X-Decaf： Android 平台社交类应用的缓存文件泄露检测

李 晖 王 斌* 张 文 汤 祺 张艳丽
(北京邮电大学计算机学院 北京 100876)

摘 要：由于社交类应用涉及的隐私数据类型非常多，导致这类应用在被广泛使用的同时，频繁出现用户隐私泄露事件，但是目前还鲜有针对社交应用的隐私泄露检测机制的研究。该文结合 Android 系统的特性，提出一个面向 Android 社交类应用检测框架 X-Decaf(Xposed-based-detecting-cache-file)，创新性地利用污点追踪技术以及 Xposed 框架，获取应用内疑似泄露路径，监测隐私数据的缓存文件。此外，该文给出了对隐私泄露进行评级的建议，并利用该框架对 50 款社交类应用进行了检测，发现社交类应用普遍存在泄露用户隐私信息的漏洞。

关键词：隐私泄露；污点追踪；缓存文件；Xposed；Android 系统
中图分类号： TP309 **文献标识码**：A **文章编号**：1009-5896(2017)01-0066-09
DOI: 10.11999/JEIT160555

X-Decaf： Detection of Cache File Leaks in Android Social Apps

LI Hui WANG Bin ZHANG Wen TANG Qi ZHANG Yanli
(*School of Computer Science*, *Beijing University of Posts and Telecommunications*, *Beijing* 100876, *China*)

Abstract: Since social applications involve various types of information related to the user privacy, events of privacy leakage occur frequently along with their popular applications and few studies are available on the privacy leakage detection for social applications. With the combination of the characteristics of the Android system as well as the exploitation of the taint tracking technology and Xposed framework, a privacy leakage detection tool named X-Decaf (Xposed-based-detecting-cache-file) is proposed, which is oriented to social applications on Android platform. It suspects the leakage paths within the applications and detects the privacy data's cache files. This paper also presents a suggestion for the evaluation of the privacy leakage. Evaluation results of 50 kinds of Android social applications show that many vulnerabilities of user privacy leakage exist in the social applications on Android platform.

Key words: Privacy leakage; Taint tracking; Cache file; Xposed; Android system

1 引言

iiMedia Research 在“2015 年中国手机 APP 市场研究报”的调查报告中显示，截止 2015 第 2 季度，中国手机用户规模达到 6.57 亿人，智能手机用户规模为 6.01 亿人，庞大的用户基础推动了中国手机应用软件的快速发展。同时，报告中显示，在 2014～2015 年中国手机网民对各类型移动应用欢迎状况调查中，即时通讯与社交应用所占比为 64.1%。2016 年 2 月份，微信、QQ 活跃用户占比均超过 70%。可见，微信、QQ 等应用成为了当前手机用户使用最多和最为频繁的移动社交应用。

特别是近年来随着大数据平台的建立与发展，当用户隐私数据成为一种资产，社交应用毫无疑问地成为大数据平台机构的首要监测目标。用户在使用应用的过程中产生了大量的隐私数据，然而用户无法确信应用开发者是否对隐私数据做了保护处理。因此，有必要对社交应用的用户数据保护情况进行研究分析。

Android平台上应用的隐私数据泄露问题已经引起国内外的广泛关注。由于Android是一个基于权限控制的操作系统，因此很多安全解决方案集中在针对权限的分析上。文献[1]提出通过动态分析平台VetDroid，重建细粒度的权限分析，进而判别应用程序的敏感行为；文献[2]设计了一个基于上下文的访问控制机制，通过这个机制，基于用户的特定的上下文，可动态授予或撤销应用程序的权限；文献[3]提出一个细粒度的、以用户为中心的权限模型，

收稿日期：2016-05-28；改回日期：2016-10-12；网络出版：2016-12-14
*通信作者：王斌 wangbin_bupt@163.com
基金项目：国家自然科学基金资助(61370195)，中兴通讯产学研项目
Foundation Items: The National Natural Science Foundation of China (61370195), ZTE Corporation and University Joint Research Project

这个模型允许用户对他们安装的软件进行有选择的授予权限。文献[4]基于访问控制提出了组件级别的权限方案让开发者更好地管理应用中组件的安全性。在这些研究中，由于基于权限控制对用户数据泄露检测的方式需要用户的频繁参与，被认为是一种不实用的手段。

还有一部分学者侧重于分析某一类数据的隐私检测与隐私保护。文献[5]提出了 CHEX，一种静态分析方法，可以自动审查 Android 应用程序的组件劫持漏洞，保护用户数据。文献[6]提出的一个基于内容的细粒度运行时访问控制系统，对应用程序的照片访问进行控制。文献[7]通过研究 Android 手机的外部设备对用户隐私数据的泄露，提出了针对蓝牙、NFC 等方式连接的外部设备的管理。在文献[8,9]中，针对地理位置信息的泄露，学者们提出了相应的保护策略。

