基于动静态结合分析的Android应用安全性多重检测模型

张钦尧 解梦飞 刘俊 陈明翔

(中国民航大学计算机科学与技术学院 天津 300300)

摘要

本文在对大量Android应用程序的研究基础上，提出并设计一种评估Android应用安全性的多重检测模型。模型从动态分析，静态分析两个衡量软件安全性的方向入手，结合机器学习算法，对Android应用中可能存在的恶意代码攻击，漏洞代码利用，隐私泄露三种问题进行了研究并设计检测方案。动态分析基于Android API Hook技术进行敏感行为监控，并自行设计风险评估模型进行预警，同时还基于代理和VPN设计实现了应用流量的分析、拦截模块。静态分析则分为三方面，第一部分通过提取权限信息与敏感API建立特征向量，结合对不同机器学习算法的性能评估，提出了一种基于随机森林算法的恶意软件鉴别模型，第二方面辅助以数据流分析技术实现高精度的漏洞扫描，第三方面采用FlowDroid提供的污点分析算法实现隐私泄露的检测。

关键词 Android，Hook，机器学习，静态分析，数据流，漏洞扫描，隐私泄露。

This paper presented and designed a multiple detection model on assessing Android application security based on studying on lots of Android malwares. The model starts from the dynamic analysis and static analysis of two directions to measure software security. Combining with Machine Learning Algorithms, it studies and designs three detection methods for malicious code attacks, vulnerability code utilization, and privacy leaks in Android applications. Dynamic analysis module implements monitoring sensitive behaviors based on the technology of Android API Hook, and designs an algorithm of risk feature to alert users. The module also implements traffic analysis and interception feature based on proxy and VPN. Static Analysis module is divided into three parts. The first part through extracting permission information and sensitive APIs to create eigenvectors, and combining with assessing on the performance of different Machine Learning Algorithms, presents a model on malware identification based on Random Forest Algorithm. The second part implements High-precision Vulnerability Scanning based on data stream analysis. The last part adapts stain analysis algorithm presented by FlowDroid to implement privacy leaks detection.

0 引言

近年来，Android系统智能设备在全球市场的占有率越来越高，Android系统的安全问题却层出不穷，一方面各大应用市场宽松的审查机制导致Android恶意软件泛滥，在网络上流传不止；另一方面开发者的水平不一也导致Android应用自身的安全性良莠不齐。随着时间的推移，各大厂商的手机大多都已安装了自家的安全软件，并配套了具有审查机制的应用市场。但是，由于Android系统的开源性，许多智能手机爱好者会为系统进行root提权，从而对手机进行高度定制。然而，各大厂商原生的安全机制对root手机不再适用，用户在没有对Android应用的安全性进行确认下，经过root提权的手机（下简称root手机）更容易被恶意软件以高权限入侵。

与此同时，目前常见的Android应用安全性检测方案主要使用动态和静态两种检测技术，但大多数设计方案多只针对其中一项技术进行改进或直接应用，而少有对这些技术检测效果的相关性与互补性进行研究。这明显无法对root手机进行全面、高效的安全防护。

在此背景下，本文提出并设计一种综合多种检测技术的Android应用安全性检测模型，基于现有的检测技术进行改进，并通过不同检测方案间的互补性来设计检测流程，一方面可以提高检测的精度，另一方面则可以扩展应用安全性的评估面。此外，本系统针对root手机的特点，在应用安全性评估模型中还着重实现了动态的风险评估和危险行为拦截，具有全面性、精确性、高效性。



图1 系统各模块及其组成部分

1 基于动静态结合分析的Android恶意应用多重检测模型的总体设计

1.1 整体功能

本系统包括：动态监测部分和静态检测部分，分别运行在本系统的Android客户端和服务器端。图1为本系统的各模块和它们工作组成部分。动态监测部分包括敏感行为监测模块和流量监控模块，静态检测部分包括恶意软件鉴别模块、漏洞检测模块和隐私泄露检测模块。

1.2 系统工作流程

图2为本系统工作流程图。



图2 系统工作流程

本系统Android客户端启动后，动态监测模块即开始工作，在后台监测所有第三方应用API调用和流量使用状况，若发现某非白名单应用恶意行为，则询问用户是否信任该应用，若用户选择信任，则该应用会被添加到白名单中；若用户选择不信任，则客户端会将应用上传到服务器端，同时对该应用进行恶意软件鉴定、漏洞扫描和隐私泄露检测，并生成一份鉴定结果和分析报告，发送回客户端反馈给用户，同时部分结果和报告内容加入分析规则。

2 动态监测部分

2.1 动态监测部分综述

动态监测部分运行在Android客户端，在用户运行第三方程序时为其监控应用行为，并向用户预警恶意行为。动态监测部分主要划分为两个模块，分别从两个不同方面进行Android第三方应用行为监控。敏感行为监测模块基于API Hook对Android应用敏感API调用进行钩取、分析，并设计高效风险评估算法对应用风险进行评估；恶意流量监控mokuai 使用代理接管Android设备所有网络通信，按规则对Android应用进行流量监控和拦截。

2.2 基于API Hook的敏感行为监测模块

Android 操作系统维护着自己的一套事件分发机制，Android应用程序、及其触发事件和后台逻辑处理，都是根据该机制一步步向下执行，恶意软件的敏感行为也不例外。若能在事件传送到终点前将事件截获并对其操作进行一定修改，就可以有效地抑制恶意行为的效果，这就是Hook机制。

