软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software, [doi: 10.13328/j.cnki.jos.000000]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

面向 Android 生态中的第三方 SDK 安全性分析*

马凯[1]，郭山清[2]

[1](山东大学 计算机科学与技术学院,山东 济南 250101)

[2](山东大学 密码技术与信息安全教育部重点实验室,山东 济南 250101)

通讯作者：马凯, E-mail: makaisdu@gmail.com

摘 要: 现如今，许多 Android 开发人员为了缩短开发时间，选择在其应用程序中内置第三方 SDK。第三方 SDK 是一种由广告平台，数据提供商，社交网络和地图服务提供商等第三方服务公司开发的工具包，它已经成为 Android 生态系统的重要组成部分。令人担心的是，一个 SDK 有安全漏洞，会导致所有包含该 SDK 的应用程序易受攻击，这严重影响了 Android 生态系统的安全性。因此，我们在市场上选取了 129 个流行的第三方 SDK 并对其安全性进行了全面分析。为了提高分析的准确性，我们将第三方 SDK 的 demo 应用作为分析对象并使用了在分析 Android 应用中有效的分析方法（例如静态污点追踪、动态污点追踪、动态二进制插桩等）和分析工具（例如 flowdroid、droidbox 等）。结果显示，在选取的这些 SDK 中，超过 60%含有各种漏洞（例如：HTTP 的误用, SSL/TLS 的不正确配置, 敏感权限滥用，身份识别, 本地服务，通过日志造成信息泄露，开发人员的失误）。这对于相关应用程序的使用者来说都造成很大的威胁。

关键词: Android；第三方 SDK；安全性分析；漏洞检测
中图法分类号: TP311

Risk Analysis of the Third-party SDKs in the Android Ecosystem

Ma Kai[1] , Guo Shanqing[12]

1（College of Computer Science and Technology, Shandong University, Jinan 250101）

2 （Laboratory of Network and Information Security（Shandong University）, Jinan 250101）

Abstract: To shorten the application development time, many Android developers include third-party SDKs in their apps. Thirdparty SDKs are toolkits developed by third-party service companies such as advertising platforms, data providers, social

* 收稿时间: 0000-00-00; 修改时间: 0000-00-00; 采用时间: 0000-00-00; jos 在线出版时间: 0000-00-00
CNKI 在线出版时间: 0000-00-00

network, and map service providers. These third party SDKs has become an important part of the Android ecosystem. If an SDK contains security vulnerabilities, all the apps that include it would become vulnerable, which affects the security of the Android ecosystem severely. Therefore, we select 129 popular third-party SDK in the market and make comprehensive analysis of their security. In order to improve the accuracy of the analysis, we take demo apps of third-party SDKs as analysis object and make use of effective android-app analysis methods (such as static taint tracking, dynamic taint tracking, dynamic binary instrumentation etc.) and analysis tools (such as flowdroid, droidbox etc.).The result shows that more than 60% of the collected third-party SDKs contain various of vulnerabilities(e.g..Misuse of HTTP, Misuse of SSL/TLS, Abuse of sensitive permissions, Identification, Vulnerabilities brought by the Local Server, Information Leakage Through Logging, Mistakes of Applications Developers), which is a threat to the related applications and the users of these applications.

Key words: Android; third-party SDK; security analysis; vulnerability detection

随着 Android 市场份额的快速增长，Android 应用程序开发人员的数量也增长迅速[40]。为了缩短开发时间和提高开发效率，Android 应用开发商将多种类型的第三方 SDK(软件开发工具包)集成到他们的应用程序中。这些 SDK 是由广告，数据，社交网络，地图和推送平台等第三方服务提供商所开发的工具包[28]，可以提供专业的服务，其中封装了复杂的逻辑实现以及请求响应的过程，使其更便于开发人员使用。不难看出，第三方 SDK 已经成为 Android 生态系统的重要组成部分。

在应用中集成第三方 SDK 的优势是显而易见的。首先，应用程序可以获得专业公司在各个领域提供的高质量资源，例如地图（如 GoogleMap[29]），交通数据，天气数据等。其次，如果应用程序将类似 PayPal[30]的第三方 SDK 包含在内，那么它就可以执行像付款这样复杂的功能。这些 SDK 有助于提高应用的开发效率，并且实现更加健壮，尤其是对于小型的开发团队。另外，广告平台（例如 AdMob[27]）这种第三方 SDK 可以帮助应用程序的开发者获得收益。

然而与此同时，第三方 SDK 也会对 Android 用户的隐私和安全性产生威胁。许多研究证实[1][2][3][4][5]，一些第三方 SDK 存在隐私泄露问题。Taomike（中国最大的移动广告提供商）和百度提供的第三方 SDK 被曝出存在安全漏洞，这两个软件秘密监视用户，将短信上传到远程服务器，并在用户的设备上开启后门[6][7]。不幸的是，这些 SDK 已经集成到众多 Android 应用中，影响了超过 1 亿的 Android 设备用户。除了侵犯用户隐私，有些第三方 SDK 还会采取不安全的实现方式，增加其宿主应用程序的攻击面，从而对用户安全造成威胁。最近，甚至是 Facebook 和 Dropbox 这种信誉良好的软件公司的 SDK 也被发现存在严重的安全漏洞。这些漏洞带来的攻击包括：将敏感数据泄露到公开可读的数据源[8]，代码注入攻击[9][10]，帐户劫持[11]，将受害者设备连接到攻击者控制的 Dropbox 帐户[12][13][14][15]等。

不难看出，减少第三方 SDK 中的安全问题将有效缓解 android 用户所面临的安全风险。然而，对第三方 SDK 进行系统性的安全分析不是一件轻松的工作。一方面，我们只能在官网上找到最新版本，但是它们的旧版本仍然在大量的应用中使用。另一方面，不同类型的第三方 SDK 具有不同的架构，这使得设计出一个分析 SDK 的系统是十分困难的。另外，我们发现很多的第三方 SDK 将其关键代码放到.so 中，这增大了逆向工程的难度。

