编号_____

西安科技大学

毕业设计（论文）

（ 2021 届）

题　　目	App 隐私政策明示性自动检测系统
学生姓名	高洋洋
学　　号	17408070726
专业班级	软件工程 1703
指导教师	刘晓建
所在学院	计算机科学与技术学院
日　　期	2021 年 6 月

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：高洋洋　　日期：2021 年 6 月 7 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：高洋洋　　指导教师签名：刘晓建

2021 年 6 月 7 日

分 类 号　　密 级
学校代码 10704　　学 号 17408070726

西安科技大学

学 士 学 位 论 文

题目：App 隐私政策明示性自动检测系统

作者：高洋洋

指导教师：刘晓建　　专业技术职称：副教授
学科专业：软件工程　　申请学位日期：2021 年 6 月

摘　要

随着智能手机的普及，各种各样的 App 进入人们的生活中。这些 App 在方便人们生活的同时，也在收集着使用者的个人信息。大多数 App 是正规的，它们会在征用使用者信息时给予提示。例如，首次安装时弹出的隐私政策协议，其中规定了 App 收集信息的目的和用途，但也有部分 App 收集信息时未提示用户，甚至在发布时没有任何有关隐私政策的声明。App 的隐私政策声明已经成为 App 规范管理的重要方面，很多 App 由于未经提示而收集用户信息的行为，而被判定为违规，遭到下架的处理。

由于 App 程序以一组代码文件的形式存在，依靠人工代码审查或者动态测试方法检测程序是否具有隐私明示，需要消耗大量的人力和时间，特别是对信息安全监管部门，几乎不可能通过运行每个 App 程序来判断其合规性。因此，有必要研究一种静态检测 App 合规性的方法，并开发一套相应的检测工具。

本文设计的系统能够检测出 App 中是否存在隐私政策以及隐私政策中是否对 App 申请的权限进行了说明。通过对比隐私政策和权限文件，检测出 App 申请的不合理权限。该系统可以帮助审核 App，在一定程度上将人们从复杂的检查审核任务中解放出来。本文的主要工作如下：

1. 了解了国内外对 App 隐私信息的相关研究。介绍了本系统中使用到的第三方工具以及它们在系统中担任的角色。

2. 通过阅读 App 违法违规收集使用个人信息行为的认定方法等相关法律、规定和技术标准，提炼出了隐私政策明示性的认定标准，并根据这些认定标准，研究了隐政策明示性检测方法，设计和开发了相应的检测系统。

3. 构建了包含不同种类和不同大小的 App 的数据集，开发了测试用例集，对系统的能、性能和兼容性进行了相应测试。测试结果显示，隐私政策明示性检测方法和相关系统能够有效检测出 App 中申请的不合理权限，取得了显著的效果。

关键词：隐私政策；权限文件；静态检测；认定方法；App

ABSTRACT

With the development of smart phones, various Apps have entered people's lives. While these Apps facilitate people's lives, they also collect user information. Many Apps are formal, and they give prompts when requisitioning user information. For example, the privacy policy that pops up for the first installation specifies the purpose of the App for collecting information, but some Apps do not prompt the user when collecting information, and even do not have a privacy policy at the time of publishing. The privacy policy statement has become an important aspect of App's standardized management. Many Apps have been judged as violations and removed from the shelves because they collected user information without prompts.

An App program is a set of code files. It takes a lot of manpower and time to rely on manual code review or dynamic test methods to detect the explicitness of the program. Especially for the information security supervision department, it is almost impossible to judge its compliance by running the App program. Therefore, it is necessary to study a testing tool for statically testing App compliance.

The system can detect whether there is a privacy policy in the App and whether the permissions applied by the App are explained in the privacy policy. Detect unreasonable permissions of App by comparing privacy policies and permissions files. Its system can audit App, liberating people from complicated auditing. The main work of this thesis is as follows:

1. Understand the relevant research on App privacy information at home and abroad. The third-party tools used in this system and their roles in the system are introduced.

2. By reading App’s laws, regulations and technical standards for identifying methods of collecting and using personal information in violation of laws and regulations,we extracted the explicit identification standards of the privacy policy, and obtained detection methods and designed and developed the system.

3. Constructed App data sets of different types and sizes, developed a set of test cases, and tested the function, performance and compatibility of the system. The results show that the explicit detection method of privacy policy and related systems can effectively detect unreasonable permissions applied for in the App, and have achieved significant results.

Key Words: privacy policy; permission file; Static detection; Identification method; App

目 录

1 绪　论

1.1 本课题研究的背景及意义

1.1.1 研究背景

随着移动互联网时代的到来，智能手机作为接入移动互联网的设备，已经在全世界得到了广泛使用，并因此改变了人们的生活习惯和生活方式。在当前的移动操作系统中，Android 以其开源性和高性价比成为了人们使用最多的移动操作系统，并由此产生大量功能丰富的第三方应用，但其中不乏带有恶意行为的应用，这些应用会在用户不知情或者未意识到的情况下，对用户的隐私信息进行收集。例如，“宾果消消消”、“我的汤姆猫”、“和平精英”、“天天飞车”等多款游戏，在 App 首次运行时未通过弹窗等明显方式提示用户阅读隐私政策等收集使用规则，或以默认选择同意隐私政策等非明示方式征求用户同意。因此，有必要研发一种可以检测 App 隐私政策是否明示的系统。

1.1.2 研究意义

当今智能手机已成为人们生活中必不可少的设备，运行在其上的大量 App 在改变我们生活方式的同时也在收集着我们的各种隐私信息，但有的 App 没有经过用户的同意，而是选择默认同意的方式。这样用户的信息在不知情的情况下被泄露出去。对于这种隐私政策未进行明示的行为，需要能够预先检测并提示，提示用户使用的 App 是否有隐私政策的明示以及收集个人信息的可能性，让用户进行判断是否需要继续使用该 App。

另一方面，本系统也可以作为一个软件评测工具。目前很多的 App 存在违法违规收集用户信息的行为，由于软件作为一种软件制品，物理上难以观察和触摸，这就给包括政府监管部门、App 开发者和 App 使用者对违规行为的认定带来困难。认定的困难直接导致 App 的检测需消耗大量的人力和时间，而检测系统可以将人从中解放出来。随着系统的不断升级，为用户安全使用 App 带来保障的同时，也可以为政府部门的认定工作提供技术支持。最后，检测隐私政策的明示性也可以间接作为一个评判标准，用来评判 App 的合规性。

