9th International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment. Berlin Heidelberg: Springer-Verlag, 2012: 82-101.
- [25]Wu Zhou, Yajin Zhou, Xuxian Jiang, etc. Detecting Repackaged Smartphone Applications in Third-party Android Marketplaces[C]. Proceedings of the second ACM conference on Data and Application Security and Privacy. New York: ACM Press, 2012.317-326.
- [26] Yajin Zhou, Xuxian Jiang. Dissecting Android Malware: Characterization and Evolution[C]. Proceedings of the 2012 IEEE Symposium on Security and Privacy. USA:IEEE Computer Society Press. 2012: 95-109.
- [27]廖明华,郑力明. Android 安全机制分析与解决方案初探[J]科学技术与工程,2011,11(26):6350-6355.
- [28]李嘉周.资讯类 Android 应用开发框架的设计与实现[D].北京邮电大学,2015.
- [29]Nyman N. Using Monkey Test Tools Monkey testing refers to automated testing done randomly without any"typical user" bias. here's a look at how to use such random testing techniques to cost-effectively catch bugs you might otherwise miss[J]. Software Testing and Quality Engineering. 2000, 2: 18-23.
- [30]兰娅勋, 李振坤. MonkeyRunner 环境用例脚本化的 Android 软件测试模型[J]. 科技通报, 2017, 33(9).
- [31]Dostal M, Eichler Z. A Hybrid Approach to User Activity Instrumentation in Software Applications[M]// HCI International 2011 – Posters' Extended Abstracts. Springer Berlin Heidelberg, 2011:566-570.
- [32]蒋崇武, 刘斌, 王轶辰,等. 基于 Python 的实时嵌入式软件测试脚本[J]. 计算机工程, 2009, 35(15):64-66.
- [33]杨怡君, 黄大庆. Android 手机自动化性能测试工具的研究与开发[J]. 计算机应用, 2012, 32(2):554-556.
- [34]Wolfson M. Android Developer Tools Essentials: Android Studio to Zipalign[C]// O'Reilly Media, Inc. 2013.
- [35]马红素, 郭燕慧. Android 应用自动化动态测试工具的研究及实现[J]. 2012.
- [36] Sridharan M, Artzi S, Pistoia M, et al. F4F: taint analysis of framework-based web applications[J]. Acm Sigplan notices,2011,46(10):1053-1068.
- [37]杨广亮,龚晓锐,姚刚,等.一个面向 Android 的隐私泄露检测系统[J]计算机工程,2012,38(23):1-6.
- [38] Sandeep s. Process tracing using ptrace[J]. Linux Gazette, 2002, 8(81): 32-35.
- [39] Six J. Application Security for the Android Platform[J]. Oreilly Media, 2011, 9(4): 187-198.
- [40]xu R, Anderson R. Aurasium: practical policy enforcement for Android applications[C]// Usenix Conference on Security Symposium. USENIX Association, 2012: 27-27.
- [41]鲁慕瑶,邓芳 Linux 系统内核调用分析[J].湖北第二师范学院学报,2011(8):77-79.
- [42]Love R. Are S H W. Linus A C, et al. Linux Kernel Development Second Edition[M]. Novell Press: Sams Publishing, 2005.
- [43]张步忠,金海平. Linux 内核系统调用扩展研究[J].计算机技术与发展,2007,17(5):163-165.
- [44]Chiang H S, Tsaur W J. Mobile Malware Behavioral Analysis and Preventive Strategy Using Ontology[C]//
- [45]Talha K A, Alper D I, Aydin C. APK Auditor: Permission-based Android malware detection system[J]. Digital Investigation, 2015, 13:1-14. IEEE Second International Conference on Social Computing. IEEE Computer Society, 2010:1080-1085.
- [46] S. Bugiel, L. Davi, A. Dmitrienko, T. Fischer, A- R. Sadeghi, and B.Shastry. :Towards taming privilege-escalation attacks on Android. In 19th Network and Distributed System Security Symposium(NDSS 12), San Diego, CA, Feb.2012.
- [47]L. Davi, A. Dmitrienko, A.-R Sadeghi, and M. Winandy. Privilege escalation attacks on Android. In 13th Information Security Conference(ISC), 2010.
- [48] Yibing Zhongyang, Zhi Xin, Bing Mao, Li Xie: Droidalarm: an all-sided static analysis tool for Android privilege-escalation malware ASIACCS 2013: 353-358.

-
- [49]Bhattacharya A, Goswami R T. Comparative Analysis of Different Feature Ranking Techniques in Data Mining-Based Android Malware Detection[J]. 2017.
- [50]Zhou Y, Jiang X. Dissecting Android Malware: Characterization and Evolution[c]//IEEE Symposium on Security and Privacy. IEEE Computer Society, 2012: 95-109.
- [51] RIVEST R. The MD5 message-digest algorithm[J]. Internet Request For Comments 1321, 1992, 473(10):492-492.
- [52]李静. Android 中 Binder 机制研究与应用[J]. 工业控制计算机, 2012, 25(4):66-67.

附录 1 攻读硕士学位期间撰写的论文

(1) 王倩文、沈苏彬、吴镇宇, 基于 Android 平台的恶意软件动态监测的研究, 计算机技术与发展, 已录用。

附录 2 攻读硕士学位期间参加的科研项目

- (1) 江苏省科技厅（未来网络前瞻性研究项目） 支持 SDN 的未来网络架构及实现技术研究（BY20130951108）；
- (2) 国家自然科学基金（No. 61502246）。

致谢

一寸光阴一寸金，寸金难买寸光阴。转眼间，我的三年研究生学习进入了尾声。百感交集之际，回想这些年的点点滴滴，在各位教授和老师的教导下，同学们的帮助和鼓励下我逐渐成长。

在此我要对我的导师沈苏彬老师表达最衷心的感谢，在沈老师的悉心指导以及大力帮助之下我才得以完成这篇毕业论文，沈老师有着全面系统的专业知识以及辽阔的视野，这让我深深折服。同时耐心且细致的教导方式也让我的毕业论文得以逐渐完善，这对于我未来的工作有着深刻的意义。

在此感谢所有帮助过我老师以及全体教务工作者，感谢毛燕琴老师、吴镇宇老师，正是他们的以身作则和谆谆教导，让我不断进步。

感谢共同努力奋斗的同窗好友，我们互相勉励，共同进步，让我感觉到校园生活的丰富多彩。激烈的讨论的场面还历历在目。在今后的人生中，希望我们都能够不忘初心。

最后感谢我的父母，是他们的鼓励和帮助让我有了现在的成绩。