DOI:10.7544/issn1000-1239.2015.20150572
52(10):2167-2177,2015

Android 安全漏洞挖掘技术综述

张玉清[1] 方喆君[1,3] 王 凯[1] 王志强[2,4] 乐洪舟[2] 刘奇旭[1] 何 远[1] 李晓琦[1] 杨 刚[1]
[1](中国科学院大学国家计算机网络入侵防范中心 北京 101408)
[2](综合业务网理论及关键技术国家重点实验室(西安电子科技大学) 西安 710071)
[3](国家计算机网络应急技术处理协调中心 北京 100029)
[4](北京电子科技学院 北京 100070)
(zhangyq@ucas.ac.cn)

Survey of Android Vulnerability Detection

Zhang Yuqing[1], Fang Zhejun[1,3], Wang Kai[1], Wang Zhiqiang[2,4], Yue Hongzhou[2], Liu Qixu[1], He Yuan[1], Li Xiaoqi[1], and Yang Gang[1]
[1](*National Computer Network Intrusion Protection Center, University of Chinese Academy of Sciences, Beijing* 101408)
[2](*State Key Laboratory of Integrated Services Networks (Xidian University), Xi'an* 710071)
[3](*National Computer Network Emergency Response Technical Team/Coordination Center of China, Beijing* 100029)
[4](*Beijing Electronic Science and Technology Institute, Beijing* 100070)

Abstract Vulnerability plays a critical role in Android security. Therefore it is very meaningful to do research on vulnerability detection techniques, which can enhance Android security and protect user's privacy. In this paper, we firstly summary the number trends and categories of Android vulnerabilities from 2008 to 2015. Then we analyze the research progress of Android security from 2012 to 2014 and propose an overview of Android vulnerability detection techniques. After that, we detail the techniques frequently using in current researches, such as taint analysis, reachable path discovery, symbolic execution and fuzzing test. In addition, we also focus on the techniques combining static analysis and dynamic test such as concolic testing and directed fuzzing. At last, we conclude the status quo and open source tools in Android vulnerability detection, and propose valuable issues which are worth further studying.

Key words Android security; survey; vulnerability detection; static analysis; dynamic analysis

摘 要 安全漏洞在 Android 系统的安全性中处于核心地位,因此如何有效挖掘 Android 系统安全漏洞,已成为增强移动终端安全性、保护用户安全和隐私的重要技术手段,具有重要的理论和现实意义.首先对 Android 领域 2008—2015 年间漏洞数量趋势和种类进行了汇总,然后分类分析了 Android 安全领域顶级会议上 2012—2014 年间的学术研究进展.在此基础上,给出了 Android 漏洞挖掘技术的总体概览,并针对漏洞挖掘领域中使用较多的污染流传播分析、可达路径分析、符号执行、Fuzzing 测试等技术进行详细阐述,还对混合符号执行和定向 Fuzzing 等动静态结合的技术进行了介绍.最后对 Android 漏洞挖掘领域的开源工具进行了总结,并讨论了值得进一步深入研究的安全问题.

关键词 Android 安全;综述;漏洞挖掘;静态分析;动态分析

中图法分类号 TP309.1

收稿日期:2015-06-15;**修回日期**:2015-08-26

基金项目:国家自然科学基金项目(61272481,61572460);国家发改委信息安全专项(发改办高技[2012]1424 号)

截止 2014 年底 Android 操作系统的智能手机市场份额为 81.5%[1],与此同时 Android 官方市场 Google Play 中应用数量也从 2010 年的 8 万种增长至 2014 年的 143 万种[2],2014 年开发者数量更是达到了创纪录的 38.8 万. 但是,Android 应用开发者绝大多数都是小型创业公司和独立开发者,往往缺少大型软件公司常备的代码审计和代码安全规范检查,也缺乏足够的安全编程培训,这使得 Android 应用软件的安全隐患尤为严重.

这类和安全相关的缺陷和隐患统称为安全漏洞(vulnerability),安全漏洞可以在需求、设计、实现、配置、运行等软件生命周期的各个阶段中有意或无意产生,一旦被攻击者所利用,会对软件的机密性、完整性和可用性造成损害. 按照受影响对象的不同,在 Android 平台安全漏洞可分为系统漏洞和软件漏洞. 前者存在于 Android 系统的内核和框架层代码,包括 Android Open Source Project(AOSP)源代码以及手机厂商、驱动厂商的定制代码;后者主要影响系统原生应用和第三方开发的应用软件,包括各个手机厂商的自有应用和 Google Play、木蚂蚁等应用市场的海量应用. 按照攻击目标的不同,Android 漏洞可分为数据、权限、功能三大类,其中数据目标包括篡改、泄露等,权限目标包括滥用、提升、绕过、泄露等,功能目标包括破坏、泄露等.

安全漏洞在整个信息系统的安全性中处于核心地位,Android 系统虽然有精心设计过的安全体系,但是攻击者可以利用漏洞来绕过防护机制. 由于这个特性,大量恶意代码和安全事件往往围绕安全漏洞展开. 2011 年 6 月被截获的 Android. KungFu 病毒利用 Android 平台已公开的 NPROC_RLIMIT 漏洞获取手机 Root 权限并进行感染. 2012 年 RSA 会议中,Alperovitch 展示了 NickiSpy. C 木马,该木马利用了浏览器漏洞进行安装并记录用户通话内容[3-4]. 2014 年截获的 Pileup 病毒[5]会使用同年公布的 Android 升级机制漏洞[6]进行提权. 2014 年百度安全实验室检测到的“黑暗潜伏者”病毒[7],就使用 FramaRoot 漏洞组合利用工具[8]获取 Root 权限,进而控制受害者手机.

由此可见,安全漏洞是影响 Android 安全性的关键因素. 因此,如何在攻击者之前有效挖掘 Android 智能终端安全漏洞并进行及时修补,已成为增强移动终端安全性、保护用户安全和隐私的重要技术手段,具有重要的理论和现实意义. 但是,传统 PC 平台漏洞挖掘技术却无法在智能手机上直接应用. Android 操作系统应用多入口的程序架构与 PC 平台不同,直接影响了以程序逻辑分析为主的静态挖掘技术的使用;其次,智能手机更为关注以电话簿、短信记录和地理位置为代表的隐私数据,与 PC 平台相比这是一个显著的不同[9];最后,Android 操作系统实行较为严格的安全机制,在极大地限制了代码执行等漏洞的存在形式的同时也引入了新型的安全漏洞. 研究界在 Android 漏洞挖掘领域已做了大量工作,提出不同的应用静态分析技术和动态测试技术实现的漏洞挖掘方案,并取得了可观的成果.

