第三章 实现方案

本章主要介绍 Android 重打包应用程序安全策略加固系统的具体实现，包括各模块设计思路以及所用到的关键技术，同时针对该加固系统提出一种可行的部署方式。

3.1 安全策略加固系统设计

3.1.1 系统功能

本系统主要针对 Android 权限机制存在的一些漏洞，包括系统权限划分的粗粒度以及权限授权一次性等所带来的安全隐患。利用重打包技术实现对应用程序的安全策略加固，该系统主要包括以下功能：

(1) 对于安装后的应用，用户可以根据需要对应用程序的权限进行动态管理，从而实现对程序可疑行为的可控监管；

(2) 针对应用的可疑行为，系统都会进行日志记录。以便日后用户能够根据日志记录，实施更为有效的安全加固策略；

(3) 根据应用程序权限列表提供更细粒度安全策略，完成不同用户对应用程序的不同的加固需求。

3.1.2 系统整体框架

该系统主要分为 3 个模块，安全加固模块，手机端监控/管理模块，服务端重打包模块，各个模块相互协调构成整个安全策略加固系统。下面将对系统各个模块进行介绍：

(1) 安全加固模块

该模块即所要嵌入的安全加固代码，根据可疑权限划分为以下类别：

隐私类：包括录音、摄像头、查看短信、查看通讯录、获取位置信息；

扣费类：包括发短信、打电话；

拦截类：包括短信过滤、电话过滤等行为；

启动类：开机自启，后台联网自启；

该模块针对应用程序的这几类可疑行为进行监控，同时各类可疑行为触发时所传递的数据内容也会被该模块捕获。

(2) 手机端监控/管理模块

该模块位于手机上，用来管理经过打包处理的加固应用。主要负责与加固程序的通信，辅助加固程序安全策略的施行。同时是用户策略定制的主入口，主要体现为用户对权限的自主控制，以及对特定应用程序的日志审计等。

(3) 服务端重打包模块

该模块完成代码嵌入，打包等功能，主要涉及到对 AndroidManifest.xml 的解析、dex 的逆向分析、核心代码注入、apk 的重组、签名等，是整个系统流程的关键点，使用 python 语言自动化实现该流程。

