doi:10.11959/j.issn.1000-436x.2016127

Android 智能终端安全综述

许艳萍[1]，马兆丰[1]，王中华[2]，钮心忻[1]，杨义先[1]
（1. 北京邮电大学信息安全中心，北京 100876；2. 国家计算机网络应急技术处理协调中心，北京 100029）

摘 要：针对 Android 智能终端安全问题，构建 Android 智能终端安全分层体系。首先从远程防盗、生物身份验证和硬件安全模块方面阐述了 Android 设备安全的安全威胁及保护措施，然后从无线安全网络、病毒传播查杀和防钓鱼攻击说明了 Android 网络安全的隐患及防范，之后从内核安全、本地库运行时环境安全和应用框架安全角度介绍了 Android 操作系统安全的研究内容，接着从静态检测和应用行为动态检测、应用加固和应用评估方面展示了 Android 应用安全的研究成果，接下来着眼于数据本身总结了 Android 数据的追踪、加密和备份等安全保护技术，最后结合实际需求展望了 Android 安全未来在安全增强框架、智能应用行为分析等方向的发展。
关键词：Android；设备安全；网络安全；系统安全；应用安全；数据安全
中图分类号：TP309 **文献标识码**：A

Survey of security for Android smart terminal

XU Yan-ping[1], MA Zhao-feng[1], WANG Zhong-hua[2], NIU Xin-xin[1], YANG Yi-xian[1]
(1.Information Security Center, Beijing University of Posts and Telecommunications, Beijing 100876, China;
2. National Computer Network Emergency Response Technical Team/Coordination Center of China (CNCERT/CC), Beijing 100029, China)

Abstract: Aiming at the security, the layered security system was constructed. Firstly, the devices safety protection based on remote anti-theft, biometric identity verification and hardware security module was expounded. Secondly, network security referring to the wireless security network, virus propagation killing and anti-phishing was illustrated. Thirdly, the OS safety was introduced from the perspective of system kernel, runtime environment and application framework. Fourthly, application security was showed containing the reverse engineering static analysis, behavior dynamic analysis, safety reinforcement and safety assessment. Fifthly, the privacy data protection was summarized including tracking, encryption and backup. Finally, the future development direction was prospected on the security framework and intelligent behavior analysis.
Key words: Android, device security, network security, system security, application security, data security

1 引言

Android 系统是 2007 年 Google 公司发布的基于 Linux 内核的开源操作系统，主要应用于智能手机、平板 Pad、智能手表等智能终端。自发布以来，系统的开源特性广受智能设备厂商及开发人员的推崇，广泛应用于金融、政府、交通、教育、军事、汽车、家居、能源等重要领域。IDC 调研显示，2015 年 Android 智能手机的出货量达到 14.3 亿部，占全球智能手机市场份额的 82%。

360 互联网安全中心发布《2015 年手机安全状况报告》[1]显示，2015 全年截获的 Android 平台新增恶意程序样本为 1 874 万个，较 2013 年，2014 年分别增加了 27.9 倍和 5.7 倍，累计约 3.7 亿人次 Android 用户感染到恶意程。根据危害结果将 Android 病毒分 5 大类：移动支付(60%)、资费消耗(14%)、隐私窃取(20%)、远程控制(2%)、黑客工具(4%)[2]。Android 病毒最常做出的前 3 位恶意行为是资费泄露隐私、权限

收稿日期：2015-10-19；**修回日期**：2016-02-19
基金项目：国家自然科学基金资助项目（No.61272519）；“十二五”国家科技支撑计划基金资助项目（No.2012BAH23F00）；国家科技支撑计划基金资助项目（No.2012BAH45B00）
Foundation Items: The National Natural Science Foundation of China (No.61272519), The Twelfth-five National Science and Technology Support Program (No.2012BAH23F00), The National Science and Technology Support Program (No.2012BAH45B00)

提升、流量消耗[3,4]，智能终端安全正面临着严峻的考验。

由于 Android 平台开放的生态，每一个 Android 设备中安装的软件可能来自于 Google、手机开发商、芯片厂商和不同应用开发者等，软件组件很难遵从统一的开发、测试、审计标准，因而质量参差不齐。对恶意攻击者和安全研究者来说，却是一个很好的研究目标。

参照信息安全技术体系设计原则[5]，将 Android 智能终端安全层次划分为设备安全层、网络安全层、系统安全层、应用安全层和数据安全层，每层针对受到的威胁采取相应的安全防护措施[6~9]，从而达到 5 个安全目标：保密性、完整性、可用性、可控性和不可否认性。Android 智能终端安全分层体系如图 1 所示。

设备安全层。Android 终端设备是应用和数据的载体，主要研究设备丢失情况下的远程防盗、基于生物特征的身份验证和硬件安全模块内容。Android 设备中存储了大量的个人隐私数据，如短信、通讯录、照片等[3]，如果在设备丢失或者绕过身份验证被非法人员操作的情况下，隐私数据很容易被窃取。基于 XMPP 的协议远程对终端设备下发指令控制设备[10]；基于指纹、人脸、虹膜等人体生物特征的身份验证拒绝非法终端用户登录，打造安全的设备环境；采用硬件安全模块，基于硬件实现高性能安全加密和身份验证等功能。

网络安全层。Android 智能终端互联互通，形成无线通信网络，主要研究无线安全网络、病毒传播查杀和防钓鱼攻击等。Android 智能设备通过采用安全无线路由器、加密 VPN 服务[11]等方式，构建安全网络接入环境，保障数据安全传输；Android 病毒以短信、彩信、二维码、伪装应用安装等方式传播扩散，通过特征识别，利用云计算、大数据技术智能化地检测病毒是阻断病毒传播扩散的有效方式[12,13]；钓鱼攻击是移动互联网面临的重大威胁，检测方式包括黑名单、异常检测和机器学习智能检测等。

系统安全层。针对 Android 智能终端操作系统，按照系统层次研究内核安全、本地库和运行时环境安全、应用程序框架安全。通过内核增强[14]、漏洞挖掘[15]、内存防泄漏、系统防 ROOT 方式保护内核安全；通过安全沙箱机制、ART 机制、组件间安全通信保障本地库和运行时环境安全；通过权限机制和签名机制保障应用程序框架安全。

应用安全层。针对运行在 Android 智能终端上的应用安全，主要研究恶意应用静态检测、恶意应用动态检测、应用安全加固和应用安全评估。静态检测研究应用逆向工程、特征提取、机器学习智能检测、重打包检测等内容；动态检测主要研究动态行为获取；应用安全加固可采用代码混淆、应用加壳、防反编译[16]等方式。通过建立评估体系和评估标准，评估恶意应用给系统和数据带来的风险[4,17]。

数据安全层。保护 Android 终端设备存储上的数据安全，研究数据伪造及跟踪[18,19]、加密[20]、云备份、数字版权保护、安全支付等内容。隐私数据是 Android 终端设备上最具有核心价值的部分，保护设备、网络、系统、应用的安全，终极目的是保护数据的机密性、完整性和可用性，通过伪造及跟踪、加密、云备份、数字版权保护、安全支付等保护方式，直接保护隐私数据的安全性。

图 1 Android 智能终端安全分层体系

鉴于 Android 广泛的市场应用，学术界的专家学者对其安全机制进行了诸多研究，主要围绕权限增强管理、应用安全分析、隐私数据防泄露等方面，并取得显著成果。本文在广泛阅读文献的基础上，总结了 Android 平台在设备安全、网络安全、系统安全、应用安全和数据安全方面的研究成果，并根据当前研究工作的不足展望了几点未来的研究方向。

2　Android 移动终端设备安全

2.1　远程防盗

实现 Android 智能终端远程控制的方式有基于 HTTP 协议的远程数据传输、基于 RDP 协议的远程桌面控制[21]、基于 XMPP 的即时消息推送[10]等。

基于 XMPP 协议的远程防盗是当前比较流行的终端设备保护方式，XMPP 协议具有超强的可扩展特性，服务端和智能终端可约定通信指令[22]。应用时，首先将用户信息和设备信息注册到控制端，制定保护策略，如数据备份和删除、远程锁定、远程定位等。一旦终端设备丢失，通过控制端下发指令对终端设备定位、对屏幕远程锁定、更换 SIM 卡锁定设备、删除短信和通讯录等隐私数据等，减少因设备丢失造成的损失。目前，主流的智能终端厂商都提供对终端远程防盗的服务。

远程防盗已经成为终端安全的标配功能之一，其优点是即使终端不在身边，只要确定身份合法也能够方便地实施操作，尤其是在设备丢失的情况下，远程定位、远程删除和锁定对于找回手机、保护隐私有重大意义，但对于网络的要求很高，一旦设备离线，便不能发送控制指令，而通过短信发送指令是另一种有效的途径。

2.2　生物身份验证

人体的生物特征往往具有唯一性，如指纹、掌纹、声音、虹膜、人脸、步态、脉搏、运动特征等，科技界已经逐渐将这些特征作为身份识别和验证的重要手段[23]。Android SDK 从 1.0 版本中就已经集成了简单的人脸识别功能，而 OpenCV 开发包集成了更专业、快速、高级的人脸识别算法。自 2013 年 7 月芬兰 Uniqul 公司首次推出基于脸部识别系统的支付平台，目前互联网金融企业正在研究将人脸支付运用于 Android 支付系统。