1 Android 漏洞现状

近年来随着 Android 应用市场的飞速发展和漏洞挖掘技术的不断完善,漏洞数量也呈现井喷趋势增长. 漏洞增长趋势如图 1 所示,数据取自 National Vulnerability Database(NVD)截至 2015 年 6 月. 可以看到,2014 年是 Android 漏洞的一个高峰期,这主要是由于 SSL 协议类库出现了多个影响范围非常大的漏洞,导致大量应用受到波及,此类漏洞占到了当年漏洞总数的 92%(1401/1518). 除此之外,2014 年还出现 FakeID 代码注入漏洞[10]、TowelRoot 本地提权漏洞[11]和 Webview 组件 UXSS 漏洞[12]等多个严重级别系统漏洞,也引起了人们的重视.

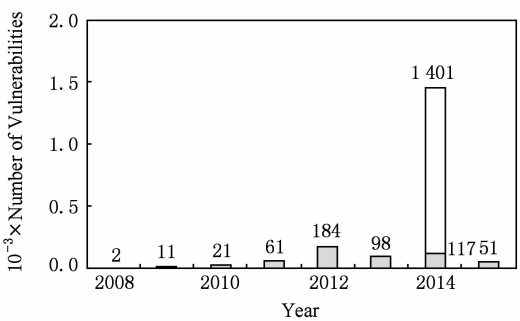


Fig. 1 Number of Android Vulnerabilities in NVD in Year 2008—2015.

图 1 2008—2015 年 NVD 中的 Android 漏洞数量变化

NVD 中 Android 漏洞类型排名统计如表 1 所示,可以发现除了 2014 年大爆发的加密错误类漏洞(CWE-310)之外,缓冲区类漏洞(CWE-119)、和权限相关的访问控制漏洞(CWE-264)、输入验证漏洞(CWE-20)和信息泄露漏洞(CWE-200)都是高发的漏洞类型,这几类漏洞也是研究者关注的重点.

总之,随着 AOSP 代码质量的提高,系统漏洞的数量在不断减少,但是应用市场的繁荣却导致应用漏洞的数量不断攀升,需要研究者格外重视.

Table 1 Top10 of Categories of Android Vulnerabilities in NVD

表 1 NVD 中的 Android 漏洞种类的前 10 名

Index	Vulnerability Category	Number
1	CWE-310, Cryptographic Issues	1 414
2	CWE-119, Buffer Errors	121
3	CWE-264, Permissions, Privileges, and Access Control	83
4	CWE-20, Input Validation	82
5	CWE-200, Information Leak/Disclosure	45
6	CWE-189, Numeric Errors	21
7	CWE-22, Path Traversal	12
8	CWE-79, Cross-Site Scripting (XSS)	11
9	CWE-94, Code Injection	8
10	CWE-399, Resource Management Errors	8

2 国内外研究现状

从 2012—2014 年间 S&P(IEEE Symposium on Security and Privacy)、CCS(ACM Conference on Computer and Communications Security)、Security(Usenix Security Symposium)、NDSS(ISOC Network and Distributed System Security Symposium)、ESORICS(European Symposium on Research in Computer Security)、RAID(International Symposium on Recent Advances in Intrusion Detection)、ACSAC(Annual Computer Security Applications Conference)、DSN(The International Conference on Dependable Systems and Networks)、IMC(Internet Measurement Conference)和 CSF(IEEE Computer Security Foundations Symposium)10 个顶级安全会议的论文发表情况可以发现,有 84 篇与 Android 安全有关.我们将这些论文按照研究领域进行了划分,其分布如图 2 所示:

从图 2 可以看出安全漏洞相关的研究(权限机制、系统安全、应用安全、隐私安全)占据核心地位,所占比例达到了 63%,这些文章针对不同研究内容各自使用了不同的静态或动态的漏洞挖掘方法,我们针对其中使用较多的污染流传播分析(taint analysis)、可达路径分析(reachable path analysis)、符号执行(symbolic execution)、Fuzzing 测试等技术详细阐述.总的来说,Android 漏洞挖掘技术的一个显著特点是强调对海量 APP 的检测,对扩展性和精确性要求较高,因而研究者尽量不使用类型系统、抽象解释等重量级分析技术.

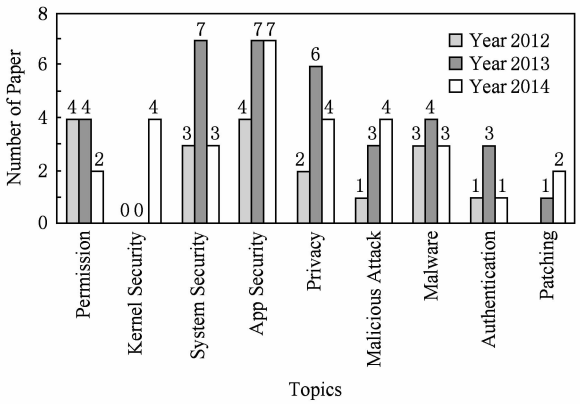


Fig. 2 Paper Statistics in Top Security Conferences in Year 2012—2014.

图 2 2012—2014 年顶级安全会议论文类型统计

上述漏洞挖掘技术的概览如图 3 所示.其中污点传播分析通过对数据流的分析来挖掘漏洞,分为动态和静态两种解决方案;与之形成对比的是,可达路径分析关注的是控制流分析.符号执行和可达路径分析都是静态技术,Fuzzing 测试是动态分析技术.为了能结合动态技术和静态技术的优点,研究者们也提出了将具体执行和符号执行结合起来的混合符号执行(concolic testing)技术,以及将 Fuzzing 测试和符号执行结合起来的定向 Fuzzing(directed fuzzing)技术,这 2 个技术也会在对应小结进行介绍.

