软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2017,28(9):2248−2263 [doi: 10.13328/j.cnki.jos.005177]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

使用敏感路径识别方法分析安卓应用安全性*

缪小川[1,2], 汪 睿[1,2], 许 蕾[1,2], 张卫丰[3], 徐宝文[1,2]

[1](计算机软件新技术国家重点实验室(南京大学),江苏 南京 210023)

[2](南京大学 计算机科学与技术系,江苏 南京 210023)

[3](南京邮电大学 计算机学院,江苏 南京 210023)

通讯作者: 许蕾, E-mail: xlei@nju.edu.cn

摘 要: 安卓系统在手机端操作系统中长期占据主导地位,但由于安卓系统开放共享的特性和不够严谨的第三方市场审核机制,安卓平台受到众多恶意应用的侵扰.结合静态程序分析和机器学习方法,提出了基于敏感路径识别的安卓应用安全性分析方法.首先,针对恶意应用中存在的恶意行为以及触发条件,定义了敏感路径;其次,针对安卓应用中存在大量组件间函数调用关系问题,提出了一种生成应用组件间函数调用关系图的方法;再次,由于提取出的敏感路径信息无法直接作为识别特征,实现了一种基于敏感路径信息抽象的特征提取方法;最后,从 Google Play、豌豆荚、Drebin 等来源收集了 493 个应用 APK 文件作为实验数据集,该方法的准确率为 97.97%,高于基于 API-Feature 的检测方法(90.47%).此外,在恶意应用和良性应用检测的精度、召回率、*F* 度量等方面,该方法均优于 API-Feature 方法.另外,实验结果表明:APK 文件大小会影响实验的结果,尤其体现在分析时间上(0~4MB 大小的 APK 平均分析用时 89s;文件增大后,平均分析用时增长明显).

关键词: 安卓恶意应用;敏感路径;组件函数调用关系;程序静态分析;特征抽象

中图法分类号: TP311

中文引用格式: 缪小川,汪睿,许蕾,张卫丰,徐宝文.使用敏感路径识别方法分析安卓应用安全性.软件学报,2017,28(9):2248−2263. http://www.jos.org.cn/1000-9825/5177.htm

英文引用格式: Miao XC, Wang R, Xu L, Zhang WF, Xu BW. Security analysis for Android applications using sensitive path identification. Ruan Jian Xue Bao/Journal of Software, 2017,28(9):2248−2263 (in Chinese). http://www.jos.org.cn/1000-9825/5177.htm

Security Analysis for Android Applications Using Sensitive Path Identification

MIAO Xiao-Chuan[1,2], WANG Rui[1,2], XU Lei[1,2], ZHANG Wei-Feng[3], XU Bao-Wen[1,2]

[1](State Key Laboratory for Novel Software Technology (Nanjing University), Nanjing 210023, China)

[2](Department of Computer Science and Technology, Nanjing University, Nanjing 210023, China)

[3](School of Computer Science, Nanjing University of Posts and Telecommunication, Nanjing 210023, China)

Abstract: Android system dominates the mobile operating systems at present. Compared with iOS system, Android system is more open and has lots of third-party markets with loose audit mechanism. Therefore, there are more malwares in Android platform. In this paper, an Android security analysis based on sensitive path identification, which includes the static analysis and machine learning methods, is presented. Firstly, since malicious behaviors in malwares have their trigger conditions, the definition of sensitive path is provided. Secondly, a method is proposed to generate the inter-component call graph based on APK files base in the fact that there are a lot of

* 基金项目: 国家重点基础研究发展计划(973)(2014CB340702); 国家自然科学基金(61272080, 91418202, 61403187)

Foundation item: National Basic Research Program of China (973) (2014CB340702); National Natural Science Foundation of China (61272080, 91418202, 61403187)

收稿时间: 2016-07-13; 修改时间: 2016-09-04; 采用时间: 2016-11-10; jos 在线出版时间: 2017-02-20

CNKI 网络优先出版: 2017-02-20 14:05:19, http://www.cnki.net/kcms/detail/11.2560.TP.20170220.1405.014.html

inter-component call relations in Android applications. Thirdly, since the sensitive paths cannot be directly used as features, a method is designed to abstract the sensitive paths. Finally, 493 applications APK files are collected from Android markets and the existing data sets, such as Google Play, Wandoujia and Drebin, to construct a benchmark. Experiments indicate that the proposed method has higher accuracy (97.97%) than the method based on API-feature (90.47%), and its precision, recall and F-measure are also better than API-feature method. Furthermore, the scale of the APK file has influence to the experiment results, especially in analyzing time (when the APK files are within 0-4MB, the average analyzing time is 89 seconds; and when the files become larger, the time increases significantly).
Key words: Android malware; sensitive path; inter-component commnication; program static analysis; feature abstract

随着手机功能的日益强大以及移动网络环境质量的逐步提升,越来越多的人使用手机接入移动网络.据中国互联网信息中心 2015 年第 37 次中国互联网络发展状况统计报告(http://www.cnnic.net.cn/hlwfzyj/hlwxzbg/201601/P020160122469130059846.pdf)称,我国网民互联网接入设备从电脑端向手机端迁移的趋势明显.截至 2015 年 12 月,我国手机网民规模达 6.20 亿,网民中使用手机上网人群占比由 2014 年的 85.8%提升至 90.1%.安卓系统是常用的手机系统之一.数据网站 NetMarketShare(https://netmarketshare.com/)公布的信息表明:2016 年,全球智能手机市场中,安卓系统仍占据主导地位,所占份额为 59.65%.

安卓平台的开放性,使得其占据市场份额主导地位;与此同时,在安卓平台上开发和发布恶意软件也更加容易,这使得安卓平台成为众多恶意软件开发者的活跃地盘.目前,具有恶意口味、信息窃取等高危恶意行为的恶意软件快速增长,且这些恶意软件几乎全部都在瞄准安卓平台,已经严重威胁到网民的财产安全和信息安全.据新华网报道(http://news.xinhuanet.com/fortune/2016-01/29/c_1117940700.htm):2015 年,全国经检测发现的移动互联网恶意程序已经超过 145 万件,较 2014 年增长 52%.

如今,市面上涌现了大量针对移动平台的杀毒应用,但大量恶意应用通过代码混淆等手段隐蔽其恶意行为,导致现有的杀毒应用存在大量漏报、误报行为.因此,恶意应用,特别是安卓平台恶意应用的分析与检测成为移动平台的研究热点.本文提出一种基于敏感路径识别的安卓应用安全性分析方法,通过分析安卓应用的安装文件获取安卓应用的函数调用关系,并从其中找出进入点以及其所能到达的敏感行为的路径,并根据进入点和敏感行为的类型将其抽象为特征,进一步使用机器学习方法区分恶意和良性应用.本文的主要贡献包括:

(1) 提出了一种通过分析安卓应用安装 APK 文件获得其组件间函数调用关系图的方法.为了获取应用的敏感路径信息,首先需要获取应用的函数调用关系图.由于该图不包含安卓应用内组件间的调用关系,需要分析应用中的 Intent 参数来获取组件间的调用关系,以扩展原有的调用图,进而得到组件间函数调用关系图;

(2) 设计了敏感路径的概念以及敏感路径的分析方法.为了寻求进行安全性分析时可用的特征,定义敏感路径由敏感行为以及触发该行为的函数所构成.在组件间函数调用关系图中寻找代表敏感行为的节点,并通过敏感行为遍历调用图,找到敏感触发;

(3) 提出了将敏感路径数据化的特征抽象方法.由于应用所包含的敏感路径信息不能直接用做机器学习所用的特征值,根据敏感行为以及敏感触发的信息对它们进行划分和抽象,既约减了敏感路径的种类和数量,又能满足用户对手机应用安全不同程度的要求;

(4) 实现了结合静态分析与机器学习的、基于敏感路径识别的安卓应用安全性分析方法.使用静态分析的方法获取应用的敏感路径信息,再将这些信息特征抽象化,并使用机器学习(文中主要采用了 C4.5 决策树)的方法完成安卓应用安全性的分析.

