Android SDK安全性研究综述

许腾, 刘路遥, 姜灏宇, 罗畅, 李珩, 袁巍

引用本文

许腾, 刘路遥, 姜灏宇, 罗畅, 李珩, 袁巍. Android SDK安全性研究综述[J]. 计算机科学, 2026, 53(1): 285-297.

XU Teng, LIU Luyao, JIANG Haoyu, LUO Chang, LI Heng, YUAN Wei. Survey on Security of Android SDKs [J]. Computer Science, 2026, 53(1): 285-297.

计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.250500023

Android SDK 安全性研究综述

许 腾[1] 刘路遥[2] 姜灏宇[1] 罗 畅[1] 李 珩[1] 袁 巍[1]
1 华中科技大学电子信息与通信学院 武汉 430074
2 武汉船舶通信研究所 武汉 430070
(xuteng@hust.edu.cn)

摘 要 Android SDK 是 Android 应用开发所使用的软件工具包。由于单个 Android SDK 可以被集成至多个应用,给 Android 生态带来的安全性影响是链式的,因此 Android 生态安全面临着来自 SDK 的全面威胁。近年来一系列与 Android SDK 有关的安全问题,如 SDK 跨库调用隐私数据、SDK 库资源合并覆盖等,引起了工业界和学术界的高度重视。但目前对于 Android SDK 的安全性研究缺乏完备性综述,对此,对现有相关工作进行详细整理,从 Android SDK 的内部组件代码安全与运行数据交互安全展开,在内部组件代码安全上整理了系统 SDK 层面和三方 SDK 层面的研究,在运行数据交互安全上整理了 SDK 本体违规和 SDK 外部入侵方面的研究,并分析了近年来的 Android SDK 安全性工作,引入了性能指标进行横向对比,梳理了其发展脉络和演化过程。最后展望了该领域与如今 AI 大模型等新兴技术结合使用的未来研究方向。

关键词:Android SDK;安全性分析;代码安全;数据交互安全
中图分类号 TP311.5

Survey on Security of Android SDKs

XU Teng[1],LIU Luyao[2],JIANG Haoyu[1],LUO Chang[1],LI Heng[1] and YUAN Wei[1]
1 School of Electronic Information and Communication,Huazhong University of Science and Technology,Wuhan 430074,China
2 Wuhan Marine Communication Research Institute,Wuhan 430070,China

Abstract Android SDK is a software toolkit used for Android application development. Since a single Android SDK can be integrated into multiple applications,its security implications for the installation ecosystem are chain-like,exposing the Android ecosystem to comprehensive threats from SDKs. In recent years,a series of security issues related to the Android SDK,such as SDK cross-library harvests private data and SDK library resource merging and overlay,have attracted high attention from both industry and academia. However,there remains a lack of comprehensive reviews on the security of Android SDKs. This paper systematically organizes existing related work,focusing on two key dimensions:the security of internal component code in Android SDKs and the security of runtime data interaction. For the former,it compiles research findings at both the system SDK and for third-party SDKs. For the latter,it summarizes studies on SDK self-violations and external intrusions into SDKs. Additionally,this paper analyzes recent advancements in Android SDK security research,introduces performance metrics for horizontal comparison,combs through its development context and evolutionary process. Finally,prospects the future research directions for combining this field with emerging technologies such as current AI large language models.

Keywords Android SDK,Security analysis,Code security,Data interaction security

1 引言

随着移动互联网的高速发展,Android 系统因其开源性、灵活性及广泛的市场占有率,已成为全球最主流的移动操作系统之一。根据最新报告显示,截至 2025 年 9 月,Android 设备在全球智能手机中占比达 73.9%,用户超过 36 亿[1]。Android 设备的普及虽然给人们带来了便利,但也使其成为网络安全威胁的主要目标。

Android 应用的开发高度依赖 Android SDK(Software Development Kit,软件开发工具包)。SDK 是进行 Android 应用开发所使用的软件工具包,它提供了硬件基础信息、软件功能接口以及开发所需的工具和库等,支持开发者在 Android 系统上直接调用预置能力实现相应功能模块,不仅可以降低应用开发门槛、加速开发周期,而且具有很强的三方服务拓展功能,且能支持跨平台开发。这给开发人员提供了显著便利,也优化了用户体验,避免了从零编写代码的繁琐步骤。所以,SDK 是构建开发者与操作系统之间交互的重要桥梁。

在过去的十年间,三方 SDK 的数量呈爆发式增长势态,早期大数据中心收录的 SDK 工具包仅有 414 个,而根据最新统计,如今三方 SDK 数量共计超过 2 万款,其集成范围超过

到稿日期:2025-05-09 返修日期:2025-09-15
通信作者:李珩(liheng@hust.edu.cn)

350 万款移动应用[2-3]。随着 SDK 的爆发式增长，其安全问题愈加严峻。如图 1 所示，移动应用安全大数据平台利用安全检测引擎对 SDK 进行 107 项漏洞扫描发现，93.48%以上的 SDK 存在高危漏洞风险，且三方 SDK 的广泛集成导致单个漏洞可能波及超 10 万款应用[3]，威胁形态已从孤立风险演变为系统性危机[4]。

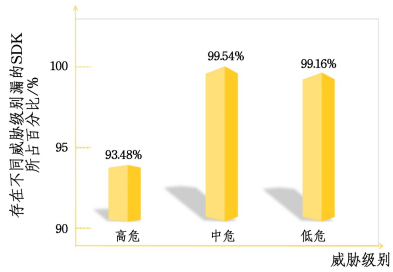

图 1 SDK 风险漏洞情况

Fig. 1 Situation of SDK risk vulnerabilities

随着 SDK 应用规模的持续扩大，Android 安全研究已从单一应用层面向更深层次的生态系统安全维度延伸[5]。研究者逐渐意识到，仅关注应用本身的安全问题不足以应对日益复杂的威胁。但作为应用开发的核心组件，SDK 的广泛使用也增加了后门植入、跨库搜集等安全威胁的数量。从系统安全分析的视角审视，SDK 安全风险的降低对缓解 Android 用户终端面临的安全威胁具有显著的效能。因此，SDK 安全研究作为一个独立且重要的领域开始受到关注。

开发者上传至 Google Play 等市场的应用往往依赖三方 SDK 来实现广告推送、数据分析、社交分享等功能。由于许多三方 SDK 未经过严格的安全审查，早期一些不法分子在 SDK 中有意嵌入包含恶意代码或漏洞的三方库[6-7]。之后随着代码审计规模的扩大，注入恶意代码的方式难以通过审查，因此不法分子利用正常 SDK 的 API 接口缺陷，获取用户的个人数据[8]。这种不直接插入恶意代码的行为增加了检测的难度。近几年，由于 SDK 使用率的增加，部分研究发现，Android系统 SDK 由于本身的更新机制，存在三方库资源覆盖等漏洞[9]；由于需要检测的版本与资源众多，因此风险 SDK 检测的难度进一步加大。SDK 安全威胁呈现出持续演进的特征。

Android SDK 的广泛应用不仅给开发者带来了研发效率上质的飞跃，引入的大量便捷组件和全新功能也给用户带来了更好的使用体验。但对开发者来说，应用开发中的过度便利往往会让开发者忽略 SDK 本身的安全问题，尤其是在缺乏额外代码安全审计的情况下。根据 Ma 等[10]的研究，部分三方 SDK 中可能存在如敏感权限滥用、Web 配置漏洞或是本地日志泄露等问题[11-13]。这些恶意的 SDK 很可能会被开发者无意间引入，一些流行 SDK 甚至会进行大面积链式扩散，影响所有集成的应用[14]。所以，对于开发者，针对 SDK 安全性的研究，可以构建一个更加安全的开发环境，减少在应用安全审计上的额外开销。对用户而言，部分广告 SDK 会通过调用社交媒体公司 SDK 的 API，窃取用户隐私数据[15]，并发送至外部服务器[14]，对用户隐私造成严重伤害。当用户通过各种 SDK 接口实现登录验证等功能时，自身的隐私信息可直接被恶意 SDK 搜集，且根据大多数地区法律规定，这种隐私窃取行为的直接责任方为这些社交媒体 SDK，因为它们并未根据法律要求保护用户隐私，这同样会给正常 SDK 企业带来严重的经济处罚[16]。这些典型案例揭示，SDK 安全风险既表现为系统层面的隐私数据恶意采集与传输，又体现在企业开发端与用户运行端之间的多层级传导，这已经贯穿了整个 Android 系统安全生态框架。因此，无论是对开发方还是用户来说，SDK 的安全性研究至关重要，其不仅能加固企业的安全壁垒，也能给用户营造一个更加安全的使用环境。

2 相关知识

在正式进行 Android SDK 安全研究分析之前，先介绍 Android SDK 的相关知识，包括系统 SDK 与三方 SDK、SDK 三方库，以及 SDK 文本服务条款的主要内容。

系统 SDK 是 Google 为 Android 开发者提供的软件开发工具包[17]，包括一些示例项目、开发工具和模拟器等，开发者通过 SDK 的 API 接口直接与系统交互，实现从基础功能到高级特性的开发，且每个 Android 版本都会引入新的 API 以支持更新功能。此外，还有许多由其他公司或社区开发者组织创建的三方 SDK[10,18]，其包含许多用于集成特定的服务或功能的 Android 三方库[19]，如社交媒体登录、支付处理、AR 等直接实现功能[20-22]，也能提供网络请求、图片处理等功能模块[12-23]，帮助开发者快速解决问题，提升效率。三方库不仅能被集成到三方 SDK 中，也能独立存在。SDK 服务条款一般由 SDK 开发方提供，条款以文本形式列出 SDK 的使用方式与行为准则，旨在规范 SDK 使用。这些组件共同构成了 Android 开发的技术生态。

