面向移动系统安全的应用分析技术综述①

郑冠仕[1,2,3], 张 铮[4], 张为华[1,2,3]

[1](复旦大学 软件学院, 上海 201203)
[2](复旦大学 上海市数据科学重点实验室, 上海 201203)
[3](复旦大学 并行处理研究所, 上海 201203)
[4](解放军信息工程大学 数学工程与先进计算国家重点实验室, 郑州 450001)

摘 要: 随着移动设备越来越普及, 针对移动系统的恶意软件也越来越多. 为了保障用户的隐私和财产安全, 国内外的专家学者针对现有的移动系统应用安全问题提出了许许多多的解决方案. 介绍了当前主流的移动操作系统, 并且对于近年来面向这些系统的应用安全分析技术进行了全面的分析和讨论. 在此基础上, 对于未来移动系统应用安全分析技术的发展方向进行了展望和总结.

关键词: 移动系统; 安全; 应用; 分析

Review of Application and Analysis Techniques for Mobile System Security

ZHENG Guan-Shi[1,2,3], ZHANG Zheng[4], ZHANG Wei-Hua[1,2,3]

[1](Software School, Fudan University, Shanghai 201203, China)
[2](Shanghai Key Laboratory of Data Science, Fudan University, Shanghai 201203, China)
[3](Parallel Processing Institute, Fudan University, Shanghai 201203, China)
[4](State Key Laboratory of Mathematical Engineering and Advanced Computing, PLA Information Engineering University, Zhengzhou 450001, China)

Abstract: The popularity of mobile system has attracted malware developers. To ensure the security of users' privacy and property, there has been numerous solutions targeting current security issues of mobile system. In this paper, it introduces the prevalent mobile operating systems and the mobile application security analysis methodologies. In addition, it makes analysis and discussion. At last, it proposes the prospects of possible directions of mobile application analysis approaches and endswith a conclusion.

Key words: mobile system; security; application analysis

移动设备通常是指小型的计算设备, 现有的移动设备主要包括智能手机、平板电脑、智能眼镜等. 移动设备因具有方便快捷的特性而为人们所青睐. 据统计, 在 2015 年的第三个季度, 全球移动设备的数量达到了 73 亿. 与此同时, 随着 IT 产业的迅猛发展, 面向移动设备的应用程序也越来越丰富. 到 2015 年 2 月为止, 仅仅是 Google Play 上应用的数量就达到了 140 万[1].

随着移动设备的普及, 尤其是随着越来越多的企业、政府等机构开始将移动设备投入常规使用, 针对移动系统的恶意软件也越来越多. 据统计, Google Play 上 3.15%的应用程序会泄漏用户的隐私信息或者有其他的恶意行为, 而目前国内最大的手机应用市场之一的 91 应用助手上的恶意软件比例则高达 19.70%[2].

当前移动系统的安全主要面临几个问题:

1) 用户承担的责任. 包括安卓在内的一些移动操作系统, 将授予应用程序权限的责任交给用户, 而用户很可能会因为缺少必要的专业知识而将一些重要权限授予了恶意应用;

2) 应用缺少可靠的下载途径. 移动设备应用数量庞大的一个直接结果就是它们往往没有各自的官方下载途径, 移动应用的下载一般是通过各式各样的应用市场. 如果应用市场对于应用的审查力度不够, 恶意

① 基金项目:国家自然科学基金(61370081);国家高技术研究发展计划(863)(2012AA010905)
收稿时间:2016-01-08;收到修改稿时间:2016-03-01 [doi:10.15888/j.cnki.csa.005354]

软件就有可能会混杂其中, 给用户带来威胁;

3) 风险更多, 资源更少. 移动系统比桌面系统面临着更多的风险, 因为移动系统上有着更多的用户隐私信息, 同时又因为它的计算资源有限, 很难像桌面系统那样部署强力的防护软件;

4) 系统破解. 一些用户为了获取一些原生系统没有提供的功能, 对系统进行了破解(如针对安卓操作系统的 root 和针对 iOS 操作系统的越狱), 给系统的安全造成了一定的威胁[3].

针对移动系统安全的问题, 有关专家学者做了大量的研究, 提出了许许多多的解决方案, 从静态的应用程序分析技术, 到动态的恶意行为检测技术, 再到加强防护的应用隔离技术等等. 这些解决方案根据其目的主要可以分为两类: 一类是针对移动应用程序进行分析检测; 一类是对移动系统的防护能力进行加强. 本文主要针对第一类研究, 尤其是近几年移动应用分析技术的主要研究进展进行讨论.

本文对当前移动应用安全分析技术的最新研究进展进行了分类总结并且给出了对比分析. 本文根据移动应用安全分析技术所针对的问题, 将其分为恶意行为分析, 应用缺陷分析和行为模式分析三类, 并对各类分析的相关研究进展进行了总结. 在此基础上, 本文讨论分析了各种移动应用安全分析技术的优劣. 最后, 本文对未来移动应用安全分析技术可能的发展方向进行了展望与总结.

本文的剩余部分组织结构如下: 首先对当前移动系统的发展情况以及恶意应用的发展情况进行介绍, 具体介绍了当前最主流的两种移动操作系统——安卓操作系统和 iOS 操作系统, 并且对其安全机制与存在的缺陷进行了讨论; 其次, 本文对当前移动应用分析技术的最新研究进展进行了介绍; 然后, 本文对于不同的移动应用分析技术的优点和不足进行了对比; 最后, 本文对移动系统应用安全分析技术的发展进行了总结和展望.

