腾讯安全 | 科恩实验室 KEEN SECURITY LAB

腾讯安全科恩实验室

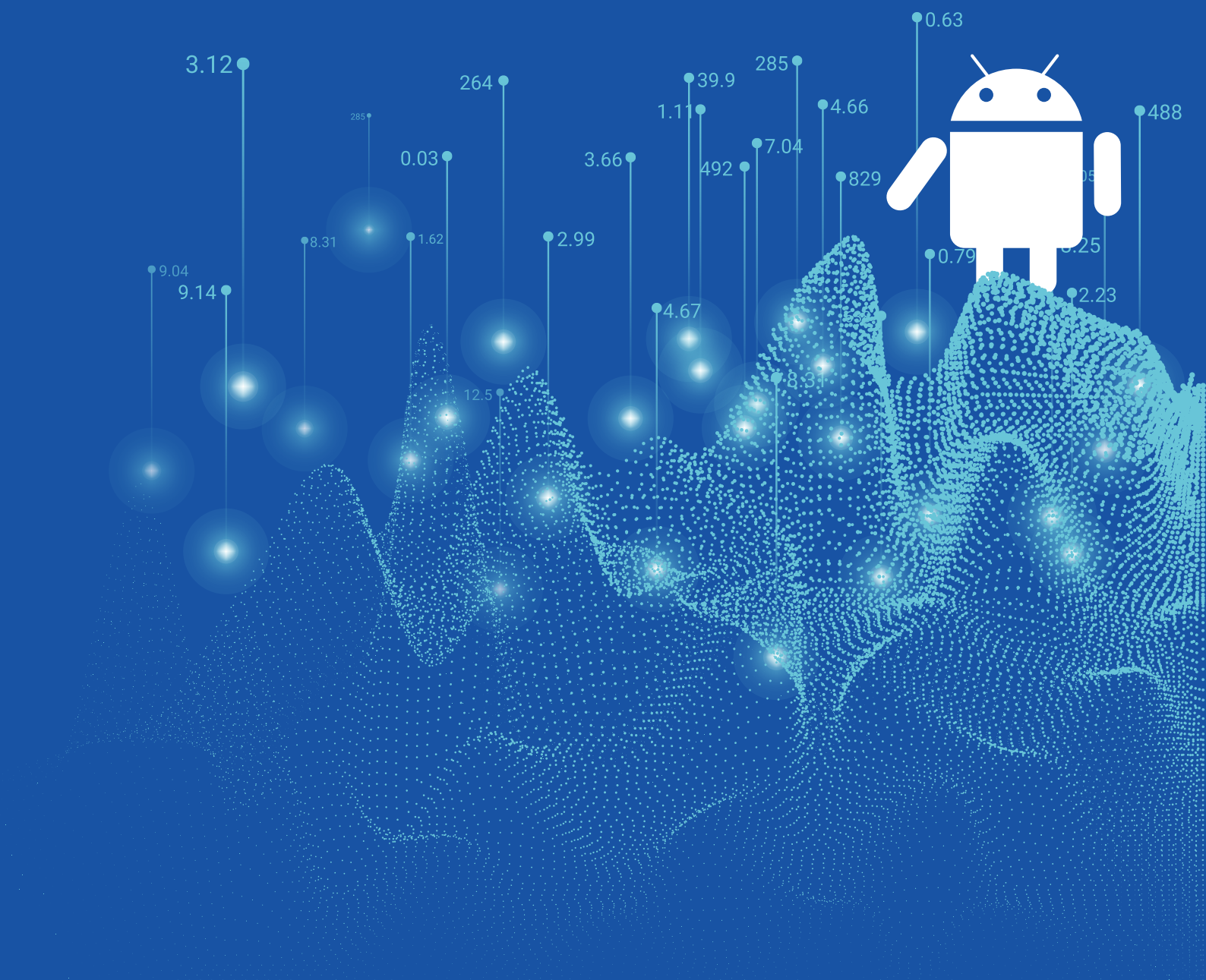
2018年Android应用安全白皮书

科恩实验室
KEEN SECURITY LAB

信息安全顶级服务

护航产业数字化变革 守护全网用户信息安全

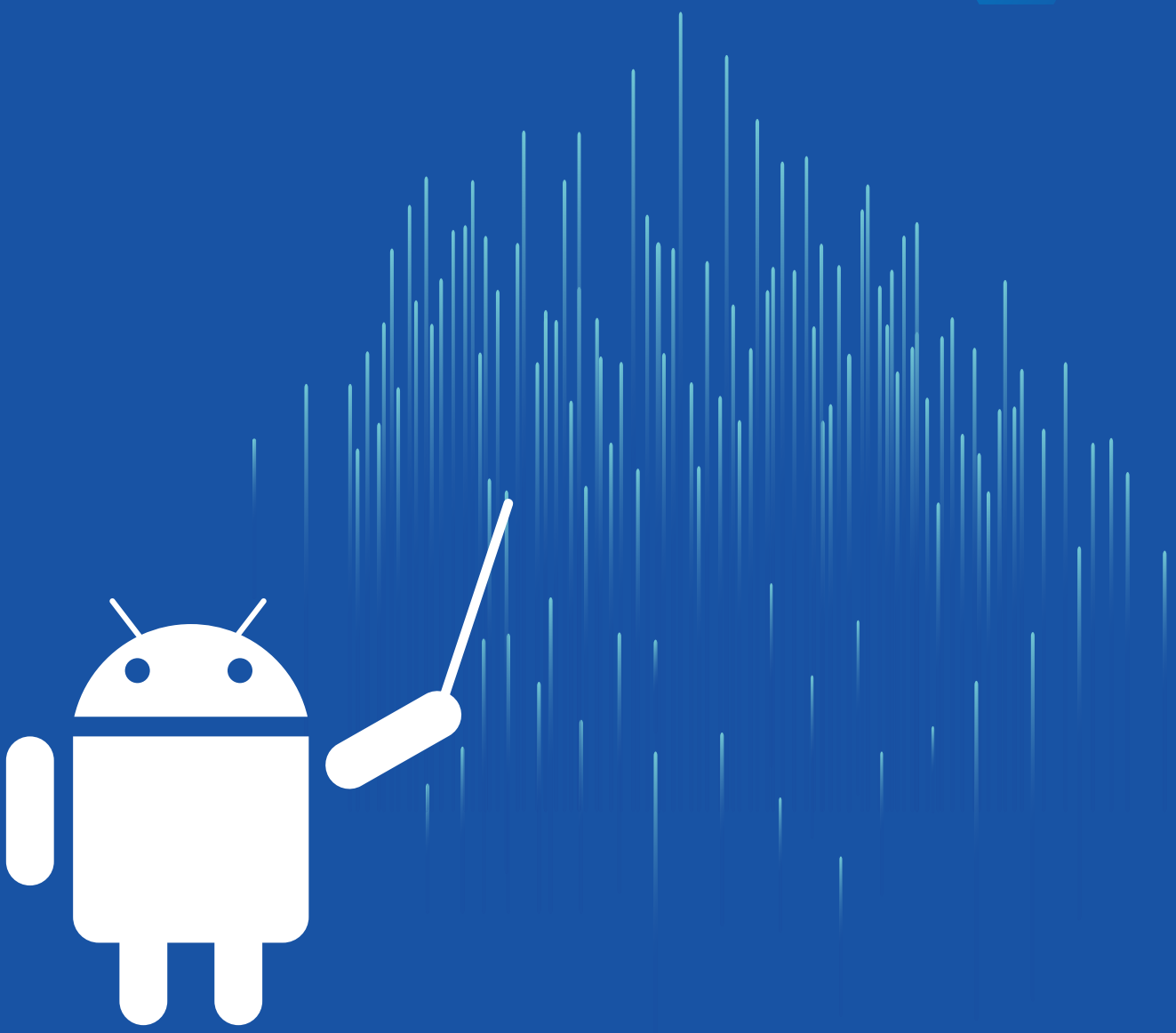
科恩实验室
KEEN
SECURITY LAB



Contents
目录

科恩实验室
KEEN
SECURITY LAB

Android应用行业概览



腾讯安全科恩实验室
2018年Android应用安全白皮书

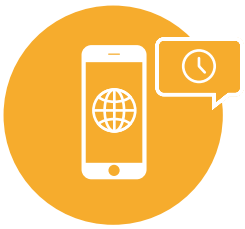
Keen Security Lab
2018 Android Application Security White Paper

Android移动应用蓬勃发展

根据最新全球数字报告数据[1]，目前全球有 51.1 亿独立移动用户，占全球人口总数的 2/3。2018 年用户平均每天花费在移动设备上的时间达到 3 个小时，移动应用市场蓬勃发展并不令人意外。

5.112
BILLION

67%
NUMBER



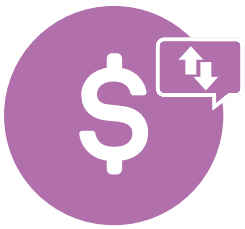
3
HOURS

2018 年全球 App 下载量超过 1940 亿次，年增速超过 9%。移动用户还在付费下载、App 内购买和 App 内订阅等模式下进行消费。这类服务在 2018 年收入超过 1010 亿美元[2]（包含 iOS 数据），增速到达了 23%，智能手机用户在单个移动应用上平均花费 20.15 美元。

194
BILLION

+9%
NUMBER

$101
BILLION

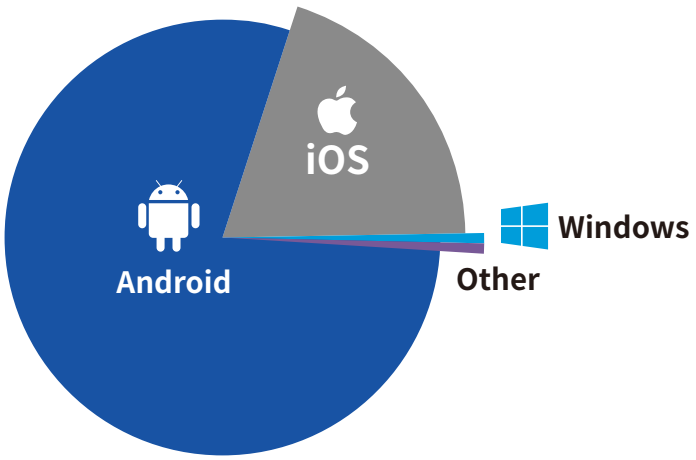


23%
SPEED UP

$20.15
SPEND

中国移动互联网的发展也在全球排名前列。截至 2018 年 12 月，我国手机网民规模达 8.17 亿，全年新增手机网民 6433 万。智能手机操作系统市场份额 [3] 显示在我国 Android 系统占比已经超过 80%。