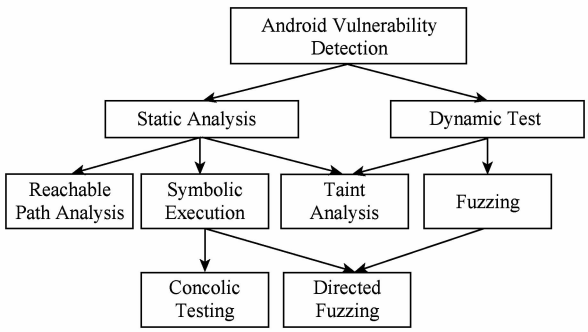


Fig. 3 Overview of Android vulnerability detection techniques.

图 3 Android 漏洞挖掘技术概览

3 污点传播分析

污点传播分析是根据数据依赖关系对程序进行分析的一种技术,常用于检测数据相关漏洞,比如隐私泄露漏洞和组件劫持漏洞.按照是否需要目标程序运行,污点传播分析可分为静态和动态两种类型.

静态方法主要利用控制流分析、数据流分析及

结构分析等技术[13],检测模式包括3部分:1)source,即污染源,是数据传播的起点,在程序执行路径上所有和source存在数据依赖关系的变量都将被标记为污染数据;2)sink,即陷入函数,如果有污染数据进入sink就可认为存在安全隐患;3)sanitizer,即验证函数,这类函数会对污染的有效性和合法性进行检查,所有流入sanitizer的污染数据会去污染化,被重新标记成普通数据.

静态污点传播分析技术可用于检测Android应用组件劫持漏洞,通过在对外开放的接口和内部敏感sink之间进行传播分析来挖掘漏洞.2012年Lu等人[14]依此思路实现了自动化检测工具CHEX.CHEX先是自动检索待检测App所有入口点,再通过程序分割来进行全局数据流分析,并对每一个入口点都精确建模其执行过程.Lu等人还实现了一个针对Android字节码进行静态分析的框架Dalysis,CHEX正是在Dalysis的基础上实现.

静态污点传播分析技术还常用于检测隐私泄露漏洞.此时检测模式3元组中的source可被定义为敏感数据的访问操作,sink为向外发送数据的操作,sanitizer通常不做定义.这样若找到一条从source到sink的数据传播路径,就可认定存在隐私泄露漏洞.2012年,Gibler等人[15]依此思路提出了一个大规模检测应用程序中的隐私信息泄露漏洞的工具AndroidLeaks.AndroidLeaks通过自动化分析Android Framework代码来提取API和权限之间的对应关系,并应用这个对应关系来制定sink列表,发现了应用市场中大量存在的电话信息、位置信息、Wifi信息和录音信息等隐私泄露问题.AndroidLeaks是基于WALA[16]实现的,能有效处理Android组件的生命周期和回调函数.

但是由于Java代码存在隐式函数调用(如虚函数等),静态分析的时候难以有效确定真实调用目标.同时,由于缺乏精确的字符串分析,污染流传播分析对于Android组件间通信机制的适配性较差,无法准确确定通信目标,污染流也就无法传播到其他应用,难以检测多个恶意应用配合下的共谋攻击.为解决该问题,2013年Octeau等人[17]提出了一个较为精确的组件间通信机制的寻址方案,他们通过将组件间通信的分析问题转化为IDE(interprocedural distributive environment)问题,再应用现有算法[18]来求解.

Android异步工作机制和应用生命周期模型也为分析带来了不小的困难.为了解决该问题,2014年Arzt等人[19]在Soot[20]的基础上实现了Flowdroid,该工具是一个上下文相关、流敏感和生命周期敏感的污染流传播分析工具,该工作利用Heros IFDS/IDE框架来解决过程间分析问题并针对Android系统的应用生命周期和异步工作机制进行了大量优化,并对外提供源码.

除此之外,sink和source的选择也会影响污染传播分析的精度,研究者引入机器学习技术来试图解决该问题.为了解决sink列表和source列表存在错误的问题,2014年Tripp等人[21]提出了通过Bayesian模型细分Sink类型的方案,来排除正常信息发送对污染流传播分析中的误报影响.为了提高sink列表和source列表的完备性,同年Rasthofer等人[22]提出了通过机器学习方式补完source和sink列表的研究工作.

以上是静态污点传播分析的研究工作,与之形成鲜明对比,动态污点传播分析技术是在定制的Android沙箱(sandbox)中动态连续跟踪数据的流向,目前动态污点跟踪技术广泛用于漏洞挖掘等方面[23-25].将动态污点跟踪应用于检测隐私数据泄露时,将隐私数据作为source,然后动态监控其传播,当监测到这些污点数据通过sink被发送出去时,则可以判定该程序存在隐私泄露的问题.

2010年,Enck等人[26]在Android平台上实现了一个名为TaintDroid的动态污染追踪和分析系统,利用Android虚拟机框架从变量、方法、消息和文件4个角度进行细粒度的内存级动态污染传播分析,实时截获隐私泄露攻击,是目前较为完备和开源的研究工作.在它基础上,开源项目Droidbox实现了对程序行为和传播数据的细粒度监控,并得到广泛应用.2011年Hornyack等人[27]、Beresford等人[28],2012年Yang等人[29]的研究都在TaintDroid的基础上进行了扩展,弥补了部分缺点和不足.

总的来说,静态和动态污点传播分析技术均能有效地挖掘和数据流相关的漏洞.但是,静态污点传播分析技术目前主要的问题仍然是对动态加载、组件间通信、异步机制等Android特性造成的数据流中断,污染流的传播寻址存在困难;而动态污点传播分析存在能够检测的隐私数据类型不够全面,隐私泄露点不完备,污染追踪粒度不够细、存在"过污染"和"欠污染"等问题.

4 可达路径分析

可达路径分析是静态分析技术的一种,和污染

流追踪不同,可达路径分析更关注控制流分析和控制流分支条件,因此在检测混淆代理漏洞(confused deputy vulnerability)方面格外有效.混淆代理漏洞是由于存在预期外的控制流而使得预设的安全机制被绕过而产生的漏洞,包括功能泄露(capability leaks)、隐私泄露和污染(privacy leaks and pollution)、权限重委派(permission re-delegation)、组件劫持(component hijacking)等.如果能发现程序某个入口点和敏感 API 调用点之间存在控制流上的可达路径,则可以认定存在混淆代理漏洞.

