

编号_____

西安科技大学
毕业设计（论文）
（ 2022 届）

题 目	基于代码植入技术的 App 隐私明示性动态检测方法
学生姓名	王丹婷
学 号	18408010111
专业班级	软件工程 1801
指导教师	刘晓建
所在学院	计算机科学与技术学院
日 期	2022 年 6 月 2 日

西安科技大学

学位论文诚信声明书

本人郑重声明：所呈交的学位论文（设计）是我个人在导师指导下进行的研究（设计）工作及取得的研究（设计）成果。除了文中加以标注和致谢的地方外，论文（设计）中不包含其他人或集体已经公开发表或撰写过的研究（设计）成果，也不包含本人或其他人在其它单位已申请学位或为其他用途使用过的成果。与我一同工作的同志对本研究（设计）所做的任何贡献均已在论文中做了明确的说明并表示了致谢。

申请学位论文（设计）与资料若有不实之处，本人愿承担一切相关责任。

学位论文（设计）作者签名：王丹婷 日期：2022年6月2日

学位论文知识产权声明书

本人完全了解学校有关保护知识产权的规定，即：在校期间所做论文（设计）工作的知识产权属西安科技大学所有。学校有权保留并向国家有关部门或机构送交论文的复印件和电子版。本人允许论文（设计）被查阅和借阅；学校可以公布本学位论文（设计）的全部或部分内容并将有关内容编入有关数据库进行检索，可以采用影印、缩印或其它复制手段保存和汇编本学位论文。

保密论文待解密后适用本声明。

学位论文（设计）作者签名：王丹婷 指导教师签名：刘晓建
2022 年 6 月 2 日

分 类 号

学校代码 10704

密 级

学 号 18408010111

西安科技大学
学 士 学 位 论 文

题目：基于代码植入技术的 App 隐私明示性动态检测方法

作者：王丹婷

指导教师：刘晓建

学科专业：软件工程

专业技术职称：副教授

申请学位日期：2022 年 6 月

摘 要

近几年对 App 中隐私政策的相关问题的研究层出不穷且快速发展，引起了社会各界的高度关注。而隐私政策是我们在使用 App 时，App 运营者所展示的涉及用户个人信息保护、收集以及使用的条款。如今，隐私政策的内容、隐私政策的实现方式等是否合法引起了用户和国家的高度重视。2020 年 12 月，国家印发了《App 违法违规收集使用个人信息行为认定方法》，其中对 App 隐私政策的呈现方式提出了进一步的要求。

本文的主要工作如下：

（1）主要调研隐私政策在 App 中的实现方式，即调研程序如何将隐私政策的长文本集成到 App 源代码中，并如何触发隐私政策的弹出以及明示内容；

（2）采用代码植入技术，跟踪隐私政策的弹出过程，判断隐私政策的触发是否符合规定（在进入主界面后，不超过四次就能访问到隐私政策）；

（3）采用 Xposed 框架，驱动 App 运行，通过程序运行中抛出的日志信息，分析植入的代码是否正确以及隐私政策的弹出过程是否满足相关规定要求。

本文还对目前动态检测的主要方法和代表性工作进行了归纳与探讨，在此基础上进行了基于代码植入技术的研究，还总结了基于代码植入技术的 App 隐私明示性动态检测方法目前存在的问题与未来的研究方向。

关键词:代码植入技术；Xposed；反编译；反打包；App 隐私政策

ABSTRACT

In recent years, research on the issues related to privacy policies in the App has emerged in an endless and rapid manner, which has aroused great concern from all walks of life. The Privacy Policy is the terms that the App Operator displays when we use the App, which relate to the protection, collection and use of the user's personal information. Nowadays, whether the content of the privacy policy and the implementation of the privacy policy are legal have attracted great attention from users and the state. In December 2020, the state issued the "Method for Determining the Illegal Collection and Use of Personal Information by Apps", which puts forward further requirements for the presentation of the App's privacy policy.

The main tasks of this article are as follows:

(1) Mainly investigate the implementation of the privacy policy in the App, that is, how the research program integrates the long text of the privacy policy into the app source code, and how to trigger the pop-up of the privacy policy and the explicit content;

(2) Using code implantation technology, tracking the pop-up process of the privacy policy, and judging whether the trigger of the privacy policy complies with the regulations (after entering the main interface, the privacy policy can be accessed no more than four times);

(3) Adopt the Xposed framework to drive the Operation of the App, analyze whether the implanted code is correct and whether the pop-up process of the privacy policy meets the relevant requirements through the log information thrown during the operation of the program.

This paper also summarizes and discusses the main methods and representative work of dynamic detection, on the basis of which the research based on code implantation technology is carried out, and the current problems and future research directions of App privacy explicit dynamic detection method based on code implantation technology are also summarized.

Key Words: Code Embedding Technology; Xposed; Decompilation; Anti-Packaging; App Privacy Policy

目 录

1 绪论	1
1.1 本课题研究的背景和意义	1
1.2 国内外研究现状及趋势	1
1.2.1 国外研究现状	2
1.2.2 国内研究现状	2
1.3 相关技术的现状分析	3
1.3.1 App 首次弹窗实现	3
1.3.2 App 隐私政策明示性说明	5
1.4 论文结构	6
2 系统开发使用的工具和方法	7
2.1 Android Studio	7
2.2 jadx-gui	7
2.2.1 工具介绍	7
2.2.2 系统中的使用	8
2.3 Xposed 框架	8
2.3.1 Xposed 框架原理	9
2.3.2 Xposed 组成部分	9
2.3.3 Xposed 框架使用步骤	10
3 系统设计	12
3.1 功能分析	12
3.2 研究思路	12
3.3 工作流程	13
3.4 实现方案	13
4 系统实现	15
4.1 隐私政策在 App 中的实现方式	15
4.2 隐私政策在 App 中的集成	15

4.3 植入代码	16
4.4 运行结果	17
5 系统测试	19
5.1 测试集	19
5.2 测试环境	19
5.3 系统功能性测试	20
5.3.1 隐私政策获取的测试	20
5.3.2 日志模块的测试	21
5.3.3 隐私政策明示性分析测试	21
5.4 系统非功能性测试	22
5.4.1 兼容性测试	22
5.4.2 性能测试	22
5.5 本章小结	23
6 总结和展望	24
6.1 本文工作总结	24
6.2 展望	24
致 谢	25
参考文献	26

1 绪 论

1.1 本课题研究的背景和意义

在全球移动网络迅猛发展的今天，作为连接移动因特网的智能手机，已经在全世界得到了广泛的使用，并因此大大影响了人们的生活习惯和生活方式。Android 是目前手机操作系统中最受欢迎的一种手机操作系统，它具有很强的开放性和较高的性价比。但与此同时，在使用应用程序和登录平台时，个人信息暴露的危险也越来越大。所以，追踪应用程序中的隐私保护策略的实施及合法性，就成了迫切需要解决的问题。

由于 Android 系统的开源特性、手机端软件的广泛应用以及应用分发市场的无序竞争，国家相关部门对 App 上架制定了严格的安全性要求，其中一个要求是 App 必须弹出隐私政策说明文本。如何检测 App 隐私政策说明符合规定要求是相关部门面临的重要需求。例如工信部等部门陆续发布的《App 违法违规收集使用个人信息行为认定方法》《App 违法违规收集使用个人信息自评估指南》等进一步对隐私政策提出了要求。这种要求甚至具体到“进入 App 主界面以后，多于 4 次点击才能访问到隐私政策”即为违法。因此，有必要调研隐私政策在 App 中的实现方式，即调研并跟踪程序如何将隐私政策集成到代码中，并如何触发隐私政策的弹出和明示；采用 Xposed 框架，驱动 App 运行，通过程序运行中抛出的日志信息，分析弹出过程是否满足相关规定要求，发现隐藏在 App 运行中违法违规收集使用个人隐私信息的行为，为政府监管部门的认定和执法提供技术支撑，为 App 运营者、开发者自检、自评估提供辅助检测工具，为用户安全使用 App 提供技术保障。

