计算机研究与发展
Journal of Computer Research and Development
DOI:10.7544/issn1000-1239.2014.20140098
51(7):1385-1396,2014

Android 安全综述

张玉清 王 凯 杨 欢 方喆君 王志强 曹 琛
(中国科学院大学国家计算机网络入侵防范中心 北京 101408)
(zhangyq@ucas.ac.cn)

Survey of Android OS Security

Zhang Yuqing, Wang Kai, Yang Huan, Fang Zhejun, Wang Zhiqiang, and Cao Chen
(*National Computer Network Intrusion Protection Center, University of Chinese Academy of Sciences, Beijing* 101408)

Abstract Android is an operating system applied to smart mobile device which claims a huge market share. The study of its security has attracted wide attention. In this paper, we introduce Android's system architecture and security mechanism, discuss its security performance and the current research situation from two perspectives: system security and application security. Android's system security includes kernel security, architecture security and user authentications mechanism security. The threats on kernel security and architecture security are mainly from vulnerability. The study of kernel security is focused on how to introduce SELinux into the kernel to improve the security performance, and the study of architecture security is focused on how to improve the performance of permission mechanism and how to implement APIs (application programming interface) securely and to guide developers to use APIs normatively. User authentications mechanism is closely related to user's privacy security and can be implemented flexibly, so that the study on its security has received wide attention. Android's application security includes two technologies which are malicious application detection and vulnerability mining. We discuss on malicious application detection from the counterfeit technology of malicious applications and detection technology of malicious application at installation or running process, and discuss on vulnerability mining from component exposed vulnerabilities and security APIs related vulnerabilities. Finally, we summarize current research situation of Android's security study and propose the issues which are worth further study.

Key words Android; security; vulnerability; malicious application; authentication mechanism; survey

摘 要 Android是一款拥有庞大市场份额的智能移动操作系统,其安全性受到了研究者的广泛关注.介绍了Android的系统架构,分析了Android的安全机制,从系统安全和应用安全2个角度对其安全性能和相关研究进行了讨论.Android系统安全包括了内核层安全、架构层安全和用户认证机制安全3个方面.内核层和架构层的安全威胁主要来自于安全漏洞,内核层的安全研究集中于将SELinux引入内核层以增强安全性能,架构层的安全研究集中于权限机制的改进和应用编程接口(application programming interface, API)的安全实现、规范使用.用户认证机制直接关系到整个系统的隐私数据安全,实现方式灵活多样,得到了研究者的广泛关注.Android应用安全的研究包括了恶意应用检测和漏洞挖掘2项技术,对恶意应用的伪造技术、应用安装时恶意应用检测技术和应用运行过程中实时行为监控技术进行了

收稿日期:2014-02-10;**修回日期**:2014-03-26
基金项目:国家自然科学基金项目(61272481);北京市自然科学基金项目(4122089);国家发展和改革委员会2011年信息安全专项项目(发改办高技[2012]1424号)

讨论,对组件暴露漏洞和安全相关 API 的调用漏洞 2 类漏洞的相关研究进行了介绍. 最后,总结了 Android 安全研究现状,讨论了未来的研究方向.

关键词 安卓;安全;漏洞;恶意应用;用户认证;综述

中图法分类号 TP309.1

2007 年 11 月,Google 发布了基于 Linux 内核的开源智能移动操作系统 Android. 该系统拥有庞大的用户数量和应用市场:来自 Gartner 统计数据显示,2013 年第 3 季度全球智能手机的销售量为 2.5 亿多台,其中 Android 系统占据了 81.9%[1];而截止 2014 年 1 月 8 日仅 Android 官方应用市场 Google Play 上的应用数量就达到了 103 万[2].

尽管 Android 设备的用户数量庞大,但用户的安全意识却有待增强. 来自美国《消费者报告》的年度《State of the Net》报告中则指出,在美国近 40% 的手机用户没有采取适当的安全措施,2012 年有 560 万人遭遇过账户未经准许被擅自访问等问题[3]. 同时,虽然 Android 的应用数量巨大,但其应用安全性令人堪忧. TrustGo 公司的分析应用报告显示,Google Paly 上 3.15% 的应用有可能泄露用户隐私或者存在恶意行为,而在国内知名的 91 应用市场上该比例则为 19.7%[4]. 我国用户无法直接从 Google Play 上下载应用,导致了大量的、管理混乱的第 3 方应用市场的存在,对于 Android 设备的安全造成了严重的威胁.

目前,越来越多的研究者开始关注 Android 上的安全问题和解决方案,Android 安全的研究主要从系统安全和应用安全两方面入手.

系统安全可根据系统层次划分为内核层安全和框架层安全. Android 系统以 Linux 为内核,内核漏洞与整个系统的安全息息相关,而且由于处于系统底层,其后果往往较为严重,比如编号为 CVE-2013-6282 的 Linux 内核漏洞可以实现 Android 系统的本地代码提权[5]. 为了减小漏洞发生的可能、减轻漏洞产生的危害,研究者们尝试将更为安全的 SELinux 引入到系统内核中,并取得了较好的成果[6-8]. Android 系统在 Linux 内核之上实现了自己的系统架构层,一旦该层出现安全问题,既有可能导致 Root 权限的泄露,如 CVE-2010-1119 又有可能对基于系统类库的所有应用产生严重的安全威胁,如 OpenSSL 库的随机数生成器初始状态可预测漏洞[9]. 为了增强架构层的安全,研究者们尝试增强现有的安全机制和系统类库的安全性,并取得了较好效果.

针对 Android 应用安全的研究集中在恶意应用的检测与应用漏洞的挖掘技术上. 恶意应用往往冒充正常应用安装到 Android 系统中,并在其中携带恶意代码,从而达到窃取用户隐私、破坏系统等目的,目前的检测方案主要有安装时的安全审核[10-12]和运行过程中的行为检测[13-14]等;应用漏洞的挖掘通常是针对开发过程中无意留下的安全漏洞,这些漏洞被攻击者用以实现拒绝服务攻击、权限提升、窃取应用数据和用户隐私等目的,目前,主要工作集中于利用 Fuzzing 技术或者静态分析技术进行组件暴露漏洞的挖掘[15-19],或是基于静态分析的安全相关 API 调用过程中的漏洞挖掘[9,20-21].

本文从系统安全和应用安全两个角度对 Android 的安全性能和相关研究进行了讨论,其中系统安全包括了内核层安全、架构层安全和用户认证机制安全 3 个方面;应用安全包括了恶意应用检测和漏洞挖掘两项技术. 最后总结了 Android 安全研究现状,讨论了未来的研究方向.

1 Android 系统简介

通常,Android 系统自下向上分为 4 个层次:Linux 内核层、系统库层、应用框架层、应用层,如图 1 所示[22]. 本文将应用框架层和系统库合并为系统架构层,其功能是作为共同内核与应用的中间件,便于按照系统层次对 Android 安全进行分析. 下面就各层功能和关键技术进行详细介绍.