本文着力于对 Android 生态系统中较流行第三方 SDK 的安全性进行全面分析。由于具有网络通信能力的 SDK 更加重要，更加流行，所以我们把焦点放在这种类型的 SDK 上面。本文汇总了 129 个第三方 SDK 包含广告、数据、推送、登录等 10 个类型，并且对每个类型中流行的第三方 SDK 进行了深入分析。在分析过程中，我们使用了有效的 android 应用分

析方法（例如静态污点分析、动态污点分析和动态二进制插桩）和分析工具（例如 flowdroid、taintdroid 和 frida）。贡献如下：

4. 本文系统分析了具有网络通信能力的流行第三方 SDK，并将其总结为两种基本类型：在应用端开启本地服务器的 SDK 和无本地服务器的 SDK。
5. 本文对具备网络通信能力的 SDK 的安全风险进行了首次全面分析。我们收集了 129 个知名的第三方 SDK，通过静态分析，从其文档和源代码中提取相关信息（例如组件，权限，网络连接），然后检测可能存在的漏洞并通过动态测试进行验证。
6. 经过分析，我们将在第三方 SDK 中发现的弱点总结为六种类型，这些类型将在第四节中介绍。我们的分类非常全面，为开发人员和服务提供商提供了分析第三方 SDK 安全风险的依据。我们从这些 SDK 中找到并验证了几个意料之外的漏洞。除此之外，本文揭露了较流行的第三方 SDK 中存在的尚未被发现的安全弱点。研究结果表明，第三方 SDK 面临许多安全问题，这些问题给大量应用程序以及用户带来威胁。

本文的其余部分安排如下：第一节描述第三方 SDK，研究具有网络连接功能的 SDK 的操作机制，提出分析和调查第三方 SDK 的动机。第二节详细介绍我们使用的方法以及分析过程。第三节介绍了我们在分析中发现的在第三方 SDK 中普遍存在的漏洞。第四节讨论了 SDK 中存在的安全问题所带来的影响，并提出一些补救措施。

6 背景和动机

在 android 生态系统中，第三方 SDK 是指由第三方服务公司提供的具有特定功能的软件开发工具包，例如广告，推送，图像识别技术，移动支付技术等。为了提高应用开发效率，开发人员将第三方 SDK 集成到项目中来实现这些功能，包括广告，第三方登录，转发评论，虚拟现实等。不难看出，大多数第三方 SDK 只是第三方服务的客户端。当这些 SDK 被调用时，它们需要连接远程服务器来提供服务。

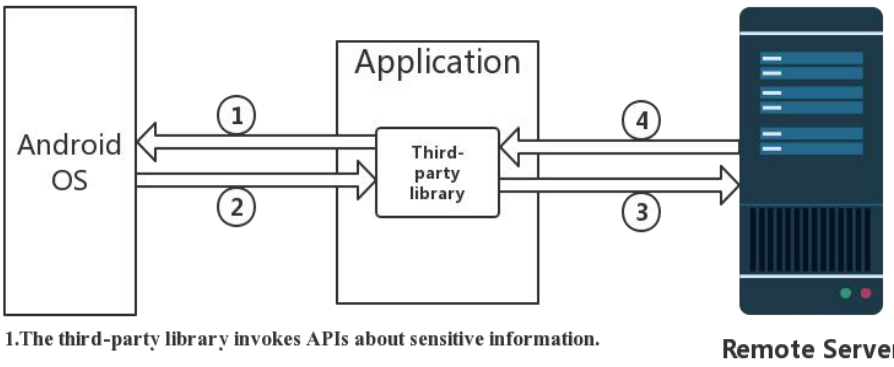
本文重点讨论具备网络通信能力的第三方 SDK。根据他们的功能，我们进一步将 SDK 分为以下几类：广告服务，支付平台，推送消息服务，第三方登录服务，监控工具，地图服务，数据服务，社交评论平台，社交共享平台。针对每种类型，我们下载流行的第三方 SDK，分析它们的运行机制，并检测安全漏洞。

6.1 运行机制

如第一节所述，本文专注于具有网络连接功能的第三方 SDK。我们分析了这些 SDK 的运行机制，并总结为两种类型：

（3） 第三方 SDK 向远程服务器发送请求：图 1 展示了这种类型 SDK 的运行机制。例如提供广告服务，推送消息服务的第三方 SDK 采用这种机制。由于 HTTP 没有加密传输数据，因此不能保证数据隐私以及完整性。HTTPS（超文本传输安全协议）是一种用于在不可信网络上实现安全连接的网络通信协议，它将 SSL / TLS 的安全功能添加到标准 HTTP（超文本传输协议）通信中，因此只要正确实现并配置，就可以防止窃听和中间人攻击。使用 HTTPs 替代 HTTP 已成为一种趋势，然而仍有许多第三方 SDK 使用 HTTP 协议通过网络连接到云服务器，这意味着很高的安全风险。

（4） 第三方 SDK 启动本地服务：有些第三方 SDK 在主机应用程序中设置本地服务器，来确保这些 SDK 的制造商可以可控地监视移动设备。本地服务器能够收集当前设备中的位置，IMEI，安装信息等信息，而远程服务器可以发送请求并从本地服务器检索这些信息。甚至是远程安装与卸载也可以通过与本地服务通信来实现。图 2 描述了这种类型的 SDK 的工作机制。虽然我们可以采取一些措施来使得本地服务器由正确的远程服务器控制，但是我们的通信信道可能被攻击者劫持，这使得攻击者能够与本地服务器交互。我们将在第 2 节给出一个劫持攻击的真实例子。



1.The third-party library invokes APIs about sensitive information.