本文第 1 节介绍相关研究工作.第 2 节展示敏感路径的示例和概念.第 3 节给出基于敏感路径识别的安卓应用安全性分析方法框架及其实现过程.第 4 节是实验及结果分析部分.第 5 节总结全文并展望未来工作.

1 相关研究工作

文献[1]介绍了安卓安全研究的最新进展.最常用的安卓平台恶意应用检测方法有 3 种,分别是基于静态分析的方法、基于动态分析的方法以及基于机器学习的方法,各种研究方法都存在各自的优缺点,具体介绍如下.

为了检测程序行为是否与已知恶意行为模式相匹配,常规做法是采用静态分析方法,包括构建程序控制流图、数据流图;分析变量依赖关系、函数调用关系等.静态分析方法的优点是代码覆盖率高;缺点是会有误报,无法解决代码混淆、加密以及在动态执行中才遇到的恶意代码问题.Felt 等人给出了一个权限许可(permission)分析工具 stowaway[2],该工具能够分析一个应用是否越权(超过了它本身功能的权限)访问,但是如果仅根据是否越权来判断是否为恶意软件,则会导致大量的误报.文献[3]通过触发安全敏感行为的上下文信息(包括事件和条件)来区分恶意应用和良性应用,使用静态程序分析方法来提取安全敏感行为(包括权限保护方法、信息流中的 source 方法或 sink 方法、反射方法和动态代码加载方法)的上下文信息.文献[4]通过静态污点分析来检测安卓应用中组件间的隐私泄露,通过寻找从敏感数据(source)到发送数据至应用或设备以外的语句(sink)的路径,考虑组件间上下文信息的传递(现有方法通常只考虑组件内),提高了分析的精度并尽可能降低误报.文献[5]根据收集到的应用间通信信息,以检测隐私数据泄露为具体安全需求,构建信息流矢量图展示应用通信路径,基于通信内容利用图的遍历进行恶意行为分析.文献[6]使用静态污点分析,通过大量挖掘 App 中正常的数据流模式来检测恶意行为.DroidAnalyzer[7]基于关键词查找技术,对于检测 rooting 功能非常有效.文献[8,9]通过生成数据依赖图并分析用户界面与程序敏感行为的一致性来检测恶意软件.文献[10]利用污点传播分析识别恶意行为之间的依赖关系,以此为基础构造控制依赖图和数据依赖图.

动态分析方法用沙盒或虚拟机模拟程序运行,监控、拦截程序运行行为.其优点是绕过了静态方法遇到的代码混淆和加密等方面的问题,缺点是需要应用程序实时运行,并且需要较长的时间采集应用程序的动态数据,另外,动态分析会有漏报:代码覆盖率低,并且有些恶意程序可防止自身在模拟器下运行.即:在模拟器下运行时,程序会自动崩溃.TaintDroid[11]是一个有效的实时污点分析工具,Droidbox[12]工具能够模拟用户的行为,两者功能互补,能够记录和分析对用户私有信息的实时访问信息.但这两种方法都缺乏对本地 API 的跟踪能力.Crowdroid[13]利用 strace(Linux 的一个调试工具)来监控安卓系统的每一次系统调用和获得的信号,但是它没有考虑来自 Dalvik VM 的信息,很难在与虚拟机上运行的程序之间建立程序依赖关系,从而不利于与虚拟机上获取的动态信息综合分析.AASanbox[14]和 DroidRanger[15]实现了静态方法和动态方法的组合,因而相对于单独的静态和动态方法有较好的效果.但是 DroidRanger 需要恶意软件样本来抽取恶意软件的行为特征,所以它不能发现那些复杂的恶意软件.文献[16]利用沙盒技术,通过静态和动态分析方法,采用特别的技术记录调用本地 API 的日志,最后采用 SVM 方法进行分类.文献[17]在安卓框架层设置权限检查点,并调用请求评估算法进行授权评估,从而实现对应用行为的监控.

基于机器学习的方法将恶意软件的检测看成是一个分类问题,通过分析良性应用与恶意应用在特征方面的差异,选择那些具有统计差别的特征,然后采用机器学习的方法进行学习分类.其优点是实现代价较小、操作简便,缺点是受训练集中应用的差异、特征的选取等因素影响较大.文献[18]、DroidMat[19]、Adagio[20]、MAST[21]和 DroidAPIMiner[22]通过静态分析方法抽取特征,然后采用机器学习方法进行分类.文献[23]针对这些问题给出了一种高效的恶意应用检测方法,但不能防止恶意应用的感染,因为该方法主要基于静态分析获取特征;其次,检测模型需要依赖已知的正常软件样本和恶意软件样本,不能加入最新样本;此外,混淆技术、重组、代码重排序和活动改名都会影响到它的效果.文献[24]对 1 173 个安卓应用进行了统计分析、相似度计算、聚类以及交叉对比,利用多个维度的安卓应用特征相似度分析,得到了安卓应用多个维度的相关规律.文献[25]基于机器学习的研究现状,通过分析用户评论来自动学习安全/隐私相关的行为.为提取 API 数据依赖关系,文献[26]进行了上下文敏感、流敏感、过程间数据流分析,进而通过子图挖掘和图压缩算法获取频繁子图,以表征特定的行为模式,再通过自然语言生成技术将这个图表示为用户可以理解的文本.文献[27]提出一种基于语义的恶意代码行为特征提取及检测方法,提取恶意代码的关键行为及行为间的依赖关系,再利用抗混淆引擎识别语义无关及语义等价行为,获取具有抗干扰能力的恶意代码行为特征.文献[28]提出一种综合考虑安卓多类行为特征的算法 THEA,先采用动静态结合的方法提取应用特征,再构建适合多类特征的最优分类器来综合评判应用是否恶意.

2 敏感路径识别

2.1 恶意应用的特征

文献[29]使用机器学习算法进行安全性分析,发现特征的选取与最终结果的准确性存在很大的关联.我们罗列了恶意应用从产生到恶意功能执行的生命周期各阶段中主要的特征,需要从中寻找恶意应用区别于良性应用的且容易获取的、便于数据化的特征信息.

(1) 产生传播.恶意应用产生后主要通过网络来传播,多数第三方市场较为松散的监管制度为恶意应用大范围的传播提供了温床.恶意应用的生产者可以较为容易地将恶意应用在第三方市场上架,使其获得了向大量安卓系统使用者进行推广的机会.同时,安卓系统在安装应用时并没有限定 APK 文件的来源,非市场渠道的 APK 文件也可以极为简易地进行安装,恶意应用的生产者只需要提供一个下载链接即可;

(2) 用户下载安装.虽然恶意应用有了传播途径,但是不同于部分电脑病毒只需要插上 U 盘等简单的动作就能感染病毒,恶意应用需要用户下载安装才能感染用户的手机.由于 APK 文件的可反编译性,恶意应用的生产者会选择下载一些热门的应用,将其反编译,再将包含恶意功能的代码与其一起打包得到新的 APK 文件,通过这样的手段来骗取用户下载.还有部分生产者选择将更新组件打包至热门应用中,在用户安装后,更新组件会下载包含恶意功能的代码,这样更增加了第三方市场的审查难度.除了采取重新打包的手段骗取用户信任外,部分恶意应用也会采用取相似的名称或者提供吸引用户的描述来诱骗用户下载安装;

(3) 恶意功能的激活.在安装至用户的安卓系统中以后,恶意应用还需要事件来触发其自身执行恶意功能.比较常见的方法是通过监听系统事件来激活恶意功能,因为采用该种手段不需要用户的参与,被发现的概率较小.其中最常用的系统事件是 BOOT_COMPELTED,即,系统启动成功后会触发恶意功能的执行.同时,部分恶意应用采取欺骗用户点击的方式,即:图形化界面 UI 上的文字表明这个按钮会开启一些良性的功能,但实际触发的却是一些恶意功能;

(4) 恶意功能的执行.恶意功能主要有以下 4 种:破坏系统,获取对安卓系统底层的使用权限,对系统造成损害等;直接的金钱损失,未经用户授权的发送信息、打电话、连接互联网等功能,直接造成用户的话费损失;隐私泄露,获取用户的私密信息,诸如账户、住址等信息的泄露,会间接造成用户财产甚至人身安全的损害;远程控制,通过远程控制来完成生产者意图执行的功能.