1.1 Linux 内核层

Android 是基于不同版本的 Linux 内核开发出来的,Linux 内核层包括系统层安全机制、内存管理、进程管理、网络堆栈及一系列的驱动模块,位于硬件与其他的软件层之间,提供与硬件的交互.

Binder 是 Linux 内核层的进程通信机制,这种机制为进程间的共谋攻击提供便利,因此在此进行介绍. Android 中每个应用都是一个独立的系统进程,而资源的管理和分配是以进程为单位进行的,通常情况下一个进程不能直接访问另一个进程的资源. 为了实现进程通信,Android 系统引入了 Binder

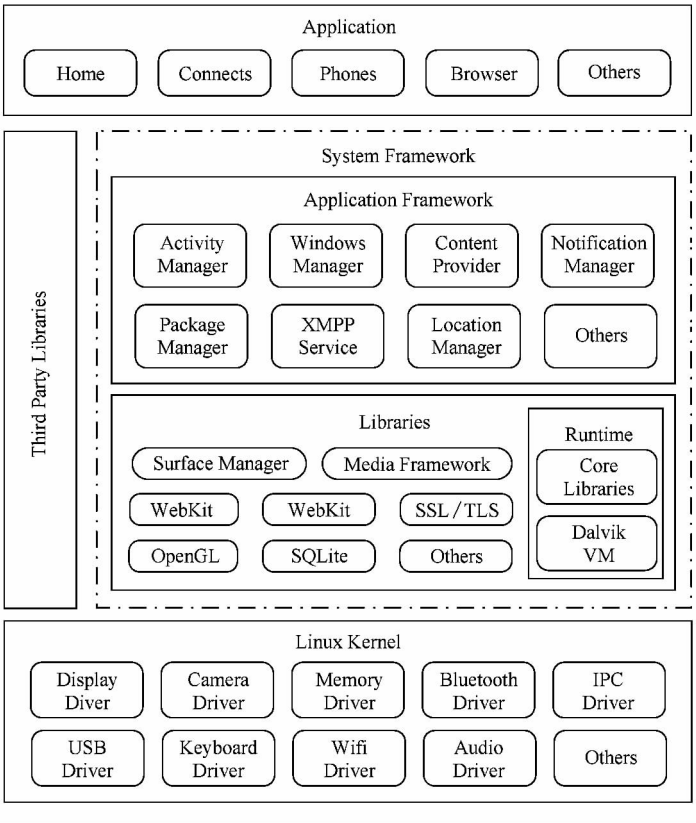


Fig. 1 System architecture of Android.

图 1 安卓系统架构

机制,该机制是 OpenBinder 在 Linux 中的具体实现.这种通信基于 Client/Service 模型,通信的双方都必须创建一个 IBinder 接口,进行通信时,Client 首先通过系统的 Service Manager 获取目标 Service 的代理对象,并通过这个代理对象调用 Service 提供的功能接口,调用请求会通过 Binder 驱动发送给 Service,而 Service 的处理结果也会通过 Binder 驱动发送给 Client.

1.2 系统架构层

系统架构层包含了 Android 系统的系统库、运行环境和应用框架层.系统库是为应用程序运行提供服务的一些 C/C++库;核心库中既包含了 Java 核心类库的大多数功能,也包含了利用 Java 本地调用(Java native interface, JNI)等方式封装的 C/C++库,向应用框架层提供调用底层程序库的接口,还包含了解释运行 Java 字节码的 Dalvik 虚拟机.应用框架层为应用开发者提供了用以访问核心功能的 API 框架.在遵循框架安全性限制的前提下,任何一个应用都可以调用这些核心功能 API 来发布自己的功能组件.应用框架层提供了各种服务和管理工具,包括了应用开发所需的界面管理、数据访问、应用层的消息传递、应用包的管理、电话管理、定位管理以及 Google Talk 服务等功能.

第 3 方类库是指来自第 3 方平台的 API 接口集合,开发者可以在自己的应用中使用这些接口来方便地实现官方 API 未实现的功能.例如,Facebook 提供了一个开源的 Java 库,从而可以让开发者在自己的应用中嵌入 Facebook 的部分社交功能;Admob 是全球最大的移动广告网络,为开发者提供了方便的广告接口.第 3 方类库独立于 Android 系统架构实现,但与系统架构处于相同的地位,即使用内核层提供服务,实现、封装功能模块,供应用层调用.

1.3 应用层

Android 自带的应用有主桌面(Home)、E-mail、SMS/MMS、日历、地图、浏览器、联系人管理等等.这些应用程序通常是用 Java 语言编写,通过调用应用程序框架层所提供的 API 完成.在个人的开发过程中,也可以使用 Java 通过 JNI 的方式,配合 Android NDK 来开发原生程序.它允许 Java 代码和其他语言写的代码(通常为 C 和C++,形成的代码称为本地代码)进行交互.使用 Java 与本地已编译的代码交互,通常会丧失平台可移植性.但是,有些情况下这样做是可以接受的,甚至是必须的.例如,使用一

些旧的库与硬件、操作系统进行交互,或提高程序的性能. JNI 标准保证了本地代码能工作在任何 Java 虚拟机实现下.

2 Android 安全机制

Android 是以 Linux 为内核实现的,保留了 Linux 中的一些安全机制,比如 Linux 中的自主访问控制机制. 而且,Android 应用以 Java 为主要开发语言,确保了类的安全,避免了 C 语言中可能出现的类型转换时没有进行类型检测、数组操作时没有进行边界检测等情况带来的安全隐患.

Android 还根据智能手机上隐私数据较多、运算能力有限等特点,设置了应用签名、权限审核和沙盒运行 3 项重要的安全机制. 这 3 项安全机制的设立实现了鉴别开发者、限制应用行为和保障应用独立运行的功能,分别保障了应用发布、安装和运行过程中的安全[23].

2.1 签名机制

所有 Android 应用都要被打包成. apk 文件,这个文件在发布时都必须被签名,与通常在信息安全领域中使用数字证书的用途不同,Android 利用数字签名来标识应用的作者和在应用间建立信任关系,而不是用来判定应用是否应该被安装. 而且,这个数字证书并不需要权威的数字证书签名机构认证,而是由开发者来进行控制和使用的,用来进行应用包的自我认证的.

对 Android 应用签名可以采用调试模式和发布模式:使用 Android 开发工具(命令行和 Eclipse 等)开发的应用是用一个调试私有密钥自动签名的,这些应用被称为调试模式应用,这些应用只能用于开发者自行测试,不能够发布到 Google Play 官方应用市场上. 当一个开发者需要将自己的应用程序发布到 Google Play 上时,必须生成一个发布模式的版本,即用其私有密钥签署应用.

应用签名在发布时进行,在应用安装时被验证,用以实现对应用来源的鉴定.

2.2 权限机制

为了向用户通知应用使用各项关键功能的情况,Android 定义实现了权限机制. 一个权限主要包含 3 个方面的信息:权限的名称、属于的权限组、保护级别. 一个权限组是指把权限按照功能分成的不同的集合. 每个权限通过 Protection Level 来标识保护级别:normal, dangerous, signature, signature or system. 不同的保护级别代表了程序要使用此权限时的认证方式. normal 的权限只要申请了就可以使用;dangerous 的权限在安装时需要用户确认才可以使用;signature 和 signature or system 的权限需要使用者的应用程序和系统使用同一个数字证书.