2.1 系统 SDK 与三方 SDK 相关知识

系统 SDK 是开发 Android 应用程序的基础，是 Android SDK 代码组件部分的核心之一，也是 Android 应用开发的必要组件，包含了开发所需的关键组件，如开发工具、模拟器和使用示例等，开发者通过 SDK 中的 API 直接与这些组件进行交互。除了系统 SDK，还有许多三方 SDK。三方 SDK 也是 Android SDK 代码组件部分的核心，但并非应用开发的必要组件。三方 SDK 由其他公司或组织提供，用于帮助开发者在 Android 应用中集成特定的服务或功能。这些三方 SDK 通常包含一组相关的库、工具和文档，可帮助开发者更容易地集成这些服务。

系统 SDK 通过 Android Studio 进行安装和管理，开发者可以通过 SDK Manager 来更新和配置所需的 SDK。

开发工具包括 SDK 命令行工具、构建工具与平台工具。其中命令行工具用于构建和调试 Android 应用，例如虚拟设备管理、代码扫描等；构建工具则用于编译打包与签名；平台工具则用于构建应用与 Android 平台交互，如应用安装。

Android 模拟器允许开发者在虚拟设备上运行和测试应用，而无需物理设备。

使用示例是 SDK 根据不同的版本更新提供的文档、示例代码和教程，帮助开发者快速上手和解决问题。

系统 SDK 与三方 SDK 的一个重要特点是其持续更新，为了支持 Android 操作系统或是应用开发的新功能和改进，

SDK 每次更新都会引入新的 API，开发者往往需要定期更新 SDK，防止因为兼容性引入新的安全问题。

2.2　Android 三方库相关知识

Android 三方库是由 Google 以外的开发者或组织创建并维护的可重用代码库，也是大部分三方 SDK 的核心组成部分[24]。这类库通常以 JAR，AAR 或 Gradle 依赖[25]的形式提供，涵盖网络请求、图片加载、动画效果等多类功能模块与组件。

三方库在 Android 生态系统中发挥着关键作用，不仅显著降低了开发成本，更通过社区协作实现了持续迭代与功能优化。许多主流应用均依托多方库实现复杂功能。然而，开发者在选型时需重点关注其安全性：部分恶意库虽功能丰富、适配开发需求，却可能包含恶意代码或违规功能模块[26]。此外，三方库的使用也可能引入安全漏洞，甚至导致隐私泄露[27]。因此，开发者需在选型时严格筛选，并定期开展安全审计。

2.3　Android SDK 服务条款相关知识

Android SDK 的服务条款（Terms of Services，ToS）[28]常涵盖行为规范、数据搜集原则等责任要求。条款明确限定了 SDK 的使用范围，禁止未经许可的修改或其他商业用途，同时要求开发者遵守最小必要原则来收集数据，且必须通过隐私政策告知用户数据收集的目的[29]，并提供撤回同意的途径。这些条款为 SDK 行为合规性判断提供了文本依据。

服务条款本身也会面临外部的法律规范限制，对于一些社交广告类 SDK，条款需符合 GDPR[30]，CCPA[31]等数据保护法规，明确数据跨境传输限制和安全措施[32]。Meta 公司就曾因为搜集用户喜好数据进行广告推送而遭受欧洲数据安全局的处罚[33]，这也警示了企业需对其 SDK 服务条款内容进行合规性修正。

2.4　相关知识总结

SDK 的相关知识可以分为代码组件与文本组件，前者包括三方 SDK、系统 SDK 与三方库，后者涵盖了所有的服务条款。各组件详情与安全问题如图 2 所示。

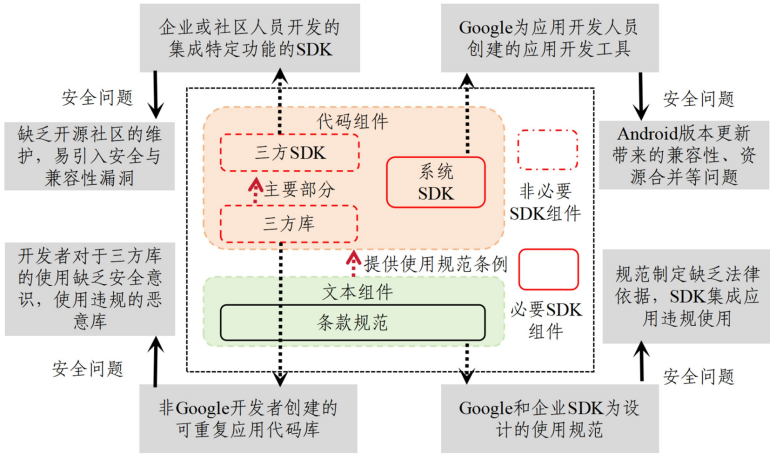

图 2　Android SDK 组件关系与安全问题

Fig. 2　Android SDK component relationships and security issues

系统 SDK 是 Google 公司为应用开发人员创建的应用开发工具，其安全问题在于版本更新带来的兼容性、资源合并等问题。

三方 SDK 是企业或社区人员开发的集成特定功能的 SDK，但容易缺乏开源社区的维护，易引入安全与兼容性漏洞。

三方库是大部分三方 SDK 中的重要组成部分，是非 Google 开发者创建的可重复应用代码库，其安全问题在于开发者对于三方库的使用缺乏安全意识，易引入违规恶意库。

SDK 服务条款则是 Google 和其他企业为开发者以文本形式提供的 SDK 使用样例以及一些数据共享规范等，为代码组件提供了使用规范。但是部分 SDK 的规范制定缺乏法律依据，且在实际行为上可能未遵循规范准则。

3　Android SDK 安全分析方式分类

现有工作针对 Android SDK 的安全性研究提出了多样化的分析方法，因此需要从多维视角对这些研究进行系统性归类，图 3 展示了 SDK 安全性分析的具体分类方式。综合领域研究现状，主要分类依据包括 SDK 研究对象、SDK 检测方法类型以及 SDK 安全风险源头。其中，基于 SDK 安全风险源头的分类方法最能体现 Android 生态中 SDK 安全治理的实际需求，且与移动应用供应链安全的特殊性高度契合。因此，本文在第 4 章将其作为核心分类依据展开论述。

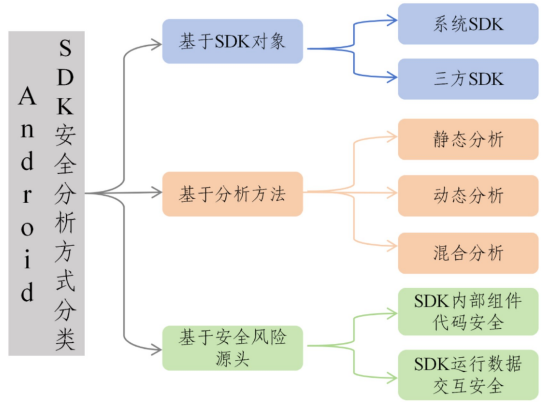

图 3　SDK 安全性分析方法分类

Fig. 3　Classification of SDK security analysis methods

3.1　基于 SDK 对象

Android SDK 包括系统 SDK 与三方 SDK，在系统层面，

其安全问题多集中于版本兼容性与资源管理机制。相较之下，三方 SDK 则完全不同，由于外部开发者缺乏统一的安全管理方式，三方 SDK 的开放性集成模式导致的安全漏洞传播与隐私泄露成为主要风险。由于安全问题来源不同，针对不同 SDK 安全性的研究可以分为系统 SDK 与三方 SDK。

3.1.1 系统 SDK

系统 SDK 的安全问题主要来源于 API 版本兼容性漏洞[34]和资源合并机制缺陷[9]两方面。在 API 兼容性层面，Android 存在版本差异，容易导致 SDK 在跨设备运行时崩溃，所以需要建立跨版本的 API 映射。在资源处理层面，Android 由于资源编译器（Android Resource Compiler，ARC）本身存在设计缺陷，因此在处理多 SDK 集成场景时会发生资源冲突，引发解析错误，需要设计资源编译隔离方案。

3.1.2 三方 SDK

三方 SDK 的安全研究主要涵盖漏洞传播风险、隐私泄露路径与合规性缺陷[35]。在漏洞层面，部分三方 SDK 更新滞后导致漏洞长期未修复[36]；此外，跨库调用机制（即三方 SDK 间通过 API 接口调用数据）可能被恶意 SDK 利用，加剧敏感数据泄露风险，亟需构建 SDK 更新验证框架以阻断漏洞扩散[37-38]。在隐私泄露方面，广告、社交类 SDK[39]隐蔽收集用户兴趣和社交信息等行为与其隐私声明存在显著不一致性，因此亟需完善数据流追踪机制以强化隐私泄露检测能力。在合规性层面，部分三方 SDK 的数据运作行为违反相关规范（包括企业服务条款、Android 开发者规范等），导致与隐私声明的完全背离[40]，需强化 SDK 行为与规范之间的交叉验证。上述研究表明，三方 SDK 因广泛集成和权限滥用，易引发移动生态中链式扩散的系统性安全风险。

3.2 基于分析方法

基于 SDK 检测方法的分类源于不同的技术需求差异，现有检测方法根据 SDK 检测范围、精度需求与规模应用分类为静态分析、动态分析与混合分析[41-42]。当前 SDK 安全分析技术已从人工经验演进至自动化、智能化分析。早期研究主要依赖于人工审计和静态规则匹配，其检测效率低下且误报率较高，难以应对大规模应用场景。随着技术迭代，研究者逐步引入多种自动化分析方法，使检测精度大幅提升，进而显著提升了安全检测的工业化和自动化进度。

其中，静态分析通过逆向解析代码结构可以实现高效批量化检测，但其对动态加载、代码混淆[43]等技术存在解析盲区。动态分析则通过在沙箱中监控 SDK 的运行时行为，借助开源工具实现轻易、直接部署，但难以覆盖所有代码路径。混合分析通过融合静动态数据，旨在平衡效率与精度，但二者结合进行大规模化应用的难度较高。