1 背景介绍

移动设备因具有方便快捷的特性, 同时又有着数量众多且功能强大的应用程序, 开始在越来越多的行业(如教育、医疗、通信等行业)中发挥作用. 移动设备的形式也开始变得多种多样, 除了常见的智能手机, 还以平板电脑、智能眼镜、智能手表、智能手环等形式出现. 据估计, 在 2015 年的第三个季度, 全球移动设备的数量达到了近 73 亿, 其中包括 8700 万新设备[1].

1.1 移动操作系统

对于移动设备而言, 操作系统是其最重要的组成部分之一, 其处于设备的硬件和应用软件之间, 为用户的各种应用软件提供运行环境和安全保护. 目前最主流的移动操作系统是谷歌公司的安卓系统和苹果公司的 iOS 系统.

安卓是一款由谷歌所开发的基于 Linux 内核移动操作系统, 其主要应用于智能手机和平板电脑. 2007 年 1 月, 由谷歌、HTC、索尼、三星等多家机构组成的开发手机联盟发布了安卓的第一个版本. 从 2007 年到 2015 年, 安卓操作系统不断地更新换代, 其支持的设备也从最初的智能手机和平板电脑发展到了包括智能手表、智能手环、数字摄像机在内的诸多电子设备. 目前, 安卓占领了智能手机市场中最大的份额. 2014 年, 全球搭载安卓操作系统的智能手机出货量超过 10 亿台, 并且在 2015 年将持续以两位数的百分比速度增长. 据估计, 在 2014 年, 安装安卓操作系统的智能手机占据全球智能手机总数的 75%[4].

iOS 是苹果公司专门为其包括 iPhone, iPad, iPod 在内的移动设备开发的操作系统. 2007 年, 苹果在 Macworld 大会上同时发布了 iPhone 和 iOS 的第一个版本, 到 2015 年 12 月为止, iOS 最新的版本是 iOS 9.2. 经过八年的更新发展, iOS 是现在最主流的移动操作系统之一. 截至 2015 年 2 月, 苹果公司的 App Store 上已经有超过 1400 万个 iOS 应用, 这些应用的下载次数总计超过 1000 亿[5].

1.2 移动操作系统的安全机制

安卓操作系统采用了沙盒机制和权限机制来保证系统和用户数据的安全. 沙盒机制通过利用 Linux 的访问控制和进程保护机制来实现应用的隔离和控制. 其具体实现是在每一个安卓应用程序在安装的时候都会有一个唯一的 Linux UID, 因此两个不同应用不可能会运行在同一个进程之中, 从而实现了隔离应用的目的. 安卓操作系统的权限机制通过限制应用调用敏感 API 来控制应用对于受保护资源的的访问.

尽管安卓系统采用了一定的安全机制来确保系统和用户数据的安全, 它还是存在着一定的安全缺陷:

1) 应用缺少可靠的下载途径. 安卓系统上的应用

大都是发布在各种各样的应用市场上, 这些应用市场对于上传到市场上的应用的审查力度大小不一;

2) 用户承担的责任. 安卓系统在安装应用时将一些权限的授予权交给了用户, 这相当于是把一部分保障系统安全的责任转移到了用户身上;

3) 系统缺陷. 安卓系统本身存在着一定的缺陷, 如安卓系统的系统升级模块[6], 多任务功能[7]都有可能会被恶意软件利用.

与安卓操作系统类似, iOS 操作系统采用了沙盒机制来隔离应用程序, 从而控制每个应用程序的行为. 和安卓操作系统不同的是 iOS 操作系统的所有应用都是发布在官方的应用市场 App Store 上, 所有上传到 App Store 上的应用都会经过苹果公司的审核, 确认应用程序符合苹果公司制定的安全规则. 同时, iOS 采用了签名机制来防止设备安装非官方来源的应用程序[8,9]. 此外, iOS 还采用了权限机制来限制应用程序对于设备资源的访问. 和安卓操作系统不同的是, iOS 中大部分权限的授予都由系统直接完成而不需要用户的参与[10].

苹果公司建立了应用审查制度和代码签名制度来确保用户安装软件的安全性. 然而, 苹果公司的应用审查制度并不能完全排除恶意应用上传到 App Store 上. 2009 年, 苹果公司发现 Storm8 开发的应用会收集用户的手机号码和其他隐私信息, 因而移除了 App Store 上所有 Storm8 的应用[8].

1.3 恶意软件的发展

移动设备相比于其他的电子设备与用户的关系更加紧密, 因而也保存着更多的用户隐私数据. 随着移动设备所占据的市场份额也越来越大, 针对移动设备的恶意软件也变得越来越多. 据统计从 2012 年 9 月到 2013 年 9 月的一年时间里, 针对安卓操作系统的已知恶意软件家族增加了 69%, 已知的恶意软件样本从 32,000 增加到了 273,000[11]. 2014 年, 仅美国而言, 针对安卓操作系统的恶意软件增加了 75%[12].

恶意软件的数量在不断增加的同时, 其破坏性也在不断增强. 2011 年, Googel Play 发现 DroidDream 感染了应用市场上将近 60 款应用, 受感染的用户数以百万计. DroidDream 能够破坏安卓应用的沙盒, 对安卓设备进行 root 操作, 窃取用户敏感信息然后发送到远程服务器[13]. 2015 年, 恶意软件 KeyRaider 感染了超过 22 万个苹果用户账户, 这是目前为止苹果公司历史上规模最大的恶意软件侵害事件. KeyRaider 针对的是越狱的 iPhone 设备, 它能够窃取用户的证书、私人密钥以及购物收据, 然后会将这些数据上传到服务器上[14].