Android 系统中定义了一个系统权限集,实现了针对诸如照相、录音、通话等关键功能的权限控制. 系统上的应用也可以自定义访问应用中某个功能组件的权限,从而在对外提供服务的同时确保组件安全. 应用的开发过程中,开发者必须在 AndroidManifest. xml 文件中使用〈uses-permission〉标签来声明所要使用的权限. 这些申请的权限会在应用安装之前通知用户,只有在用户授权其使用这些权限的情况下才能允许其安装并在运行过程中使用相关功能.

权限机制被用来在应用安装时监督、限制应用行为.

2.3 沙盒机制

为了保障每个应用运行过程中不被其他应用影响,Android 系统为每个应用在运行过程中提供了一个沙盒. 其具体实现是,Android 系统为每个应用提供一个 Dalvik 虚拟机实例,使其独立地运行于一个进程,并且为每一个应用创建一个对应于 Linux 底层的用户名,并设置用户 ID. 利用这个用户 ID 可以保护应用的文件、数据和内存. 如果希望 2 个应用共享权限、数据,可以通过设置 sharedUserID 来声明 2 个应用使用同一个用户 ID,运行于同一进程,并共享其资源和权限. 但是声明使用 sharedUserID 的应用必须有相同的用户签名.

这种机制不但可以在应用运行过程中保障其独立性,还可以提高系统的鲁棒性. 当单个应用运行出现问题时,可以消除其虚拟机实例来保障整个系统的安全运行.

3 Android 系统安全

本文将 Android 系统分为 3 层,最底层为 Linux 内核层,进程往往以较高权限运行,其漏洞可能导致高级权限泄露;中间为系统架构层,该层位于内核与应用之间,其安全漏洞既可能导致高级权限的泄露也可能给使用框架层接口的所有应用的安全带来威胁;最上层为应用层,其安全性将在第 4 节中进行详细介绍. 内核层和系统架构层的安全威胁往往来自于安全漏洞,现已有工作将成熟的 Linux 漏洞挖掘

技术应用于其内核层,而关于其系统架构层的安全研究也得到了部分研究者的关注,更多的研究则着重于如何增强这 2 层的安全性能来降低系统漏洞带来的威胁,抵御恶意应用的攻击行为.

此外,本节还将讨论用户认证机制的实现,用户认证机制直接关系到整个系统的隐私数据安全,其实现过程往往涉及到内核层的数据采集、框架层的策略定制与分析以及应用层的用户交互,而且该机制的实现方式多样化.

3.1 内核层安全

Linux 内核层的安全威胁来自于 Linux 内核所携带的漏洞. 这些安全漏洞往往威胁着整个系统的安全,恶意应用可以利用这些漏洞来提升自己权限,甚至是获取 Root 权限,从而肆意窃取数据、破坏系统. 例如 CVE-2012-0056 漏洞,影响了 Linux 版本大于等于 2.6.39 版本的系统,其产生原因是 Linux 不能正确完成/proc/〈pid〉/mem 的权限控制,导致攻击者通过改写一个特权进程的内存,可以获取 root 权限来执行任意的 root 权限的代码,该漏洞被用于 Android 4.0 版本的本地提权;CVE-2013-6282 漏洞,ARMv6/v7 架构的 Linux 内核中的 get_user/put_user 接口是没有验证目标地址,由于硬件架构的更迭,get_user/put_user 最初用于实现和控制域切换的功能被弃用了,导致任何使用该 API 的内核代码都可能存在安全隐患,让任意应用来读写内核内存,造成权限泄露.

面对层出不穷的内核漏洞,一方面,实现行之有效的漏洞挖掘工具是必要的. 早在 2010 年,美国软件分析公司 Coverity 就曾宣布 Android 内核中存在大量漏洞,其中包括 88 个高风险漏洞,有可能会导致用户敏感信息泄露;刘剑等人[24]则开发了一套基于控制流挖掘的 Android 系统代码漏洞分析工具,针对 Android 内核代码的多种典型错误构建相关的分析脚本进行了分析,从而检测发现了代码库中一系列脆弱点.

另一方面,如何增强 Linux 内核的安全性,避免这些漏洞带来的危害同样值得关注[25]. 2010 年,Shabtai 等人[26]提出了以 SELinux 为内核实现 Android 安全机制增强的思路;2011 年,Bugiel 等人[7]以 TOMOYO Linux 为安全增强内核,开发了 TrustDroid 系统;2013 年 Smalley 等人[8]实现了基于 SELinux 内核的 Android 系统. 不论是将 TOMOYO Linux 还是 SELinux 作为 Android 系统内核,都是从 Linux 层面上改变了文件的访问控制方式,即由 DAC(自主访问控制)转变为 MAC(强制访问控制). 在 DAC 中,对某个客体具有所有权(或控制权)的主体能够将对该客体的一种访问权或多种访问权自主地授予其他主体,并可在随后的任何时刻将这些权限回收. 而 MAC 中,则由系统对用户所创建的对象进行统一的强制性控制,按照规定的规则决定哪些用户可以对哪些对象进行什么类型的访问,即使是创建者用户,在创建一个对象后,也可能无权访问该对象. 通过用 MAC 替换 DAC 可以有效地减弱内核层出现漏洞时产生的安全威胁.

3.2 架构层安全

Android 系统架构位于内核层与应用层之间,是 Android 系统的主体. 系统架构层的安全威胁同样往往来自于安全漏洞.

例如,Android Zygote 漏洞是 Android 2.3 版本之前存在的漏洞,Dalvik 虚拟机对应用程序的执行过程审核不严格,造成应用程序可以不断地创建进程,当耗尽所有的 PID 时,PID 循环使用 0,导致 Root 权限的泄露;CVE-2010-1119 漏洞是一个 WebKit 库的 Use-after-free 类型的漏洞,受影响的系统包括 Android 2.0 至 Android2.1.1 操作系统的浏览器,该漏洞可在受影响的系统中执行任意代码或者造成拒绝服务攻击,可以读取手机中的 SMS 短信数据或者其他敏感数据.

通过这 2 个漏洞可以看出,Android 系统架构层的漏洞既有可能造成系统底层 Root 权限泄露,也有可能对上层应用产生巨大的安全威胁.

针对以上所述安全威胁,相关研究集中在了以下 3 个方面:

1) SELinux 引入系统内核,为系统架构层提供更加安全的服务,实现更加安全的策略,例如 Bugiel 等人[27]在 SELinux 为基础的 Android 4.0 系统架构层指定了一套协议语言,提供了定制安全策略的服务,从而使 SELinux 的安全特性在 Android 系统的架构层得以应用.