1.2 国内外研究现状及趋势

静态检测^[1]无需执行代码，只需对源程序或者二进制代码进行分析，即可得到其静态特性，并藉此来判定程序是否有恶意。因为恶意程式码必须即时更新，因此会产生延迟，而且无法辨识可能或尚未侦测到的恶意代码。

动态分析技术^[2]是在沙箱环境、实际设备或者虚拟机上运行的一种应用程序，它可以从不同的角度对程序的访问模式、权限的访问等方面进行检测，从而获得程序的动态特性，从而判定程序的恶意。因为该方法可以捕获程序运行时的行为，所以可以增加检测的精确度，但代价也会更大。

目前，最常见的动态探测技术是沙盒技术，它在沙盒中监测应用程序的行为，而沙

盒只是用于存储程序的系统活动。在沙箱环境下，可以在任何时候中止或中止程序，以避免恶意程序对系统环境的损害。

1.2.1 国外研究现状

Xposed 框架是一种在国外非常受欢迎的安卓平台 HOOK 架构，它的创建者根据自己对安卓系统的了解，分析了安卓系统功能的运作过程，并通过改变 HOOK 功能的入口指针来实现对安卓功能的 HOOK。Xposed 框架修改了安卓的原生 Zygote 过程，它可以在安卓系统启动时，通过截取 Zygote 进程，使 Xposed 框架可以在安卓系统的主要系统功能上进行 HOOK，确保在应用程序开始前，该框架可以装载相应的 Xposed 模块。但它要求对安卓系统 Zygote 进程进行更改，因此 Xposed 框架在使用时必须 root，从而影响到手机的安全。

Li, S.等作者^[3]在 Android 系统上开发了一个实时监控系统（App），用以监测可能的个人资料滥用。这个应用程序可以监测所有安装程式的使用许可要求，并分析可能存在的侵犯隐私的问题。

Lo.Nai-WeiYeh 等^[4]提出了一个基于 Android 平台的用户隐私分析框架 LRPdroid。LRPdroid 的目标是对安装在基于 Android 的移动设备上的应用程序实现信息泄漏检测、用户隐私泄露评估和隐私风险评估。通过一个正式定义的用户隐私模型，LRPdroid 可以有效地支持移动用户管理自己在目标应用上的隐私风险。此外，在 LRPdroid 中引入了新的隐私分析观点，如用户感知和泄漏感知。通过原型系统对两种常见的应用程序使用场景进行了评估，验证了 LRPdroid 框架在用户隐私管理方面的可行性和实用性。

Naoya,Kajiwara 等^[5]主要研究在调用 API 获取手机 ID 时，通过远程过程调用来获取信息，设计了一种通过插入 Log.v 方法来记录 API 调用的方法。通过检查方法，并根据经验确认了获取电话 ID 的 API 调用行为的记录。

1.2.2 国内研究现状

李晖等作者^[6]根据 Android 系统的特性，设计了 X-Decaf (Xposed-based-detecting-cache-file)，并将其与 Xposed 框架相结合，在软件中发现可疑泄漏的路径，并对其进行了监控。

阿里巴巴无线业务部门发布了其第一个大型安卓开放源代码项目 Dexposed，这是 Android 系统下的 AOP 架构。其主要应用领域包括：AOP 编程、测试、性能监控、在线补丁、SDK hooking 等，以提供更好的开发体验。在 Dexposed 中，AOP 的原则来源于 Xposed。在 Dalvik 虚拟机中，主要是在 Dalvik 虚拟机中更改一种方法的定义，即将这种方法的类型变为 native，然后把这种方法的实现链接到一个普通的原生 Dispatch 方法。

该 Dispatch 方法通过 JNI 向 Java 端调用一个统一的处理方式，并在一个统一的过程中调用 before, after 函数来完成 AOP。

刘效伯^[7]设计和开发了 Android 应用中的隐私泄漏探测系统: 通过对 TaintDroid 的缺陷探测技术进行了改良, 能够实现高覆盖率、高效率或高自动化地采集应用中的敏感数据的采集; 采用静态分析方法, 将所抽取到的应用特性与敏感的传输资料相结合, 并对其风险评价, 以获得隐私权泄漏的资讯。本系统采用了图形用户接口, 对安卓应用程序进行了隐私泄漏检测, 并将检测结果以报告的方式进行反馈。

张小贝^[8]综合特征检测、静态检测、动态检测等技术, 提出一种有效、切实可行的联合检测技术, 并根据该技术建立了一种基于网络服务的网络系统, 通过网络服务, 可以检测到 Apk 软件的恶意代码。

王奕钧^[9]提出了一个基于授权机制的隐私保护模型 PRIMOD, 它包含了 Android 平台的体系结构和应用程序, 并对其自身的安全机制和不足进行了分析。

王竹等作者^[10]针对 Android 应用程序中因第三方域名收集用户信息而导致的隐私权泄漏问题, 并利用 TF-IDF 模型和分级聚类技术, 提出了一种基于 TF-IDF 模型的 HostRisk 模型, 该模型利用应用程序中的域名的行为特性, 计算出该域名与应用程序之间的业务关联性, 并采用平均联结的凝聚式分级聚类法, 对没有显示出 App 商业相关行为的业务相关域名进行优化, 最后根据应用中的域名排序, 计算出其隐私权泄漏的风险。试验表明, 该方法是有效且有效的。

杨金宝^[11]做出的用户隐私保护系统是针对 Android 手机应用软件开发的, 它由服务器和客户端两部分组成, 通过模块化的方式来保证服务器安全、手机端的权限监控、信任度分析、隐私策略、日志监控、流量监控等, 并通过详细权限管理、服务器加固、可信度统计、行为审计等权限管理, 使用户隐私权限动态监控管理、可信度审计可视化、文本智能分析检测等功能。

1.3 相关技术的现状分析

1.3.1 App 首次弹窗实现

经过调查, 安卓工程的开发通常都是以 Java 语言为基础, 或者其它框架的开发, 在手机上进行编译和运行。

uni-app 是一种基于 VUE 语法的前端架构, 可以在许多不同的应用平台中进行开发。它允许开发人员在 Android, IOS, H5 和微信, 支付宝, 头条等主流平台上开发和安装应用软件。

鉴于安卓工程的发展方法多种多样，以下将介绍安卓本地的弹窗和 uni-app 两种不同的方法：

(1) Android 原生实现弹窗

Android 原生实现弹窗，会通过变量保存 App 首次进入的状态。首次进入，利用 Android 自带的 SpannableStringBuilder 拼接提示信息并使用 AlertDialog 进行弹窗^[14]。

如图 1-1 所示为 AlertDialog 启动窗口，图 1-2 所示为 SpannableStringBuilder 拼接信息：

```
final AlertDialog alertDialog = new AlertDialog.Builder(this).create();
alertDialog.show();
alertDialog.setCancelable(false);
Window window = alertDialog.getWindow();
if (window != null) {
    window setContentView(R.layout.dialog_initmate);
    window.setGravity(Gravity.CENTER);
    TextView tvContent = window.findViewById(R.id.tv_content);
    TextView tvCancel = window.findViewById(R.id.tv_cancel);
    TextView tvAgree = window.findViewById(R.id.tv_agree);
    String str = "感谢您对本公司的支持!本公司非常重视您的个人信息和隐私保护。";
```