1.2 国内外研究现状

1.2.1 国内研究现状

孟霞和岳鹏宇[1]进行了移动终端 App 隐私政策内容的分析，结合隐私政策文本中的具体表述，对移动终端 App 隐私政策现状、问题的驱动因素和内在原因进行全面分析，研究了我国移动终端隐私政策存在的意义和深层症结。并根据此研究结论，构建了移动终端 App 隐私政策的编写基本要素，并建议企业贯彻国家标准，针对终端优化、创新信息授权机制，政府部门加强监管评审，并适当借助第三方社会力量监督。

秦克飞[2]对手机 App 隐私政策的可读性进行了研究。旨在为手机 App 企业制定更具可读性的隐私政策提供决策参考。通过采用可读性公式法和非参数检验方法对 63 款手机 App 隐私政策的可读性进行了定量研究。结果显示：对现阶段国内大多数网民而言，隐私政策的可读性低，阅读和理解的难度大；根据研究提出了制定隐私政策时的注意事项：（1）多使用短句，力求简而短，使用户不至于被字里行间的复杂语义关系所迷惑；（2）少用非常用字，对于一些英文单词术语尽量使用中文含义表达或添加便于用户阅读的相关解释页面的链接。

张梦秋[3]从 App 模块、用户隐私保护意识、外部攻击三个方面探讨了用户隐私泄露的主要原因。App 模块中，主要因为 App 经营者过度获取个人信息，用户操作不当和黑客恶意攻击造成用户隐私泄露。针对用户隐私保护意识薄弱，提出了三个对策，规范使用隐私条款和健全法律法规监管机制以及用户规范操作。对于外部攻击则建议，提高 App 安全防护机制和加强 App 应用市场的监管。

刘百灵和万璐璐[4]结合移动商务环境的特点，在 Petri 网的基础上设计了基于移动服务的隐私政策 Petri 网协商算法，该算法兼顾了用户的隐私偏好与服务商的隐私政策，双方以移动服务为粒度进行隐私政策协商，并支持协商冲突检测与缓解，实现了移动用户的服务需求与隐私保护之间的权衡。仿真实验发现，相比传统的隐私政策协商方法，该算法有更高的协商效率和协商成功率。

李清文的工作[5]探究了社交网络隐私保护的现状及大数据给社交网络隐私侵权带来的新威胁；分析了社交网络隐私侵权的原因，进而从用户的信息安全素养、安全防护技术以及法律法规方面提出了相应的对策，以期待在大数据时代的背景下，保证社交网络中的个人信息安全。

杨金宝等人[6]实现了一种面向 Android 手机应用程序的用户隐私保护系统，该系统由服务器端和客户端两部分组成，运用模块化设计方法实现了服务器安全化、手机端权限监控/控制、可信程度分析、隐私策略、日志监控、流量监控等模块，综合采用了细粒度权限管理、服务器加固、可信度统计、行为审计等权限管理机制，实现了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。

刘娇、白净等人[7]对 75 款 App（55/20）调研发现，中外 App 用户隐私声明文本存

在许多差异。通过调研 55 个中文和 20 个英文移动 App 用户隐私声明文本内容，查看这些 App 用户隐私声明的名称、张贴位置、文本内容详细程度、App 获取用户权限数量、文本是否包含告知用户“与第三方分享”内容表述、是否包含告知用户“数据加密技术”内容表述、是否包含“ 未成年使用问题”表述等七个部分，找出了中文和英文移动 App 隐私声明文本中的差异，检视这些文本是否符合标准隐私声明文本框架。

王靖瑜等人[8]提出了一种自动化检测应用隐私条例文档是否与应用行为相一致的工具。首先，使用一种改进的自然语言处理方法提取隐私条例文档中的隐私信息和应用敏感行为；然后，使用静态分析和动态分析相结合的方法分析应用实际的隐私行为，同时区别于传统的白名单对照方式，使用了基于聚类的第三方库的检测方法提高了检查的准确性，最后将文本中声明的隐私信息行为和代码中分析出的隐私权限进行一致性校验。实验对 455 个应用进行分析，得出大约有 50%的应用存在着应用行为和隐私条例文档不一致的问题。

张永兵[9]深入研究了隐私信息检索技术在位置隐私保护中的应用。随着云计算的发展与普及，大量的位置数据外包给云服务器，使隐私信息检索技术得到了快速发展。在该模式中，用户可以在不是露用户检索的数据项信息的前提下，检索不可信服务器上的任意数网。基于 PIR 的位置隐私保护技术的基本原理为：将数据库假设为一个字符串 x，该字符串由 n 位的二进制数构成，当用户查询字符申中第 i 位值 xi 时，直接发起查询肯定会泄露该值的相关信息，从而导致隐私泄露。为了保护数据隐私，用户在发起查询之前。先对查询 i 进行加密，然后将经过加密的 LBS 查询 q 发送给服务器。服务器收到查询请求后，对数据库进行查询，将查询结果返回给用户。用户收到查询结果后，进行解密操作，得到最终查询结果 Xi。将 PIR 方法应用在位置隐私保护中，增强了位置总私保护效果。但是，该方法大大增加了软硬件资源开销，并且降低了检索效率。

1.2.2 国外研究现状

Muharman Lubis，Rahmat Fauzi，Ahmad Almaarif 等作者[10]研究了将在未来几十年内通过收集符合条件标准的所有经验证据，对涉及隐私保护驱动因素的研究问题做出细致的总结，从而探索隐私权的概念。这些问题与隐私保护的驱动力有关。

Zi-Peng Zhang，Ming Fu，Xin-Yu Feng 提出了一种基于信息流控制的防止合谋信息泄漏的混合方法[11]。结合了静态信息流分析和动态运行时检查。通过静态信息流控制防止单个进程引起的信息泄漏，并在运行时进行动态检查以防止合谋信息泄漏。这样的组合可以有效地减少纯动态检查的运行时开销，并减少纯静态分析中的错误警报。

Le Yu，Xiapu Luo，Jiachi Chen，Hao Zhou，Tao Zhang，Henry Chang，Hareton K N Leung 提出一种新颖的方法来自动识别隐私政策中的五种问题，对隐私政策进行了系