2) 完善 Android 现有的安全机制、细化安全策略也是极为重要的手段[28],Android 系统的 3 大安全机制——签名机制、权限机制和沙盒机制中,签名机制和沙盒机制设计较为良好,只是偶尔爆出安全漏洞,较少得到安全研究者的关注;但权限机制因其“全是或全否”的粗粒度管理方式备受人们的诟病,而且应用安装后的行为得不到任何监控,对于应用运行过程的行为监控基本为零.

3) Android 系统框架 API 接口的实现如果存

在安全问题,将严重威胁到调用该 API 的应用安全,而这些 API 中,SSL/TLS 等加密 API 显得尤为重要,研究者们在这方面的研究也取得了良好的成果.

上述 3 点中提到的第 1 点中的 MAC 机制已于 3.1 节进行了详细阐述,在此不做赘述,下面针对 Android 权限机制和 API 接口的安全实现进行讨论研究.

3.2.1 权限机制的改进

之所以称 Android 系统是粗粒度的,主要是由于其现有的"全是或全否"的授权规则,和对应用的运行过程的"零"监控. 现有的研究大致可以分为 3 个方面:对现有权限机制的分析,主要针对现有权限集合理性的讨论以及"Permission-API"对应关系映射集的建立;安装时的选择性授权的研究则修正了"全是或全否"的授权规则,实现了细化;而进程间通信机制则导致了共谋攻击的可能,研究如何防范这种攻击受到了人们的关注[29].

1) 现有权限机制分析

很多研究者针对现有的权限机制进行了细致的分析研究.

根据一项对于 Android 生态系统中权限机制的演变过程研究[30],大量的应用仍然存在着过量申请权限的问题,这对于利用权限申请判别应用是否为恶意应用造成了巨大障碍;许多设备预装应用更是拥有大量的高安全级别权限,如果这些应用存在组件暴露漏洞等,可能造成高级权限的泄露;虽然 Android 应用权限集的数量在日益增多,但是其目标主要是为新的硬件设备提供安全保证而不是对原有权限集的改进.

另有研究表明[31],Android 系统中有超过 75 个权限,但是如果应用开发者只调用开发文档中声明的 API 接口的话,那么大约 22%的非系统级别的权限是不需要设置的,根据这项研究,可以给出一个精简的权限集合.

此外,应用的权限与系统架构层中提供的 API 接口是绑定的,即如果应用想要调用一个安全相关的 API 接口,则需要拥有相应的权限. 但是这种"Permission-API"的对应关系 Google 并没有直接给出. 来自 Felt 等人[32]的工作完成了这一对应关系的映射集,以此为基础通过比对应用的 API 调用和所申请权限,检验应用是否存在申请权限过多的现象.

2) 安装时的选择性授权

为了改善应用安装时的授权过程中"全是或全否"的授权规则,实现可选择的授权机制是有意义的. 名为 Apex[33]的工具对此进行了改进,在应用安装的过程中让用户选择可以授予的权限,并给出更加详尽的配置方案,比如可使用该权限多少次等. 但这种方案的不足之处在于如果应用没有得到所需的授权却调用了相应的 API 接口,那么很有可能造成应用的异常,甚至会因为 Java 层抛出的异常没有得到正确的处理,导致整个进程的崩溃.

为了保障应用的正常运作,如果不想授予其某个权限,将伪造的 API 返回值返回给目标应用优于直接拒绝其拥有该项权限[34].

3) 进程间通信给权限机制带来的挑战

Android 系统的权限机制以进程为监控单元,但是应用的进程间通信(inter-process communication, IPC)给这种安全机制带来了不小的挑战.

共谋攻击是指 2 个或多个应用分别获取一部分关键功能但各自却不会对系统安全造成威胁,但是这几个应用的功能共同作用则产生后果严重的攻击,并不易被安全监控系统发觉. IPC Inspection[35],通过修改系统,实现了当 2 个应用发生通信时权限较高的应用会被削减权限的安全策略,从而使其丧失进行共谋攻击的能力.

远程过程调用(remote process call, RPC)是 Android 进程间通信的重要组成部分,如何保障合法的被调用者的数据不被非法的调用者窃取,也是一个关键的问题. Dietz 等人[36]于 2011 年给出了一种名为 Quire 的解决方案:首先设计了 RPC 过程中的签名方案,从而在整个调用链中的每个调用节点上有效的标识调用者的识别信息;然后利用这种识别方案,强制被调用者做出选择:在维持远程过程调用的同时降低权限进行操作,或者中断远程过程调用,独立完成操作. 通过这种方法可以有效地防止被调用者关键功能权限的泄露.

3.2.2 API 安全实现和规范使用

API 是指应用编程接口,是软件系统不同组成部分衔接的约定. 操作系统提供的 API 为开发者提供了方便的开发接口,而又无需访问源码,或理解内部工作机制的细节,便可用其实现具体功能. API 接口的安全性直接关系到 Android 上层应用的安全. 现有的 API 实现漏洞往往集中于 Webkit 这样的通用组件上,但是在研究领域,对于 SSL/TLS 这样的安全相关 API 的安全性则更多地得到了研究者的关注,这包括了 OpenSSL 中的漏洞,以及安全 API 的不规范使用.

虽然 Android 架构中提供的 API 接口都有良好的安全性,但是研究者依然发现了安全相关的 SSL/TLS 库 OpenSSL 中存在的一个漏洞[9]. OpenSSL 的安全性极大地取决于伪随机数生成器(pseudo random number generator, PRNG)的不可预测性. Android 系统启动之初 Dalvik 虚拟机就为 OpenSSL 的 PRNG 进行了初始化,所有应用在生成随机数时的初始状态都是相同的,这导致有一定概率恢复出 PRNG 的初始状态. 这个漏洞可能对所有使用 OpenSSL 的应用产生严重的安全威胁,由此可见,Android 系统架构中 API 接口的重要性.

尽管 Android API 安全漏洞影响广泛,但是其漏洞并不常见. 相比而言,应用在调用 API(特别是安全相关的 API)过程中的不规范操作,往往会带来更多的安全后果. Fahl 等人[20]在研究了 13 500 个热门的免费应用后,发现其中有 1 074 个应用可能存在 SSL/TLS 没有被正常使用,并且可能遭受中间人攻击,在对 100 个应用进行了手动测试后,发现了 41 个应用被成功实现了中间人攻击. Egele 等人[21]对 Google Play 市场上的 11 748 个应用进行了测试,发现有 10 327 个应用使用了加密 API 并在使用过程中犯下了至少一个错误.

通过这些研究可以看出,在应用使用安全相关 API 完成功能时的错误操作往往更会对应用运行过程和用户隐私数据的安全造成威胁. 这说明 Android 应该提供给开发者更加详尽的开发文档来指导其使用安全 API;同时,也从另一个方面说明了 Android 架构层提供的安全相关的 API 接口需要改进以帮助开发者规避这些安全问题.