图 1-1 AlertDialog 弹窗

```
String str = "感谢您对本公司的支持!本公司非常重视您的个人信息和隐私保护。" +
    "为了更好地保障您的个人权益，在您使用我们的产品前，" +
    "请务必审慎阅读《隐私政策》和《用户协议》内的所有条款，" +
    "尤其是:\n" +
    "1.我们对您的个人信息的收集/保存/使用/对外提供/保护等规则条款，以及您\n" +
    "的用户权利等条款;\n" +
    "2. 约定我们的限制责任、免责条款;\n" +
    "3.其他以颜色或加粗进行标识的重要条款。 \n" +
    "如您对以上协议有任何疑问，" +
    "可通过人工客服或发邮件至 sharetronicos@163.com 与我们联系。您点击“同\n" +
    "意并继续”的行为即表示您已阅读完毕并同意以上协议的全部内容。" +
    "如您同意以上协议内容，请点击“同意”，开始使用我们的产品和服务!";
SpannableStringBuilder ssb = new SpannableStringBuilder();
ssb.append(str);
final int start = str.indexOf("《"); //第一个出现的位置
ssb.setSpan(new ClickableSpan() {

    @Override
    public void onClick(@NonNull View widget) {
        Toast.makeText(SplashScreenActivity.this, " 《 隐 私 政 策 》 ",
Toast.LENGTH_SHORT).show();
    }

    @Override
    public void updateDrawState(@NonNull TextPaint ds) {
        super.updateDrawState(ds);
        ds.setColor(getResources().getColor(R.color.gaoqing));
        ds.setUnderlineText(false);
    }
}, start, start + 6, 0);
```

图1-2 SpannableStringBuilder 拼接信息

(2) uni-app 实现弹窗

在 uni-app 框架中，弹窗采取配置来实现。在 uni-app 项目 manifest.json 文件的源码视图中找到 app-plus 并在 app-plus 节点上^[15]，添加如图 1-3 所示配置即可。

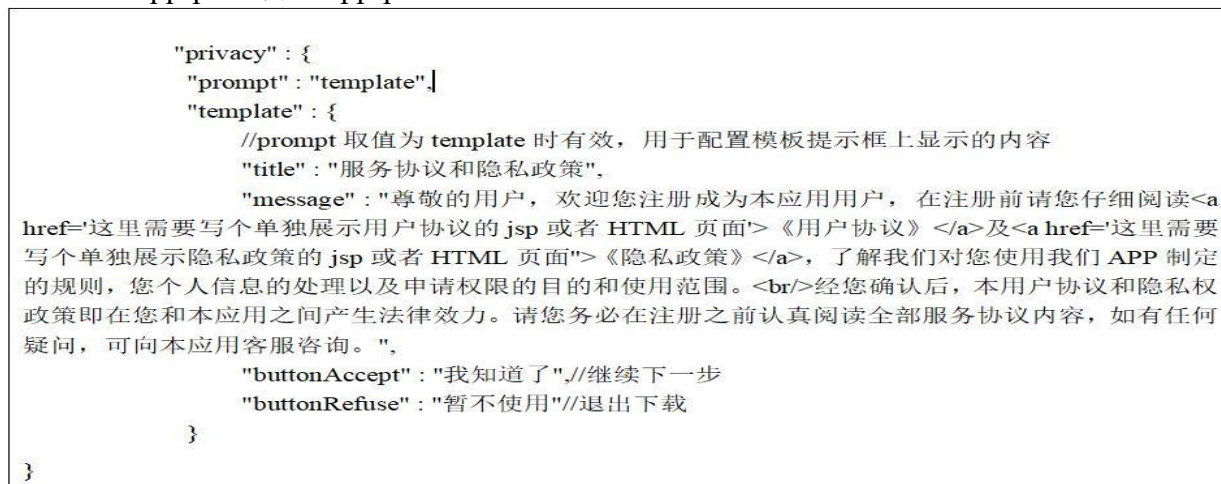


图 1-3 manifest.json 配置

1.3.2 App 隐私政策明示性说明

App 隐私政策明示性的界限很难界定，国家也出台了相关规定来认定 App 违规收集个人信息的违法行为。

应用程序的隐私政策明确的边界比较模糊，而且国家也制定了相应的法律法规，对非法获取用户数据的程序进行了认定。《App 违法违规收集使用个人信息行为认定方法》规定，下列情况属于非法收集个人信息：

- (1) 没有在应用程序中的隐私政策，也没有在隐私政策中收集和利用私人信息的规定；
- (2) 应用程序首次安装运行时，没有以显著的弹窗等方式提醒使用者阅读有关隐私的规定；
- (3) 无法获取诸如隐私政策之类的收集和利用规则，例如，在登录 App 的主页后，需要单击 4 次以上的操作才能访问；
- (4) 应用程序的隐私保护条例等收集和使用规定很难看懂，例如字体太小、颜色太淡、模糊不清等。

更多的细节，请参阅《App 违法违规收集使用个人信息行为认定方法》。根据第一条，系统会检查 App 有没有私密的规定，然后根据第三条，用户在进入 App 后，会不会点击四个以上的按钮，来判定程序是否正确。此外，本论文还就第二条判定规则，对当前市场上一款应用程序弹窗的技术进行了研究，并给出了相应的解释。

1.4 论文结构

第一章：本章主要阐述了选题的背景、意义以及论文后面几章要做的工作。

第二章：在这一章中，我们将对系统中所用到的各种工具和他们在系统中所扮演的角色进行了详细的描述。

第三章：详细介绍了应用程序的体系结构和应用程序的隐私策略，并对应用程序中的各个模块进行了详细的设计。

第四章：本章重点介绍了三大模块的具体实现，并给出了具体的模块设计，并给出了具体的代码说明。

第五章：通过对系统的功能、兼容性和系统的性能进行了详细的测试，以确定系统满足期望的要求。

第六章：对论文的工作进行了总结，并介绍了系统的优点和不足，并对今后的工作进行了展望。

2 系统开发使用的工具和方法

2.1 Android Studio

Android Studio 是谷歌推出的一个 Android 集成开发工具，基于 IntelliJ IDEA，类似 Eclipse ADT，Android Studio 提供了集成的 Android 开发工具用于开发和调试。Dalvik 是为 Android 量身打造的 Java 虚拟机，它与标准 Java 虚拟机 JVM 的差别在于 Dalvik 是基于寄存器设计的，而 JVM 是基于栈结构设计的；JVM 通过解码 class 文件（java 编译生成的：.java—>.class 的 class 文件）中的内容来运行程序；而 Dalvik 运行时是由 java 字节码文件进一步转化而来的文件，并被打包成一个 .dex 可执行文件，Dalvik 虚拟机通过解释 .dex 文件来执行这些字节码，即 android 的 class 文件实际上只是编译过程中的中间目标文件，需要变成 .dex 文件后才能在 dalvik 上运行；Dalvik 能够更快的编译较大的应用程序，允许在有限的内存空间中同时运行多个虚拟机的实例，每一个 Dalvik 应用作为一个独立的 Linux 进程执行，这样可以防止某一虚拟机崩溃时所有的应用都被关闭。

Android 系统是一种多任务操作系统，它能在使用手机的时候，还能完成许多其它的软件。每次运行更多的应用都会消耗更多的内存，当大量的程序被同时运行或者被关闭的程序没有被正确的释放，那么这个系统将会变得更加缓慢，甚至变得不稳定。为了解决这一问题，Android 推出了一种新的生活循环机制。Android 系统的应用软件寿命是通过安卓框架来管理的，而非应用程序的直接控制。一般来说，每个应用（入口通常为活动创建方式）都会生成一个过程。在系统内存快要耗尽时，会根据优先顺序自动恢复过程。无论是用户还是开发者，都不能决定什么时候可以再利用。因此，要有效地预防数据丢失和其它问题，必须要知道生命周期。

