移动应用安全：回顾与展望

周亚金 任 奎
浙江大学

关键词：移动安全 移动应用

随着移动操作系统的兴起和移动设备的快速普及，针对移动设备的攻击也随之出现。流氓应用窃取用户隐私，恶意应用则造成用户财产损失，盗版游戏和盗版金融类应用不但侵犯开发者知识产权还威胁用户财产安全，系统级高危漏洞频频被曝光，攻击者利用漏洞安装后门，推送恶意应用和广告等等。这些都给用户造成了极大困扰，严重阻碍了移动设备的生态安全。

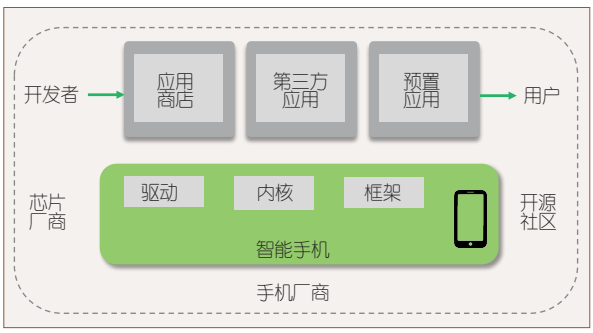

图1 移动应用生态系统示意图

学术界和工业界都在尝试解决这些问题。研究人员发现，应用市场的审核机制不严格是导致恶意应用快速传播并感染用户的途径。和传统的桌面系统的病毒传播方式不同，由于用户获取应用的集中性导致恶意应用传播的速度和影响范围都超过了桌面系统。在恶意应用大规模传播之前，如何进行事先的检测和预警成为研究问题之一。学术界对移动设备安全性的研究也促进了工业界对安全的重视。谷歌每月发布安全公告，推送安全补丁。以华为为代表的国内设备厂商也开通了用户和安全人员的沟通渠道，完善了安全预警机制。2013 年开始，安卓(Android) 设备上的内核代码段是可读可写的，如今 PXN 支持已经成为标配，PAN 和 Pointer Authentication 等新兴的硬件安全技术也正快速发展中。

移动应用生态

移动应用生态链上的各个参与者彼此依赖、相互促进，推动了生态的整体发展。图 1 描述了生态链上的各个参与方。

开发者和用户 应用和应用商店是整个生态中最为关键的一环，它是连接用户和开发者的纽带。用户从应用商店中获取应用，开发者根据用户的需求开发应用。应用商店完成应用的分发、新版本的升级等功能。然而，由于应用商店的审核不严格，恶意应用通过应用商店广泛传播。开发者收集用户数据，应用内的广告库追踪用户。如果开发者缺乏安全知识，就会导致应用存在漏洞，这些都会给用户的隐私和安全带来威胁。由于移动应用的多样性和复杂性以及应用场景的差异性，如何检测这些问题并提出解决方案，一直是移动安全领域研究人员的研究热点。

手机厂商、开源社区 Android[1] 框架为应用提供了运行时环境，它由开源社区的软件和谷歌贡

[1] 本文以 Android 系统为例，介绍移动应用安全。

献的代码构成。预置应用是手机厂商为了差异化的需要预先安装在设备中的，通常很难卸载或者不可卸载[2]。厂商为了丰富手机功能，也会对 Android 框架进行深度定制。

开源社区和芯片厂商 Android 系统构建在开源的 Linux 内核和芯片厂商的设备驱动程序之上。随着移动设备的流行，对内核和芯片驱动的安全性研究越来越深入。比如基于编译器的内核和驱动程序安全漏洞系统性挖掘，使用新的硬件安全特性（TrustZone, ARM Domain, PXN, PAN 等）提供更强的安全保证[1,2]。

移动应用安全研究

第三方应用隐私泄露和用户数据保护

由于 Android 系统在权限授予方面的局限性，应用容易滥用权限获取和泄露用户隐私数据。隐私数据泄露虽然可以通过污点分析来检测[3]，但污点分析的一个局限性在于难追踪隐式数据流。FlowDroid[4]是一种能检测隐式数据流的方法。研究者在这个方向上进行了后续研究，比如将污点分析应用到 ART 运行时的 TaintART[5]，追踪非结构化数据（用户输入）的 UIPicker[6]。