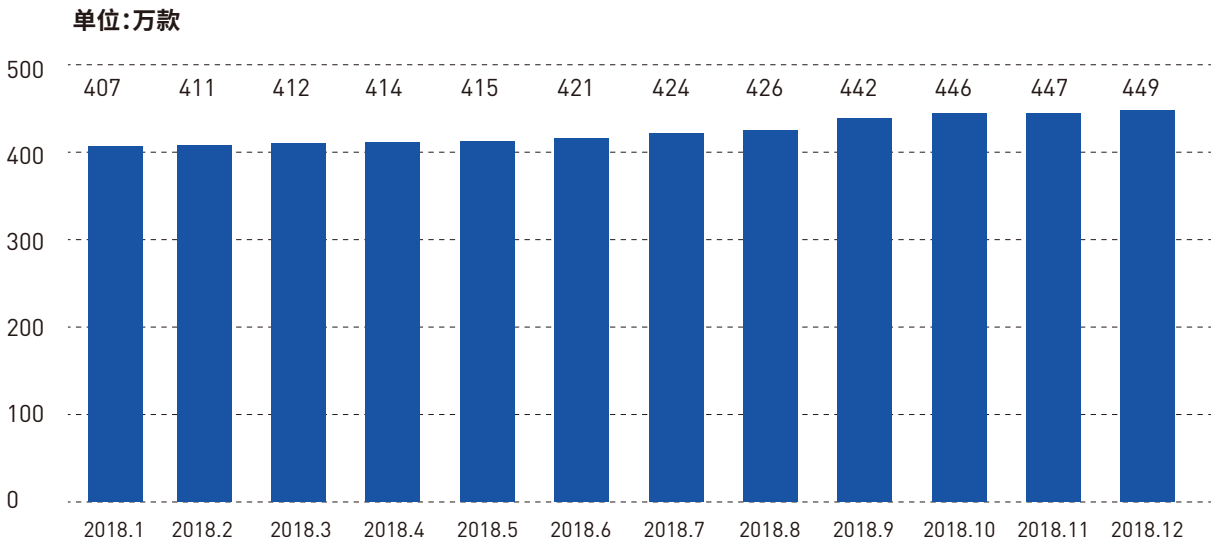
我国智能手机系统占比统计图



Android	Android	80.1%
iOS	iOS	19.7%
Windows	Windows	0.1%
Blackberry	Blackberry	0%
其它	Other	0.1%

截至 2018 年 12 月，我国市场上监测到的在架 App 数量为 449 万款 [4]。中国移动应用下载量在 2018 年总下载量占比将近 50%，是目前全球移动应用下载量最大的国家 [2]。

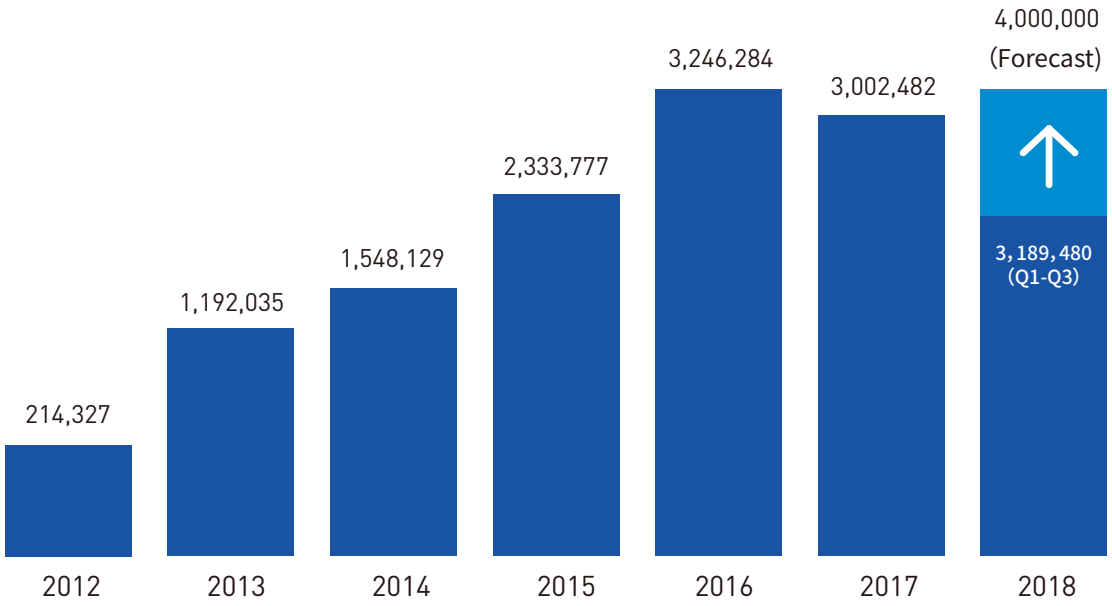
2018年移动应用在架数量统计图



Android应用安全亟待加强

Android 系统到 2018 年已经有 10 年的历史，系统进驻超过 20 多亿台设备，用户在谷歌应用商店可以访问的应用数量也达到三百万个以上。但在系统本身迅速发展的过程中，Android 系统上的安全问题一直备受瞩目。数据显示，从 2012 年到 2018 年，Android 平台的恶意程序数量增长迅猛。截至 2018 年第三季度末，G DATA 统计数据 [5] 显示在 Android 系统发现超过 320 万个新的恶意样本，平均每天发现超过 11000 个新的恶意软件样本。

新增Android恶意样本数量统计图(截止到2018年第三季度)
New Android malware samples
(per year)



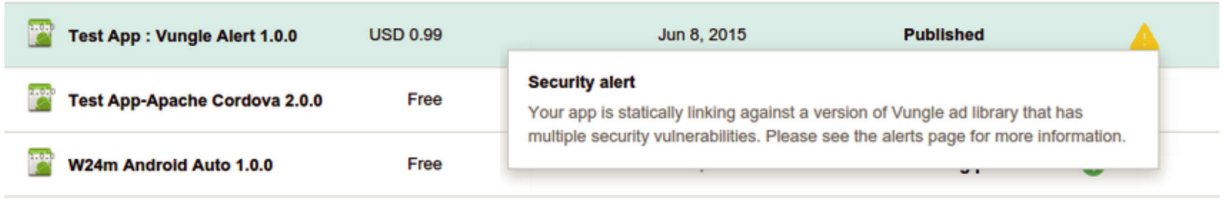
国家互联网应急中心表示，我国移动互联网安全亟待加强。2017 年，通过国家互联网应急中心自主捕获和厂商交换获得的移动互联网恶意程序数量达 253 万余个，同比增长 23.4%。其中，排名前三的恶意程序类型分别为流氓行为类、恶意扣费类和资费消耗类。

同时，由于开源组件自身安全存在诸多问题，加上当前应用程序普遍基于开源组件进行开发，使得应用开发过程中会产生大量安全漏洞严重影响安全性。例如，Insignary 公司在检测 Google Play 商店最受欢迎的 Android 应用程序后发现 [7]，约有 20%包含具有已知安全漏洞的开源组件，这些漏洞可能被黑客利用。尽管 Android 移动应用程序的开发在功能上越来越强大，但在开发阶段产生的安全问题却成为制约 App 发展的重要因素。

安全投入现状仍无法满足安全需求

在 Android 应用安全性提升方面，谷歌一直充分重视并投入大量成本推进 [8]。从 2015 年开始，谷歌启动应用安全改进计划（App Security Improvement），旨在提高应用的安全性。应用开发者的应用程序在 Google Play 商店上架前需要进行安全扫描，扫描出的安全问题需要按照开发安全提示和建议进行修复。迄今为止，该计划已帮助开发人员在 Google Play 上修复了超过一百万个应用。除此之外，Google Play 商店已经上架的应用也会不断被重新扫描，以应对新出现的安全漏洞以及其他威胁。通过持续改进现有检查安全漏洞，应用安全改进计划仍在不断更新检测内容。

应用程序安全性改进警报示例



Java 语言编写 Android 应用程序打包成 Apk 文件后，使用反编译工具可以进行逆向分析，容易暴露风险。添加恶意代码后二次重打包等手段可以进一步破坏原应用的安全性。为了更好地对抗这些风险，App 技术加固技术也不断地发展。实现形式从最开始将 dex 文件动态加载、内存不落地加载，到函数级别的指令抽取、指令转换，发展到 VMP 虚拟机保护以及多种方式结合使用。各种加固方案或多或少都有优缺点，具备加固解决方案和能力的厂商也在不断增加。但与此同时，应用安全防护措施整体实施情况并不乐观。

2018年Android应用安全问题频发

2018 年初，安卓 App 被曝存在应用克隆风险，利用多点耦合产生的漏洞，可以实现“克隆”用户账户、窃取隐私信息，甚至进一步实现盗取账号及资金。

同月，三星安卓浏览器被爆严重漏洞，攻击者可以通过利用高危同源策略绕过漏洞开启网站服务，如果用户访问攻击者控制的站点，攻击者就可以从浏览器中的不同标签页窃取用户数据。该漏洞相关受影响设备多达 3 亿。

年底，上海警方成功捣毁一个利用网上银行漏洞非法获利的犯罪团伙。该团伙利用银行 App 安全漏洞，使用黑客技术软件成倍放大定期存单金额，从中非法获利 2800 余万元。随着数字化金融的迅猛发展，类似业务漏洞一旦被不法分子发现并加以利用，造成的经济损失十分惨重。

造成严重安全问题的并不一定是通过新颖的漏洞利用形式，还可以是非常简单、“历史悠久”的漏洞模式，但这类漏洞却往往容易被大多数开发者忽视，例如 WebView 组件所带来的安全问题。

Android 移动应用安全需要自上而下整个产业闭环的各个角色共同维护和提高，不应是开发厂商、设备厂商和应用分发渠道之间割裂的责任。

应用安全问题产生原因复杂

Android 移动应用从开发到用户实际交互使用涉及多个环节，每个环节都存在引发安全问题的诸多因素。从 Android 系统本身的安全背景，到应用开发环节、应用上架过程，不同阶段都有可能为最终漏洞完整攻击链“添砖加瓦”。

系统安全存隐患，提权破坏安全机制

Android 本身是开源系统并且是开放的，所有人都可以研究源码。硬件厂商可以根据自身定制化需求在修改源码的基础上发布自己的系统并提供给用户使用，第三方 ROM 开发者可以根据自己的需求设计开发并提供给用户下载。甚至进一步，用户还可以实现手机提权并获取 root 权限。但厂商或者第三方 ROM 开发者的修改并没有任何门槛，在自定义优化的部分可能会引入新的安全隐患，root 权限的获取更是打破了系统本身的安全权限管理机制。

开源组件碎片化，安全修复困难

应用开发厂商在系统上进一步根据需求开发对应的 Android 移动应用程序。厂商在开发过程中也充分享受系统开源与开放的便利性，引入公开现成的开源组件。开源组件的引入确实为应用开发厂商的开发过程提高效率，减少类似功能的重复开发成本。但数量较多的第三方开源组件本身存在的安全问题，也会随着直接调用或者间接优化而转嫁给正在开发的移动应用。在当今开发采购模式下，开发厂商甚至几乎不可能知道哪些开源组件驻留在应用程序中，更不必说根据组件类别、版本进行安全问题追踪和修复。

安全能力不足，避雷力不从心

移动应用开发本身门槛低，尤其是更高效便捷的开发集成开发环境工具（Integrated Development Environment）的发展、更多开源组件的复用降低了 Android 应用开发难度。更多初学者也可以迅速学习并实现基本的功能。部分开发人员在开发的过程中，也有意识地避免写容易产生安全漏洞的代码，努力按照安全开发规范进行开发工作。但由于安全能力有限，遗漏、误用安全措施反而会造成新的安全隐患。例如，为了提高数据传输保密性而使用对称加密算法，但却将密钥硬编码在 Java 代码中，加密操作形同虚设。此外，使用 https 安全协议进行网络连接，但却调用 setHostnameVerifier 函数设置 ALLOW_ALL_HOSTNAME_VERIFY 标志位，导致中间人攻击的漏洞。