2.2 jadx-gui

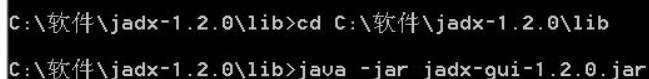
2.2.1 工具介绍

jadx 是一个反向工程的工具，它不但可以对 jar、class 文件进行反向编译，还可以在 Apk、dex、aar 和 zip 中使用 Dalvik 字节码，并从 resources.arsc 中解码 AndroidManifest.xml 和其它资源。jadx 有两个版本，一个是命令行，一个是 UI，jadx-gui 支持突出的关键词，它可以跳跃到类、方法、域声明、方法、查找方法，支持全文检索，可以直接拖动。jadx-gui 的更新非常快，到现在为止，jadx-gui v1.4.0 是在 2022 年 5 月 20 日发布的。jadx-gui 需要 Java 环境的支持，Java 的跨平台特性使得 jadx-gui 可以在 Windows 和

Linux 等主要平台上运行。

反编译的过程与编译的过程正好相反，即将编译完成的低级语言还原回高级语言。例如在 JAVA 中，将编译完成的.class 文件还原回.java 文件的过程叫做反编译。jadx 是一款功能强大的反编译工具，使用起来简单方便（拖拽式操作），不光提供了命令程序，还提供了 GUI 程序。jadx 由 Java 语言编写，使用 Gradle 进行构建。克隆到本地之后，你可以直接使用 Gradle 命令进行构建。一般情况下，为了项目的安全，我们在打包发布一个 apk 之前都会对其代码进行混淆加密比如用无意义的短变量去重命名类、变量、方法，以免代码被轻易破解泄露。经过混淆的代码在功能上是没有变化的，但是去掉了部分名称中的语义信息。为了代码的易读性，我们可以对代码进行反混淆。在 jadx 中，我们通过 Tools -> Deobfuscation 即可开启反混淆功能。

工具的输入是 Apk 文件，Win+ R 进入 cmd, cd 到 lib 的文件夹，然后输入 java-jar jadx-1.2.0.jar 命令执行 jadx-1.2.0.jar 图形接口或双击 jadx-gui-1.2.0.jar，然后打开文件就可以使用：



```
C:\软件\jadx-1.2.0\lib>cd C:\软件\jadx-1.2.0\lib
C:\软件\jadx-1.2.0\lib>java -jar jadx-gui-1.2.0.jar
```

图 2-1 进入图形化界面命令

反编译完成后可以得到以下目录文件：

assets：存放一些静态资源，App 用到的文本文件，图片文件等。

lib：App 开发用到的依赖包，包含第三方依赖库等。

res：存放布局文件，常量字符串文件等。

AndroidManifest.xml：Android 工程中的清单文件，App 申请的权限以及项目的启动类都在该文件中配置。

2.2.2 系统中的使用

jadx-gui 工具功能非常强大，不仅可以反编译 Apk 文件，还包含了反混淆功能。本系统中，用到的是 jadx-gui 的反编译功能。官方文档对于 jadx-gui 的用法通常是 UI 方式，本系统通过将 WanAndroid.Apk 在 jadx-gui 软件中，直接搜索查看隐私政策的实现方式和弹出过程。

2.3 Xposed 框架

Xposed 框架是在著名的智能手机开发者论坛 XDA 上诞生的一款特殊的 Android 系统的 App，由开发者 rovo98 进行开发和升级。Xposed 作为安卓系统下的框架服务程序，可以在不修改系统 apk 的情况下改变系统配置，影响系统运行。其主要功能是建立了一个模块安装平台，在安装 Xposed 框架之后，使用者可以通过安装 Xposed 模块应用的方式，实现强大的功能。Xposed 框架类似于越狱后的 iOS 系统 Cydia 平台，但由于安卓系统的开源性，Xposed 框架可以提供更丰富的功能并让用户获得更好的体验。

2.3.1 Xposed 框架原理

Xposed 框架是一个特别的 Android 应用程序，它是在知名的智能手机开发者论坛 XDA 开发的。Xposed 是一种应用于 Android 的架构，它可以在不修改 Apk 的前提下更改系统的结构，从而影响到系统的正常工作。它的主要功能是搭建一个模组安装平台，用户在安装 Xposed 框架后，就可以利用 Xposed 程序来实现强大的功能。Xposed 框架与后来的 iOS 系统 Cydia 平台很相似，但是 Xposed 框架能够为用户带来更多的特性和更好的体验，这是因为 Android 的开源性。

Xposed 框架^[7]的工作原理是阻止 Android 系统的开始，一般的 Android 程序在启动时会先支持 Zygote 进程，Zygote 是一个安卓进程的孵化器，它的作用是创建 Dalvik 虚拟机的示例，然后装载 Dalvik 虚拟机的资源。Xposed 框架的实现功能是通过修改 /System/bin/app_process 文件来实现，该文件是 Zygote 孵化器在创建 Dalvik 虚拟机时所使用的，它的作用是启动 Android 系统的运行时的库，然后启动虚拟机，修改后的文件可以在 Dalvik 虚拟机创建时使用，Xposed 可以使用 HOOK 来实现所有的功能，Xposed 框架的实现是将一个函数替换为 native 方法 XposedCallHandler，Dalvik 虚拟机在解释执行函数时，如果碰到了在执行中 HOOK 的函数，则会直接调用 Xposed CallHandler，XposedCallHandler，然后调用 XposedBridge 类的 handleHookedMethod 已注册的 beforeHookedMethod。

2.3.2 Xposed 组成部分

Xposed 框架主要由四个部分构成：

- (1) Xposed: Xposed 是由 C++编写的，它的作用是取代/system/bin/app_process 文件，并且为 XposedBridge 提供 JNI (JNI) 方式来进行 Java 与 C++的通讯；
- (2) XposedBridge: Xposedbridge 是 Xposed 所提供的 jar 文件，当 app_process 开始时，为了 HOOK, Xposedbridge 是其它模块开发的基础；
- (3) XposedInstaller: Xposed 的安装文件，它提供 Xposed Framework 的安装、安装、下载、管理等功能，是 Xposed Framework 的管理软件；

(4) XposedMods: XposedMods 是一个以 Xposed 架构为基础的应用程序模块，它具有很强的功能和丰富的性能。

2.3.3 Xposed 框架使用步骤

Xposed 框架的应用分为三个阶段：安装 Xposed Installer，编写 Xposed 模块，向 Xposed 框架中添加一个模块。下面将会详细描述每一步：

(1) 安装 XposedInstaller

因为 Xposed 需要 root 的权限才能进行安装，因此在安装前必须 root (Xposed 在安装时必须 root，在使用时不需要 root)。下载 Xposed 安装程序包、XposedBridge.jar 软件包以及 zip 软件包。把下载好的 Xposed-arm.zip 复制到移动电话内存。安装 Xposed Installer3.0_alpha4.Apk 到移动电话。安装完毕，打开 XposedInstaller，单击 Frame，会发现当前的 Framework 无法运行。关闭并重新启动到恢复模式。Nexus5 的输入方法是关闭，并同时按下电源键和音量减，再用音量键选择“恢复模式”，并确定电源。然后点击安装 zip，电源键决定；接下来，点击“choose zip form/sdcard”按钮，点击“Xposed-arm.Zip”。登录后，他重新启动了手机。重新启动手机，打开 Xposed Installer，单击该框架，显示已完成安装。

(2) 编写 Xposed 模块（在 Android Studio 中编写）

创建一个 Android 工程，将三个 meta-data 加入到 AndroidManifest.xml 文件的应用程序标签中。然后添加 XposedBridge.jar 程序包。将 XposedBridgeApi-54.jar 程序包移至 app->libs 目录，右击->添加库。在软件->主目录下右击创建 assets 文件夹。然后在 assets 目录下创建一个 Xposed_init 文件。Xposed_init 文件为 module 指定了一个条目。开发人员在此类中需要完成 HOOK 代码。