为了解决应用隐私泄露问题，需要对应用的行为进行细粒度和运行时的控制。TISSA[7]是第一个在此方向上进行尝试的系统。它提供了运行时的应用行为策略控制，将 Android 安装时的权限授予机制扩展到运行时，在不改变开发者开发习惯和用户使用习惯的前提下对应用行为进行细粒度的控制，保护用户的隐私。运行时行为控制的方法后来被国内手机厂商广泛采用，并且在 Android 6.0 版本中被默认支持。

TISSA 的不足在于需要对系统框架层进行修改，这在没有手机厂商支持的情况下很难被用户直接使用。因此，后续的研究主要集中在如何研发更易部署更易被用户采用的方案。Aurasium[8], Boxify[9] 和 Appcage[10] 从不同的角度提出了解决方法。Aurasium 通过对底层库的插桩完成应用行为控制，但它较难解决本地库访问隐私数据的问题。Appcage 通过应用沙盒和软件故障隔离 (Software Fault Isolation, SFI) 相结合的方法解决了这个问题。Boxify 则利用 Android 系统自身的特性——隔离进程 (isolated process)——来控制应用行为。随着 Android 系统对应用运行时行为控制的官方支持，在这一方向的研究会逐渐减少。不过这些研究提出的机制在第三方库（比如广告库）的行为控制中同样有用武之地。

恶意应用检测和分析

Android 的开放性促进了整个移动生态的快速发展，但与此同时，恶意应用也通过应用市场正在快速传播。如何对恶意应用进行快速检测是一个迫切需要解决的问题。DroidRanger[11] 和 RiskRanker[12] 是最早对恶意应用进行系统化检测的工作。DroidRanger 基于行为的签名来检测已知恶意应用的新变种，而 RiskRanker 则通过对应用进行风险评估来检测恶意应用新家族。基于这两个工作，作者通过 Android 恶意应用基因工程共享了恶意应用数据集[13]，成为后续研究人员评估新检测方法有效性的基础数据集。

由于应用市场中的应用数量非常庞大（超过数百万个），有一些工作从提高检测的效率入手，比如在应用代码方法级别进行差分扫描 (MassVet)[14] 就是一个很有意义的探索。通过应用描述的语义[15]来评估应用的风险也是比较巧妙的检测方法。

检测到恶意应用后，如何分析其行为是一个关键问题。DroidScope[16] 基于模拟器来分析恶意应用，难点在于如何消除应用和模拟器之间的语义差。Malton[17] 尝试从多个层次观察应用行为，从而避免语义差的问题，同时它提出有效的路径探索机制提高动态分析的代码覆盖率。不过，如何自动地触发恶意

[2] 预置软件不可卸载的情况正在改变。根据工信部 2015 年底出台的“移动智能终端应用软件 (APP) 预置和分发管理暂行规定”，预置应用必须可以自由卸载。不过这一规定的执行情况还需要进一步观察。

行为，提高分析效率仍然是一个待解决的研究问题。

盗版应用和应用保护

Android 应用容易被逆向分析和重新打包，重打包应用（或者盗版应用）也是恶意应用传播的重要途径之一。我们可以利用代码和资源相似度来检测盗版应用。DroidMOSS [18] 抽取应用的代码特征，采用模糊哈希的方法来检测相似度。此方法准确度高，但计算复杂度大。ResDroid [19] 则利用资源快速比较应用的相似度。FSquaDRA2 [20] 系统地研究了采用不同的资源做相似度检测的效果。

从防御的角度来看，既然应用容易被重打包，那么是否可以采用一些自我保护的机制，提高应用被重打包的难度呢？ DIVILAR [21] 引入一个新的中间语言在运行时检测应用的状态。除此之外，还可以通过将安卓代码转化到原生代码的方式 [22] 提高重打包难度。即便如此，攻击者还是可以通过自动化的方式来逆向被保护的代码 [23, 24] 。如何保护应用不被重打包和如何自动逆向重打包应用一直是一个你追我赶的过程。