统的研究[12]。并在名为 PPChecker 的系统中实现了该方法，并通过实际应用及其隐私权政策对其进行了评估。实验结果表明，PPChecker 可以高精度地有效识别可疑的隐私策略。将 PPChecker 应用到 2500 个流行的应用程序中，发现 1,850 个应用程序（即 74.0%）存在至少一种问题。这项研究为改善和规范应用程序的隐私政策提供了新的思路。

Abu Bakar Anizah，Mahinderjit Singh Manmeet，Mohd Shariff Azizul Rahman 提出了一种使用树结构和演算知识的形式化的隐私模型 PRiMo[13]。应用传感器移动数据收集器（AMoDaC）是在现实生活中开发和实现的，用于通过授予的权限和所涉及的风险来分析移动应用访问的用户数据。通过将拟议的 PRiMo 结果与现有的可用测试指标进行比较，提出了一个基准。结果表明，与其他类别的应用程序相比，“工具与实用程序/生产力”应用程序构成最高风险。此外，有 29 位用户面临低风险和可接受的风险，而有 2 位用户面临中等风险。根据建议的基准，面临低于 25％风险的用户被认为是安全的。这项工作的有效性和准确性为 96. 8％。

Xing Liu，Jiqiang Liu，Sencun Zhu，Wei Wang，Xiangliang Zhang 通过研究流行的 Android 应用程序中集成的分析库收集的信息，设计并实现了一个称为“ Alde”的框架[14]。对于给定的一个应用程序，Alde 同时使用静态分析和动态分析来检测由分析库收集的用户的应用程序内操作。通过分析应用程序的隐私策略，以查看应用程序开发人员是否已通知用户其应用程序内操作数据是由分析库收集的。除此之外，开发了一个名为“ALManager”的应用程序，该应用程序利用 Xposed 框架来进行管理其他应用程序中的分析库。

Cha Youngrok，Pak Wooguil 提出了一种用于隐藏与用户联系人相关的数据或在 Android 应用请求访问它们时根据预先配置的策略提供虚拟数据的系统[15]。通过隐藏与联系人有关的数据，提出的系统可以保护他们免受恶意应用的攻击。通过使用虚拟数据，它甚至可以检测到泄漏私人数据的恶意应用。与类似问题的通用解决方案相比，该系统需要较少的存储空间并提供对用户联系人的更快访问。

Baalous Rawan，Poet Ronald 研究了句子嵌入的优势和局限性，以检测 Android 应用程序隐私策略中的危险权限[16]。隐私策略分析依赖于理解句子的含义，以便识别隐私相关应用程序感兴趣的句子。在研究中，Sent2Vec 句子嵌入模型已在 130,000 个 Android 应用程序的隐私权政策中得到利用和训练。作者将句子嵌入模型提取的术语与 564 个隐私策略的数据集上的黄金标准进行比较后为相关的研究人员和开发人员提供答案。此外，它还可以帮助有兴趣部署句子嵌入模型的监管机构检查隐私策略是否符合政府法规，并找出不一致或违反的地方。

1.3 相关技术的现状分析

1.3.1 静态分析技术

1. 反编译

为了实现对程序的静态分析，首先需要对 Apk 文件进行反编译处理，得到以下文件目录的结构：

表 1-1 apk 反编译主要目录

assets 目录
lib 目录
res 目录
smali 目录
AndroidManifest. xml

其中，assets 目录下存放普通的文本和图片资源，lib 目录存放程序依赖的第三方工具，res 目录下存放程序的界面文件等，smali 目录是程序在 Dalvik 虚拟机运行的一种字节码，AndroidManifest.xml 文件是程序申请的权限文件，也是系统着重分析的文件！

目前市面上拥有很多用于安卓逆向工程的第三方工具，其中包含 Apktool，dex 文件反编译工具-dex2jar，jar 反编译工具-jd-gui，jadx 工具等等。本文使用的是 Apktool 和 jadx 这两种工具。Apktool 工具目前已经更新到 Apktool 2.5.0 版本，它支持命令行的形式，适合集成到程序中，并能输出 smali 文件便于检索，因此考虑选择了 Apktool 工具。jadx 工具拥有图形化界面，操作容易，适合前期分析 apk 使用，本文正是使用该工具进行了前期的人工分析步骤。

2. XML 解析

反编译后产生的文件中包含大量的 XML 文件，XML 文件的分析是必不可少的。XML 是一种用于标记电子文件使其具有结构性的标记语言[17]。它拥有易读性，可扩展等优点，对开发人员非常友好！XML 本身是作为一种编码的格式，以纯文本对数据进行保存，因此读出 XML 文件中包含的有效信息，必须先将保存的信息解析出来[18]。

XML 的解析是一个把代表 XML 文档的无结构的字符序列转换为满足 XML 语法的结构化组件的过程，基本的解析方法有两种，一种是基于事件流的解析 SAX，另一种是基于文档树结构的解析 DOM。在具体实现上，本文采用 Dom4j 解析工具。Dom4j 是一个非常优秀的读取 XML 文件的开源软件，具有性能优异、功能强大和易于使用的特点。

1.3.2 余弦距离

1. 余弦距离的计算方式

余弦距离，也称为余弦相似度，是利用向量空间中的两个向量夹角的余弦值作为衡量两个个体间差异的大小度量。余弦值越接近 1，就表明夹角越接近 0°，也就是两个向量越相似，这就是余弦相似性。

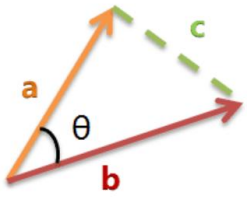

图 1-1 二维向量夹角的余弦值

如图 1-1 所示，以二维空间为例，计算 a 和 b 向量的夹角θ。余弦定理可知，利用公式（1-1）可得。

$$\cos\theta = \frac{a^2 + b^2 - c^2}{2ab} \tag{1-1}$$

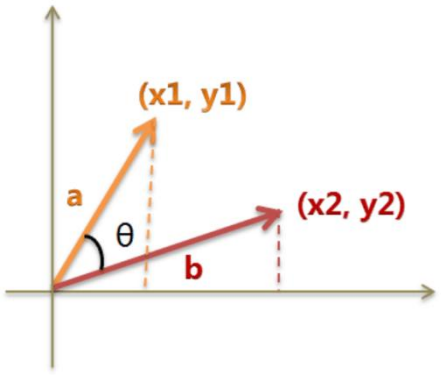

图 1-2 二维坐标中向量的夹角

如图 1-2 所示，假定 a 向量为(x1,y1)，b 向量为(x2,y2)，根据余弦定理得到公式(1-2)。

$$\cos\theta = \frac{x_1 x_2 + y_1 y_2}{\sqrt{x_1^2 + y_1^2} \times \sqrt{x_2^2 + y_2^2}} \tag{1-2}$$