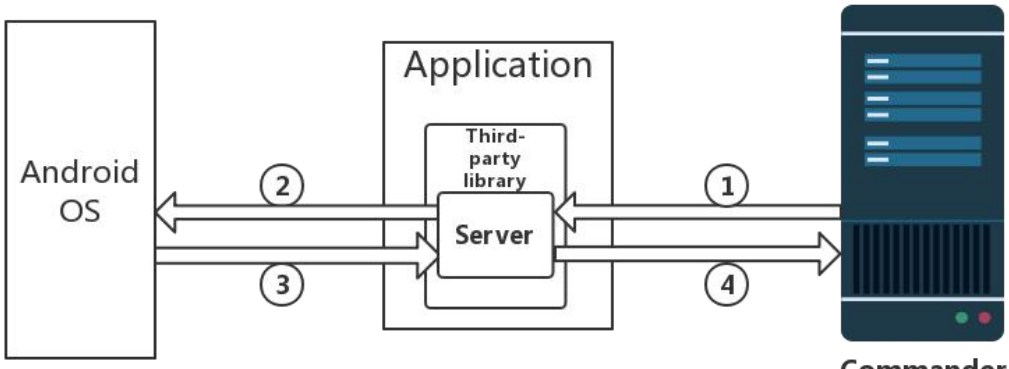
2.Android OS returns the sensitive information.

3.The third-party library requests an advertisement with device ID, package name using HTTP or HTTPS.

4.The server responses data about the advertisement using HTTP or HTTPS.

Fig 1 The third-party SDK initiates a request to the server

图 1 第三方 SDK 向服务器发送请求



1.The commander sends command to the server of the third-party library.

2.The server checks the received command and invoke APIs to execute the command.

3.Android OS returns the related information.

4.The server returns result of executing the command.

Fig 2 The localserver launched by the third-party SDK

图 2 第三方 SDK 运行的本地服务

6.2 动机

许多移动应用开发人员将第三方 SDK 视为黑盒，只关注他们的功能，因此容易忽视内部的安全漏洞。显然，使用具有安全漏洞的 SDK 会使应用程序易受攻击。举例来说，如果

一种 SDK 中存在后门，则使用此 SDK 的所有项目都会有该后门，这使得攻击者可以非法访问应用程序的核心代码。第三方 SDK 具有与其主机应用程序相同的权限，因此即使应用程序的核心功能不需要，应用程序也必须请求 SDK 所需的权限。SDK 中存在安全漏洞会造成很大的危害，这会使包含这些 SDK 的应用程序都受到影响。本节的剩余部分给出了一些易受攻击的 SDK 的例子，这些例子是本文工作的动机。

广告 SDK：Kugou 是移动广告平台类的 SDK，它已经被开发商集成到许多应用程序中来获利。Kuguo 的官方网站宣布，智能手机用户数已达 3.6 亿户，有 71042 种移动应用正在使用 Kuguo 平台。在官网上下载到 Kugou 的文档以及 SDK 文件非常容易。通过对这些文件的分析（详细分析过程将在第 2 节中给出），我们可以了解这个 SDK 的运行机制，如图 3 所示。我们发现 Kuguo SDK 使用 HTTP 协议与远程服务器通信，因此我们设置了代理服务器来检测 HTTP 通道上的数据包传输，请求和响应数据如图 3 所示。我们可以替换响应中的数据，例如将 URL 替换为我们构建的钓鱼网站，当用户点击广告时，假冒的 URL 将被打开并显示在浏览器中。攻击者可以利用此钓鱼网站对用户采取进一步的攻击。

Fig 3 Request data and Response data

图 3 Kuguo SDK 的请求和返回数据(在请求数据包中可以看到 IMEI、包名等信息。在返回数据包中可以看到广告的链接和广告图片)

中国银行登录 SDK：中国银行的登录 SDK（BOCOPPaySdk）使用 HTTPS 与远程服务器通信。如图 4 的代码所示，登录 SDK 无法验证服务器发送的证书，因此攻击者有机会启动中间人攻击拦截证书并用伪造的证书来替换。在实验中建立了一个代理服务器并使用伪造证书对服务器所返回的证书进行了替换实现中间人攻击。

```
TrustManager tm = new X509TrustManager()
{
  public void checkClientTrusted(X509Certificate[] chain, String authType) throws CertificateException
  {
  }

  public void checkServerTrusted(X509Certificate[] chain, String authType) throws CertificateException {
  }

  public X509Certificate[] getAcceptedIssuers() {
    return null;
  }
};
this.sslContext.init(null, new TrustManager[] { tm }, null);
```

Fig 4 Certification check

图 4 证书查验实现，该 SDK 中的证书查验函数实现为空，可以被中间人通过替换证书进行攻击。

具有本地服务器的第三方 SDK：Moplus 是由中国搜索引擎巨头百度开发的 SDK。由于我们无法下载 Moplus 的原始文件，所以我们选择分析包含它的应用来研究其权限和行为。经过对主机应用的 manifest 文件进行解析，我们发现 SDK 在一个单独的进程中，其主要服务被称为 com.baidu.android.moplus 该服务可以通过不同的广播来触发，例如系统启动的广播。

当用户启动包含 moplus SDK 的应用程序时，此 SDK 将自动并秘密地设置本地 HTTP 服务器，以便通过 socket 消息进行监视。moplus 通过内置 NanoHttpd 来达到这一目的，NanoHttpd 是由 JAVA 编写的一个开源 HTTP 服务器。HTTP 服务器不断监听 TCP 端口，接收并解析从远程服务器或客户端发送的消息。一旦有新的 HTTP 请求，本地服务器将用接收到的消息替换 NanoHttpd 服务器的 action，然后开始执行恶意操作。远程服务器可以发送请求来获取位置信息，搜索框信息，包信息和用户设备的其他敏感数据。此外，攻击者可以向用户的设备添加联系人，扫描下载的文件，并上传特定文件。所有这些操作都可以通过简单地发送 HTTP 请求来完成。Sendintent 是一种特殊的命令，用于向主机的应用程序发送指令，可以在未经用户同意的情况下进行远程调用，发送错误消息并安装任何应用程序。本地服务不会对请求消息进行身份验证，因此可以由攻击者触发而不需通过远程服务器。此外，攻击者可以使用 Nmap 扫描所有 Android 设备的 TCP 端口 40310 并检查端口是否打开。