3.2.1 静态分析

静态分析方法通过逆向解析 SDK 代码[44]，在不执行程序的情况下检测潜在风险。通过检测 API 兼容性、权限分析与漏洞扫描等方式，可以大规模检测 SDK 代码层面的问题，但难以处理代码混淆等加密技术[43]，即便通过特征匹配识别关键逻辑片段，但对重度混淆或加密保护的 SDK 仍存在解析盲区。这种对代码结构强依赖的静态检测局限性，促使研究者转向研究运行时行为的动态监控手段。

3.2.2 动态分析

动态分析方法通过在沙箱环境中运行 SDK 组件，实时监控其运行时行为。借助如 Xposed[45]和污点分析[40]等方法捕获被反射调用、动态加载等方式混淆的代码。受限于沙箱环境与真实设备的差异，一些恶意逻辑代码可能无法被激活发现。此外，动态分析对执行场景的强依赖导致其难以实现大规模并行检测，同时缺乏对 SDK 内部代码结构的全局认知，可能无法发现深层逻辑漏洞。为了平衡静态分析精度与动态分析覆盖面问题，研究者尝试结合二者优点的混合分析方式。

3.2.3 混合分析

混合分析方法通过整合静态代码特征与动态行为日志，构建多维度检测模型。典型范式是静态阶段提取 SDK 的权限与 API 映射关系，动态阶段通过提取运行时的敏感数据传递与权限调用行为[8]。混合分析结合了静态代码特征数据与动态行为特征数据，具有较好的精度与覆盖面，能够同时识别代码层面的潜在漏洞和运行时触发的隐蔽威胁[46]。但二者数据结合过程较为复杂，需要人工参与判定，难以进行大规模工业化应用。因此，目前的研究方向是采用机器学习等方式进行多维度协同分析，从而优化检测过程。

3.3 基于安全风险源头

SDK 安全分析研究可以根据其安全风险源头分为 SDK 内部组件代码安全与 SDK 运行数据交互安全两类。SDK 内部组件代码安全关注 SDK 组件的漏洞；而运行数据交互安全则针对应用运行时 SDK 数据搜集行为的合规合法性。

这种分类方式相比传统安全分析方式在风险视角、责任归属、防护策略以及开发测试上有明显区别和优势，详细对比如表 1 所列。

表 1 传统分类与基于风险来源分类的对比

Table 1 Traditional classification compared with classification based on risk sources

	传统应用安全分析	基于 SDK 安全风险来源的安全分析
风险视角	聚焦应用自身代码、数据存储、权限声明等单一范围	区分 SDK 内部代码安全与外部数据交互安全，覆盖全生命周期风险链
责任归属	通常将 SDK 风险视为应用整体问题，难以区分责任归属	明确划分 SDK 自身缺陷与外部恶意使用
防护策略	缺乏防范依据细节，采用通用模式防护，效率低下	根据内部代码安全或是外部数据交互灵活选择高效防护策略
开发测试	需要投入多种资源进行全生态链级别安全防范	根据风险来源划分安全防范区域，灵活高效选择安全测试方法

首先，在风险视角上，这种分类方式系统性覆盖了从 SDK 开发、运行调试、运行组件交互、敏感数据传输等 SDK 全生命周期风险，便于研究者进行安全测试方案制定；其次，这种分类方式明确了 SDK 开发者与应用开发者之间的责任主体，在代码层面的责任主体主要是 SDK 开发者，在数据交互上则是根据应用权限策略进行责任协定划分；最后，内部代码安全通常具有高危性，而数据交互安全可以通过权限设置等方式快速缓解，这种分类方式也为安全防护的优先级提供了参考。

3.3.1　SDK 内部组件代码安全

SDK 内部组件代码安全风险源于其构成组件中因兼容性或维护缺乏等引入的代码漏洞,需通过系统性治理框架阻断其风险传播。在兼容性层面,不同 Android 版本间差异易引发运行时崩溃[34],需建立跨版本 API 映射验证机制以规避兼容性风险[47]。在安全维护方面,一些三方 SDK 长期沿用含已知漏洞的旧版本三方库与 API,其高危漏洞需要通过检测工具进行修复[44]。这些问题的本质在于开发与维护阶段的安全编码规范缺失以及漏洞维护响应滞后,需构建覆盖全生命周期的治理框架,整合安全设计准则、自动化检测工具响应机制,从源头遏制代码层漏洞的扩散。

3.3.2　SDK 运行数据交互安全

SDK 运行数据交互安全风险表现为其数据搜集行为的合法合规性,包括 SDK 本身的违法数据搜集与三方 SDK 的数据违规调用。SDK 本身的数据搜集行为可能直接违反 GDPR/CCPA 等隐私法规[48],所收集敏感数据类型远超法律边界[35];三方 SDK 调用集成至同一应用的企业 SDK 数据,并违反了企业声明的数据搜集原则,且未提供有效声明或是提供虚假声明[48]。此类问题暴露了当前 SDK 生态的治理困境:SDK 本身数据行为缺乏外部法律监管;开发者为降低开发成本,默认保留一些违规搜集数据的三方 SDK,平台方也缺乏有效的自动化应用合规验证工具。需要构建基于规范的协同治理框架,使数据安全符合条款规定。

4　SDK 安全性分析方法

本章关于 SDK 安全性的分析从如下两方面展开。

1)SDK 内部组件代码安全。SDK 作为软件开发工具包,其内部结构的组件代码可能具有潜在风险,例如使用了存在兼容性问题的 API 接口或是风险漏洞的三方库等,缺乏详细审计的 SDK 被集成至应用后会直接影响应用的正常运行。

2)SDK 运行数据交互安全。SDK 的结构包含各类文本规范,应用运行时 SDK 的数据交互可能违背了规范的行为准则。部分 SDK 的规范内容缺失或描述模糊,导致 SDK 数据收集行为过度,严重违反数据保护法。另一些三方 SDK 通过接入正常企业 SDK 的 API 接口,并将搜集数据发送至外部服务器,这种数据交互直接违背了企业 SDK 规范。

通过将 SDK 的内部组件代码安全和运行数据交互安全作为主要分类方式,对现有分析 Android SDK 的研究进行汇总和梳理,根据研究对象、方法、创新性与文章贡献整理了相关文献,具体内容表 2 所列[49-60]。

表 2　现有研究总结

Table 2　Summary of existing research

分析方法	文献	研究对象	方法	创新	贡献
基于 SDK 内部组件代码安全	Li 等[49]	SDK 的 API	静态分析	构建 API 生命周期模型	系统性检测调用问题
	Huang 等[50]	SDK 的 API	静态分析	检测核心组件控制流	系统性检测 API 回调问题
	Mahmud 等[34]	SDK 的 API	静态分析	API 差异与轻量化分析	检测 SDK 调用与回调兼容性问题
	Wang 等[9]	SDK 的三方库	静态分析	提出为库资源添加唯一命名空间前缀	ARC 导致的三方 SDK 覆盖漏洞
	Derr 等[44]	SDK 三方库	静态分析	结合主观开发者调查与客观代码分析验证	揭示三方库现状与更新障碍需求
	Aniketh 等[52]	SDK 三方库	混合分析	首次大规模实证分析多个 SDK 的行为模式	揭示三方无线扫描 SDK 的安全性
	梁等[53]	云备份 SDK	混合分析	代码-文本结合分析	三方 SDK 漏洞检测
	Ma 等[54]	SDK 三方库	静态分析	API 调用频率特征哈希化与聚类预分析结合	SDK 三方库检测的效率与准确率
	Li 等[55]	SDK 三方库	静态分析	实现静态代码分析和聚类算法的自动化	提高 SDK 三方库检测抗混淆能力
	Wang 等[56]	SDK 三方库	静态分析	首个基于模糊签名和分层相似性摘要的分析方法	解决 SDK 代码混淆场景下的三方库识别问题
	Zhan 等[57]	SDK 三方库	混合分析	提出了一个多维度的三方库检测性能评估框架	为优化库检测技术提供了实证依据和方向指导
基于 SDK 运行数据交互安全	Cabanas 等[22]	Facebook SDK	混合分析	结合库检测与接口动态监控以及流量捕获	发现三方库的隐私设置与数据搜集行为异常
	Chen 等[58]	SDK 三方库	混合分析	首个通过白名单过滤方式检测广告 SDK 数据搜集	基于特征向量匹配实现了抗混淆
	He 等[59]	SDK 三方库	动态分析	通过 Xposed 框架 Hook 隐私相关 API	实现实时的 Android 动态隐私泄露分析
	Wang 等[8]	SDK 三方库	混合分析	结合 NLP 进行代码行为与 TOS 联合分析	首次提出了跨库调用 XLDH 安全问题
	Zhang 等[38]	SDK 服务条款	混合分析	设计了一种隐私合规风险分析框架 PICOSCAN	揭示了当前生态中的 SDK 数据搜集违规模式
	Lu 等[16]	SDK 服务条款	混合分析	静态隔离架构与动态策略执行的深度协同	实现无需修改 SDK 代码的数据保护
	Hiroki 等[40]	SDK 服务条款	混合分析	结合 BERT 模型和动态污点分析	为隐私标签的准确性验证提供了新工具和数据集
	Meng 等[60]	SDK 三方库	静态分析	构建数据声明-实现行为模型,引入大模型文本解析	首次系统性评估了 158 个 SDK 安全性

4.1　SDK 内部代码组件安全

SDK 内部代码组件安全包括系统 SDK 安全与三方 SDK 安全。其风险不仅源于 SDK 自身 API 接口的兼容性缺陷,更涉及三方库安全隐患等。早期研究围绕人工分析或静态特征匹配,效率低下且覆盖范围有限。近年来,随着研究朝自动化分析发展,研究者创新性地引入差异分析、轻量化特征哈希、代码图依赖关系建模等方法,显著提升了检测精度与工业实用性,但面对高级混淆和多包结构变异时仍存在优化空间。因此,构建兼顾效率与鲁棒性的代码安全检测体系,成为当前研究的核心方向。