控制流分析[30]是针对程序控制流的分析技术,根据程序执行方向将代码解析成一个有向图 $G=(B,E)$,其中 B 是基本块构成的节点集合$\{B_1,B_2,\cdots,B_n\}$,E 是有向边集合$\{(B_i,B_j),(B_k,B_l),\cdots\}$,其中$(B_i,B_j)$这条有向边表示在某个执行序列中 B_j 位于 B_i 之后.随后再通过对图连通性质的研究来提取代码中的可达路径以及路径之上的约束求解.

可达路径分析技术能有效检测混淆代理漏洞,特别是针对其各个子类漏洞的挖掘已经有大量的研究工作.2011 年,Felt 等人[31]提出了一个基于控制流分析来检测 Android 组件间通信机制中的权限重委派漏洞的方案,并给出了解决方案 IPC Inspection.IPC Inspection 是对 Android 原有权限机制的增强,核心策略是当低权限应用调用高权限应用时,高权限应用的权限将临时被降低至二者的权限交集.

2012 年,Grace 等人[32]提出了检测应用程序中显式和隐式功能泄露漏洞的检测工具 Woodpecker,该工具先将 Dalvik 字节码变成 smali 格式,采用过程间分析抽取控制流图,得到从程序进入点传播到敏感 API 的路径.这一步会提取出大量无效路径,所以需要进行剪枝.通过过程间数据流分析,采用域和路径敏感的符号执行的办法,筛选得到可达路径.如果可以提取出从程序进入点到敏感 API 的可达路径,则认为存在显式的功能泄露.而对于隐式的功能泄露则还要分析初始化代码,如果一个应用会去执行任何他没有申请的权能,就可以认为其存在隐式功能泄露.

2014 年,Zhang 等人[33]针对组件劫持漏洞(component hijacking vulnerability,混淆代理漏洞的一个子类),提出了检测及修补框架 AppSealer,该工具不但通过可达路径分析可以发现此类漏洞,还能做到自动化地为应用程序增加补丁.

除此之外,由于可达路径分析技术的准确性部分依赖于敏感 API 的列表,需要系统分析 Android 权限机制,可通过静态分析技术从 Android 整个系统源代码中抽取包含权限的 API 操作方法来提取敏感 API 和权限的映射关系.2012 年 Au 等人[34]完成了该工作,发现大量未公开的包含权限的敏感 API,并向公众开放结果.

在 Android 平台可达路径分析技术能有效地检测和控制流相关的漏洞.但应用的主要困难是 Android 应用生命周期模型和回调函数等异步机制对控制流的割裂.目前主流的解决方法是根据 Android 系统工作机理将断裂的控制流接续起来[35],对于 Android 应用的多入口点问题,根据生命周期模型将不同的入口点函数整合在一个顺序执行的主函数体内;对于回调函数,将函数调用点人工设置在对应组件的适当位置[36].这些机制虽然不够精确,但是却能最大程度上在不违反直观的程序执行预测情况下,将控制流重新接续.

5 符号执行

符号执行技术[37]是一种对程序变量进行精确计算的技术.符号执行采用抽象符号代替程序变量,并模拟程序执行.最终所有程序变量都是由符号组成的表达式构成,再通过约束求解引擎对其进行求解,获得深度语义信息.

符号执行能够在复杂的数据依赖关系中发现变量之间本质的约束关系,比污点传播分析精度高,这种精确的数据流分析能帮助理解程序的内在逻辑;在模拟程序运行的过程中,符号执行还会精确记录执行路径上所有的约束条件,可以提高控制流分析中路径可达性问题判定的精确性.

符号执行的劣势在于路径执行空间爆炸问题难以解决;需要功能强大的符号计算系统支撑,对硬件计算能力要求较高;符号执行很难处理循环或者递归,因为很多时候对结束条件的取值的判定并不总能得到精确解;最后,符号执行对基础数据类型较为有效,对于复杂结构的数据类型(比如数组等)处理难度较高.

为了解决上述问题,近年来研究者倾向于将符号执行和实际的程序执行相结合,当符号执行遇到无法判定或者计算取值的情况时,使用具体值来代替部分符号值,使符号执行能够继续进行.这种结合了具体执行和符号执行的技术通常被称为混合符号执行技术[38].

符号执行技术在 Android 漏洞挖掘领域的应用从 2012 年开始较多出现. 为了降低计算复杂度,一个符号执行的实现方案是将 Dalvik 字节码转换成某种更简单更适于分析的第三方语言. 2012 年 Jeon 等人[39]依此思路提出了一个纯静态的针对 Dalvik 字节码的符号执行框架 SymDroid. Symdroid 首先将 Dalvik 字节码转换成 μ-Dalvik 字节码,该语言只包含 16 条指令,是有 200 多条指令的 Dalvik 语言的简化版本. 同时,Symdroid 还对 Android 平台部分系统库和生命周期控制代码(如 Bundle 和 Intent)进行了建模,使符号执行能适应事件驱动的 Android 平台.

和 Jeon 等人[39]工作形成鲜明对比,同年 Anand 等人[40]提出了基于 concolic executor 的 Android 应用符号执行框架 ACTEve,能自动生成事件序列来模拟用户交互行为. ACTEve 使用 ded[13,41]来将 Dalvik 字节码转换成 Java 字节码(该过程不是无损的),然后使用 Soot 框架[42]来进行符号执行. 但是最关键的如何处理本地代码以及 Android 运行时代码,ACTEve 并没有详述.

基于 Java PathFinder 的 Symbolic PathFinder[43]是一个较为成熟的混合符号执行框架. 要想将该工具移植到 Android 平台,需要将 Dalvik 字节码变换为 Java 字节码[44],技术难点是 Android 程序与 JVM 架构的不兼容性、Android 程序生命周期导致的路径分支判断问题以及如何计算 API 调用参数与返回值之间的符号关系.

利用符号执行进行精确的值推断可以提高漏洞挖掘的效率和精度. 隐私泄露行为检测的一个难点是无法区分用户授权的和非用户授权的隐私发送行为,这会导致很严重的误报. 一个解决方案是利用符号执行技术生成 GUI 交互事件序列[45],这样就可以将隐私泄露授权判定问题转化为事件序列是否由用户驱动的判定问题.