在以上罗列的各种恶意应用特征中,产生传播阶段的特征难以将其与良性应用区分开来,用户下载安装阶段的特征容易获取却难以数据化.因此,我们还是关注恶意功能的本身,选取了能够表示恶意行为及其触发组成的恶意路径信息作为应用的主要特征.

2.2 例 子

安卓平台的恶意应用在执行其恶意功能时,往往在无授权或者用户无法察觉的情况下调用、执行敏感的 API 函数.安卓平台恶意应用的检测方法绝大多数都是抓住此特征来判断应用是恶意还是良性.

例如,ExSms 是一个恶意应用,它会在未经使用者允许的情况下发送短信,其函数调用关系图(call graph,简称 CG)如图 1 所示.

该图经过简化,只保留了部分与短信发送函数 *SmsManager.sendTextMessage*()相关的信息.发送短信可能会导致用户话费的损失以及用户隐私的泄露,在本文中将其归类为敏感行为.图 1 中有 3 种方法会调用该函数,即:从 *DummyMainMethod*()函数出发,我们可以找到 3 条可以达到该函数的路径.

- 第 1 条路径经由 *Activity$a.OnCreate*()函数,由 *Activity$a.OnClick*()调用 *SmsManager.sendTextMessage*()函数.其中,*Activity$a.OnClick*()函数表明,最终调用 *SmsManager.sendTextMessage*()函数需要用户点击 UI 中的按钮.由此我们可以得知,该应用进行短信发送的操作是需要使用者操作的,即,发送短信是使用者知晓的行为;

- 第 2 条路径同样经由 *Activity$a.OnCreate()*函数,直接由 *Activity$a.b()*函数调用 *SmsManager.sendTextMessage()*函数.整条路径中没有出现需要使用者交互的函数节点,因此我们认为,使用者并不知道该应用进行了短信的发送.即,这是一种恶意行为;
- 第 3 条路径经由 *Service$d.OnCreate()*函数,最终由 *Service$d.e()*函数调用 *SmsManager.sendTextMessage()*函数.Service 组件是在后台运行的安卓组件,其运行不需要使用者与之交互.Service$d 由函数 *Recevier$c.OnCreate()*通过调用 *ContextWrapper.StartService()*函数创建.如图 2 所示,Broadcast Receiver 组件 *Recevier$c* 监听了 *android.intent.action.DATA_STATE* 事件,当手机的数据连接状态发生改变时,*Service* 组件 *Service$d* 就会被创建.该路径同样表明了 ExSms 可能会在使用者不知情时发送短信,也是一种恶意行为.

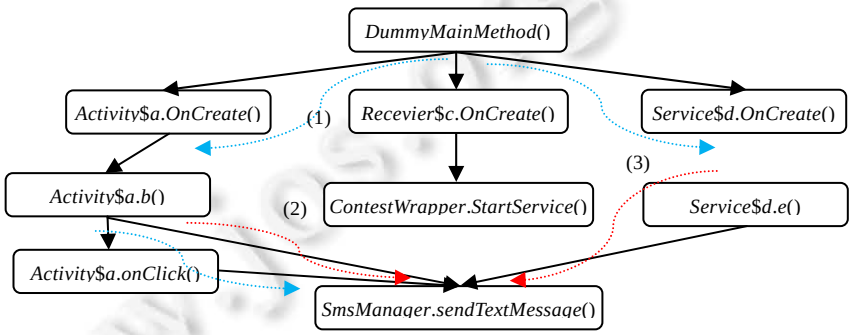


Fig.1 Simplified function call graph of App ExSms

图 1 应用 ExSms 的简化函数调用关系图

```
1  ⟨service android:name="com.android.main.Service$d" android:process=":main"/⟩
2  ⟨receiver android:name="com.android.main.Receiver$c"⟩
3      ⟨intent-filter⟩
4          ⟨action android:name="android.intent.action.DATA_STATE"/⟩
5      ⟨/intent-filter⟩
6  ⟨/receiver⟩
```

Fig.2 Code snippet of file Manifest.xml for App ExSms

图 2 ExSms 应用 Manifest.xml 文件中的相关代码片段

通过以上分析我们得知:应用 ExSms 中存在可疑的函数调用路径,这些路径表明,ExSms 会在使用者没有察觉的情况下进行短信发送这样的敏感行为.

2.3 相关定义

本文所提出的安卓应用安全性检测方法是基于敏感路径(sensitive path,简称 SeP)识别的,敏感行为(sensitive behavior,简称 SeB)和敏感触发(sensitive activation,简称 SeA)是构成敏感路径的两个要素.

- 敏感行为:系统对使用者敏感信息进行相关操作的行为,可分为两类:(1) 需要向系统申请使用许可的 API 函数,安卓系统提供了许多与信息发送、打电话、网络链接、应用安装、外设使用相关的 API,在应用安装时,会向系统申请自身所包含的这些 API 的使用许可,使用者同意后才会继续安装流程;(2) 与动态加载相关的函数,越来越多的恶意应用采用了动态加载技术来躲避杀毒工具的检测,应用本身不包含执行恶意行为的代码,在应用运行时才会加载恶意代码来执行恶意行为;
- 敏感触发:触发应用执行敏感行为的函数,有以下两种情况:(1) 若一条包含敏感行为的执行路径中不包含需要与使用者交互的函数(如 *OnClick*),则该敏感行为对应的敏感触发是该路径的进入点函数(没有被其他函数调用的函数);(2) 若一条包含敏感行为的执行路径中包含需要与使用者交互的函数,则该敏感行为对应的敏感触发是该执行路径中最后出现的需要与使用者交互的函数;

- 敏感路径:该路径是以触发敏感行为为目标的执行路径,该路径的生成需要对安卓应用的 APK 文件进行静态分析,构建函数调用关系图 CG 并扩充为组件间调用关系图 ICCG(inter-component communication graph),进而通过 Intent 过滤,最终识别出敏感路径.具体处理过程见第 3.1 节;
- 标签敏感触发(labeled sensitive activation,简称 LSeA):由于 ICCG 只包含应用内的组件间调用关系,并不包含应用与系统乃至与其他应用间的交互,因此引入标签敏感触发,用于标识安卓系统与应用内组件的交互关系.标签敏感触发完善了对应敏感行为的触发信息,在系统事件与敏感行为之间建立了联系,包含系统事件的敏感路径能够更直观地反应敏感行为在何种情况下被触发.

3 安全性分析

我们设计的基于敏感路径识别的安卓恶意应用安全性分析方法可分为静态分析过程和机器学习过程两部分.其中,静态分析过程即对应用进行敏感路径分析的过程;机器学习过程则是对敏感路径信息进行处理抽象以及使用机器学习算法进行安全性判断的过程.

3.1 敏感路径分析

安卓系统使用了 Intent 机制来进行组件间、应用间、系统与应用间的通讯,因此需要分析应用所使用的 Intent,得出组件间调用关系 ICC,并需要将 CG 扩充为 ICCG,才能完整表达应用的函数调用关系.

安卓系统提供了 Intent 机制来协助组件的通信,它是一种运行时绑定(run-time binding)的机制.使用 Intent 机制完成组件间的通讯过程如下:组件 *A* 向安卓系统发送了 Intent 参数 *i*,安卓系统会根据 *i* 的属性内容选择合适的、能处理 *i* 的组件 *B*,*C*,…来完成 *A* 的请求.例如,某个 *Activity* 组件需要拨打电话,该活动组件会向系统发送 ACTION_VIEW,安卓系统会根据该 Intent 的需求找到拨号应用的对应组件来完成拨打电话的操作.

Intent 包含 ACTION,DATA,CATEGORY,COMPONENT 等几种属性,具体用途和释义如下.

(1) ACTION,表示 Intent 需要完成的动作.安卓系统提供了一些标准的 ACTION 常量,如 ACTION_VIEW 动作表示显示数据,会根据数据的具体类型,调用相应的组件进行后续操作;
(2) DATA,表示执行动作的数据对象,由一个 URI 变量表示;
(3) CATEGORY,表示能处理该 Intent 的组件所属的种类,一个组件可处于多个 CATEGORY 下,例如,CATEGORY_HOME 表示能够回到 HOME 界面的 *Activity* 组件;
(4) COMPONENT,表示该 Intent 的目标组件.有两种方式来寻找 Intent 参数的目标组件:一种是显式方式,Intent 参数的 COMPONENT 属性不为空,可以直接与目标组件建立联系;一种为隐式方式,Intent 参数的 COMPONENT 属性为空.