系统整体框架如下图 3-1 所示：

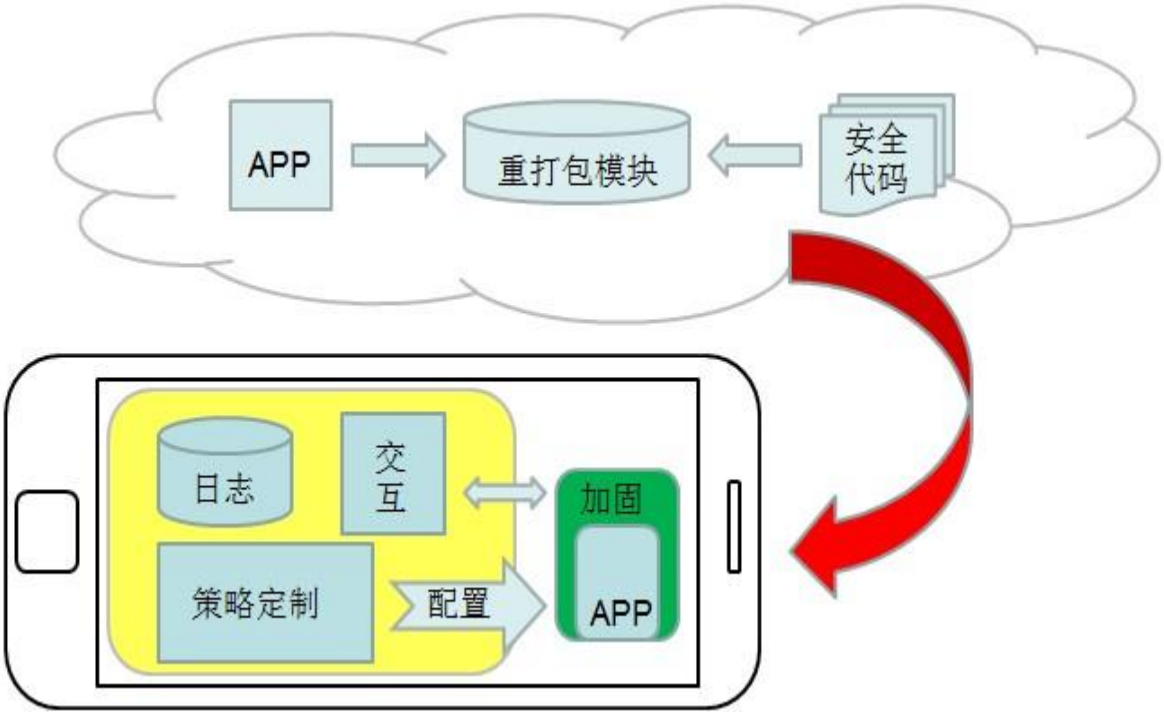

图 3-1 系统整体框架

3.1.3 系统执行流程

该系统按运行平台划分可分为 2 大块，一块位于服务端的重打包模块，一块位于手机运行的手机管理模块。两大块可根据网或者 Usb 数据线进行通信，用来解决待处

理 APP 的上传以及加固 APP 的下载功能。该系统执行流程主要分为以下步骤：

(1) 若为需要加固的 APP，则通过 Usb 或者网络上传到服务端重打包模块。

(2) 服务端执行由 python 语言编写的重打包模块，对上传 APP 进行解包、安全代码注入、重新打包等过程，完成对 APP 的安全策略加固。

(3) 加固 APP 安装到手机后，可以根据手机端管理模块，对加固 APP 进行管理，包括权限策略定制，日志查看等等。

(4) 在加固 APP 运行的过程中，一旦触发可疑行为将会与手机端管理模块进行通信，对其行为进行决策，同时也将该行为具体信息通知用户并进行日志记录。

下图 3-2 展示了系统的执行流程：

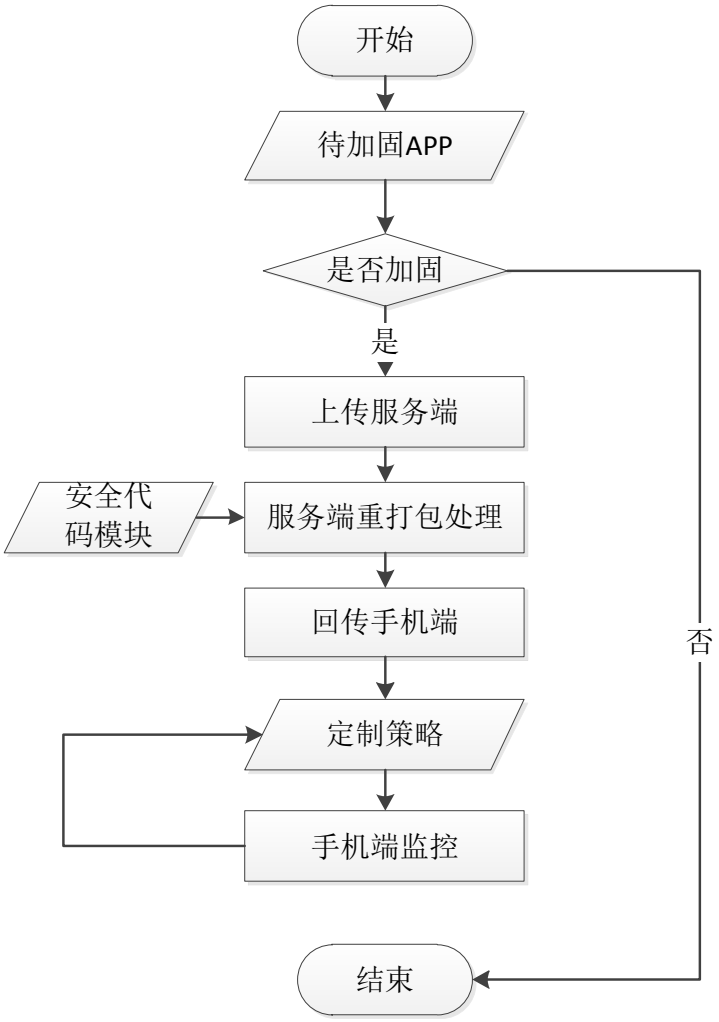

图 3-2 系统执行流程

3.2 系统模块关键技术

整个系统的设计根据功能的不同，划分为不同模块，其所采用的技术以及测试的侧重点也将有所不同，下文将对其各个模块进行阐述。

3.2.1 细粒度权限控制

由于 Android 应用申请的权限与程序功能有很大的关联性，因此我们可以根据应用在 AndroidManifest.xml 文件中所声明的常用权限，并结合恶意代码恶意行为常用的权限特征，将权限划分为 5 大类，以作为安全策略实施的凭据，如图 3-3 展示了该系统对权限行为的分类：