3.3 用户认证机制

Android 系统安全既包括了内核层、架构层中不为用户所知的安全机制,也包括了用户使用过程中经常使用的用户认证机制. 所谓用户认证机制,就是判定当前设备的使用者是否为合法用户. 该机制通常作用于屏幕解锁的过程,利用 PIN 密码、解锁图案、人脸特征识别、指纹特征识别等方式进行认证. 这种机制可以有效地保护智能设备不被非法用户使用或查看设备中的信息,直接关系到整个系统的隐私数据安全,其实现过程往往涉及到内核层的数据采集、框架层的策略定制与分析以及应用层的用户交互,而且该机制的实现方式多种多样. 现有的用户认证机制无法在保障安全和低用户参与度中达到良好的平衡,因此值得更多的关注研究.

1) 传统认证方式的暴力破解

但是这些用户认证机制往往并不是完全可靠的. 针对 PIN 密码的暴力破解技术已较为成熟,而 3×3 解锁图案的安全性能也存在着一定的问题. Uellenbeck 等人[37]的团队研究了利用解锁图案作为用户认证方法的安全性能,在统计了大量用户的解锁图案,构造出了一个基于 Markov 链的模型,来量化分析 Android 解锁图案的统计模型,结果发现用户在解锁图案的选择过程中有明显的偏好,并非完全的随机化. 从而可以指导暴力破解解锁图案的过程.

2) 生物特征的优劣评判

除了 PIN 密码和解锁图案等常规的用户认证方法外,很多基于生物技术的认证方式也日渐流行. 2013 年 9 月 20 日上市的 iPhone 5S 便以指纹识别作为用户认证的方式. 为了评判各种解锁方案的优劣,Trewin 等人从响应时间、运算强度、误差和任务可中断性等方面,对基于声音、面部、指纹等生物特征的用户认证机制和原始的 PIN 密码用户认证机制进行了对比,结果表明语音 PIN 密码识别速度最快,面部识别不需用户记忆密码,将 2 种生物验证方式进行叠加可以获取更好的用户体验,但是会耗费更多的系统内存.

3) 低参与度的用户认证技术

此外,每次设备唤醒都需要用户认证的机制严重影响了用户的使用体验,导致 30%的用户选择放弃使用用户认证,以保证使用的流畅性. Riva 等人[38]开发了一套判断何时需要用户授权的智能决策系统. 该系统结合了多种信号,如生物识别、连续性、真实性等来判定是否需要用户的授权,从而在保证安全的同时,减少 42%的认证需求. Li 等人[39]研究了一种通过长期监控使用者的操作来获取用户的操作习惯,判定手机是否正在被非合法用户使用,从而在不打扰用户的情况下进行数据收集和身份认证,但 Serwadda 等人[40]设计了名为“Lego”的机器人来逐渐改变用户操作习惯数据模型,从而最终破解了这种认证机制获取了相应权限. 由此可见,安全的、低用户参与度的认证机制仍然有待于研究者们的深入探讨.

4 Android 应用安全

Android 系统上的应用安全包括了 2 方面的内容:一方面针对应用是否存在恶意行为进行检测,保证系统及其他应用不受这种恶意行为的影响;另一方面研究这些应用中可能存在的安全漏洞,及时发

现可能产生的功能、数据泄漏的情况,确保应用的独立、正确、安全运行.

4.1 Android 恶意应用检测技术

恶意应用是指携带有恶意代码,对系统和其他应用的正常运行和关键数据等产生安全威胁的应用.恶意应用通常在热门应用中插入恶意攻击代码从而吸引用户安装,并利用安全管理混乱的第 3 方应用市场进行发布和传播.为了拥有需要的功能权限,往往利用部分用户安装应用时不仔细审核权限列表或者本地提权漏洞等方式实现.而且,随着设备性能的增强和研究的深入,恶意应用的隐私窃取能力也日益提高.Templeman 等人[41]开发的视觉工具 PlaceRaider,可以利用智能手机多种传感器进行用户所处室内空间的 3 维空间模型构建,并窃取房间中的隐私信息.

恶意应用的伪造方式除了原有的反编译后插入恶意代码外,还可能利用 MasterKey 漏洞实现不改变应用签名的可执行文件嵌入,危害巨大;对于恶意应用的分析,可以在应用安装时进行静态审核,也可以在定制的系统中实现行为的实时监控.

4.1.1 恶意应用的伪造方式

通常恶意应用的开发者插入恶意攻击代码到目标应用中的方式是将正常应用进行反编译,添加上恶意代码后再重新打包,这种方式生成的恶意应用代码与应用的原生签名不同.但是,自 2013 年 7 月先后爆出的 Android 系统中签名认证绕过漏洞[42],让恶意攻击者可以将新的字节码文件 classes.dex 置于 APK 文件中,从而绕过签名机制.这一漏洞可能导致系统中原有正常应用被非法替换,或者伪造的应用被安装到系统中.虽然 Google 及时给出了修补措施,但是 Android 系统的安全补丁并不会及时推送给设备,而是在新的版本中实现修改.由于 Android 设备生态系统的碎片化严重,很难大范围地推行新版本的更新,所以,这个漏洞在今后一段时间里将会成为恶意应用伪造正常应用的极为重要的手段之一.

4.1.2 应用安装时的恶意应用分析

王菲飞等人[43]开发了一种基于特征码的恶意代码检测方法.国外著名的 Android 恶意代码检测工具 Androguard[44]则是基于开发者签名匹配来实现的.通过建立庞大、有效的数据库,这 2 种方法可以快速有效地发现已知的恶意应用,但是,对于新的应有是否为恶意行为的判定却无法实现.

为了有效地判定未知应用是否为恶意的,需要新的方法的出现.在应用的 APK 文件中,包含了其字节码文件 classes.dex 和声明权限申请的 AndroidManifest.xml 文件.通过对 classes.dex 文件进行逆向处理可得其 smali 或 Java 格式的源码,针对这些源码进行静态分析得到其 API 调用情况;通过对 AndroidManifest.xml 文件的逆向可以获知这个应用申请的权限信息.恶意应用在 API 的使用和权限申请方面往往表现出不同于同类型正常应用的特性,比如一个相机应用却申请了发送短信息的功能.在应用安装时便可对其 API 调用、权限申请或 2 者一起进行限制或者统计分析.来自 Enck 等人[11]的轻量级应用检验工具 Kirin,为 Android 上的应用安装提供了一个可定制策略的应用验证方案,当安装新的应用时,首先交由 Kirin 使用安全需求工程技术来判定一个应用是否有可能是恶意应用.这是一种基于安全策略的防范方法.

此外,根据所申请权限和调用 API 情况的统计数据,利用数据挖掘技术判定目标应用是否含有恶意行为也是一种较为有效的方法.杨欢等人[12]的工作对提取 Android 应用申请的权限信息,构建权限特征集合,使用权限频繁模式挖掘算法构建恶意应用家族的权限关系特征库,然后对未知应用进行匹配来检测未知恶意应用.Peng 等人[10]利用类似的思路提出了一种是基于概率生成模式的 Android 应用风险评估方法,并取得了良好的效果.这种基于数据挖掘的方法可以有效地检测未知应用是否存在恶意行为,但是由于应用普遍存在过分申请权限的现象,单纯针对权限申请的数据挖掘存在较大的误报率.