4.1.1　系统 SDK

Android SDK 的 API 在不同设备与版本中运行可能导致严重的兼容性问题。早期一些研究分别针对 API 调用和回调兼容性提出了解决方案,但由于分析效率低、误报率高或覆盖范围有限,这些方案难以在实际开发中广泛应用。近年来,研究开始探索更高效的兼容性检测方法,通过结合轻量化静态分析与 API 变更报告解析,显著提升了检测精度和效率。

Android 应用程序通过软件开发工具包 SDK 构建,其应用程序接口 API 使开发者能够调用设备功能与服务。然而,API 随 SDK 版本频繁演化,当应用安装设备的 API 级别与开

发者设定的目标 API 不匹配时，可能引发兼容性问题。早期研究如 CiD[49]率先针对 API 调用兼容性问题提出解决方案，通过分析 Android 框架源码构建 API 生命周期模型，并结合条件调用图定位应用中的高风险调用。然而，CiD 存在显著缺陷，其分析框架效率低下，仅支持 API 26 以下版本，且无法检测回调问题，误报率较高。同期出现的 CiDER 方法[50]将焦点转向 API 回调兼容性，通过手动构建协议不一致图捕捉 Activity 等 4 类核心组件的控制流变化，但受限于人工建模的覆盖范围，漏检率高达 75%，且无法处理非核心类回调。为突破这些局限，ACID[34,51]创新性地整合了 API 差异分析与轻量化静态分析，通过自动化解析 Google 官方发布的版本间 API 变更报告，精准锁定新增/删除的 API 方法，无需全量扫描框架代码，使检测效率提升 1.5 倍。同时，ACID 首次实现调用与回调问题的联合检测，覆盖 Android 全量 API 类，并通过追踪方法覆盖条件大幅降低了误报率，在 20 个基准应用中实现了较高的召回率和精度，较 CiD 和 CiDER 的 F1 值显著提升。这一技术演进标志着系统 SDK 安全从局部修补迈向系统性解决，且为应对 Android 生态安全提供了自动化、高精度的方案。

4.1.2 三方 SDK

Android 应用开发中三方库引发的资源冲突[25]问题长期被忽视，可能带来严重的安全隐患。由于 Android 资源编译器(Android Resource Compiler，ARC)的默认处理机制，如图 4 所示，不同库之间的同名资源可能被意外覆盖或合并，这为恶意攻击者提供了可乘之机。近年来，一些研究开始关注这一潜在威胁，并提出相应的防御方案，但现有方法在兼容性、性能开销或防护范围上仍存在局限[61]，如何在不影响开发效率的前提下实现可靠的资源隔离，成为亟待解决的关键问题。

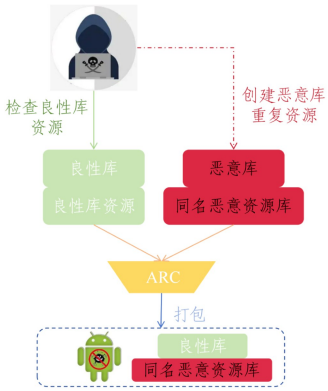

图 4 恶意库资源合并覆盖攻击过程

Fig 4 Process of duplicate resource mismediation

Wang 等[9]针对 Android 软件供应链中由三方库资源冲突引发的新攻击面展开研究，其首次发现 Android 资源编译器(ARC)在构建应用时，高优先级库的重复资源会覆盖或合并低优先级库资源，导致恶意库无需代码即可污染支付 SDK 的 CDN 地址、篡改网络证书配置等，引发数据泄露、中间人攻击等风险。对此，他们提出编译时资源隔离方案，通过为库资源添加唯一命名空间前缀并逆向修改代码引用，同时建立清单组件归属机制以阻断非所有者修改权限，开发者可通过白名单控制跨库访问。结果显示，部分三方库包含敏感资源，且挖掘了 428 个存在资源冲突风险的真实应用；提出的隔离方案使构建时间仅增加 1.46 s。但该方法依赖专家知识，敏感资源分类依赖人工审核，难以扩展到高度定制化或新兴资源类型；且版本覆盖不全，仅分析库的最新版本，未考虑历史版本中潜在的冲突或漏洞。

三方 SDK 的广泛使用在提升开发效率的同时，也带来了严峻的安全隐患。早期研究主要采用人工分析方法，针对特定 SDK 展开案例研究，揭示了包括数据泄露、协议漏洞在内的多种安全问题。然而，此类方法的规模化应用受限，加之人工分析存在固有局限，导致其覆盖范围较为有限。近年来，随着自动化分析技术的进步，研究者开始探索更高效的三方库检测方案，通过静态特征提取、动态行为分析等手段提升检测精度与效率。尽管现有工具在抗混淆能力和检测范围上已有显著改进，但在面对复杂代码结构或高级混淆技术时仍存在短板，亟需更鲁棒、可扩展的解决方案来应对日益复杂的软件供应链安全挑战。

Derr 等[44]针对 Android 应用中三方库长期未更新的安全问题展开研究。LibScout[62]已揭示 70%的库版本严重过时，且漏洞修复的采纳周期长达一年，但未深入分析根源。Derr 通过 3 项研究提出了解决方案，量化了库更新的可行性并提供了实践指导，但库开发者版本管理习惯难改变，自动化更新可能引入新风险，且部分库依赖[63]项或 API 级别限制阻碍了升级。

之后，研究者针对特定类型的 SDK 安全性问题展开了研究。Cai 等[46]针对 Android 生态系统中三方 SDK 的安全性问题展开研究，提出了一个结合静态与动态分析的通用框架：静态方面，通过反编译 SDK 的测试应用，利用污点追踪和控制流图分析数据流；动态方面，采用 Xposed 框架[45]注入 Hook 代码捕获反射调用，并通过 Fiddler 代理解析 HTTPS 流量。实验对应用中的 149 个常用 SDK 进行分析，发现超过半数的 SDK 存在漏洞，包括 HTTP 滥用、SSL/TLS 配置错误等[12]。然而，该方法的静态分析方式无法覆盖反射调用完整路径，且动态分析依赖模拟环境，可能遗漏真实场景情况。Liang 等[53]针对 Android 云备份 SDK(Dropbox、Google Drive、OneDrive、百度云)的代码安全漏洞及集成这 4 种 SDK 的应用展开研究，对比了四大 SDK 的协议差异，指出部分 SDK 易受攻击，所有 SDK 均未实现客户端加密，存在严重安全问题。Aniketh[52]创新性地提出混合分析框架，针对移动应用中无线扫描 SDK 展开系统性研究，首次大规模实证分析 52 个无线扫描 SDK 的行为模式，证实了现有平台政策执行与技术防御存在显著漏洞。然而，该研究仍受限于 Android 版本碎片化导致的检测盲区，且无法完全覆盖动态加载或混淆的 SDK 代码。

这些早期针对具体 SDK 的研究由于依赖人工分析的研究范围，受限于特定 SDK，难以规模化。目前随着工业化的发展，三方 SDK 开始逐渐转向自动化分析流程。

Ma 等[54]给出了一个 LibRadar 工具，旨在解决 Android 应用中三方库检测的准确性和效率问题。之前，AdRisk[64]，Centroid[58]等主要依赖白名单匹配包名，但这种方法覆盖率

低且无法应对混淆问题;采用聚类的方法虽能发现问题三方库,但依赖大规模应用分析,无法实时检测。LibRadar 采用了一种基于稳定 API 调用频率特征的解决方案,通过特征哈希和轻量级表示实现快速匹配。其关键点包括:从 100 万款应用中提取库的 API 特征并聚类生成指纹库;使用哈希和元特征构建抗混淆的库标识;通过多级聚类和包名推断技术提升检测鲁棒性。实验表明,LibRadar 可识别 29 279 个库,检测时间仅为秒级。其局限性在于预处理阶段耗时较长,仍需定期更新指纹库以适应新库版本。LibRadar 在速度与准确性上显著优于传统方法,但依赖包名或目录结构进行库检测,易受代码混淆和多包结构变异的影响,导致精度不够。为此,Li 等[55]针对 Android 生态中三方库检测的准确性和混淆鲁棒性问题展开研究,提出了 LibD,其核心方法包括:基于代码依赖关系而非包名划分库候选;特征哈希算法,通过哈希基本块操作码序列生成混淆无关的特征;动态阈值聚类,结合库实例出现频率判定真实库。实验在 142 万款 App 上验证,结果表明,LibD 精度较 LibRadar 提升 25%,且支持多包库和结构变体识别。然而,LibD 仍存在阈值依赖经验设定的问题,且对高级混淆的适应性有限,需结合反混淆技术进一步优化。之后,Wang 等[56]再次针对 Android 应用中三方库检测在代码混淆场景下的挑战展开研究,提出了 ORLIS 工具,其核心创新点在于融合了方法调用图与类层次结构特征,引入了全新相似性摘要技术,通过将代码段和调用图分别进行模糊签名和字符串特征提取,代替传统哈希特征,并通过两阶段匹配(应用库级筛选与类级映射)实现了混淆库的代码检测,同时大幅降低了计算开销。ORLIS 为混淆环境下的库检测提供了更可靠的解决方案。

为了分析这些三方库检测工具的性能,Zhan 等[57]提出了一个性能检测分析框架。该框架通过多维度实验评估三方库(TPL)性能,覆盖准确性、效率、鲁棒性和实用性。首先构建包含复杂包结构的数据集,对比 TPL 样本与真实样本的边界匹配度以验证解耦能力,并基于真实应用和开源应用构建涵盖 59 个 TPL 的 2115 个版本的数据集,通过精确度、召回率和 F1 值衡量识别能力,结果显示,多数工具精确度高但召回率低,主要原因是受库混淆影响较大。库混淆也是未来一段时间内在 SDK 安全性上待解决的重要问题。