7 第三方 SDK 的选择和分析：

本节将介绍如何选择和分析第三方 SDK。

7.1 第三方SDK的选择：

从 Android 应用程序中提取第三方 SDK 不是一个明智的选择。首先，第三方 SDK 在应用程序构建过程中被静态链接到应用程序字节码中，这模糊了应用程序和 SDK 代码之间的界限,从而使 SDK 高度集成到应用程序中。其次，应用程序开发人员通常使用类似 ProGuard[16]的字节码混淆工具。标识符重命名是一种无副作用的字节码混淆技术。它将标识符转换成短而且无意义的字符串，如将报名 com.google 转换为 a.c.。还有一个问题是缺少对于隐私和安全违规的追责方法。举例来说，许多与安全相关的分析研究了应用程序中的隐私与安全问题，提高了人们对于各种问题领域的认识，包括隐私泄露[17][18][19][20][21]，权限使用[22]，动态代码加载[23]，SSL / TLS（内部）安全性[24][25]，加密 API 的误用[26]等。然而，这些报告不能区分应用程序开发人员代码和第三方 SDK 中的代码，而且分析结果是基于每个应用程序的，不能识别不正当行为到底是由应用程序开发人员还是第三方 SDK

开发人员造成的。

为了提高第三方 SDK 分析的效率，同时保证问题制造者（应用程序或开发者）对破坏安全或隐私负责，我们从官方网站收集了一些第三方 SDK 和相关应用程序。第三方 SDK 有几个热门的汇总网站，例如 AppBrain 和 SDK.cn。网站中有数十种流行的第三方 SDK 可用，但它们只是第三方 SDK 中的一小部分。AppBrain 专注于 Google Play，而 SDK.cn 更侧重中国市场。许多第三方 SDK 只在一个开发者小组或一个特定市场（北美，欧洲，亚洲等）流通。比较受欢迎的第三方 SDK 往往有多个版本。此外，一些第三方 SDK 对于普通开发人员不可用，而是仅针对他们自己的产品开发。

与 Android 应用程序可以从 Google Play 等市场上获取不同，第三方 SDK 一般由第三方供应商提供，因此在收集有些 SDK 并不容易。中国已经封锁了 Google Map 等多项服务，因此中国的开发商在 Android 应用市场开发了定制的 SDK 以及应用（例如百度地图）。考虑到中国智能手机市场份额较大，我们从中国市场和国际市场收集 SDK。针对在第 1 节中介绍的每个种类，我们从中国市场和国际市场收集了尽可能多的流行 SDK。对于无法直接下载的 SDK，我们选择收集包含这些 SDK 的应用程序。

7.2 分析过程

分析过程包括三个阶段。第一阶段是浏览第三方 SDK 的开发清单，包括集成手册、demo 应用程序的代码和 manifest 文件。第二阶段是分析阶段，包含静态自动化分析、数据包抓取和动态污点分析三个方面。在静态自动化分析过程中，我们首先对第三方 SDK 的 demo 应用进行反编译，然后进行代码审计（例如 WebView 不安全 API 使用、SSL/TLS 配置）和污点分析（图 5），重点关注网络连接和敏感函数操作（例如获取通讯录）。在动态污点分析过程中，我们可以避免静态分析因无法真实运行而产生的误报和漏报（例如应用程序使用 Java 反射机制）。基于分析结果，我们推测第三方 SDK 可能存在的安全问题。第三阶段，我们使用动态二进制插桩等方法对可能存在的安全问题进行验证。

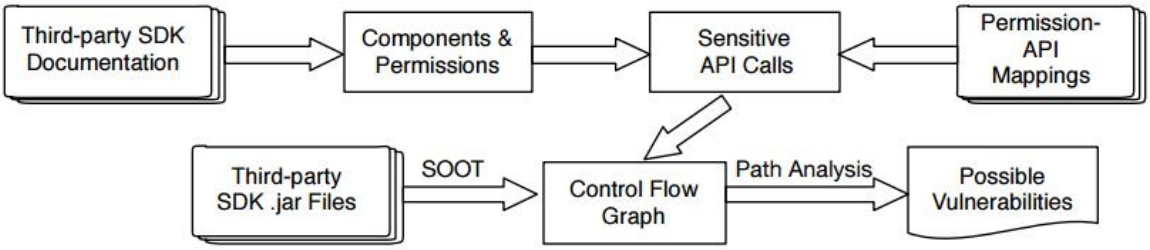


Fig 5 Static analysis of the .jar file of Third-party SDKs

图 5 对于第三方 SDKs 中的 jar 文件的静态分析

阶段一：浏览第三方 SDK 的开发信息

第三方 SDK 供应商提供文档或用户指南来帮助开发人员将 SDK 集成到应用程序中。我们可以从文档中提取关于将 SDK 集成到应用程序中时需要将哪些组件和权限添加到应用程序清单的信息。图 6 展示了应用程序引入 kuguo SDK 需要在 manifest 文件中添加的权限与组件。

```
<uses-permission android:name=
 "android.permission.INTERNET" />
<uses-permission android:name=
 "android.permission.WAKE_LOCK" />
<uses-permission android:name=
 "android.permission.READ_PHONE_STATE" />
<uses-permission android:name=
 "android.permission.ACCESS_WIFI_STATE" />
<uses-permission android:name=
 "android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name=
 "android.permission.WRITE_EXTERNAL_STORAGE" />

<activity android:name="com.xx.xx.Cka"
        android:excludeFromRecents="true"
            android:launchMode="singleInstance" />
<service android:name="com.xx.xx.Cks" />
<receiver android:name="com.xx.xx.Ckr">
   <intent-filter>
      <action android:name=
        "android.intent.action.PACKAGE_ADDED" />
         <data android:scheme="package" />
   </intent-filter>
</receiver>
```