除此之外, 恶意软件的感染途径也在变得更加的多样化, 这极大地增加了预防和检测恶意软件的难度. 2014 年, Lookout 的安全研究人员发现在销往亚洲和非洲一些地区的安卓手机上预装了 DeathRing——一个伪装成手机铃声软件的恶意应用程序. 当用户重启设备超过五次或者用户离线在线切换超过十五次就会激活 DeathRing. 激活后的 DeathRing 会偷偷地从服务器上下载 SMS 和 WAP 内容到用户的手机上, 然后再利用这些下载的内容来完成一些恶意操作. 例如, DeathRing 可以通过伪造短信内容来骗取用户的输入或者通过 WAP 内容来诱使用户下载其他恶意应用程序[15].

2 移动系统应用安全性分析

对于移动系统应用安全性的分析主要可以分为三类: 第一类分析是检测应用是否为恶意软件, 即该应用是否会执行一些恶意行为; 第二类分析是检测应用是否存在漏洞, 对于没有恶意行为的软件, 其存在的一些漏洞可能会被恶意软件利用, 从而对用户的隐私和财产造成威胁; 第三类分析是检测应用的行为模式, 主要是分析应用的行为是否存在安全隐患, 这些安全隐患的威胁通常比应用漏洞要低.

2.1 恶意行为分析

恶意行为分析主要是通过自动化半自动化的技术来检测应用程序是否会执行某种特定的恶意行为, 从而为用户提供应用程序安全性的依据. 恶意行为一般都包含两个步骤:先通过 UI 或者 API 获取敏感信息, 然后再通过日志、网络等途径将获取的信息泄漏出去. 按照分析目标的不同, 可以将对于恶意行为的分析分为三类: 针对应用 UI 的分析、针对应用申请和使用权限的分析以及针对应用泄露信息相关的分析.

2.1.1 UI 相关的分析

UI 是应用和用户之间的接口, 无论是获取信息(如查阅短信, 接听电话, 浏览网页), 还是发送信信息(如发送短信, 输入账号密码, 拨打电话)都是通过 UI 完成的. 正因为 UI 的关键性, 越来越多的恶意软件开始将其作为攻击的目标.

现今流行的移动系统都支持同时运行多个应用,

如用户可以在播放音乐的同时玩游戏. 用户主要是通过各个应用的视觉外观来辨识当前正在运行的应用. 当前主流移动操作系统的 GUI 都没有给出可靠的应用标识. 这就使得恶意应用有机会通过伪装成另外一个关键的应用(如支付宝)来骗取用户输入信息(如支付密码), 从而实现 GUI 混淆攻击. Bianchi 等人[16]提出了两种新的方式来防止安卓系统中 GUI 混淆攻击的发生: 第一种是部署在应用市场的防护, 主要是通过代码的静态分析技术来找出可能利用安卓 API 实现 GUI 混淆攻击的恶意应用; 第二种是部署在安卓设备上的防护, 主要是通过修改安卓的 GUI 来显示一个标识, 以便提醒用户当前正在交互的应用的真实身份.

除了可以通过伪装成其他应用的 GUI 直接获取用户的输入, 恶意软件还有可能窃取其他应用的输入. 为了保护敏感的用户输入, 南雨宏等人[17]提出了 UIPicker——一个能够自动识别安卓系统用户敏感输入的框架, 并且基于 UIPicker 实现了一个运行时保护系统, 该系统会在敏感信息离开设备的时候给用户发送提醒. UIPicker 能够有效地保护用户输入的隐私数据, 然而由于现有的大量安卓设备安装了不同版本的安卓系统, 要将这些设备的 UI 替换为 UIPicker 有较大的难度. 和 UIPicker 不同, SUPOR[18]能够检测应用程序中的敏感用户输入而不需要对安卓系统进行修改. SUPOR 结合了 UI 的渲染、布局分析以及自然语言处理技术来识别应用获取的用户敏感数据, 通过将用户输入的数据和应用中的变量相匹配并且对代码进行静态分析从而确定该应用是否会泄漏用户隐私信息. SUPOR 是第一项针对智能手机平台上的敏感用户输入的研究, 它能够检测出 97%以上的敏感用户输入. 之所以还存在较小比例的敏感输入无法检测出来是因为 SUPOR 对 UI 的布局分析存在一定的误判, 同时, 由于 SUPOR 对于文本的分析是上下文无关的, 也会造成一些误判, 导致分析结果的不准确.

2.1.2 权限相关的分析

安卓系统采用了权限机制来控制应用访问敏感资源. 然而安卓的权限机制的存在一定的缺陷使得恶意的安卓应用可以在安装的时候申请合法的权限, 在成功安装之后再设法直接或间接地获取更高的权限, 这也就是通常所说的权限扩大攻击(privilege escalation attack). Xing 等人[19]指出安卓系统上的恶意应用有可能利用安卓版本的更新来实现权限扩大攻击, 即 Pileup 攻击. 其攻击主要是通过在旧版本的安卓系统上声明一些在新的安卓系统中新增的 permission、permission tree、package name、sharedUId 等实现的. 在安装这些恶意应用的时候, 旧版本的安卓系统要么是无法识别应用申请的内容(如在新版本的安卓系统中才会有的 permission 或者 permission tree), 要么就是检查通过, 没有发现冲突(比如说恶意应用申请了新版本的安卓系统中新增加的系统应用的 package name 或者 UId), 因而完成了恶意应用的安装. 一旦安卓系统升级到新的版本, 在更新的过程中系统检测到新增加的系统应用的 permission、permission tree、package name、sharedUId 和已安装应用冲突. 为了保证系统更新前后用户数据的完整性和一致性, 安卓会采取保守的措施, 选择完全放弃或部分放弃新增加的系统应用的安装, 这就使得恶意应用能够完全替换新增系统应用程序或者替换新增系统应用的数据. 为了防止这种攻击, Xing 等人实现了 SecUP, SecUP 是一个安卓应用程序, 它能检测出系统中可能实行 Pileup 攻击的应用. SecUP 只会获取安卓系统的版本信息以及系统上安装的应用程序申请的 permission、permission tree、package name、sharedUId, 然后将获取的数据发送给服务器. 然后由服务器上的系统来完成主要的检测分析任务. 在保证了分析结果准确性的同时大大地降低了对于移动系统自身的影响.