目前有很多 Android 终端设备搭载了指纹识别技术，利用指纹识别传感器及指纹模式支持指纹锁，作为安全工具、硬件平台和操作系统三方支持的安全模式，不仅简便了解锁的过程，还把非法用户阻挡在外。但是这种防护手段已被黑客攻陷，他们发现了指纹识别框架下的漏洞，轻松地绕过指纹识别器，从而执行任何操作，而且还能够直接拷贝用户的指纹信息，在更大范围内威胁用户的个人信息安全和隐私内容。在此情况下催生出了更高级、更安全的身份验证方式——眼球识别技术。目前这种技术的安全性还没被突破，眼球样本信息不能被复制，但在识别范围和识别准确率方面还需要进一步改进。另外，Google 正在测试利用用户的动作震动或者任意活动手机，将动作幅度或者频率作为锁屏和解锁的方式，从进展来看，这是项极具挑战性的工作。

从发展趋势上看，利用人体生物特征作为唯一的识别标识，是 Android 智能终端领域经久不衰的应用研究，在识别校准和准确率方面还需要提高，而其性能在大数据方面也在不断探索。

2.3　硬件安全模块

出于对移动终端中隐私数据提供集中的安全保护考虑，智能终端厂商设计并实施了硬件安全模块（HSM, hardware security modules），直接连接到系统总线上，基于硬件实现高性能密码和密钥管理功能，提供的服务包括安全存储、安全加密、身份验证等。嵌入式安全技术供应商 Discretix 公司研发的 CryptoCell 安全模块具备 PKI 引擎、对称引擎、散列引擎、随机数发生器、安全密钥和完整性验证等 6 个可配置的功能模块，加速执行各种验证和加密功能，对整个智能终端设备的发展将产生积极的影响。在国内，华为作为手机终端的领跑者，通过硬件安全模块与麒麟操作系统协作，彻底杜绝刷机解密等操作的可能性。

硬件安全模块在保证数据保护、安全支付等方面更专业，效果更明显，随着智能终端对安全愈加重视，必将成为重要的保护方式。

3　Android 移动终端网络安全

3.1　无线安全网络

Android 智能终端通过 GPRS、无线 Wi-Fi 接入网络可满足用户随时随地上网的需求。无论使用哪种方式，开放的网络环境中，用户执行登录账号、支付等敏感操作时，易遭到黑客攻击导致隐私泄露，常用的攻击方式有嗅探、钓鱼攻击、中间人攻击、恶意 DNS 篡改、远程执行、XSS 跨站脚本、

电磁信号干扰等[24]。

Ma 等[25]针对商业 Wi-Fi 网络提出混合式的恶意网络接入点检测方案，主要采用无线扫描和集中流量监控方式发现恶意 AP，并且保护网络免受恶意 AP 的侵害，优点是不需要专门的硬件并且适用于任何网络标准。

无线路由器通常采用 WPA/WPA2 加密的方式保障密码的安全性，但是通过多次逆运算之后，这种加密能够被破解，需要探索另外的方式保证无线口令的安全性。现在的手机安全防护软件也集成了安全监测功能，防范 DNS、ARP、虚假 Wi-Fi、加密及钓鱼等攻击。

Wang 等[26]针对 Wi-Fi 驱动的 Fuzzing 测试框架，通过构建数据分组实施针对 Wi-Fi 通信的攻击，这种框架不仅能发现已知攻击，在发现未知攻击方面也效果显著。

针对 Wi-Fi 窃听问题，可采用加密的 VPN 服务，将所有传输流量都进行加密，无论通信链路是否安全，都能保证数据的安全[11]。

3.2 病毒传播查杀

自 2010 年来，每年都爆发多次手机病毒危机，在移动互联网络上迅速传播，波及范围广泛，十分具有破坏性。每个季度网络安全公司都会发布 Android 手机安全报告，通报当前手机病毒及手机感染病毒的情况等。将 Android 病毒按照危害特征划分为多个家族：AnserverBot、BaseBridge、DroidKungFu、DroidDream、FakePlayer、Geinimi、GoldDream、GingerMaster、Zsone、SndApps 等[3,27]。Android 病毒按照危害行为划分为木马类、后门类、蠕虫类、僵尸网络类、间谍窃取类、欺骗类等[3]。手机病毒利用通讯录群发短信、彩信、电子邮件和扫描验证码、二维码方式扩散传播，造成隐私数据泄露、资费消耗、破坏系统等后果。

病毒具有逐利性，发展技术和种类总是层出不穷，但是反病毒技术的发展只能仅仅追逐于病毒更新的速度，往往反病毒技术只能应对已知的病毒，对于未知的病毒，专家正着眼于利用云计算、大数据资源基于机器学习、深度学习算法等方式，自动化、智能化地根据行为特征对其进行查杀，并按照其执行的操作将其划分到不同的种类[27,28]。

3.3 防钓鱼攻击

手机终端绑定银行、支付和社交等账户，容易遭受不法分子的觊觎，他们往往通过网页钓鱼、短信钓鱼、应用钓鱼、邮件钓鱼等诱惑用户[29]，2015 全年 360 手机安全防护软件共拦截各类钓鱼网站攻击 48 亿次[1]。

钓鱼网站攻击检测和防范成为当务之急，常用的方法有黑名单、异常检测和机器学习智能检测等。Bottazzi 等[30]提出钓鱼检测框架 MP-Shield，分析 URL 链接，集成了基于公共黑名单的浏览器检测插件，同时实现了机器学习检测引擎，包含 J48、SMO、BayesNet、IBK 和 SGD 算法，确保手机终端免受 0Day 钓鱼攻击。

Wu 等[31]提出轻量级检测工具 MobiFish，检测手机 Web 网页、应用和登录账户上遭遇的钓鱼攻击。MobiFish 应用 OCR 技术提取截屏中的文本信息，通过文本中声称的身份信息与网页 URL 和 APP 中实际的身份信息比较，给出钓鱼警示，而不像其他方法那样，分析 HTML 源码、IP 地址、搜索引擎[32]，或者采用机器学习算法[33]证明该方法能有效检测和防范钓鱼攻击。

受到利益的驱使，钓鱼网站层出不穷，广大终端用户应当使用主流的安全防护软件，它们基本涵盖了已知的钓鱼网站黑名单，能够很好地拦截非法钓鱼链接。同时，用户应提高警惕，不受诱惑，不点击未知链接，并且不泄露身份信息和财产账户信息等。

4 Android 系统安全

4.1 Android 系统内核安全

4.1.1 内核增强

Android 系统内核的安全威胁一方面继承自 Linux 系统漏洞，另一方面源于 Android 平台本身不完善。在 Android 系统内核访问控制增强研究方面，Shabtai 等[34]提出将强制访问控制子系统 SELinux 植入到 Android 平台中，以限制应用对系统资源的访问，增强 Android 系统安全，是首次提出的高效低耗的系统增强方案。Smalley 等[35]实现了基于 SELinux 内核开发一套 MAC 中间件扩展 Android 权限模型，隔离应用间的数据交互。

实际应用上，从 Android 4.3 以来，正式引入基于 MAC 的 SELinux 安全机制，称为 SEAndroid，用来强化操作系统对应用的访问控制，起到类似沙箱的执行隔离效果，阻止恶意应用对系统或其他应用的攻击，还有效地减弱内核层出现漏洞时产生的

威胁。

4.1.2 漏洞挖掘

任何系统在逻辑设计和实现上都存在缺陷或者错误，Android 系统亦不例外，2015 年，Google 成立了 Android Security Rewards 项目，用于奖励 Android 系统漏洞发现者。历年共公布 600 多条系统漏洞，覆盖到内核层、Native 层、框架层及应用层，包括内存溢出、权限提升、组件交互、硬件资源调配等各个方面[36]。

传统的漏洞挖掘方式分为主动方式和被动方式，主动挖掘方式分为人工挖掘、静态挖掘和动态挖掘，被动挖掘方式分为攻击分析和补丁分析[37]。Android 平台的漏洞主要分为常规漏洞和权限提升漏洞，通过分析逻辑规则的合理性，编写 Fuzz 测试用例，构建漏洞知识库是一种有效的漏洞发掘方式。

Zhang 等[37]提出 Android 平台漏洞挖掘 4 层模型，针对攻击样本、攻击方法、修复补丁，分析漏洞存在的可能性，利用 Fuzz 测试方法和攻击挖掘方法，有效地挖掘出系统漏洞、应用漏洞、权限提升漏洞等。

比较严重的 Android 系统漏洞执行方便、涉及范围广、无需主动触发、隐蔽性强，一旦被黑客利用，造成的损失不可估量。对于 Android 系统本身来说，漏洞挖掘是个持续的过程，漏洞挖掘得越彻底，系统将修补得越完善。

4.1.3 内存防泄漏

Android 系统每个版本中都存在内存泄漏问题，即应用进程中一些对象没有使用价值了，但还占据着内存空间，拒绝被 GC 回收，导致实际可使用内存变小，系统运行越来越慢，用户体验越来越差。内存泄漏产生的原因有类的静态变量持有大数据对象、资源对象使用后未关闭、注册对象未反注册、集合中不用的对象没清理、Handler 临时性等[38]。使用 DDMS、MAT 工具可查看内存使用情况。