综上所述可以看出,在应用安装时的恶意应用分析技术除了传统的特征码和签名信息检测外,通过静态分析、数据挖掘等技术的应用行为审计和检测同样起到了较好的效果,这些工作不但可以在本地设备上进行,还可以在云平台上进行[45].

4.1.3 定制系统中的实时监控

4.1.2 节中叙述的方法只对于应用安装时的威胁分析,一旦应用成功通过了检测机制,其运行过程将不受安全机制的监督.由于 Android 系统中设计良好的沙盒机制,第 3 方应用很难监测到目标应用的具体行为.但是得益于 Android 系统的开源性,研究者转而定制自己的安全测试系统,并将其用于恶意应用的实时监测和行为控制中[46-47].名为 VetDroid 的动态分析系统[13],可以监测权限的相关功能何时被使用以及具体的使用情况,从而提供一套细粒度

的应用行为监测数据. Kwong 等人[14]的工作则更加全面细致,他们基于 Android 源码实现了名为 DroidScope 的恶意应用动态监控系统,该系统提供了硬件层、系统层和 Dalvik 虚拟机层 3 个不同层面的监控 API,方便研究者自定义分析策略. 利用 DroidScope 提供的接口,可以收集发生在本地代码层和 Java 层的应用行为,实现多种丰富的安全策略.

除了系统中自动实现对应用行为监测之外,通过引入用户行为和应用的具体操作之间的关联关系进行行为监控也可以有效地发现恶意应用的攻击行为. Yang 等人[13]认为没用用户参与下的关键信息的传输是一种十分可疑的恶意行为,他们开发的 APPIntent 框架可以判定系统后台正在发生的关键数据的传输是否起源于用户的操作. 此外,Livshits 等人[34]在使用 Windows Phone 8 操作系统的 Lumia 中实现了应用获取安全数据时及时通告设备用户的机制,借鉴这一工作,在 Android 中实现操作关键数据时的实时通告可以通过以下 2 种方式进行实现:1)静态分析 Android 应用,并在关键数据的操作之前插入提示;2)监控 Android 框架中的 API 接口,针对目标应用的 API 调用行为进行实时的监控.

4.2 Android 应用漏洞挖掘技术

Android 应用中的安全漏洞,不仅威胁着应用运行过程中的稳定性、私有数据的安全性,也可能因为暴露关键功能而威胁到整个系统的安全. WebView 接口漏洞是 2013 年 9 月在诸多流行互联网应用集体爆发的安全漏洞. 这一漏洞产生的原因是,互联网应用使用了 Android 提供的 WebView 控件来实现应用内的嵌入式网页显示,WebView 中提供了一个名为 addJavascriptInterface 的接口函数来实现本地 Java 和 Javascript 的交互,这使得攻击者可以通过 Javascript 将攻击指令注入到本地 Java 代码中执行,远程进行命令执行. 虽然这一漏洞在国内今年 9 月份才得到人们的广泛研究,但是 Luo 等人[48]早在 2011 年的 ACSAC 会议上便提出了该攻击方式.

面对数量庞大的 Android 应用,如何实现自动而有效的漏洞挖掘工具是国内外研究者研究的重点. 通常的漏洞挖掘技术可以分为动态挖掘和静态挖掘 2 类,目前针对 Android 应用的较为成熟的漏洞挖掘技术,集中于利用 Fuzzing 测试、静态分析发现组件暴露漏洞和利用静态分析发现安全相关 API 调用的漏洞.

4.2.1 组件暴露漏洞

Android 应用分为 Activity,Service,Broadcast Receiver 和 Content Provider 四种组件[15],并且可以设置是否对外可见. 如果可见,那就意味着这些组件可以接收来自第 3 方的应用数据. 如果这些数据没有得到足够的安全审核,那么极有可能将组件所实现的功能暴露在外,被攻击者恶意调用;或者实现恶意数据注入,影响应用的正常运行.

针对这一安全问题,研究者们展开了广泛而深入的讨论. Zhou 等人[16]的研究只针对于 Content Provider 组件,由于缺少必要的访问控制导致应用可能被动地暴露一些应用内的私有数据,或者无意中操作某些敏感的应用内安全设置. Miller 等人[17]的 JarJarBinks 工具实现了针对应用组件的 Fuzzing 测试,并得到了一系列的应用运行异常数据;ComDroid[49]则利用逆向工程和静态分析查看应用可能存在的组件暴露漏洞. DroidChecker[18]和 CHEX[19]也利用静态分析的方法,研究应用组件间通信过程中的数据流和控制流,来挖掘可能造成组件劫持、功能暴露的安全漏洞.

关于组件暴露漏洞的研究已经得到了研究者们的广泛关注,动静态测试技术都得到了实现,并取得了较好的成果.

4.2.2 安全相关 API 调用漏洞

本文 3.2.2 节曾讨论过 OpenSSL 在 Android 中的实现过程中由于伪随机数生成器的初始状态相同,可能造成可以推算出伪随机数树生成器状态的严重后果[9]. 这种底层实现上的漏洞导致如果应用使用了 org.webkit 包,并且密钥交换方案是 RSA,那么就可能造成 SSL 对话 PMS 的泄露. 根据这一特点,可以在大范围内实现该漏洞的静态挖掘.

Fahl 等人[20]的工作则针对应用使用 SSL/TLS 进行加密数据传输的实现过程进行了分析. 通过静态分析,作者发现了大量应用可能在 SSL/TLS 的使用过程中存在漏洞,容易受到中间人攻击,但是在后期的针对 100 个可能存在漏洞的应用的手动测试中,只有 41 个应用被成功实施了中间人攻击. 如不考虑其实现中间人攻击的手法上不成熟的情况,这说明其自动的漏洞挖掘过程有较大的误报率. Egele 等人[21]的工作则针对 Android 应用使用加密算法时可能出现的问题进行了静态分析,并发现有 88% 之多的应用在使用加密 APIs 的时候出现了至少一处错误.

安全相关 API 调用漏洞的研究自 2012 年起得到了研究者的广泛关注,研究方法多以静态分析为主,并且发现了大量安全威胁,效果显著.

5 思考与讨论

Android 系统安全方面的研究已得到了广泛关注,不论是系统层面还是应用层面的研究都有着较好的成果,但这并不代表 Android 系统的安全研究已经完备. 随着 Android 系统的发展,越来越多的新性能被引入,其中的安全研究问题也将层出不穷.

从系统层面考虑,Android 4.4 版本中已经将 SELinux 加入到官方发布的系统之中,这表示着更加安全的 Android 系统的来临,而 Android 系统架构层的安全研究也将因此拥有更加广阔的发展空间,此外 API 接口的设计与规范使用、低用户参与度的用户认证机制,都有着较大的研究价值.

1) Android 已经将 SELinux 引入到官方发布的系统版本中,如何利用 MAC 带来的安全优势增强现有的 Android 安全机制,特别是权限机制,实现恶意应用的及时发现与控制是值得进行研究探讨的;

2) Android 安全相关的 API 接口需要设计更易操作的调用方式,或者提供更加详尽的使用规范,这方面的安全问题刚被研究者披露出来,值得更加深入地研究讨论;

3) 设法实现一种低用户参与度的用户认证机制,可以在便利性与安全性之间取得平衡,现有的工作或是过于复杂、或是安全性不够,尚无令人满意的方案出现,而且用户认证的实现方式多样,值得拓宽思路,深入研究.