一个应用的所有组件能够处理的 Intent 参数的属性均注册在应用的 Manifest 文件中,系统需要在其中寻找与 Intent 参数的 ACTION,CATEGORY,DATA 属性相匹配的目标组件.各组件注册在 Manifest 文件中能够处理的 Intent 参数的列表被称为 Intent 过滤,包含了各组件的 ACTION,DATA,CATEGORY,COMPONENT 信息.

在此基础上,生成 ICC 的方法流程图如图 3 所示.

首先,需要分析函数中所定义的 Intent 参数,查看该 Intent 参数的 Component 属性是否为空.

- 如果不为空,Component 的属性值即是目标组件名,则在该函数与目标组件间建立调用关系;
- 如果为空,则进行 Action 属性的适配,分析在 Intent 过滤中组件注册的 Intent 参数的 Action 属性值.
  - ➢ 若完全适配,则建立调用关系;
  - ➢ 如果不适配,则筛选出 Intent 过滤中所有 Action 属性为空的组件,进行 Category 属性的适配:
    - ✧ 若完全适配,则建立调用关系;
    - ✧ 如果不适配,则在上一轮选出的组件中过滤出所有 Category 属性为空的组件,进行 Data.scheme 属性的适配:若完全适配则建立调用关系;如果还是不存在完全适配的目标组件,则该 Intent 参数无法被传向任何组件.

在得到应用的 ICC 调用关系后,可以将函数调用关系图 CG 扩充成组件间调用关系图 ICCG.过程如下:由

FlowDroid工具生成的CG包含一个虚拟的*main*函数节点*DummyMainMethod*(),由于该函数会影响之后的分析,将其删除;将ICC中包含的组件间调用关系作为有向边,添加至删除虚拟节点的CG中即可.

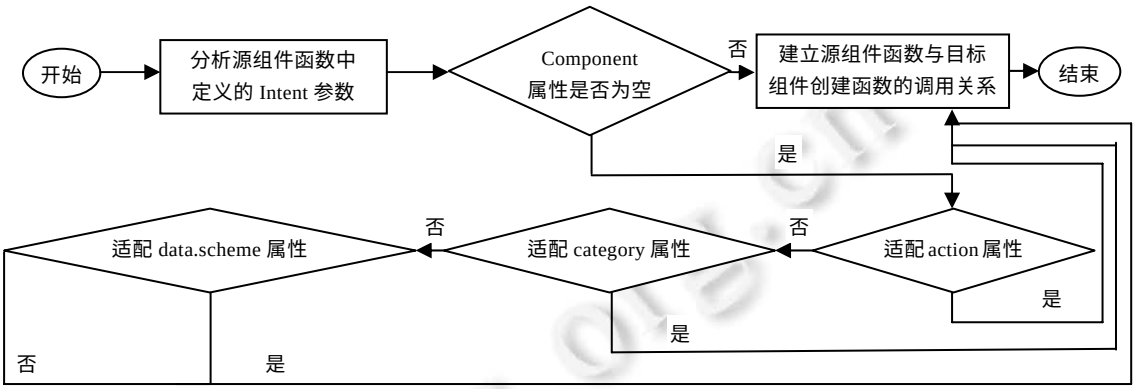


Fig.3 Process graph of generating ICC

图 3 ICC 生成流程图

得到ICCG后,进行敏感路径识别的过程见算法1.具体解释如下.

- 第1行、第2行,根据敏感行为的定义,在ICCG的节点中寻找代表敏感行为的函数节点.本文中,敏感行为是需要允许请求的API函数和与动态加载相关的函数;
- 第3行、第4行,针对每个敏感行为遍历ICCG,找到每条包含该敏感行为的执行路径.由于代表敏感行为的节点不会再调用其他函数,所以这些执行路径中不会再包含其他敏感行为;
- 第5行~第11行,分析每条包含敏感行为的执行路径,如果其包含与用户交互相关的函数,找到执行路径距离敏感行为节点最近的代表用户交互函数的节点,将该节点与敏感行为节点组成的敏感路径加入敏感路径的集合;
- 第12行、第13行,如果当前分析的执行路径中不包含与用户交互相关的函数,则将该条路径的进入点与敏感行为节点组成的敏感路径计入敏感路径的集合;
- 第14行~第18行,得到该应用包含的所有路径后,针对该应用的每个敏感触发,在Intent过滤中寻找敏感触发所属组件能够处理Intent参数的ACTION属性,将ACTION属性与敏感触发组成的标签敏感触发加入集合;
- 第19行、第20行,输出该应用所包含的敏感路径集合以及标签敏感触发集合.

算法 **1**. 敏感路径识别算法.

Inputs: ICCG: Inter Component Call Graph;

IF: A set of intent filters;

OutPuts: SeP: A set of sensitive paths;

LSeA: A set of labeled sensitive activations.

```
1  SeP←∅
2  SeB←getSensitiveBehavior(ICCG)
3  for each b in SeB
4     P←findExcutivePath(b,ICCG)
5     for each path∈P
6        a←findCaller(b)
7        while isNotEntryPoint(a)
8           a←findCaller(a)
```

```
9            if isUserIntractionFunction(a) then
10               SeP.add(a,b)
11               break
12           if isEntryPoint(a) then
13               SeP.add(a,b)
14 LSeA←∅
15 SeA←getSensitiveActivation(SeP)
16 for each a∈SeA
17     l←getAction(a,IF)
18     LSEA.add(l,a)
19 return LSeA
20 return SeP
```

3.2 特征抽象

一个安卓应用中包含了数量巨大的敏感行为和敏感触发,若将每一个敏感行为和敏感触发均视做特征,则特征种类数量过多,会造成特征矩阵过于稀疏,分析时间过长,影响准确率.因此,我们需要对敏感触发和敏感行为进行特征抽象,即:约减特征数量,使得特征的读写更加便捷.

敏感触发事件可分为硬件触发、用户触发以及系统触发这 3 大类.

- 硬件触发是指由手机或便携设备的硬件所产生的事件,诸如手机的锁屏键、音量键、内置的陀螺仪等硬件设备;
- 用户触发是指需要用户与触摸屏交互才会发生的事件,如单击、双击、长按等操作;
- 系统触发则是指系统自发产生的事件,包括两大类:一类是组件生命周期中各个阶段的常规事件;还有一类是 BroadCast Receiver 组件所监听的系统事件.

表 1 列出了本文处理的部分敏感触发函数,并对这些敏感触发函数进行了抽象表达.

Table 1 Abstraction of sensitive activation

表 1 敏感触发的抽象

所属种类	原始表达	抽象表达	详细
硬件触发	*boolean onKey*( )	KEY	按键事件
	boolean onKeyLongPress( )		
	boolean dispatchKeyShortcutEvent( )		
	boolean dispatchTrackballEvent( )	OTHERHARDWARE	轨迹球事件
	…		
用户触发	*void onClick*( )	CLICK	用户单击
	void onClickCancel( )		
	void StartbuttonClicked( )		
	boolean onLongClick( )	LONGCLICK	用户长按
	boolean onDoubleTap( )	DOUBLECLICK	用户双击
	boolean onDoubleTapEvent( )		
	void btZoomInClick( )	ZOOMIN	用户缩小
	void btZoomOutClick( )	ZOOMOUT	用户放大
	…		
系统触发	*void onCreate*( )	CREATE	组件创建
	void onPause( )	PAUSE	组件暂停
	void onStop( )	STOP	组件停止
	void onDestroy( )	DESTROY	组件销毁
	void onRestart( )	RESTART	组件重启
	void onResume( )	RESUME	组件复位
	void onReceive( )	详见表 2	
	…		

(1) 硬件触发.其中比较常见的是按键事件,在用户实际使用的过程中,诸如锁屏键之类的按钮使用频率相当高,很有可能被选为恶意功能的触发事件.还有诸如轨迹球等一类硬件设施,由于目前的手机使用到这些功能,将其作为 OTHERHARDWARE 抽象表达;

(2) 用户触发.同样发送信息的敏感行为,一个是不通知用户、直接通过后台监听系统事件触发,另一个是告知用户并由用户点击发送按钮来发送.显而易见,前者是恶意行为,后者是良性行为.是否由用户触发,是将良性行为与恶意行为区分开的主要判断依据.因为现实中存在用 UI 文字迷惑用户、诱骗用户进行操作但实际上却是激活恶意功能的情况,所以需要进一步将用户不同的操作手势区分开来;

(3) 系统触发.组件生命周期的各阶段可能会触发各种敏感行为,考虑到恶意应用的家族性(恶意应用被划分为多个种类,每个种类内的恶意应用从恶意功能的激活到恶意功能的执行都是类似的),将每个阶段的函数分别抽象.另外,根据标签敏感触发中的标签,针对基于监听系统事件的触发事件进行抽象,具体见表 2.