Shahriar 等[39]在研究资源对象导致内存泄漏的情况时，编写 Fuzz 测试用例，研究应用 bitmap and imageview、event listener、animation activity、static object、AdView object 等对象时是否产生内存泄漏。而针对 context、内部类、数据库操作等其他行为是防内存泄漏需要进行的进一步深入研究。

4.1.4 系统防 ROOT

Android 系统刷机是将刷机包中的系统文件写入 ROM 存储区，替换原生系统文件，再次启动时，从 ROM 中载入最新替换过的系统文件。本质就是系统文件的覆盖和替换操作，会破坏原有系统的权限机制，而来源各异的刷机包中可能集成了大量的恶意软件或病毒，留下隐患，因而刷机要谨慎[40]。

另一种破坏终端安全机制的方式是对终端 ROOT，利用操作系统漏洞获取最高的 ROOT 管理权限，任意访问和修改所有的文件。这种方式的优点是用户可以完全控制终端，缺点是黑客也能够利用病毒轻而易举获得 ROOT 权限，导致终端完全暴露出来，可能产生严重后果。Android 5.0 版本采用增强 SELinux 后，常用的 ROOT 方案已行不通，必须通过刷内核的方式，这需要解锁 bootloader，而一旦锁定 bootloader，能够实现防 ROOT 破解。

4.2 本地库和运行时环境安全

4.2.1 安全沙箱机制

Android 4.4 以前的运行时环境为应用程序的正常运行提供了核心链接库和 Dalvik 虚拟环境，采用沙箱机制隔离各应用程序之间的运行环境。每个应用作为一个 Dalvik 虚拟机实例独立地运行在一个进程内，Android 系统为每个应用分配一个 UID，保证应用的文件、数据相互独立存储。通过设置 sharedUserID，使多个应用程序具备相同的 UID，从而共享数据和权限。

这方面的安全问题主要集中在应用之间利用 IPC 突破沙箱隔离，导致权限提升、信息泄露、恶意攻击等。权限提升是一种很严重的威胁，使未申请权限的应用执行授权操作[41,42]，破坏了程序安全运行环境，因而对 Android 应用沙箱的安全防护一定程度上体现在对权限提升漏洞的检测和修复，对权限提升攻击的检测和防护将在 4.3.1 节进行详细介绍。

4.2.2 ART 机制

Android 5.0 运行时环境采用 ART 机制，只在应用安装时进行一次预编译，将程序语言转化为机器语言代码存储在本地，而 Dalvik 虚拟机机制在应用每次编译时都要编译，每次程序运行时都将程序语言转换为机器语言。因此 ART 机制使应用的启动和执行更加迅速，效率也更高，但是在应用安装时耗费更多时间预编译，运行时占有更多内存。如何优化 ART 机制产生的内存问题需要不断深入研究。

Demertzis 等[43]提出 SAME，在 ART 虚拟机中分

析应用程序的类，采用基于 BBO 的多层感知器，并行运行粒子群优化（PSO）、蚁群优化（ACO）和遗传算法（GA）等算法，以区分应用是否是恶意。

4.2.3 组件间安全通信

Android 应用由 Activity、Service、Broadcast Receiver 和 Content Provider 4 类组件组成，组件之间通过 Intent 进行交互，隐式 Intent 容易引发恶意调用、恶意发送广播、activity 劫持、恶意 service 启动等[44]问题。这方面的研究方案主要集中在基于上下文感知环境构建组件调用图检测数据泄露的路径。

Enck 等[45]指出组件间通信缺少对信息流的监控，存在信息泄露的风险。傅建明等[46]针对国内流行的应用程序研究采用静态分析方法构建其 CFG（控制流程图）、FCG （函数调用图）和 CCG（组件调用图）主要检测隐式 Intent 可能引发组件劫持、信息泄露和组件的权限泄露问题。

确保组件间安全通信可采取的措施有最小化组件暴露、设置组件访问权限、检查暴露组件的代码等。

4.3 应用程序框架安全

4.3.1 权限机制

1) 保护 ROOT 权限

ROOT 是 Android 平台中唯一的超级用户，管控所有系统资源，据 iiMedia Research 统计，近 30% 的用户对自己的手机终端 ROOT，利用获取的 ROOT 权限卸载预装的应用程序，再安装第三方软件管理权限、系统文件等，这无疑为病毒非法入侵打开了方便之门。

Android 4.4 之前，用户无法自主管理应用程序的使用权限，除非使用第三方软件，而 Android 4.4 之后，Android 原生系统集成了 SuperUser 权限管理程序，这对 Android 权限管理具有里程碑式的意义。

ROOT 权限对于系统资源的安全保护具有很重要的作用，轻易不要对终端设备 ROOT，而在未破解 ROOT 的情况下，如何更深层次地调配系统资源并保护系统安全稳定地运行是需要考虑的。

2) 应用程序权限增强管理

Android 4.3 以前的原生系统对权限的管理是粗粒度的，即应用在安装时，权限之间的关联关系没有明确告知用户，用户设置全接受或全否定应用申请的权限；在应用运行过程中对权限的使用零监控。Google 开发文档提供了详细的开发资料，却对权限的使用说明涉及较少[47]，这导致开发者对权限申请过度。权限的使用贯穿应用程序的全生命周期，在应用的开发、安装和应用过程中都占据重要地位，Android 权限机制实现如图 2 所示。

① 权限提升监测和防护

应用运行时，一些应用在自身没有申请权限的情况下，通过其他应用获取权限进而获得访问资源的资格，这种行为称为权限提升。Schlegel 等[48]提出一种手机木马 Soundcomber，本身不具有操作音频的权限，却通过其他应用的权限获取音频数据，这是利用 Android 系统隐蔽通道的典型例子。Davi 等[41]展示了在没有短信权限的情况下如何发送短信。Grace 等[49]提出权限提升检测工具 Woodpecker，检测出有些软件不遵守权限机制，将保护隐私数据的权限暴露给其他应用。

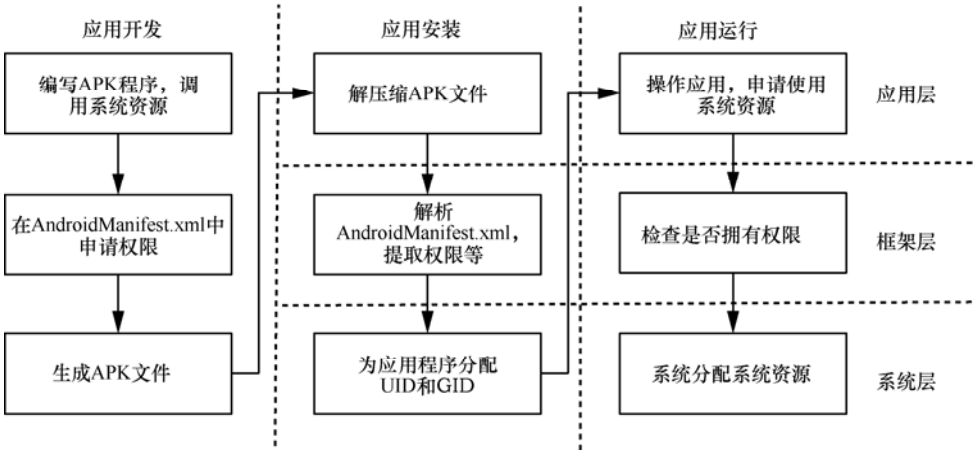

图 2 Android 权限机制

在权限提升防护方面，研究成果包括 Saint[50]、XMAndroid[51]、Quire[52]、IPC Inspection[47]、CHEX[53]、DroidAlarm[54]等。权限管理增强框架 Saint[51]，对应用 APK 文件增加权限和策略信息，安装时规定具体权限被授予给其他应用的情况，运行时检查组件间的通信，一旦应用之间权限验证失败，就无法交互。安全监控框架 XMAndroid[52]，利用 Android 系统的隐蔽通信通道，构建无向图，动态监视应用之间的权限使用情况，通过设置安全策略，阻断应用之间利用间接获得的权限执行未经授权的操作。各方案比较如表 1 所示。

表 1　　**权限提升监测和防护方案比较**

方案	方法	隐蔽通道	工作层
Saint	动态，安装	否	中间层
XMAndroid	动态，运行	是	中间层，内核层
Quire	动态，运行	—	中间层，内核层
IPC Inspection	动态，运行	—	中间层
CHEX	静态	否	—
DroidAlarm	静态	否	—

抗混淆副攻击系统 Quire[53]，跟踪 IPC 和 RPC 的调用链，每个应用向被调用者传播其调用链的上下文以验证权限是否符合，并验证接收到的数据是否可信，如果不符合则拒绝。防跨应用越权操作的 IPC Inspection 机制[47]，通过分析 IPC 传递事件信息和权限信息，将被调用应用的有效权限减少为调用与被调用应用权限的交集，防止应用发起混淆代理人攻击导致恶意入侵。同 Quire 一样，IPC Inspection 也无法检测所有的通过隐蔽通道传输的信息流。

组件漏洞检测方式 CHEX[54]，静态分析应用的数据流检测应用是否存在权限泄露、未授权数据访问、Intent 欺骗等问题，检测应用是否存在组件劫持漏洞，对 Android 系统开销低。

权限提升发现工具 DroidAlarm[55]，静态提取恶意应用的所有公开接口及声明敏感权限信息，采用数据流跟踪算法发现权限提升问题。

权限提升攻击检测的方式有多种，主要都是通过监控应用间的权限使用情况，发现越权非法访问资源的威胁。就静态检测方式而言，如果应用受加壳、混淆保护等，将无法有效检测；而动态检测能够实时检测应用进程间的通信，是比较有效的检测方式。