本模块设计一种基于Xposed框架的Hook模块，运行于Android客户端上，利用Xposed框架对Zygote进程及其创建的Dalvik虚拟机的劫持[1]，在开机时完成对所有的Hook方法的重定义或改写，实现对于应用中调用的系统敏感API的钩取、分析、预警。

2.2.1 确定敏感行为

通常，移动平台的恶意行为包含：恶意控制设备、强行推送广告、偷跑流量、窃取用户隐私、获取用户实时动态等[2]，根据以上恶意行为，Hook模块预设表1行为为高风险行为：

表1 高风险操作

|  |  |  |
| --- | --- | --- |
| 行为分类 | 具体行为 | 相关类 |
| 账户操作 | 获取本机账户 | AccountManager |
| 应用管理 | 关闭、重启应用 | ActivityManager |
| 蓝牙操作 | 打开蓝牙、蓝牙连接 | BluetoothAdapter |
| 浏览器操作 | 获取用户书签、历史记录 | Broswer |
| 执行Shell | 各种方式执行Linux Shell | Runtime |
| 系统数据库操作 | 对于联系人、信息、通话记录、  浏览器书签的增删改查 | ContentResolver |
| 下载操作 | 设置URL、下载 | DownloadManager |
| HTTP连接 | 设置URL、连接 | HttpUrlConnection |
| 音频录制 | 录音、设置存放位置 | MediaRecoder |

2.2.2 模块工作流程

模块工作流程包含API Hook模块、风险评估模块、风险预警模块。

(一). API Hook

Hook模块随Android客户端安装后托管于Xposed模块，运行于后台。由于不少敏感API会在系统运行的过程中被大量的系统应用调用，所以Hook模块为用户提供选择接口。为防止恶意API之间过度耦合地相互递归调用而使Hook模块占用大量系统资源，采用在Xposed框架提供的beforeMethodHook和afterMethodHook方法中分别对调用敏感API的线程进行标记和消除标记的方法，只对含有线程标记的敏感API进行分析[3]。

API Hook模块每钩取到一个API，就会记录其调用时间、重要参数，加入到相应Android应用程序的API调用链中。API Hook模块只记录固定时间内的API调用，若超过该固定时间，API调用链中过早调用的API会被删去。若该应用在固定时间内程序调用链长度达到阈值或者API调用链中出现了固定的恶意API组合，模块就会启动恶意风险评估流程。

(二). 风险评估

本文提出并设计一种风险评估模型，首先提出以下前提和概念：

a) 本文预设敏感API集为。

b) 定义 为App调用的敏感API集，包含于。

c) 对APP中的敏感API调用链进行分析、评估，分以下两种情况执行：

1) 若无匹配恶意行为特征API集的调用出现：

Hook模块设计前期，对大量恶意软件的反编译提取其中敏感API调用代码。本文通过大量的动态监测分析其恶意程度，定义了各敏感API的恶意权重。部分敏感API及其恶意权重表2所示。

表2 部分敏感API及其恶意权重列表

|  |  |  |
| --- | --- | --- |
| 序号 | 敏感API | 恶意权重 |
| 1 | android.net.Uri: parse(): | 0.02 |
| 2 | android.database.sqlite.SQLiteDatabase: execSQL(): | 0.05 |
| 3 | android.content.Intent: getAction(): | 0.02 |
| 4 | org.apache.http.client.HttpClient: execute(): | 0.06 |
| 5 | android.telephony.TelephonyManager: getDeviceId(): | 0.08 |
| 6 | java.lang.Runtime: exec(): | 0.09 |
| 7 | java.net.URLConnection: connect(): | 0.001 |
| 8 | android.content.ContentResolver: query(): | 0.06 |
| 9 | android.os.Handler: obtainMessage(): | 0.05 |

Hook模块提出一种风险评估算法：

定义 为执行过程中调用的敏感API集。

定义 为的调用权重，特别地若，则。

定义为该App的带权恶意系数。通过机器学习分类算法计算，若，则判定该应用为恶意软件。

2) 若匹配到恶意行为特征API集：

设为在应用运行过程中被调用的时间。定义

为分别被调用的时间差。

设为该应用中包含的一种恶意API组合集，则定义

为该恶意API组合调用的组合时间恶意系数，若，则将该App评估为恶意App。

使用现有的App自动化检测系统得出在100款恶意软件上，该风险评估算法成功率为87%。

2.2.5 风险预警

对于判定为恶意App的应用，API Hook模块提醒用户卸载，并将其包名、签名添加到黑名单中。Hook模块监听系统安装程序，对于黑名单中的应用直接予以卸载处理。

对于未判定为恶意但进入风险评估流程的App，本系统会在用户允许下将其上传到服务器，启动静态检测流程。

2.2.6 敏感行为监测系统总结

本系统的优势在于：

一、 对Xposed框架提供的SDK进行封装后，Hook代码容易管理，便于bug修复和系统更新。

二、Hook模块给每个用户应用生成完善、精确的敏感API调用日志，对于系统保护、0day漏洞发掘、研究提供可靠数据保障。

三、API调用链更新机制保证了大量低危API调用（如HTTP连接、加密解密等）在调用时由于其较低的恶意权重不会预警，保证了多数安全APP在联网操作时不会被误判。

四、位于客户端的Hook模块代码托管于Xposed框架， Xposed框架对于系统内存、设备电量占用、消耗极少

五、为每个API设置恶意权重、调用权重保证了风险评估算法的高效性，使其具有较高的恶意软件检测成功率。

2.3 Android流量监控、分析、拦截模型

多数Android恶意应用通过联网下载App、展示广告等方式盈利，这不仅导致用户消耗大量流量，还可能威胁到用户隐私和Android系统安全。基于恶意应用依赖互联网盈利这个特点，在Android设备上搭建一个流量监控、分析、拦截平台，建立恶意流量规则库，对恶意应用的请求报文进行拦截，从而阻止恶意应用的恶意行为，保护用户的流量和隐私。