此外，研究人员通过修改 Android 系统源代码和 Android 应用程序，对隐私数据的泄露进行追踪。文献[10]基于 Linux Container 机制，实现应用程序分离以及敏感组件隔离，并保护隐私数据。文献[11]将解决隐私保护作为一种学习的问题，利用贝叶斯定理等建立定量/概率的二元判断模式，根据扩散点所处的环境来判断它是不是隐私泄露。TaintDriod[12]通过对 Android 系统虚拟机以及解释器的修改，提供了完备的动态污点追踪功能。许多学者也在 TaintDroid 基础上提出了更为优化的方案[13,14]。文献[15,16]通过对 Android 应用程序进行字节码重写，加入相应的隐私检测策略，通过应用重打包的方式实现隐私追踪与漏洞检测。

以上是国内外研究学者针对 Android 平台应用软件隐私泄露以及污点追踪的常用技术。由于社交类应用涉及的系统权限以及隐私数据类型种类很多，因此很难使用现有的基于某类权限或者隐私数据的框架方案检测隐私泄露；同时，如果利用基于 TaintDroid 的思路，采用修改 Android 系统的方法策略，或者修改并重打包应用的方式去分析应用的隐私泄露，不仅实验成本增加，而且对应用的运行效率产生很大影响。可见，现有的隐私泄露分析方法并不适用于社交类应用。

针对社交类应用，本文提出了一个检测框架 X-Decaf(Xposed-based-detecting-cache-file)，创新性地以社交应用运行中产生的缓存文件作为研究对象，利用污点追踪技术以及 Xposed 框架，获取应用内疑似泄露路径，对敏感数据的泄露情况进行检测。Xposed 是 Android 平台成熟的钩子技术(Hook)框架，可运行在 dalvik, art 虚拟机上，并能替换任意 java 方法。Xposed 框架作用于 Android 系统 Zygote 启动阶段，通过替换 app_process 进程，以及对 Android 虚拟机运行时的修改，完成对 Zygote 进程的劫持。Android 系统中应用是由 Zygote 创建，应用所运行的进程中也包含 Xposed 框架，因此，利用 Xposed 提供的接口编写对应的 Hook Module，可实现对应用运行中的行为动作进行监控、修改甚至替换。利用 Xposed 的这个特性，本文所设计的 X-Decaf 可对社交应用的敏感数据进行分类统计，灵活地选择待检测的隐私数据类型，在待测应用运行时动态监测应用本身对缓存文件的处理情况，同时具有与 Android 系统低耦合、无需修改应用程序、低功耗等特点。

本文第 2 节对社交应用的隐私数据进行了定义，并给出泄露评判标准；第 3 节具体介绍 X-Decaf 框架系统；第 4 节给出了利用 X-Decaf 框架对国内主流的 50 款社交应用进行泄露分析的实验数据和结果分析；第 5 节为本文结论，并讨论了本框架的不足以及未来相关工作。

2　隐私数据与泄露

2.1　隐私数据

广泛使用的移动社交类应用涉及到大量的用户数据信息，因此首先需要明确本文所讨论的社交类应用的隐私数据类型。本文通过对市场主流 50 款社交应用进行分析统计(如图 1)，发现图片、视频、语音、地理位置、联系人、电话短信等用户隐私数据在社交应用中均不同程度地被涉及到。由于本文是对社交类应用缓存数据泄露的研究，并且在实验过程中显示，多媒体类型用户数据(图片、视频、语音等)常常产生大量缓存文件。因此，本文将图片、视频、语音等作为首要隐私数据，研究社交应用对这些隐私数据的泄露情况。

2.2　隐私泄露定义

在 Android 系统中，系统在/data/data/目录

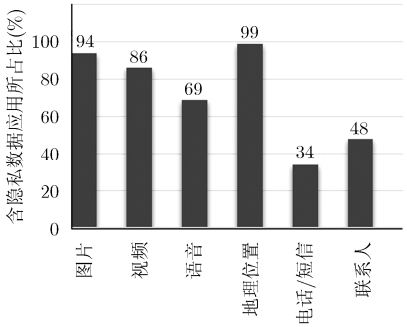

图 1　社交类应用涉及用户隐私数据类型统计图

下，为每个应用创建独享的文件目录。应用程序在运行过程中默认在该目录下创建数据文件。该文件目录的安全性由 Android 系统本身提供的安全保护机制来保证。当某应用进程访问文件时，首先需要通过基于 UID/GID 的 DAC 安全检查，接着进行基于 SEAndroid 的 MAC 安全检查。然而，即便应用的隐私数据保存在该目录下，也面临手机被破解(root)后，恶意软件冲破 Android 保护机制，获取用户隐私数据的风险。