安全投入不足，缺失安全生命周期管理

由于安全漏洞产生的大部分环节都在开发过程中，因此应用安全的保障和提升需要在整个产品开发生命周期进行安全管理。一旦其中某些环节出现问题，都可能产生安全漏洞。由于涉及到的角色以及人员数量庞大，这对应用的安全生命周期管理提出了更高的要求。为了能更系统、完整地管理应用安全生命周期，需要应用开发厂商投入相应的人力、物力资源。部分厂商难以充分重视以及投入足够的资源，在完成开发后，没有进一步进行代码审计、安全扫描、漏洞修复以及应急响应。应用开发厂商将可能包含漏洞风险的应用提供给用户下载和使用，也意味着将安全问题暴露给恶意攻击者。由于系统本身特殊性，除了部署常规的安全策略，Android 应用安全生命周期管理还需要考虑保障手机被提权或者病毒入侵后应用自身的数据安全。

科恩实验室
KEEN SECURITY LAB

Android应用安全现状

腾讯安全科恩实验室
2018年Android应用安全白皮书

Keen Security Lab
2018 Android Application Security White Paper

科恩Android应用自动化漏洞扫描系统——ApkPecker

随着移动安全威胁与日俱增，攻击手段和安全防护技术也在相互对抗中发展。应用安全问题产生原因复杂而琐碎，难以用单一的系统或者工具完全解决相关问题。近年来，市场上也出现了大量与 Android 移动安全相关的工具，涉及应用漏洞扫描、恶意软件分析、实时安全监测等多个领域。相比较而言，针对应用开发厂商以及应用市场，使用应用漏洞扫描工具可以辅助相关安全人员定位应用安全隐患。在此背景下，腾讯安全科恩实验室研究并开发了面向攻击面的 Android 应用自动化漏洞扫描系统——ApkPecker。该系统是一款全自动 Android 应用漏洞扫描工具，通过对Android应用生命周期建模和应用攻击面建模，采用静态数据流分析和污点分析的技术，提高漏洞发现的准确率，同时保证检出的漏洞质量并给出漏洞触发的完整路径，最终以报告的形式全面分析应用的安全风险并给出修复建议。

为了客观地评估当前 Android 移动应用安全，我们使用腾讯安全科恩实验室 Android 应用自动化漏洞扫描系统对应用样本进行检测。我们对检测结果进行数据统计，然后从应用安全风险数量、等级、类型等角度分析，多维度展现 Android 应用安全风险现状。

ApkPecker主要有以下特性：

高准确率：

检测报告根据需求高精度筛选不同安全等级的漏洞结果，准确率处于行业领先水平；

攻击面全覆盖：

覆盖九大漏洞类型，精准定位三十余种漏洞检测项的风险位置。检测内容与检测插件保持持续更新，全面覆盖各种攻击面；

完整攻击路径：

基于数据流分析技术跟踪从攻击入口点到漏洞触发点的完整代码路径，精准定位漏洞，提供漏洞修复建议以及最佳安全开发范例以协助进行针对性修复；

动静结合检测模式：

一方面进行静态扫描结果的动态验证提高结果的准确性，另一方面针对应用的服务端系统做进一步的扫描，完善漏洞检测的维度；

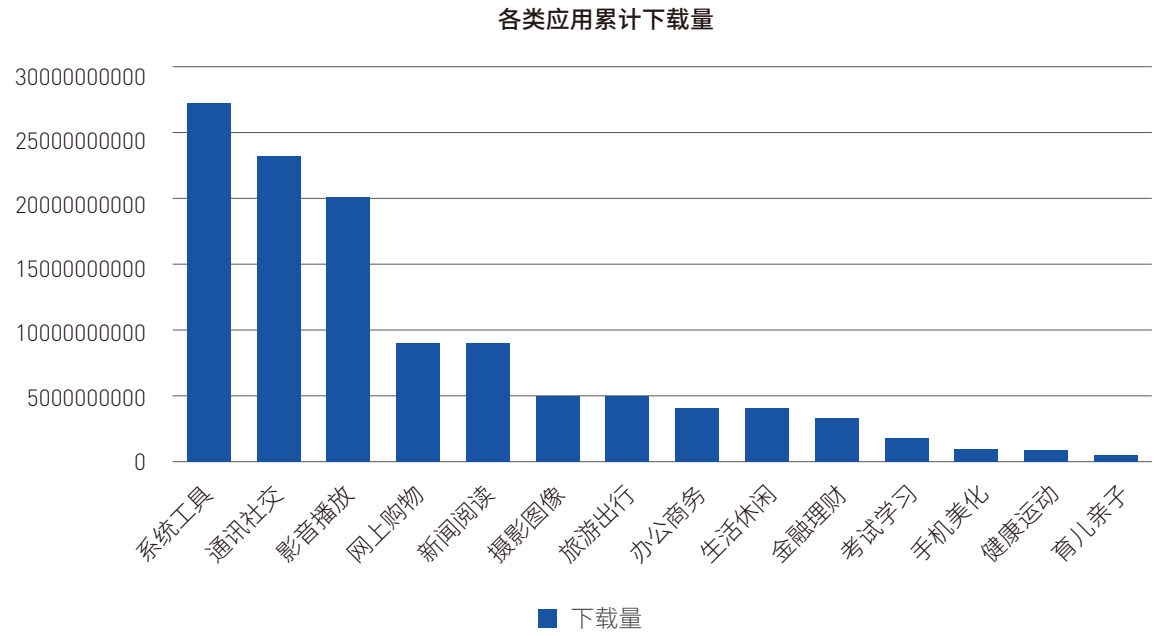
自动生成 POC：

支持自动化生成漏洞利用 POC 代码，直观体会漏洞可能产生的影响。

2018年Android应用检测样本统计分析

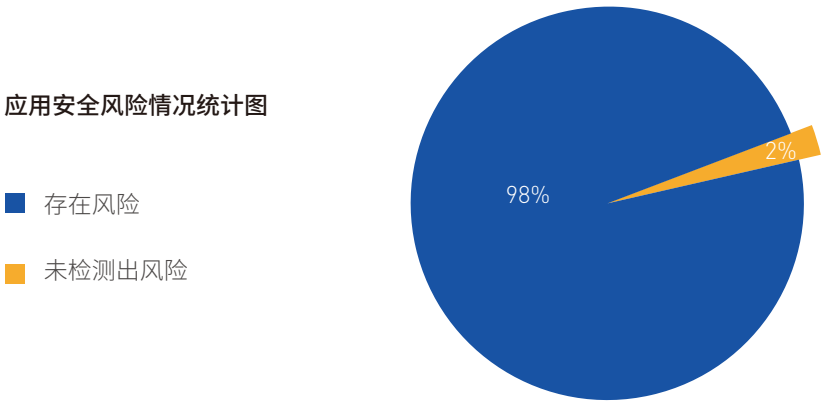
本次 Android 移动应用安全检测，我们选取 2018 年下载量较高的共计 1404 个应用进行漏洞扫描，在 1381 个 App 中共发现 37920 个安全问题，相关结果将在本白皮书中进行展示。

1404 个检测样本选取应用市场中 14 个分类下载量前 100 名左右的 App ，应用类型包括：影音播放、系统工具、通讯社交、手机美化、新闻阅读、摄影图像、考试学习、网上购物、金融理财、生活休闲、旅游出行、健康运动、办公商务、育儿亲子。根据应用宝数据，检测样本累计下载量超过 1100 亿次，各分类的下载量情况如下图所示：

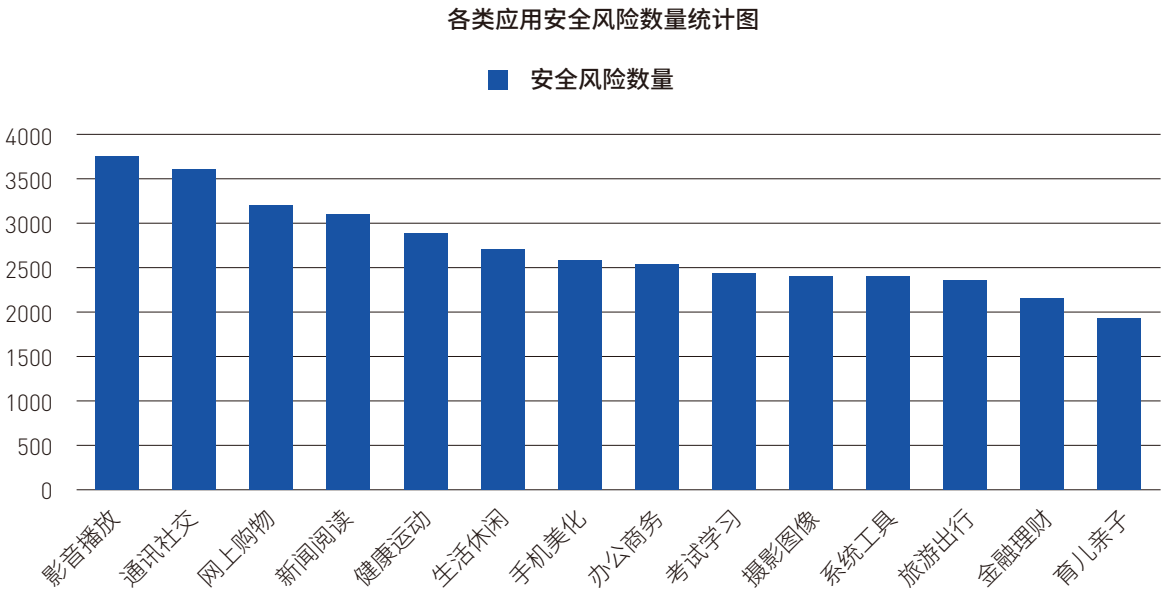


下载量排名前5位的应用类型是系统工具、通讯社交、影音播放、网上购物、新闻阅读，相关累计下载量将近900亿次。

1404 个 App 中仅 23 个未检测出安全风险，1381 个 App 平均每个应用发现 27.5 个安全风险，存在安全风险的 App 占比高达 98%：

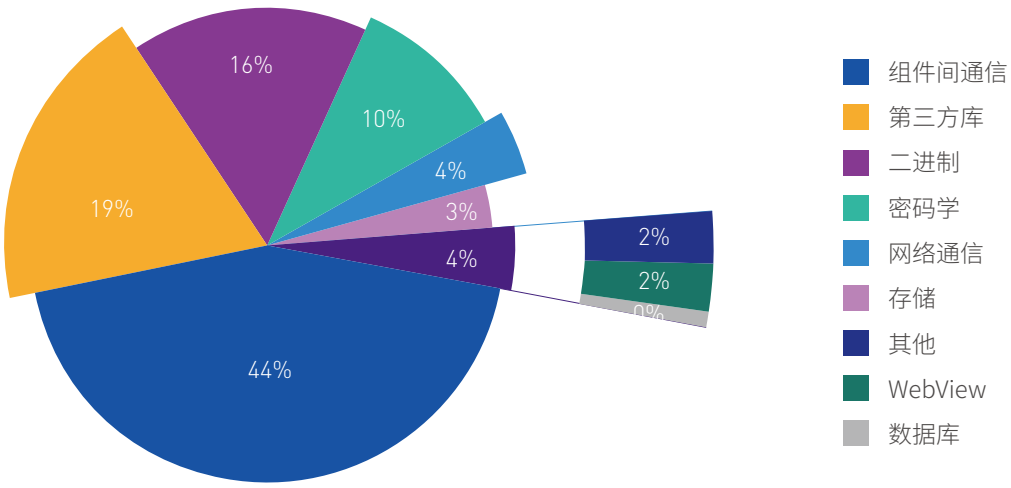


存在安全风险的 1381 个 App 中共检测出 37920 个安全风险，安全风险数量按应用类型分布情况如下：



安全风险数量排名前三的应用类型是影音播放、通讯社交、网上购物。这三类应用类型用户黏性强，平均应用使用时长也位列前三。相比较而言，通讯社交、网上购物、影音播放这几类应用产品功能多、交互方式丰富，一旦有安全漏洞被不法分子利用，影响的用户群体数量大、范围广，造成的损失严重。