Table 2 System activation and abstraction based on monitoring system events

表 **2** 基于监听系统事件的系统触发及抽象

所属类别	抽象描述	原始描述	所属类别	抽象描述	原始描述
系统启动	BOOT	BOOT_COMPLETED	短信	SMS	SMS_RECEIVED
电话	CALL	PHONE_STATE NEW_OUTGOING_CALL ...	网络链接	NET	CONNECTIVITY_CHANGE DATA_STATE PICK_WIFI_WORK ...
电池电源	POWER	BATTERY_LOW BATTERT_OKAY ...	其他事件	OTHER	SIG_STR SIM_FULL USER_PRESENT ...

本文所提到的敏感行为可分为两大类:一类是动态加载相关的函数,一类为需要 Permission 权限才可以使用 API 函数.在本文中,前者被描述为 DynamicLoad,后者根据其所需的 Permission 对应至 Permission 描述.

表 3 列出本文对 Permission 的种类划分,大类里再按照功能以及敏感程度给予了进一步的划分.

(1) ReadInfo.与读取信息相关的 API 函数,根据其读取信息的不同,还可以细分为以下 3 种:AccountInfo,读取账户信息的行为,包括手机账户、Gmail 账户等信息;SystemInfo,读取系统信息的行为,不仅包括系统本身的配置信息,还包含系统内正在运行的进程列表等信息,例如 GET_TASK 所对应的函数;UserInfo,读取用户信息的行为,包括浏览器书签、历史记录等,也包含信息文本这种敏感程度较高的信息;

(2) Call.与拨打电话相关的 API 函数,CALL_PHONE,PROCESS_OUTGOING_CALLS 下所对应的敏感行为被划分至此类.如今已鲜有恶意应用执行拨打电话的恶意行为,但由于其敏感级别较高,特将其单列为一类;

(3) UseHardware.使用硬件设备(不包含触摸屏)的 API 函数,使用硬件设备很容易被用户察觉,但也是可能的安全隐患.根据设备所执行功能的不同,可以分为 4 类:BlueTooth,蓝牙设备;Camera,摄像头;NFC,较少手机配备的硬件设备,但是考虑到 NFC 可以读取交通卡等,将其单独列出;OtherHardware 包括震动马达、闪光灯等,考虑其对电池的损耗,将其列为一类;

(4) AccessLocation.能够获取手机所在位置的 API 函数,手机的位置也代表了用户的位置,用户在实际使用应用时,也相当关注自己的位置信息是否暴露;

(5) ChangeNetConfiguration.能够改变手机网络配置的 API 函数,其中既包括与 wifi 配置相关的函数,也包括与手机所使用的蜂窝网络相关的函数.连接网络是恶意应用执行多种恶意功能的必要条件,大部分恶意应用在执行恶意功能时,会使用到与网络相关的函数;

(6) SendMessage.与发送短信相关的 API 函数,私自发送短信也是常见的恶意功能之一,在进行安全性检测时,该类行为是需要着重监测的敏感行为;

(7) SystemOperation.对系统进行操作的函数,可分为 5 类:Data,更改数据的行为,如清除应用的缓存数据

CLEAR_APP_CACHE;DeviceConfiguration,更改设备配置的行为;ProcessOperation,操作进程的敏感行为;UserConfiguration,更改用户个性化配置的行为;OtherSystemConfiguration,无法划分至其他类的函数;

(8) WriteUserInfo.修改用户信息的函数.

Table 3 Abstraction of sensitive behaviors

表 3 敏感行为的抽象

Permission 权限/抽象描述 (敏感级别:低)	功能/抽象描述 (敏感级别:中)	类别/抽象描述 (敏感级别:高)
GET_ACCOUNTS, ...	AccountInfo	ReadInfo
GET_TASKS, READ_PHONE_STATE, ...	SystemInfo	
READ_HISTORY_BOOKMARKS, READ_SMS, ...	UserInfo	
CALL_PHONE, PROCESS_OUTGOING_CALLS	Call	
BLUETOOTH, BLUETOOTH_ADMIN	BlueTooth	UseHardware
CAMERA, RECORD_AUDIO, ...	Camera	
NFC	NFC	
VIBRATE, ...	OtherHardware	
ACCESS_COARSE_LOCATION, ACCESS_FINE_LOCATION, ...	AccessLocation	
CHANGE_NETWORK_STATE, CHANGE_WIFI_STATE, ...	ChangeNetConfiguration	
SEND_SMS, ...	SendMessage	
CLEAR_APP_CACHE, ...	Data	SystemOperation
CHANGE_CONFIGURATION, ...	DeviceConfiguration	
RESTART_PACKAGES, BROADCAST_STICKY, KILL_BACKGROUND_PROCESSES, ...	ProcessOperation	
SET_TIME_ZONE, ...	UserConfiguration	
SERIAL_PORT, ...	OtherSystemOperation	
WRITE_CALENDAR, ...	WriteUserInfo	
DynamicLoad		

3.3 使用决策树进行安全性分析

经过上述静态分析和特征抽象过程,我们得到了应用所包含的敏感路径信息.为了体现应用特征选取的全面性,我们引入应用 Manifest 文件中所申请的 Permission 请求作为特征的补充.首先,我们构建应用特征矩阵.

应用特征的矩阵构成如图 4 所示.

- *PID* 为应用的唯一标识符,用于区别其他应用;
- *M* 为代表应用是否为恶意应用的标签,“是”标为 1,“否”标为 0,安全性分析所要做的就是根据特征来计算该标签值;
- RF_i 为敏感路径特征,应用包含 RF_i,“包含”标为 1,“不包含”标为 0;
- 允许请求特征 PF_j 类似于 RF_i.

在此基础上,我们采用 Weka 工具中的决策树算法 C4.5 来完成对应用安全性的分析,即:给定一个数据集,其中每个元组都能用一组属性值来描述(对应图 4 中的一条特征向量),每一个元组属于一个互斥的类别中的某一类,通过有监督的学习,找到一个从属性值到类别的映射关系(为 0 或为 1),并且这个映射能用于对新的类别未知的实体进行分类(判断一个应用为恶意或良性).实际操作流程包括:

- 输入应用特征矩阵 *M*:如果 *M* 不可被划分(*M* 内各个应用的特征取值无区别)或达到算法停止的条件(如决策树达到一定深度),即完成了决策树的构造;
- 如果 *M* 可被划分,计算 *M* 内每个特征 *F* 的信息增益率,选取信息增益率最大的特征 *f*;根据特征 *f* 的取

值不同,将 M 划分为不同的子矩阵;针对每个子矩阵,重复以上流程.