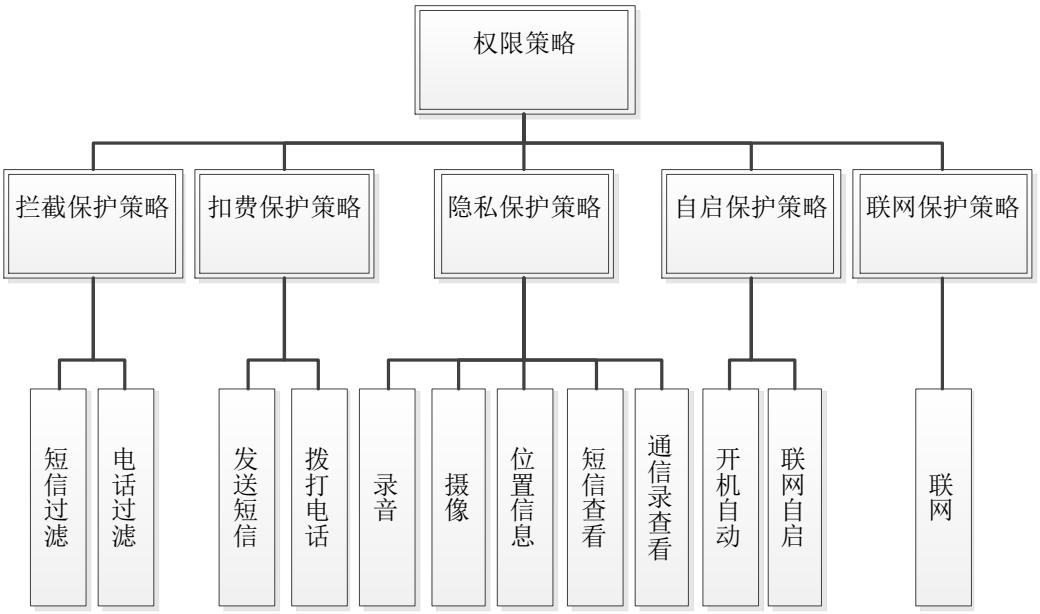

图 3-3 权限策略分类

(1) 隐私保护监管权限

```
<uses-permission android:name="android.permission.RECORD_AUDIO"/>
<uses-permission android:name="android.permission.CAMERA"/>
<uses-permission android:name="android.permission.READ_SMS"/>
<uses-permission android:name="android.permission.READ_CONTACTS"/>
<uses-permission
```

android:name="android.permission.ACCESS_FINE_LOCATION"/>

(2) 扣费监管权限

<uses-permission android:name="android.permission.SEND_SMS"/>

<uses-permission android name="android.permission.CALL_PHONE">

(3) 拦截监管权限

<uses-permission android:name="android.permission.RECEIVE_SMS" />

<uses-permission android:name="android.permission.READ_PHONE_STATE" />

(4) 自启动监管权限

<uses-permission

android:name="android.permission.ACCESS_NETWORK_STATE"/>

<uses-permission

android:name="android.permission.RECEIVE_BOOT_COMPLETED"/>

(5) 联网监管权限

<uses-permission android:name="android.permission.INTERNET"/>

以上权限也正是在重打包过程中解析 AndroidManifest.xml 的原因所在，根据应用声明的权限，对其进行解析。从而嵌入不同的安全代码。

3.2.2 安全加固模块

安全加固模块是该系统的核心模块，主要包括应用程序可疑行为的监控等。该安全模块的策略主要根据现有恶意代码行为分析而设计的。因此对恶意代码的行为分析是该模块稳固健全的关键因素。本文事先人工分析了一些恶意代码典型的恶意行为作为定制策略的基础并结合恶意代码所申请的应用权限，制作成了如下权限函数映射表 3-1，以用于安全加固代码实现的依据。

表 3-1 权限函数映射

权限名称	映射函数
android.permission.READ_SMS	Landroid/content/ContentResolver;->query
android.permission.READ_CONTACTS	
android.permission.RECORD_AUDIO	Landroid/media/MediaRecord

	er;->start Landroid/media/MediaRecorder;->stop
android.permission.CAMERA	Landroid/hardware/Camera;->open
android.permission.ACCESS_FINE_LOCATION	Landroid/location/LocationManager;->getLastKnownLocation
android.permission.SEND_SMS	Landroid/telephony/SmsManager;->sendDataMessage Landroid/telephony/SmsManager;->sendTextMessage Landroid/telephony/SmsManager;->sendMultipartTextMessage
android.permission.CALL_PHONE	Landroid/content/Context;->startActivity
android.permission.RECEIVE_SMS	Landroid/content/Context;->registerReceiver Landroid/content/Context;->unregisterReceiver
android.permission.READ_PHONE_STATE	
android.permission.INTERNET	Ljava/net/URL;->openConnection