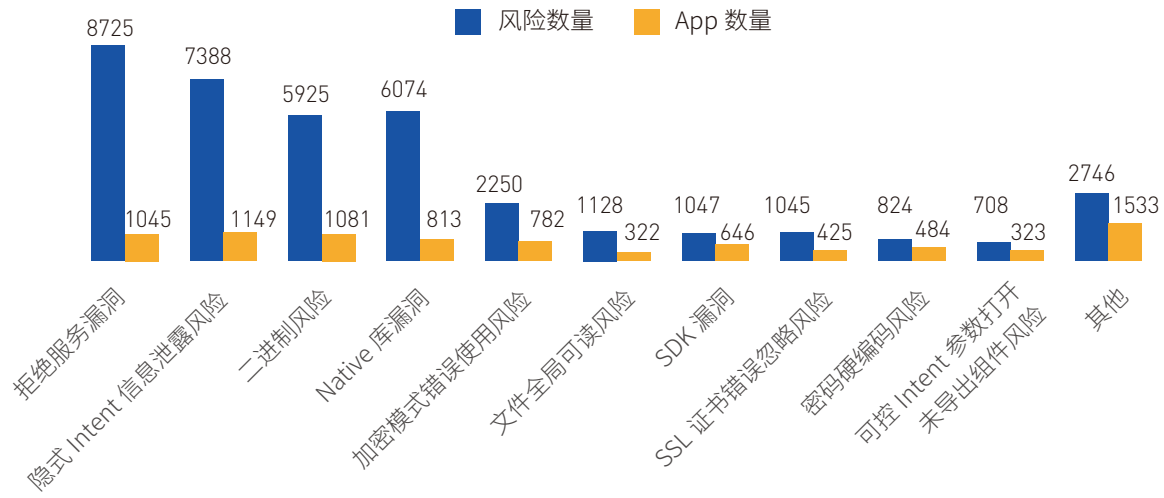
本次检测针对九大安全检测类共 31 种安全风险类型，每种检测内容都发现数量不等的安全风险。九大安全检测类分别为：组件间通信、二进制、密码学、存储、第三方库、网络通信、WebView、数据库、其他。检测出的安全风险数量按九大安全检测类分布情况如下：



安全风险数量最多的检测类型是组件间通信，数量超过 16000 个，占比超过 44%。其次第三方库、二进制、密码学相关安全风险数量也相当多。

根据更细分的 31 种安全风险类型分类，排名前 10 的类型所对应检测出的安全风险数量以及受影响 App 数量分布情况如下：

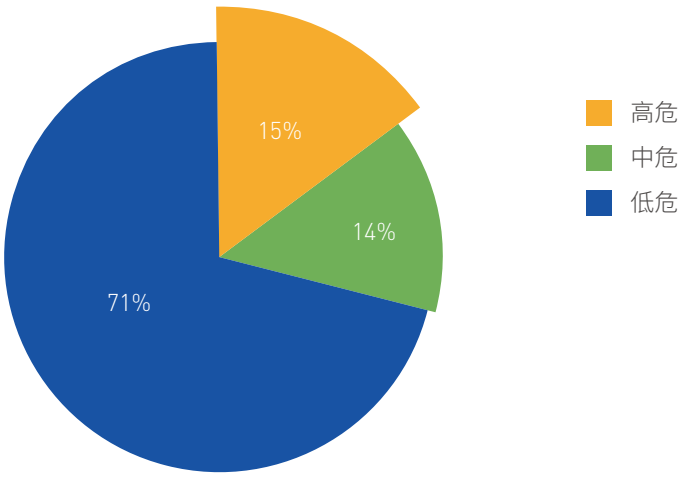
不同风险类型数量以及相关 App 数量统计图



拒绝服务漏洞、隐式 Intent 信息泄露风险以及二进制风险三种类型安全风险影响的 App 数量最多，数量超过均 1000 个。Native 库漏洞安全风险也不容忽视。

检测出的安全风险按照安全等级划分，分布情况如下：

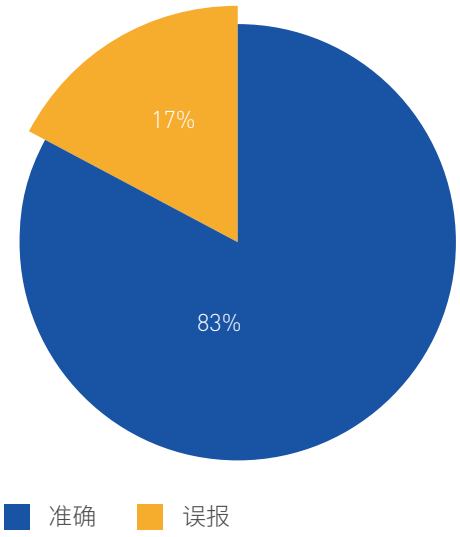
安全风险等级分布图



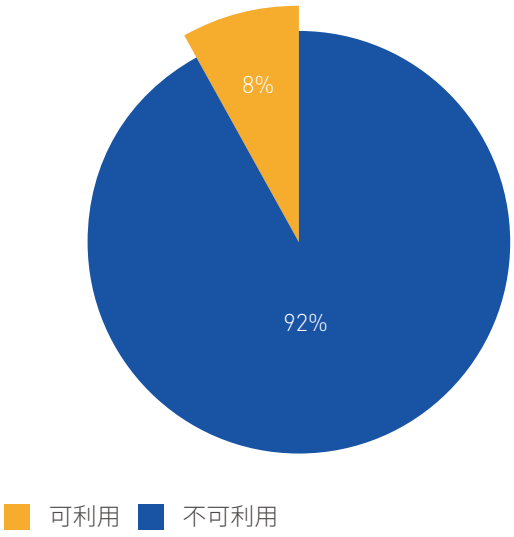
由于拒绝服务漏洞、隐式 Intent 信息泄漏风险、二进制风险等数量排名靠前的安全风险类型的安全等级属于低危，因此在本次检测中，低危占比超过 70%。尽管高危、中危比例并不突出，但在安全风险总量大的前提下，高危和中危安全风险可以造成的安全威胁仍然十分严重。

上述由腾讯科恩实验室自动化漏洞扫描工具 ApkPecker 检测出的结果，我们抽样选取 140 个样本后进行进一步安全风险确认。在选取的 140 个样本中，检测结果中的安全风险误报率约为 17%，可利用漏洞数量超过 240 个，占抽样样本检测结果中总漏洞数量 8%。

ApkPecker 检测结果准确率

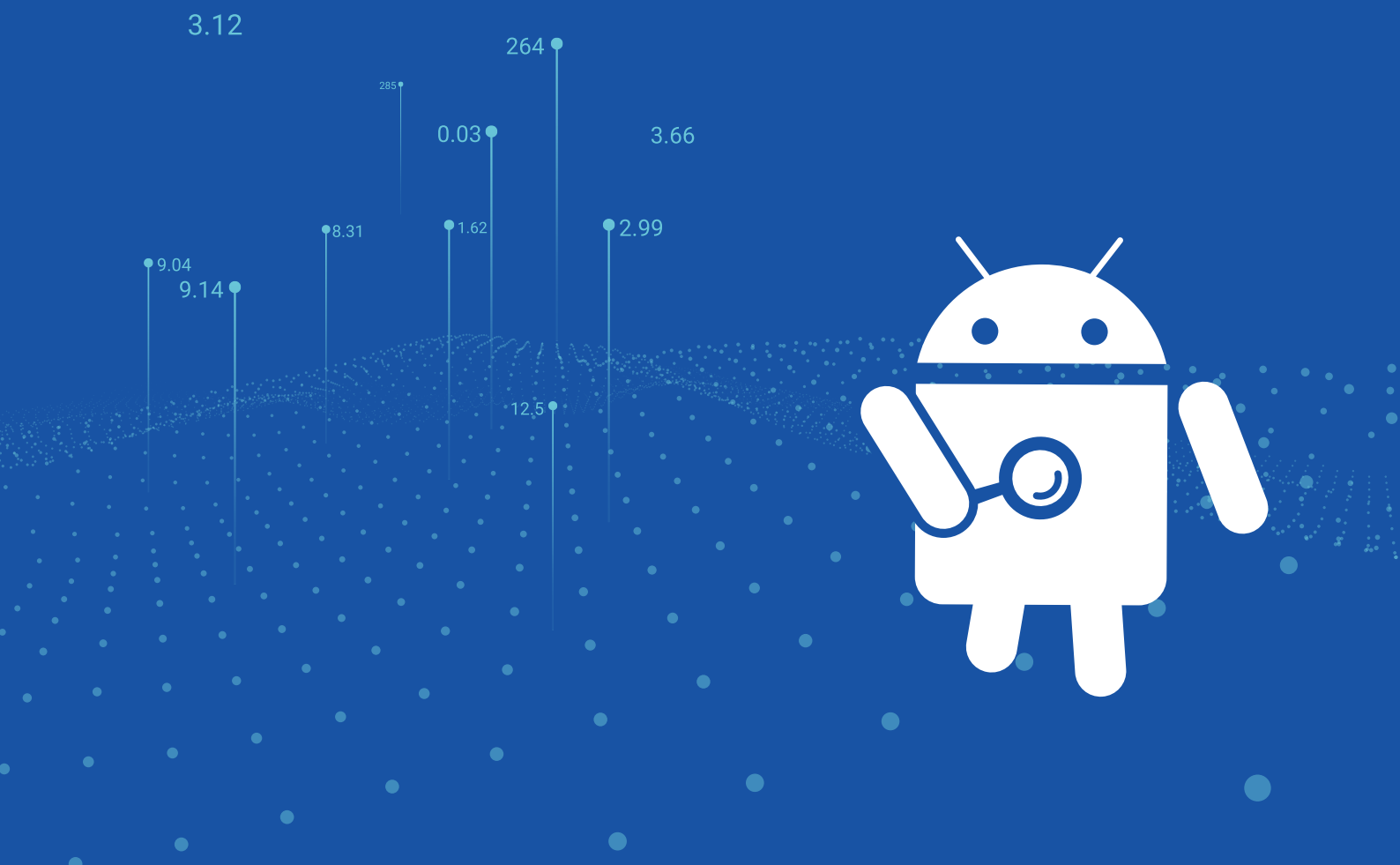


安全风险可利用比例



科恩实验室
KEEN SECURITY LAB

Android应用安全风险分析



腾讯安全科恩实验室
2018年Android应用安全白皮书

Keen Security Lab
2018 Android Application Security White Paper

本章根据安全风险触发场景来对移动安全常见风险进行分析。数据泄漏、组件间通信风险、SDK 库漏洞、Native 库漏洞等问题，在当前移动应用频繁出现。App 的安全风险并不是相互独立、彼此割裂的关系，甚至结合多种安全风险构建成完整攻击链的趋势越来越明显。这些利用场景会对用户造成非常直接、严重的安全威胁。除了上述几类移动应用内常见检测项容易发现的问题，越来越多的后台问题也需要引起重视。此外，2018 年是移动应用隐私问题频发的一年，针对移动应用隐私检测的分析也会在本章进行客观展示。

数据泄露

随着移动应用功能复杂化与交互多样化，越来越多的应用将数据进行本地存储。存储的数据内容可能是涉及个人隐私信息的聊天数据和身份信息，或者是健康状况的医疗数据，甚至是敏感的金融类移动支付数据。移动设备上数据存储的性能、安全等需求，对应用和系统在用户数据保护方面提出了更高的要求。