Fig 6 Documentation of kuguo SDK

图 6 kuguo SDK 的文档

阶段二：静态自动化分析、数据包抓取和动态污点分析

在静态自动化分析中，关于 SSL / TLS 漏洞的自动检测，我们使用 Mallodroid 对于所有第三方 SDK 相关的 demo 应用进行自动分析。但是该分析仅提供了一个问题代码的基本指标，它既不能排除死码，也不能检测到设计中存在的一些主要缺陷。例如，它不能保证敏感数据流使用 SSL / TLS，同样无法检测到一些系统漏洞，包括在远程服务器上使用不被推荐的，易受攻击的或不正确的 SSL / TLS 配置。Mallidroid 存在所有自动化方法都会有的限制：在最好的情况下，自动化分析可以提供一个不完整的分析；但是在最坏的情况下得到的分析可能是不正确的。Mallodroid 的文章作者对其检测到的 1,074 个工具中的 100 个（9.3％）进行了手动分析，来验证它的发现，然而，只有 41％的应用程序容易受到 SSL / TLS 中间人攻击。因此，必须进一步验证此工具的调查结果以消除假阳性和假阴性。

对于每个第三方 SDK 相关的 demo 应用，我们对其使用的敏感 Android API（例如 getDeviceId()、sendtextmessage()）进行可达性的自动化分析（分析工具：flowdroid），检查潜在的危险行为。除此之外，我们还检查了反射，动态代码加载，权限探测，JavaScript 使用。SDK 使用反射需要使用 Java.lang.reflect 包，这允许对方法进行可编程调用并访问字段。由于静态分析无法处理使用反射 API，因此我们将通过动态的污点分析来弥补这个缺陷。

我们使用 Fiddler 对第三方 SDK 与服务器端的网路数据进行抓包，并且使用 Droidbox 进行动态污点追踪。

根据上述分析，我们可以深入了解第三方 SDK 与服务器端的通信机制和对敏感信息的使用。基于分析结果，我们可以推测出第三方 SDK 中可能存在的的安全威胁（第 3 节介绍了威胁的分类）。

阶段三：对在第三方 SDK 中所发现的安全问题进行验证

验证漏洞通过动态执行第三方 SDK 相关的示例应用来完成，该过程如图 7 所示。在与敏感信息和网络连接相关的代码周围添加日志指令可以检查这些代码在应用程序运行时是否会被执行。图 8 展示了以 smali 格式插入 log 指令，const-string v1 表示日志的标签，const-string v2 表示我们希望在 DDMS 中打印输出的信息。我们可以根据 DDMS 中输出的信息来验证之前的分析结果。

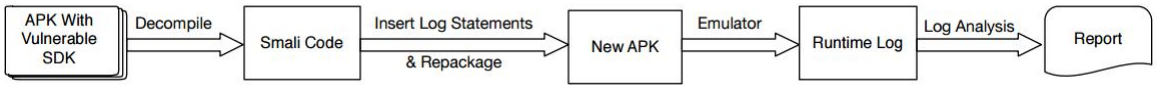


Fig 7 Injection of log statements

图 7 log 指令的注入

```
const-string v1, "injection"
const-string v0, "sensitive-information"
invoke-static {v1, v0},
Landroid/util/Log;
->d(Ljava/lang/String;Ljava/lang/String;)I
```

Fig 8 Injection of Log code in smali format

图 8 在 smali 格式下注入 log 代码

我们重新打包包含 SDK 相关的示例应用，将其在 Android 模拟器上运行，并将 Fiddler 设置为转发第三方 SDK 和远程服务器之间的流量的代理。如果没有正确配置 SSL / TLS，那么攻击者可以在 Fiddler 中创建假认证来替换真实认证。执行完后，我们使用 adb shell 来访问相关应用程序的私有目录。在这个目录下，我们可以获取数据库，lib，sharedPreference，缓存，文件，并且可以在数据库没有被加密的前提下查找数据库中的信息。

因为无法拿到第三方 SDK 的服务端代码，我们只能使用黑盒的方式对服务端进行测试。结合静态分析和动态二进制插桩（注入工具：Frida），我们可以通过代码注入的方式对 SDK 提交给服务端的参数进行修改，来验证安全问题。

分析方法	分析工具
静态污点分析	FlowDroid
动态污点分析	TaintDroid
动态二进制插桩	Frida
Fuzzing	Intent Fuzzer
动态 Java 代码注入	Xposed

Table 1 Analysis methods and tools

表 1 在分析过程中使用的分析方法和工具

8 结果：

我们的分析验证了第三方 SDK 中存在多个漏洞，我们将其分为六种类型。本节将详细介绍这些漏洞，并描述一些易受攻击的 SDK 的示例。表 2 给出了对于分析结果的总结。

8.1 V1：滥用HTTP：

虽然使用 HTTP 协议进行网络连接已被认为是不安全的，但是我们发现许多第三方 SDK 仍然使用此通道与远程服务器进行通信。更糟糕的是，一些重要数据还通过 HTTP 通道以明文或密文的形式传输，如 IMEI（International Mobile Equipment Identity）。

明文：如第 1 节所述，广告平台 Kuguo 使用 HTTP 以明文形式传输敏感数据。攻击者可以劫持这个通道来检索敏感数据。我们总共分析了 129 个第三方 SDK，其中 35 个 SDK 包含此漏洞。它的影响及其解决方案已被广泛讨论，因此在这里我们不再赘述。

密文：在我们收集的第三方 SDK 中有 7 个将通过 HTTP 通道传输的数据进行加密。问

题是他们使用自定义加密系统进行加密，加密密钥是本地生成的，而不是通过与远程服务器密钥协商生成（如 HTTPS 协议）。加密算法和加密密钥写在 SDK 的.so 文件中，攻击者可以通过解析.so 文件来破解加密算法，这使得应用程序的完整性及其隐私保证受到威胁。