已经得到证实，余弦的计算公式同样适用于 n 维向量。已知 A 与 B 两个 n 维向量，A=(A1,A2,...,An)，B=(B1,B2,...,Bn)，则 A 与 B 的夹角θ的余弦值为：

$$\cos\theta = \frac{\sum_{i=1}^{n}(A_i \times B_j)}{\sqrt{\sum_{i=1}^{n}(A_i)^2} \times \sqrt{\sum_{i=1}^{n}(\mathrm{B}_i)^2}} = \frac{A \cdot \mathrm{B}}{|A| \times |B|} \tag{1-3}$$

2. 基于余弦距离的文本相似度

系统检测需要获得 App 中的隐私政策条款。经过调研发现，目前绝大多数 App 隐私政策是通过访问 URL 获得，隐私政策的 URL 存在于反编译后的程序资源中，但是程序资源中存在大量的 URL，导致通过 URL 获取的字符串不知是否属于隐私政策。

基于余弦距离的文本相似度可以求取两篇文本的相似度，以此筛选出隐私政策。本系统采用的就是求余弦相似度的方法。

根据公式(1-3)可知，求余弦距离需要向量作为基本。文本中并不存在向量，所以首先需要将文本转换为一个 n 元向量。本系统利用了分词工具对文本进行分词，之后统计词语出现的词频，通过词频建立起一个 n 元向量。

1.3.3 App 首次安装弹窗实现

经调研发现，Android 项目的开发可以是 Java 语言开发或是其他框架开发完成的项目编译运行在移动设备端。

uni-app 是一个在 VUE 语法基础上进化而来的，可以在多种不同应用平台内进行开发的现代化前端框架[21]。它可以让开发者使用框架内部 API 实现在 Android、IOS、H5，以及诸如微信、支付宝、头条等主流厂商平台小程序的产品研发及部署[22]。

由于 Android 项目开发的不同方式，下面介绍 Android 原生实现弹窗和 uni-app 实现弹窗两种不同方式：

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[23]。

图 1-3 为 AlertDialog 启动弹窗代码，图 1-4 为 SpanableStringBuilder 拼接信息代码：

```
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 1-3 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 sharetronicios@163.com 与我们联系。您点击“同意并继续”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 1-4 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[24]，添加如图 1-5 所示配置即可。

```
            "privacy" : {
             "prompt" : "template",
             "template" : {
                  //prompt 取值为 template 时有效，用于配置模板提示框上显示的内容
                  "title" : "服务协议和隐私政策",
                  "message" : "尊敬的用户，欢迎您注册成为本应用用户，在注册前请您仔细阅读<a href='这里需要写个单独展示用户协议的 jsp 或者 HTML 页面'>《用户协议》</a>及<a href='这里需要写个单独展示隐私政策的 jsp 或者 HTML 页面'>《隐私政策》</a>，了解我们对您使用我们 APP 制定的规则，您个人信息的处理以及申请权限的目的和使用范围。<br/>经您确认后，本用户协议和隐私权政策即在您和本应用之间产生法律效力。请您务必在注册之前认真阅读全部服务协议内容，如有任何疑问，可向本应用客服咨询。",
                  "buttonAccept" : "我知道了",//继续下一步
                  "buttonRefuse" : "暂不使用"//退出下载
             }
}
```

图 1-5 manifest.json 配置

1.4 论文结构

第一章：在本章节中，主要描述了本课题的研究背景和意义，并介绍了本文后续章节的前置知识。

第二章：在本章节中，详细介绍了系统开发中使用的工具以及它们各自在系统中担当的角色。

第三章：在本章节中，进行了系统的架构设计和 App 隐私政策明示性说明，并完成了 App 检测流程中模块的设计。

第四章：在本章节中，主要针对系统的 6 个模块展开了详细的实现描述，包括模块的实现细节和核心代码解释。

第五章：在本章节中，针对系统的功能、兼容性以及性能三方面进行了详细测试，以此验证系统是否符合预期。

第六章：在本章节中，对本文的工作内容进行了总结以及对系统的优缺点和将来可能改进的地方进行了描述。

2 系统开发使用的工具

2.1 Apktool 反编译工具

2.1.1 工具介绍

Apktool 是一个用于将 apk 文件反向编译的工具，它能够最大程度的还原 apk 文件为最初的源代码形式。Apktool 版本迭代迅速，截至目前，最新版本为 Apktool v2.5.0，它于 2020 年 12 月 2 日发布。使用 Apktool 需要 Java 环境的支持，Java 的跨平台性使 Apktool 能够运行在各大平台，例如 Windows 和 Linux 系统。

工具的输入为 Apk 文件，通过命令的方式进行反编译操作：

图 2-1 Apktool 反编译命令

反编译完成后可以得到如图 2-2 所示目录文件：

图 2-2 Q 音探歌.apk 反编译文件

其中主要有：

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。

smali：Apktool 将 App 运行在 Dalvik 虚拟机运行的字节码文件输出到 smali 文件夹下，smali_classes2 和 smali 一样都是 Java 源代码的 smali 形式。

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。

2.1.2 系统中的使用

Apktool 工具功能非常强大，除了反编译获取源程序代码和静态资源外，还可以在修改源代码后进行重新构建打包为 apk 文件。本系统中，用到的是 Apktool 的反编译功能。官方文档对于 Apktool 的用法通常是命令行方式，本系统通过将 Apktool.jar 文件集成到项目中，实现了代码方式的调用。

代码方式调用如图 2-3 所示：

```
File inFile = new File("D:\\adobe.apk");
ApkDecoder decoder = new ApkDecoder();
decoder.setOutDir(new File("D:\\apktool"));
decoder.setApkFile(inFile);
decoder.decode();
```

图 2-3 Java 调用 Apktool

2.2 HanLP 分词

2.2.1 工具介绍

HanLP 由自然语义（青岛）科技有限公司开发，主要用于自然语言处理，它的基础类接口功能繁多，包含中文分词、词性标注、命名实体识别、格式转换、信息提取、文本聚类等等。

HanLP 拥有强大的性能优势，采用全球范围内已知最大的亿字级别的中文分词词库，具有很高的准确率。

2.2.2 系统中的使用

本系统使用了 HanLP 的中文分词功能，主要用于求余弦相似度时的文本分词处理。HanLP 分词模块下包含了 CRF 分词、N-最短路径分词、NLP 分词、极速词典分词、标准分词以及深度学习分词。本系统使用的是标准分词，兼顾分词速度和准确率，分词结果包含词性。标准分词秉着“开箱即用”的原则，在代码中调用简单：