Android 系统提供了多种方式对应用数据进行存储和读写。存储区域包括应用私有目录和外置公共存储区域，数据读写方式除了常规的文件操作以外，还提供了如系统组件 ContentProvider 和 WebView 特性 FileScheme 等方式。而在使用这些方法时，由于开发人员安全意识的缺失或是对系统功能理解的不透彻，可能会导致应用数据的过度泄露，甚至导致账号权限的劫持。

调试信息和日志文件暴露重要数据

开发人员在应用开发的过程中，出于方便调试和日志分析的目的，会将类似 Token、网络明文包等关键数据通过 log 打印或者写入本地日志文件。当数据存储到 SDcard 等不安全的区域时，就会引起关键数据的泄露。

例如，某款音箱应用将网络交互包明文记录在 SDcard 上的日志文件，其中包括了关键的 union_access_token 数据，获取该 token 意味着可以实现音箱的远程控制。

```
#: 2018 年 08 月 24 日  21:34:27
content: request---  https:[redacted]/app/invoke/
?sign=9e6250921adaeb7d9d3cb568099c5a2f&text=nonetext&params={"client_id":"[redacted]1503470121484",
"device_type":"1","deviceid":"8573dae889c34259","dt":"1535117667","imei":"","lat":"","lng":"","os_type":"1",
"os_version":"7.1.1","ovs_sdk_os":"android_mobile","ovs_sdk_version":"1.1.9","speaker_sn":"032201809000412",
"union_access_token":"eyJhbGciOiJIUzI1NiJ9.eyJjbGllbnRJZCI6Im9yaW9uLnVjZW50ZXIuZTBlMWRiYzVjMmVkZTY0ZCIsIm9wZW5JZCI6IjVmMDd
mMzg5MDhhYTMzNmNiOTUyYzU5Y2U2NDZhMzJiIiwic2NvcGUiOiJvdnM6YXBwIiwiaXNzIjoib3Jpb24iLCJleHAiOjE1MzUyMDM3MDQsInR5cGUiOiJhY2Nlc
3NfdG9rZW4iLCJpYXQiOjE1MzUxMTczMDQsInZlcnNpb24iOiIxLjAifQ.7_8wJrlXxDel2R8HdKtffsNA_P5-gZXBHb9S5ocDCAU",
"version":"1.1.19"}&sid=cc342b1d-e68e-4aa4-8807-4c6f4653a957&intent=[{"domain":"BizSpeaker","english_domain":"BizSpeaker",
"intent":"getStatus","intent_type":"interface","slots":[{"action_key":"","app_device_info":{"device_id":"8573dae889c34259",
"device_type":"android"},"speaker_device_id":"032201809000412"}],
"source":"2"}]request method ----GETcurrentTime = 1535117667944--->will start
```

安全建议：随着新权限管控粒度的引入，Android Q 系统加强了对于外置存储的权限控制，对该类问题有一定程度上的缓和。开发者在开发过程中也应严格管理正式发布版本中调试与日志的信息脱敏。

文件分享存在隐患

当前不少应用之间存在文件分享功能，例如可以在一个应用中处理编辑照片视频再一键分享到别的应用进行其他操作。文件分享过程涉及分享内容路径的传递。但倘若没有对分享内容路径进行安全性校验，可能会导致泄漏应用自身数据。在 1404 个移动应用样本检测结果中，共有 151 个移动应用被检测出文件分享数据泄露问题。

在某应用旧版本 SDK 中，存在一个公开组件来完成分享的功能：

```
private void shareImage() {
    Intent intent = getIntent();
    Message message = parseIntent(intent);
    this.setContentView(this.rootlayout);
    if(message.imageObject != null) {
        String v6 = ImageUtil.copyFileToSdcard(this, message.imageObject.path, 0);
    }
}
```

文件复制

在实现分享功能的过程中，会将分享的文件拷贝到外置某存储目录。由于缺少对分享文件路径的安全性验证，可以将应用的任何私有文件拷贝到外置存储，导致了应用的数据泄露。

安全建议：

对于分享内容需要进行严格的安全性校验。

ContentProvider数据接口需多重防护

Android 系统中提供了 ContentProvider 的数据共享机制，可以通过 URI Scheme 机制来实现数据的访问。当对外公开的 Provider 缺少数据访问权限的检查时，也会导致应用数据的泄露。本次共检测出 10 个移动应用存在 ContentProvider 数据接口泄露问题。

例如，我们检测发现在某应用中包含公开 ContentProvider 组件，对外部传入的参数路径没有做严格的安全校验，导致了文件路径穿越的漏洞，最终可以导致核心的账号数据泄露。

```
<provider android:authorities="com.[redacted]br.ApptentiveAttachmentFileProvider"
          android:enable="true"
          android:exported="true"
          android:grantUriPermission="true"
          android:name="com.apptentive.android.sdk.debug.ApptentiveAttachmentFileProvider" />

public ParcelFileDescriptor openFIle(Uri arg4, String arg5) {
    arg4.getLastPathSegment();
    if (this.uriMatcher.match(arg4) != 1) {
        throw new FileNotFoundException(new StringBuilder("Unsupported uri: ").append(arg4.toString()).toString());
    }
    return ParcelFileDescriptor.open(new File(new StringBuilder())
        .append(((ContentProvider)this).getContext().getCacheDir())
        .append(File.separator)
        .append(arg4.getLastPathSegment())).toString(), 0x10000000);
}
```

在进一步的情况中，对于未公开但设置了 "grantUriPermissions=true" 的 Provider，可以通过在 Intent 调用中设置 Flags 来授予 Provider 的访问权限。如果应用中同时存在可控的 Activity 调用漏洞时，通过可控的 Intent 数据回调到攻击应用，攻击应用同样可以获取 Provider 的数据访问权限，导致 Provider 接口完全暴露，造成未公开数据泄露。

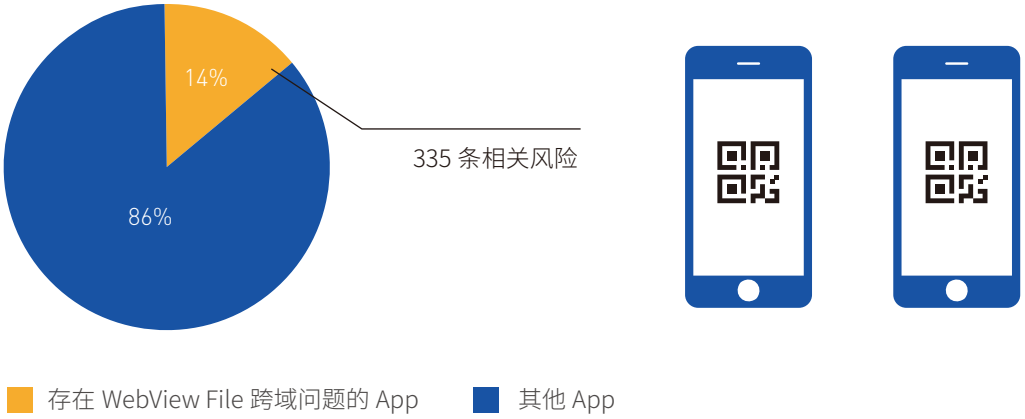
```
Intent intent = new Intent();
intent.setComponent(new ComponentName("com.tencent.keenlab", "com.tencnet.keenlab.PocActivity"));

intent.addFlags(FLAG_ACTIVITY_NEW_TASK);
intent.addFlags(FLAG_GRANT_READ_URI_PERMISSION);
intent.addFlags(FLAG_GRANT_WRITE_URI_PERMISSION);

Uri uri = Uri.parse("content://" + AUTHORITY + "/my_downloads");

intent.setData(uri);
```

安全建议：在使用 ContentProvider 对外提供数据访问时，除了做好权限的配置，也要对外部输入的数据做严格的校验，同时也需要防范多个漏洞所引起的漏洞链，做好整体的安全防护。

WebView FileScheme疏忽造成应用克隆威胁

在 2018 年 1 月，腾讯安全首次公开“应用克隆”这一移动攻击威胁模型。该威胁是利用 WebView 支持使用 FileScheme 的模式访问文件的特性，实现读取应用私有文件，造成应用数据泄露。在本次检测结果中，共 190 个移动应用被发现仍有 335 个 WebView FileScheme 数据泄露问题。

WebView File 跨域相关 App 占比图



在现有的应用开发模式中，Web 和原生 Android 代码的混合开发变的越来越普遍，WebView 在应用中扮演的角色也越来越重要。由于业务开发的需要，Web 页面可以通过 JavaScript Interface 与 Java 代码进行复杂的通信以及访问文件。WebView 可以通过三个 API 控制 File 域访问文件的功能：

```
webView.getSettings().setAllowFileAccess(boolean);
webView.getSettings().setAllowFileAccessFromFileURLs(boolean);
webView.getSettings().setAllowUniversalAccessFromFileURLs(boolean);
```

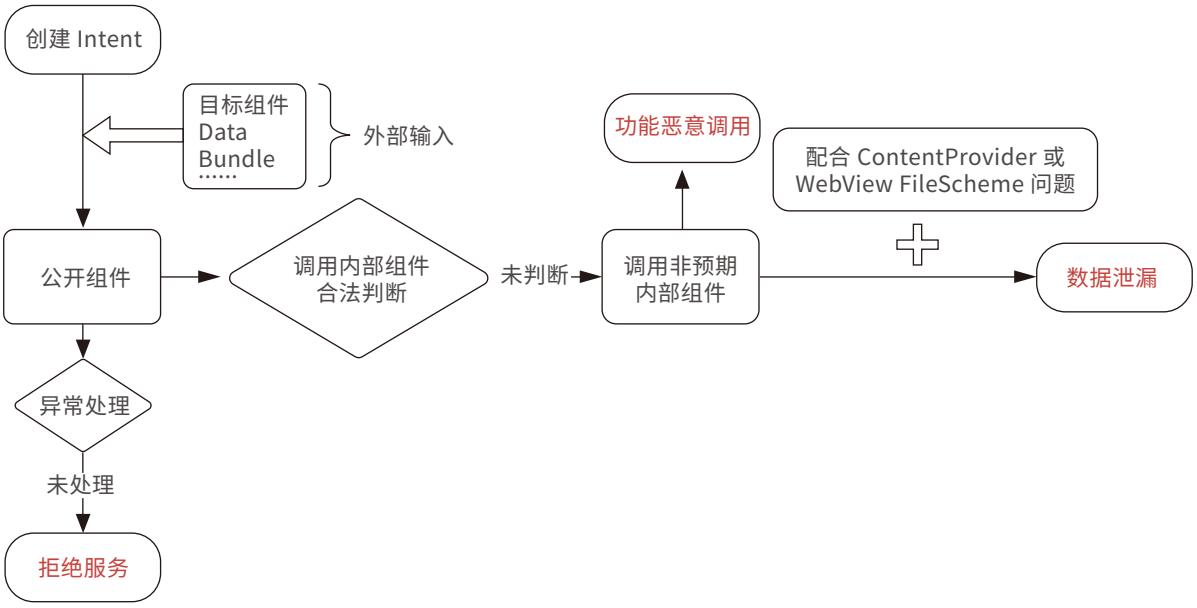
当在 WebView 的配置中开启了 setAllowFileAccessFromFileURLs 或者 setAllowUniversalAccessFromFileURLs 时，即可通过让 WebView 以 File 域加载网页的方式去访问其他的应用文件，最终导致数据的泄露。对于包含 WebView 的组件，某些情况下为了支持浏览器的调用，会在配置文件中加入浏览器调用的支持，从而会导致更严重的远程利用。