安卓系统安全机制中的另外一个缺陷是它将部分权限的授予权交到了用户手上, 这其实是把一部分保障系统安全的责任转移到了用户的身上, 然而普通的用户可能由于缺乏对于安卓系统各种权限的相关知识而无法做出正确的抉择从而给恶意应用以可趁之机. 为了帮助用户做出正确的判断, Qu 等人[20]实现了 AUTOCOG. AUTOCOG 能够自动化地对应用的“描述-权限”可信度进行判断. “描述-权限”可信度主要是对比安卓市场的网页上对于应用的“描述”是否和应用实际申请的“权限”相符合, 即应用申请的权限是否都是其所描述的功能所需要的. AUTOCOG 结合了最新的自然语言处理技术和学习算法, 对文本模式和应用申请的权限进行匹配, 进而辅助用户做出决策. AUTOCOG 的不足之处在于: 一、其采用的模型属于非监督学习, 因而有可能会得出一些实际上并不存在的匹配结果; 二、无法防范一些恶意的开发人员提供的对于应用的错误描述. 和 AUTOCOG 类似, 诸姣等人[21]通过信息

检索和语义分析技术构建了安卓应用的功能描述和申请的权限之间的关系模型.

同样是利用应用申请的权限与其行为之间的对应关系, 杨欢等人[22]实现的 PApriori 算法能够根据应用申请的权限来判断一个应用是否为恶意应用. 首先, 通过静态分析的方法获取已知应用程序申请的权限信息, 构建权限集合; 然后, 通过数据挖掘技术分析每个恶意软件家族申请的权限之间的依赖关系, 构建权限关系库; 最后, 提取要分析的未知应用申请的权限, 并且跟权限关系库中的数据进行匹配, 从而判断该应用是否为恶意应用.

2.1.3 信息泄露相关的分析

移动设备上的应用通常会请求访问用户的隐私数据, 这就使得恶意软件有可能伪装成正常的应用, 窃取用户的隐私信息. 为了保护用户不受恶意软件的危害, 苹果公司为 iOS 系统引入了审核机制, 该机制会在应用发布在 App Store 之前确保其遵守苹果的隐私保护规则. 然而这一验证过程并不能充分过滤恶意软件, 之前出现过一些恶意应用, 直到有用户投诉才被苹果公司从 App Store 上移除. 针对这一问题, Egele 等人[8]实现了 PiOS, PiOS 能够通过静态分析来检测 Mach-0 二进制文件(从 Objective-C 代码编译成)中的数据流, 从而判断是否有隐私数据从移动设备流向了第三方. PiOS 的不足之处在于, 其缺乏应用的运行时信息, 因而在分析一些通过字符串变量确定的接收方的时候无法做出准确的判断, 进而影响了分析结果的准确性.

不同于 iOS 系统, 安卓系统为了保证较好的移植性, 在系统中采用了一些虚拟化技术(如 Dalvik 虚拟机). 这导致安卓系统中应用和系统之间、应用和应用之间的交互更为复杂, 增加了对安卓系统的应用进行静态分析的难度. DroidSafe 对安卓系统中应用程序的运行环境进行建模, 结合建立的模型来对安卓应用进行静态信息流分析, 从而提升了分析的准确性[23]. 对 94 个安卓应用进行分析的结果表明, DroidSafe 取得了比当前最好的安卓信息流分析技术(FlowDroid+IccTa)更高的准确度和精确度.

不同于文献[23, 8]采用的传统的数据流分析的方法, BAYESDROID 采用了一种基于统计学的方法来对恶意软件进行识别. BAYESDROID 将保护隐私视为一个可学习的问题, 运用贝叶斯原理, 基于已有的信息输出(指用户的信息通过网络、短信等途径离开了设备)的实例来判断当前信息输出是否合法[24].

同样是基于统计学的理论, 程际桥[25]采用了基于支持向量机的方法来检测安卓恶意软件. 该方法首先通过分析大量恶意和良性的安卓应用, 总结出恶意软件的行为特征; 然后通过对应用进行静态的权限分析, 排除一些不包含危险权限的应用程序; 最后, 对于包含危险权限的应用程序, 使用基于支持向量机的检测算法, 根据总结出的恶意软件行为特征, 检测分析对象是否有泄漏用户隐私等恶意行为. 张焕[26]实现的 Neighbor Watcher 系统则是通过对安卓应用程序的函数调用图进行聚类, 从而判断应用程序是否会窃取用户隐私或者有其他的恶意行为. 宋卫卫等人[27]实现的 RecEye 通过对安卓应用程序进行静态的信息流和控制流的分析, 提取应用程序的行为特征, 然后再利用 WEKA 数据挖掘系统来判断应用程序是否有恶意窃听的行为.

2.2 应用缺陷分析

应用缺陷分析主要是通过包括静态分析技术和动态分析技术在内的分析手段来找出应用程序中存在的一些漏洞. 这些应用程序并不会直接对用户造成危害, 但是其自身的漏洞可能会被恶意软件利用, 导致用户的隐私和财产安全受到威胁.

2.2.1 支付软件的缺陷

随着移动设备越来越普及, 一些互联网公司和金融机构开始推出移动端的钱包业务, 如国外的 Square、PayPal Here, 国内的支付宝、微信支付等. 这些支付软件在给用户带来便利的同时也有可能会带来一定的风险. Reaves 等人[28]对 46 个安卓钱包应用进行了分析并且着重分析了其中的 7 个应用. 分析结果表明 7 个应用中的 6 个应用存在着各种各样的系统缺陷——不完善的证书验证机制、不专业的加密机制以及其他形式的信息泄露. 这些缺陷的存在使得攻击者有可能冒充合法用户, 修改用户的交易以及窃取财务记录.