在隐私泄露漏洞检测中,符号执行还可用于精确量化特定行为对隐私的破坏程度. 2014 年,Cox 等人[46]提出了追踪隐式信息流来保护应用程序密码安全的方案 SpanDex. SpanDex 使用一个数据流敏感的安全沙箱来运行不可信代码,并通过动态符号执行和约束求解来精确量化一个信息传递行为对隐私的破坏程度. SpanDex 最大的优点是不需要修改待测程序,也不会造成性能的严重下降.

总之,Android 平台符号执行技术通常在现有的 Java 字节码符号执行技术(如 JPF)基础上实现 Dalvik 字节码的符号执行技术,以及在现有可执行代码符号执行技术(KLEE)的基础上实现 Android 本地代码的符号执行技术,但是目前的解决方案仍然不成熟,是下一步研究的重点.

6 Fuzzing 测试

Fuzzing 测试又称模糊测试,是黑盒测试技术,通过不断生成大量畸形测试数据来测试程序的鲁棒性和安全性. 该技术的核心正是测试用例生成技术,良好的测试用例生成技术能保证更高的代码覆盖率和测试效率. 而测试用例生成技术,又可细分为生成型和变异型两种,生成型根据预设的规则直接生成测试用例,而变异型是通过给定的种子测试用例不断生成新用例.

1990 年,Miller 等人[47]提出了 Fuzzing 测试的思想,并对 Unix 系统的命令行工具进行了测试,发现了大量程序鲁棒性问题. 在这之后的若干年里,Miller 等人再次对 Unix 的网络服务和 GUI 界面程序[48]、Windows NT 系统的 GUI 界面程序[49]、Mac OS X 系统的命令行和 GUI 程序[50]进行了 Fuzzing 测试. 从结果看,经过 15 年的发展,这些系统核心程序鲁棒性有所增强,但仍存在相当程度的崩溃现象.

在 Android 平台,该技术被广泛用于测试组件间通信机制[51-52]、关键系统组件和系统功能[53]、NFC 等通信模块[54]、核心类库等,取得了大量成果. 2009 年,Mulliner 和 Miller[53]对 Android、iPhone 和 Windows Mobile 平台上的短信功能进行了 Fuzzing 测试,通过在底层 modem 和用户层之间的信道上插入测试程序的方式,极大地提高了测试效率,并且不需要基站辅助. 发现了 3 个平台上的大量短信处理程序崩溃漏洞. 使用类似的思路,2013 年 Stirparo 等人[54]、2014 年王志强等人[55]对同样开源的 NFC 协议数据交换格式的不同字段进行了随机化测试.

对于历史上出现过大量漏洞的 Android 组件间通信机制,Fuzzing 技术可以有效挖掘其漏洞[51-52]. 首先通过二进制代码的静态分析和动态监控来获得 Intent(Android 应用通信传递媒介)中的数据格式,再采用生成型测试用例生成技术,拼接组成测试用 Intent 并发送给待测组件. 这之后,利用定制的 Android 应用程序沙盒监控待测组件,检测可能发生的崩溃或者权限泄露.

同时,由于定向测试和深度测试的需要,对

Android 原生的 Monkey 等基于事件的自动化测试工具进行改进也是一个研究方向.为了改进 Monkey 工具目标不明确、随机性强的缺点,可对 Android 应用建立 GUI 测试模型[56],预测可能的输入并对组件和窗口进行追踪以减少重复性探索[57].

由于 Fuzzing 测试技术存在路径覆盖率低、效率差等缺点,使用静态分析技术来提高测试效率的定向 Fuzzing 方向也有不少研究.2012 年,Zheng 等人[58]提出了针对 Android 恶意软件的自动化测试框架 SmartDroid,SmartDroid 使用动静结合的分析技术查找与敏感 API 相联系的 UI 组件,以便于找到执行到敏感 API 的程序执行路径,触发敏感的程序行为.

2014 年,Bhoraskar 等人[59]提出了专门针对 Android 第三方库的定向安全测试工具 Brahmastra,该工具使用静态分析的方法构建页面转移图并找出所有能够触发第三方库程序的程序执行路径.然后采用二进制重写的方法沿着程序执行路径进行跳跃执行,有效地减少了无效执行路径.

目前 Fuzzing 测试仍然是挖掘漏洞最有效、最多产的方法,在工业界得到广泛使用.目前主要研究热点是通过可达路径分析、符号执行等技术来指导 Fuzzing,以及利用语法制导来针对高度结构化和有严格语法的数据进行 Fuzzing,提高路径覆盖率和测试效率.

7 现有工作总结

Android 平台漏洞挖掘技术经过多年的快速发展,涌现了大批科研成果.其中可达路径分析技术发展较为成熟,能有效挖掘与控制流相关的漏洞,特别是混淆代理漏洞.污点传播分析技术发展也较为完备,能有效地从不同分析精度来挖掘与数据流相关的漏洞,特别是隐私泄露漏洞.Fuzzing 技术是挖掘漏洞最有效的技术,在现有云计算和并行计算技术的支持下针对绝大多数漏洞类型均有优异的表现.符号执行技术作为高精度挖掘技术,在 Android 平台的应用尚不成熟,值得进一步研究.

目前已有不少优秀开源和非开源安全工具出现,研究者利用这些工具,挖掘出了大量系统和应用中的安全漏洞,极大地促进了 Android 生态圈的安全.后续研究者可以利用这些成果开展自己的研究工作.较为成熟的 Android 开源工具汇总如表 2 所示:

Table 2 Open Source Tools for Android Vulnerability Detection

表 2 Android 漏洞挖掘开源工具汇总

Technique Category		Open Source Tool
Taint Analysis	Static	Flowdroid
	Dynamic	TaintDroid,Droidbox
Symbolic Execution		Java PathFinder,KLEE,Android_S2E
Fuzzing		Peach Fuzzer,Intent Fuzzer
Others		WALA,Soot,Pscout,Androguard,Drozer

污点传播分析中 TaintDroid 和 Droidbox 已经发展较为成熟,基于它们的可供参考的研究工作也较多.Flowdroid 出现较晚,但是在静态污点传播分析领域性能较好.