```
List<com.hankcs.hanlp.seg.common.Term> termList = HanLP.segment(s);
    return termList.stream().map(term->new
Word(term.word,term.nature.toString())).collect(Collectors.toList());
```

图 2-4 HanLP 使用

2.3 Dom4j .XML 文件解析

2.3.1 工具介绍

Dom4j 是 Dom4j.org 开发的用于解析 XML 文件的工具。它结合了 Java 集合框架的特性，且完全支持 DOM。通过 Dom4j 解析的 XML 文件可以获取 XML 文件中的属性内容以及标签内的文本信息等。

Dom4j 的解析速度非常出色，在多种测试中都表现优异，所以本系统中采用了 Dom4j。

2.3.2 系统中的使用

本系统主要分析 apk 反编译出的文件。由于 Android 工程中包含大量的 XML 文件，其中就有清单文件 AndroidManifest.xml 文件、layout 目录下的布局文件、values 目录下的常量字符串配置文件 string.xml 等等。系统需要进行进一步工作，则需要对 XML 文件进行解析。

Dom4j 在系统中主要负责解析 AndroidManifest.xml 文件，用于获取 AndroidManifest.xml 文件中的 user-permission（用户权限）标签对应的各种信息。user-permission 标签收集到的信息则是目标 apk 文件向用户申请的权限，后期作为主要分析的内容。

2.4 Jsoup 爬虫

2.4.1 工具介绍

Jsoup 是一个用于处理实际 HTML 的 Java 库。它采用 HTML5 最佳的 DOM 方法和 CSS 选择器，为处理数据提供了非常方便的 API。

Jsoup 的数据输入有 4 种方式：从字符串解析文档、解析<body>标签内容、从 URL 加载文档以及从文件加载文档。输入数据后，Jsoup 有 4 种提取有效数据的方式：使用 DOM 方法浏览文档、使用选择器语法查找元素、从元素中提取属性、文本和 HTML 以及使用网址的方式。

2.4.2 系统中的使用

在本系统中，分析 App 的隐私政策明示性，需要获取到目标 App 的隐私政策。绝大多数 App 的隐私政策以 URL 的形式存放于反编译后的程序文件中。获取隐私政策不可避免的需要解析 URL 对应的 HTML 文件内容，本系统正是采用 Jsoup 来进行解析。

本系统使用的是 URL 加载数据的方式，输入为 URL 地址，通过 get 请求方式获取隐私政策的 HTML 文件，之后获取 body 标签的文本即为隐私政策文本：