2.3.1 常用Android流量监控方法

获取Android设备流量是分析应用网络行为的前提，目前分析人员常用的Android流量监控方法有如下几种：

1）使用代理服务器。Android常用代理服务器有FiddlerCore、browermob-proxy等。分析人员设置Android网络环境的代理IP和端口，在这上面运行流量代理程序，从而获取和控制所有应用层流量。FiddlerCore和browermob-proxy都提供了完善的API接口，使得开发代理程序较为容易。

2）使用VPN。 Google在Android API level 14中加入了VPNService框架，用来创建Android VPN程序。在VPNService框架中，分析人员可以获取和控制所有网络层流量。此外，网络层中包含的传输层端口信息可使分析人员区分出不同应用的流量。

3）改造Linux防火墙[4] 。Andriod系统基于Linux，改造现有的Linux防火墙，使用iptables链钩子技术，劫持内核中的流量并隐藏劫持行为。这种方法可以不依赖分析人员的任何操作，但实现较为复杂。

4）使用tcpdump。 tcpdump是一个命令行抓包工具，支持多种协议的过滤抓包，但是tcpdump只能抓取流量而不能拦截控制流量，因此不适用于本模型。

5）借助PC端流量分析软件。PC端具有功能强大的流量分析软件，如wireshark、fiddler、charles等。Android设备连接PC端热点，PC流量分析软件监听热点即可间接获取Android设备的流量。由于此法简单方便，临时分析Android流量一般采用此法。

根据以上的比较分析，本文采用的是VPN和代理服务器的方式实现对恶意应用的流量监控拦截。

2.3.2 Android流量监控、分析、拦截模型的实现



图3 Android流量监控、分析、拦截平台架构

如图3所示，Android流量监控、分析、拦截平台采用C-S模式，客户端实现监控和拦截功能，服务器中实现恶意流量规则的汇总、判优和同步。

1）客户端的主要设计

客户端的Android VPN程序可区分出不同应用的流量，适合用来实现Android流量监控功能。流量监控模块运行流程如图4所示。用户首先选择并运行待分析的应用，VPN程序获得待分析应用产生的IP数据包，根据数据包的端口，确认IP数据包属于哪个应用。最后对IP数据包进行格式化解析，向用户展示解析结果。用户参考hook模块得到的应用网络请求报告，编写自定义拦截规则添加到数据库中。



图4 基于VPN的流量监控模块流程

客户端的流量拦截模块基于browermob-proxy代理工具进行设计[…]。browermob-proxy是优质的开源项目，除了使用API接口提供的功能外，本文通过修改其源码有效实现了Android流量拦截功能。同时为了保证代理不被系统回收，将代理写入Android前台服务中并在通知栏中显示运行状态。通过SQLite搭建一个本地恶意URL模式数据库，代理根据数据库中的模式，匹配每个请求包的URL，匹配成功则判断该请求为恶意请求，向应用返回空回应报文，从而阻断应用的恶意请求。代理检查一个请求URL是否为恶意URL，需要对数据库进行遍历。在网络负载较大的情况下，要求代理具有较高的效率。为了平衡内存占用和数据库访问次数，设计了下图5所示的匹配算法。



图5 代理URL模式匹配流程图

每次访问数据库都取出100条恶意URL模式放入内存中，对请求包的URL的模式匹配在这100条中进行。若在100条记录中未能命中，则取下100条记录重复上述过程。

2）服务器端的设计

如果hook模块判断运行的应用为恶意应用，将hook分析报告中的URL全部添加到URL黑名单数据库中，代理默认全部拦截。应用下次运行时，恶意应用的所有网络请求需要得到用户的同意，代理才会放行。

在用户同意分享规则库的情况下，服务器负责汇总用户编写的恶意流量规则，添加到共享恶意URL模式数据库中，并向所有用户公开。用户可对共享规则进行评分。服务器对用户积累的高评分优质规则定期汇总到公共恶意URL模式数据库中，再定期同步到所有客户端中。随着用户共享规则的积累，逐步加强对恶意流量拦截的广度和深度。

3 静态检测模块

3.1 静态检测模块综述

静态检测模块位于Android应用安全性评估模型的服务器端，在动态检测模块识别出应用运行中出现的异常行为后，用户可以选择将此应用上传到服务器端，通过服务器端的静态检测模块，对应用进行更加详细、全面的安全性评估，并向Android客户端返回该应用的检测报告。

静态检测模块主要划分为三个功能模块，分别从三个不同的方面进行应用安全性的评估。第一个功能模块基于权限与敏感API的机器学习分类算法实现恶意软件鉴别，并返回应用是否为恶意软件的判断结果。第二个功能模块用于对Android应用进行漏洞扫描，以检测出影响应用安全性的漏洞代码，此外，模块结合数据流分析技术对应用进行漏洞检测，可以有效提高漏洞检测的准确性。第三个功能模块用于检测Android应用中可能存在的隐私泄露问题，系统在对Android的隐私API与隐私泄露API进行归类整理后，使用FlowDroid寻找隐私数据的传播路径。

通过上述的三个功能模块，服务端在静态分析的基础上，分别从应用是否具有恶意行为、应用的代码编写是否存在漏洞、应用是否存在隐私窃取行为三个角度来综合衡量被上传的Android应用的安全性。