4.2　SDK 运行数据交互安全

SDK 运行时与应用的交互数据会被 SDK 本体违法搜集或是遭遇外部违规窃取,这类安全风险正逐渐成为用户隐私安全与企业数据保护的关键挑战。现有研究表明,默认隐私设置不当、数据收集行为不透明等问题普遍存在,数据面临过度收集或滥用的风险。尽管已有工作通过流量分析、文本解析等方法揭示了部分问题,但对于 SDK 实际行为与隐私声明的一致性、跨库数据交互等深层次风险仍缺乏系统化解决方案。当前研究正探索结合自然语言处理等技术,以建立更全面的隐私风险评估框架,为构建可信的数据治理体系和安全移动生态提供技术支撑。

4.2.1　SDK 本体违法数据搜集

许多 SDK 本体存在隐蔽设计缺陷,通过预置数据采集接口、强制授权机制和提供模糊隐私声明等非法行为,系统化实施数据违规收集。这些 SDK 往往利用其底层框架特权,在未经用户同意的情况下,违法搜集数据。当前研究通过逆向分析、流量审计等技术发现了部分显性违规行为,这些隐患可能直接导致数据被过度收集或滥用。但在实际场景中,由于 SDK 行为的复杂性,现有解决方案仍存在检测精度不足、覆盖范围有限等瓶颈。如何实现更系统化的 SDK 违法数据搜集检测,成为当前移动安全领域亟待解决的关键问题。

在欧盟《通用数据保护条例》[30](GDPR)生效前夕,Cabanas等[22]通过实证研究揭示了 Facebook 对敏感数据的商业化利用问题。该研究团队开发了 FDVT[65]浏览器扩展工具,收集了 4577 名欧盟用户的广告偏好数据,并采用基于 spaCy 的语义分析技术结合 12 名专家的人工标注,从数万个广告标签中识别出上千个涉及种族、政治倾向、宗教信仰、健康状态和性取向的敏感标签。量化分析显示,七成的欧盟 Facebook 用户被标记至少一个敏感标签。该团队在 FDVT 工具中新增敏感标签提醒功能,并指出此类数据滥用可能导致风险。该研究为后续数据保护局对 Facebook 处以 120 万欧元罚款提供了关键证据,也为 GDPR[66]实施后的移动平台合规性监管提供了重要判例。

Inayoshi 等[40]针对 Android SDK 的标签指南页与实际数据收集行为不一致的问题展开研究。由于之前的工作主要关注隐私标签[67]的完整性和准确性,尚未系统分析 SDK 提供商发布的指南页的可用性和正确性,因此提出了一种半自动化系统,结合机器学习和动态污点分析来检测指南页声明与实际数据收集的差异。定义命名实体类型以统一提取指南页内容,动态分析 SDK 样本应用的数据泄露行为,通过匹配算法和情感分析检测不一致性。实验收集了 159 个样本,发现 8 个 SDK 存在位置或标识符相关的未声明数据收集问题,且超过一半的 SDK 未提供指南页。Meng 等[60]在之后构建“数据声明-实现行为”双维度评估模型,结合自动化污点追踪与语义分析技术,利用 LLM 解析 109 份隐私政策,揭示了 37%的 SDK 存在过度收集行为,88%存在虚假声明,暴露出 SDK 开发者普遍漠视 GDPR 等法规。该研究推动了 Google SDK 运行时隔离机制的改进,并为行业监管提供了量化依据。但采用静态分析方法的误报率较高,且 LLM 对于一些模糊文本分析效果较差。

Chen 等[58]初期通过白名单检测广告 SDK 的隐私数据泄露风险,采用 LibRadar 和 LibD 过滤风险库,然而这类基于特征向量匹配实现抗混淆的检测方法无法实现实时的三方库行为检测,且存在效率或精度不足的问题。因此,He 等[59]在研究数据泄露问题时提出了一种动态数据泄露分析工具,结合静态检测与动态 Xposed 框架,设计混合检测方法识别三方库,通过 Hook 技术监控隐私 API 调用链,区分宿主应用与 SDK 三方库的行为,且定义 4 类隐私泄露路径并量化风险等级。通过实验对 150 款流行应用进行分析,结果显示工具准确率达 97.4%,实现了实时的数据泄露检测与评估。

4.2.2　SDK 正常交互数据被违规搜集

SDK 在实现功能交互时产生的行为数据同样面临外部违规收集风险,其核心症结在于以下 3 方面:1)SDK 数据交互配置缺乏规范标准,这使得数据泄露的责任划分困难,尤其

是跨库数据搜集行为的责任方往往会被归咎于企业 SDK 对于数据的保护不足，而非三方 SDK 的实质违规搜集行为；2)社交类 SDK 在功能交互与数据保护间的矛盾未能有效解决，目前采用沙箱隔离的方式直接对一些可能存在违规行为的广告库进行限制，制约了 SDK 与应用的复杂功能的协同交互；3)对 SDK 提供商声明的可信度验证不足，一些 SDK 的数据搜集范围往往超越了其声明范围，给 SDK 提供商带来了严重的信任问题。近期的研究针对 SDK 合规性提出了统一数据安全分析框架与防御方式，并针对大量 SDK 进行了行为测试分析，为 SDK 设计模式的革新及未来监管机制的完善提供了关键技术支撑。

Wang 等[8]在研究 Facebook SDK[48]时最先发现 SDK 会遭受来自外部的数据违规搜集行为，即跨库调用 XLDH (Cross-library Data Harvesting)。如图 5 所示，该研究首次系统性地展示了 Android 平台上长期被忽视的跨库数据窃取 XLDH 问题(恶意三方库通过同一应用内其他 SDK 窃取用户敏感数据)。针对现有隐私分析工具无法解析 SDK 服务条款 ToS(Terms of Services)语义的局限，研究者提出了自动化检测工具 XFinder，其创新性地结合自然语言处理技术从 ToS 中提取数据共享策略，并通过静态分析追踪跨库反射调用及数据流。通过实验分析 130 万 Google Play 应用后，发现了 42 个 XLDH 库，影响了约 1.9 万应用，促使 Facebook 等平台采取下架措施。但该方法仍存在动态配置解析不足、复杂数据流检测困难等局限，且对 ToS 中模糊条款的覆盖有限，但为未来移动隐私保护研究指明了方向。

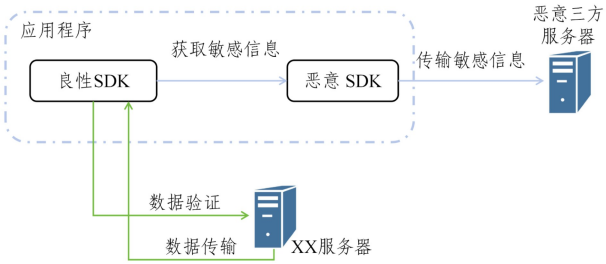

图 5 SDK 跨库攻击

Fig. 5 SDK cross-library attack

Zhang 等[38]研究了三方 SDK 数据交互合规配置中的风险问题。Irwin[68]等和 Du 等[69]主要通过流量分析或用户选择与数据收集的对比来评估数据交互合规性，但未能深入代码层面揭示根本原因。对此，Zhang 等[38]设计了一种自动化分析框架，通过静态与动态结合的方法检测应用和 SDK 在隐私 API 使用与实现中的 9 类违规模式，包括缺失配置、默认隐私侵犯、委托失效等。此外，构建了包含 65 个流行 SDK 的元数据库，设计了针对应用、SDK 和封装器的隐私原则，开发了基于流分析和运行时监控的多层次检测技术。实验表明，该方法在 48305 个 Google Play 应用中发现了部分存在数据泄露风险的应用，其中四成的 SDK 存在默认数据侵犯，未能正确进行配置。该项研究揭示了当前数据交互保护配置生态在标准化、工具支持和责任划分上的缺陷，为改进 SDK 设计和监管提供了实证依据；其局限性包括对混淆代码的覆盖不足、未分析服务器端配置等。

Lu 等[16]针对 Android 应用中三方 SDK 面临的跨库数据窃取(XLDH)问题展开研究。现有防护方法仅针对广告库隔离，无法处理社交 SDK 与宿主应用复杂的功能交互需求。为此，作者提出隐私保护社交 SDK 范式(PESP)，将社交 SDK 隔离在独立运行时，通过数据句柄避免数据泄露至不可信空间，引入敏感模块沙箱处理数据转换与 UI 展示，通过数据共享策略实现数据收集的可控和可审计性。实验表明，PESP 在保护 Facebook/Twitter SDK 数据的同时兼容现有功能，性能开销平均增加 39～74 ms。然而，该方法仍需解决应用服务器返回敏感数据的潜在泄露风险，且对其他类型 SDK 的普适性需进一步验证，但 PESP 仍为社交 SDK 提供了首个数据交互安全设计方案，为后续的数据交互安全发展提供了指引。

4.3 总结

4.3.1 SDK 安全分析方法分类总结

本节对现有 SDK 安全性分析方法进行了系统性的分类整理，如图 6 所示。同时，对表 2 中的现有研究方法进行了量化分析，从准确率、分析效率、抗混淆能力与分析局限性出发，对列举的工作进行了评估，结果如表 3 所列。