② 权限模型扩展和增强

对原生 Android 权限框架进行扩展，应用安装时，设置策略检测权限并修改权限申请，应用运行时，跟踪并操作权限使用情况，从各方面增强权限机制，从根本上改善权限管理。

2016127-7

Enck 等[55]提出轻量级应用检验工具 Kirin 安全规则，设置静态权限组合策略，应用安装时，对申请的权限进行检测，若违反策略则认定应用为恶意应用并拒绝安装，以防止隐私泄露。这在业界是首次提出的权限增强方案。

Nauman 等[56]提出权限策略扩展框架 Apex，在应用安装时，检测应用权限，允许用户有选择地授予权限，并且还能够自定义限制资源的使用情况。

Felt 等[57]提出权限静态检测工具 Stowaway，根据 API 与权限的映射关系，得到应用申请的最小权限集，并用其检查应用过度申请权限的情况。之后，有很多学者基于此方案进行了扩展和深入的研究。Geneiatakis 等[58]结合运行时的信息和静态分析的数据判断应用是否过度申请权限，是否遵从了最小权限集原则，这种方式不用修改应用的源码也不用修改 Android 系统的底层代码，最大程度上减少对系统运行的影响。

Holavanalli 等[59]提出权限扩展机制 Flow Permissions，根据数据与敏感 API 的对应关系，利用工具 Blue Seal 静态分析应用，显示应用的权限关联信息及与其他应用间暗含的权限关联信息，提醒用户可能存在的威胁。

Barrera 等[60]用自组织映射方法分析应用程序，研究权限的设置粒度是否合理及权限间的关系，并提出权限增强安全模型以识别频繁使用的权限。

以上这些权限管理方案都能从某个角度发现问题并起到保护作用，但又都各有局限性，如果将 XMAndroid、Kirin、Saint、Quire 等方案结合，将更全面地保护 Android 系统的安全。

随着 Android 原生系统的不断改进及安全防护软件的升级，权限管理的粒度越来越细化，支持用户自主管理应用的权限，根据需要主动打开或禁用具体应用的指定权限，如用户可禁用聊天软件开启摄像头的权限或者禁用网银软件读取通讯录的权限等。

4.3.2　签名机制

应用的签名信息表明 APK 安装程序的来源。在 Android 平台上，所有的程序都必须有签名，否

则不允许安装。签名是保护应用的第一道防线，比较签名信息可判断应用是否重打包。

2013 年先后爆出 3 个 MasterKey 签名漏洞，恶意应用能够绕过数字签名校验过程，在不被用户察觉的情况下进入 Android 系统，达到恶意目的。漏洞爆出之后，百度安全实验室很快提出 DroidSploit 解决方案，保护 Android 免受此漏洞的威胁。

实际上，Android 应用的签名机制还不够完善，签名的时间戳和文件路径能够被随意修改，这致使攻击者能够绕过验签过程，存在恶意应用能够轻易安装的隐患。

5 Android 应用安全

Android 平台上的应用安全研究主要集中在通过静态分析和动态分析的方式[61]提取应用的特征，判断是否是恶意应用，确保应用的安全运行。Android 恶意应用的研究方法和理论主要借鉴 PC 上的经验，困难之处在于 Android 平台硬件资源相对匮乏，对性能消耗过大。现在正逐渐针对 Android 平台的特性，借助云计算等计算优势检测安全。

5.1 恶意应用静态检测

静态检测就是使用反编译等逆向工程手段，分析代码文件，提取应用程序的特征，如签名、权限、敏感 API 调用等，再通过样本比对或机器学习的方式，判定是否恶意。静态检测方式简单且效率高，但是无法分析混淆、加密等恶意代码。

5.1.1 逆向工程

静态分析的第一步就是逆向工程。一种方式是采用 dex2jar 工具将 DEX 反编译成 Java 源码，采用 AXMLPrinter2 工具处理 AndroidManifest.xml 文件，采用 jd-gui 工具查看 Java 源码，提取表征应用的特征信息。另一种方式是使用 APKtool 和 Smali，将 APK 文件逆向成 Smali 文件[62]。其他反编译工具还包括 Dexdump、Dedexer、androguard、IDA PRO 等。

Enck 等[63]提出 Dalvik 反编译器 Ded，将 DEX 文件反编译成 Java 文件，再对应用进行控制流、数据流、数据结构和语义分析，发现其漏洞和威胁。

5.1.2 应用特征提取和分析

逆向工程之后得到可读的 Java 或者 Smali 代码文件，再基于语义从文件中提取能代表应用的特征，如 Intent、签名、API 函数、类、常量、权限、组件等，通过分析这些特征，判断是否是恶意应用[64]。

Shabtai 等[65]直接从 APK 文件中提取应用的 APK、XML 和 DEX 文件信息，再使用机器学习算法自动化区分应用的类别。

AndroidManifest.xml 中是 APK 文件中最重要的一个文件，描述了应用实现的类、权限、组件、签名、各种被处理的数据和启动信息等。解析该文件中的信息标签，可以在一定程度上判定是否是恶意应用。这方面的研究成果来自 Sarma [66]、Sato [67]、Wu[68]、Weichselbaum[69]等。

以权限为判断特征，分析申请的权限组合，或者设置 Kirin 安全规则[55]，或者将权限与 API 调用合并起来作为特征构建高维特征向量[13]，或者基于权限的行为路径[70]，都可以对应用进行安全检测。

函数调用能够反映应用的行为，Strace 是 Android 平台上的系统调用跟踪器，显示其他进程的系统调用信息，包括参数及返回值；Androguard 是 Google 提供的静态分析工具，将 APK 文件中的 DEX 文件、类、方法等都映射为 python 对象，用于恶意软件检测和恶意评估，分析过程可视化；使用 PScout 工具[71]，分析 API 函数调用与权限检查的情况，构建函数调用图，形成各个 API 函数的权限映射关系；Aurasium[72]在应用程序中嵌入跟踪代码，截获程序的系统调用实现对短信、终端信息、网络等资源使用的监测；基于语义学的 DroidSIFT 原型系统[73]，提取应用的 API，构建基于加权上下文的 API 调用依赖图，并引入应用的相似性来判定变种恶意应用或 0Day 应用。

利用 Intent 传递消息可以被嗅探、篡改或窃取，恶意程序借助 Intent 伪造、注入恶意消息使得用户数据被污染。Chin 提出[44]Comdroid，根据 Intent 分析应用，警告存在的漏洞。Arzt 等[74]提出污点分析系统 FlowDroid，实现对应用信息流的管理。

签名能够判断应用，Hu 等[75]提出使用命令从 DEX 文件中提取签名信息作为训练集，再采用字符串相近算法提取签名信息以检测恶意应用，有较高的检测效率和准确度。

使用单一特征分析应用存在漏检和误检的情况，可考虑采用多类特征综合分析的方式。通过动态和静态分析应用，提取权限、组件、敏感 API、行为调用序列等特征，对各类特征分别选取最优的基础分类算法，得到恶意应用综合判定结果。

5.1.3 机器学习智能检测

机器学习具有大规模数据处理能力，能在相似的数据结构中做出对目标的判断。在对大量

Android 应用进行恶意判定时，引入机器学习算法对应用进行智能分析是非常合适的，常用机器学习算法包括朴素贝叶斯（NB, naive Bayes）、决策树（DT, decision tree）、*K*-最近邻（KNN, *K*-nearest neighbor）、支持向量机（SVM, support vector machine）、*K*-means 等。早在 20 年前，就有专家学者基于机器学习的各种算法对恶意代码和恶意应用进行检测，而在 Android 应用检测方面，也已经有很多尝试。Android 机器学习恶意应用检测结构如图 3 所示。

Schmidt 等[76]采用 CART、prim 和 KNN 算法通过静态分析对 DEX 文件中的调用函数进行机器学习；Wu 等[68]提出 DroidMat，以权限、Intent、API 为特征，使用 *K*-means 和 EM 聚类算法对恶意应用行为建模再采用 KNN 和 NB 分类算法判定恶意应用，有较高的执行效率和可扩展性；以 API 调用和权限为特征，Peiravian 等[13]使用 SVM、CART 和 Bagging 3 种机器学习方法，Gates 等[77]采用基于正则逻辑回归的概率判别模型，检测应用分类；而 Yerima[27]等运用 NB 方法对 Android 恶意软件进行分类。

在对上述机器学习算法综述的基础上，进一步总结出各主要算法的特点和优缺点，如表 2 所示。

利用机器学习归纳出已知应用的类别，有效预测未知的恶意代码。目前，基于机器学习的检测技术可以去人工化，提高应用分析的效率，有很广的应用前景，但是这种方式严重依赖于提取的应用特征和应用样本的质量，工业应用时，往往根据不同的应用场景，调用不同的机器学习算法，而且其复杂度和准确率都待改善，这将成为未来恶意检测研究的重点。

5.1.4　重打包检测

对应用逆向工程之后，若继续将应用安装到智能终端上，就需要对其重打包，重打包的应用很可能携带恶意代码，对此，检测应用是否重打包意义非凡。先后有专家学者提出检测工具 DroidMOSS[78]、DNADroid[79]、Juxtapp[80] 和 SCSdroid[81]，分别采用模糊散列算法、构造应用程序类中的方法构造程序依赖图、特征散列算法，计算软件市场上的应用软件与官方市场中的软件的相似性，当相似性大于一个阈值时，则判定为重打包软件，用户安装需谨慎。