```
public void JsoupTest(){
    try {
        Document doc = Jsoup.connect("https://static.51xuexiaoyi.com/help/yinsi.html")
                .timeout(5000)
                .ignoreContentType(true)
                .get();
        String content = doc.body().text();
        System.out.println(content);
        System.out.println(doc);
    } catch (IOException e) {
        System.out.println("超时的请求就不进行处理了");
        e.printStackTrace();
    }
}
```

图 2-5 Jsoup 获取隐私政策文本

其中，timeout 为设置请求超时时长为 5 秒，超过 5 秒仍未得到响应，说明该 URL 不能请求，系统采取放弃措施。App 的隐私政策 URL 为文本 HTML，对于 get 请求的请求体没有严格要求，并未对请求体进行身份验证等，可以直接进行访问。Jsoup 也支持一些简单的身份伪装，但是在获取隐私政策时，不需要进行身份伪装，可以直接获取到隐私政策。由此可知，5 秒以上仍未得到响应的 URL 是不能直接访问的，它们需要一定的验证，只有验证通过后才可以获取到内容。

本系统断定，5 秒以上未得到的响应的 URL 不是隐私政策的 URL，于是采取了放弃措施。测试结果也显示本系统的断定是正确的，5 秒时间足以获取到目标 App 的隐私政策文本。

2.5 WebSocket 全双工通信

2.5.1 工具的介绍

WebSocket 是一直全双工通信协议，它建立在一个 TCP 连接上，支持服务端/客户端

双向通信。WebSocket 不仅支持客户端请求服务端，服务端进行响应，同时也支持服务端主动实时的推送消息到客户端。现在，很多地方需要实现推送技术，所用的技术都是通过客户端的 Ajax 轮询技术实现。轮询是指定时的向服务端进行 HTTP 请求，客户端得到服务端的响应进行处理。这种方式非常消耗客户端的性能，部分 HTTP 请求包含较长的头部，其中有效数据只占一小部分，重复的转发这些请求，会造成不必要的浪费！

HTML5 定义的 WebSocket 协议，能更好的节省服务器资源和带宽，并能够更好的进行实时地通讯。

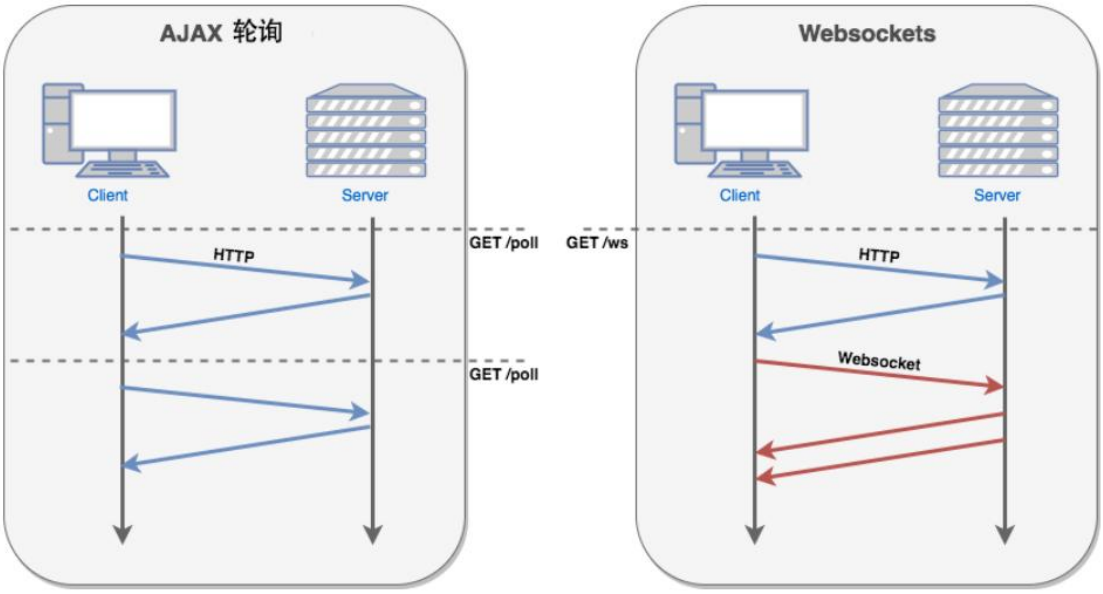

图 2-6 Ajax 轮询方式和 WebSocket 通讯

使用 WebSocket 需要了解相关的事件监听：

表 2-1 WebSocket 事件

事件	事件处理程序	描述
open	Socket. onopen	连接建立成功触发
message	Socket. onmessage	服务端推送数据时触发
error	Sokcet. onerror	通信发生错误时触发
close	Socket. onclose	连接关闭时触发

2.5.2 系统中的使用

WebSocket 主要用于系统的过程日志输出，其中涉及到了分析目标 App 的反编译进度，获取到的目标 App 包含的全部 URLS 等信息。由于轮询的方式消耗带宽资源，本系统采用了 WebSocket，让服务器在分析的时候主动推送目前的分析进度，给予及时的反

馈！

2.6 本章小结

在本章节，详细介绍了系统中使用到的一些第三方工具，包含 App 反编译用到的 Apktool、中文分词 HanLP、Dom4j XML 文件解析包、Jsoup 爬虫、WebSocket 全双工通信协议。除此之外，对这些工具在系统中各自的职责和工具的选择原因进行了一定的阐述。

3 系统设计

3.1 系统整体架构设计

App 隐私政策明示性检测属于 App 合规性检测的一个子模块。App 合规性检测主要包括如下几个方面，其中包括：“隐私政策明示性检测”、“App 收集个人信息行为”检测、“收集个人信息的必要性”检测以及“个人信息泄露”检测。本系统主要目的是检测目标 App 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，进行一系列的筛选从而得到对应的隐私政策。另外，还需要获取目标 App 的 AndroidManifest.xml 文件，其中规定了 App 实现功能需要申请的权限以及 Android 程序的启动类等信息。