3.2 基于权限与敏感API的机器学习分类算法静态检测模型

本模块针对Android应用程序的权限机制与敏感API的调用情况，分别建立基于权限与敏感API的特征向量，然后使用Native Bayes与Random Forest两种机器学习算法建立分类模型。通过对两种分类模型的性能评估，提出了基于敏感API调用的随机森林算法分类模型，从而完成对恶意软件与正常应用的基本鉴别，并反馈用户鉴别结果。

3.2.1 特征选取

1)权限信息

Android应用程序的一系列行为都需要它所申请的权限[5]作为支撑，所以权限在一定程度上反应了应用程序的行为模式。通过对Android应用程序的配置文件AndroidManifest.xml的分析，提取了关于应用注册的权限信息。

由于Android应用程序的权限信息太过冗杂，而且权限之间有较大的相关性，例如在发送短信过程中，必然伴随着接收短信，所以本文对提取出的权限信息进行分类处理，去除权限之间的相关性。这里使用皮尔逊相关系数[6]这种度量方法计算两种权限的相似度，皮尔逊相关系数是一种线性相关系数，用来反应两个变量线性相关程度。计算公式如下所示：

*X*与*Y*表示两个权限，表示X和Y的相似度，它的值从-1到1。通过大量测试得出，如果两个权限的大于0.6，说明这两个权限具有强相关性，那么就把这两个权限归为一个权限簇，表3是经过分类后得到的7个权限簇，其中列举了部分权限。

表3 部分高危权限簇展示

|  |  |
| --- | --- |
| 类别 | 权限簇 |
| 1 | android.permission.WRITE\_CONTACTS  android.permission.GET\_ACCOUNTS |
| 2 | android.permission.READ\_CALENDAR  android.permission.WRITE\_CALENDAR |
| 3 | android.permission.READ\_CALL\_LOG  android.permission.READ\_PHONE\_STATE  android.permission.CALL\_PHONE |
| 4 | android.permission.ACCESS\_FINE\_LOCATION  android.permission.ACCESS\_COARSE\_LOCATION |
| 5 | android.permission.READ\_EXTERNAL\_STORAGE  android.permission.WRITE\_EXTERNAL\_STORAGE |
| 6 | android.permission.RECORD\_AUDIO |
| 7 | android.permission.READ\_SMS  android.permission.RECEIVE\_WAP\_PUSH |

通过去除权限之间的相关性，可以提高分类器的准确率，而且也在一定程度上减轻了分类器的工作量。

b) 敏感API调用

敏感API[7]包括涉及窃取用户隐私行为的高危函数接口，通过这些函数的调用，程序可以直接或间接地获取一些敏感数据，例如手机联系人、短信、账号、密码等。本文通过对应用程序反编译之后的中间代码进行提取，整理出53个调用次数较多的敏感API，其中部分API如表4所示：

表4 部分敏感API列表

|  |  |
| --- | --- |
| 类别 | 敏感API |
| 1 | obtainMessage() |
| 2 | getLastKnownLocation(java.lang.String) |
| 3 | void execSQL(java.lang.String) |
| 4 | sendMultipartTextMessage() |
| 5 | getAccounts() |
| 6 | setDataAndType(Uri,String) |
| 7 | killBackgroundProcesses(java.lang.String) |

将敏感API的调用次数作为特征值构建特征向量，由于不同大小的应用程序相对调用API的次数不同，所以本文将应用程序的大小与API调用次数做了加权平均，来降低应用程序大小带来的影响。

3.2.2 机器学习分类算法

不同分类算法的分类性能直接影响了鉴别恶意应用的准确率，本文使用的机器学习算法包括朴素贝叶斯(Native Bayes)与随机森林(Random Forest)算法。

a) 朴素贝叶斯

贝叶斯算法是基于概率框架下实施决策的基本方法，对分类任务来说，在所有相关概率都已知的理想情况下，贝叶斯决策论考虑如何基于这些概率和误判损失来选择最优的类别标记，实现公式如下所示：

 (1)

其中，为是类“先验概率”，是类“后验概率”，朴素贝叶斯模型采取“判别式模型”策略：即给定*x*，通过直接建模来预测*c*。对于本模型，为该样本是存在某一权限的概率，表示已确定样本类型的情况下，存在某一权限的概率，将与输入到(1)中，就能得到该应用为恶意应用的概率。

b) 随机森林

随机森林[8] [9]是一种重要的基于Bagging的集成学习方法，可以用来做分类，回归等问题。随机森林使用CART决策树作为弱学习器，再次基础上，对决策树的建立做了改进，对于普通的决策树，会在其节点上所有的样本特征中选择一个最优的特征来做决策树的左右子树划分，而随机森林通过随机选择节点上的一部分特征，再从这些特征中选取一个最优的特征作为决策树的左右子树划分，这样进一步增强了模型的泛化能力，避免过拟合问题。

对于本模块，由于随机森林算法具备善于选取最优特征的能力，所以很适合从具备大量特征的权限与敏感API中找出最具代表性的恶意特征。

3.2.3 模型分析与评估

取正常应用与恶意应用各1500个作为训练集，另取250个作为测试集，其中恶意应用来自公开的恶意软件库VirusShare[10]，正常应用爬取自GooglePlay商店，基本涵盖了所有应用类别。

首先从应用程序的配置文件提取权限信息，然后将权限信息与高危权限簇匹配，生成特征向量，最后将所有训练集与测试集的特征向量输入到分类器中。采用精确率(pre)与召回率(recall)[11]作为衡量分类性能的指标，实验结果如表11所示：