2.2.2 ADB(Android Debug Bridge)滥用

随着安卓系统的不断普及, 广大用户对于安卓系统的各种需求已经远远超过了其最初的设计所能提供的功能. 为了满足用户的需求, 安卓应用的开发人员寻求各种方法来实现用户需要的功能. 其中一种常见的方法是利用 ADB(Android Debug Bridge). ADB 可以连接安卓设备和 PC, 通过 ADB, 安卓系统中 JVM 层

的应用可以使用本地网络 socket 和本地的可执行文件进行通信. 普通安卓应用使用 ADB 的一种常见情况是为了实现截屏功能——安卓系统本身并不支持截屏功能, 因此普通应用只能通过 ADB 来实现截屏功能. Lin 等人[19]在研究中实现了一个恶意应用 Screenmilker, Screenmilker 可以通过利用 ADB 监视用户的屏幕, 窃取包括密码在内的隐私信息.

2.2.3 劫持相关的缺陷

安卓系统的多任务处理极大地增强了用户体验, 然而, 其存在的一些缺陷使得系统容易遭受部件劫持攻击[7]. 部件劫持是指一个应用没有正确地实现对于外部请求的访问控制, 从而将隐私数据或者权限泄漏给了恶意应用. 通常恶意软件会通过部件劫持来执行一些对于未授权的敏感数据的读写操作[29].

CHEX 通过对安卓系统中互相交错的程序执行进行建模并且对其中的数据流进行追踪, 同时对整个系统的变量依赖关系进行可达性分析, 从而找出可能会发生部件劫持的数据流, 定位应用中的漏洞[29]. CHEX 能够高效准确地检测出安卓应用的部件劫持漏洞, 用 CHEX 对 5,486 个应用进行分析, 发现了 254 个潜在的部件劫持漏洞, 分析每个应用所需时间的中位数为 37.02 秒. 不足之处在于 CHEX 并没有考虑安卓系统上部件之间执行的偏序, 同时它也无法识别出较为复杂的部件劫持逻辑, 因而存在一定的误报率.

尽管 CHEX 能够有效地找出恶意应用中的部件劫持漏洞, 这些漏洞的修复还是需要开发人员发布相应的补丁. 然而, 想要完全依靠应用的开发人员来修复应用的部件劫持漏洞并不现实, 一方面开发人员可能并没有足够的时间来确认每一个存在的漏洞并且发布相应的补丁, 另一方面, 开发人员可能也没有足够的经验来修复相关的漏洞. 针对这一问题, Zhang 等人[30]实现了 AppSealer. AppSealer 能够自动地为安卓应用中的部件劫持漏洞生成相应的补丁, 其工作流程主要分为三个阶段: 一、通过对应用的字节码进行分析确定导致漏洞的程序切片; 二、在程序切片中插入一些变量和指令, 从而实现对运行时敏感信息的追踪; 三、通过一系列的优化手段来移除冗余的指令, 从而尽可能地减小生成的补丁带来的影响. 用 16 个有部件劫持漏洞的应用对 AppSealer 进行测试, 结果表明, AppSealer 生成的补丁能够有效地保护应用, 同时生成的补丁带来的应用程序的运行时开销较小, 平均为 2%. AppSealer 的不足之处在于它采用静态分析的方法分析分析事件驱动的、面向对象的异步程序时, 不可避免地会产生一些假阳性的的错误.

2.3 行为模式分析

行为模式分析主要是通过对大量的应用程序进行分析, 挖掘应用中普遍存在的一些安全威胁或安全隐患.

2.3.1 代码加载

安卓系统允许应用在运行时加载外部代码, 这就使得恶意软件有机会在经过应用市场或者反病毒软件的检查之后加载恶意的代码, 此外, 普通应用的开发人员在使用代码加载功能的时候也有可能会引入一些漏洞. Poeplau 等人[31]采用静态分析技术对来自 Google Play 的 1,632 个应用进行了分析, 发现高达 9.25%的应用以不安全的方式加载代码, 这一比例在排名前 50 的应用中甚至高达 16%.

2.3.2 权限申请和使用

安卓系统根据应用程序在安装时申请的权限来限制其对用户隐私以及系统安全相关 API 的访问, 从而保证系统的安全. 然而, Stowaway[32]对 940 个安卓应用程序进行分析后, 发现有 1/3 左右的应用程序有过度申请权限的行为. 过度申请权限使用户面临了一些不必要的风险, 同时也扩大了应用的漏洞和缺陷可能造成的危害. Stowaway 的实现主要分为两部分, 一部分通过静态分析的方法来确定应用程序调用了哪些 API, 另一部分根据系统 API 和权限之间的映射来判断应用程序实际使用了哪些权限. 最后, Stowaway 再把应用程序实际使用的权限和它申请的权限进行对比, 判断其是否有过度申请权限. Stowaway 的不足在于它无法处理一些安卓程序的反射调用, 主要是因为无法确定一些基于非静态环境变量的方法名, 因而在判断应用程序使用的权限的时候会存在一定的误差.

VetDroid[33]对应用获取权限后如何访问和使用敏感资源和数据进行了分析. VetDroid 采用了动态分析技术来分析应用, 它会拦截所有应用程序对于安卓系统 API 的调用, 然后对比安卓系统的权限信息从而有效地构建应用的权限使用情况. VetDroid 对 Google Play 上的 1,249 个应用进行了分析, 结果表明, VetDroid 能够比 TaintDroid 发现更多的信息泄漏, 而且能够更精确地追踪信息泄露发生的原因. VetDroid 的不足之处在于它采用的动态分析技术开销较大, 会造

成平均 32%的时间开销以及 14%的内存开销.