Juhe SMS 验证 SDK 是一种能让主机应用程序执行 SMS 消息验证的第三方 SDK。经过分析，我们发现它使用自定义的对称加密算法来加密消息。图 9 描述了当 Juhe SMS 的方法被主机调用时发送的 POST 数据。数据由 Juhe SMS 加密，但它没有关于生成加密密钥的信息，此外，加密密钥在本地生成，而不是通过与远程服务器的密钥协商生成（如在 HTTPS 协议中）。Juhe SMS 的.jar 文件调用.so 文件的加密功能。我们可以修改.so 文件，打印出对数据进行加密的密钥。

```
POST http://112.124.2.236/op/mob/tk_qvcbpn.php HTTP/1.1
Content-Length: 160
Content-Type: text/plain; charset=UTF-8
Host: 112.124.2.236
Connection: Keep-Alive
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1;
zh-CN; rv:1.9.2) Gecko/20100115 Firefox/3.6

3ABAA13C165CE637F85F1545A293686B89C537881F3D49FF0C02EF7
8F3520B899F2C66E742F31A98BA3FED17A47763780AC6D709DEE863
46E6F40D8ED95A5CF510831F8BFB92E67AF16E18A6E849981A
```

Fig 9 ”Encrypted message” sent by Juhe SMS

Juhe SMS 所发送的加密信息

8.2 V2：滥用SSL / TLS

HTTPS（SSL / TLS 上的 HTTP）只有在恰当的实现和配置下才会使通信信道安全。要想建立安全的 SSL / TLS 连接，客户端必须检查证书链和主机名是否有效。如果主机名与服务器的域名匹配，则该主机名有效。如果证书链符合以下要求，它被认为有效：(1）链中的每个证书都没有过期或撤销。(2）根证书由 CA 在客户端的密钥库中发起。(3）在多于一个证书的情况下，每个证书必须在放入链中后立即由 CA 签名。[39]系统地研究了不正确的 SSL / TLS 证书验证过程所带来的威胁。通过分析，我们发现这些威胁在第三方 SDK 中也很常见。

Android 中的 X509TrustManager 接口可以作为 X509 证书的信任管理器来为安全的 sockets 执行身份验证。开发人员可以实现 X509TrustManager 接口，重写证书验证过程来代替库中的实现。在分析第三方 SDK 的过程中，我们发现这些方法的例程是空的，这意味着它们什么也不做，即使在出现非法证书的情况下也不会抛出异常。此外，虽然一些 SDK 执行证书验证，但即便是证书过期或被吊销，它们也没有抛出异常。这种类型的漏洞在第三方 SDK（例如广告平台和推送式消息平台）中相当普遍。根据分析结果，有 20 个 SDK 包含与 SSL / TIS 相关的问题。

8.3 V3：滥用敏感权限：

通常情况下，Android 应用程序会请求比所需要的更多的权限。他们使用额外的权限来窥探用户的隐私信息，甚至植入恶意背景的插件。我们的分析显示 16 个 SDK 有上述恶意行为。当应用程序开发人员将第三方 SDK 加入到应用程序中时，会将某些权限，组件，数据等信息添加到 manifest 文件中。

Umeng 是一个推送消息 SDK，可以请求用来发送 SMS，读取 SMS 和接收 SMS 的权限。在对其他推送消息 SDK 分析之后，我们认为这些权限对于核心功能来说并不是必要的。

另外，第三方 SDK 可以与主机应用程序共享 manifest 文件中的权限，也就是说，即使 SDK 在开发文档中没有声明需要某些权限，如果 manifest 文件声明，它就可以使用这些权

限。这些 SDK 利用代码来检查宿主应用程序中否请求了某个权限（执行此检查的代码示例如图 10 所示）。

```
PackageManager pm = getPackageManager();
boolean permission =
     (PackageManager.PERMISSION_GRANTED ==
pm.checkPermission(
 "android.permission.RECORD_AUDIO", "packageName"));
if (permission) {
......
}
else {
......
}
```

Fig 10 Permission Check in Android App

图 10 Android 应用中的权限检查

8.4 V4：身份识别：

推送消息 SDK 是第三方 SDK 中一个比较常见的类型，它能够帮助移动应用程序开发商向在用户设备上运行的 APP 传递消息和通知。推送消息 SDK 的结构如图 11 所示。找到这个服务的结构并不困难，但是因为该服务需要协调开发人员与应用之间的交互，这使得它容易出错。

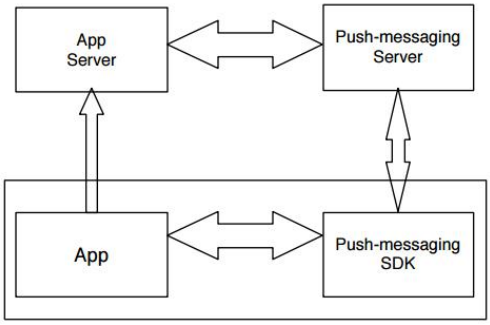


Fig 11 Push-messaging SDK structure

图 11 推送 SDK 的结构

由 Google 提供的 Google Cloud Messaging（GCM）SDK 被许多应用程序订阅，包括 Facebook，Oracle，Skype 等，它的运行机制类似于 Apple Push Notification Service。据报道，一些网络犯罪分子使用 GCM 来控制恶意软件[45]。除了 Google 和苹果之外，还有许多其他的第三方推送消息服务提供商都为应用程序开发人员提供 SDK。

华为 Push-messaging SDK：Push-messaging SDK 的云服务器通常利用注册 ID 与相应的应用程序进行通信。不幸的是，我们的分析显示注册 ID 会泄露用户的敏感信息。经过分析，我们发现华为 Push-messaging SDK 所使用的注册 ID 是基于其国际移动设备标识(IMEI)，国际移动用户身份（IMSI），MAC 地址和包名称在设备上确定生成的。因此，当我们卸载带有推送消息 SDK 的应用程序，然后再在同一个设备上重装后，该应用程序的注册 ID 不会改变。如果设备的敏感信息暴露，攻击者可以根据它计算注册 ID。由于 SDK 通过注册 ID 来识别特定设备上的应用，所以，如果攻击设备计算出 ID 并且具有受害者设备，那么它就有权从远程服务器接收推送消息。为了防御这种攻击，我们应该增加随机性，例如引入盐值，使得注册 ID 可以不确定地产生。