Zhou 等[78]提出 DroidMoss，比较官方应用与第三方应用市场上应用的开发者信息和代码指令的散列值，判断应用是否被二次打包。缺点是难

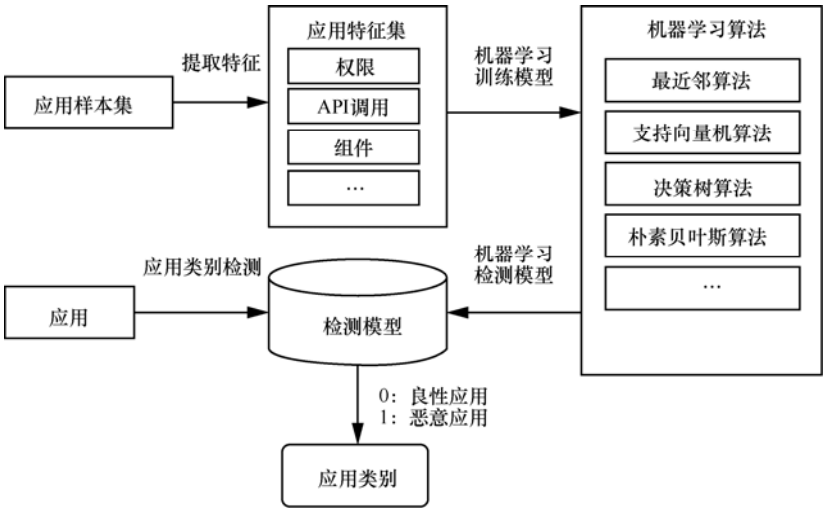

图 3　Android 机器学习恶意应用检测结构

表 2　　机器学习算法的特点和优缺点

机器学习算法	算法应用特点	算法优点	算法缺点
朴素贝叶斯（naive Bayes）	在统计数据的基础上，依据样本特征，计算各个类别的概率，实现分类	算法简单、高效，对缺失数据不太敏感，误差率小；数据较少时仍然有效	独立性假设有时不合理，影响准确率
支持向量机（SVM）	训练数据分布在 *N* 维平面上，通过训练，找到分类间的边界	解决小样本、非线性及高维模式识别的优势明显	对大规模训练样本难以实施；噪声数据影响准确率
决策树（DT）	通过训练数据构建决策树，高效地对未知数据分类	灵活，高效；在面对数值缺失、变量数较多等问题时稳健	难于处理大训练集；对连续性数据比较难预测；易产生过拟合问题
K-最近邻（KNN）	一个样本在特征空间中的 *K* 个最相邻的样本中的大多数属于某一个类别	精度高，对异常值不敏感，无数据输入假定	计算量大；样本不平衡时，准确率低

以获得所有的官方原始应用，因而检测覆盖面较小。Lin 等[81]提出 SCSdroid，提取线程的公共系统调用序列，根据公共调用序列可以检测应用，而不用获取原始应用，实验证明这种方式的正确率高达 95%。

5.2 恶意应用动态检测

对恶意应用动态检测，提取应用运行中的关键数据为特征，通过异常检测或机器学习的方式，实现智能化的检测分析。动态检测绕过了静态方法遇到的代码混淆和加密等问题，对于新出现的未知恶意软件也能保持较好的检测准确率。但是实时性不高，提取特征耗费时间较长，消耗较多系统资源，而且很难覆盖到应用的全部执行路线，比较难发现特定条件下发生的恶意行为。

随着云计算技术的不断发展，通过云计算对恶意样本进行维护，有效解决样本库内容不断增加的难题，突破了样本库维护的局限，还把应用智能分类的核心计算部分移到了云端，降低了终端上的开销。

基于云计算的 Android 应用动态检测，首先在手机终端捕获应用行为，提取特征，传输到云端，再在云端对应用行为进行异常检测，再将检测结果返回到 Android 终端，由终端对异常行为做成阻塞或者中断处理[82,83]，Android 动态行为检测架构如图 4 所示。

应用动态分析可监控软件安装和卸载、联网、收发短信息、后台拨打和接听电话、操作数据库、权限检查、文件接收和发送、摄像头操作等行为。

可采用 3 种方式获取行为信息：第 1 种是静态注入技术，对应用程序反编译嵌入监控 API 调用的源码，再编译和重打包，将修改后的应用安装到 Android 平台，应用运行时，监控代码便能实现对行为的跟踪；第 2 种是 API Hook 技术，在 Android 平台上对敏感 API 进行 Hook，一旦系统或应用对特定的 API 调用时，可截获调用函数，将其重定向到代理函数，在代理函数中获取该 API 调用的详细信息，即可获取行为信息；第 3 种是进程捕获，提取关键行为的特征。如何实现应用行为监控，如何提取行为特征，研究者们开展了广泛而深入的研究。

Xu 等[72]提出 Aurasium，在应用程序中嵌入跟踪程序，截获应用程序的系统调用，检测短信、设备信息、网络等资源的使用情况。

Bläsing 等[84]提出应用沙箱 AASandbox，在隔离的环境中对应用进行静态和动态分析，检测恶意应用；Shabtai 等[14]提出基于行为的恶意软件检测系统 Andromaly，收集 Android 平台运行各种特征，通过 *K*-means、DT、NB、LR 等机器学习算法对收集到的数据分类，并比较了各种算法的准确性。

Burguera 等[83]提出 Crowdroid，在 Android 端使用 strace 追踪器采集行为数据，传到分析服务端，利用分类器训练这些行为样本，使用 KNN 算法判断应用是否含有恶意行。

另外，还有其他几种应用动态分析工具，如 Droidbox、SandDroid 等。它们都能够捕获网络数据，包括监控网络、文件、短信泄露的信息等，是具有

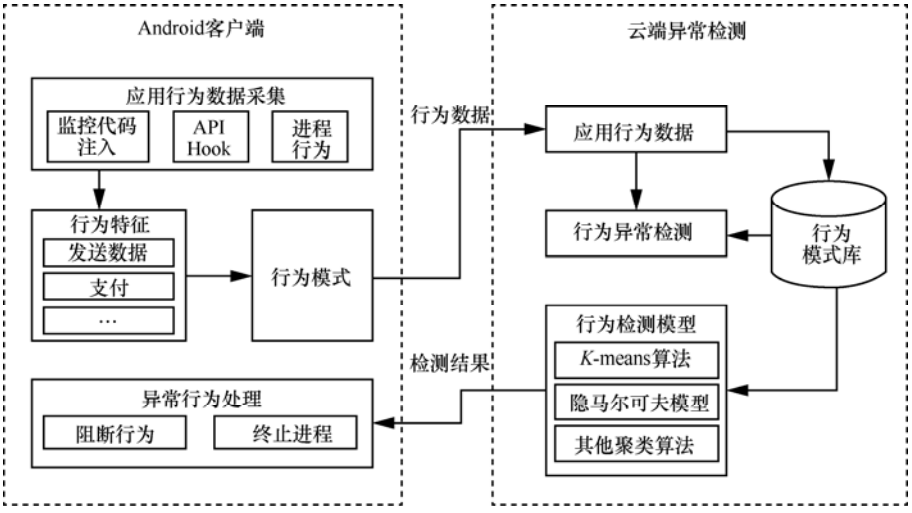

图 4 基于云计算的 Android 动态行为检测架构

一定成熟度的动态分析工具。

5.3　应用安全加固

随着 Android 应用逆向工程的快速发展，对应用破解变得轻而易举，对应用开发者来说，辛苦开发出的成果可能被恶意者植入病毒、篡改、劫持等非法利用，把良性应用变成恶意应用被用户抛弃，对用户来说，安装被反编译过的应用存在很大风险。鉴于此，应用的安全加固必不可少。

应用安全加固主要采用的技术是防反编译，禁止调用 gdb、gcore 从内存中截取 DEX 文件以防内存窃取，禁止向进程中动态注入代码以防动态跟踪，校验 APK 完整性以防恶意篡改等，且加固后的程序不影响其运行效率和用户体验。

比较流行的应用防反编译的方式有代码混淆、加壳等，但最有效的是 NDK 开发应用，编写 Native 代码，即使应用被反编译，也无法得到 C/C++代码，导致无法阅读和修改全部应用代码。另外，还可以通过代码加密技术对 NDK 应用加壳，使程序破解难上加难，但这也对开发人员的技能要求很高。

总之，对 Android 应用程序的保护方式有很多种，而真正保证安全，还是要寻求更可靠的加密方式，希望在这方面引起学者足够的重视及更多的投入。

5.4　应用安全评估

恶意应用产生的威胁程度需要通过建立评估体系和评估标准，从多个角度进行评价，最后综合得到一个结果，反映出应用的脆弱性和风险。

Grace 等[61]将恶意应用的风险等级分为 3 级：高级、中级和低级。根据利用系统漏洞跳过合法权限破坏系统判定为高级风险，根据造成用户隐私泄露和经济损失判定为中级风险，根据收集设备信息和用户信息判定为低级风险。通过这 3 种风险判别，发现了大量 0Day 漏洞。MWR InfoSecurity 提出应用安全评估框架 Drozer，通过分析应用程序与 Dalvik、其他应用及底层操作系统的交互数据，发现漏洞。