同时，许多原因造成应用开发者并未将应用的所有运行数据存放在对应的/data/data/目录下。这些原因主要如下：(1)应用开发者在开发应用时，未考虑应用的安全性，随意调用 Android 系统 API(例如 getExternalStorageDirectory( )等)，并将应用运行时产生的缓存文件、数据信息等放在一些公共目录下；(2)Android 手机配置参差不齐，内存小的手机需 SD 卡扩展，因此，一些应用运行的缓存文件被存放在 SD 卡公共目录下，同时应用并未对这些公共目录下的文件数据进行管理，手机内的任何应用均可以访问 SD 卡目录下的文件，这时便存在隐私泄露问题。

传统的隐私泄露行为指应用未显式告知用户需远程收集相关信息，却利用网络等途径收集并传播用户隐私。本文以一种新的角度定义用户的隐私泄露。本文讨论的隐私泄露行为是指应用由于自身设计的缺陷以及运行期间的表现，产生了涉及用户隐私的缓存文件，并且缺乏良好的策略管理这些缓存文件。为了分析社交软件是否面临着隐私泄露的风险，我们利用 X-Decaf 框架对社交应用运行时产生的缓存文件进行检测分析，分析缓存文件是否产生含有隐私数据，以及应用本身是否对这些敏感缓存文件进行生命周期管理。

2.3 隐私泄露评判标准

缓存文件隐私泄露的评判标准的定义，应遵照以下依据：

(1)缓存文件存储路径：根据 2.2 节的讨论，Android 系统中，应用在运行时将在以下文件路径中产生缓存文件，如表 1 所示。

表 1 缓存文件产生路径

缓存文件产生路径	路径简记
/data/data/包名/	DATA_PRI
/data/data/包名/(全局可读)	DATA_PUB
/storage/emulated/0/Android/data/包名/	SD_PRI
/storage/emulated/0/	SD_PUB

在 Android 系统中，文件的存储路径一定程度上反映了文件的访问权限。根据文件的存储路径分析 Android 平台上常见的攻击场景如下：

DATA_PRI：只限应用自身可以访问，攻击者在不破解的情况下很难获取该目录下文件信息；

DATA_PUB：若目录下的文件为全局可读，手机中任意应用均可访问。攻击者可以通过恶意应用访问该目录文件；

SD_PUB, SD_PRI：由于是在 SD 卡公共目录下，攻击者可以随意篡改。

此处区分 SD_PUB 与 SD_PRI，是由于 SD_PUB 目录反映出来的是应用开发者的不规范行为，并且利用 Android 系统设置中的清理数据功能或者删除应用可以删除 SD_PRI 目录下的所有数据；然而除非应用自身去管理，否则 SD_PUB 目录下的数据是不被删除的。

(2)缓存文件保护状态：通过上述攻击场景分析可看出，仅仅依靠 Android 系统的文件目录安全机制无法保证应用产生的数据文件的安全。常见的文件保护手段有混淆、加密等，即便攻击者冲破 Android 系统的保护机制，获取到了应用产生的文件，也需花费很大代价将保护后的文件还原。经保护处理的缓存文件提高了安全性，同时也避免了隐私泄露。因此，文件保护与否也是对文件的隐私泄露分析的重要依据。

(3)缓存文件生命周期：缓存文件的生命周期是指缓存文件从产生、转移、存储以及删除等过程。在整个生命周期中，缓存文件都有可能遭受攻击者的恶意攻击。因此，对于应用在缓存文件的生命周期中对缓存文件的管理，情景模拟主要有以下 3 种：

情景 1：应用在运行时产生缓存文件，并且缓存文件在应用退出后销毁；

情景 2：缓存文件在应用退出后依据存在，但应用内部提供清理缓存等相关功能，可将缓存文件删除；

情景 3：应用内部提供清理缓存等相关功能，依然无法将缓存文件删除。

针对以上 3 个评判依据，缓存文件隐私泄露的评判标准定义了如下：对于保护处理的缓存文件，我们默认应用的保护手段是安全的，因此不存在泄露情况。对于未保护的缓存文件，根据其存储路径以及生命周期，隐私泄露评判标准如表 2 所示。

根据上述隐私泄露评判标准的定义，并结合目前常见的 Android 平台攻击场景，将隐私泄露等级分为无泄露、轻度泄露、中度泄露、重度泄露以及超重度泄露，具体为：

表 2 未保护隐私泄露评判标准

	DATA_PRI	DATA_PUB	SD_PRI	SD_PUB
情景 1	轻度泄露	中度泄露	中度泄露	中度泄露
情景 2	轻度泄露	重度泄露	重度泄露	重度泄露
情景 3	轻度泄露	超重度泄露	超重度泄露	超重度泄露