符号执行领域的这些工具都是从 PC 平台上较为成熟的符号执行工具迁移过来,因此使用中需要做的工作就是根据 Android 平台 Java 虚拟机和 Dalvik 虚拟机、ARM 架构和 x86 架构的不同进行合理的移植.

Fuzzing 测试工具中 Peach 是一个通用的测试框架,用于根据预先给定的规则生成测试数据.Intent Fuzzer 是专门针对 Android 组件间通信机制的 Fuzzing 测试工具,但是还只是一个原型系统,只能测试空类型的数据.

其他类型的工具中,Pscout 并没有公布源码,而只是公布了分析结果,即不同 Android 版本中公开或未公开 API 和权限的映射表.而 WALA 和 Soot 是目前最流行的 2 个通用分析框架,支持过程间分析、上下文敏感分析、程序切片、指针分析等功能.在这 2 个工具的基础上诞生了大量研究成果,就目前来看基于 Soot 的开源工具更多一些,Flowdroid 就是在 Soot 和 Heros 的基础上完成.Androguard 和 Drozer 是 Android App 安全审计工具,支持结构分析、调用图分析等功能,常用于 Android 应用的恶意代码分析,也能进行简单的漏洞挖掘工作.

总的来说,目前 Android 漏洞挖掘领域的开源工具已经较为完备,研究者可以很容易找到这些工具的源码和相关文档,并在其基础上开发自己的工具.

8 下一步研究方向

Android 漏洞挖掘领域的研究已经开展多年,整体的技术体系已经基本构建完毕,产生了一大批科研成果和 0day 漏洞.下一步研究应该更多地关注

在提高漏洞挖掘效率和覆盖率、扩展新目标上,具体存在4个方向:

1) 符号执行技术在Android漏洞挖掘领域的应用.只根据数据依赖关系进行漏洞挖掘的精度过低,需要应用符号执行技术来提高分析精度,才能更好地挖掘Android漏洞.具体来说,进行控制流分析和污染流传播分析的时候,由于Android组件间通信机制和Java反射技术会导致控制流中断,需要使用符号执行技术来进行精确的字符串分析,才能明确通信目标,将被隔断的控制流接续起来.这是一个值得进一步探讨的问题.

2) 静态分析辅助下的Fuzzing自动化测试.为了解决Fuzzing测试覆盖率不高、测试效率低问题,需要预先推断出输入需要满足的数值条件.这需要应用符号执行技术和约束求解技术,这2个技术由于包含大量的运算,并没有得到广泛应用.但是,Android应用普遍代码量较小,在现有的计算性能下可以满足上述技术的计算需求.因此,需要研究基于符号执行和可达路径分析的定向Fuzzing技术,来提高Fuzzing测试效率和覆盖率.这可以作为后续工作的一个重点.

3) 代码反加固技术在Android漏洞挖掘领域的应用.近年来大量Android应用软件使用安全加固技术来提高反编译、反调试、反篡改的能力,这对于以可执行代码为目标的静态分析技术带来了较大的阻碍.因此,有必要研究Android代码反加固技术,才能从混淆和加固后的代码中提取出真正的可执行代码,配合静态分析技术完成漏洞挖掘工作.

4) 对系统功能模块和第三方类库安全性的研究.目前,漏洞挖掘技术主要应用于针对Android系统应用层的漏洞挖掘,如组件暴露、功能泄露等,系统层的功能模块以及应用开发所依赖的第三方类库上的漏洞挖掘技术虽然取得了一定的突破,但所研究部分仅占诸多Android系统模块和第三方类库种类的小部分,研究成果较为有限.一方面,智能穿戴、智能家居等新兴应用场景使Android智能设备的功能模块不断更新,导致安全需求也随之不断更新变化;另一方面,功能日益丰富的应用程序在开发过程中开始使用功能和形式各异的第三方类库,但目前第三方类库的安全性尚缺少有效的分析工具.相比单一应用,系统模块和第三方类库漏洞的影响更严重、危害范围更广泛,并且,由于其实现的复杂性,现有的动静态漏洞挖掘技术很难直接得到应用,需要研究者进一步的研究.

9 结束语

安全漏洞是Android安全的核心,Android平台近年来的安全事件和恶意代码都围绕安全漏洞展开,研究安全漏洞的挖掘具有重要的现实意义.本文首先对Android安全领域近3年来的学术研究进展特别是漏洞挖掘领域的研究进行了汇总,再针对使用较多的污染流传播分析、可达路径分析、符号执行、Fuzzing测试等技术详细阐述.最后对Android漏洞挖掘领域的研究现状和开源工具进行了总结,并讨论了值得进一步深入研究的安全问题.

参 考 文 献

[1] Egham. Gartner says sales of smartphones grew 20 percent in third quarter of 2014 [EB/OL]. [2015-06-01]. http://www.gartner.com/newsroom/id/2944819

[2] 199IT. appFigures: The number of applications in Google Play for the first time surpass App Store in 2014 [EB/OL]. [2015-06-01]. http://www.199it.com/archives/321519.html (in Chinese)
(199IT. Appfigures: 2014年Google play增加新应用数量首次超过App store [EB/OL]. [2015-06-01]. http://www.199it.com/archives/321519.html)

[3] TheHackerNews. Zero-day smartphone vulnerability exposes location and user data [EB/OL]. [2015-06-01]. http://thehackernews.com/2012/02/zero-day-smartphone-vulnerability.html

[4] Schlegel R, Zhang K, Zhou X, et al. Soundcomber: A stealthy and context-aware sound trojan for smartphones [C]//Proc of the 18th Network and Distributed System Security. San Diego: NDSS, 2011: 17-33

[5] Nerney C. Sneaky "pileup" malware can exploit android upgrades, researchers say [EB/OL]. [2015-06-01]. http://www.citeworld.com/article/2114420/mobile-byod/android-pileup-malware.html

[6] Xing L, Pan X, Wang R, et al. Upgrading your android, elevating my malware: Privilege escalation through mobile os updating [C]//Proc of 2014 IEEE Symp on Security and Privacy (SP). Piscataway, NJ: IEEE, 2014: 393-408

[7] Jia Jack. "DARK"—New Attack Mode of Smartphone Virus[EB/OL]. 2014 [2015-06-01]. http://blog.csdn.net/androidsecurity/article/details/412079590 (in Chinese)
(Jia Jack. "黑暗潜伏者"——手机病毒新型攻击方式[EB/OL]. 2014 [2015-06-01]. http://blog.csdn.net/androidsecurity/article/details/41207959)