安全建议：

合理配置 WebView 进行 File 域访问文件的功能，对于以 File 域加载网页的访问方式还需要进一步进行判断访问文件权限。

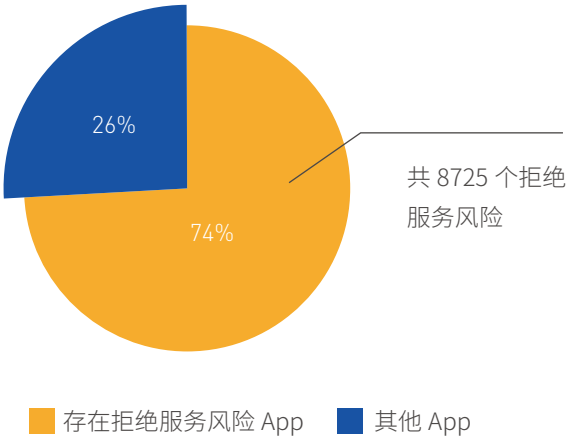
组件间通信问题

App 的组件分为导出组件和非导出的内部组件。导出组件在为同设备 App 提供访问接口的同时，也会引入相应的安全风险。在组件间通信过程中，有多种常见被利用的方法。如果开发人员缺少对外部输入的校验和异常处理，通过构造异常 Intent 输入数据，则可以造成拒绝服务攻击。如果公开组件根据输入数据，会新生成 Intent 来启动内部组件，同时 Intent 数据访问未被限制，外部输入控制新生成 Intent 的目标组件、Data、Bundle 等参数，可以访问内部非公开组件，造成功能被恶意调用。在此基础上，如果配合 ContentProvider 和 WebView File Scheme 问题，可以造成数据泄漏。

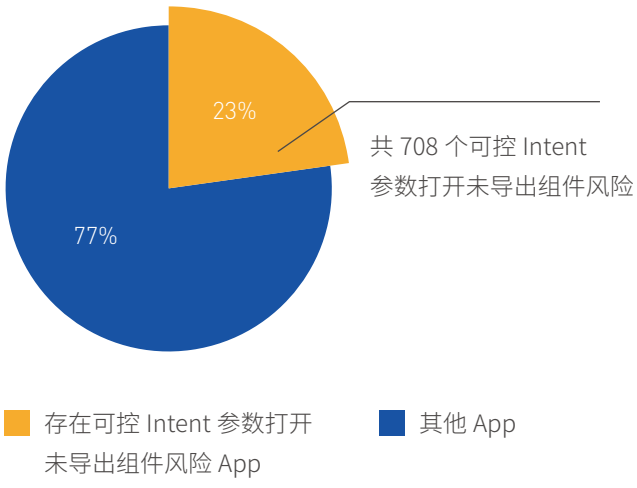


在检测的 1400 多个应用中，我们在 1045 个 App 中检测出存在 8725 个拒绝服务风险，在 323 个 App 中发现 708 个可控 Intent 参数打开未导出组件导致功能被恶意调用风险。数据泄漏问题已在上一小节进行详细介绍，本节不再赘述。

存在拒绝服务风险 App 占比统计图



存在可控 Intent 参数打开未导出组件风险 App 占比图



组件间通信过程中，通过导出组件收到 Intent 来生成新的 Intent 以启动不同的内部组件。这种场景下的 Intent 的创建有四种常见方式：

(1)Intent 复制，通过 new Intent(intent) 函数实现；

(2) Intent 转发，通过 Intent.getParcelable 系列函数实现；

(3) Intent 的组合拼装，通过 Intent.setComponent, Intent.putExtra，Intent.setData 等函数实现；

(4)URI Scheme 生成 Intent，通过 Intent.parseUri 实现。

以 Intent 转发创建 Intent 被利用的安全风险为例，在我们的检测结果中发现某应用中存在导出组件 ClientUiProxy-Activity：

```
//com.[redacted].ui.client.ClientUiProxyActivity
protected final void onCreate(Bundle bundle) {
        super.onCreate(bundle);
        Intent intent = (Intent) getIntent().getParcelableExtra(
            "com.[redacted].PROXY_INTENT");
        if (intent != null) {
            startActivityForResult(intent, 1000);
            return;
        }
}
```

通过控制 Intent 中的 PROXY_INTENT 参数，可以控制 startActiviy 的整个 Intent，以该应用权限启动任意 Activity。同时该 App 中存在一个内部组件 PrebundledGameActivity 存在 WebView file 跨域问题。组合利用这两个漏洞，可以实现读取应用私有文件从而实现信息泄漏。

```
//com.[redacted].games.features
//.builtingames.PrebundledGameActivity
protected final void onCreate(Bundle bundle) {
    webView = (WebView)findViewByID(R.id.webview);
    webView.getSettings.setAllowUniversalAccessFromFileURLs(true);
    //...
    String data = getIntent().getDataString();
    webView.loadUrl(data);
}
```

安全建议：

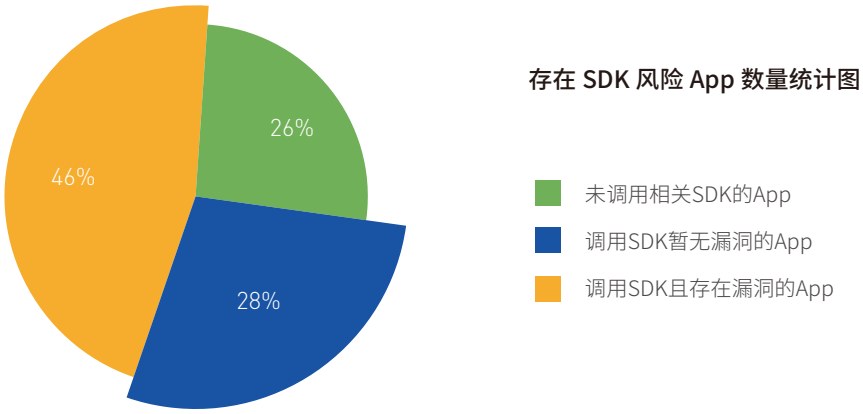
在 Intent 的创建过程中，严格校验外部输入数据，考虑异常处理，避免非预期的内部组件被外部 App 访问。

第三方库漏洞

集成第三方库的开发方式可以增强应用功能以及加快开发周期，因此大量 App 在开发周期中会直接调用第三方库。但部分第三方库在开发过程中没有注重代码的安全性，存在安全问题。而这些安全问题往往会成为攻击者攻击宿主应用的“跳板”。这样即使宿主应用本身有着良好的安全性，应用整体也会面临着严重的安全风险。除此之外，第三方库爆出安全问题并发布安全补丁更新新版后，应用没有及时跟进安全修复方案，也可能会在窗口期被恶意利用。本节将从 SDK 库以及 Native 库两方面分析第三方库的安全问题。

SDK库

本次自动化检测分析了市面上数十款主流 SDK，在 1404 款检测样本 App 中，共有 1038 个 App 有 2166 次 SDK 相关调用。统计显示，接近一半比例共 646 个 App 上存在 1047 个 SDK 漏洞。



近来SDK的安全逐渐受到大家的广泛关注，17年友盟SDK被爆出越权漏洞。友盟SDK中一个导出组件存在以下逻辑：

```
//Construct message and start service
protected void onMessage(Context arg7, Intent arg8) {
        super.onMessage(arg7, arg8);
        String body = arg8.getStringExtra("body");
        UMessage v3 = new UMessage(new JSONObject(body));
        Intent intent = new Intent();
        intent.setClassName(v3.pulled_package, v3.pulled_service);
    this.startService(intent);
}
```

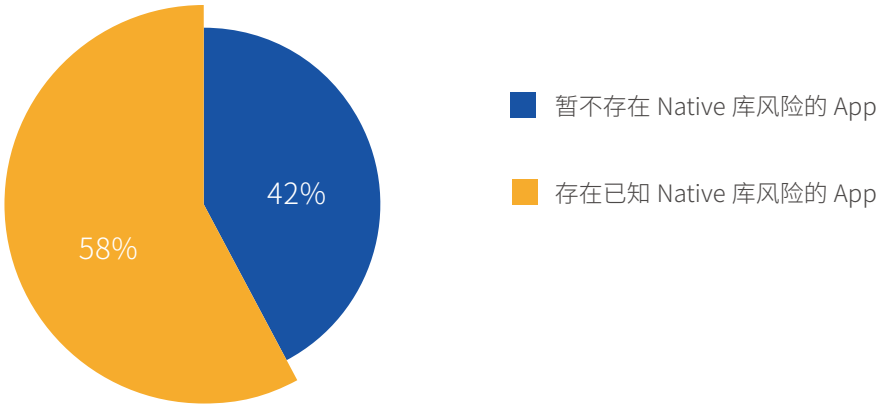
该 SDK 会使用 Intent 构造 UMessage，进而启动 Service，因此，攻击者可以构造恶意的 Intent 启动任意非导出的 Service，同时配合 App 其它一些逻辑最终可以实现任意代码执行。

此外，除了 SDK 本身的安全问题，还有一些问题是由于开发者使用不当而导致的安全问题，例如某 SDK 文档中要求组件设置为不导出，而开发者擅自将组件导出，导致攻击者可以构造 Intent 打开任意非导出 Service。

Native库

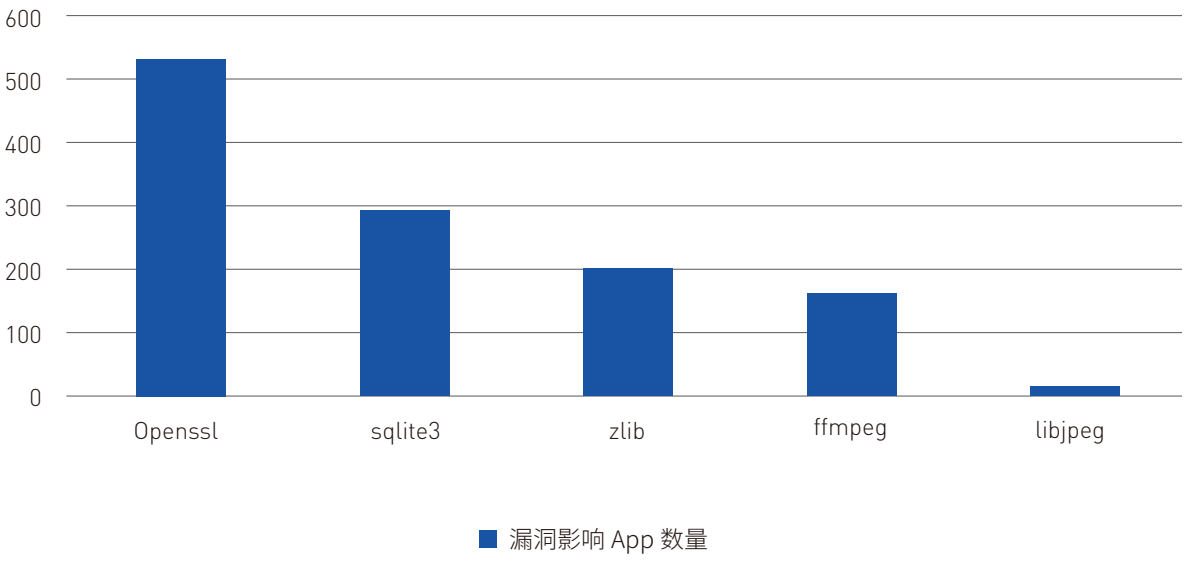
本次检测 1404 款检测样本 App 中，存在 6565 次 Native 库相关调用。扫描结果显示，共有 813 个 App 上存在超过 6074 个 Native 库 CVE 漏洞。包含存在漏洞 Native 库的 App 比例超过 58%。