表5 基于权限特征的实验结果

|  |  |  |  |
| --- | --- | --- | --- |
| 算法类型 | ACC | pre | recall |
| 朴素贝叶斯 | 0.888 | 0.958 | 0.812 |
| 随机森林 | 0.926 | 0.946 | 0.907 |

由表中数据可见，朴素贝叶斯算法预测的精确度较高，但召回率较低，即对所有恶意软件的鉴别效果较差，而随机森林算法的分类性能普遍优于前者，pre与recall分别达到了94.6%与90.7%。

不同于权限，敏感API信息包含每个API的平均调用次数，所以恶意特征的表现方式会更加明显，通过反编译模块得到每个应用的API调用情况，结合调用次数建立基于敏感API的特征向量，并输入到分类器中。实验结果如表6所示：

表6 基于敏感API特征的实验结果

|  |  |  |  |
| --- | --- | --- | --- |
| 算法类型 | ACC | pre | recall |
| 朴素贝叶斯 | 0.860 | 0.855 | 0.867 |
| 随机森林 | 0.957 | 0.966 | 0.947 |

可以看出，随机森林算法的分类性能依然优于朴素贝叶斯算法，无论准确率(ACC)还是召回率(recall)都接近95%，精确率(pre)也接近97%，说明利用敏感API作为鉴别特征，基于随机森林分类算法能达到很好的分类效果。

3.2.4 样本容量对分类性能的影响

本文在进行模型测试过程中，发现并不是所有模型的分类性能都随样本容量的增加而提升，有时会出现负增长，鉴于此现象，本文通过控制样本容量进行循环测试，测试过程中依然保证正常应用与恶意应用的比例为1:1，以每次增加100个应用的速率递增，测试结果如图6所示:

图6 基于权限信息的样本容量对分类性能的影响

从图中可以明显看出使用朴素贝叶斯分类器在鉴别恶意应用时准确率出现了负增长，这是因为在样本容量达到1000后出现了过拟合问题。

图7 基于敏感API的样本容量对分类性能的影响

结合样本容量对检测准确度的影响曲线(图7)发现：

(1) 样本容量的增加并不意味着分类性能提升

(2) 随机森林分类器的分类性能由于朴素贝叶斯，而且不会发生过拟合现象

(3) 敏感API的恶意特征比权限特征更加明显，即两种分类器对敏感API的分类结果普遍优于权限特征。

综上所述，通过对不同算法的性能分析，本模块决定使用随机森林算法，结合敏感API特征鉴别恶意应用。

3.3 Android应用漏洞检测模块

在3.2所述的恶意应用鉴别模块判别出是否为恶意应用后，紧接着运行的是静态检测中的第二个模块——漏洞检测模块。Android应用中或多或少都可能存在一些编写上存在漏洞的代码，这些Android应用漏洞虽然可能并不含有恶意代码，无法直接被恶意应用鉴别模块检测出来，但在被攻击者恶意利用的情况下，同样可以对用户产生相当严重的威胁，降低应用的安全性。基于上述情况，本模型从静态分析角度入手，设计了Android应用漏洞检测模块。

传统的检测方案是利用反编译工具将dex文件转换为smali中间代码，之后对漏洞特征的匹配都是在smail代码上基于字符匹配完成的，这种方式虽然简单，但无法对需要结合上下文语义、逻辑复杂的漏洞实现高精度的识别，因此本模块采用Soot[4]进行反编译并在初始化中执行一些额外的工作。另一方面，为了提高漏洞检测精度，本文提出了一种基于过程内数据流分析技术[**13**]的变量行为记录算法，在漏洞检测中可以用于实现函数内变量到相关常量的追溯，提高检测准确率。

3.3.1 Android应用漏洞检测模块的工作流程

漏洞检测模块的完整工作流程如图8所示。首先使用Soot对APK进行初始化工作，输出的两个结果一个是Jimple中间代码，另一个则是控制流图，其中控制流图使用3.3.2小节中介绍的追溯算法为函数内的变量生成行为记录。初始化工作结束后，则开始进行漏洞特征匹配，最后输出检测结果。



图8 漏洞检测模块的工作流程

3.3.2 基于数据流分析的变量行为记录算法

在对漏洞特征进行匹配时，经常将特定的常量参数做为漏洞特征，然而应用开发中有很多情况是先将常量赋值到一个变量对象里，进行处理之后再作为参数传值到漏洞特征函数中，一个最简单的例子就是使用StringBuilder连接字符串，最后调用toString()方法将生成的字符串作为参数传给函数，对于这种情况，传统基于字符匹配的检测方案是无法匹配到特征的，本小节提出的算法用于则可以解决这种问题，算法基于过程内数据流分析生成变量的行为记录，之后基于此记录追溯变量的常量特征。具体算法流程如图9所示



图9 变量行为记录算法

在利用Soot生成控制流图后，使用Soot提供的过程内数据流分析框架来模拟完整控制结构下的语句遍历分析，并在此过程中基于传入的记录规则，详细记录所有变量的初始化、变量的方法和成员变量调用、变量赋值、变量值传递等所有变量相关的行为，力求在函数内为所有局部变量的行为建立准确详细的模型。表7是过程内数据流分析框架中需要实现的两个主要接口，用于指定变量行为记录规则。