应用开发不安全实践

开发者由于缺乏安全知识或者漠视安全性，开发的应用往往存在一些安全隐患，包括不正确地使用 Content Provider 组件导致用户敏感数据（如社交应用的聊天记录）被泄露 [25]，远程服务器 SSL 证书验证逻辑错误 [26]，不正确地使用加密算法 [27] 等。

除了客户端逻辑之外，应用还依赖于云端提供的服务。应用和云端服务的交互引入了新的安全威胁。比如将服务器登陆令牌硬编码在应用中 [28]，云服务对于应用登陆验证逻辑设计有缺陷，用户的登陆过程容易被暴力破解 [29] 等。这一系列问题说明了提高开发者的安全实践能力，将应用的潜在安全问题消除在源头的重要性。一个可行的解决方案是在开发端提供自动化的检测和修复工具。

应用内广告库

在应用内嵌入广告是开发者获得收益的主要方式。但嵌入的第三方广告库会在用户和开发者不知情的情况下侵犯用户隐私，追踪用户，加载不安全的代码 [30]。广告内容也会误导用户，甚至欺骗用户下载恶意软件 [31]。

Android 平台没有提供对第三方库和应用进行隔离的机制，因此研究人员对 Android 系统进行了扩展，对应用本身和应用内的广告库赋予不同的权限来限制广告的行为。后续的工作进一步限制了广告和用户的交互 [32]。最新的进展是基于编译器来实现应用和第三方库（包括广告库）的隔离 [33]，此方法与之前的方法相比，对 Android 框架修改较少并且更容易集成。目前研究的局限性在于没有把开发者引入到解决问题的框架中来。我们认为在开发工具端提供插件，在开发流程中自动完成应用和第三方库的隔离是更有效的解决方法。

预置应用的权限泄露

权限泄露是指具有某一种权限的应用没有对接口进行保护，使得没有权限的应用可以通过这个接口来获取执行特殊操作的能力。权限泄露虽然在第三方应用和预置应用中都存在，但是预置应用通常具有执行高敏感操作的权限（比如后台安装应用），因此权限泄露问题在预置应用中引起的后果更为严重。

Woodpecker 是第一个在预置应用中检测权限泄露的系统 [34]，它通过静态分析应用的接口和敏感 API 之间的可达性来检测权限泄露。检测路径可达性是比较成熟的程序分析方法。通过对流行的 Android 设备的分析，Woodpecker 发现了 11 种可能被泄露的权限，后续的工作则证实厂商的定制是引起权限泄露的原因之一 [35]。

新应用场景下的安全问题

移动支付等新场景的出现带来了新的安全挑战。

移动 POS 系统。移动 POS 系统是能在智能手机上进行支付的设备，它利用外接的读卡设备（通常通过耳机接口）读取银行卡数据，然后通过手机

上的应用完成支付。这些设备在安全性的设计上是有欠缺的，比如运行在手机上的应用可以获取保护银行卡信息的加密密钥，甚至可以对读卡设备的固件进行任意替换，插入恶意代码，进行攻击[36]。

应用内支付。开发者往往使用应用内支付功能来完成应用内商品的出售。而由于第三方支付服务设计的不当、开发者缺乏安全意识等原因，导致应用内支付并不安全[37]。通过对国内常用的应用内支付的研究发现，10%以上的应用存在安全问题[38]。

移动在线支付。支付应用通常会采用令牌来验证支付交易的有效性。但是攻击者可以嗅探支付令牌，中止当前支付服务，然后将嗅探的令牌用在其他交易中（比如购买价值更大的商品）。这种攻击方法常见于三星和阿里支付[39]。

移动安全研究展望

移动安全正处在一个新旧问题交替的阶段，出现了一些新的研究方向。

新的认证方法。移动设备目前都采用了基于生物特征的认证方法，比如指纹、虹膜以及人脸等识别方法。这些认证方法提高了用户使用的便利性，但是其在安全性方面的评估还需要进一步深入。由于生物特征的唯一性以及不可替换性，生物特征信息被泄露所引起的后果远比密码泄露更为严重。因此对基于生物特征的认证体系进行全面的安全评估是将来可能的研究方向。