无泄露：应用在运行中并未产生缓存文件或者缓存文件是经过保护处理；

轻度泄露：需要攻击者破解手机后，才能进行相关隐私窃取；

中度泄露：无需破解手机，需攻击者监听应用运行时的动作；

重度泄露：攻击者仅需利用 Android 提供的接口或直接利用文件夹工具查看相关路径，便可进行攻击；

超重度泄露：应用自身提供的清理缓存功能并未对缓存文件进行清除，本文视为“超重度泄露”。

X-Decaf 框架依据以上隐私泄露定义以及标准，将应用、隐私数据、缓存文件联系起来，对应用的泄露情况进行高效的分析。

3　检测框架

3.1 总体框架

本文结合动态检测与静态分析技术，设计并实现了 X-Decaf(图 2)框架，该框架具有与 Android 系统低耦合，无需修改应用程序，检测精度高以及对应用运行影响低的特点。

X-Decaf 主要由 3 部分组成：敏感函数库、污点追踪(由动态追踪与污点标记组成)以及缓存文件分析(分为策略判定与人工检验)。

(1)敏感数据库：通过对市场上社交应用大量分析，统计出应用调用系统 API 的情况，筛选出与敏感数据产生、传播相关的系统 API，最终形成 X-Decaf 框架的敏感函数库，针对性地分析隐私数据的泄露情况。

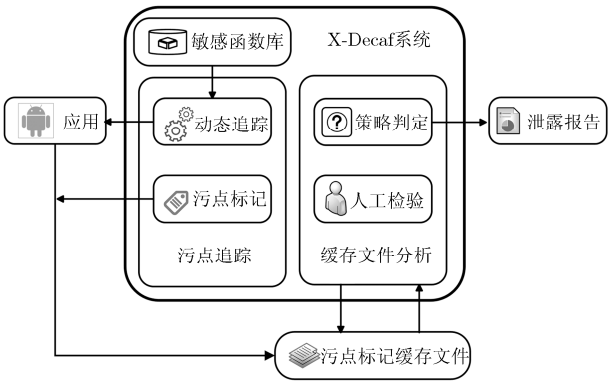

图 2　X-Decaf 总体框架

(2)污点追踪：污点追踪主要由两部分组成：动态追踪模块是基于 Xposed 框架，通过向敏感数据库请求相关隐私数据的系统 API 函数，对应用的敏感函数进行钩子技术(Hook)操作，实现在隐私数据产生、传播以及最终的存储过程中的监控；污点标记模块针对不同的隐私数据类型进行污点标记。通过源隐私数据类型、源文件名，制定相应策略，实现基于文件的污点标记。

(3)缓存文件分析：首先，人工验证阶段是依据第 2 节中对隐私泄露的定义以及标准来执行对应的检测；其次，策略判定为自动化的监测脚本，将监测所有污点追踪阶段产生的污点标记缓存文件。策略判定模块依据污点缓存文件的 TAG 值以及隐私泄露评判标准(见表 2)，监测污点缓存文件变化情况，并最终形成泄露报告。

3.2 具体技术

图 3 显示了 X-Decaf 整个的运行流程，其各模块介绍如下：

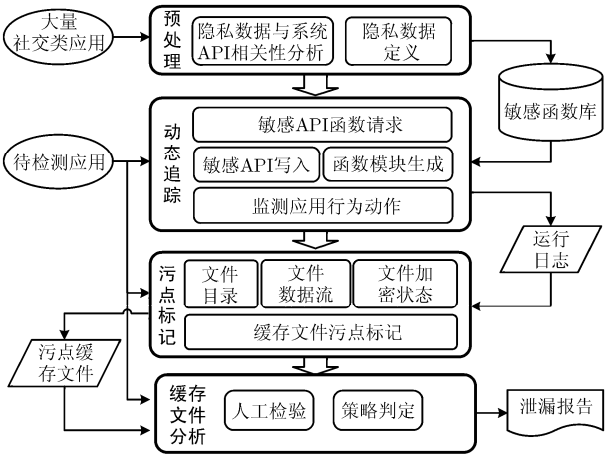

图 3　X-Decaf 具体工作流程

(1)敏感函数库建立：在 X-Decaf 框架的设计初期，我们对市场上近百款社交应用进行统计采集，分析社交类软件的隐私数据与对应的系统 API 的相关性。应用程序产生、获取或者传播用户隐私数据需申请相关权限并调用相关系统 API，且这些系统 API 是固定的(例如照片涉及调用相机功能、读取系统图库、图片压缩、文件 I/O 等系统 API)。通过常用的软件分析工具(如 apktool, IDA pro 等)，对应用软件逆向反汇编，分析并统计出与敏感数据产生、传播相关系统 API，最终形成 X-Decaf 框架的敏感函数库。

本文主要针对社交软件的缓存文件进行分析，因此，敏感函数库将涉及语音、照片、视频这 3 类包含大量用户信息的敏感数据的系统接口。为方便自动化脚本申请检测的敏感函数，敏感函数库以 XML 格式对外开放敏感 API 策略。表 3 列出了敏感函数库中语音数据对应的部分系统敏感 API 策略。

sensitive-policy 标签。该标签为指定隐私数据的根标签。其后紧跟的属性显示了该隐私策略标签对应的隐私类型**(privacy-type)**。

uses-permission 标签。该标签表示该类型隐私数据涉及到系统敏感权限，它与 Android 应用中声明的权限一致。

class-info 标签。该标签表示该类型敏感数据涉及到的 Android 系统 API 的类信息，其后紧跟的属性(**class-name**)表示具体的类名。

method-info 标签。该标签列出了该类型敏感数据具体的方法信息，其后紧跟的属性分别表示方法名(**method-name**)以及方法参数(**method-args**)。