从应用层面考虑,为了保障 Android 应用的安全,恶意应用检测技术已经得到了长足的发展,但是随着系统的发展仍有更多的研究空间;而漏洞挖掘技术在 Android 系统中的切入点较少,符号执行等先进的漏洞挖掘技术的应用研究并不充分,值得研究者更加广泛地关注.

1) 现有的恶意应用检测技术已较为完善,但如何实现一套有效的恶意应用检测方案,在保障检测的及时性、准确性的基础上,提高部署的灵活度和方便性,从而使其拥有广泛推广、应用的能力,值得更加深入地研究和探讨;

2) 国内存在大量参差不齐的第 3 方应用市场,为恶意应用提供了滋生传播的渠道,如何在这些市场上实现应用的安全检测,如何对其进行有效的监控和安全评级等问题,值得研究者和业界重视;

3) 从漏洞挖掘的切入点来看,目前 Android 上漏洞挖掘的相关研究有较强的局限性,主要集中于安全相关 API 的错误使用和组件暴露漏洞 2 个方面,找到新的挖掘漏洞的切入点可以为 Android 安全的研究找到新的出路.

6 总 结

本文介绍了 Android 的系统架构,总结了其安全机制,从系统和应用 2 个方面阐述了 Android 上现有的安全问题和研究方案. Android 系统安全又可根据其系统层次分为内核安全、架构安全和用户认证机制安全 3 个方面:内核安全往往与相应的 Linux 内核中的安全问题相关,通过将 SELinux 的 MAC 机制引入可以有效的加强其安全性;系统架构是沟通内核与应用的中间层,是 Android 系统的主体,通过优化权限机制、加强 API 调用监控、及时发现并修复其安全漏洞等可以有效地提高其安全防护能力;用户认证机制则是保护设备不被非法用户所用的重要方法. Android 应用安全包含了恶意应用检测和漏洞挖掘技术 2 方面内容,分别从检测应用攻击行为和发现应用安全脆弱点 2 个角度实现了安全性能的增强.

参 考 文 献

[1] Gartner. Worldwide smartphone sales in Q3 2013 [EB/OL]. [2014-01-08]. http://www.gartner.com/newsroom/id/2623415

[2] AppBrian Stats. Number of available Android applications [EB/OL]. [2014-01-08]. http://www.appbrain.com/stats/

[3] Consumer Reports. Keep your phone safe-How to protect yourself from wireless threat [EB/OL]. [2014-01-08]. http://www.consumerreports.org/cro/net0613.htm#info

[4] TrustGo. BSides Las Vegas: Your droid has no clothes [EB/OL]. [2014-01-08]. http://blog.trustlook.com/

[5] National Vulnerability Database. Vulnerability summary for CVE-2012-0056 [EB/OL]. [2013-12-09]. http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2012-0056&cid=2

[6] Nakamura Y, Sameshima Y. SELinux for consumer electronics devices [C] //Proc of Linux Symp. Ottawa: Linux Symp Inc, 2008: 125-133

[7] Bugiel S, Davi L, Dmitrienko A, et al. Practical and lightweight domain isolation on android [C] //Proc of the 1st ACM Workshop on Security and Privacy in Smartphones and Mobile Devices. New York: ACM, 2011: 51-62

[8] Smalley S, Craig R. Security enhanced (SE) Android: bringing flexible MAC to Android [C/OL] //Proc of the 20th Annual Network and Distributed System Security Symp. 2013 [2014-03-20]. http://www.internetsociety.org/events/ndss-symposium-2013/papers-and-presentations

[9] Kim S H, Han D, Lee D H. Predictability of Android OpenSSL's pseudo random number generator [C] //Proc of the 2013 ACM SIGSAC Conf on Computer & Communications Security. New York: ACM, 2013: 659-668

[10] Peng H, Gates C, Sarma B, et al. Using probabilistic generative models for ranking risks of android apps [C] //Proc of the 2012 ACM Conf on Computer and Communications Security. New York: ACM, 2012: 241-252

[11] Enck W, Ongtang M, McDaniel P. On lightweight mobile phone application certification [C] //Proc of the 16th ACM Conf on Computer and Communications Security. New York: ACM, 2009: 235-245

[12] Yang Huan, Zhang Yuqing, Hu Yupu, et al. Android malware detection method based on permission sequential pattern mining algorithm [J]. Journal on Communications, 2013, 34(Z1): 106-115 (in Chinese)
(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[J]. 通信学报，2013，34(Z1)：106-115)

[13] Yang Z, Yang M, Zhang Y, et al. Appintent: Analyzing sensitive data transmission in android for privacy leakage detection [C] //Proc of the 2013 ACM SIGSAC Conf on Computer & Communications Security. New York: ACM, 2013: 1043-1054

[14] Yan L K, Yin H. Droidscope: Seamlessly reconstructing the OS and dalvik semantic views for dynamic android malware analysis [C] //Proc of the 21st Usenix Security Symp. Berkeley: USENIX Association, 2012: 29-29

[15] Android Developer. Application Components [EB/OL]. [2013-12-03]. http://developer.android.com/guide/components/fundamentals.html#Components

[16] Zhou Y, Jiang X. Detecting passive content leaks and pollution in android applications [C/OL] //Proc of the 20th Annual Network and Distributed System Security Symp. 2013 [2014-03-20]. http://www.internetsociety.org/events/ndss-symposium-2013/papers-and-presentations

[17] Miller B P, Fredriksen L, So B. An empirical study of the reliability of UNIX utilities [J]. Communications of the ACM, 1990, 33(12): 32-44

[18] Chan P P F, Hui L C K, Yiu S M. Droidchecker: Analyzing Android applications for capability leak [C] //Proc of the 5th ACM Conf on Security and Privacy in Wireless and Mobile Networks. New York: ACM, 2012: 125-136

[19] Lu L, Li Z, Wu Z, et al. Chex: Statically vetting android apps for component hijacking vulnerabilities [C] //Proc of the 2012 ACM Conf on Computer and communications security. New York: ACM, 2012: 229-240

[20] Fahl S, Harbach M, Muders T, et al. Why eve and mallory love Android: An analysis of Android SSL (in) security [C] //Proc of the 2012 ACM Conf on Computer and Communications Security. New York: ACM, 2012: 50-61

[21] Egele M, Brumley D, Fratantonio Y, et al. An empirical study of cryptographic misuse in android applications [C] //Proc of the 2013 ACM SIGSAC Conf on Computer & Communications Security. New York: ACM, 2013: 73-84

[22] Android APP Market. Android architecture—The key concepts of Android OS [EB/OL]. [2013-12-03]. http://www.android-app-market.com/android-architecture.html#comments

[23] Google Inc. Android security [EB/OL]. [2013-06-26]. https://source.android.com/tech/security/