(3) 添加模块到 Xposed

将编写的 module 当做一个正常的 App 安装到手机。由于有些 module 没有编写 Activity，所以模块不会显示在 Launcher 中。打开 Xposed Installer，选择模块，可以在模块列表中看到第 1 步中安装的 module，打钩之后重新启动手机，使模块生效。可以在 Xposed Installer 的日志模块以看到记录手机重启之后运行过程中打开的应用的包名，同时也意味着模块运行成功。

Xposed 框架有两大安全隐患：

首先，在安装 Xposed 框架时，必须使用 root 访问权限，从而使 Android 的安全体系结构发生了变化，导致安卓的沙盒、权限保护等功能失效。比如，如果一部手机上装有 Xposed 框架，那么黑客就可以利用 root 权限来修改手机，并在手机中设置一个后门，

从而威胁到用户的安全。因此，Xposed 框架面临的一个安全隐患就是必须使用 root 的安装。

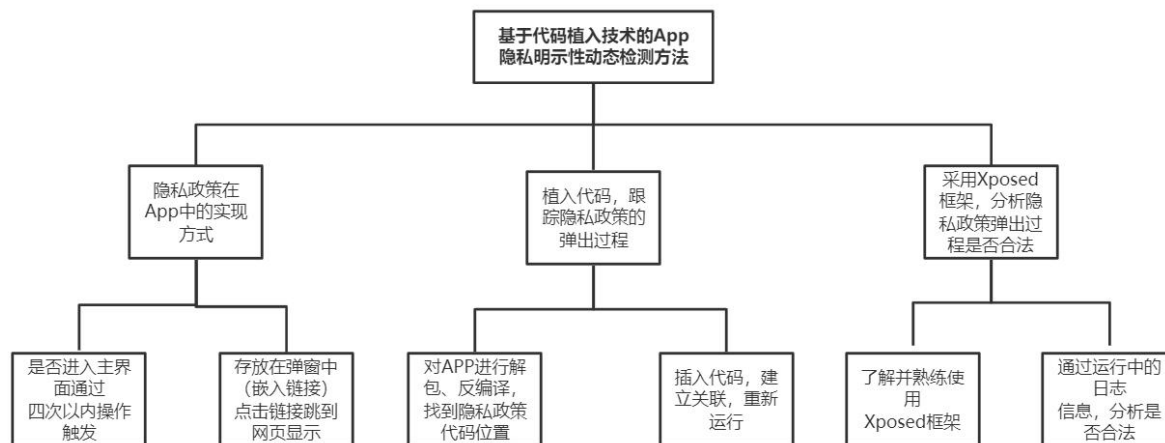
其次，Xposed 框架并未验证加入的 module，从而使攻击者能够在 Xposed Installer 中开发出一些恶意 module。当前，Xposed 安装程序不会对 module 进行安全性分析，它将会对用户的数据安全构成威胁。比如，攻击者可以开发一个恶意的 module，比如 HOOK 系统中的 API，写下 HOOK method，然后在 API 呼叫之前就开始录制，这会危及到用户的安全。因此，Xposed 框架现在面临的另外一种安全隐患是，它没有对新加入的 module 进行分析。

3 系统设计

3.1 功能分析

本课题，主要调研隐私政策在 App 中的实现方式，即调研程序如何将隐私政策集成到代码中，并如何触发隐私政策的弹出和明示；其次采用代码植入技术，跟踪隐私政策的弹出过程是否符合规定（在进入主界面后，不超过四次就能访问到隐私政策）；最后采用 Xposed 框架，驱动 app 运行，通过程序运行中抛出的日志信息，分析弹出过程是否满足相关规定要求。应用程序的隐私政策明示性检查是应用程序合规检查的一个子模块。本系统的主要目的是检测目标 App 的隐私政策的明示性，主要从隐私政策条款入手，通过程序将目标 App 进行反编译后，进行一系列的过滤从而得到相应的隐私政策。本系统主要是通过对 AndroidManifest.xml 文档中所要求的权利和隐私政策的解释进行比较，得出了相应的结论。

具体内容如图 4-1 所示：



3.2 研究思路

- (1) 调研隐私政策在 App 中的实现方式，即调研程序如何将隐私政策集成到代码中，并如何触发隐私政策的弹出和明示；
- (2) 采用代码植入技术，跟踪隐私政策的弹出过程；
- (3) 采用 Xposed 框架，驱动 App 运行，通过程序运行中抛出的日志信息，分析弹

出过程是否满足相关规定要求。

3.3 工作流程

首先调研隐私政策（长文本内容）在 App 中的实现方式;其次采用代码植入技术，跟踪隐私政策的弹出政策；最后调研动态植入方法 Xposed，用来做隐私政策的跟踪。

具体工作流程安排如下图：

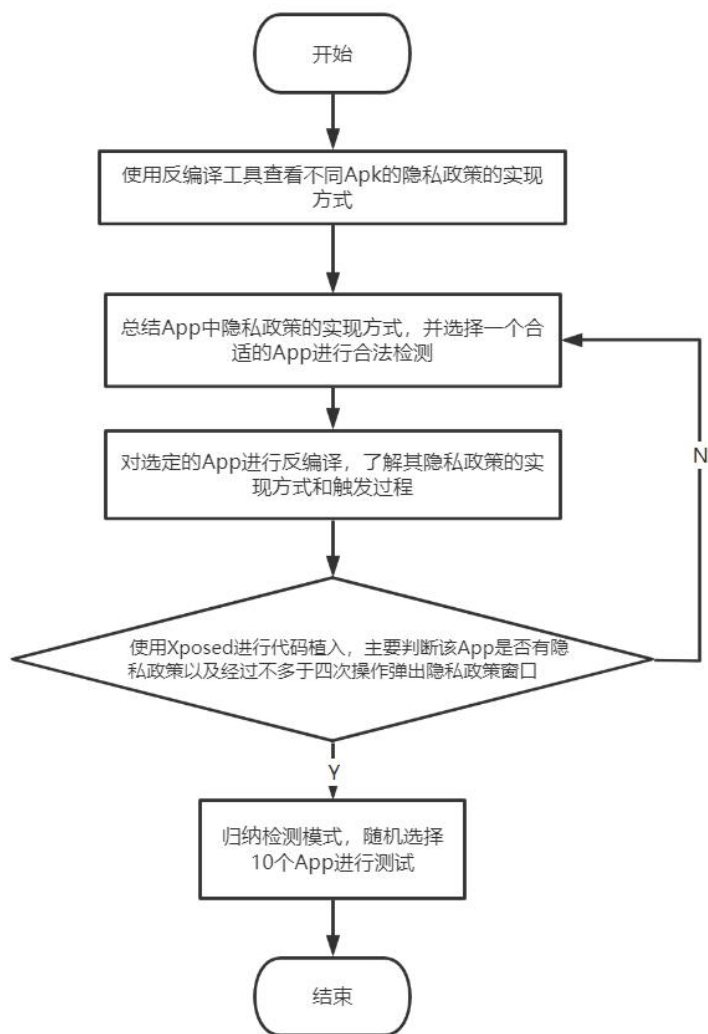


图 3-2 工作流程图

3.4 实现方案

开发环境及工具：Windows 11 x64、JDK1.8、Android Studio、Xposed 等

实现方案：Android 操作系统平台有着良好的开放性，完备的应用程序框架，任何个

人和组织都可以根据其需求进行相应的应用开发，Android 官方也发布了 Android Studio 用来进行应用开发。借助 Android 开发技术，同时，结合软件工程相关技术原理，进行快速开发出一套基于 Android 的 App 隐私政策明示性自动检测系统。

对于用户隐私信息安全，开发出一款隐私政策明示性检测系统是很有必要的。针对不同应用，在开发过程中，更关注于特定权限的检测，如地图软件需要位置权限是必须的，但对于听歌软件则不是必要的。除了功能的实现，同时，应设计良好的交互界面，满足用户在视觉上以及操作上的体验感。在设计完成后，对设计的调系统进行测试，验证其功能可满足性。