2.3.3 用户位置信息的使用

随着移动设备用户安全意识的加强, 越来越多的人开始注意保护自己的位置信息. 而如何保证用户位置信息的安全也一直是学术界研究的主题. 之前已经有研究提出并且实现了一些针对用户位置隐私信息相关的保护机制, 而这些机制往往需要对应用或者系统进行修改, 这就使得它们对于用户而言不那么有吸引力. 为了提出有效且易为用户所接受的位置隐私信息保护机制, Fawaz 等人[34]首先对 Googel Play 的 1000 个免费应用进行了分析, 并且收集了超过 400 个位置敏感的应用和 70 个 AA(Advertisement and Analytics)库使用位置信息的具体情况. 分析结果显示, 70%的应用和 AA 库在使用用户信息的时候都存在一些隐患. 基于分析的结果, Fawaz 等人[34]实现了一个用户级的应用——LP-Doctor 来保护用户的位置信息. LP-Doctor 在每一个应用启动之前会预估启动应用可能造成的隐私威胁, 一旦发现有威胁, 它就会采取行动来保护用户的隐私. LP-Doctor 是一个用户级的应用, 它可以让用户通过改变保护的等级在隐私保护和设备的可用性之间做出权衡.

3 移动应用安全分析研究的分析

这一部分对比了 2 中提到的各项应用分析相关研究所采用的技术及其运行环境(表 1), 并且从兼容性、运行开销和程序执行路径的覆盖率三个方面对其进行了分析和比较(表 2). 完善的移动应用安全分析技术应该能够在不经过修改或者经过较少的修改的情况下应用于不同发行版本和定制版本的移动系统, 这就要求分析技术有着较好的兼容性. 与此同时, 考虑到移动系统的计算资源有限, 移动应用安全分析技术带来的运行时开销应该尽可能的低. 此外, 为了保证分析结果的准确性, 分析技术对于程序的执行路径应该有着较高的覆盖率.

表 1 各项应用分析研究采用的技术和运行环境

类别	研究	采用的技术	运行环境
UI 安全	Bianchi 等	静态分析	离线+在线
	UIPicker	UI 框架	在线
	SUPOR	静态分析	离线
信息泄露	PiOS	静态分析	离线
	DroidSafe	静态分析	离线
	BAYESDROID	静态分析	离线
权限分析	SecUP	静态分析+应用	离线
	AUTOCOG	静态分析+自然语言处理	离线
	Stowaway	静态分析	离线
	VetDroid	动态分析	在线
劫持相关的攻击	CHEX	静态分析	离线
	AppSealer	静态分析+动态分析	在线+离线

表 2 各项研究优缺点对比

类别	研究	兼容性	运行开销	覆盖率
UI 安全	Bianchi 等	中	低	高
	UIPicker	低	中	低
	SUPOR	高	低	高
信息泄露	PiOS	高	低	高
	DroidSafe	高	低	高
	BAYESDROID	高	低	高
权限分析	SecUP	高	低	高
	AUTOCOG	高	低	高
	Stowaway	高	低	高
	VetDroid	低	高	低
劫持相关的攻击	CHEX	高	低	高
	AppSealer	低	高	低

3.1 分析技术

对于应用程序的分析技术主要有静态分析和动态分析两种. 静态分析的优点是能够检测应用程序的代码而不需要执行它, 有着较高的代码覆盖率. 缺点是由于没有执行应用程序, 它缺乏一些应用程序执行的路径信息和上下文, 因而在检测的准确性上会有一些受损. 动态分析主要是在真实或者模拟的环境中执行应用程序, 进而对其执行情况进行分析. 动态分析的优点是它能够获取真实的程序执行信息, 因而有着较高的准确性. 其缺点是只能覆盖有限的程序执行路径[31].

在分析应用权限的申请和使用情况时, Stowaway 采用了静态分析的方法来追踪应用程序的调用的 API, 而 VetDroid 则是采用了动态分析的技术在程序运行时拦截所有的安卓 API 调用. Stowaway 能够保证较高的代码覆盖率, 但是由于 Stowaway 并不会执行程序, 因而会将一些不会被执行的死代码考虑进去, 造成了分

析准确性的下降[32]. 相比较而言, VetDroid 采用的动态分析技术能够精确地分析每一次的程序执行, 但是无法保证较高的程序执行路径覆盖率. 此外, 动态分析方法往往会对应用程序的性能有影响, 如用 VetDroid 对应用进行分析时, 平均应用的执行时间增加了 32%, 内存占用增加了 14%[33].

同样是对 UI 进行分析, Bianchi 等人采用的是静态分析的技术, 而 UIPicker 采用的是动态的运行时保护技术. Bianchi 等人采用的 UI 分析技术在最好的情况下能够检测出 76.34%的 UI 攻击, 而 UIPicker 则能检测出 90%以上的敏感用户输入, 然而, UIPicker 会带来一定的运行时开销.

完善的移动应用安全分析技术应该能够充分理解程序的语义, 尽可能多地覆盖程序可能的执行路径, 并且适当地结合程序的运行时信息, 保证分析的准确性.

在对应用程序进行分析的时候, 可以采取静态分析技术来提高对于程序执行路径的覆盖率, 同时适当地结合动态分析技术获取应用的运行时信息, 从而提升分析结果的准确性.

3.2 兼容性

一些针对移动应用的分析方案需要对系统已有的部分进行修改, 如 UIPicke, 如果想要采用 UIPicker 来保护用户输入的敏感信息, 则需要替换或者修改安卓系统已有的 UI 部件. 如何与当前数量庞大, 系统版本和定制版本众多的安卓系统相兼容将会是一个很大的问题. 此外, 如何保证用户已有的应用和数据不被破坏也是一个很大的问题.