(2)动态追踪：敏感函数库中的敏感函数是应用运行期间的主要监测对象。动态追踪模块首先向敏感函数库请求这些敏感函数，然后利用 Xposed 框架以这些敏感函数作为检测对象，生成对应的函数劫持模块(Hook Module)，之后在待测应用运行前将对应的模块(Module)加载到手机系统中，最后，在应用运行时进行检测，产生运行日志。

(3)污点标记：X-Decaf 框架的污点标记模块主要是对缓存文件的筛选与污点标记，主要步骤如下：

(a)缓存文件筛选： 在动态追踪阶段，X-Decaf 监控系统 I/O 操作。由于系统在运行过程中存在许多系统级文件 I/O 操作，如果不进行过滤操作，污点标记将标记出许多无关项，同时影响整个系统的运行速度。为了提高整个污点标记的精度以及尽可能小地影响待测应用的性能，X-Decaf 框架对文件进行了细粒度的过滤。

首先，X-Decaf 一旦发现缓存文件后缀名为.jpg, .jpeg 和.bmp 等格式，直接判断为敏感缓存文件；对于非常规文件后缀名，X-Decaf 框架才对其做进一步数据流检测(例如常见的 jpg 格式文件的固定文件头由“JFIF”字符串组成)；此外，如果应用本身对缓存文件做了保护处理(混淆、加密等手段)，X-Decaf 也可以检测出来，方便污点标记阶段进行处理。这样，无论缓存文件以何种形式存在，通过 X-Decaf 框架对文件数据流进行过滤，依旧可以追踪到对应的泄露缓存文件。

(b)污点标记： 通过前一阶段的缓存文件筛选，X-Decaf 已经筛选出敏感缓存文件(包含已保护处理的缓存文件)。X-Decaf 采用污点标记机制对敏感缓存文件进行标记，即在有缓存文件名后追加对应的 TAG。TAG 由缓存文件对应的隐私数据类型、缓存文件的哈希值以及文件的保护状态(0 表示未保护，1 表示经过保护处理)3 部分组成。例如，照片类缓存文件的原文件名为“cache.tmp”，该文件的哈希值为 hash，并且文件未经应用保护处理，则经过 X-Decaf 标记后缓存文件名变为“cache_photo_hash_0.tmp”。这样处理的好处有：①不改变待检测应用对缓存文件的调用逻辑； ②污点 TAG 可以传播，并且可以将 source 文件与之后的所有缓存文件关联起来；③可监测出数据流在各缓存文件传播过程中文件保护状态是否改变。

(4)人工验证：通过 X-Decaf 对应用的污点追踪，产生了一系列污点标记的缓存文件，需对这些污点标记的缓存文件进行分析。首先，通过人工测试(如 2.3 节中描述的情景模拟)，验证社交应用软件本身

表 3 敏感函数库语音数据敏感 API 函数(部分)

```
<sensitive-policy privacy-type = "voice">
    <user-permission name = "android.permission.RECORD_AUDIO"/>
    <user-permission name= "android.permission.WRITE_EXTERNAL_STORAGE"/>
    <user-permission name = "android.permission.READ_EXTERNAL_STORAGE"/>
    <class-info class-name = "android.media.AudioRecord">
        <method-info method-name = "startRecording" method-args = " "/>
        <method-info method-name = "read" method-args = "byte[ ], int,int"/>
        <method-info method-name = "native_read_in_byte_array" method-args = "byte[],int,int,boolean"/>
        ⋮
    </class-info>
    ⋮
</sensitive-policy>
```

是否对缓存文件进行生命周期管理；其次，依据 2.3 节隐私泄露的评判标准来执行检测，根据文件的存储路径以及文件保护状态，观察缓存文件在测试过程中，是否存在或已被删除，在“策略判定”阶段执行相应策略。

(5)策略判定：策略判定与人工验证协调配合完成。策略判定为自动化的监测脚本，将监测所有污点追踪阶段产生的污点标记缓存文件，即依据污点缓存文件的 TAG 值，监测污点缓存文件的保护状态变化、文件路径、文件生命周期，同时根据隐私泄露评判标准(表 2 所示)，最终形成泄露报告。