表7 过程内数据流分析框架的两个主要接口

|  |  |
| --- | --- |
| 方法名 | 作用 |
| flowThrough(d.inFlow,d.data,d.outFlow) | 对传入语句d.data进行语法分析，在d.inFlow参数的记录基础上，添加新的记录并作为d.outFlow传出 |
| merge(in1,in2,out) | 对分支结构两条路径的记录结果in1、in2进行合并，生成整合的记录结果out传出 |

flowThrough(d.inFlow, d.data, d.outFlow)函数的具体工作是对传入语句d.data进行分析，并添加新的记录，系统实现中根据不同的语句类型进行相应的解析来生成新的记录，几种常见的语句类型对应的操作如下：

1) IdentityStmt为变量初始化语句，对左边的被初始化变量以及右边的初始化类型进行记录。

2) DefinitionStmt为定义类型语句，左操作数是变量的话，根据右边的表达式种类，可能的操作有：使用工厂函数进行变量初始化、变量赋值变量方法调用操作，根据右侧表达式类型进行分析记录。

3) InvokeStmt为函数调用类型语句，里面可能存在变量的方法调用，将调用的方法解析出来并存放到对应变量的调用记录中去。

merge(in1,in2,out)函数用于将分支结构中两条路径的记录进行合并以整合成一条记录，实现中为了保证所有分支路径下变量调用记录的完整性，对两条记录做并集来生成整合记录，即。

通过如上定义的数据流分析规则，即可生成函数内变量的行为记录，然后在如下位置提取常量作为变量的常量特征：

1) 变量的函数调用中所应用的常量参数，如果参数为变量的话则进行递归查询。

2) 变量的定义处涉及到的常量，若定义源头来自另一个变量的话，则对此变量进行递归查询

3.3.3 以WebView远程代码执行漏洞的检测为例介绍漏洞检测过程

WebView远程代码执行漏洞[13]是Android应用中相当常见的一个漏洞，其危害也相当广泛，本文将以此为例介绍使用Soot的初始化，并结合变量行为记录算法下的WebView远程代码执行漏洞的检测方案。

检测的第一步使用Soot进行初始化工作，一方面进行反编译，另一方面则为应用构造控制流图。初始化分析结束后，需要提取出应用所使用的API版本号，对于API<17的应用，使用3.3.2中说明的算法生成函数内所有变量的行为集合，并在集合中分别寻找WebView类型以及WebViewSetting类型的变量，并在行为集合中定位到以下三个成员函数的调用记录

1) setJavaScriptEnabled()

2) removeJavascriptInterface()

3) addJavascriptInterface()

没有调用setJavaScriptEnabled()来开启JavaScript调用的WebView变量不存在此漏洞，而对于将JavaScript调用设置为允许的WebView，则需要进行进一步的检查。定位removeJavascriptInterface()函数的参数，如果参数为变量，则使用3.3.2中介绍的常量追溯算法将其转化为常量，之后检查参数是否为searchBoxJavaBridge，如果不是的话则说明没有移除默认接口，可判定为存在WebView远程代码执行漏洞。此外，如果移除了默认接口但又添加了新Java实例的WebView同样存在问题。



图10 API<17时的WebView漏洞检测方案

对于API≥17的应用，因为提供了注解机制，所以直接查找addJavascriptInterface()函数的调用位置，之后从其中提取出传入的Java实例对象，并对此Java对象中使用JavaScript注解的成员方法进行分析，如果方法内含有危险API的话，同样判定此WebView存在漏洞。



图11 API≥17下的WebView漏洞检测方案

3.4 隐私泄露检测模块

在智能设备大量普及的当下，手机支付、手机社交都与移动设备紧密相关，可以说手机比电脑中存放着更多用户的隐私信息，这些信息一旦被应用恶意获取，就可能给用户带来无法挽回的损失。因此用户隐私安全性与Android应用的安全性是紧密相关的，隐私泄露检测模块便从隐私泄露的角度入手，进行Android应用安全性的评估。

在下面的3.4.1小节中，首先对Android系统中涉及到用户隐私的API（即污点分析算法中的Sources点，以下简称Sources点），以及可能发送出这些隐私数据的API（即污点分析算法中的Sinks点，以下简称Sinks点）进行详细的分析和归类。在3.4.2小节中，介绍基于过程间数据流分析技术[15]，寻找从Sources点到Sinks点的传播路径的污点分析算法。在最后的3.4.3小节中，则展示本模块在批量测试中，对应用隐私泄露问题的检测效果。

3.4.1 对Sources点及Sinks点的分析归类

Android系统中含有很多涉及到获取用户隐私的API，例如能返回用户设别MEID的getDeviceId()、获取用户精确位置的getLastKnownLocation()、获取浏览器书签信息的的getAllBookmarks()、获取浏览器历史记录的信息getAllVisitedUrls()等，通过归类整理本文将其大致分为以下6类，如表8所示

表8 对Android中涉及用户隐私API的分类

|  |  |
| --- | --- |
| 隐私类别 | Android系统中相关API条目数 |
| 用户位置信息 | 7 |
| 用户唯一标志信息 | 6 |
| 用户私人数据 | 9 |
| 用户网络访问相关信息 | 10 |
| 应用组件信息 | 9 |
| 程序运行间数据信息 | 98 |

在恶意应用获取到用户隐私数据之后，同样可以使用多种手段将数据发送到攻击者手中，例如通过sendTextMessage()发送短信、通过openConnection()的HTTP请求发送数据、通过Log.d()以日志形式记录数据、通过FileOutputStream.write()将数据写入到本地文件等，在进行归类后我们将其大致分为以下5类