[24] Liu Jian, Sun Keqin, Wang Sunlü. Vulnerability analysis of the Android operating system code based on control flow mining [J]. Journal of Tsinghua University: Science and Technology, 2012, 52(10): 1335-1339 (in Chinese)
(刘剑，孙可钦，汪孙律. 基于控制流挖掘的 Android 系统代码漏洞分析[J]. 清华大学学报：自然科学版，2012，52(10)：1335-1339)

[25] Coker R. Porting nsa security enhanced linux to hand-held devices [C] //Proc of Linux Symp. Ottawa: Linux Symp Inc. 2003: 117-127

[26] Shabtai A, Fledel Y, Elovici Y. Securing Android-powered mobile devices using SELinux [J]. Security & Privacy, 2010, 8(3): 36-44

[27] Bugiel S, Heuser S, Sadeghi A R. Flexible and fine-grained mandatory access control on Android for diverse security and privacy policies [C] //Proc of the 22nd Usenix Security Symp. Berkeley: USENIX Association, 2013: 131-146

[28] Ongtang M, McLaughlin S, Enck W, et al. Semantically rich application-centric security in Android [J]. Security and Communication Networks, 2012, 5(6): 658-673

[29] Enck W, Gilbert P, Chun B G, et al. TaintDroid: An information-flow tracking system for realtime privacy monitoring on smartphones [C] //Proc of the 9th USENIX Conf on Operating Systems Design and Implementation. Berkeley: USENIX Association, 2010: 255-270

[30] Wei X, Gomez L, Neamtiu I, et al. Permission evolution in the Android ecosystem [C] //Proc of the 28th Annual Computer Security Applications Conf. New York: ACM, 2012: 31-40

[31] Au K W Y, Zhou Y F, Huang Z, et al. Pscout: analyzing the android permission specification [C] //Proc of the 2012 ACM Conf on Computer and Communications Security. New York: ACM, 2012: 217-228

[32] Felt A P, Chin E, Hanna S, et al. Android permissions demystified [C] //Proc of the 18th ACM Conf on Computer and Communications Security. New York: ACM, 2011: 627-638

[33] Nauman M, Khan S, Zhang X. Apex: Extending android permission model and enforcement with user-defined runtime constraints [C] //Proc of the 5th ACM Symp on Information, Computer and Communications Security. New York: ACM, 2010: 328-332

[34] Livshits B, Jung J. Automatic mediation of privacy-sensitive resource access in smartphone applications [C] //Proc of the 22nd Usenix Security Symp. Berkeley: USENIX Association, 2013: 113-130

[35] Felt A P, Wang H J, Moshchuk A, et al. Permission Re-Delegation: Attacks and Defenses [C] //Proc of the 20th Usenix Security Symp. Berkeley: USENIX Association, 2011: 331-346

[36] Dietz M, Shekhar S, Pisetsky Y, et al. QUIRE: Lightweight provenance for smart phone operating systems [C] //Proc of the 20th Usenix Security Symp. Berkeley: USENIX Association, 2011: 347-362

[37] Uellenbeck S, Dürmuth M, Wolf C, et al. Quantifying the security of graphical passwords: The case of android unlock patterns [C] //Proc of the 2013 ACM SIGSAC Conf on Computer & Communications Security. New York: ACM, 2013: 161-172

[38] Riva O, Qin C, Strauss K, et al. Progressive authentication: Deciding when to authenticate on mobile phones [C] //Proc of the 21st Usenix Security Symp. Berkeley: USENIX Association, 2012: 301-316

[39] Li L, Zhao X, Xue G. Unobservable re-authentication for smartphones [C/OL] //Proc of the 20th Annual Network and Distributed System Security Symp. 2013 [2014-03-20]. http://www. internetsociety. org/events/ndss-symposium-2013/papers-and-presentations

[40] Serwadda A, Phoha V V. When kids' toys breach mobile phone security [C] //Proc of the 2013 ACM SIGSAC Conf on Computer & Communications Security. New York: ACM, 2013: 599-610

[41] Templeman R, Rahman Z, Crandall D, et al. PlaceRaider: Virtual theft in physical spaces with smartphones [C/OL] //Proc of the 20th Annual Network and Distributed System Security Symp. 2013 [2014-03-20]. http://www. internetsociety. org/events/ndss-symposium-2013/papers-and-presentations

[42] WooYun. Android uncovers master-key vulnerability analysis [EB/OL]. [2013-12-09]. http://drops. wooyun. org/papers/219

[43] Wang Fei Fei. Study on detection and protection techniques of mobile phone malicious code under the Android platform [D]. Beijing: Beijing Jiaotong University, 2012 (in Chinese)
(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: 北京交通大学, 2012)

[44] Google Project Hosting. Androguard [EB/OL]. [2013-12-09]. https://code. google. com/p/androguard/

[45] Jarabek C, Barrera D, Aycock J. ThinAV: Truly lightweight mobile cloud-based anti-malware [C] //Proc of the 28th Annual Computer Security Applications Conf. New York: ACM, 2012: 209-218

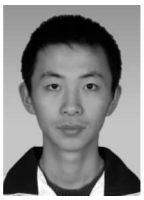
[46] Hornyack P, Han S, Jung J, et al. These aren't the droids you're looking for: Retrofitting android to protect data from imperious applications [C] //Proc of the 18th ACM Conf on Computer and Communications Security. New York: ACM, 2011: 639-652

[47] Xu R, Sa?di H, Anderson R. Aurasium: Practical policy enforcement for android applications [C] //Proc of the 21st Usenix Security Symp. Berkeley: USENIX Association, 2011

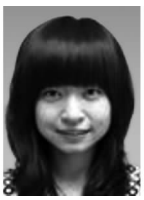
[48] Luo T, Hao H, Du W, et al. Attacks on WebView in the Android system [C] //Proc of the 27th Annual Computer Security Applications Conf. New York: ACM, 2011: 343-352

[49] Chin E, Felt A P, Greenwood K, et al. Analyzing inter-application communication in Android [C] //Proc of the 9th Int Conf on Mobile Systems, Applications, and Services. New York: ACM, 2011: 239-250

Zhang Yuqing, born in 1966. PhD. Professor in the University of Chinese Academy of Sciences. His research interests include network and information system security.



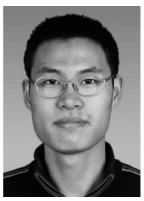
Wang Kai, born in 1990. PhD candidate in the School of Computer and Control Engineering, University of Chinese Academy of Sciences. His research interests is mobile security.



Yang Huan, born in 1984. PhD candidate in Xidian University. Her research interests include information security, vulnerability discovery and malicious code detection.

Fang Zhejun, born in 1986. PhD candidate in the School of Computer and Control Engineering, University of Chinese Academy of Sciences. His research interests include vulnerability mining and Android security.

Wang Zhiqiang, born in 1985. PhD candidate in Xidian University. His main research interests include network and information security.



Cao Chen, born in 1989. PhD candidate in the School of Computer and Control Engineering, University of Chinese Academy of Sciences. His research interest is mobile/Android security.