[8] Alephzain. Framaroot, a one-click apk to root some devices [EB/OL]. [2015-06-01]. http://forum.xda-developers.com/apps/framaroot/root-framaroot-one-click-apk-to-root-t2130276

[9] Zhang Yuqing, Wang Kai, Yang Huan, et al. Survey of Android OS security [J]. Journal of Computer Research and Development, 2014, 51(7): 1385-1396 (in Chinese)
(张玉清, 王凯, 杨欢, 等. Android 安全综述[J]. 计算机研究与发展, 2014, 51(7): 1385-1396)

[10] Kingkong Team. Analysis of Android Fake ID Vulnerability which can Lead to Arbitrary Code Injection and Execution [EB/OL]. [2015-06-01]. http://security.tencent.com/index.php/blog/msg/64 (in Chinese)
(金刚项目团队. Android fakeid 任意代码注入执行漏洞简析[EB/OL]. [2015-06-01]. http://security.tencent.com/index.php/blog/msg/64)

[11] National Vulnerability Database. Nvd vulnerability summary for cve-2014-3153 [EB/OL]. [2015-06-01]. http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2014-3153

[12] National Vulnerability Database. Nvd vulnerability summary for cve-2014-6041 [EB/OL]. [2015-06-01]. http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2014-6041

[13] Enck W, Octeau D, McDaniel P, et al. A Study of Android Application Security [C] //Proc of the 20th USENIX Security Symp. Berkeley, CA: USENIX Association, 2011: No. 2

[14] 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

[15] Gibler C, Crussell J, Erickson J, et al. AndroidLeaks: Automatically Detecting Potential Privacy Leaks in Android Applications on a Large Scale [M]. Berlin: Springer, 2012

[16] IBM T. J. Watson Research Center. WALA [EB/OL]. [2015-06-01]. http://wala.sourceforge.net/wiki/index.php/Main_Page

[17] Octeau D, McDaniel P, Jha S, et al. Effective inter-component communication mapping in android with epicc: An essential step towards holistic security analysis [C] //Proc of USENIX Security 2013. Berkeley, CA: USENIX Association, 2013: 543-558

[18] Sagiv M, Reps T, Horwitz S. Precise interprocedural dataflow analysis with applications to constant propagation [J]. Theoretical Computer Science, 1996, 167(1): 131-170

[19] Arzt S, Rasthofer S, Fritz C, et al. Flowdroid: Precise context, flow, field, object-sensitive and lifecycle-aware taint analysis for android apps [C] //Proc of ACM SIGPLAN Notices. New York: ACM, 2014, 49(6): 259-269

[20] Lam P, Bodden E, Lhoták O, et al. The Soot framework for Java program analysis: A retrospective [C/OL] //Proc of Cetus Users and Compiler Infastructure Workshop. 2011 [2015-09-01]. http://cetus.ecn.purdue.edu/cetusworkshop/papers/slides.3-3.pdf

[21] Tripp O, Rubin J. A Bayesian approach to privacy enforcement in smartphones [C] //Proc of the 23rd USENIX Security Symp. Berkeley, CA: USENIX Association, 2014: 175-190

[22] Rasthofer S, Arzt S, Bodden E. A machine-learning approach for classifying and categorizing android sources and sinks [C] //Proc of 2014 Network and Distributed System Security Symp (NDSS). San Diego: NDSS, 2014: 1-15

[23] Sabelfeld A, Myers A C. Language-based information-flow security [J]. IEEE Journal on Selected Areas in Communications, 2003, 21(1): 5-19

[24] Ligatti J, Bauer L, Walker D. Edit automata: Enforcement mechanisms for run-time security policies [J]. International Journal of Information Security, 2005, 4(1/2): 2-16

[25] Wang T, Wei T, Gu G, et al. TaintScope: A checksum-aware directed fuzzing tool for automatic software vulnerability detection [C] //Proc of 2010 IEEE Symp on Security and Privacy (SP). Piscataway, NJ: IEEE, 2010: 497-512

[26] Enck W, Gilbert P, Han S, et al. TaintDroid: An information-flow tracking system for realtime privacy monitoring on smartphones [J]. ACM Trans on Computer Systems (TOCS), 2014, 32(2): No. 5

[27] 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

[28] Beresford A R, Rice A, Skehin N, et al. Mockdroid: Trading privacy for application functionality on smartphones [C] //Proc of the 12th Workshop on Mobile Computing Systems and Applications. New York: ACM, 2011: 49-54

[29] Yang Guangliang, Gong Xiaorui, Yao Gang, et al. A privacy leakage detection system for Android [J]. Computer Engineering, 2012, 38(23): 1-6 (in Chinese)
(杨广亮, 龚晓锐, 姚刚, 等. 一个面向 Android 的隐私泄露检测系统[J]. 计算机工程, 2012, 38(23): 1-6)

[30] Allen F E. Control flow analysis [C] //Proc of ACM Sigplan Notices. New York: ACM, 1970, 5(7): 1-19

[31] Felt A P, Wang H J, Moshchuk A, et al. Permission re-delegation: Attacks and defenses [C] //Proc of USENIX Security Symp. Berkeley, CA: USENIX Association, 2011: 1-16

[32] Grace M C, Zhou Y, Wang Z, et al. Systematic detection of capability leaks in stock Android smartphones [C] //Proc of the 19th Network and Distributed Security Symp (NDSS). San Diego: NDSS, 2012: 1-15

[33] Zhang M, Yin H. Appsealer: Automatic generation of vulnerability-specific patches for preventing component hijacking attacks in android applications [C] //Proc of the 21st Annual Network and Distributed System Security Symp (NDSS). San Diego: NDSS, 2014: 1-15

[34] 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

[35] Cao Y, Fratantonio Y, Bianchi A, et al. EdgeMiner: Automatically detecting implicit control flow transitions through the Android framework [C] //Proc of the ISOC Network and Distributed System Security Symp (NDSS). San Diego: NDSS, 2015: 1-15

[36] Fang Z, Liu Q, Zhang Y, et al. IVDroid: Static detection for input validation vulnerability in Android inter-component communication [C] //Proc of the 11th Int Conf on ISPEC. Berlin: Springer, 2015: 378-392