Wang 等[17]提出基于权限的安全风险评估框架 DroidRisk，将应用的风险量化，并且能够提示潜在的恶意行为，但是一旦应用申请的权限过度，需要考虑其他因素使用其他方式来评估风险。

对 Android 应用风险评估，帮助用户了解恶意软件可能存在的攻击路径，指导使用者更好地保护他们的隐私数据，以规避可能受到的攻击。

6　Android 隐私数据安全保护

6.1　隐私数据伪造和跟踪

隐私数据伪造是保护数据的一种方式，当应用被恶意访问时，为了保护自己的隐私数据不被泄露，提供虚假的数据。在这方面的研究成果主要有 TISSA[85]、MockDroid[18]和 AppFence[86]等。他们都能够对外提供伪造的数据，包括位置信息、设备 ID、联系人信息等。

对敏感数据做污点标记是隐私数据保护的另一种方式，跟踪数据的流向，记录到日志，为数据溯源提供依据。在这方面第一个研究成果是 Enck 等[19]提出的污点标记系统 TaintDroid，此后，有多名学者在此基础上进一步研究，提出了各种污点数据跟踪系统，如 AndroidLeaks[87]、Kynoid[88]等，都能够跟踪数据去向，分析应用程序当中是否存在隐私泄露行为。

6.2　数据加密

加密是对数据保护最直接的一种保护方式，Android 平台上的数据加密主要体现在对短信、通讯录、通话记录等 SQLite 数据库中的数据加密，以及对图片、音频、视频等文件数据加密。通常采用透明加解密方式，即加解密过程都在后台进行，用户感知不到，且不改变用户对数据的使用习惯，这就对加解密在效率、安全性、开发难度、移植性上达到平衡有较高要求。

对 SQLite 数据库数据加密的方式一般有 2 种，一种是在数据入库、出库时进行加解密操作，但是不利于加密信息的检索；另一种是对整个数据库文件加解密，普遍采用这种方式，通常采用 AES 或 DES 等加密算法。

Wang 等[89]实现基于 FUSE 的文件透明加密系统 EncFS，用户创建目录作为文件系统，所有写入该系统的文件都会被加密。该系统支持 AES 和 Blowfish 加密算法，经过不断优化，加密文件系统对原系统的性能影响不大。

Android 5.0 采用全盘加密来提升安全性能，所有写入硬盘的数据均需要先加密，所有读取的数据都需要先解密。不足之处在于，大多数硬件设备上的闪存并没有自带加密标准，在开启全盘加密后，使机器性能大幅下降。然而，可通过采用更快的闪存芯片和更快的文件系统如 F2FS、优化 SoC 等方式进行改进。

6.3 数据云备份

云服务已经成为智能终端的另一标配，基于云计算将智能终端上的数据远程备份到云端，包括通讯录、短信、图片等个人隐私数据。日常时将终端设备上的数据同步到云端，当终端数据遭受损失时通过网络恢复数据到本地，或者实现多个设备间的数据统一。同时，云备份还具备定时增量备份、加密存储、数据管理和恢复等功能。

为了保证数据在网络传输过程中不被窃听、非法拦截，智能终端与云端备份服务器通信主要是基于 SSL 协议，提供双向认证、数据加密、数据完整性验证等服务。Android 对 SSL 协议有很好的支持，用到的相关类在 javax.net.ssl、java.security 及 javax.crypto 包中。

6.4 数字版权保护

数字版权保护（DRM, digital right management）从技术上防止数字内容被非法复制、篡改，只有授权后才能合法使用，达到数字内容版权保护的目的。Android 平台集成了 OMA DRM 2.0 的部分功能，提供了一个可扩展的数字版权管理框架，Marlin DRM 也是广受认可的内容保护及内容管理的开放标准，不同的 DRM 解决方案可通过 Plugin 方式集成到 Android 系统中，允许应用程序管理和保护自己的数据。

随着研究的发展，PC 上成熟的 DRM 技术，如基于时间/空间约束的 DRM 授权模型、添加水印的音视频 DRM 技术、构建智能终端共享域并设计域内 DRM 安全授权模式、基于角色的可信数字版权安全许可授权模型等，在 Android 平台上的扩展使用是 DRM 发展的又一重要方向。

6.5 安全支付

随着移动互联网的发展，移动支付在生活中扮演越来越重要的角色，而基于 NFC 技术的 Android 支付平台，更加即时、安全和快捷地实现近场付款及远程支付[90,91]。

由于支付与用户的经济利益切实相关，其安全不容忽视，对于移动支付安全来说，隐患主要来自 3 个方面：一是外部钓鱼网站和恶意入侵，二是不安全无线网络接入环境，三是终端安装了恶意的应用。采取的安全防护类型包括以下 5 种：病毒扫描，保证手机不被感染；运行环境监测，监视应用的正常运行；无线 Wi-Fi 网络扫描，阻止恶意攻击；短信验证和密码保护保障支付安全；分辨钓鱼网站，避免账户信息泄露。

7 趋势和展望

Android 智能终端安全保护相关技术经过专家学者的不懈付出，已经取得了显著的成绩，而全能有效地解决实际应用的成果还亟待出现，因此，针对 Android 安全的研究还将会是一项持久艰巨的任务，主要可能围绕以下几个方面展开。

1) 构建通用的 Android 内核安全增强框架，及时发掘系统漏洞，增强对系统的监控能力；隔离系统内核层和应用程序框架层，减少病毒对系统内核的侵害；细粒度应用权限设置，增强用户对权限管理的参与性。

2) 构建轻量级的 Android 动态恶意应用检测模型，建立行为模式，针对应用当前运行状态，检测出潜在的威胁，并实现对应用进行安全风险评估，实时展示评估结果，及时给出安全预警，增强用户使用各种应用软件的安全性。

3) 构建高效的大规模恶意应用检测系统，针对层出不穷、数量庞大的恶意应用，提取应用特征，并进行特征优化选择，再基于机器学习算法，智能化地对恶意应用检测，不断改进检测算法，提高检测的准度、精度和效率，杜绝恶意应用的传播扩散。

4) 构建全面的 Android 移动终端隐私数据保护系统，针对短信、通讯录等数据库数据及图片、音视频等文件数据，加密保护隐私数据的内容，验证隐私数据访问的身份，限制隐私数据使用的方式，跟踪记录隐私数据的流向，通过多种手段，全面有效地保护隐私数据的安全。

5) 构建基于云计算和大数据的 Android 智能终端防护体系，整合云计算、大数据资源在数据处理和存储方面的优势，弥补 Android 计算资源的不足，通过大数据挖掘及时检测 Android 病毒，通过云服务实时维护 Android 终端信息和数据、查杀病毒、防盗防骗等，拓宽智能终端安全防护的体系。

8 结束语

Android 的开放性和流行性吸引了众多攻击者和安全研究者的关注，攻击者利用系统漏洞、入侵应用软件、病毒植入等各种可能的手段试图获取隐私数据以谋求非法利益。安全防护者针对各种威胁，从终端设备本身的硬件安全、所处网络环境的安全到 Android 系统的内核增强、权限细粒度增强

管理，到应用静态提取特征、应用签名、动态行为智能分析，再到隐私数据加密、备份、版权保护、支付安全等各个方面提出防范措施，让破坏分子无处遁形。本文主要对 Android 的设备安全、网络安全、系统安全、应用安全和数据安全进行了分析研究，并展望了未来的发展方向，希望对后续的研究有所帮助。

参考文献：

[1] 360 互联网安全中心.2015 年手机安全状况报告[R]. 2015.
360 Internet Security Center. 2015 mobile security status report [R]. 2015.

[2] 腾讯移动安全实验室. 2015 年上半年手机安全报告[R]. 2015.
Tencent Mobile Security Laboratory. Mobile security report in the first half of 2015 [R]. 2015.

[3] FAUKI P, BHAMAL A, LAXMI V, et al. Android security: a survey of issues, malware penetration, and defenses[J]. Communications Surveys & Tutorials, 2015, 17(2): 998-1022.

[4] SEO S H, GUPTA A, MOHAMED S A, et al. Detecting mobile malware threats to homeland security through static analysis[J]. Journal of Network and Computer Applications, 2014, 38(2): 43-53.

[5] 冯登国, 孙悦, 张阳. 信息安全体系结构[M]. 北京: 清华大学出版社, 2008: 1-14.
FENG D G, SUN Y, ZHANG Y. Information security architecture[M]. Beijing: Tsinghua University Press, 2008: 1-14.

[6] JANG J, W J Y, MOHAISEN A, et al. Andro-AutoPsy: anti-malware system based on similarity matching of malware and malware creator-centric information[J]. Digital Investigation, 2015, 14(6): 17-35.

[7] NIKOLAY E. Android security internals: an in-depth guide to android's security architecture[M]. No Starch Press, 2014.

[8] STEFFEN L, MATTHIAS L. Android security, pitfalls and lessons learned[C]//The 28th International Symposium on Computer and Information Sciences. Berlin, Germany, c2013: 409-417.

[9] KARI K, ELENA R, JAN ERIK E, et al. Old, new, borrowed, blue: a perspective on the evolution of mobile platform security architectures[C]//The First ACM Conference on Data and Application Security and Privacy. New York, USA, c2011: 13-24.