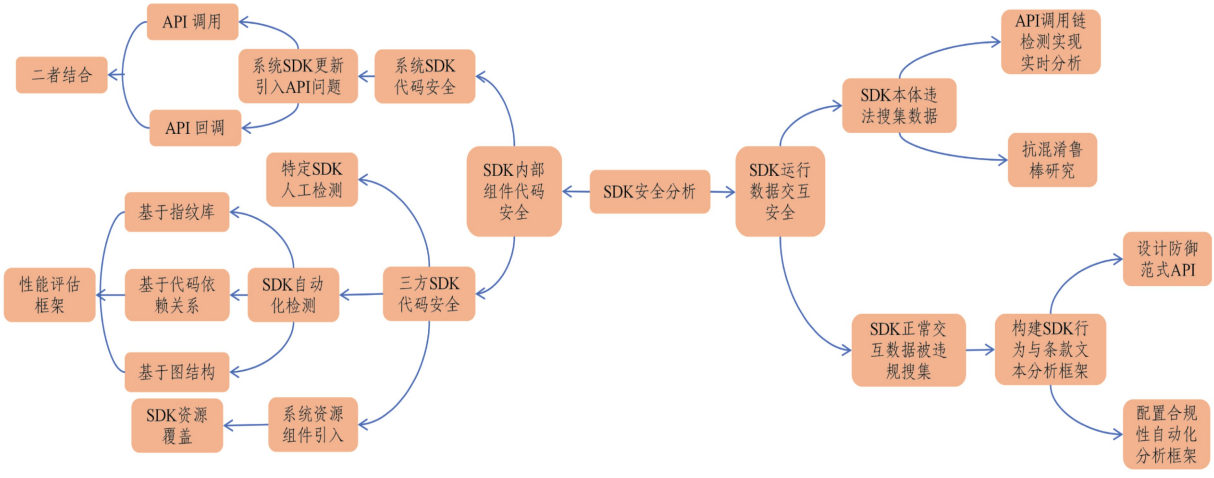

图 6 SDK 安全分析方法分类总结

Fig. 6 Summary of classification of SDK security analysis methods

表 3　现有分析方法效果评估

Table 3　Evaluation of existing analysis methods

方法	准确率	分析效率	抗混淆能力	局限性
Li 等[49]	高	未提及	低，依赖启发式方法，易导致误判	仅限 API 调用处理混淆代码时易出现漏报或误报
Huang 等[50]	高	未提及	未提及	仅限 API 回调，误报率高
Mahmud 等[34]	高	高	未提及	依赖 API 差异报告
Wang 等[9]	未提及，通过实际案例验证，未明确提及准确率	高	中，对名称加密、动态代码引用等抗混淆能力有限	依赖专家审核，泛化性不高
Derr 等[44]	高	高	高，对常见混淆技术抗混淆能力强	对于数据库覆盖要求较高
Aniketh 等[52]	未提及，仅分析了 52 个无线扫描 SDK	中	未提及	依赖 Android 版本环境，且无法覆盖动态加载 SDK
Liang 等[53]	未提及，仅分析 4 个 SDK	低	未提及	覆盖范围低，仅限文章分析的 4 个 SDK
Ma 等[54]	高	高	高，针对代码特征设计	依赖特征库的收集，可能遗漏小众库
Li 等[55]	高	高	高，基于代码依赖关系设计	缺乏对深度混淆技术的检测能力，且计算资源需求高
Wang 等[56]	高	中	高，方法融合了图与类层次结构特征	缺乏对深度混淆技术的检测能力
Zhan 等[57]	未提及，提出了一个三方库检测性能评估框架	未提及，提出了一个三方库检测性能评估框架	未提及，提出了一个三方库检测性能评估框架	未提及，提出了一个三方库检测性能评估框架
CABANAS 等[22]	高	高	未提及，采用替代名词与人工审核方式降低误判风险	仅分析了个别群体数据，结果可能具有偏差
Chen 等	中	中	中，仅能有效防御常见混淆方式	针对特定应用类型，缺乏对深度混淆技术的检测
He 等[59]	高	高	中，对混淆库具备一定抗性	缺乏数据流跟踪，可能遗漏间接隐私泄露
Wang 等[8]	高	低	未提及	NLP 对复杂语法处理能力有限，且训练依赖人工标注
Zhang 等[38]	高	中	低，难以抵抗代码混淆	检测方式规则依赖人工总结，可能遗漏
Lu 等[16]	未提及，采用实例证明方法有效性	中	未提及，本文仅设计了隔离机制保障数据安全交互	依赖 SDK 本身长期维护，仅能防止三方库数据泄露而不能保证本体违规
Meng 等[60]	未提及	高	未提及	仅采用静态分析，且大模型解析可信性一般
Hiroki 等[36]	高	高	未提及抗混淆措施	依赖官方文档数据和人工标注，覆盖率有限

在 SDK 内部组件代码安全研究上，目前的研究分别从系统 SDK 与三方 SDK 两点出发：在系统 SDK 研究上，现有研究分析了由系统更新引入的 SDK 中 API 调用回调等问题，此外，对于 Android 系统 SDK 组件的运作方式，提出了一种隐藏三方库资源覆盖攻击预测；在三方 SDK 研究上，现有研究从人工检测特定类型 SDK 逐渐向自动化类型检测发展，并基于指纹库、代码关系等特征提出了不同类型的检测方式，同时设计了性能评估框架。

在 SDK 运行数据交互安全研究上，目前的研究分别从 SDK 本体违法搜集数据与 SDK 正常交互数据被违规搜集两点出发：对于 SDK 本体违法搜集数据问题，现有研究对一些 SDK 服务条款的合规性进行了分析，提出了检测方式，并进行抗混淆性能加强，促使开发企业遵从各类隐私保护法；对 SDK 正常交互数据被违规搜集，分析了三方库对企业 SDK 数据的违规调用行为，研究者根据各类隐私泄露案例构建了 SDK 行为与条款的分析框架，并设计了防御范式 API 与自动化检测配置。

4.3.2　SDK 安全分析发展脉络

本节结合现有关于 Android SDK 安全性分析的研究内容与 Android SDK 的威胁演化情况，绘制了 Android SDK 安全性分析演化过程，如图 7 所示。

SDK 安全性分析的演化脉络呈现技术驱动与风险影响的双重推动，形成内部组件代码安全与外部运行数据交互安全两条核心演进路径，二者随移动应用生态复杂化而不断深化，共同构建了 SDK 安全研究的立体演化框架。

早期研究以人工审计与工具辅助检测为主要手段，聚焦 SDK 自身代码的基础安全性分析，并基于规则匹配检测第三方库与 API 回调兼容性问题，形成对 SDK 内部代码的安全性判断。随着 SDK 集成复杂度提升，研究逐步转向系统性安全评估：动态分析技术被引入以追踪敏感数据流转路径，兼容性测试框架从单一功能验证扩展为覆盖多版本 Android 系统适配性评估，构建更加抗混淆与鲁棒性的安全体系。近年来，研究视野进一步拓展至全生态级。针对 ARC 机制缺陷导致的覆盖漏洞等问题，提出覆盖 SDK 开发、集成、运行、更新全生命周期的治理方案，将安全治理从技术修复层面提升至供应链透明化与动态监测的体系化层面。

外部数据交互安全的演进早期受单一 SDK 安全事故（如 Facebook 数据泄露事件）的驱动，研究以动态监测工具为核心，通过运行时数据流追踪识别异常数据交互。随着跨库调用攻击（XLDH）的出现，研究范式转向结合规范与应用行为结合的高覆盖面防御系统研究：研究揭示了 Android SDK 在官方规范与实际数据收集行为间的实施偏差，提出包含数据流向审计的隐私合规风险分析框架；同时提出静态隔离与动态

策略执行相结合的防御架构，通过沙箱机制隔离跨库调用环境，设计更细粒度的权限控制策略，实现更优秀的防御保护。

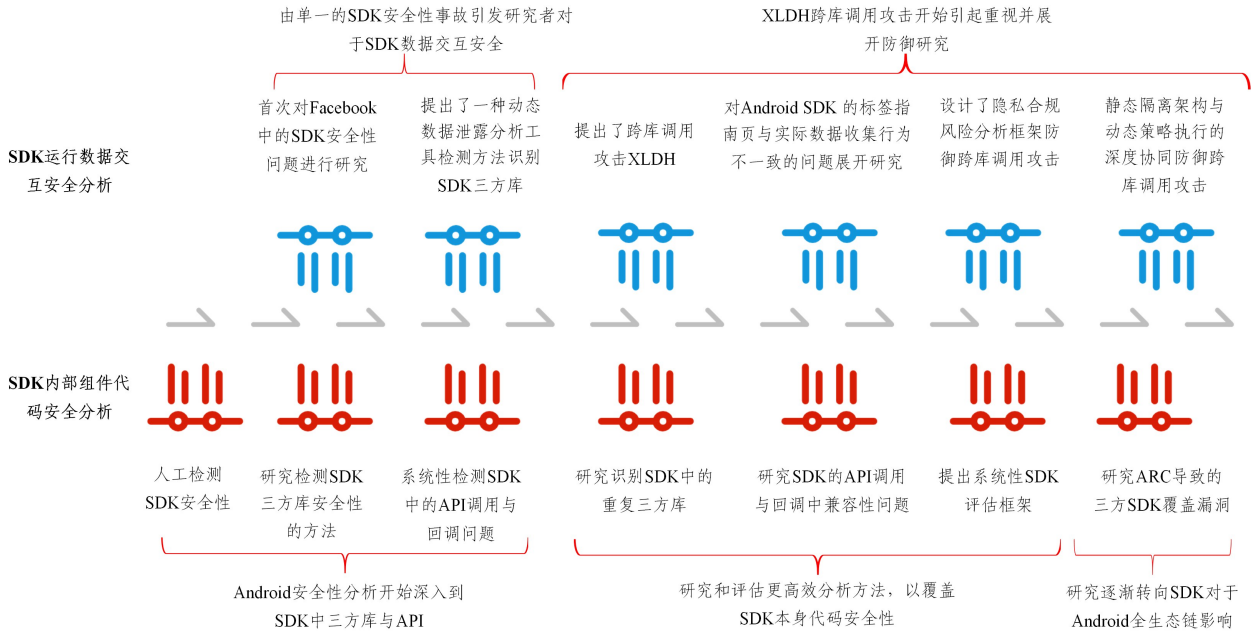

图 7 Android SDK 安全性分析演化过程

Fig. 7 Evolution process of Android SDK security analysis

5 现存挑战与未来趋势

当下 SDK 安全性研究面临着严峻挑战，现有检测模型的安全性分析覆盖面不全，难以应对新型攻击，在安全检测效率与用户体验[70]上进退维谷，且在安全责任划分上十分模糊。未来需结合新型技术，以 AI 大模型等建立 SDK 的全链路检测模型，覆盖 SDK 全生态链；同时构建结构化合规体系，并推动开发工具链革新，结合立法机构、标准组织与应用商店等多方资源协同发力[27,61-62,71]，以实现从单点防护到生态共治的根本性转变。