新的交互场景。物联网设备可以通过移动应用来进行远程控制。这涉及到物联网设备和云服务器的交互，云服务器和智能手机的交互。过去的研究经验告诉我们，多个实体之间的交互是最容易出现安全问题的。由于物联网设备资源受限的特点，系统通常没有安全隔离机制，而设备暴露在联网环境下又增大了攻击界面。在智能手机、云服务、物联网设备交互的场景下，新安全问题的探索和解决方案也是将来的研究方向。

新的安全架构。目前的Android系统主要应用在消费产品市场，电子政务、移动办公等场景的出现，对系统的安全性提出了更高的要求。基于硬件安全特性，软硬件一体化的协同安全系统架构是值得探索的研究方向。 ■

周亚金

浙江大学网络空间安全研究中心研究员。主要研究方向为移动系统安全。
yajin_zhou@zju.edu.cn

任　奎

CCF专业会员。浙江大学网络空间安全研究中心主任，国家千人计划特聘教授。主要研究方向为物联网安全、云安全和隐私保护。 kuiren@zju.edu.cn

参考文献

[1] Zhou Y, Wang X, Chen Y, et al. ARMlock: Hardware-based Fault Isolation for ARM[C]// *ACM Sigsac Conference on Computer and Communications Security*. ACM, 2014:558-569.

[2] Azab A M, Shah J, Shah J, et al. Hypervision Across Worlds: Real-time Kernel Protection from the ARM TrustZone Secure World[C]// *ACM Sigsac Conference on Computer and Communications Security*. ACM, 2014:90-102.

[3] Enck W, Gilbert P, Chun B G, et al. TaintDroid: an information flow tracking system for real-time privacy monitoring on smartphones[J]. *ACM Transactions on Computer Systems*, 2014, 32(2):1-29.

[4] Arzt S, Rasthofer S, Fritz C, et al. FlowDroid:precise context, flow, field, object-sensitive and lifecycle-aware taint analysis for Android apps[J]. *ACM SIGPLAN Notices*, 2014, 49(6):259-269.

[5] Sun M, Wei T, Lui J C S. TaintART: A Practical Multi-level Information-Flow Tracking System for Android RunTime[C]// *ACM Sigsac Conference on Computer and Communications Security*. ACM, 2016:331-342.

[6] Nan Y, Yang M, Yang Z, et al. UIPicker: user-input privacy identification in mobile applications[C]// *Usenix Conference on Security Symposium*. USENIX Association, 2015:993-1008.

[7] Zhou Y, Zhang X, Jiang X, et al. Taming Information-Stealing Smartphone Applications (on Android)[M]// Trust and Trustworthy Computing. Springer Berlin Heidelberg, 2011:93-107.

[8] Xu R, Anderson R. Aurasium: practical policy enforcement for Android applications[C]// Usenix Conference on Security Symposium. USENIX Association, 2012:27-27.

[9] Backes M, Bugiel S, Hammer C, et al. Boxify: full-fledged app sandboxing for stock android[C]// Usenix Conference on Security Symposium. USENIX Association, 2015:691-706.

[10]Zhou Y, Patel K, Wu L, et al. Hybrid User-level Sandboxing of Third-party Android Apps[C]// ACM Symposium on Information, Computer and Communications Security. ACM, 2015:19-30.

[11]Zhou Y, Wang Z, Zhou W, et al. Hey, You, Get Off of My Market: Detecting Malicious Apps in Official and Alternative Android Markets[J]. Proceedings of Annual Network & Distributed System Security Symposium, 2014.

[12]Grace, M.C., Zhou, Y., Zhang, Q., Zou, S., Jiang, X.: Riskranker: Scalable and accurate zero-day android malware detection. In: MobiSys (2012).

[13]Y. Zhou 和 X. Jiang, “Dissecting Android Malware: Characterization and Evolution”.

[14]K. Chen, P. Wang, Y. Lee, X. Wang, N. Zhang, H. Huang, W. Zou 和 P. Liu, “Finding Unknown Malice in 10 Seconds: Mass Vetting for New Threats at the Google-Play Scale”.

[15]R. Pandita, X. Xiao, W. Yang, W. Enck 和 T. Xie, “WHYPER: Towards Automating Risk Assessment of Mobile Applications”.