本系统主要通过对比分析 AndroidManifest.xml 文件中申请的权限和隐私政策对权限的说明来进行得出结论。

3.1.1 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。根据《App 违法违规收集使用个人信息行为认定方法》，以下行为可被认定为违法收集个人信息：

1. 在 App 中不存在隐私政策，或者隐私政策中没有收集使用个人信息规则；

2. 在 App 第一次安装运行时并未通过弹窗等明显方式提示用户阅读隐私政策等收集规则；

3. 隐私政策等收集使用规则难以访问，比如进入 App 主界面后，需要多于 4 次点击操作才可以访问到；

4. App 隐私政策等收集使用规则难以阅读，比如文字过小、颜色过淡、模糊不清等。

认定的规则不止本文列出的这 4 条，更详细的需阅读《App 违法违规收集使用个人信息行为认定方法》。本系统从隐私政策的角度进行是深入分析，依据第一条规则，本系统先对 App 是否包含隐私政策进行检测，之后分析隐私政策是否对收集个人信息的行为进行了说明解释。另外，本文针对第二条认定规则，调研了目前市面上 App 首次运行进行弹窗的实现技术，并对其进行了说明。

3.1.2 系统体系结构

系统采用的是 B/S(Browser/Client)结构，即浏览器服务器结构。少部分的事务逻辑在前端(Browser)实现，主要的检测分析在服务器端(Server)实现。这种 B/S 体系结构，极大

的简化了客户端的计算机负载，减轻了系统的维护工作等，同时也降低了用户的成本。

如图 3-1 为系统的体系结构图：

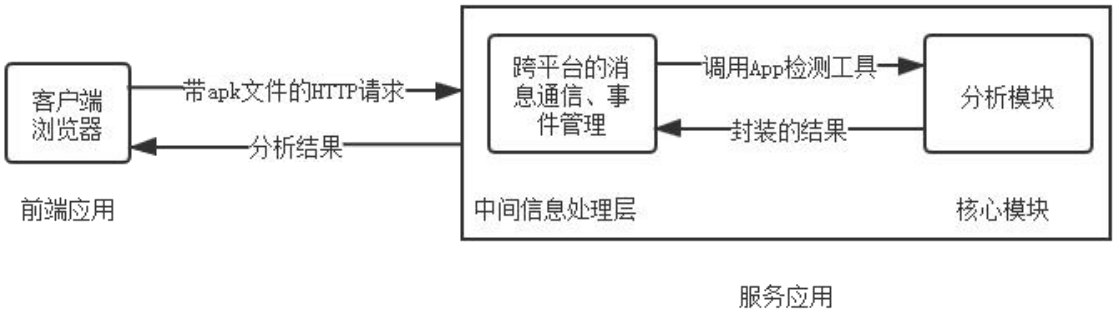

图 3-1 系统体系结构图

3.1.3 系统运行环境

1. 操作系统：Windows、Linux
2. 开发平台和工具：Intellij IDEA
3. 通信协议：HTTP、WebSocket
4. JDK：1.8 版本以上

3.2 系统检测方案

系统检测的侧重点是 App 中的隐私政策是否对 App 申请的权限进行了相应的说明。隐私政策中有针对 App 申请的权限进行的说明，比如目标 App 申请了麦克风权限，隐私政策中会告知用户，申请麦克风权限的用途和原因。如果隐私政策并未对这种情况进行说明，则是违规收集个人信息，违反了第一条认定规则。

该检测方式面临一个难点是如何将隐私政策和 AndroidManifest.xml 中申请的权限集关联起来。

经过调研发现，每个 App 申请的权限在 20 个左右，其中涉及到可能会收集个人信息的权限则更少。然而这只是特定的 App，如今，市面上存在不同种类的 App 数不胜数。最常见的包含吃、穿、住、行各个方面，其中最熟悉的有：美团外卖、滴滴出行、携程旅行、小红书、淘宝等等。不同种类的 App 申请的权限各不相同，美团外卖申请的位置权限、文件管理申请的存储权限、聊天软件申请的通话权限等等。各类 App 侧重的权限不同，导致了从隐私政策入手难上加难。

最终，系统通过构建权限语料库的方式将隐私政策和 AndroidManifest.xml 文件关联

起来。

3.2.1 语料库构建

目前 Android 拥有 100 多种权限，权限中有涉及隐私的权限，也有普通的权限。常见的涉及隐私的权限有：位置权限、存储权限、日志权限、读写权限等。不会涉及用户隐私的权限也有许多，如 WIFI 权限、网络权限等。

系统将涉及到隐私的权限整合在一起构成了语料库，权限语料库也将隐私政策和 AndroidManifest.xml 文件关联了起来。App 中的 AndroidManifest.xml 文件中申请的一个权限对应权限语料库里的一个记录。在系统检测目标 App 时，记录至关重要。

权限语料库的构建只能通过人工构建，通过了解 Android 的所有权限，筛选出其中涉及用户隐私的相关权限，之后将筛选出的权限进行提取关键词。比如，短信权限对应的关键词有短信、短信息、SMS 等。关键词即代表了特定的权限，有了关键词，判断隐私政策中对权限是否有解释说明就比较容易了，这是权限语料库的重要作用。

权限语料库在系统中以 XML 文件形式存在，以 XML 文件的形式存在不仅方便配置，还方便阅读和修改。权限语料库的关键词提取越好，系统的检测功能就越优秀。权限语料库的格式，如图 3-2 所示：