本课题将通过对 App 隐私政策的检测和是否越权进行判定某应用是否违规。一方面，分析 App 的个人隐私政策规定是否合理，另一方面采取搜索和识别 App 程序文件和资源文件中是否出现关键词或敏感 API 的调用来检测是否明示隐私政策，同时也会通过分析程序中是否具有敏感信息源到信息汇的数据流来检测用户个人信息泄露问题。

通过了解软件的构架以及相关技术后，从系统的总体需求分析下手，设计系统的整体模型。应用系统的开发应采用面向对象的思想进行各模块功能的设计,将不同的模块功能抽象为类和接口，通过提高代码利用率来更好的实现功能。在对应用程序完成开发后，设计并编辑测试用例对软件进行了考查，期望达到预期效果。

4 系统实现

4.1 隐私政策在 App 中的实现方式

通过大量的调查，我们发现，现在市面上的大部分应用程序都有关于个人隐私的规定，并且首次运行时会以弹窗等明显的形式来提醒使用者阅读相关的隐私保护条例。通过对 100 多种应用程序进行反编译，这些应用程序的隐私策略一般都是长文本网页，主要是通过 URL 来实现。当应用程序安装后，会出现一个“隐私政策同意”窗口，该“隐私政策”会以一个链接的形式出现。此外，还可以在应用程序“关于”的网页上发现一些隐私规则。调查结果显示，当无网路时，个人隐私政策是无法存取的。

下面是一些关于 App 的隐私政策的具体实例：

表 4-1 App 隐私政策的实现方式

App 名称	隐私政策对应的网址	没有网络是否可以访问
蘑菇伙伴	https://h5.mgzf.com/minisite/appStaticHtml/saasPrivacyPolicy.html	否
云达人	file://android_asset/404.html	否
考研题库通	file://android_asset/404.html	否
得物	https://cdn-fast.dewu.com/nezha-plus/detail/61b1c2ca11d1cb2fe24f756a?duWebviewBg=%23ffffff	否

4.2 隐私政策在 App 中的集成

应用程序中必须包含隐私权政策声明，不然就意味着应用程序在执行隐私权政策。在 jadx-gui 中打开 Apk 文件，在“隐私政策”中直接进行查询，以确定应用程序接口文件中的隐私关键字。如果有，就表示应用程序包含了隐私策略；没有，没有提示应用程序没有隐私政策。确定应用程序是否包含了隐私条款，然后在 URL 中进行查询，然后根据这些 URL 来选择与隐私相关的 URL。

打开 AndroidManifest.xml 文件，查找应用程序的启动类；查找对应的路径下的函数和类别，跟踪呼叫流程在 WanAndroid.Apk 中，启动类是 Splash Activity 类，之后会跳转到引导类 GuideActivity；在引导类种又会设置按钮监听；点击 GuideActivity\$onCreate\$1.onClick 会进入到 MainActivity；MainActivity 的 onCreate 方法又会调用 showPrivacyDialog；接着又会调用 com.cxz.wanandroid.utils.DialogUtil 的 getPrivacyConfirmDialog 方法；又设置了关键词点击，点击《隐私政策》会触发 setClickTextView->onTextClick；然后又会调用 com.cxz.wanandroid.p007ui.activity.ContentActivity 的 start 方法，继续调用 start\$default->start，调用完后会调用 initView->initWebView 显示隐私政策界面。

最终，包含隐私政策的在线页面将通过 WebView 组件控制来显示。Web View 组件加载 html 网页是通过函数 loadUrl(网页地址)的方式把网页展示在 App 的窗口中，而不是在浏览器中打开。WebView 对网页的缓存策略有 4 种，分别是 LOAD_CACHE_ONLY：永远不使用网络，只取本地缓存，没有缓存则不会加载。LOAD_CACHE_ELSE_NETWORK：只要本地有缓存，无论是否过期都会去使用本地缓存，没有缓存才会去加载网络。LOAD_DEFAULT：（默认配置）根据 cache-control 决定是否从网络获取。LOAD_NO_CACHE：永远不使用缓存，只从网络获取。通过 setCacheMode 方法设置缓存策略。

4.3 植入代码

先拷贝 XposedBridgeApi.jar 到 WanAndroid 项目的 libs 目录并导入模块，并设置 Configuration 为 compileOnly；在 AndroidManifest.xml 中增加 Xposed 相关内容，即模块的名称、最低版本号、是否配置为 xposed 插件；接着新建 hook 类，编写 hook 代码。实现 IXposedHookLoadPackage 接口，该接口在 App 被加载时调用。然后在 handleLoadPackage 函数内编写 Hook 代码；需要调用 XposedHelpers.findAndHookMethod()函数，其参数分别是要 HOOK 的类,classLoader,方法名,参数,回调对象；还要对所有在隐私政策触发过程中被调用的函数重写 beforeHookedMethod 和 afterHookedMethod 两个方法。这两个方法会在原始的方法的之前和之后执行。使用 beforeHookedMethod 方法来打印/篡改方法调用的参数(通过 param.args) ,甚至阻止调用原来的方法（发送自己的结果）;afterHookedMethod 方法可以用来做基于原始方法的结果的事情。用它来操纵结果，也添加自己的代码，它将会准确地原始方法的前或后执行。具体植入代码如下图所示：

```
/*App 被加载时调用 IXposedHookLoadPackage 接口，对源代码进行植入*/
public class HookYinSi implements IXposedHookLoadPackage {public static int count = 0;
public static Context mContext;
public void handleLoadPackage(XC_LoadPackage.LoadPackageParam
loadPackageParam) throws Throwable {
if (loadPackageParam.packageName.equals("com.cxz.wanandroid")) {
```

```

XposedBridge.log("starthook com.cxz.wanandroid");
/*XposedHelpers.findAndHookMethod(要 HOOK 的类,classLoader,方法名,参数,回调对象)*/
XposedHelpers.findAndHookMethod(ContextWrapper.class,
"attachBaseContext", new Object[] {Context.class, new XC_MethodHook() {
/* class com.lingzhiyi.hookyinsi.HookYinSi.C08761 */
/*access modifiers changed from: protected */
public void beforeHookedMethod(XC_MethodHook.MethodHookParam param)
throws Throwable {
HookYinSi.mContext = (Context) param.args[0];
HookYinSi.super.beforeHookedMethod(param);
}
/* access modifiers changed from: protected */
public void afterHookedMethod(XC_MethodHook.MethodHookParam param) throws
Throwable {
HookYinSi.super.afterHookedMethod(param);
}
}});
XposedHelpers.findAndHookMethod("com.cxz.wanandroid.ui.activity.GuideActivity$onCreat
e$3", loadPackageParam.classLoader, "onClick", new Object[] {View.class, new
XC_MethodHook() {
/* class com.lingzhiyi.hookyinsi.HookYinSi.C08772 */
/* access modifiers changed from: protected */
public void beforeHookedMethod(XC_MethodHook.MethodHookParam param)
throws Throwable {
HookYinSi.super.beforeHookedMethod(param);
}
/* access modifiers changed from: protected */
public void afterHookedMethod(XC_MethodHook.MethodHookParam param) throws
Throwable {
HookYinSi.super.afterHookedMethod(param);
HookYinSi.count++;
Toast.makeText(HookYinSi.mContext, "当前已点按钮" +
String.valueOf(HookYinSi.count) + "次", 0).show();
}
}});

```

图 4-1 植入的代码

4.4 运行结果

当第一次打开并运行 WanAndroid.Apk 时,每当用户进行一次操作,则该界面会跳出弹窗进行显示并统计操作次数,直到隐私政策窗口弹出。具体实现效果如图 5-11 所示;