[10] LUBKE R, SCHUSTER D, SCHILL A. A framework for the development of mobile social software on Android[C]//Third International Conference, MobiCASE 2011. Los Angeles, CA, USA, c2011: 207-225.

[11] HONG Y R, DONGSOO K. Security enhancement of smart phones for enterprises by applying mobile VPN technologies[C]//Computational Science and Its Applications(ICCSA). Santander, Spain, c2011: 506-517.

[12] KORKMAZ I, METIN S K , GUREK A, et al. A cloud based and Android supported scalable home automation system[J]. Computers & Electrical Engineering, 2015, 43(5): 112-128.

[13] PEIRAVIAN, N. ZHU X Q. Machine learning for Android malware detection using permission and API calls[C]//IEEE 25th International Conference on Tools with Artificial Intelligence (ICTAI). Herndon, VA, c2013: 300-305.
2016127-13

[14] SHABTAI A, KANONOY U, ELOVICI Y, et al. Andromaly: a behavioral malware detection framework for android devices[J]. Journal of Intelligent Information Systems, 2012, 38(1): 161-190.

[15] SCHREUDERS Z C, MCGILL T, PAYNE C. The state of the art of application restrictions and sandboxes: a survey of application-oriented access controls and their shortfalls[J]. Computers & Security, 2013, 32(2): 219-241

[16] JUNFENG X, LI Z , DONG L, et al. Recommendable schemes of anti-decompilation for android applications[C]//2015 Ninth International Conference on Frontier of Computer Science and Technology (FCST). Dalian, c2015: 84-90.

[17] WANG Y, ZHENG J, SUN C, et al. Quantitative security risk assessment of Android permissions and applications[C]//27th Annual IFIP WG 11.3 Conference. Newark, NJ, USA, c2013:226-241.

[18] BERESFORS A R, RICE A, SKEHIN N, et al. MockDroid：trading privacy for application functionality on smartphones[C]//The 12th Workshop on Mobile Computing Systems and Applications. New York, USA, c2011: 49-54.

[19] ENCK W, GILBERT P, CHUN, et al. Taintdroid: an information-flow tracking system for realtime privacy monitoring on smartphones[C]// The 9th USENIX Conference on Operating Systems Design and Implementation. Berkeley, CA, USA, c2010: 1-6.

[20] TEUFL P, THOMAS Z, CHRISTOF S. Mobile device encryption systems[C]//8th IFIP TC 11 International Conference. Auckland, New Zealand, c2013: 203-216.

[21] HUANG T Y, WANG H, PENG C L, et al. A new remote desktop approach with mobile devices: design and implementation[M]//Ubiquitous Computing Application and Wireless Sensor, 2015, 331: 305-321.

[22] NAKAO K. NAKAMOTO Y. Toward remote service invocation in Android[C]//Ubiquitous Intelligence & Computing and 9th International Conference on Autonomic & Trusted Computing (UIC/ATC). Fukuoka, c2012: 612-617.

[23] BELKEDE M, GULHANE V, BAJAI P. Biometric mechanism for enhanced security of online transaction on Android system: a design approach[C]//Advanced Communication Technology (ICACT). Pyeong Chang, c2012: 1193-1197.

[24] KHANDELWAL A, MOHAPTRA A K. An insight into the security issues and their solutions for android phones[C]//Computing for Sustainable Global Development (INDIACom). New Delhi, c2015: 106-109.

[25] MA L, TEYMORIAN A Y, CHENG X. A hybrid access point protection framework for commodity Wi-Fi networks[C]//The 27th Conference on Computer Communications. Phoenix, AZ, c2008: 1894-1902.

[26] WANG D, ZHOU M. A framework to test reliability and security of Wi-Fi device[C]//Electronic Packaging Technology (ICEPT). Chengdu, c2014: 953-958.

[27] YERIMA S Y, SEZER S, MCWILLIAN G. Analysis of Bayesian classification-based approaches for Android malware detection[J]. Information Security, IET, 2013, 8(1): 121-129.

[28] SHINA S, ANITHA R, NATARAJAN V. Android based malware detection using a multifeature collaborative decision fusion approach[J]. Neurocomputing, 2015, 151(3): 905-912.

[29] FELT A P, WAGNER D. Phishing on mobile devices[M]. NA, 2011.

[30] BOTTAZZI G, CASALICCHIO E, CINGOLANI D, et al. MP-Shield: a framework for phishing detection in mobile devices[C]//2015 IEEE International Conference on Computer and Information Technology, Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing, Pervasive Intelligence and Computing (CIT/IUCC/DASC/PICOM). IEEE, c2015: 1977-1983.

[31] WU L, DU X, WU J. MobiFish: a lightweight anti-phishing scheme for mobile phones[C]//2014 23rd International Conference on Computer Communication and Networks. IEEE, c2014: 1-8.

[32] HE M, HORNG S J, FAN P, et al. An efficient phishing webpage detector[J]. Expert Systems with Applications, 2011, 38(10): 12018-12027.

[33] BASNET R, MUKKAMALA S, SUNG A H. Detection of phishing attacks: a machine learning approach[M]//Soft Computing Applications in Industry. Springer Berlin Heidelberg, 2008: 373-383.

[34] SHABTAI A, FLEDEL Y, ELOVICI Y. Securing Android-powered mobile devices using SELinux[J]. Security & Privacy, 2010, 8(3): 36-44.

[35] SMANEY S, CRAIG R. Security enhanced (SE) android: bringing flexible MAC to Android[C]//The 20th Annual Network and Distributed System Security Symposium. Switzerland, c2013: 20-38.

[36] AVD Android 漏洞库[EB/OL]. http://android.scap.org.cn/avdview.html. AVD Android vulnerability database[EB/OL]. http://android.scap.org.cn/avdview.html

[37] ZHANG W, CAO C, LIU W, et al. Vulnerability mining techniques in Android platform[J]. CCIS-13, 2013, 52(1391):535-540.

[38] STIRPARO P, FOVINO I N. KOUNELIS I. Data-in-use leakages from Android memory-test and analysis[C]//Wireless and Mobile Computing. Networking and Communications (WiMob). Lyon, c2013: 701-708.

[39] SHAHRIAR H, NORTH S, MAWANGI E. Testing of memory leak in Android applications[C]//High-Assurance Systems Engineering (HASE). Miami Beach, FL, c2014: 176-183.

[40] ALESSANDRO A, ALESSIO M, MAURO M, et al. Breaking and fixing the android launching flow[J]. Computers & Security, 2013, 39: 104-115.

[41] DAVI L, DMITRIENKO A, SADEGHI A. Privilege escalation attacks on Android[M]//Information Security. Springer Berlin Heidelberg, 2010: 346-360.

[42] BUGIEL S, DAVI L, DMITRIENKO A, et al. Poster: the quest for security against privilege escalation attacks on Android[C]//The 18th ACM Conference on Computer and Communications Security. ACM, c2011: 741-744.

2016127-14

[43] DEMERTZIS K, ILIADIS L. SAME: an intelligent anti-malware extension for android ART virtual machine[C]//Computational Collective Intelligence 7th International Conference, ICCCI 2015. Madrid, Spain, c2015: 235-245.

[44] CHIN E, FELT A P, GREENWOOD K, et al. Analyzing inter-application communication in android[C]//The 9th International Conference on Mobile Systems, Applications, and Services. New York, USA, c2011: 239-252.

[45] ENCK W, ONGTANG M, MCDANIEL P. Understanding Android security[J]. IEEE Security & Privacy, 2009, 7(1): 50-57.

[46] 傅建明, 李鹏伟, 易乔. Android 组件间通信的安全缺陷静态检测方法[J]. 华中科技大学学报（自然科学版）, 2013, 41: 259-264.
FU J M, LI P W, YI Q. A static detection of security detects between inter-components' communication [J]. Journal of Huazhong University of Science and Technology (Natural Science Edition), 2013, 41: 259-264.

[47] FANG Z R, HAN W L, LI Y J. Permission based Android security: issues and countermeasures[J]. Computers & Security, 2014, 43(6): 205-218.

[48] SCHLEGEL R, ZHANG K, ZHOU X, et al. Soundcomber: a stealthy and context-aware sound trojan for smartphones[C]//NDSS. San Diego, California, USA, c2011: 17-33.

[49] GARCE M, ZHOU Y, WANG Z, et al. Systematic detection of capability leaks in stock Android smartphones[C]//The 19th Network and Distributed System Security Symposium (NDSS 2012). San Diego, CA, c2012.

[50] ONGTANG M, MCLAUGHLIN S, ENCK W, et al. Semantically rich application-centric security in Android[C]//In ACSAC '09. Computer Security Applications Conference. Honolulu, HI, c2009: 340-349.

[51] BUGIEL S, DAVI L, DMITRIENKO A, et al. XManDroid: a new Android evolution to mitigate privilege escalation attacks[R]. Technical Report TR-2011-04, Technische Universitat Darmstadt. Germany, c2011: 1-17.

[52] DIETZ M, SHEKHAR S, PISETSKY Y, et al. QUIRE: lightweight provenance for smart phone operating systems[C]//The 20th USENIX Conference on Security. USENIX Association Berkeley, CA, USA, c2011:23.

[53] LU L, LI Z C, WU Z Y, et al. CHEX: statically vetting android apps for component hijacking vulnerability[C]//ACM Conference on Computer and Communications Security. New York, USA, c2012: 229-240.