表9 对Android中数据发送相关API的分类

|  |  |
| --- | --- |
| 数据发送方式类别 | Android系统中相关API条目数 |
| 通过短信发送数据 | 3 |
| 通过网络请求发送数据 | 13 |
| 通过日志记录数据 | 14 |
| 通过本地文件记录数据 | 21 |
| 通过应用组件传递数据 | 108 |

基于对Sources点和Sinks点的归类整理，之后便可以使用基于过程间数据流分析技术的污点传播算法，寻找从Sources点到Sinks点的传播路径，从而发现这些隐私泄露问题。

3.4.2 使用污点分析算法寻找传播路径

Android应用程序的运行方式不像其他程序一样有个固定的入口点，然后顺序执行下去，Android应用中每一个组件都有自己完整的生命周期，组件在运行过程中随时可能被触发调用，因此要对Android应用程序进行过程间数据流分析，就需要先为Android应用程序的生命周期建模，并构造一个虚拟的main方法，在方法内模拟触发所有组件，然后再对这个虚拟的main方法进行数据流分析，寻找Sources到Sinks间的路径。

模型使用FlowDroid[16]进行实现污点传播路径的寻找。FlowDroid是基于流分析技术的隐私泄露分析工具，它对Android应用程序的生命周期做了完整建模，并构建了一个虚拟的dummyMainMethod方法来模拟生命周期，在对应用内的Sources点及Sinks点做了标记后，使用过程间数据流分析技术寻找这些点间的传播路径。

本模型基于FlowDroid所提供的分析接口以及3.4.1归类的Sources点与Sinks点进行Android应用中的隐私泄露分析。

3.4.3 隐私泄漏检测模块的运行测试效果

基于3.4.1中整理的Sources点和Sinks点，模型对3053款恶意应用进行了隐私泄露检测，并对检测结果做了分类整理。在这3053款应用中，总共发现了1218条窃取用户隐私的行为，统计结果如表9、10所示。

表10 泄露的隐私类别统计

|  |  |
| --- | --- |
| 隐私类别 | 在测试中的发现数量 |
| 用户位置信息 | 79 |
| 用户唯一标志信息 | 145 |
| 用户私人数据 | 176 |
| 用户网络访问相关信息 | 270 |
| 应用组件信息 | 409 |
| 程序运行间数据信息 | 139 |

表11 隐私数据发送方式统计

|  |  |
| --- | --- |
| 隐私类别 | 在测试中的发现数量 |
| 通过短信发送数据 | 388 |
| 通过网络请求发送数据 | 223 |
| 通过日志记录数据 | 267 |
| 通过本地文件文件记录数据 | 157 |
| 通过应用组件传递数据 | 183 |

4 多重检测模型检测案例分析与结果

本节选取 “二手车市场”APP，进一步展现本文提出的多重检测模型的工作过程。选定应用安装后开始运行，敏感API监测模块和恶意流量监控模块在后台同时工作。

其中敏感行为监测模块在App正常运行的17秒中，监测到敏感API共121处，超过数量阈值，风险评估算法启动，在带权恶意系数达到0.63后（超过0.61的阈值）。所有的API调用和用户操作都记录在该日志文件中，由于该APP进行了大量的https连接，所以日志内容多为url、http、crypto相关API的记录，由于其恶意权重较低，所以仅列出部分日志内容，如图12所示：

表12 恶意行为监测模块部分日志内容

|  |  |  |
| --- | --- | --- |
| 时间 | 日志内容 | 备注 |
| 20:39:25 | Runtime->exec:cmd=sh | root命令 |
| 20:39:29 | Runtime->exec:  cmd=cat/proc/uid\_stat/10067/tcp\_snd | 读取本机tcp发送文件 |
| 20:39:29 | Runtime->exec:  cmd=cat /proc/uid\_stat/10067/tcp\_snd |  |
| 20:39:31 | Intent->getAction |  |
| 20:39:32 | ContentResolver->registerContentObserver:uri=content:settings/secure/accessibility\_captioning\_locale | 读取本机安全配置相关内容 |
| 20:39:33 | LocationManager->  getLastKnownLocation | 获取本机位置 |
| 20:39:36 | BluetoothAdapter->getAddress | 获取本机MAC地址 |
| 20:39:36 | WARNING!! Judge this App as a malware!! | 判定为恶意应用 |
| 20:39:36 | Runtime->exec:cmd= mount | 挂载手机内存 |
| 20:39:36 | Runtime->exec:cmd=  cat /proc/uid\_stat/10071/tcp\_rcv | 读取本机tcp接收文件 |
| 20:39:37 | Cipher->getInstance:Cipher  (RSA/NONE/OAEPWithSHA1AndMGF1Padding) | RSA加解密传输信息 |
| 20:39:41 | User accepted the warning. | 用户接收并通过预警 |

可以看到通过Hook技术钩取到该APP多处敏感行为，包括终端命令、数据库操作、位置获取、连接恶意链接等，通过风险评估算法判定该应用风险超过阈值。敏感API监测模块发送系统通知向用户预警，用户接受预警后选择通过预警通知关闭应用。

同时，流量拦截模块开启代理和VPN抓包程序，运行可疑应用，得到应用流量信息。在待分析应用中，存在大量广告、泄漏准确位置信息、跳转应用下载页等行为。

表13列举该应用流量中的部分恶意流量典型特征。

表13 部分恶意流量典型特征

|  |  |
| --- | --- |
| 行为 | 流量特征 |
| 广告 | https://j.autohome.com.cn |
| 位置信息 | 行政地址、经纬点、朝向等 |
| 应用下载 | https://collectionpv.che168.com  https://app.autohome.com.cn |