5.1 SDK 安全研究的深层挑战

在技术持续突破的背景下，当前研究仍存在亟待解决的系统性缺陷。首先，检测技术对新型攻击手段的适应性不足，例如现有工具多聚焦显式数据流追踪，却对隐式泄露束手无策。其次，现有防护方案往往顾此失彼，例如 PESP 方案虽实现了社交 SDK 的沙箱隔离，却导致功能交互延迟增加，在用户体验与安全之间难以平衡。更严峻的问题是，开发者对 SDK 的使用与 Android 安全生态治理的矛盾日益凸显。例如，某些 SDK 未提供可读的隐私指南页，导致 XFinder 等规范文本分析工具无法判断其合规性，使得存在隐私侵犯的 SDK 提供商逃避追责，而宿主应用却要承担几乎全部的法律风险。

这些缺陷深刻揭示了移动生态安全治理的深层矛盾，即技术创新与落地实践的脱节，亟需构建覆盖 SDK 全生命周期的可信验证体系。

5.2 SDK 安全的未来方向展望

面向未来，SDK 安全研究需要突破技术工具优化的传统路径，向跨学科协同治理的方向演进。

在检测技术层面，现有 SDK 检测技术覆盖不全，动态监测依赖沙箱环境，合规性校验效率较低。未来可探索大语言模型 LLM[72]与静态代码分析引擎、动态行为监控工具的深度融合方案，构建“代码语义理解-行为模式识别-合规规则匹配”的三级检测架构，提升 SDK 分析覆盖率，同时开发法规条款动态映射工具，基于 LLM 技术自动提取 SDK 隐私声明与 GDPR、《个人信息保护法》等国内外法规，缩短目前的合规校验时间，降低误判率。

在标准建设方面，当前 SDK 安全标准存在规范分散、执行模糊、责任边界不清等问题。展望未来，可以设计 SDK 隐私声明的“结构化标签体系”[38]，制定标签与技术实现的一致性验证标准，使隐私声明与实际行为的匹配准确率提升，并结合 SDK 跨库数据泄露风险等特征，构建 SDK 漏洞库的“统一合规性审计框架”，并要求 SDK 提交漏洞修复情况报告。

现有安全防护集中于运维事后检测，而 SDK 安全风险源于开发阶段的合规意识缺失，需要建立应用开发全周期防护。例如，在 Android Studio 中集成实时检测插件[73]，基于静态代码分析与 API 调用监控技术，在编码阶段实时预警“非必要权限申请”“敏感数据明文传输”等风险行为，将违规行为预先拦截，将安全防线从运维前移至开发端。

这些改变不仅需要技术创新，更依赖立法机构、标准组织与应用商店的协同推进，构建多方参与治理的环境，才能破解当前 SDK 安全治理的困局。

结束语 目前，三方 SDK 的覆盖率已提升至 80%以上，攻击面不断扩张，威胁形态日益复杂，SDK 安全性研究是连接系统和跨平台[67,74-75]开发的关键纽带。面对 SDK 漏洞可能引发的链式反应或预装后门的隐蔽攻击，仅靠应用分析级别的防御已力不从心。本文对近年来 SDK 的安全性研究进行了详细的梳理，在 SDK 内部组件代码安全方面，从 SDK 的三方库、API 等代码分析层面入手，总结了有关检测方法以及

发展过程；在 SDK 外部数据交互方面，从近年来出现的新型 SDK 攻击方式入手，对其检测以及防御框架进行了详细说明。未来，需要构建覆盖开发、部署、运行全生命周期的 SDK 安全治理体系，从源头把控风险，方能守护 Android 生态安全，实现开源与安全的平衡。

参 考 文 献

[1] KUMAR N. Android Usage Statistics (2025)-Global Market Share [EB/OL]. [2025-09-10]. https://www.demandsage.com/android-statistics/.

[2] iJiami. HOT! iJiami Releases the National Mobile Application SDK Market Share Analysis Report [EB/OL]. [2019-06-06]. https://www.ijiami.cn/new

[3] iJiami. SDK Security Monitoring Report: How Should We Strengthen Prevention? [EB/OL]. [2023-03-29]. https://www.ijiami.cn/newsInfo?id=1336.

[4] MA J. Research on the Detection of Security Vulnerabilities in External SDKs of the Android System [J]. Information Technology and Network Security, 2019, 38(8): 6-12.

[5] XIA X W, QIAN C, LIU B, et al. Android Security Overview: A Systematic Survey[C]// Proceedings of the 2nd IEEE International Conference on Computer and Communications (ICCC), IEEE, 2016.

[6] SARKAR A, GOYAL A, HICKS D, et al. Android Application Development: A Brief Overview of Android Platforms and Evolution of Security Systems[C]// Proceedings of the 2019 Third International conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud). 2019.

[7] QIU J, YANG X W, WU H M, et al. LibCapsule: Complete Confinement of Third-Party Libraries in Android Applications [J]. IEEE Transactions on Dependable and Secure Computing, 2022, 19(5): 2873-2889.

[8] WANG J C, XIAO Y, WANG X Q, et al. Understanding Malicious Cross-library Data Harvesting on Android[C]// Proceedings of the 30th USENIX Security Symposium. 2021.

[9] WANG X Q, ZHANG Y F, WANG X F, et al. Union under Duress: Understanding Hazards of Duplicate Resource Mismediation in Android Software Supply Chain[C]// Proceedings of the 32nd USENIX Security Symposium. 2023.

[10] MA K, GUO S Q. Security analysis of third-party SDKs in the Android ecosystem [J]. Journal of Software, 2018, 29(5): 1379-91.

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

[12] FAHL S, HARBACH M, MUDERS T, et al. Why eve and mallory love android: an analysis of android SSL (in) security [C]// Proceedings of the 2012 ACM Conference on Computer and Communications Security. 2012: 50-61.

[13] ZHANG J, LI R X, TANG J W, et al. Detection of collusion behaviors in Android third-party libraries [J]. Computer Science, 2019, 46(5): 83-91.

[14] DUAN R, BIJLANI A, XU M, et al. Identifying Open-Source License Violation and 1-day Security Risk at Large Scale[C]// Proceedings of the 24th ACM-SIGSAC Conference on Computer and Communications Security. 2017.

[15] ANDOW B, MAHMUD S Y, WANG W Y, et al. PolicyLint: Investigating Internal Privacy Policy Contradictions on Google Play[C]// Proceedings of the 28th USENIX Security Symposium. 2019.

[16] LU H R, LIU Y C, LIAO X J, et al. Towards Privacy-Preserving Social-Media SDKs on Android[C]// Proceedings of the 33rd USENIX Security Symposium. 2024.

[17] GOOGLE. SDK tools guides[EB/OL]. [2025-05-06]. https://developer.android.com/tools.

[18] MA K. Research on Privacy Leakage and Security of Third-party SDKs in the Android Ecosystem [D]. Jinan: Shandong University, 2018.

[19] GAO P. Research on Detection Techniques for Android Third-party Libraries [D]. Wuhan: Wuhan University of Science and Technology, 2023.

[20] SUZANNA, SASMOKO, GAOL F L, et al. Augmented Reality SDK Overview for General Application Use [J]. International Journal of Advanced Computer Science and Applications, 2023, 14(11): 54-60.

[21] MAHMUD S Y, ENGLISH K V, THORN S, et al. Analysis of Payment Service Provider SDKs in Android[C]// Proceedings of the 38th Annual Computer Security Applications Conference. 2022.

[22] CABAÑAS J G, CUEVAS A, CUEVAS R, et al. Unveiling and Quantifying Facebook Exploitation of Sensitive Personal Data for Advertising Purposes[C]// Proceedings of the 27th USENIX Security Symposium. 2018.

[23] ZHANG Y. MVC Algorithm Design of Smart Mobile Marketing Micro-Classroom System based on Android SDK Technology [C]// Proceedings of the 2022 International Conference on Sustainable Computing and Data Communication Systems. 2022.

[24] ZHAN X, LIU T M, FAN L L, et al. Research on Third-Party Libraries in Android Apps: A Taxonomy and Systematic Literature Review [J]. IEEE Transactions on Software Engineering, 2022, 48(10): 4181-4213.

[25] WANG Y, WEN M, LIU Z W, et al. Do the Dependency Conflicts in My Project Matter? [C]// Proceedings of the 26th ACM Joint Meeting on European Software Engineering Conference (ESEC)/Symposium on the Foundations of Software Engineering (FSE). 2018.

[26] ZHAN X, FAN L L, CHEN S, et al. ATVHUNTER: Reliable Version Detection of Third-Party Libraries for Vulnerability Identification in Android Applications[C]// Proceedings of the 43rd IEEE/ACM International Conference on Software Engineering-Software Engineering in Practice (ICSE-SEIP)/43rd ACM/IEEE International Conference on Software Engineering-New Ideas and Emerging Results. 2021.

[27] KHANDELWAL R, NAYAK A, CHUNG P, et al. The Overview of Privacy Labels and their Compatibility with Privacy Policies [J]. arXiv: 2303.08213, 2023.

[28] WIKIPEDIA. Terms of service [EB/OL]. [2025-06-25]. ht-

tps://en. wikipedia. org/wiki/Terms_of_service.

[29] KHANDELWAL R, NAYAK A, CHUNG P, et al. Unpacking Privacy Labels: A Measurement and Developer Perspective on Google's Data Safety Section[C]// Proceedings of the 33rd USENIX Security Symposium. 2024.

[30] GDPR. General Data Protection Regulation[EB/OL]. [2016-04-27]. https://gdpr-info. eu/.

[31] CCPA. California Consumer Privacy Act of 2018[EB/OL]. https://www. oag. ca. gov.