[54] ZHONG Y, XIN Z, MAO B, et al. DroidAlarm: an all-sided static analysis tool for android privilege-escalation malware[C]//The 8th ACM SIGSAC Symposium on Information, Computer and Communications Security. New York, USA, c2013: 353-358.

[55] ENCK W, ONGTANG M, MCDANIEL P. On lightweight mobile phone application certification[C]//ACM Conference on Computer and Communications Security. Chicago, USA, c2009: 235-245.

[56] NAUMAN M, KHAN S, ZHANG X. Apex: extending Android permission model and enforcement with user-defined runtime constraints[C]//The 5th ACM Symposium on Information, Computer and Communications Security. New York, USA, c2010: 328-332.

[57] FELT A, CHIN E, HANNA S, et al. Android permissions demystified[C]//The 18th ACM Conference on Computer and Communications Security. New York, USA, c2011: 627-637.

[58] GENEIATAKIS D, FOVINO I N, KOUNELIS I, et al. A permission verification approach for android mobile applications[J]. Computers & Security, 2015, 49(3): 192-205.

[59] HOLAVANALLI S, MANUEL D, NANJUNDASWAMY V, et al. Flow permissions for Android[C]//2013 IEEE/ACM 28th International Conference on Automated Software Engineering (ASE). Silicon Valley, CA, c2013: 652-657.

[60] BARRERA D, KAYACIK H G, OORSCHOT P C, et al. A methodology for empirical analysis of permission-based security models and its application to Android[C]//The 17th ACM Conference on Computer and Communications Security. New York, NY, USA, c2010: 627-637.

[61] GRACE M, ZHOU Y J, ZHANG Q, et al. RiskRanker: scalable and accurate zero-day Android malware detection[C]//The 10th International Conference on Mobile Systems, Applications, and Services (MobiSys). ACM, Lake District, UK, c2012: 281-294.

[62] ZHENG M, SUN M, LUI J. Droid analytics: a signature based analytic system to collect, extract, analyze and associate Android malware[C]//The 12th IEEE International Conference on Trust Security and Privacy in Computing and Communications. c2013: 163-271.

[63] ENCK W, OCTEAU D, MCDANIEL P, et al. A study of Android application security[C]//The 20th USENIX Conference on Security. USENIX Association Berkeley, CA, USA, c2011:1175.

[64] FEIZOLLAH A, ANUAR N B, SALLEH R, et al. A review on feature selection in mobile malware detection review article[J]. Digital Investigation, 2015, 13(6): 22-37.

[65] SHABTAI A, FLEDEL Y, ELOVICI Y. Automated static code analysis for classifying Android applications using machine learning[C]//The 2010 International Conference on Computational Intelligence and Security(CIS). Nanning, China, c2010: 329-333.

[66] SARMA B P, LI N, GATES C, et al. Android permissions: a perspective combining risks and benefits[C]//The 17th ACM Symposium on Access Control Models and Technologies. New York, USA, c2012: 13-22.

[67] SATO R, CHIBA D, GOTO S. Detecting Android malware by analyzing manifest files[C]//The Asia-Pacific Advanced Network. Tokyo, c2013: 23-31.

[68] WU D J, MAO C H, WEI T E, et al. DroidMat: Android malware detection through manifest and API calls tracing[C]//Seventh Asia Joint Conference on Information Security (Asia JCIS). Tokyo, c2012: 62-69.

[69] WEICHSELBAUM L, NEUGSCHWANDTNER M, LINDORFOR M,et al. Andrubis: Android malware under the magnifying glass[R]. Vienna University of Technology, 2014. TRI-SECLAB- 0414-001 . 2016127-15

[70] HOU Y, WANG Z, ZHOU W, et al. Hey, you, get off of my market: detecting malicious apps in official and alternative android markets[C]//The 19th Annual Network and Distributed System Security Symposium(NDSS). San Diego, c2012.

[71] AU K W Y, ZHOU Y F， HUAGN Z, et al. Pscout: analyzing the android permission specification[C]//The 2012 ACM Conference on Computer and Communications Security. New York, USA, c2012: 217-228.

[72] XU R, SAIDI H, ANDERSON R. Aurasium: practical policy enforcement for Android application[C]//The 21st USENIX Conference on Security Symposium. USENIX Association Berkeley, CA, USA, c2012:27.

[73] ZHANG M, DUAN Y, YIN H, et al. Semantics-aware Android malware classification using weighted contextual API dependency graphs[C]//The 2014 ACM SIGSAC Conference on Computer and Communications Security. New York, USA, c2014: 1105-1116.

[74] ARZT S, RASTHOFER S, FRITZ C, et al. FlowDroid: precise context, flow, field, object-sensitive and lifecycle-aware taint analysis for Android apps[C]//The 35th ACM SIGPLAN Conference on Programming Language Design and Implementation. c2014: 259-269.

[75] HU G, LI T, DONG H, et al. Malicious code detection for Android using instruction signatures[C]//IEEE 8th International Symposium on Service Oriented System Engineering. Oxford, c2014: 332-337.

[76] SCHMIDT A D, BYE R, SCHMIDT H G, et al. Static analysis of executables for collaborative malware detection on Android[C]// IEEE International Conference on Communications. Dresden, c2009: 1-5.

[77] CEN L,GATES C, et al. A probabilistic discriminative model for Android malware detection with decompiled source code[C]//IEEE Transactions on Dependable and Secure Computing. c2013: 1-14.

[78] ZHOU W, ZHOU Y, JIANG X, et al. Detecting repackaged smartphone applications in third-party android marketplaces[C]//The Second ACM Conference on Data and Application Security and Privacy. New York, USA, c2012: 317-326.

[79] CRUSSELL J, GIBLER C, CHEN H. Attack of the clones: detecting cloned applications on Android markets[C]//Proceedings of ESORICS. Berlin, Germany, c2012: 37-54.

[80] HANNA S, HUANG L, WU E, et a1. Juxtapp: a scalable system for detecting code reuse among Android applications[C]//The 9th International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment. Berlin, Germany, c2013: 62-81.

[81] LIN Y D, LAI Y C, CHEN C H, et al. Identifying Android malicious repackaged applications by thread-grained system call sequences[J].

Computers & Security, 2013, 39(12): 340-350.

[82] 文伟平, 梅瑞, 宁戈, 等. Android 恶意软件检测技术分析和应用研究[J]. 通信学报, 2014, 35(8): 79-85.
WEN W P, MEI R, NING G, et al. Malware detection technology analysis and applied research of android platform[J]. Journal on Communications, 2014, 35(8): 79-85.

[83] BURGUERA I, ZURATUZA U, et al. Crowdroid: behavior-based malware detection system for Android[C]//The 1st ACM Workshop on Security and Privacy in Smartphones and Mobile Devices. New York, USA, c2011: 15-26.

[84] BLASING T, BATYUK L, SCHMIDT A D, et al. An Android application sandbox system for suspicious software detection[C]//5th International Conference on Malicious and Unwanted Software (MALWARE). Nancy, Lorraine, c2010: 55-62.

[85] ZHOU Y J, ZHANG X W, et al. Taming information-stealing smartphone applications(on Android)[C]//The 4th International Conference on Trust and Trustworthy Computing. Berlin, Germany, c2011: 93-107.

[86] 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]//The 18th ACM Conference on Computer and Communications Security. New York, USA, c2011: 639-652.

[87] GIBLER C， CRUSSELL J, ERICKSON J, et al. AndroidLeaks: automatically detecting potential privacy leaks in Android applications on a large scale[C]//The 5th International Conference on Trust and Trustworthy Computing. Berlin, Germany, c2012: 291-307.

[88] SCHRECKLING D, POSEGGA J, et al. Kynoid: real-time enforcement of fine-grained, user-defined, and data-centric security policies for Android[C]//WISTP'12 The 6th IFIP WG 11.2 International Conference on Information Security Theory and Practice. Berlin, Germany, c2012: 208-223.

[89] WANG Z H, MURMURIA R, STAVROU A. Implementing and optimizing an encryption file system on Android[C]//The 2012 IEEE 13th International Conference on Mobile Data Management. Washington, DC, USA, c2012: 52-62.

[90] TAN G W, OOI K B, CHONG S C, et al. NFC mobile credit card: the next frontier of mobile payment[J]. Telematics and Informatics, 2014, 31(2): 292-307.

[91] 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, 35(z2): 118-123.
WANG Z Q, LIU Q X, ZHANG Y Q. Research of discovering vulnerabilities of NFC applications on Android platform[J]. Journal on Communications, 2014, 35(z2): 118-123.

作者简介：

许艳萍（1986-），女，安徽亳州人，北京邮电大学博士生，主要研究方向为移动互联网安全、Android 智能终端应用安全。

马兆丰（1974-），男，甘肃镇原人，博士，北京邮电大学讲师，主要研究方向为移动互联网安全技术、云计算安全技术、数字版权管理。

王中华（1986-），男，山东聊城人，国家计算机网络应急技术处理协调中心工程师，主要研究方向为移动互联网安全、网络安全攻防演练。

钮心忻（1963-），女，浙江湖州人，北京邮电大学教授、博士生导师，主要研究方向为数字水印、信息隐藏、隐写分析。

杨义先（1961-），男，四川盐厅人，北京邮电大学教授、博士生导师，主要研究方向为密码学、计算机网络与信息安全。