3.3 X-Decaf 框架分析

与同类型泄露检测框架相比，X-Decaf 具有如下优点：

(1)系统耦合度降低且无需修改应用程序：目前多数动态污点追踪或者隐私泄露检测框架均需要修改 Android 系统，例如建立在 TaintDroid 基础上的研究就需要修改 Android 系统。而现有的另外一些检查方法则将修改目标转向待检测应用本身，对应用进行字节码重写与策略插入，然而，随着应用软件的防篡改、签名机制等保护措施的不断完善使重打包应用的代价逐步增大。X-Decaf 既不需要对 Android 系统平台进行修改，也不需要对应用程序进行字节码重写，巧妙地从系统 API 入手，分析并利用隐私数据与对应的系统 API 的相关性，针对隐私数据泄露进行检测。

(2)检测多类型隐私泄露：敏感函数库对外提供了常见应用隐私数据与系统 API 的关系模型。因此 X-Decaf 针对不同隐私数据，可以灵活地选取多种敏感函数策略进行监测。

(3)多应用横向检测：通过研究发现，应用在操作敏感数据时所调用的系统 API 具有规律性。因此，X-Decaf 系统可以方便地同时监测市场上多种应用对某类敏感数据的泄露情况。

4　实验结果分析

为分析国内市场上社交应用的隐私泄露情况，利用 X-Decaf 框架对国内主流的 50 款社交应用进行了多类型隐私数据基于缓存文件的泄露分析。测试手机为 Nexus5, Android 系统版本为 5.1.1，内存 16 G，不支持外置 SD 卡，且已经被破解。

4.1 隐私数据泄露纵向分析

首先使用 X-Decaf 对国内最流行的手机社交应用-微信进行了纵向隐私数据分析，主要对微信内隐私数据(语音、图片、视频等)进行缓存文件泄露分析，同时对缓存文件泄露的路径个数进行了统计。由表 4 所示，X-Decaf 框架可以精确地监测到某一数据源在应用运行过程中，产生、转移、传播等动作对应的缓存文件。例如对于同一源照片数据，微信将为其产生 3 个缓存文件(*.jpg 为原图片拷贝，th_*为小的缩略图，th_*hd 为大的缩略图)。X-Decaf 可对应用运行中数据的精准分析，不会漏报、误报任一条隐私泄露路径。

4.2 隐私数据泄露横向分析

对于国内市场的主流社交应用，我们利用 X-Decaf 进行了横向分析，分析了应用对语音、图片、视频数据的泄露情况，并列出对应的泄露等级以及泄露路径的个数，结果如表 5 所示。

4.3 社交应用隐私泄露评分

应用对不同类型隐私数据产生不同等级、不同数量的泄露，这也是社交应用隐私泄露评分需考虑的。对于应用隐私数据泄露，我们定义了如表 6 的泄露评分规则以及式(1)所示的分数 C 计算公式：

$$C = \sum_{k=0}^{4} V_k \cdot n \tag{1}$$

其中，C 表示应用对某类型隐私数据的泄露分数，V_k 表示泄露等级对应的泄露评分，n 表示该类型隐私数据泄露的路径个数。

表 4 微信对隐私(图片、语音、视频)泄露情况 [1)]

微信(6.3.13)	泄露路径	泄露路径个数
语音	SD_PUB/tencent/MicroMsg//voice2/../msg_*.amr	1
图片	SD_PUB/tencent/MicroMsg/../image2/../*.jpg SD_PUB/tencent/MicroMsg/../image2/../th_* SD_PUB/tencent/MicroMsg/../image2/../th_*hd	3
视频	SD_PUB/tencent/MicroMsg/../video/*.mp4 SD_PUB/tencent/MicroMsg/../video/*.jpg SD_PUB/tencent/MicroMsg/../draft/.. SD_PUB/tencent/MicroMsg/../draft/*.thumb	4

[1)] “..”表示略去应用由于业务逻辑(如时间、随机数等)产生的文件夹目录；缓存文件名用“*”代替。

表 5 主流社交应用关于语音、照片、视频类隐私数据的泄露情况

应用程序	应用版本	语音		图片		视频	
		泄露等级	泄露路径个	泄露等级	泄露路径个	泄露等级	泄露路径个
微信	6.3.13	超重度泄露	1	超重度泄露	3	超重度泄露	4
手机 QQ	6.2.3.2700	超重度泄露	1	超重度泄露	3	超重度泄露	2
微博	6.3.0	无泄露	0	重度泄露	2	重度泄露	2
易信	4.3.1	超重度泄露	1	超重度泄露	2	超重度泄露	1
陌陌	6.7_0413	超重度泄露	1	超重度泄露	2	超重度泄露	3
无秘	5.3.0	重度泄露	1	重度泄露	1	无泄露	0

依据表 5 的结果，我们得出主流社交应用针对语音、照片、视频隐私数据的泄露分数(图 4 所示)。泄露分数反映了应用对隐私数据的缓存文件的处理情况。微信由于其相关复杂业务以及提供的丰富社交功能，使得其对于这 3 类隐私数据的泄露分数都居于首位也是在意料之中。