另外，华为 Push-messaging SDK 生成的注册 ID 经对称加密算法（AES）加密后，存储在共享配置中。然而，加密密钥是在本地生成，而不是通过与远程服务器的密钥协商来生成（如在 HTTPS 协议中），攻击者可以通过逆向工程获得关键字以及偏移向量，这有助于解密存储在共享配置中的注册 ID。

8.5 V5：本地服务器带来的漏洞

如第 2 节所述，具有本地服务器的第三方 SDK 可以收集设备信息，进而获得设备的控制权。如果本地服务器不适当执行访问控制，攻击者就可以访问它，检索敏感数据，甚至操纵设备。第 2 节提到，moplus SDK 无法验证请求 URL，因此它可能会被网络犯罪分子触发，这会给包含此 SDK 的所有应用程序带来巨大威胁。

Baidu Map 是一个著名的移动应用，可以提供在线和离线地图搜索服务。由于中国用户无法访问 Google 地图服务，因此百度地图在中国被广泛应用。第 2 节描述了可能的攻击细节。此外，百度及其他公司开发的约 3000 款应用程序的某些版本中包含该 SDK，其中有电话助手，输入，云端，浏览器和视频等。虽然其中一些应用的最新版本已删除此 SDK，但仍有大量用户在使用易受攻击的版本。实际上，我们使用 Nmap 扫描网络上许多设备的 TCP 端口 40310，发现其中许多设备仍然打开此端口。

事实上，还有一些其他的具有本地服务器的第三方 SDK 以及许多大众应用程序包含这些 SDK，如 Gaode Map，腾讯和 360。每个相关的应用程序都拥有大量的用户，而且这些应用的旧版本仍然在很多设备上运行。

8.6 V6：未关闭日志造成的信息泄漏

Android 日志系统为开发人员提供了记录应用程序和设备运行状态的接口。日志消息被写入设备的内部存储中。开发人员通常使用 android.util.log 打印调试信息，但是如果他们在应用上线前未关闭日志，这将会成为安全风险。在开发中，开发人员通常使用 debug 属性 该代码确定是否输出日志（图 12），这使我们很容易修改调试属性。在 Android 4.1 版本之前，具有 READ_LOGS 权限的 Android 应用程序能够读取设备上所有应用程序的日志文件。因此，将敏感数据写入日志会导致敏感数据泄露。在分析中，我们发现 mapbar SDK（专业的电子地图提供商）会将个人身份信息如 IMEI 通过日志进行记录。在我们分析的 129 个第三方 SDK 中， 其中 12 个包含此漏洞。

```
<application android:allowBackup="true"
             android:debuggable="true"
             android:icon="@drawable/ic_launcher"
             android:label="@string/app_name"
             android:theme="@style/AppTheme
 ......
```

Fig 12 Attribute about log in the AndroidManifest.xml

图 12 AndroidManifest.xml 中的有关 Log 的属性

8.7 应用程序开发人员的失误

（1）uid 误用

一些社交平台如 Facebook，Twitter，新浪微博等提供了 SDK 用于第三方登录，这可以帮助用户快速完成登录或注册过程，无需为当前访问的应用程序注册新帐户。这些 SDK 使用 OAuth2.0 协议对用户的账户进行身份验证。如果用户通过认证，SDK 的服务器将返回访问令牌和 uid （用户在该平台上的唯一标识）到当前应用程序的服务器。之后，应用程序可以使用访问令牌和 uid 访问用户授权的资源。然而一些应用程序开发人员只使用 uid 作为用户的凭证，在这种情况下，攻击者可以拦截 uid 并将其篡改为指定 uid 进行登录。

（2）使用不安全的 API

当第三方 SDK 在 WebView 中使用 JavaScriptInterface，远程 Web 页面可以通过这

个接口执行本地命令。当 WebView 显示页面时会在 JavaScript 代码中调用本地代码。远程网页可以利用反射机制来执行自己的命令（图 13）。

```
function execute()
{
  return aboj.getClass()
              .forName("java.lang.Runtime")
            .getMethod("getRuntime",null)
              .invoke(null,null)
            .exec(cmdArgs);
}
```

Fig 13 A JavaScript code to execute the local commands

图 13 JavaScript 代码执行本地命令

9 讨论

（1）第三方 SDK 漏洞的影响

越流行的第三方 SDK，如果存在漏洞，造成的安全威胁越大。例如，Moplus SDK 影响大约 3,000-4000 个下载次数达数百万的应用程序。由于第三方 SDK 是由第三方服务商各自维护，安全水平层次不齐。所以开发者应该注意引入第三方 SDK 时可能存在的安全风险。

（2）证书验证

我们的分析结果表明 SSL / TLS 漏洞在第三方 SDK 中很常见。

（3）第三方 SDK 的保护机制

开发人员可以通过以下方式保护 SDK 的代码 ：代码混淆；使用 Java 反射机制；将关键代码放在.so 文件中；SSL / TLS 应正确配置；请勿 root 您的设备。

代码混淆，使用 Java 的反射机制和将关键代码放在.so 文件使得静态分析第三方 SDK 变得更困难。SSL/TLS 正确配置能够确保通信安全。已 root 与未 root 的设备相比所面临的安全风险大大增加。

The Third-party SDKs	Numbers	V1-1	V1-2	V2	V3	V4	V5	V6
Advertising Service	21	12	4	3	2	0	0	0
Payment platform	10	0	0	0	0	0	0	0
Push-messaging	16	0	0	6	4	4	0	0
Third-party Login	23	0	0	5	2	3	0	2
Monitoring tool	11	4	0	0	0	0	0	4
Map	11	4	0	0	2	0	0	1
Data Service	6	3	2	0	1	2	0	1
Social comments	10	3	2	2	2	1	0	2
Face Recognition	7	4	0	0	2	0	0	0
Others	14	5	2	4	1	2	5	2

Sum	129	35	10	20	16	12	5	12

Table 2 Security Vulnerabilities of Different Types of third-party SDKs

表 2 对于收集的 129 个第三方 SDK 所存在的漏洞分布

10 结论

在本文中，我们分析了在 Android 生态系统中具有网络通信功能的第三方 SDK 存在的常见安全隐患。结果显示，在选取的这些 SDK 中，超过 60%含有各种漏洞（例如：HTTP 的误用, SSL/TLS 的不正确配置，敏感权限滥用，身份识别，本地服务，通过日志造成信息泄露，开发人员的失误）。我们希望我们工作可以引起第三方服务商的关注 专注于其 SDK 的安全问题。

References:

[47] T. Book, A. Pridgen, and D. S. Wallach. Longitudinal analysis of android ad library permissions. In MoST13. IEEE, 2013.