	应用 ID	是否恶意	敏感路径特征				允许请求特征			
	PID	*M*	RF_1	RF_2	…	RF_k	PF_{k+1}	PF_{k+2}	…	PF_n
	001	0	0	0	…	0	0	0	…	0
	002	0	0	0	…	1	0	0	…	0
	003	0	0	0	…	0	0	0	…	0
	004	0	1	0	…	0	0	0	…	0
	005	0	0	0	…	0	0	0	…	0
	006	0	0	0	…	0	0	0	…	0
	007	1	0	1	…	1	0	0	…	1
	008	0	0	0	…	0	0	0	…	0
	009	0	0	0	…	0	0	0	…	0
	010	0	0	0	…	0	0	0	…	0
	…	…	…	…	…	…	…	…	…	…
一条应用数据 ←	*m*	0	0	0	…	0	0	1	…	0
	…	…	…	…	…	…	…	…	…	…
	p−1	0	0	0	…	0	1	1	…	0
	p	0	0	0	…	0	0	0	…	0

Fig.4 Constitutions of Android Apps feature matrix

图 4 应用特征矩阵的构成

4 实验分析

4.1 实验环境

本文进行的所有实验均是使用处理器为 Intel Core i5-3470 3.20GHz、内存为 8G、操作系统为 Windows 7 专业版的计算机所完成.实验由 Java 语言所实现,使用的开发环境为 JDK7.0,使用的 IDE 为 Eclipse Kepler.静态分析部分的功能使用了基于污点分析的工具 FlowDroid 所得的部分结果,同时借助了安卓开发工具 APKTOOL 获取 APK 文件内的 Manifest 文件,机器学习部分使用了工具 Weka 中的 C4.5 决策树算法.

4.2 评价指标

本文的最终目的是判定一个安卓应用是否为恶意应用.M 为分析时输入的应用特征矩阵,应用的实际情况与本实验的判定结果会产生 4 种情况,如下所示.

(1) True Positive,良性应用被判定为良性应用,该类应用被记为 $TP(M)$;

(2) True Negative,恶意应用被判定为恶意应用,该类应用被记为 $TN(M)$;

(3) False Positive,恶意应用被判定为良性应用,该类应用被记为 $FP(M)$;

(4) False Negative,良性应用被判定为恶意应用,该类应用被记为 $FN(M)$.

根据以上 4 种判定结果,分别针对良性应用和恶意应用使用精度、召回率、F 度量来对实验效果进行评价,同时考察整体分析的准确率.

4.3 数据集

本文实验所使用的数据集共包含 493 个现实世界中存在的安卓应用,其中包含良性应用 342 个,绝大部分下载自 Google Play 应用商店,小部分来自于国内的第三方市场“豌豆荚”;恶意应用 151 个,来自于数据集 Drebin, Contagio,VirusShare 中的恶意应用.

因为时间、设备以及所使用工具 FlowDroid 的限制(使用 FlowDroid 分析过大的 APK 文件,会造成分析时长无法估计、内存溢出等问题),我们从 Google Play 上下载了约 400 个大小在 8MB 以下的 APK 文件用于分析.在后续的分析中,剔除了分析时间过长的 APK 文件,共留下 303 个 APK 文件.同时,我们选取了同样来自于 Google Play、大小在 8MB 与 15MB 之间的 14 个 APK 文件以及来自于“豌豆荚”的、大小在 10MB 以下的 25 个 APK 文件作为补充,以保证数据集大小以及来源的多样性.

恶意应用主要收集自数据集 Drebin、网站 Contagio Mini Dump 和网站 VirusShare.其中,Drebin 是由德国哥廷根大学的 Daniel Arp 等人所提供的数据集,共包含 5 560 个来自于 179 个不同恶意应用家族的恶意应用,这些应用是在 2010 年 8 月~2012 年 10 月之间由 the MobileSandbox 工程所收集.我们从中随机选择了 100 个恶意应用,其中 3 个应用由于文件太小无法生成函数调用关系图等信息被剔除.因为 Drebin 中包含的恶意应用产生时间较早且隐去了应用名、描述等信息,我们又从 Contagio Mini Dump 与 VirusShare 网站中选取了共 54 个恶意应用作为补充.

4.4 实验结果及分析

我们提出 3 个研究问题并通过实验寻求这些问题的答案.

- RQ1:使用本文所提出的方法进行安全性分析的准确率如何?相对于其他方法,准确度是不是更高?
- RQ2:我们提出了敏感级别的概念,不同敏感级别下的安全性分析准确率如何?
- RQ3:数据集中 APK 文件的大小会对实验结果造成影响吗?

表 4 选取了数据集中的 15 种应用(其中良性应用 10 种,恶意应用 5 种),记录了它们所包含的信息.表头内容分别为应用的名称、该应用是否为恶意应用、APK 文件的来源、APK 文件的大小、分析所花时间、包含的敏感行为数量(因为表格大小限制,只列出原始表达与低敏感级别表达的数量)、包含的敏感触发数量(只列出原始表达)和包含的敏感路径数量.其中,RR 代表敏感触发和敏感行为均为原始描述,RL 代表敏感触发为原始描述、敏感行为为低敏感级别描述,AL 代表敏感触发为抽象描述、敏感行为为低敏感级别描述.因为 Drebin 中的恶意应用均已抹去应用本身的名称信息,所以所有恶意应用均选择了从 Contagio 网站获取到的恶意应用.

Table 4 Statistics of Android Apps in our data set

表 **4** 数据集中部分应用所包含的信息

Name	Is Mal	DL	Size (KB)	Time (s)	SeB		SeA	Sensitive path		
					Raw	SeLL		RR	RL	AL
power battery	Yes	Contagio	548	40	19	3	3	24	7	2
GlamorousSmoke	Yes	Contagio	1 014	86	23	17	17	264	174	23
3dtimeclockticks	Yes	Contagio	1 294	7	9	3	3	22	5	4
fdhgkihrtjkibx	Yes	Contagio	2 456	1	5	1	1	5	1	1
assassins creed	Yes	Contagio	3 332	6	5	3	10	10	6	4
科学计算器	No	Wandoujia	589	13	0	0	0	0	0	0
Apps2SD Move	No	Google	823	27	15	6	9	61	28	26
Word2PDF	No	Google	2 929	359	19	7	18	97	92	78
Manga Browser	No	Google	3 470	1 522	53	32	77	3 572	1 305	121
Auto Comment	No	Google	3 827	1 362	34	24	89	2 660	872	241
Saida Loans	No	Google	4 289	2 283	45	33	104	2 919	859	316
SuperBeam	No	Google	5 212	1 626	44	17	85	2 899	1 108	273
Google Voice	No	Google	5 965	722	26	13	141	1 008	470	434
OpenVPN	No	Google	7 857	77	5	3	15	52	31	17
Speedtest	No	Wandoujia	15 609	912	65	39	91	2 841	1 716	218

(1) RQ1

基于敏感路径识别的安全性检测方法所得检测结果见表 5.

本表中所列出的实验中使用的敏感路径描述的构成为 Raw(SeA)与 Raw(SeB),同时我们列出了仅使用 PSCout 所提供的敏感 API[30]作为特征(即,在使用时需要对 Permission 权限进行检查的 API 函数)进行数据挖掘的检测结果.

由表 5 可知:本文所提供的安卓应用安全性检测方法的准确率为 97.97%,高于基于 API-Feature 的检测方法(90.47%);此外,本文方法在恶意应用和良性应用检测的精度、召回率、F 度量等方面均优于 API-Feature 方法.这表明考察敏感行为的触发行为(即敏感触发)能够帮助分辨敏感行为是否可能存在恶意,能够用于判断包含此类敏感行为的安卓应用是否是恶意应用.因此,我们的结论是:本文所提出的基于敏感路径识别的恶意应用检测方法相对于 API-Feature 方法具有较好的检测效果.

Table 5 Comparison of our method and API-Feature method

表 5 本文方法与 API-Feature 的比较

	API-Feature	Raw(SeA)&Raw(SeB)
TP	316	335
TN	130	148
FP	21	3
FN	26	7
Accuracy (%)	90.47	97.97
B_Precision (%)	93.77	99.11
B_Recall (%)	92.40	97.95
B_F-measure (%)	93.08	98.52
M_Precision (%)	83.33	95.48
M_Recall (%)	86.09	98.01
M_F-measure (%)	84.69	91.03

(2) RQ2

在本节中,我们进行了不同敏感级别描述下的敏感路径作为特征的实验,实验结果见表 6.