相比之下, 同样是保护用户输入, SUPOR 有着比 UIPicker 更好的兼容性. 因为 SUPOR 采用的静态分析技术主要是对应用程序的 APK 文件进行分析, 这种分析在应用安装之前就可以进行, 因而无需对应用的运行环境进行修改[18].

除了采用静态分析方法, 以应用的方式提供的应用分析方案也有着较好的兼容性. 如 SecUp 将大部分的分析任务部署在服务器上, 安装到安卓系统上的应用只是通过 API 获取一些系统和其他应用的信息, 再将获取的信息发送给服务器进行分析[19].

由于现有的移动设备的数量已经非常庞大, 完善的移动应用安全分析技术应该有着较好的兼容性——应该能够在经过较小的修改或者不经过修改的情况下就能应用于不同发行版本和定制版本的移动系统.

要取得良好的兼容性, 在对应用进行分析的时候应该尽可能地减少系统相关的依赖. 如果需要应用程序的运行时信息, 可以考虑通过轻量级的动态分析技术或者安装的应用程序获取.

3.3 运行环境

对于移动系统应用程序的分析可以根据其运行环境分为在线和离线两类. 其中在线分析是指运行在移动设备之上, 需要实时收集应用运行信息的分析. VetDroid 采用的动态分析技术以及 UIPicker 实现的运行时的保护和分析都属于在线分析. 在线分析的特点是分析的结果更加准确, 因为它能够获取应用真实的运行信息. 其缺点是能够利用的移动设备上的计算资源较少, 而且会导致应用程序的运行时间和存储空间开销.

离线分析主要是指那些不需要运行在移动设备之上的分析, 这类分析一般在 PC 或者服务器上进行. 静态分析一般都可以归类为离线分析, 如 2 中提到的 PiOS 和 DroidSafe, 这类分析能够利用的计算机资源不受移动设备计算能力的限制, 因而有着较高的查全率. 但是由于分析的时候没有考虑到应用程序运行的系统之间的差异, 同时又缺少应用程序的运行时信息, 其分析结果的准确性会受到一定的影响[8,23].

完善的移动应用安全分析技术应该能够充分结合在线获取的应用运行时信息以及离线的计算资源. 在保证分析结果准确性的前提下, 尽可能地将计算任务迁移到离线的 PC 或者服务器. 如文伟平等人[35]实现的安卓恶意软件监测技术就较好地结合了在线和离线两种运行环境. 首先通过在线的客户端程序进行轻量级的扫描获取分析对象的数据, 然后将获取的数据传送给离线的服务器端, 服务器端的检测系统做进一步的检测分析.

为了取得准确的分析结果和良好的用户体验, 在对应用进行分析的时候, 可以通过离线的静态分析技术来提高对于应用程序执行路径的覆盖率, 同时结合轻量级的动态分析技术或者安装的应用程序来获取分析对象的运行时信息, 从而在提高分析的查全率和准确性的同时保证良好的用户体验.

4 展望

完善的移动系统应用分析技术需要有着较高的查全率和和准确性, 同时, 还要考虑到移动系统之间的

兼容性问题以及可利用的计算资源. 这就要求未来的移动应用的安全分析技术有着:

1) 更加精确, 扩展性更好的静态分析技术. 通过静态分析技术来充分地理解程序的语义, 提高分析时程序可执行路径的覆盖率;

2) 更加轻量级的程序运行时信息获取技术. 通过动态分析技术或者以应用的方式获取应用程序运行时的信息, 从而提高分析的准确性;

3) 更好的动静结合的分析方式. 静态分析和动态分析有其各自的优缺点, 两者能够形成良好的互补. 结合两种技术对应用进行分析能够取得更好的分析结果;

4) 更多的分析对象. 移动系统作为一个复杂的生态环境, 其可供分析的对象不仅仅是程序的代码. 如 SUPOR[18]在分析应用程序的时候就充分利用了应用的 Layout 文件来确定应用获取的用户输入是否敏感. 在对应用程序进行分析的时候, 如果能够充分挖掘除代码以外的可分析对象, 就能为分析过程提供更多的辅助信息, 从而取得更好的分析结果.

5 总结

当前移动系统已经融入了人们生活中的方方面面, 在以后的日子里, 随着移动系统计算能力的提升以及新的形式的移动设备的出现, 移动系统在人们的生活中会占据着越来越重要的地位. 然而, 如果移动系统的安全性没有得到充分的保障, 用户的隐私和财产就会遭受极大的威胁. 本文对近年来重要的面向移动系统的应用分析技术进行了深入的归纳分析, 指出了这些技术的优点和不足, 并且对未来移动应用安全分析技术的发展进行了展望和总结. 可以预见, 随着移动应用分析技术的发展, 移动系统的安全性将会得到更有力的保障.

参考文献

1 Ericsson. Ericsson Mobility Report. http://www.ericsson.com/res/docs/2015/mobility-report/ericsson-mobility-report-nov-2015.pdf. [2015-10-03].

2 Mike Murray. BSides Las Vegas: Your Droid Has No Clothes. http://blog.trustlook.com/2013/08/05/bsides-las-vegas-your-droid-has-no-clothes/. [2013-08-05].

3 Li Q, Clark G. Mobile security: A look ahead. Security & Privacy, IEEE, 2013, 11(1): 78−81.

4 Statista. Smartphone OS worldwide by installed base in 2014 (inmillions). http://www.statista.com/statistics/385001/smartphone-worldwide-installed-base-operating-systems/. [2015-12-17].

5 Ingraham N. Apple's App Store has passed 100 billion app downloads. http://www.theverge.com/2015/6/8/8739611/apple-wwdc-2015-stats-update. [2015-06-08].