表 6 泄露评分规则

泄露等级	泄露评分 V
超重度泄露	4
重度泄露	3
中度泄露	2
轻度泄露	1
无泄露	0

最后，X-Decaf 继续扩大到 50 款社交应用，并针对社交应用最流行的隐私数据-照片进行分析。如图 5，在随机的抽检样本中，除了 4%的社交应用由于不涉及照片隐私数据而不存在泄露外，其他 96%应用均存在重度泄露以上的情况，并且这其中有 74%的应用是超重度泄露。这充分反映出了目前应用开发者在开发应用时，代码编写不规范并且未考虑用户的隐私泄露风险。

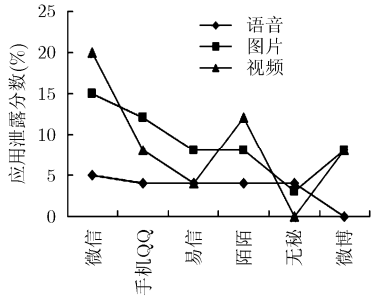

图 4 主流社交应用关于语音、照片、视频的泄露分数

4.4 对社交应用运行性能影响

为测试 X-Decaf 在检测应用隐私泄露过程中对应用运行性能的影响，我们选择 Android SDK 提供的 DDMS(Dalvik Debug Monitor Service) 中的“method profiling”工具。利用 DDMS，可以方便地查看手机终端应用进程的执行情况。其中“method profiling”工具可在无源码的情况下，对应用运行时进行动态分析。当对应用进程某段运行时开启调试后，“method profiling”会反馈调试过程中涉及到的所有 Java 方法，包括方法的执行时间、调用次数、在总体耗时中的占比等信息。这些运行信息可用于分析社交应用的运行性能。

我们依旧以微信作为研究对象，通过以下 5 个典型的测试场景，分析 X-Decaf 框架对微信运行性能耗时(单位为 ms)的影响：

测试 1：在微信非聊天界面，进行多处相同点击操作，判断 X-Decaf 对微信整体运行的性能影响；

测试 2：调用摄像头拍照并发送，判断 X-Decaf 对微信拍照发送流程影响；

测试 3：连续发送 9 张图片，判断 X-Decaf 对图片发送流程的性能影响；

测试 4：连续发送 3 个时长为 6 s 小视频[2)]，判断 X- Decaf 对语音发送流程的性能影响；

测试 5：连续发送 9 张图片以及 3 个 6 s 小视频，判断 X-Decaf 对图片、视频的发送流程的性能影响。

测试中，每项测试均在 20 次以上，且使用相同的 wifi 网络。为防止由于人工测试、网络条件以及缓存对下次实验结果的影响，在每次测试结束时，将应用运行中产生的缓存清除。

由图 6 的测试结果可以看出， X-Decaf 在未修改应用程序的情况下，对应用运行性能造成的影响很低。而 X-Decaf 作为一款缓存文件泄露检测的框

[2)]微信单次最多发送 9 张图片，并且小视频的最长录制时间为 6 s。

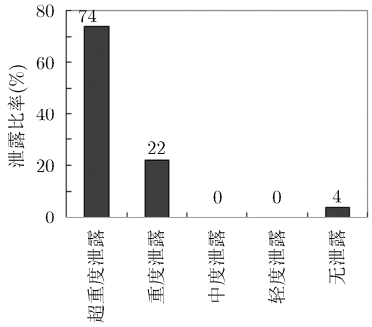

图 5 社交应用针对照片隐私数据的泄露情况统计表

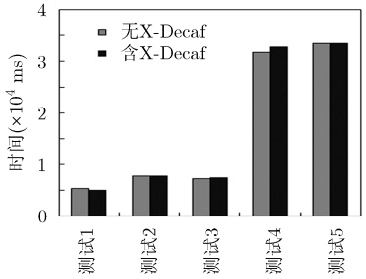

图 6　X-Decaf 对微信运行性能影响分析

架，作用于 Android 框架层，并未对待测应用的运行逻辑产生影响。

5　结束语

社交应用已然成为最受用户欢迎的移动类应用。通常情况下，用户认为该类应用在从设计开始就应该考虑如何保护用户隐私的问题。然而实验结果表明，多数的社交类应用在对多媒体类型数据使用过程中存在严重的缓存泄露情况。

尽管 X-Decaf 实现了预期目标，但同时也存在一些不足，我们将从以下方面改进 X-Decaf：(1)敏感函数库的扩建。扩大用户敏感数据的定义范围，分析这些敏感数据传播中调用的系统 API，扩充敏感函数库。(2)支持更多类型应用。将调整 X-Decaf 的相关流程，使其支持更多类型的应用缓存数据的检测，并能够支持对更多数据类型的缓存泄露追踪。(3)缓存文件的透明保护。透明保护应用产生的缓存文件，即对缓存文件进行自动加密，当应用读取缓存数据时，对加密的缓存文件进行自动透明解密，使得在不影响应用本身的业务逻辑的同时，有效防止缓存文件泄露引起的隐私数据泄露。为达到不影响原有应用的运行，这将是一个很大的挑战。

参 考 文 献

[1] ZHANG Y, YANG M, YANG Z, *et al*. Permission use analysis for vetting undesirable behaviors in android apps[J]. *IEEE Transactions on Information Forensics and Security*, 2014, 9(11): 1828–1842. doi: 10.1109/TIFS.2014.2347206.