[16]L.-K. Yan 和 H. Yin, “DroidScope: Seamlessly Reconstructing the OS and Dalvik Semantic Views for Dynamic Android Malware Analysis”.

[17]L. Xue, Y. Zhou, T. Chen, X. Luo 和 G. Gu, “Malton: Towards On-Device Non-Invasive Mobile Malware Analysis for ART”.

[18]W. Zhou, Y. Zhou, X. Jiang 和 P. Ning, “DroidMOSS: Detecting Repackaged Smartphone Applications in Third-Party Android Marketplaces”.

[19]Y. Shao, X. Luo, C. Qian, P. Zhu 和 L. Zhang, “Towards a Scalable Resource-driven Approach for Detecting Repackaged Android Applications”.

[20]O. Gadyatskaya, A.-L. Lezza 和 Y. Zhauniarovich, “Evaluation of Resource-based App Repackaging Detection in Android”.

[21]W. Zhou, Z. Wang, Y. Zhou 和 X. Jiang, “DIVILAR: Diversifying Intermediate Language for Anti-Repackaging on Android Platform”.

[22] 刘惠明 和 诸葛建伟 , “安卓应用自动原生化及混淆系统”.

[23]W. Yang, Y. Zhang, J. Li, J. Shu, B. Li, W. Hu 和 D. Gu, “Appspear: Bytecode decrypting and dex reassembling for packed android malware”.

[24]L. Xue, X. Luo, L. Yu, S. Wang 和 D. Wu, 出处 Adaptive Unpacking of Android Apps.

[25]Y. Zhou 和 X. Jiang, “Detecting Passive Content Leaks and Pollution in Android Applications”.

[26]M. Georgiev, S. Iyengar, S. Jana, R. Anubhai, D. Boneh 和 V. Shmatikov, “The most dangerous code in the world: validating SSL certificates in non-browser software”.

[27]M. Egele, D. Brumley, Y. Fratantonio 和 C. Kruegel, “An empirical study of cryptographic misuse in android applications”.

[28]Y. Zhou, L. Wu, Z. Wang 和 X. Jiang, “Harvesting Developer Credentials in Android Apps”.

[29]C. Zuo, W. Wang, R. Wang 和 Z. Lin, “Automatic Forgery of Cryptographically Consistent Messages to Identify Security Vulnerabilities in Mobile Services”.

[30]M. Grace, W. Zhou, X. Jiang 和 A.-R. Sadeghi, “Unsafe exposure analysis of mobile in-app advertisements”.

[31]V. Rastogi, R. Shao, Y. Chen, X. Pan, S. Zou 和 R. Riley, “Are These Ads Safe: Detecting Hidden Attacks through the Mobile App-Web Interfaces”.

[32]X. Zhang, A. Ahlawat 和 W. Du, “AFrame: Isolating Advertisements from Mobile Applications in Android”.

[33]J. Huang, O. Schranz, S. Bugiel 和 M. Backes, “The ART of App Compartmentalization: Compiler-based Library Privilege Separation on Stock Android”.

[34]M. Grace, Y. Zhou, Z. Wang 和 X. Jiang, “Systematic Detection of Capability Leaks in Stock Android Smartphones”.

[35]L. Wu, Y. Zhou, M. Grace, X. Jiang 和 S. Zou , “The Impact of Vendor Customizations on Android Security”.

[36]W. Frisby, B. Moench, B. Recht 和 T. Ristenpart , “Security analysis of smartphone point-of-sale systems”.

[37]D. Reynaud, E. C. R. Shin, T. Magrino, E. Wu 和 D. Song, “FreeMarket: Shopping for free in Android applications”.

[38]W. Yang, Y. Zhang, J. Li, H. Liu, Q. Wang, Y. Zhang 和 D. Gu, “Show Me the Money! Finding Flawed

Implementations of Third-party In-app Payment in Android Apps.”.

[39]X. Bai, Z. Zhou, X. Wang, Z. Li, X. Mi, N. Zhang, T. Li, S.-M. Hu 和 K. Zhang, “Picking Up My Tab: Understanding and Mitigating Synchronized Token Lifting and Spending in Mobile Payment”.