Table 6 Experiment results of considering sensitive paths as features under different sensitive levels

表 6 不同敏感级别描述下的敏感路径作为特征的实验结果

	RR	RL	RM	AR
TP	335	330	297	332
TN	148	137	121	148
FP	3	14	30	3
FN	7	12	45	10
Accuracy (%)	97.97	94.72	84.78	97.36
B_Precision (%)	99.11	95.93	90.83	99.10
B_Recall (%)	97.95	96.49	86.84	97.08
B_F-measure (%)	98.52	96.21	88.79	98.08
M_Precision (%)	95.48	91.95	72.89	93.67
M_Recall (%)	98.01	90.73	80.13	98.01
M_F-measure (%)	96.73	91.33	76.34	95.79

敏感触发固定使用原始描述,我们分别使用原始描述 RR、低敏感级别描述 RL、中敏感级别描述 RM 的敏感行为作为特征进行实验.可以看到:原始描述和低敏感级别描述的结果相差不大,但是由于在抽象过程省略了一些信息,导致低敏感级别描述的准确率(94.72%)略低于原始描述的准确率(97.97%);而使用中敏感级别描述的准确率(84.78%)远低于前两者,但仍在可以接受的范围.

敏感行为固定使用原始描述,分别使用原始描述 RR、抽象描述的敏感触发 AR 作为特征.由于敏感触发的数量较少,抽象过程造成的信息损失也较少,相对于原始描述,抽象描述的实验结果 AR 多了 3 个 *FN* 的结果.

因此,通过对比不同敏感级别描述的特征进行安全性分析,实验结果表明:特征描述的抽象虽然赋予了敏感路径易读性,但是带来了部分信息的缺失.例如,同样在实际使用时需要申请 Permission 权限 SEND_SMS 的 API 可以是 *sendDataMessage*(),*sendMultipartTextMessage*()和 *sendTextMessage*().观察数据集中良性应用和恶意应用使用这 3 种 API 的状况可以发现:恶意应用使用的均是 *sendTextMessage*(),而良性应用后两者皆有,而 *sendDataMessage*()则没有被使用.若未进行特征抽象,则部分良性应用可在由这些敏感行为构成的敏感路径特征下与恶意应用形成差别;若统一抽象为 SEND_SMS,则这些差别就消失了,无法区分出良性应用和恶意应用,造成误报或漏报.

(3) RQ3

实验中我们发现,数据集中应用的平均大小会影响实验的结果.将数据集中的应用按照 APK 文件的大小分为两个子数据集:一个子数据集中包含大小为 0~4MB 的应用,共 374 个应用;另外一个子数据集中包含大小为 4MB~15MB 的应用,共 119 个应用.分别对这两个子数据集进行实验,所使用的敏感路径描述均为原始描述,实验结果见表 7.

Table 7 Experiment results with sub-data set

表 7 在子数据集上进行实验的结果

	0~4MB	4~15MB	0~15MB
TP	239	96	335
TN	133	15	148
FP	0	3	3
FN	2	5	7
Accuracy (%)	99.46	93.28	97.97
B_Precision (%)	100.00	96.97	99.11
B_Recall (%)	99.17	95.05	97.95
B_F-measure (%)	99.58	96.00	98.52
M_Precision (%)	98.52	75.00	95.48
M_Recall (%)	100.00	83.33	98.01
M_F-measure (%)	99.25	78.95	96.73

我们发现:在 0~4MB 的子数据集上,本文方法取得了更好的效果;而对于 4MB~15MB 的子数据集,其准确率略低于在 0~4MB 的子数据集和整个数据集上进行实验的效果.由于 4MB~15MB 的子数据集中的应用数量较少,不足以全面地展现本文方法的效果,但是仍然可以看出数据集中应用的大小对实验结果有着一定的影响.

另外,实验中对数据集内的每个 APK 文件进行敏感路径分析平均用时 170s.针对不同大小的 APK 文件,其分析用时如下:0~4MB 大小的 APK 分析,平均分析用时 89s;4MB~8MB 大小的 APK 分析,平均用时 918s;8MB~15MB 大小的 APK 分析,平均用时 2 579s.考虑到数据集内大部分 APK 文件大小要小于现实中的 APK 文件,在实际应用中,需要进一步降低我们方法的分析时间.

5 总 结

本文提出了一种结合了静态分析与机器学习的基于敏感路径识别的安卓应用安全性分析方法.使用静态分析的方法获取应用的组件间函数调用关系图,并由此获取敏感路径信息,再将这些信息特征进行不同敏感级别的抽象化,并根据这些信息赋予应用特征,使用决策树完成安卓应用安全性的分析.通过多角度的分析和对比,本文所提出的方法在数据集上的表现优于其他方法,尤其是在使用较高敏感级别描述下敏感路径作为特征的实验中具备较高的准确率.本实验数据集中的应用大小主要集中在 8MB 以内,而现实世界中常用的应用大部分都在 10MB 甚至 20MB 以上.由于受限于所使用的 FlowDroid 工具,在本文的实验环境中,一旦需要分析较大 APK 文件时,会出卡死、内存溢出等问题.后期我们会逐步改善实验条件和工具,引入规模更大的安卓应用.

本文主要工作是对安卓应用进行了安全性分析.如今,越来越多新兴的恶意应用使用了动态加载等技术来规避检测工具的排查.通过添加对这些新恶意应用的关注,静态分析方法能从一定程度上解决问题,但是不能完全根除,因为静态方法无法获得动态运行时信息.因此,需要考虑使用动态分析方法.但由于安卓系统的事件触发机制(单个安卓应用可能会有上千个事件及其回调函数)和框架模型的复杂性,动态方法实施起来的难度非常大;另外,动态方法一次运行只能覆盖部分事件,存在覆盖率较低的问题.与此同时,现有安卓应用的规模越来越大,如何结合静态、动态分析方法有效地改进安全性分析的效率、提高检测的准确率,是未来工作的主要内容.

References:

[1] Qing SH. Research progress on Android security. Ruan Jian Xue Bao/Journal of Software, 2016,27(1):45−71 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/4914.htm [doi: 10.13328/j.cnki.jos.004914]

[2] Felt AP, Chin E, Hanna S, Song D, Wagner D. Android permissions demystified. In: Proc. of the ACM Conf. on Computer and Communications Security (CCS 2011). 2011. 627−638. [doi: 10.1145/2046707.2046779]

[3] Yang W, Xiao XS, Andow B, Li S, Xie T, Enck W. AppContext: Differentiating malicious and benign mobile App behaviors using context. In: Proc. of the 37th Int'l Conf. on Software Engineering (ICSE 2015). 2015. 303−313. [doi: 10.1109/ICSE.2015.50]

[4] Li L, Bartel A, Bissyand´e TF, Klein J, Traon YL, Arzt S, Rasthofer S, Bodden E, Octeau D, McDaniel P. IccTA: Detecting inter-component privacy leaks in Android Apps. In: Proc. of the 37th Int'l Conf. on Software Engineering (ICSE 2015). 2015. 280−291. [doi: 10.1109/ICSE.2015.48]

[5] Li GZ, Han Z, Zhou QH, Wang YZ. A detecting system for Android malicious behavior based on binder information flow. NetInfo Security, 2016,(2):54−59 (in Chinese with English abstract). [doi: 10.3969/j.issn.1671-1122.2016.02.009]

[6] Avdiienko V, Kuznetsov K, Gorla A, Zelle A, Arzt S, Rasthofer S, Bodden E. Mining apps for abnormal usage of sensitive data. In: Proc. of the 37th Int'l Conf. on Software Engineering (ICSE 2015). 2015. 426−436. [doi: 10.1109/ICSE.2015.61]

[7] Seo SH, Gupta A, Sallam AM, Bertino E, Yim K. Detecting mobile malware threats to homeland security through static analysis. Journal of Network and Computer Applications, 2014,38:43−53. [doi: 10.1016/j.jnca.2013.05.008]

[8] Huang JJ, Zhang XY, Tan L, Wang P, Liang B. AsDroid: Detecting stealthy behaviors in Android applications by user interface and program behavior contradiction. In: Proc. of the 36th Int'l Conf. on Software Engineering (ICSE 2014). 2014. 1036−1046. [doi: 10.1145/2568225.2568301]

[9] Elish K, Yao DD, Ryder BG. User-Centric dependence analysis for identifying malicious mobile Apps. In: Proc. of the IEEE Mobile Security Technologies (MoST 2012). 2012.