存在 Native 库风险 App 数量统计图

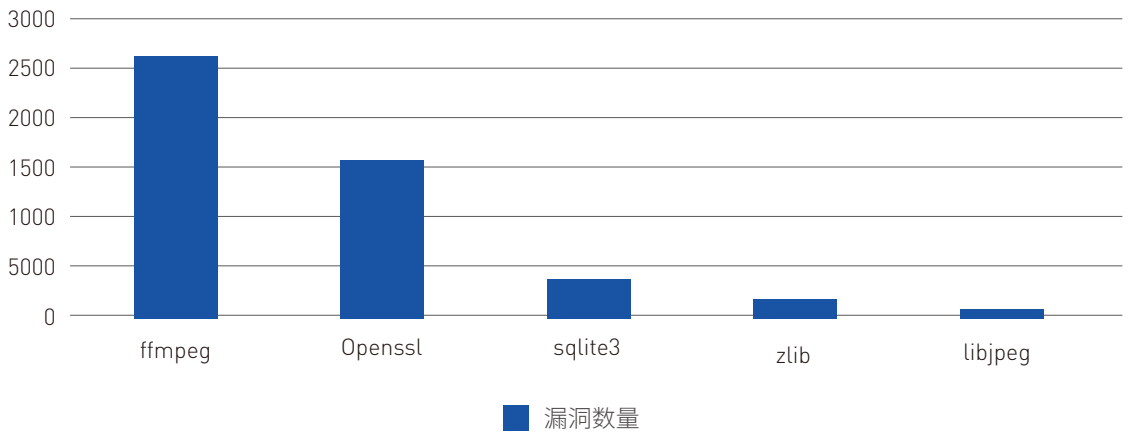


其中，检测样本中包含 Native 库数量排名前五的 Native 库种类为：openssl、sqlite3、zlib、ffmpeg、libjpeg。这五类Native库共涉及4738个CVE漏洞，影响了615个App。调用相关Native库受漏洞影响的App数量分布情况如下:

5 类 Native 库漏洞影响 App 数量统计图

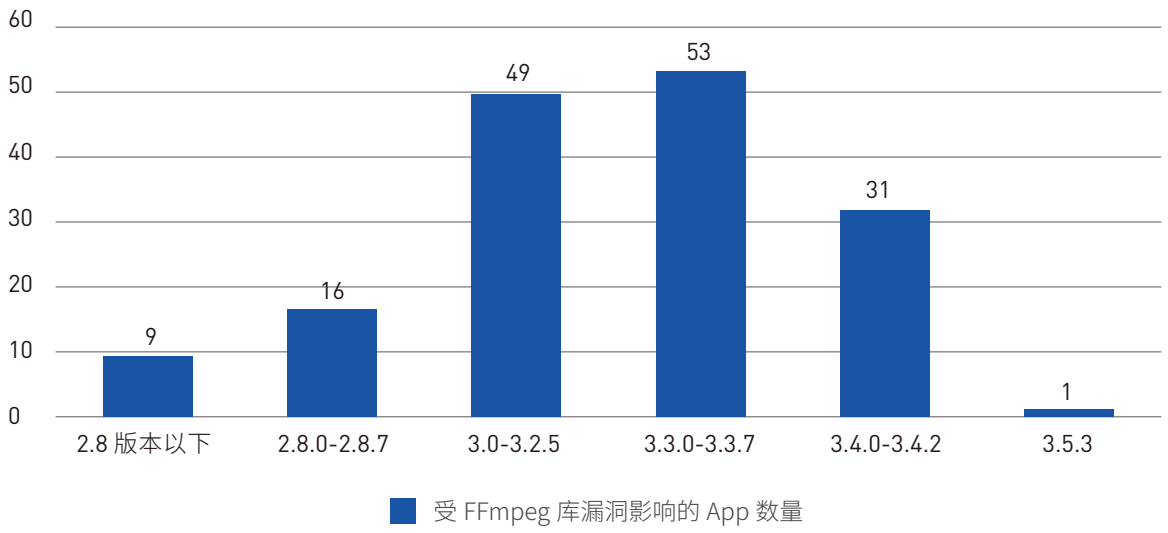


5 类 Native 库漏洞数量统计图



在调用 ffmpeg 的 254 个应用中，159 个 App 受到 ffmpeg 漏洞影响，受漏洞影响 App 的版本分布情况如下图。例如，2017 年 5 月发布的 ffmpg 3.2.5 版本存在多个中高危漏洞，攻击者利用漏洞 CVE-2017-9993 可借助特制的播放列表数据实现读取任意文件。

FFmpeg 库版本漏洞影响 App 数量统计图



与 SDK 库漏洞类似，Native 库漏洞都是公开的漏洞信息。当应用使用包含漏洞的旧版本第三方库，在应用完善漏洞补丁或者更新第三方库版本的窗口期中，结合公开的漏洞信息甚至利用代码，以第三方库为攻击面也可以实现拒绝服务、任意文件读写、远程命令执行等各种攻击。

安全建议：

第三方库安全在 App 开发生命周期中的安全管理，一直是容易被忽视的盲点。碎片化、难追溯更是进一步导致第三方库安全性问题恶性循环。为了避免第三方库可能引入的安全威胁，需要从源头介入，把控调用第三方库版本以及全生命周期进行安全问题追踪，并在保障应用功能稳定性的情况下尽可能将受漏洞影响的第三方库版本更新到最新版本。

服务后台问题

App 引入服务后台，可以突破移动设备自身的一些局限性，更好地支持应用功能来提升体验，还可以统一集中化管理数据，避免一些客户端可能会产生数据泄漏问题。因此，移动应用功能和业务的后台化也变得比较普遍。

服务端在整个架构里扮演着核心枢纽的角色，承担了数据存储、逻辑处理、业务执行等功能，服务端等安全问题也会直接影响到所有客户端用户。一旦服务端产生安全问题，轻则破坏系统功能可用性，重则导致用户数据与敏感信息泄露、服务崩溃等严重后果。而服务端也往往是恶意攻击者重点关照的对象。有效识别服务端面临的安全风险，并进行对应的防护处理，对整个 Apk 安全体系的建设也具有重大意义。

Apk 的服务端与传统 Web 服务类似，面临着所有 Web 服务所共有的潜在安全隐患，

常见安全风险有以下几个方面：

平台层面

存在传统操作系统的漏洞威胁，比如本地权限提升等问题

应用层面

例如 SQL 注入，XSS，CSRF，SSRF 等漏洞

业务逻辑层面

越权、敏感信息泄露、逻辑缺陷、弱口令等漏洞

其他

存在被 DDos/CC 攻击的风险

由于服务端还与客户端存在频繁的交互，因此接口会直接暴露在公网，这也为攻击者提供了攻击面。

安全建议：

考虑到服务端问题面临的复杂环境，首先需要在业务上线前和运行周期内进行安全风险检测和实时安全把控，争取做到先于攻击者发现安全问题并进行及时修复；同时也需要在不影响正常业务逻辑的情况下对其进行安全防护，需要结合具体业务场景定制防护策略。

隐私问题

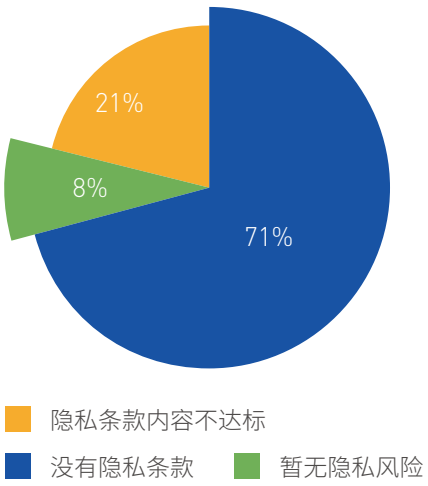
近年来，移动应用隐私信息问题被屡屡曝光，引起不少群众对隐私数据的担忧，隐私安全问题也会严重影响相关开发厂商信誉。Facebook 在 18 年被爆出多个信息泄漏问题，例如由于用户照片获取接口的程序错误，导致用户上传的私密照片被第三方应用程序访问。开发者及企业在开发 App 时，也应该科学获取隐私权限并避免越界获取。

App 违规收集隐私问题受到越来越多的关注，国家也从法律法规的角度不断推进网络用户隐私信息保护。2017 年 6 月 1 日起正式实施《国家网络安全法》，明确规定要加强个人信息保护。2018 年 5 月 1 日起实施中国国家标准化管理委员会发布的 GB/T 35273-2017《信息安全技术个人信息安全规范》，2019 年 1 月 25 日，中央网信办、工信部、公安部、市场监管总局四部委联合发布《关于开展 App 违法违规收集使用个人信息专项治理的公告》，持续加大保护力度，专项治理 App 违法违规收集使用个人信息的行为，探索长效监管机制。

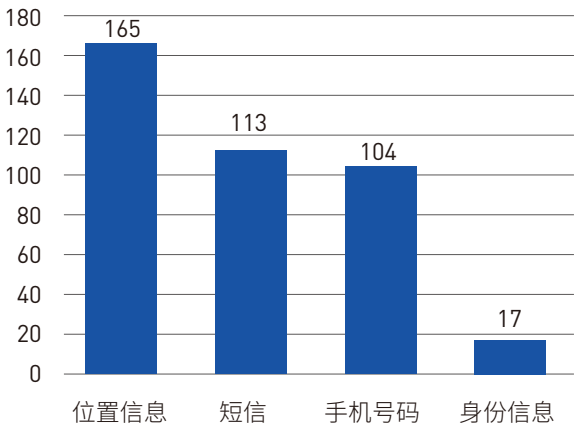
App 违法违规收集使用个人信息主要体现在 App 过度收集核心隐私权限。一方面，App 获取产品功能并不需要的权限，另一方面，没有按照隐私声明向用户保持权限获取的公开透明。

为了更好地帮助应用开发厂商发现并解决应用存在的隐私合规性问题，帮助应用商店审核上架 App 隐私内内容，腾讯安全科恩实验室提供隐私合规性检测服务，并开放相关核心能力，助力用户个人信息保护。在本次检测中，在 1404 个 App 中共发现 1292 个 App 存在过度收集核心隐私权限的问题。其中，294 款 App 隐私条款内容不达标，998 款 App 没有隐私条款。

应用隐私过度收集统计图



各类隐私过度收集 App 数量



294 款 App 隐私条款内容不达标，这意味着这些应用未明确告知收集个人信息类型，且收集敏感信息时未明确告知用户信息的用途。过度收集或使用的隐私数据主要包括位置信息、通讯录信息、手机号码等个人信息。其中，共计 165 款 App 涉嫌过度收集了位置信息。分别有 113 款、104 款、17 款 App 涉嫌过度收集或使用短信、手机号码、身份信息。

以某电商 App 为例，根据 ApkPecker 的扫描结果，运行过程中收集了位置信息、通讯录信息、身份信息、银行卡号、手机号码、短信等隐私数据，但是在隐私政策中却未明确告知应用会收集身份信息、银行卡号以及短信，且没有说明这些信息的用途。

安全建议：

移动应用开发厂商应合理申请所需的隐私数据，并通过详细的隐私条款与用户说明所需隐私数据的用途，在使用相关数据过程中，也应加强数据安全保护能力，避免被恶意窃取或者信息泄漏。