图 4-2 植入代码前后对比

5 系统测试

系统测试是软件开发完成的最为关键的一步，也是最后一步。系统测试从不一样的角度对软件进行检测以确认它与用户的期望相符。软件测试的目标是找出程序中的错误，从而实现对软件的正确、完全性和质量的确认。这一章主要从系统功能性测试和非功能性测试进行详细的描述。

5.1 测试集

测试集下载来自腾讯应用宝，主要用来测试系统的隐私政策明示性分析和性能。本文将测试集分为了不同大小和不同种类的 10 个测试用例，如表 5-1 所示：

表 5-1 测试集

测试用例	是否含有隐私政策	大小	进入 App 主界面后,访问到隐私政策需要的点击操作次数
Test1(魔力相册.Apk)	是	<50MB(15.7MB)	1
Test2(文具采批.Apk)	是	<50MB(30.4MB)	1
Test3(云达人.Apk)	是	<50MB(38.0MB)	1
Test4(养花大全.Apk)	是	<50MB(33.6MB)	4
Test5(蘑菇伙伴.Apk)	是	<50MB(38.2MB)	4
Test6(雷电模拟器.Apk)	是	≥50MB,<100MB(63.1MB)	1
Test7(考研题库通.Apk)	是	≥50MB,<100MB(53.0MB)	1
Test8(中信书院.Apk)	是	≥50MB,<100MB(78.2MB)	1
Test9(瓦拉格生活.Apk)	是	≥50MB,<100MB(64.86MB)	1
Test10(得物.Apk)	是	≥50MB,<100MB(88.0MB)	1

当中，包括两个 App 进入 App 主界面后，访问到隐私政策需要的点击操作次数为 4 次，以此测试系统的隐私政策明示性分析是否符合预期。将 App 的大小分为两种，一种为 50MB 以下，一种为 50MB 以上，以此测试系统的检测响应时间。

5.2 测试环境

系统测试环境包括很多方面，有有系统测试使用的第三方工具版本，也有测试系统兼容性时采用的移动设备端和 PC 端等，详细环境配置如表 5-2 所示：

表 5-2 测试环境

开发工具	版本号
操作系统	Windows 10 家庭中文版
jadx-gui-反编译工具	jadx-gui1.2.0 版本
Xposed	Xposed Installer3.1.5 版本
Android Studio	版本 2021.11
模拟器	雷电模拟器版本 4.0.79

5.3 系统功能性测试

这一部分重点介绍了该系统的核心模块的功能测试，其目的在于检测其功能是否满足期望的要求。

5.3.1 隐私政策获取的测试

隐私政策获取模块的测试用例包括 App 的 URLS 获取，隐私政策文本获取，隐私政策URL 获取，测试如表 5-3 所示：

表 5-3 隐私政策获取模块测试表

功能模块		隐私政策获取模块			
测试目的		测试隐私政策能否正确获取			
前提条件		网络状态良好			
操作流程	操作描述	数据	期望结果	实际结果	结论
1	文件上传成功后,点击开始检测,等待检测结果	无	检测结果中包含目标 App 的所有 URL	检测结果中包含目标 App 的所有 URL	结果符合预期
2	文件上传成功后,点击开始检测,等待检测结果	无	检测结果中包含目标 App 隐私政策 URL	检测结果中包含目标 App 隐私政策 URL	结果符合预期
3	文件上传成功后,点击开始检测,等待检测结果	无	检测结果中点击隐私政策 URL 可跳转到隐私政策文本	检测结果中点击隐私政策 URL 可以访问到隐私政策	结果符合预期

5.3.2 日志模块的测试

对日志模块的测试用例主要包含反编译日志展示、系统检测进度的实时展示、获取到的 URL 展示、隐私政策获取成功日志展示、可开关日志模块，测试表如表 5-4 所示：

表 5-4 日志模块测试表

功能模块		日志模块				
测试目的		测试日志能否正常打印，测试日志能否正常开启关闭				
前提条件		网络状态良好				
操作流程	操作描述	数据	期望结果	实际结果	结论	
1	点击日志开关按钮	无	日志开启或关闭	日志开启或关闭	结果符合预期	
2	开启日志模块并检测	无	反编译日志展示、反编译日志展示、获取到的 URL 展示、隐私政策获取成功展示	反编译日志展示、反编译日志展示、获取到的 URL 展示、隐私政策获取成功展示	结果符合预期	

5.3.3 隐私政策明示性分析测试

对隐私政策明示性分析进行测试的主要目标是：测试系统能否精准检测出不含隐私政策的 App 和从打开App到触发到隐私政策窗口的操作次数，测试表如表 5-5 所示：

表 5-5 隐私政策明示性分析测试表

功能模块		隐私政策明示性分析				
测试目的		测试系统能否准确检测出不含隐私政策的 App 和经过 4 次点击操作弹出隐私政策的 App				
前提条件		网络状态良好				
测试用例	操作描述	数据	期望结果	实际结果	结论	
Test1	上传并检测	Test1(魔力相册.Apk)	1 次	1 次	符合预期	
Test2	上传并检测	Test2(文具采批.Apk)	1 次	1 次	符合预期	
Test3	上传并检测	Test3(云达人.Apk)	1 次	1 次	符合预期	
Test4	上传并检测	Test4(养花大全.Apk)	4 次	4 次	符合预期	
Test5	上传并检测	Test5(蘑菇伙伴.Apk)	4 次	4 次	符合预期	
Test6	上传并检测	Test6(雷电模拟器.Apk)	1 次	1 次	符合预期	

续表 5-5 隐私政策明示性分析测试表

Test7	上传并检测	Test7(考研题库通.Apk)	1 次	1 次	符合预期
Test8	上传并检测	Test8(中信书院.Apk)	1 次	1 次	符合预期
Test9	上传并检测	Test9(瓦拉格生活.Apk)	1 次	1 次	不符合预期
Test10	上传并检测	Test10(得物.Apk)	1 次	1 次	符合预期

5.4 系统非功能性测试

没有一个软件系统仅仅满足功能要求，它还必须满足某些非功能要求。这个部分将会对系统的非功能性要求进行测试，包括系统兼容和性能测试。

5.4.1 兼容性测试

对于任意一个系统兼容性测试都非常重要。系统在不同装备下运行结果都可能不同，严重的话甚至可能影响到功能的正常运行。因此兼容性测试就变得尤为重要，如表 5-6 为系统兼容性测试：

表 5-6 系统兼容性测试

用例名称	系统兼容性测试
测试目的	验证系统在 PC 端不同浏览器都运行正常验证系统在智能手机上能够满足像响应式布局
测试流程	使用谷歌、火狐、IE 运行系统，检测系统是否正常运行 在智能手机上运行系统，检测布局是否正常
预期结果	PC 端不同浏览器运行正常 智能手机上满足响应式布局，功能正常
实际结果	PC 端不同浏览器运行正常 智能手机上满足响应式布局，功能正常
结论	结果符合预期

5.4.2 性能测试

性能测试对于系统也非常重要，优秀的性能能收获大量的用户。相反，也可以对用户造成极其不好的体验。本文使用不同大小的各类 App 进行了系统检测速度测试，具体测试的结果如表 5-7 所示：