该模块针对每个调用函数都进行了二次封装，以便重打包后该应用所触发的行为事先会被加固安全模块捕获，每次封装主要结合了各种行为的特征，下面主要介绍对 Landroid/content/ContentResolver;->query 的封装（查看短信，查看通信录、通话记录等 content provider 查询行为）以及对动态注册广播接收器 Landroid/content/Context;->registerReceiver 的封装（短信拦截，电话拦截等恶意动态广播行为）。

1. Landroid/content/ContentResolver;->query 封装

上文提到了 Android 应用组件 content provider，它以 URI 来标识目标数据。而 content resolver 是对其提供接口的一个封装，用来间接调用 content provider 提供的数据操作方法。短信的查询，通信录查看等操作正是通过查询共享数据的方式获取数据信息的，数据的不同在于 URI 的不同。常用的关键 URL 字符串开头如下：

通讯录：

content://contacts

content://com.android.contacts

content://icc/and（读取 SIM 卡联系人）

短消息（短信/彩信）：

content://sms

content://mms

通话记录：

CallLog.Calls.CONTENT_URI（系统定义）

加固代码中关于 query 函数的二次封装就是对关键 URI 进行匹配，如果满足以上特征则判断为可疑行为。此时加固应用会连接手机端检测模块进行行为认证，根据返回的决策来决定是否禁止此次 query 行为。

2. Landroid/content/Context;->registerReceiver 封装

与该函数特性相关的是 Android 应用组件 broadcast receiver。广播接收者注册方式分为静态注册和动态注册。动态注册广播接收具有比静态注册接收者接更高的优先级。恶意代码常采用动态注册并设置最高优先级的方式，截获系统广播以实现其恶意行为。该函数原型如下：

Intent registerReceiver (BroadcastReceiver receiver, IntentFilter filter)。在动态注册中，IntentFilter 为需要接收的广播行为（Action）。

系统中如下广播行为是恶意代码最感兴趣的：

android.intent.action.PHONE_STATE （获取电话状态，常用来截获电话信息）

android.provider.Telephony.SMS_RECEIVED （系统接收到短信时触发）

加固代码中关于 registerReceiver 的封装正是对这两个关键广播行为进行监听，若代码需要注册以上两个行为的广播接收者，将此行为上报手机端监控模块，对其进行行为认证，从而实现对该行为的监控。

以上所提到的加固代码为一些 smali 文件集合，每个 smali 文件对应一个类，该类下的方法正是对以上权限函数映射表中函数的一个封装。在重打包过程中将会对源程序调用关系进行修改，使得程序能运行至我们封装后的函数中，以此实现对可疑行为关键函数的行为捕获。

3.2.3 手机端监控/管理模块

手机端监控/管理模块主要负责与加固应用交互，完成用户安全策略的实施与行为日志的记录。该模块主要考虑的是与加固应用交互的稳定性，策略更新的时效性。由于其需要匹配安全策略，故需要一个快速而稳定的运行环境。该模块功能结构图如图 3-4 所示：