[48] W. Enck, D. Octeau, P. McDaniel, and C. Swarat. A study of android application security. In USENIX Security11. USENIX, 2011.

[49] M. Grace, W. Zhou, X. Jiang, and A.-R. Sadeghi. Unsafe exposure analysis of mobile in-app advertisements. In WISEC12. ACM, 2012.

[50] J. Seo, D. Kim, D. Cho, T. Kim, and I. Shin. FlexDroid: Enforcing In-App Privilege Separation in Android. In NDSS16, 2016.

[51] R. Stevens, C. Gibler, J. Crussell, J. Erickson, and H. Chen. Investigating user privacy in android ad libraries. In MoST12. IEEE, 2012.

[52] The Hacker News. Warning: 18,000 android apps contains code that spy on your text messages. http://thehackernews.com/2015/10/android-appssteal-sms.html. Last visited: 04/27/16.

[53] The Hacker News. Backdoor in baidu android sdk puts 100 million devices at risk. http://thehackernews.com/ 2015/11/androidmalware-backdoor.html. Last visited: 04/27/16.

[54] Parse Blog. Discovering a major security hole in facebooks android sdk. http://blog.parse.com/learn/engineering/discoveringa-majorsecurity-hole-in-facebooks-android-sdk. Last visited: 04/27/16.

[55] S. Poeplau, Y. Fratantonio, A. Bianchi, C. Kruegel, and G. Vigna. Execute This! Analyzing Unsafe and Malicious Dynamic Code Loading in Android Applications. In NDSS14, San Diego, CA, 2014

[56] Vungle Support. Security vulnerability in android sdks prior to 3.3.0. https://support.vungle.com/hc/en-us/ articles/205142650-Security-Vulnerability-in-AndroidSDKs-prior-to-3-3-0. Last visited: 05/02/2016.

[57] The Hacker News. Facebook sdk vulnerability puts millions of smartphone users accounts at risk. http://thehackernews.com/2014/07/facebook-sdkvulnerabilityputs.html. Last visited: 04/27/16.

[58] Dropbox Blog. Security bug resolved in the dropbox sdks for android. https://blogs.dropbox.com/ developers/2015/03/securitybug-resolved-in-thedropbox-sdks-for-android. Last visited: 04/27/16.

[59] S. Shekhar, M. Dietz, and D. S. Wallach. Adsplit: Separating smartphone advertising from applications. In USENIX Security12. USENIX, 2012.

[60] P. Pearce, A. Porter Felt, G. Nunez, and D. Wagner. AdDroid: Privilege separation for applications and advertisers in Android. In ASIACCS12. ACM, 2012.

[61] W. Yang, J. Li, Y. Zhang, Y. Li, J. Shu, and D. Gu. Apklancet: Tumor payload diagnosis and purification for android applications. In ASIACCS14. ACM, 2014.

[62] GuardSquare. Proguard java obfuscator. http://proguard.sourceforge.net.

[63] C. Gibler, J. Crussell, J. Erickson, and H. Chen. Androidleaks: automatically detecting potential privacy leaks in android applications on a large scale. In TRUST 12. Springer, 2012.

[64] S. Arzt, S. Rasthofer, C. Fritz, E. Bodden, A. Bartel, J. Klein, Y. le Traon, D. Octeau, and P. McDaniel. Flowdroid: Precise context, flow, field, object-sensitive and lifecycle-aware taint analysis for Android apps. In PLDI14, 2014.

[65] F. Wei, S. Roy, X. Ou, and Robby. Amandroid: A Precise and General Inter-component Data Flow Analysis Framework for Security Vetting of Android Apps. In CCS14. ACM, 2014.

[66] M. I. Gordon, D. Kim, J. Perkins, L. Gilham, N. Nguyen, and M. Rinard. Information-flow analysis of Android applications in DroidSafe. In NDSS15, 2015.

[67] M. Backes, S. Bugiel, E. Derr, S. Gerling, and C. Hammer. RDroid: Leveraging Android App Analysis with Static Slice Optimization. In ASIACCS 16. ACM, 2016.

[68] P. Wijesekera, A. Baokar, A. Hosseini, S. Egelman, D. Wagner, and K. Beznosov. Android permissions remystified: A field study on contextual integrity. In USENIX Security15. USENIX, 2015.

[69] S. Poeplau, Y. Fratantonio, A. Bianchi, C. Kruegel, and G. Vigna. Execute This! Analyzing Unsafe and Malicious Dynamic Code Loading in Android Applications. In NDSS14, San Diego, CA, 2014.

[70] M. Oltrogge, Y. Acar, S. Dechand, M. Smith, and S. Fahl. To pin or not to pinhelping app developers bullet proof their tls connections. In USENIX Security15. USENIX, 2015

[71] S. Fahl, M. Harbach, T. Muders, L. Baumg?rtner, B. Freisleben, and M. Smith. Why eve and mallory love android: an analysis of android ssl (in)security. In CCS12. ACM, 2012.

[72] M. Egele, D. Brumley, Y. Fratantonio, and C. Kruegel. An empirical study of cryptographic misuse in android applications. In CCS13. ACM, 2013.

[73] AdMob. https://developers.google.com/admob/.

[74] Amazon Webservices. http://aws.amazon.com/.

[75] Google Maps API. https://developers.google.com/maps/.

[76] PayPal. https://developer.paypal.com/.