[32] ANDOW B, MAHMUD S Y, WHITAKER J, et al. Actions Speak Louder than Words: Entity-Sensitive Privacy Policy and Data Flow Analysis with POLICHECK[C]// Proceedings of the 29th USENIX Security Symposium. 2020.

[33] GUARDIAN T. Revealed: 50 million Facebook profiles harvested for Cambridge Analytica in major datareach[EB/OL]. https://www. theguardian. com/news/2018/mar/17/cambridge-analytica-facebook-influence-us-election.

[34] MAHMUD T, CHE M R, YANG G W, et al. Android Compatibility Issue Detection Using API Differences[C]// Proceedings of the 28th IEEE International Conference on Software Analysis, Evolution and Reengineering. 2021.

[35] HUASONG MENG M, YAN C, HAO Y, et al. A Large-Scale Privacy Assessment of Android Third-Party SDKs [J]. arXiv: 2409. 10411, 2024.

[36] CHEN S, ZHANG Y, FAN L, et al. AUSERA: Automated Security Vulnerability Detection for Android Apps [C]// Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering. 2023.

[37] DIAO W R, LIU X Y, LI Z, et al. No Pardon for the Interruption: New Inference Attacks on Android Through Interrupt Timing Analysis[C]// Proceedings of the IEEE Symposium on Security and Privacy. 2016.

[38] ZHANG Y F, HU Z J, WANG X Q, et al. Navigating the Privacy Compliance Maze: Understanding Risks with Privacy-Configurable Mobile SDKs[C]// Proceedings of the 33rd USENIX Security Symposium. 2024.

[39] LIU B, LIU B, JIN H, et al. Efficient Privilege De-Escalation for Ad Libraries in Mobile Apps [C]// Proceedings of the 13th Annual International Conference on Mobile Systems, Applications, and Services. 2015: 89-103.

[40] INAYOSHI H, KAKEI S, SAITO S, et al. Detection of Inconsistencies between Guidance Pages and Actual Data Collection of Third-party SDKs in Android Apps[C]// Proceedings of the IEEE/ACM 11th International Conference on Mobile Software Engineering and Systems. 2024.

[41] DING X H, ZHANG L L, ZHAO K, et al. A privacy leakage detection method combining static and dynamic features [J]. Journal of Computer Science and Technology, 2023, 50(10): 327-335.

[42] LI R Y. Research on Vulnerability Detection Technology of Android Third-party SDKs Based on Machine Learning [D]. Beijing: Beijing University of Posts and Telecommunications, 2019.

[43] YUAN J F, LI H X, YOU W, et al. Location of Third-Party Library Functions in Obfuscated Applications [J]. Journal of Computer Science and Technology, 2023, 50(7): 293-301.

[44] DERR E, BUGIEL S, FAHL S, et al. Keep me Updated: An Empirical Study of Third-Party Library Updatability on Android [C]// Proceedings of the 24th ACM-SIGSAC Conference on Computer and Communications Security. 2017.

[45] ROVO89. Xposed[EB/OL]. http://xposed. cc.

[46] CAI Y B. Static and dynamic analysis of the security of third-party SDKs in the Android ecosystem [J]. Microcomputer Applications, 2021, 37(6): 55-57.

[47] YANG S, CHEN S, FAN L L, et al. Compatibility Issue Detection for Android Apps Based on Path-Sensitive Semantic Analysis[C]// Proceedings of the 45th IEEE/ACM International Conference on Software Engineering. 2023.

[48] RODRIGUEZ D, CALANDRINO J A, DEL ALAMO J M, et al. Privacy Settings of Third-Party Libraries in Android Apps: A Study of Facebook SDKs [EB/OL]. https://plaintext-response. com/static/papers/pets2025-rodriguez. pdf.

[49] LI L, BISSYANDÉ T F, WANG H Y, et al. CiD: Automating the Detection of API-Related Compatibility Issues in Android Apps[C]// Proceedings of the 27th ACM SIGSOFT International Symposium on Software Testing and Analysis. 2018.

[50] HUANG H X, WEI L L, LIU Y P, et al. Understanding and Detecting Callback Compatibility Issues for Android Applications [C]// Proceedings of the 33rd IEEE/ACM International Conference on Automated Software Engineering. 2018.

[51] MAHMUD T, CHE M, YANG G. Detecting Android API Compatibility Issues With API Differences [J]. IEEE Transactions on Software Engineering, 2023, 49(7): 3857-3871.

[52] GIRISH A, REARDON J, TAPIADOR J, et al. Your Signal, Their Data: An Empirical Privacy Analysis of Wireless-scanning SDKs in Android [J]. arXiv. 2503. 15238, 2025.

[53] LIANG J, LIU W, HAN W L, et al. Analysis of code security issues in the Android cloud backup module [J]. Journal of Network and Information Security, 2017, 3(1): 68-78.

[54] MA Z, WANG H Y, GUO Y, et al. LibRadar: Fast and Accurate Detection of Third-party Libraries in Android Apps[C]// Proceedings of the 38th IEEE/ACM International Conference on Software Engineering Companion. 2016.

[55] LI M H, WANG W, WANG P, et al. LibD: Scalable and Precise Third-party Library Detection in Android Markets[C]// Proceedings of the 39th IEEE/ACM International Conference on Software Engineering. 2017.

[56] WANG Y, WU H W, ZHANG H L, et al. ORLIS: Obfuscation-Resilient Library Detection for Android[C]// Proceedings of the 5th ACM/IEEE International Conference on Mobile Software Engineering and Systems. 2018.

[57] ZHAN X, LIU T M, LIU Y P, et al. A Systematic Assessment on Android Third-Party Library Detection Tools [J]. IEEE Transactions on Software Engineering, 2022, 48(11): 4249-4273.

[58] CHEN K, LIU P, ZHANG Y J. Achieving Accuracy and Scalability Simultaneously in Detecting Application Clones on An-

droid Markets[C]// Proceedings of the 36th International Conference on Software Engineering. 2014.

[59] HE Y Z, HU B H, HAN Z, et al. Dynamic Privacy Leakage Analysis of Android Third-party Libraries[C]// Proceedings of the 1st International Conference on Data Intelligence and Security. 2018.

[60] MENG M H, YAN C, ZHANG Q, et al. Assessing Privacy Compliance of Android Third-Party SDKs [J]. arXiv: 2409. 10411, 2024.

[61] HEUSER S, NADKARNI A, ENCK W, et al. ASM: A Programmable Interface for Extending Android Security[C]// Proceedings of the 23rd USENIX Security Symposium. 2014.

[62] BACKES M, BUGIEL S, DERR E, et al. Reliable Third-Party Library Detection in Android and its Security Applications[C]// Proceedings of the 23rd ACM Conference on Computer and Communications Security. 2016.

[63] BASET S A, LI S W, SUTER P, et al. Identifying Android Library Dependencies in the Presence of Code Obfuscation and Minimization[C]// Proceedings of the IEEE/ACM 39th International Conference on Software Engineering Companion. IEEE, 2017.

[64] GRACE M C, ZHOU W, JIANG X, et al. Unsafe exposure analysis of mobile in-app advertisements [C]// Proceedings of the fifth ACM conference on Security and Privacy in Wireless and Mobile Networks. 2012: 101-112.

[65] CABAÑAS J G, CUEVAS A, CUEVAS R, et al. FDVT: Data Valuation Tool for Facebook Users[C]// Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems. 2017.

[66] JIN G Z, LIU Z, WAGMAN L. The GDPR and SDK Usage In Android Mobile Apps [EB/OL]. https://www. nber. org/system/files/working_papers/w33099/w33099. pdf.

[67] BALASH D G, ALI M M, KODWANI M, et al. Poster: Longitudinal Measurement of the Adoption Dynamics in Apple's Privacy Label Ecosystem[C]// Proceedings of the 30th ACM SIGSAC Conference on Computer and Communications Security. 2023.

[68] IRWIN R, PRIMAL W, JOEL R, et al. "Won't Somebody Think of the Children?" Examining COPPA Compliance at Scale [C]// Proceedings of the 8th Privacy Enhancing Technologies Symposium. 2018.

[69] DU X L, YANG Z M, LIN J P, et al. Withdrawing is believing? Detecting Inconsistencies between Withdrawal Choices and Third-party Data Collections in Mobile Apps[C]// Proceedings of the 45th IEEE Symposium on Security and Privacy. 2024.

[70] LU D B, CUI H L, ZHANG W, et al. An application security reinforcement scheme based on Intent filtering [J]. Information Network Security, 2017(11): 67-73.

[71] TANG W, LUO P, FU J L, et al. LibDX: A Cross-Platform and Accurate System to Detect Third-Party Libraries in Binary Code [C]// Proceedings of the 27th IEEE International Conference on Software Analysis, Evolution, and Reengineering. 2020.

[72] YANG Y, WANG X, ZHAO C L, et al. Survey on automated testing of Android graphical user interfaces [J]. Journal of Computer Science and Technology, 2022, 49(S2): 756-765.

[73] GUO J, FU X, LI L, et al. Characterizing Installation- and Run-Time Compatibility Issues in Android Benign Apps and Malware [EB/OL]. https://dl. acm. org/doi/pdf/10. 1145/3725810.

[74] GARDNER J, FENG Y Y, REIMAN K, et al. Helping Mobile Application Developers Create Accurate Privacy Labels[C]// Proceedings of the 7th IEEE European Symposium on Security and Privacy. 2022.

[75] LI T S, REIMAN K, AGARWAL Y, et al. Understanding Challenges for Developers to Create Accurate Privacy Nutrition Labels[C]// Proceedings of the CHI Conference on Human Factors in Computing Systems. 2022.

XU Teng, born in 1994, Ph.D candidate. His main research interests include malware detection and machine learning.

LI Heng, born in 1995, Ph.D, is a member of CCF(No. X8790M). His main research interests include malware detection and adversarial sample attack and defense.

（责任编辑：何杨）