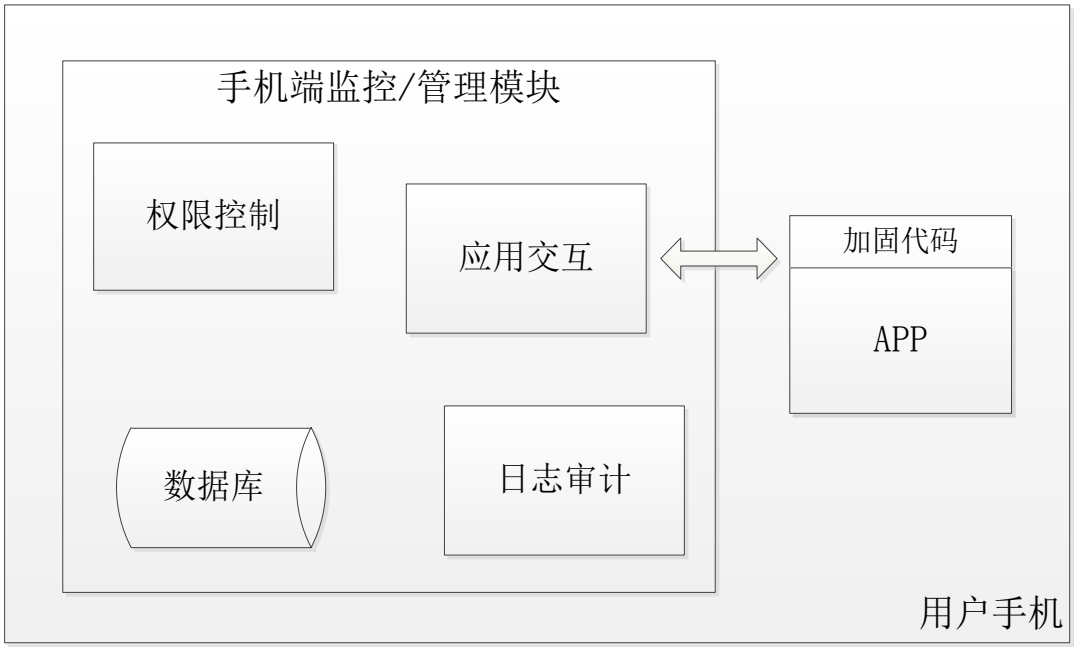

图 3-4 手机端监控/管理模块功能结构

该手机端模块根据功能可继续划分为 3 个功能模块，下文将对其子功能模块进行介绍：

1. 权限控制

为了能够记住用户上次实施的权限控制策略，该子模块采用 SQLite 用于存储用户之前配置的权限策略。同时为了保证程序的快速稳定，在内存中采用 hashmap 数据结构对其进行存储。创建该权限控制存储结构实例语句如下：

HashMap<String, Integer> policyMap = new HashMap<String, Integer>();

String 是加固应用程序包名，Integer 是加固应用权限状态集合（采用置位表示权限的启用，复位表示权限的禁止），各权限状态如下（对应于 Integer 各比特位）：

```
public static final int READ_SMS =                  1;
public static final int SEND_SMS =                  2;
public static final int RECEIVE_SMS =               4;
public static final int CALL_PHONE =                8;
public static final int READ_PHONE_STATE =          16;
public static final int READ_CONTACTS =             32;
public static final int INTERNET =                  64;
public static final int ACCESS_NETWORK_STATE =      128;
public static final int RECORD_AUDIO =              256;
public static final int ACCESS_FINE_LOCATION =      512;
public static final int CAMERA =                    1024;
public static final int RECEIVE_BOOT_COMPLETED =    2048;
```

对权限进行限制：

```
PolicyMap.policyMap.put(packageName,
PolicyMap.policyMap.get(packageName)^permState);//对应权限位置零
```

对权限进行启用：

```
PolicyMap.policyMap.put(packageName,
PolicyMap.policyMap.get(packageName)|permState);//对应权限位置一
```

在设置应用权限控制完成，并退出其对应 Activity 之后，权限 hashmap 会存入数据库中，以便保持用户决策。

2. 推荐策略

为了使用户较为便捷够有效的使用动态权限管理，我们提供了推荐策略模块。根据各个软件功能，我们将软件划分为 8 大类，办公商务类、壁纸主题类、聊天通讯类、生活地图类、网络邮件类、影音图像类、游戏类、阅读学习类。通过采集 800 个 APK 样本（每个类别 100 个），绘制出了每个类别中，系统所监控的 12 个关键权限的分布图，下图 3-4，3-5 分别展示了游戏权限分布和影音图像权限分布：