参考hook模块分析报告，结合流量分析结果，编写拦截正则表达式，作为用户自定义规则添加到模式数据库中。

表14 部分拦截正则表达式

|  |  |
| --- | --- |
| 行为 | 正则表达式 |
| 个推小广告 | http://sdk.open.phone.igexin.com |
| 应用下载 | https://[a-z]\*.autohome.com.cn |

第二次运行该应用时，针对的广告和应用下载都有效拦截并被代理所记录。

用户选择上传该应用至服务器端。服务器端首先启动恶意软件鉴别模块。鉴别模块通过对该应用程序进行反编译，从中提取敏感API调用信息，然后根据应用大小对敏感API的调用情况进行加权平均，生成特征向量。最后将特征向量输入到基于随机森林算法的检测模型中，结果为'0'，判定该应用为恶意应用（恶意应用标签为'0'，正常应用为'1'）。

接着启动漏洞检测模块，通过模块中11种漏洞的检测插件，一共检测到了121处共8类漏洞代码，图12为其中两处漏洞代码。



图12 测试样本中的两处漏洞代码

漏洞1存在的原因是在自实现的HostnameVerifier证书校验类中，校验函数verify()的返回值恒为1，没有实现有效校验，因此可能存在中间人攻击漏洞。

漏洞2存在的原因是在加解密中，密钥直接明文存储在APK中，攻击者通过反编译即可拿到密钥，在图中可以看到加解密使用的密钥是29Blp39J

对测试样本中存在的121处漏洞代码的统计结果如表15所示

表15 测试样本漏洞检测的统计结果

|  |  |
| --- | --- |
| 漏洞类型 | 数量 |
| checkServerTrusted()方法实现为空 | 4 |
| 检测到应用存在代码动态加载的行为 | 92 |
| HostnameVerifier非有效校验 | 3 |
| WebView明文存储密码漏洞 | 13 |
| 应用使用WebView，同时支持File协议 | 1 |
| WebView忽略SSL证书错误检测 | 2 |
| 应用程序加解密时密钥使用硬编码 | 5 |
| 应用外弹窗 | 1 |

最后执行的是隐私泄露检测模块，该模块检测到了存在的隐私泄露问题共141条，图13展示了其中两条隐私泄露代码。



图13 测试样本中的两处隐私泄露代码

在图13中，第一处隐私泄露代码通过getLastKnownLocation ()函数获得用户位置，再将其记录在日志中，造成用户隐私泄露。第二处隐私泄露代码通过getDeviceId()获得用户的设备ID，并通过网络请求将信息发送出去，造成用户隐私泄露。

对测试样本中存在的141处隐私泄露问题的统计结果如表13所示

表16 泄露的隐私类别统计

|  |  |
| --- | --- |
| 隐私类别 | 在测试中的发现数量 |
| 用户位置信息 | 1 |
| 用户唯一标志信息 | 18 |
| 用户网络访问相关信息 | 22 |
| 应用组件信息 | 1 |
| 程序运行间数据信息 | 99 |

表17 隐私数据发送方式统计

|  |  |
| --- | --- |
| 隐私类别 | 在测试中的发现数量 |
| 通过网络请求发送数据 | 2 |
| 通过日志记录数据 | 26 |
| 通过本地文件文件记录数据 | 27 |
| 通过应用组件传递数据 | 86 |

5 结束语

本文提出的多重检测模型有效地克服了市面上常见安全软件鉴别方法单一带来的检测不全面性，为root设备提供了安全保障；同时，由于Android软件的多样性与复杂性，本模型尚未实现对恶意行为的即时拦截与查杀，这是模型后续研究的侧重点。

参考文献

[1]于航.基于模拟器的沙箱系统研究[A]. 中国计算机学会.第30次全国计算机安全学术交流会论文集[C].中国计算机学会:,2015:5.

[2]任伟,柳坤,周金.An Da:恶意代码动态分析系统[J].信息网络安全,2014(8):28-33.

[3]蔡林,陈铁明.Android移动恶意代码检测的研究概述与展望[J].信息网络安全,2016(09):218-222.

[4]Arni Einarsson，Janus Dam Nielsen.A Survivor’s Guide to Java Program Analysis with Soot

[5]朱佳伟,喻梁文,关志,陈钟.Android权限机制安全研究综述[J].计算机应用研究,2015,32(10):2881-2885.

[6]卜义云.基于机器学习的Android恶意软件静态检测系统的设计与实现.电子科技大学.2016(03).49-53

[7]邵舒迪.基于权限和API特征结合的Android恶意软件检测方法.计算机科学.2017(4).136-138

[8]朱月俊,文爽,李剑.改进随机森林在Android恶意检测中的应用[J].信息安全研究,2017,3(11):1020-1027.

[9]张家旺.基于机器学习算法的Android恶意程序检测系统.计算机应用研究.2016(6).1776-1782

[10]VirusShare.https://virusshare.com

[11]杨宏宇,徐晋.Android恶意软件静态检测模型[J].吉林大学学报(工学版),2018,48(02):564-570.

[13]王旭.基于控制流分析和数据流分析的Java程序静态检测方法的研究

[14]韩心慧，丁怡婧，王东祺，黎桐辛，叶志远.Android恶意广告威胁分析与检测技术

[15] Eric Bodden. Inter-procedural Data-flow Analysis with IFDS/IDE and Soot

[16] Steven Arzt, Siegfried Rasthofer, Christian Fritz, Eric Bodden, Alexandre Bartel等. FlowDroid: Precise Context, Flow, Field, Object-sensitive and Lifecycle-aware Taint Analysis for Android Apps