6 Xing L, Pan X, Wang R, et al. Upgrading your android, elevating my malware: Privilege escalation through mobile os updating. 2014 IEEE Symposium on Security and Privacy (SP). IEEE. 2014. 393−408.

7 Ren C, Zhang Y, Xue H, et al. Towards discovering and understanding task hijacking in android. Proc. of the 24th USENIX Conference on Security Symposium. USENIX Association. 2015. 945–959.

8 Egele M, Kruegel C, Kirda E, et al. PiOS: Detecting privacy leaks in iOS applications. NDSS. 2011.

9 Wang T, Lu K, Lu L, et al. Jekyll on iOS: When benign apps become evil. Usenix Security. 2013, 13.

10 Mohamed I, Patel D. Android vs iOS security: A comparative study. 2015 12th International Conference on Information Technology-New Generations (ITNG). IEEE. 2015. 725–730.

11 Uscilowski B. Mobile Adware and Malware Analysis Report. http://www.symantec.com/content/en/us/enterprise/media/security_response/whitepapers/madware_and_malware_analysis.pdf. [2015-12-17].

12 Lookout. 2014 Mobile Threat Report. https://www.lookout.com/resources/reports/mobile-threat-report. [2015-12-17].

13 Lookout. Security Alert: DroidDream Malware Found in Official Android Market. https://blog.lookout.com/blog/2011/03/01/security-alert-malware-found-in-official-android-market-droiddream/. [2013-03-01].

14 Xiao C. KeyRaider: iOS Malware Steals Over 225,000 Apple Accounts to Create Free App Utopia. http://researchcenter.paloaltonetworks.com/2015/08/keyraider-ios-malware-steals-over-225000-apple-accounts-to-create-free-app-utopia/. [2015-08-30].

15 Lookout. DeathRing: Pre-loaded malware hits smartphones for the second time in 2014. https://blog.lookout.com/blog/

2014/12/04/ deathring/. [2014-12-04].

16 Bianchi A, Corbetta J, Invernizzi L, et al. What the app is that? Deception and countermeasures in the android user interface. 2015 IEEE Symposium on Security and Privacy (SP). IEEE. 2015. 931−948.

17 Nan Y, Yang M, Yang Z, et al. Uipicker: User-input privacy identification in mobile applications. USENIX Security. 2015. 993–1008.

18 Huang J, Li Z, Xiao X, et al. SUPOR: Precise and scalable sensitive user input detection for android apps . Proc. of the 24th USENIX Conference on Security Symposium. USENIX Association. 2015. 977–992.

19 Xing L, Pan X, Wang R, et al. Upgrading your android, elevating my malware: Privilege escalation through mobile os updating. 2014 IEEE Symposium on Security and Privacy (SP). IEEE. 2014. 393–408.

20 Qu Z, Rastogi V, Zhang X, et al. AutoCog: Measuring the description-to-permission fidelity in Android applications. Proc. of the 2014 ACM SIGSAC Conference on Computer and Communications Security. ACM. 2014. 1354–1365.

21 诸姣,李宏伟,彭鑫,等.安卓应用系统的功能与权限相关性研究.计算机应用与软件,2014,31(10):27–33.

22 杨欢,张玉清,胡予濮,等.基于权限频繁模式挖掘算法的 Android 恶意应用检测方法.通信学报,2013,34(1): 106–115.

23 Gordon MI, Kim D, Perkins J, et al. Information-flow analysis of Android applications in DroidSafe. Proc. of the Network and Distributed System Security Symposium (NDSS). The Internet Society. 2015.

24 Tripp O, Rubin J. A Bayesian approach to privacy enforcement in smartphones. USENIX Security. 2014.

25 程际桥.基于支持向量机的 Android 恶意软件检测方法[学位论文].武汉:华中科技大学,2014.

26 张焕.安卓平台下恶意软件的检测[学位论文].济南:山东大学,2014.

27 宋卫卫,杨哲憨,杨珉.RecEye:一种针对安卓窃听程序的检测方法.小型微型计算机系统,2015,36(6):1276–1282.

28 Reaves B, Scaife N, Bates A, et al. Mobile money, mobile)problems: Analysis of branchless banking applications in the developing world. 24th USENIX Security Symposium (USENIX Security 15). USENIX Association. 2015.

29 Lu L, Li Z, Wu Z, et al. Chex: Statically vetting android apps for component hijacking vulnerabilities. Proc. of the 2012 ACM Conference on Computer and Communications Security. ACM. 2012. 229–240.

30 Zhang M, Yin H. Appsealer: Automatic generation of vulnerability-specific patches for preventing component hijacking attacks in android applications. Proc. of the 21th Annual Network and Distributed System Security Symposium (NDSS 2014). 2014.

31 Poeplau S, Fratantonio Y, Bianchi A, et al. Execute this! analyzing unsafe and malicious dynamic code loading in android applications. Proc. of the 20th Annual Network & Distributed System Security Symposium (NDSS). 2014.

32 Felt AP, Chin E, Hanna S, et al. Android permissions demystified. Proc. of the 18th ACM Conference on Computer and Communications Security. ACM. 2011. 627–638.

33 Zhang Y, Yang M, Xu B, et al. Vetting undesirable behaviors in android apps with permission use analysis. Proc. of the 2013 ACM SIGSAC Conference on Computer & Communications Security. ACM. 2013. 611–622.

34 Fawaz K, Feng H, Shin K G. Anatomization and protection of mobile apps' location privacy threats. Proc. of the 24th USENIX Conference on Security Symposium. USENIX Association. 2015. 753–768.

35 文伟平,梅瑞,宁戈,等.Android 恶意软件检测技术分析和应用研究.通信学报,2014,35(8):78–85.