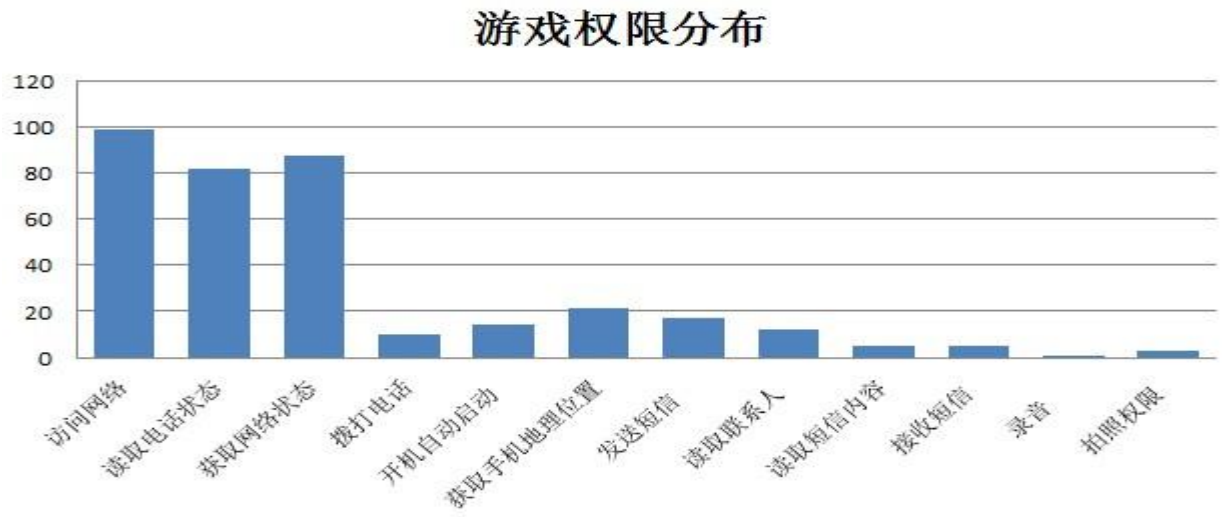

图 3-5 游戏权限分布

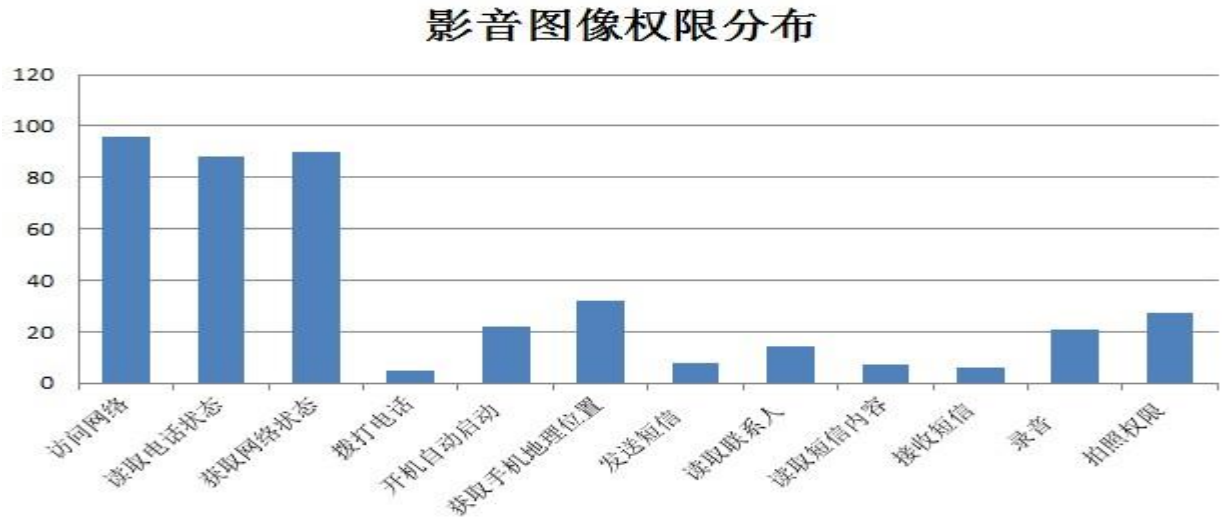

图 3-6 影音图像权限分布

根据每类别 100 个样本的统计中可以看出该类软件大致的权限分布，并从中定义一些阈值，作为推荐策略的评判标准。每个用户在进行权限控制时，可以指定当前软件所属的类别，由系统根据各类软件权限分布为用户推荐出适合的权限控制方案。

3. 日志记录

对于刚刚安装的应用，也许用户并不了解其相关的各权限功能，不能马上对其进行有效的安全策略定制。针对这种情况，对于经过加固的应用，手机端监控/管理模块能够对其可疑行为进行日志记录，在行为发生的同时以通知栏以及日志的形式进行行为记录。这样一来，除了刚刚安装时刻我们推荐的用户策略以外，用户可以随时审核应用程序的行为日志，以便用户对其行为有更加全面的了解，便于用户准确的实施安全策略。该日志采用 SQLite 进行记录。最后用户可由手机端监控/管理模块对其应用的日志进行审阅，以满足用户的需求。

4. 与加固应用交互

手机监控/管理模块需要与加固应用进行实时通信。每当加固代码监控行为被触发，其会向手机端进行通信并传递一些策略数据（何种权限触发，附带内容）。管理模块查询内存中权限策略 hashmap。根据该应用权限状态集合与应用触发的权限进行判断，以确认该权限是否处于启用状态并返回决策信息。同时开辟新线程对其行为进行通知以及日志记录。整体流程如下图 3-7 所示：