[37] King J C. Symbolic execution and program testing [J]. Communications of the ACM, 1976, 19(7): 385-394

[38] Sen K. Concolic testing [C] //Proc of the 22nd IEEE/ACM Int Conf on Automated Software Engineering. New York: ACM, 2007: 571-572

[39] Jeon J, Micinski K K, Foster J S. SymDroid: Symbolic execution for Dalvik bytecode, cs-tr-5022 [R]. Maryland: Department of Computer Science, University of Maryland, 2012

[40] Anand S, Naik M, Harrold M J, et al. Automated concolic testing of smartphone apps [C] //Proc of the 20th ACM SIGSOFT Int Symp on the Foundations of Software Engineering. New York: ACM, 2012: No. 59

[41] Octeau D, Jha S, McDaniel P. Retargeting Android applications to Java bytecode [C] //Proc of the 20th ACM SIGSOFT Int Symp on the Foundations of Software Engineering. New York: ACM, 2012: No. 6

[42] GitHub. Soot: A framework for analyzing and transforming java and android applications [EB/OL]. [2015-06-01]. http://sable.github.io/soot/

[43] Nasa Ames Research Center. Java pathfinder [EB/OL]. [2015-06-01]. http://babelfish.arc.nasa.gov/trac/jpf

[44] Mirzaei N, Malek S, Păsăreanu C S, et al. Testing android apps through symbolic execution [J]. ACM SIGSOFT Software Engineering Notes, 2012, 37(6): 1-5

[45] 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

[46] Cox L P, Gilbert P, Lawler G, et al. Spandex: Secure password tracking for android [C] //Proc of the USENIX Security Symp (Sec). Berkeley, CA: USENIX Association, 2014: 481-494

[47] 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

[48] Koski D, Lee C P, Maganty V, et al. Fuzz Revisited: A Re-examination of the Reliability of UNIX Utilities and Services [M]. Madison, Wisconsin: Computer Sciences Department, University of Wisconsin-Madison, 1995

[49] Forrester J E, Miller B P. An empirical study of the robustness of Windows NT applications using random testing [C] //Proc of the 4th USENIX Windows System Symp. Berkeley, CA: USENIX Association, 2000: 59-68

[50] Miller B P, Cooksey G, Moore F. An empirical study of the robustness of macos applications using random testing [C] //Proc of the 1st Int Workshop on Random Testing. New York: ACM, 2006: 46-54

[51] Wang Kai, Liu Qixu, Zhang Yuqing. Android inter-application communication vulnerability mining technique based on Fuzzing [J]. Journal of University of Chinese Academy of Sciences, 2014, 31(6): 827-835 (in Chinese)
(王凯, 刘奇旭, 张玉清. 基于 Fuzzing 的 Android 应用通信过程漏洞挖掘技术[J]. 中国科学院大学学报, 2014, 31(6): 827-835)

[52] Yang K, Zhuge J, Wang Y, et al. IntentFuzzer: Detecting capability leaks of android applications [C] //Proc of the 9th ACM Symp on Information, Computer and Communications Security. New York: ACM, 2014: 531-536

[53] Miller C, Mulliner C. Fuzzing the Phone in your Phone [C/OL] //Black Hat Technical Security Conference USA. [2009-06-25]. https://www.blackhat.com/presentations/bh-usa-09/MILLER/BHUSA09-Miller-FuzzingPhone-PAPER.pdf

[54] Stirparo P. A fuzzing framework for the security evaluation of NDEF message format [C] //Proc of the 5th Int Conf on Computational Intelligence, Communication Systems and Networks (CICSyN). Piscataway, NJ: IEEE, 2013: 165-170

[55] Wang Zhiqiang, Liu Qixu, Zhang Yuqing. Research of discovering vulnerabilities of NFC applications on Android platform [J]. Journal on Communications, 2014, (Z2): 117-123 (in Chinese)
(王志强, 刘奇旭, 张玉清. Android 平台 NFC 应用漏洞挖掘技术研究[J]. 通信学报, 2014, (Z2): 117-123)

[56] Amalfitano D, Fasolino A R, Tramontana P, et al. Using GUI ripping for automated testing of Android applications [C] //Proc of the 27th IEEE/ACM Int Conf on Automated Software Engineering. New York: ACM, 2012: 258-261

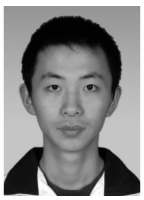
[57] Rastogi V, Chen Y, Enck W. AppsPlayground: automatic security analysis of smartphone applications [C] //Proc of the 3rd ACM Conf on Data and Application Security and Privacy. New York: ACM, 2013: 209-220

[58] Zheng C, Zhu S, Dai S, et al. Smartdroid: an automatic system for revealing ui-based trigger conditions in android applications [C] //Proc of the 2nd ACM Workshop on Security and Privacy in Smartphones and Mobile Devices. New York: ACM, 2012: 93-104

[59] Bhoraskar R, Han S, Jeon J, et al. Brahmastra: Driving apps to test the security of third-party components [C] //Proc of the 23rd USENIX Security Symp (USENIX Security 14). Berkeley, CA: USENIX Association, 2014: 1021-1036

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

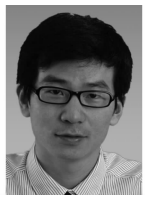
Fang Zhejun, born in 1986. PhD candidate in the School of Computer and Control Engineering, University of Chinese Academy of Sciences. His research interests include vulnerability mining and Android security.



Wang 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.

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

Yue Hongzhou, born in 1987. PhD candidate in Xidian University, accepting joint training in National Computer Network Intrusion Protection Center, University of Chinese Academy of Sciences. His main research interests include network and information security.



Liu Qixu, born in 1985. PhD. Lecturer of the University of Chinese Academy of Sciences. His research interest is mobile security.

He Yuan, born in 1977. PhD candidate in the School of Computer and Control Engineering, University of Chinese Academy of Sciences. His research interest is mobile security.

Li Xiaoqi, born in 1989. Master candidate in the University of Chinese Academy of Sciences. His main research interests include network and information security.

Yang Gang, born in 1991. Master candidate in the University of Chinese Academy of Sciences His main research interests include network and information security.