5-7 性能测试表

非功能模块		性能测试		
测试目的		测试系统的检测速度		
前提条件		网络状态良好		
测试用例	操作描述	数据	App 大小	检测时间
Test1	上传并检测	Test1(魔力相册.Apk)	<50MB(17.8MB)	18.57s
Test2	上传并检测	Test2(文具采批.Apk)	<50MB(20.7MB)	5s
Test3	上传并检测	Test3(云达人.Apk)	<50MB(38.0MB)	17.06s
Test4	上传并检测	Test4(养花大全.Apk)	<50MB(33.6MB)	3.84s
Test5	上传并检测	Test5(蘑菇伙伴.Apk)	<50MB(38.2MB)	4.80s
Test6	上传并检测	Test6(雷电模拟器.Apk)	≥50MB,<100MB(63.1MB)	16.79s
Test7	上传并检测	Test7(考研题库通.Apk)	≥50MB,<100MB(53.0MB)	39.36s
Test8	上传并检测	Test8(中信书院.Apk)	≥50MB,<100MB(78.2MB)	50.12s
Test9	上传并检测	Test9(瓦拉格生活.Apk)	≥50MB,<100MB(64.86MB)	28s
Test10	上传并检测	Test10(得物.Apk)	≥50MB,<100MB(88.0MB)	89s

通过观测性能测试表,可以得知,干扰检测速度的因素有很多,当 App 不存在隐私政策时,则后续检测步骤会省略,检测时间当然会缩短。但当 App 的体量太大则会导致反编译占据大量时长或筛选出的 URL 较多,致使爬虫执行时间漫长,最后检测时间也变长。

5.5 本章小结

这一章重点讨论了对该系统进行的测试。重点对系统功能测试中的核心模块进行了详细的测试和介绍。其中,文档上传模块、隐私政策获取模块、日志模块、隐私政策分析模块等模块。此外,还对该系统进行了一些非功能性的测试,如兼容性和性能。通过与实验数据进行比较,得到了期望与现实的结果,从而保证了该系统能够满足用户的功能要求,并具有较好的稳定性。

6 总结和展望

6.1 本文工作总结

目前，安卓的个人信息安全问题已成为全社会的焦点。本论文以个人隐私政策为切入点，对一些尚未明确规定的隐私政策进行了探讨。本文针对应用程序列表文件的权限和隐私策略，采用了相应的技术实现了对隐私政策的明确识别。

本论文主要围绕《App 违法违规收集使用个人信息行为认定方法》第 3 条进行设计和实施。第三条条款：在登录应用程序的首页之后，超过 4 次的点击就可以获得隐私保护。文章以此为基础，对应用程序的检测过程进行了设计。

本文通过在 jadx-gui 中直接键入文本检索，从而有效地提高了检索的速度。如果用 Java IO 流程来完成这一步骤，将耗费很多资源和时间。应用程序的反编译目录中包含了很多的文件夹，在阅读 JavaIO 流时要进行很多的递归操作，这就造成了很低的工作效率。但是，使用文字检索会造成大量的环境内存损耗，对其它系统的运行非常不利。

最后，在测试的结果中，我们将会给出一个关于是否弹出和弹出次数等信息，来帮助用户对应用程序的违规行为有一个完整的认识。

6.2 展望

由于本论文的研究重点在于《App 违法违规收集使用个人信息行为认定方法》第三条的实施和检查，因此在其它方面的检查上存在着不足。例如，《App 违法违规收集使用个人信息行为认定方法》中就有一条关于应用程序中没有隐私政策或者隐私政策中关于收集个人信息的规定。本文并没有针对此问题进行相应的测试设计和实施，这是未来系统升级时需要进行的一个新的内容。

另外，在未来，系统会在应用程序中第一次使用弹窗。通过这种测试，可以判断应用程序在第一次使用的时候，有没有关于隐私的提示，或者是默认的。

在今后的工作中，我们将重点研究《App 违法违规收集使用个人信息行为认定方法》中的其它判定方法，并通过不断地完善和改进，使其更加精确，更加安全地使用 App。

致 谢

时间总是在不知不觉中流逝，大学四年的生活终于结束了。随着时光流逝，我的论文即将写完，在此，我要向那些曾经给予我帮助和关怀的人们表示衷心的感谢。

首先，我要向刘晓建老师表示衷心的感谢！本论文是在刘教授的精心指导下，严格要求下进行的。在本课题的研究与撰写的过程中，刘教授给予了我大量的耐心指导与启发。我不仅学到了老师的博学，也学到了一丝不苟的精神。在毕业设计的前期，我对这个项目的认识还不够透彻，因此，在完成了毕设项目之后，就一直没有进展。刘老师在每一次会议上都会给我一点实施的想法，渐渐的，就形成了现在的体系。

最后，我要向所有在四年前指导我的老师表示感谢。书山有路勤成道。四年的学习虽然苦，但也有一些甜蜜，有了这些老师，他们的痛苦就会减轻一些。教师把复杂的知识用简单的方法解释，把复杂的东西变得简单，这是我无法完成的。所以，在四年的时间里，我对这位老师表示衷心的感谢。

我也要谢谢我的同学和室友，他们在我有困难的时候，为我提供了宝贵的时间。这四年来，我曾无数次体会到友情的强大，我始终坚信友谊会长久，哪怕马上就要分开，我也明白，这是一种更好的团聚。

最后，我要再次向所有关心和帮助我的人表示感谢！

参考文献

- [1] 方琪. 基于静动态结合方式的 Android 恶意软件检测技术研究[D].南京:东南大学,2020
- [2] 罗洋. Android 平台程序动态分析系统的研究与实现[D].北京:北京邮电大学,2016
- [3] Li, S., Chen, J., Spyridopoulos, T., Andriotis, P., Ludwiniak, R., & Russell, G. (2015). Real-time Monitoring of Privacy Abuses and Intrusion Detection in Android System. In 3rd International Conference on Human Aspects of Information Security, Privacy and Trust (Vol. 9190, pp. 379-390). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-319-20376-8_34
- [4] Lo, Nai-Wei, Yeh, Kuo-Hui, Fan, Chuan-Yen. Leakage Detection and Risk Assessment on Privacy for Android Applications: LRPdroid[J]. IEEE systems journal, 2016, 10(4)
- [5] Naoya Kajiwar, Shinichi Matsumoto, Yuuki Nishimoto, Yoshiaki Hori, Kouichi Sakurai. Detection of Privacy Sensitive Information Retrieval Using API Call Logging Mechanism within Android Framework[J]. Journal of Networks, 2014, 9(11)
- [6] 李晖, 王斌, 张文, 汤祺, 张艳丽. X-Decaf: Android 平台社交类应用的缓存文件泄露检测[J]. 电子与信息学报, 2017, 39(01): 66~74
- [7] 刘效伯. Android 系统隐私泄露检测与保护研究[D].南京:东南大学, 2017 年
- [8] 张小贝. 基于 Android 平台的恶意代码检测技术研究[D].北京:北京邮电大学, 2017 年
- [9] 王奕钧. 基于 Android 权限机制的隐私保护方法研究[D].哈尔滨:哈尔滨工程大学, 2016
- [10] 王竹, 贺坤, 王新宇, 牛犇, 李凤华. Android 设备中基于流量特征的隐私泄露评估方案[J]. 通信学报, 2020, 41(02): 155~164
- [11] 杨金宝, 马宝泽, 叶清. 面向 Android 手机应用程序的用户隐私保护系统研究[J]. 计算机与数字工程, 2019, 47(09): 2206-2211
- [12] DCloud. 什么是 uni-app[EB/OL]. 2020.
<https://github.com/dcloudio/uni-app/blob/master/docs/README.md>
- [13] 张晓明. 基于 uni-app 和 Android 的学生手机管控系统的设计与实现[D].兰州:兰州大学, 2020
- [14] Snow_Ice_Yang. APP 启动页隐私弹窗实现说明[EB/OL]. 2019.
https://blog.csdn.net/Snow_Ice_Yang/article/details/103637642
- [15] collect 哟吼. 首页服务协议和隐私政策弹窗[EB/OL]. 2020.
https://blog.csdn.net/weixin_44948683/article/details/106422131
- [16] 孟杨. Android 平台框架层 hook 技术的安全性研究[D].北京:北京邮电大学, 2017

[17] 李伟,陈忠红.安卓 Xposed 框架安全应用研究[J].电脑知识与技术,2016,12(10):49~51