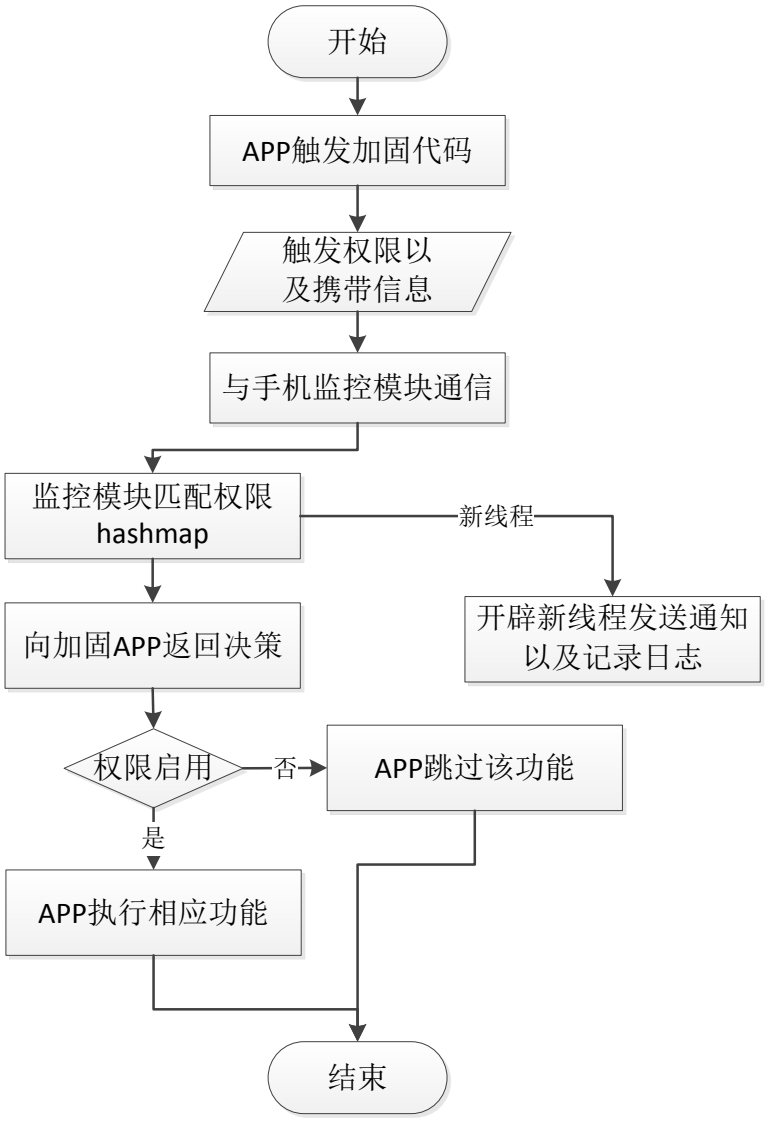

图 3-7 交互决策流程图

3.2.4 服务端重打包模块

服务端模块主要负责安全代码的注入及重打包。其主要技术在于对应用程序的静

态分析，如何快速准确的完成安全代码的注入、重打包后程序的稳定运行等将是该模块考虑的核心。该模块使用 Python 语言自动化实现。图 3-8 展示了服务端重打包整体框架：

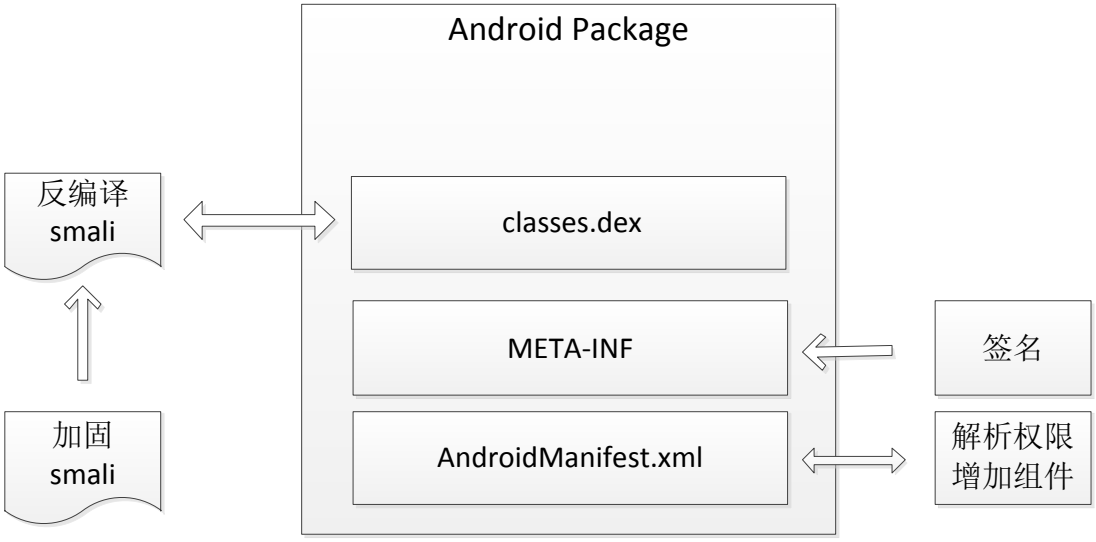

图 3-8 重打包框架

该流程主要包括对 APK 各个文件的解析（重点在于其配置文件 AndroidManifest.xml 的解析），Smali 语法的快速分析以及 Dalvik 字节码的注入等。该系统重打包流程如下：