科恩实验室
KEEN SECURITY LAB

总结与展望

腾讯安全科恩实验室
2018年Android应用安全白皮书

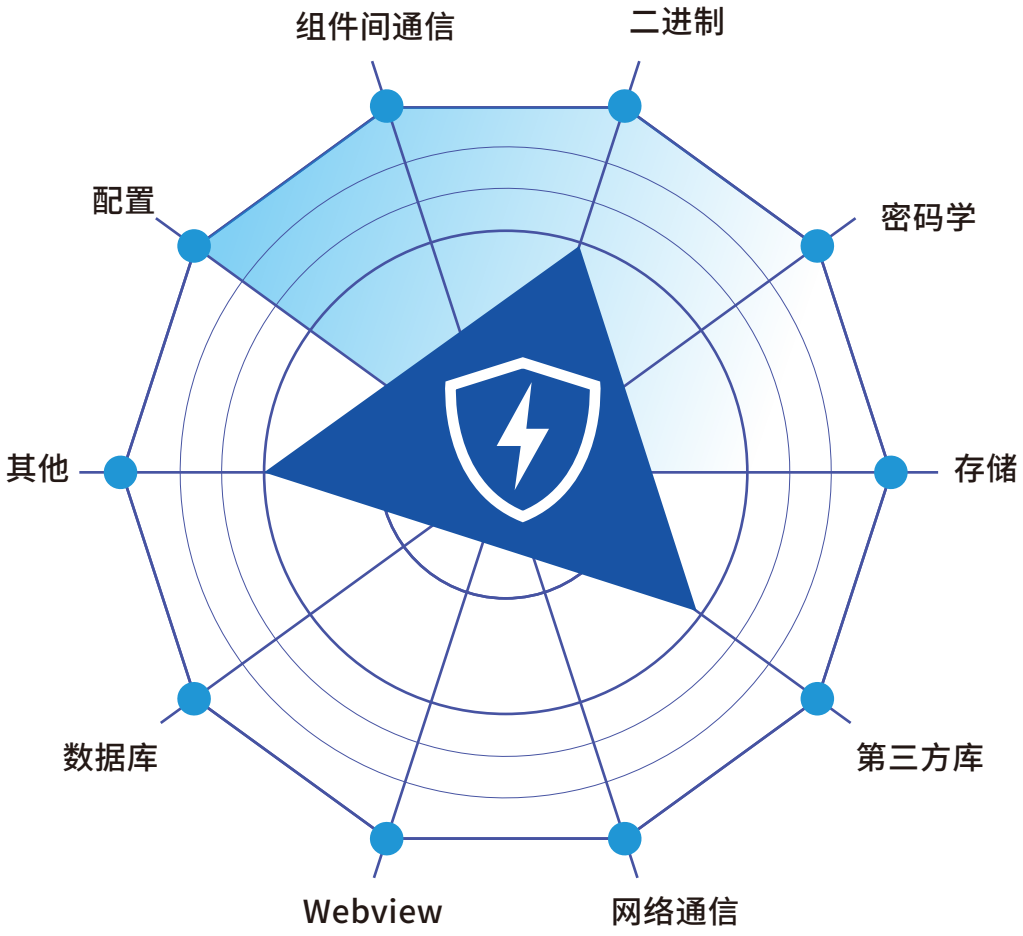
Keen Security Lab
2018 Android Application Security White Paper

Android应用安全自查雷达图

对于 Android 应用开发厂商以及应用商店等平台，使用基于面向攻击面的静态检测工具对移动应用进行安全检测相对便捷。

参考木桶原理，Android 应用安全风险并不是以最强的防御措施来衡量，而是由短板的攻击面风险情况决定的。如何快速高效地划分攻击面类型，如何定性定量地评估攻击面安全风险，都需要科学、全面以及实用的评估模型来指导。腾讯安全科恩实验室基于丰富的移动应用渗透测试经验以及前沿攻击模式分析与总结，提出适用于移动应用面向攻击面静态检测的安全自查雷达图。

Android 应用静态安全检测自查雷达图的攻击面以腾讯安全科恩实验室检测系统 ApkPecker 的九大安全检测类为主要内容，包括：组件间通信、二进制、密码学、存储、第三方库、网络通信、Webview、数据库以及其他。应用配置文件也会给上述检测项提供额外的信息，也需要在检测中予以关注。



攻击面	检测项	安全实践建议
组件间通信	IntentBridge 组件权限泄露漏洞	对外部输入 Intent 数据的合法性做针对性校验
	ContentProvider 路径穿越漏洞	对外部传入的文件路径做合法性校验
	Intent 调用拒绝服务风险	对外部数据的解析过程做异常捕获处理
	隐式 Intent 数据泄露风险	禁止将敏感信息放入对外广播的 Intent 中
	Intent 调用反射风险	对外部输入 Intent 数据的合法性做针对性校验
二进制	Native 库安全编译配置风险	在编译过程中开启安全编译选项
密码学	RSA 私钥硬编码风险	避免出现 RSA 私钥硬编码在客户端代码
	不安全随机数使用风险	使用安全的随机数种子
	固定初始化向量风险	初始化向量不固定
	加密模式使用错误风险	使用安全性较强的加密模式
	密码硬编码漏洞	避免出现密钥硬编码的情况，使用白盒密钥等方式进行缓解
	RSA 弱密钥风险	加强 RSA 的密钥强度，建议使用 1024 位以上长度的加密密钥
存储	文件全局可读风险	避免出现文件全局可读的配置，使用其他方式做数据共享
	Intent 调用路径穿越风险	对外部传入的参数做存储操作时进行文件路径的合法性校验
第三方库	第三方 SDK 漏洞	实时关注集成到应用中 SDK 的安全更新，保持 SDK 的版本更新
	第三方 Native 库漏洞	实时关注集成到应用中 so 库的安全更新，保持 so 的版本更新
网络通信	WebView SSL 证书错误忽略风险	在 WebView 访问链接发现 SSL 错误时，给出相应提示阻止访问继续或跳转到安全网页
	Socket 端口开放风险	对外开放的 Socket 端口在接收外部数据时，对数据做合法性验证或做身份验证
	证书弱校验风险	在 SSL 通信过程中严格实现证书合法性校验函数，发现非法证书时中断连接
	主机名弱校验风险	在 SSL 通信过程中严格实现主机名校验函数，发现非法域名时中断链接
Webview	WebView File 跨域风险	合理配置 File 域文件访问，针对 URL 是否支持 File 跨域做白名单校验
	WebView 回调风险	针对一些特殊的 URL WebView 会做特殊的回调处理，在做回调处理时对 URL 的合法性进行校验
	WebView 密码明文存储风险	尽量避免网页密码明文存储在 cookie 文件中，关闭 WebView 中密码存储的选项
数据库	SQL 注入风险	使用安全的 SQL 语句查询方式，避免出现命令拼接的形式
	SQLite 数据库全局可读写风险	使用加密的数据库存储方式，如 SQLCipher
其他	不安全插件加载漏洞	对外部加载的插件进行完整性校验，防止加载篡改后的插件
	解压缩路径穿越漏洞	对 Zip 文件进行解压时，校验文件名的合法性，排除 ../ 等形式的非法文件名
	Javascript Interface 接口开放风险	对 JS 访问 JavaScript Interface 做权限控制，避免不信任的页面访问敏感的 Interface 接口
	命令执行漏洞	对传入到命令执行函数的参数做严格的校验，防止出现任意命令的执行
	其他新攻击面	具体应对措施

通过结合待检测应用功能特性，自定义修改移动应用安全自查雷达图中不同攻击面的检测项，Android 应用开发厂商或者应用下载平台可以全面、客观、高效地总结当前移动应用静态检测安全风险。

Android应用安全检测趋势

近年，Android 应用安全受到越来越多重视，但依然面临巨大的挑战。安全检测作为提升应用安全性的重要手段以及安全管理的重要环节，也在与层出不穷的攻击手段对抗中演进发展。反汇编、反编译能力不断提升，数据流、控制流信息优化提取，检测模式组合集成扩展检测内容，安全检测也进一步与技术趋势紧密结合并更高效地输出结果。

Android 应用功能后台安全性不容忽视，此外还需要充分重视移动应用与服务器结合的安全性。随着业务实现重心的转移，不能仅仅局限于单客户端或者单服务器的安全检测，而需要把各个环节串联起来对整体进行全面、细致的检测。

在大数据时代，移动应用作为数据的重要来源，对移动应用数据安全与隐私保护的需求也日趋突出。对于隐私数据安全防护应该贯穿应用整个生命周期。国家从立法的角度督促隐私数据使用规范化，开发厂商也应做好自查并提高防范和抵御安全风险能力，应用商店也需要依照标准严格审查相关行为，用户也需要提升隐私数据保护意识，各个角色共同维护数据安全。

同样，人工智能和机器学习技术也在大数据时代迅猛发展。大量恶意样本以及漏洞特征的积累，为人工智能和机器学习技术在移动应用安全检测领域的落地提供了数据基础。通过这些技术，不仅可以在代码审计、安全评估等环节切入提高安全检测结果的准确率，还可以在安全风险验证环节提高效率。

提高检测结果准确率，减少无价值风险的误报一直是移动应用各类检测工具的瓶颈。尽管代码层面的分析可以一定程度定位潜在风险位置，但触发路径以及触发方式却仍然需要进一步耗时费力确认。如何充分利用动静检测方式优势，并提升自动化检测能力，也需要不断摸索和优化。

腾讯安全科恩实验室在移动应用安全研究领域有十多年的积累，技术实力和研究成果达到了国际领先水平。在保持对前沿技术研究能力的同时，腾讯安全科恩实验室也通过产品化输出移动应用安全检测能力。ApkPecker 作为一款全自动 Android 应用漏洞扫描工具，能够输出高质量漏洞扫描报告，提供高品质漏洞信息以及漏洞触发完整路径，精准定位漏洞并提供修复建议，助力移动安全人员提升应用安全性。

腾讯安全科恩实验室始终不遗余力地向全行业开放核心安全技术与能力，为各行业数字化转型变革的安全和健康发展贡献力量。

参考资料

参考来源：

[1] We Are Social and Hootsuite’ s latest collection of Global Digital 2019 reports

https://wearesocial.com/blog/2019/01/digital-2019-global-internet-use-accelerates

[2] App Annie，2019 年移动市场报告

https://www.appannie.com/cn/insights/market-data/the-state-of-mobile-2019/

[3] 凯度移动通信消费者指数，智能手机操作系统市场份额

https://www.kantarworldpanel.com/cn/smartphone-os-market-share/intro

[4] 中国互联网络信息中心，第 43 次《中国互联网络发展状况统计报告》

http://www.cac.gov.cn/2019-02/28/c_1124175677.htm

[5] G Data，Cyber attacks on Android devices on the rise，

https://www.gdatasoftware.com/blog/2018/11/31255-cyber-attacks-on-android-devices-on-the-rise

[6] ingridlunden，Android security: 0.04% of downloads on Google Play in 2018 were‘potentially harmful Apps’

https://techcrunch.com/2019/04/01/android-security-0-04-of-downloads-on-google-play-in-2018-were-potentially-harmful-Apps/

[7] Insignary.Inc，Comprehensive Android Binary Scans Find Known Security Vulnerabilities in 1 Out of Every 5 of the 700 Most Popular Apps on Google Play Store

https://www.globenewswire.com/news-release/2018/04/24/1485844/0/en/Comprehensive-Android-Binary-Scans-Find-Known-Security-Vulnerabilities-in-1-Out-of-Every-5-of-the-700-Most-Popular-Apps-on-Google-Play-Store.html?culture=en-us

[8] Google，应用安全改进计划

https://developer.android.com/google/play/asi

信息安全顶级服务

护航产业数字化变革 守护全网用户信息安全

产品、行业解决方案及商务合作请联系

KeenSecurityLab@tencent.com