[2] SHEBARO B, OLUWATIMI O, and BERTINO E. Context-based access control systems for mobile devices[J]. *IEEE Transactions on Dependable and Secure Computing*, 2015, 12(2): 150–163. doi: 10.1109/TDSC.2014.2320731.

[3] NAUMAN M, KHAN S, OTHMAN A T, *et al*. Realization of a user-centric, privacy preserving permission framework for Android[J]. *Security and Communication Networks*, 2015, 8(3): 368–382. doi: 10.1002/sec.986.

[4] WU L, DU X, and ZHANG H. An effective access control scheme for preventing permission leak in Android[C]. 2015 International Conference on Computing, Networking and Communications (ICNC), IEEE, Anaheim, CA, USA, 2015: 57–61. doi: 10.1109/ ICCNC.2015.7069315.

[5] LU L, LI Z, WU Z, *et al*. Chex: Statically vetting android apps for component hijacking vulnerabilities[C]. Proceedings of the 2012 ACM Conference on Computer and Communications Security, North Carolina, USA, 2012: 229–240.

[6] TAN J, DROLIA U, MARTINS R, *et al*. Short paper: Chips: Content-based heuristics for improving photo privacy for smartphones[C]. Proceedings of the 2014 ACM Conference on Security and Privacy in Wireless & Mobile Networks. Oxford, UK, 2014: 213–218. doi: 10.1145/2627393.2627394.

[7] NAVEED M, ZHOU X, DEMETRIOU S, *et al*. Inside job: Understanding and mitigating the threat of external device mis-binding on Android[C]. Network and Distributed System Security Symposium, San Diego, California, USA, 2014. doi: 10.14722/ndss.2014.23097.

[8] RAHMAN M, BALLESTEROS J, CARBUNAR B, *et al*. Toward preserving privacy and functionality in geosocial networks[C]. Proceedings of the 19th ACM Annual International Conference on Mobile Computing & Networking, Miami, Florida, USA, 2013: 207–210.

[9] FAWAZ K, FENG H, and SHIN K G. Anatomization and protection of mobile apps' location privacy threats[C]. 24th USENIX Security Symposium (USENIX Security 15). Washington, D.C., USA, 2015: 753–768.

[10] YAN L, GUO Y, and CHEN X. SplitDroid: isolated execution of sensitive components for mobile applications[C]. International Conference on Security and Privacy in Communication Systems. Springer International Publishing, Dallas, TX, USA, 2015: 78–96.

[11] TRIPP O and RUBIN J. A Bayesian approach to privacy enforcement in smartphones[C]. 23rd USENIX Security Symposium (USENIX Security 14). California, USA, 2014: 175–190.

[12] ENCK W, GILBERT P, HAN S, *et al*. TaintDroid: An

information-flow tracking system for realtime privacy monitoring on smartphones[J]. *ACM Transactions on Computer Systems* (*TOCS*), 2014, 32(2): 5. doi: 10.1145/2619091.

[13] HSIAO S W, HUNG S H, CHIEN R, *et al*. PasDroid: real-time security enhancement for Android[C]. 2014 Eighth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), Birmingham, UK, 2014: 229–235.

[14] BAL G, KAI R, and HONG J I. Styx: Privacy risk communication for the Android smartphone platform based on apps' data-access behavior patterns[J]. *Computers & Security*, 2015, 53: 187–202.

[15] CUI X, YU D, CHAN P, *et al*. Cochecker: Detecting capability and sensitive data leaks from component chains in android[C]. Information Security and Privacy. Springer International Publishing, Wollongong, NSW, Australia, 2014: 446–453.

[16] ZHANG M and YIN H. Efficient, context-aware privacy leakage confinement for android applications without firmware modding[C]. Proceedings of the 9th ACM Symposium on Information, Computer and Communications Security. Kyoto, Japan, 2014: 259–270.

李 晖： 女，1970 年生，副教授，硕士生导师，研究方向为密码学及其应用、移动通信安全、智能终端安全.

王 斌： 男，1992 年生，硕士生，研究方向为智能终端安全、移动操作系统安全和软件安全.

张 文： 男，1982 年生，博士生，研究方向为智能终端安全、移动操作系统安全和软件安全.

汤 祺： 女，1991 年生，硕士生，研究方向为智能终端安全、移动操作系统安全和软件安全.

张艳丽： 女，1990 年生，硕士生，研究方向为智能终端安全、移动操作系统安全和软件安全.