(1) 使用 Apktool 工具对待打包 APK 进行解包。

(2) 对 AndroidManifest.xml 进行解析，包括申请权限、组件以及 Intent Action。

(3) 根据解析出来的权限比配权限函数映射表，确定待修改的 dalvik 字节码。

(4) 根据 smali 语法对 dalvik 字节码进行修改（修改关键函数调用语句跳转至加固代码），并导入加固代码 smali 文件。

(5) 使用 Apktool 工具对其进行重新编译，生成未签名的 APK。

(6) 使用 jarsigner 对生成的 APK 进行签名。

(7) 使用 zipalign 对签名后的 APK 进行字节对齐处理。

经过以上处理步骤后，才能生成可以在 Android 系统中运行的加固应用。

以上处理流程如下图 3-9 所示：

图 3-9 服务端重打包流程

该模块关键技术在于对 smali 语法，dalvik 字节码的分析与修改。该体统中加固代码位于代码包 repackaging/func/中，该包为很多策略类。每个类中的监控函数均被声明为 static，不用构造实例即可被调用，减少代码修改篇幅。修改样例以 Landroid/content/ContentResolver;->query 作为演示，在未加修改的原程序流程中，调用语句为：invoke-virtual/range {p0 .. p5}, Landroid/content/ContentResolver;->query(Landroid/net/Uri;[Ljava/lang/String;Ljava/lang/String;[Ljava/lang/String;Ljava/lang/String;)Landroid/database/Cursor;

invoke-virtual 指明调用的是实例函数，{p0 .. p5}为函数参数。其中 p0 为

Landroid/content/ContentResolver;实例对象引用。

经过加固打包后，原代码修改为如下代码：invoke-static/range {p0 .. p5}, Lrepackaging/func/PrivacyPolicy;->query(Landroid/content/ContentResolver; Landroid/net/Uri;[Ljava/lang/String;Ljava/lang/String;[Ljava/lang/String; Ljava/lang/String;)Landroid/database/Cursor;

invoke-static 指明调用静态函数，因此不需要传递类实例对象引用。但由于加固代码在封装原函数时多了一个函数参数，p0 就变成了第一个函数参数的引用。这么做的目的在于尽可能减小原来应用代码的修改量，同时以便于加固代码内部能更好的调用 Landroid/content/ContentResolver;->query 函数。由于每个函数特征不同，代码修改方式也有所不同，但原理类似这里就不再进行阐述。

参考文献

[1] Xu R, Saïdi H, Anderson R. Aurasium: Practical policy enforcement for android applications[C].Proceedings of the 21st USENIX conference on Security symposium. USENIX Association, 2012: 27-27.

[2] 符易阳, 周丹平. Android 安全机制分析[J]. 信息网络安全, 2011, 9: 012.

[3] 杨丰盛. Android应用开发揭秘. 北京：机械工业出版社，2010.1. 5-8.

[4] 郭宏志. Android应用开发详解. 北京：电子工业出版社，2010.6. 131-142.

[5] 李洋. Android中的沙箱机制. http://mobile.51cto.com/abased-354672.htm. 2011.

[6] 张中文，雷灵光，王跃武. Android Permission机制的实现与安全分析. 第27次全国计算机安全学术交流会论文集，2012，8:3-6

[7] 肖梓航. Android软件安全攻防现状. 第五届“互联网安全论坛（ISF)，2012.11.

[8] 丰生强. Android软件安全与逆向分析. 北京：人民邮电出版社，2013. 43-47.

[9] 王梓. 移动互联网之智能终端安全揭秘. 北京:电子工业出版社,2012.1. 71-74.

[10] Felt A P, Chin E, Hanna S, et al. Android permissions demystified[C]. Proceedings of the 18th ACM conference on Computer and communications security. ACM, 2011: 627-638.

[11] Berthome P, Fecherolle T, Guilloteau N, et al. Repackaging Android Applications for Auditing Access to Private Data[C]. Availability, Reliability and Security (ARES), 2012 Seventh International Conference on. IEEE, 2012: 388-396.