[10] Yang Y, Su PR, Ying LY, Feng DG. Dependency-Based malware similarity comparison method. Ruan Jian Xue Bao/Journal of Software, 2011,22(10):2438−2453 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/3888.htm [doi: 10.3724/SP.J.1001.2011.03888]

[11] Enck W, Octeau D, McDaniel P, Chaudhuri S. A study of Android application security. In: Proc. of the USENIX Security Symp. (SEC 2011). 2011. 21−37.

[12] Grace M, Zhou YJ, Zhang Q, Zou SH, Jiang XX. Riskranker: Scalable and accurate zero-day Android malware detection. In: Proc. of the Int'l Conf. on Mobile Systems, Applications, and Services (MOBISYS). 2012. 281−294. [doi: 10.1145/2307636.2307663]

[13] Burguera I, Zurutuza U, Nadjm-Tehrani S, Crowdroid: Behavior-Based malware detection system for Android. In: Proc. of the 1st ACM Workshop on Security and Privacy in Smartphones and Mobile Devices. 2011. 15−26. [doi: 10.1145/2046614.2046619]

[14] Blsing T, Batyuk L, Schmidt AD, Camtepe S, Albayrak S. An Android application sandbox system for suspicious software detection. In: Proc. of the 5th Int'l Conf. on Malicious and Unwanted Software (MALWARE). 2010. 55−62. [doi: 10.1109/MALWARE.2010.5665792]

[15] Zhou Y, Wang Z, Zhou W, Jiang X. Hey, you, get off of my market: Detecting malicious apps in official and alternative Android markets. In: Proc. of the 19th Annual Symp. on Network and Distributed System Security (NDSS 2012). 2012.

[16] Spreitzenbarth M, Schreck T, Echtler F, Arp D, Hoffmann J. Mobile-Sandbox: Combining static and dynamic analysis with machine-learning techniques. Int'l Journal of Information Security, 2014, 1−13. [doi: 10.1007/s10207-014-0250-0]

[17] Luo Y, Zhang QX, Shen QN, Liu HZ, Wu ZH. Android multi-level system permission management approach. Ruan Jian Xue Bao/Journal of Software, 2015,26(Suppl.(2)):263−271 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/15037.htm

[18] Britton W, Elish KO, Yao DF. Comprehensive behavior profiling for proactive Android malware detection. In: Proc. of the Information Security. 2014. 328−344. [doi: 10.1007/978-3-319-13257-0_19]

[19] Wu DJ, Mao CH, Wei TE, Lee HM, Wu KP. Droidmat: Android malware detection through manifest and API calls tracing. In: Proc. of the Asia Joint Conf. on Information Security (Asia JCIS). 2012. 62−69. [doi: 10.1109/AsiaJCIS.2012.18]

[20] Gascon H, Yamaguchi F, Arp D, Rieck K. Structural detection of Android malware using embedded call graphs. In: Proc. of the ACM Workshop on Artificial Intelligence and Security (AISEC). 2013. 45−54. [doi: 10.1145/2517312.2517315]

[21] Chakradeo S, Reaves B, Traynor P, Enck W. Mast: Triage for market-scale mobile malware analysis. In: Proc. of the ACM Conf. on Security and Privacy in Wireless and Mobile Networks (WISEC). 2013. 13−24. [doi: 10.1145/2462096.2462100]

[22] Aafer Y, Du W, Yin H. DroidAPIMiner: Mining API-level features for robust malware detection in Android. In: Proc. of the Int'l Conf. on Security and Privacy in Communication Networks (SecureComm). 2013. 86−103. [doi: 10.1007/978-3-319-04283-1_6]

[23] Arp D, Spreitzenbarth M, Hübner M, Gascon H, Rieck K. Drebin: Effective and explainable detection of Android malware in your pocket. In: Proc. of the 21th Annual Symp. on Network and Distributed System Security (NDSS 2014). 2014. [doi: 10.14722/ndss.2014.23247]

[24] Zhang XY, Zhang G, Shen LW, Peng X, Zhao WY. Similarity analysis of multi-dimension features of Android application. Computer Sceinece, 2016,43(3):199−205, 219 (in Chinese with English abstract). [doi: 10.11896/j.issn.1002-137X.2016.3.037]

[25] Kong DG, Cen L, Jin HX. AUTOREB: Automatically understanding the review-to-behavior fidelity in Android applications. In: Proc. of the 22nd ACM Conf. on Computer and Communications Security (CCS 2015). 2015. 530−541. [doi: 10.1145/2810103.2813689]

[26] Zhang M, Duan Y, Feng Q, Yin H. Towards automatic generation of security-centric descriptions for Android apps. In: Proc. of the 22nd ACM Conf. on Computer and Communications Security (CCS 2015). 2015. 518−529. [doi: 10.1145/2810103.2813669]

[27] Wang R, Feng DG, Yang Y, Su PR. Semantics-Based malware behavior signature extraction and detection method. Ruan Jian Xue Bao/Journal of Software, 2012,23(2):378−393 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/3953.htm [doi: 10.3724/SP.J.1001.2012.03953]

[28] Yang H, Zhang YQ, Hu YP, Liu QX. A malware behavior detection system of Android applications based on multi-class features. Chinese Journal of Computers, 2014,37(1):15−27 (in Chinese with English abstract). [doi: 10.3724/SP.J.1016.2014.00015]

[29] Zhou Y, Jiang X. Dissecting android malware: Characterization and evolution. In: Proc. of the IEEE Symp. on Security and Privacy. 2012. 95−109. [doi: 10.1109/SP.2012.16]

[30] Au KWY, Zhou YF, Huang Z, Lie D. Pscout: Analyzing the Android permission specification. In: Proc. of the 2012 ACM Conf. on Computer and Communications Security. 2012. 217−228. [doi: 10.1145/2382196.2382222]

附中文参考文献:

[1] 卿斯汉.Android 安全研究进展.软件学报,2016,27(1):45−71. http://www.jos.org.cn/1000-9825/4914.htm [doi: 10.13328/j.cnki.jos.004914]

[5] 李桂芝,韩臻,周启惠,王雅哲.基于 Binder 信息流的 Android 恶意行为检测系统.信息网络安全,2016,(2):54−59. [doi: 10.3969/j.issn.1671-1122.2016.02.009]

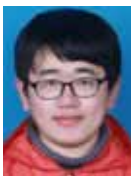
[10] 杨轶,苏璞睿,应凌云,冯登国.基于行为依赖特征的恶意代码相似性比较方法.软件学报,2011,22(10):2438−2453. http://www.jos.org.cn/1000-9825/3888.htm [doi: 10.3724/SP.J.1001.2011.03888]

[17] 罗杨,张齐勋,沈晴霓,刘宏志,吴中海.多层次的 Android 系统权限控制方法.软件学报,2015,26(Suppl.(2)):263−271. http://www.jos.org.cn/1000-9825/15037.htm

[24] 张希远,张刚,沈立炜,彭鑫,赵文耘.多维度的安卓应用相似度分析.计算机科学,2016,43(3):199−205,219. [doi: 10.11896/j.issn.1002-137X.2016.3.037]

[27] 王蕊,冯登国,杨轶,苏璞睿.基于语义的恶意代码行为特征提取及检测方法.软件学报,2012,23(2):378−393. http://www.jos.org.cn/1000-9825/3953.htm [doi: 10.3724/SP.J.1001.2012.03953]

[28] 杨欢,张玉清,胡予濮,刘奇旭.基于多类特征的 Android 应用恶意行为检测系统.计算机学报,2014,37(1):15−27. [doi: 10.3724/SP.J.1016.2014.00015]



缪小川(1990－),男,江苏南通人,硕士,主要研究领域为 Android 应用安全漏洞分析检测.

汪睿(1993－),男,学士,CCF 学生会员,主要研究领域为 Web 应用安全缺陷检测.

许蕾(1978－),女,博士,副教授,CCF 专业会员,主要研究领域为 Web 应用分析测试,Web 服务选择组合,并发缺陷检测.

张卫丰(1974－),男,博士,教授,CCF 专业会员,主要研究领域为软件分析测试,大数据处理.

徐宝文(1961－),男,博士,教授,博士生导师,CCF 会士,主要研究领域为程序设计语言,软件工程,并行与网络软件.