```
<?xml version="1.0" encoding="UTF-8"?>
<manifest>
  <permission name="android.permission.READ_CALL_LOG" keyword="通话 通话记录">允许应用读取用户的通话记录(危险)</permission>
  <!--第 11 个权限-->
  <permission name="android.permission.READ_CONTACTS" keyword="联系人 通讯录">允许应用读取用户的联系人数据(危险)</permission>
  <!--第 12 个权限-->
  <permission name="android.permission.READ_EXTERNAL_STORAGE" keyword="存储 外部存储">允许应用程序从外部存储读取(危险)</permission>
</manifest>
```

图 3-2 权限语料库格式

1. <manifest>：根标签。

2. <permission>：一级标签，每个权限对应一个 permission 标签，其中 name 属性是权限的名称，keyword 是针对权限提取的关键词组，标签体内容代表的是该权限的解释。

权限语料库作为配置存在在系统中，有可插拔的优点。随着权限语料库的更新升级，系统的检测能力会越来越优秀，这构成了系统的优秀扩展性。

3.2.2 App 检测流程

系统的检测有 3 大步骤，上传 apk 文件->检测 apk 文件->得出检测结果。细分则有 6 个模块，分别包含文件上传、反编译、隐私政策获取、App 清单文件解析、App 隐私政策明示性分析以及日志模块。其中每个模块负责一个子任务，除去日志模块，其他 5 个模块顺序执行完成系统的检测功能。

App 检测流程整体归纳为图 3-3 所示：

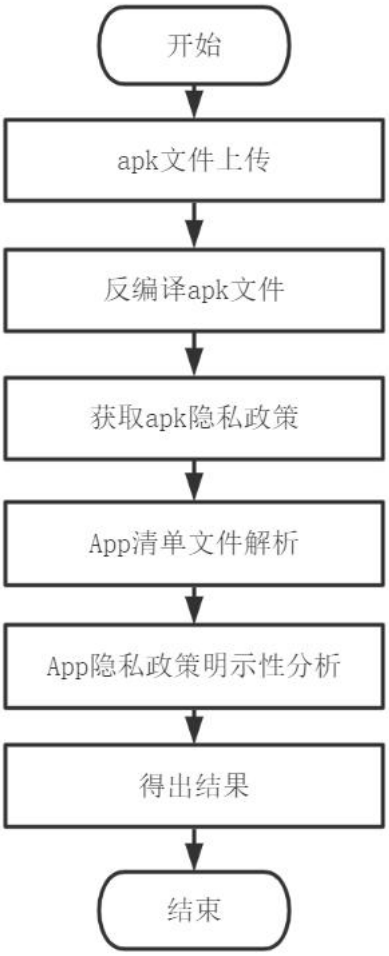

图 3-3 系统整体流程

3.3 模块设计

3.3.1 文件处理

【功能描述】

完成 apk 文件的上传功能

【输入】

用户在客户端上传 apk 或拖拽上传 apk 文件

【处理流程及说明】

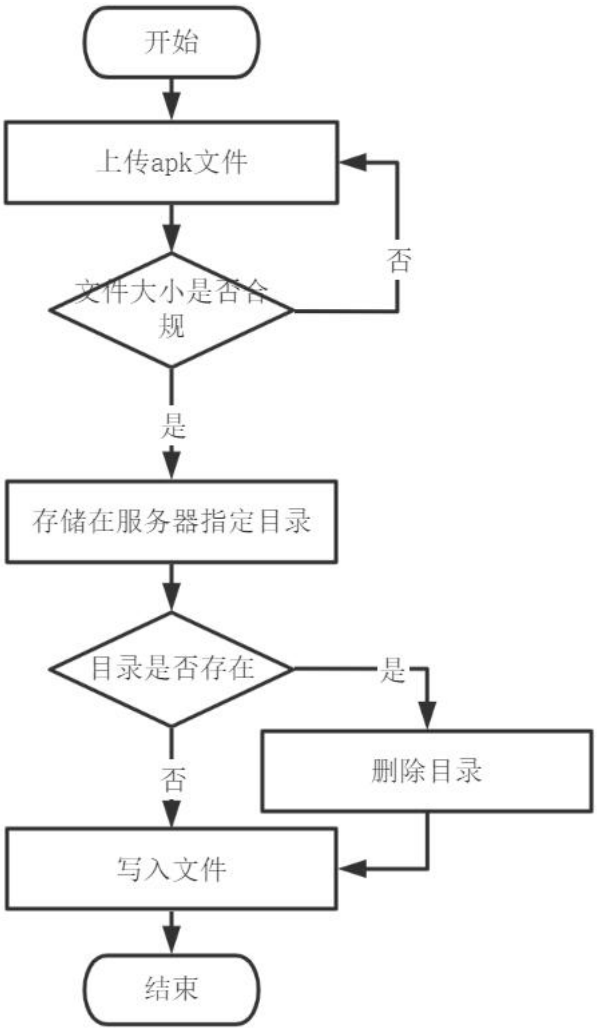

图 3-4 文件上传流程

说明：文件类型指定 apk 文件，大小限制 100MB 以内，实时上传进度条

【输出】

在服务端指定的路径存储上传的 apk 文件

3.3.2 反编译

【功能描述】

将 apk 文件做逆向工程

【输入】

输入为服务器端存储的 apk 文件，即文件处理的输出文件

【处理流程及说明】

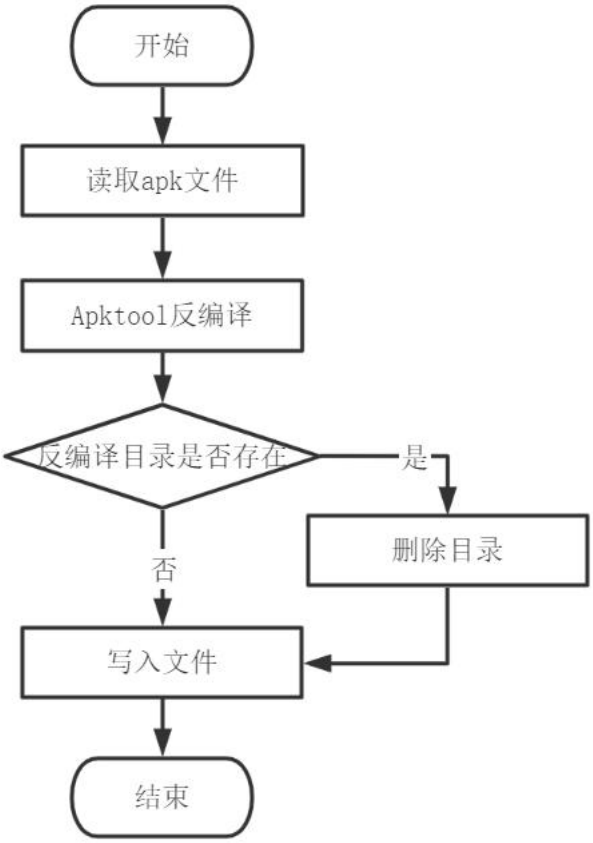

图 3-5 反编译流程

说明：输出反编译产物时，指定目录有文件需先删除目录再写入

【输出】

输出反编译后的文件并写入服务器指定路径

3.3.3 隐私政策获取

【功能描述】

获取目标 App 的隐私政策文本和 URL 地址

【输入】

反编译得到的目录文件夹

【处理流程及说明】

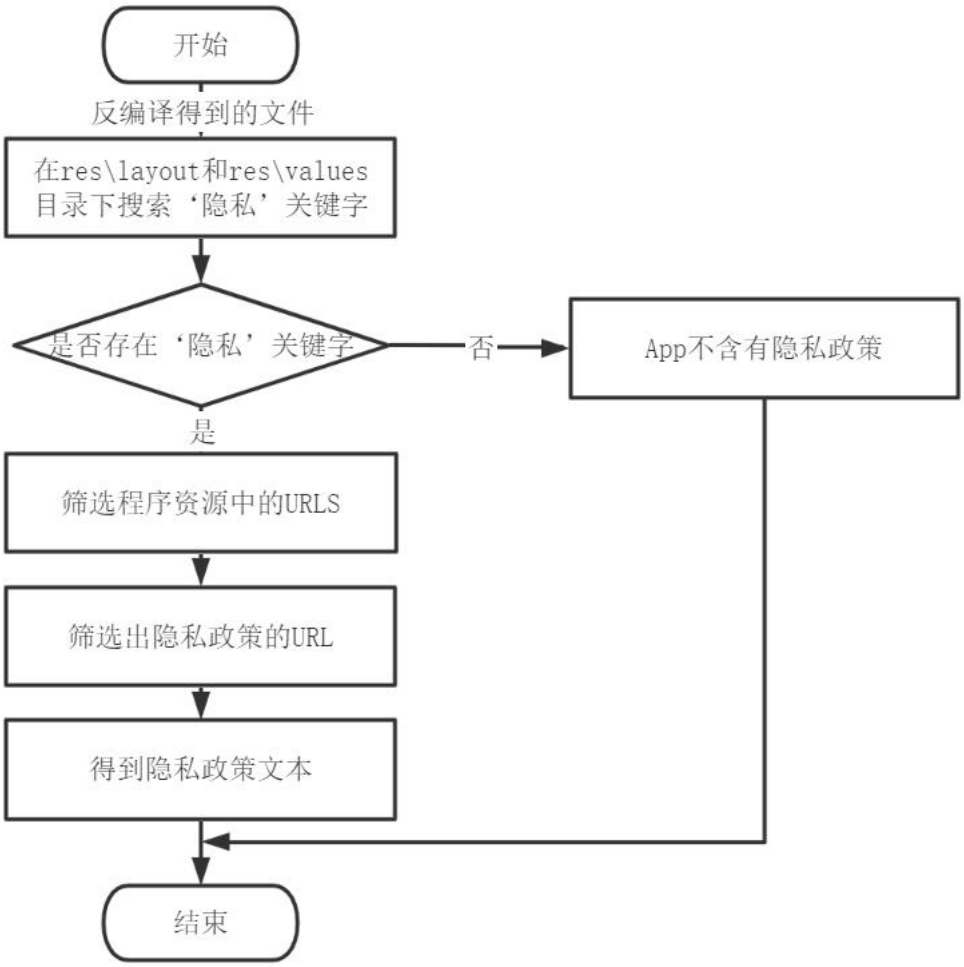

图 3-6 获取隐私政策流程

说明： App 不含有隐私政策视为未明示，需提醒用户

【输出】

目标 App 的隐私政策文本和对应的 URL

3.3.4 App 清单文件解析

【功能描述】

解析 AndroidManifest.xml 文件获取 App 申请的权限

【输入】

AndroidManifest.xml 文件

【处理流程及说明】

流程：从反编译产生的文件根目录找到 AndroidManifest.xml 文件，利用 Dom4j 解析该文件得到<uses-permission>标签中的 name 属性的值，即获取到 App 申请的所有权限

说明：注意 SDK 23 版本以上申请权限的新标签<uses-permission-sdk-23>

【输出】

App 申请的权限集合

3.3.5 App 隐私政策明示性分析

【功能描述】

得出目标 App 隐私政策对收集用户信息是否进行了明示

【输入】

目标的 App 隐私政策、App 申请的权限集合、权限语料库

【处理流程及说明】

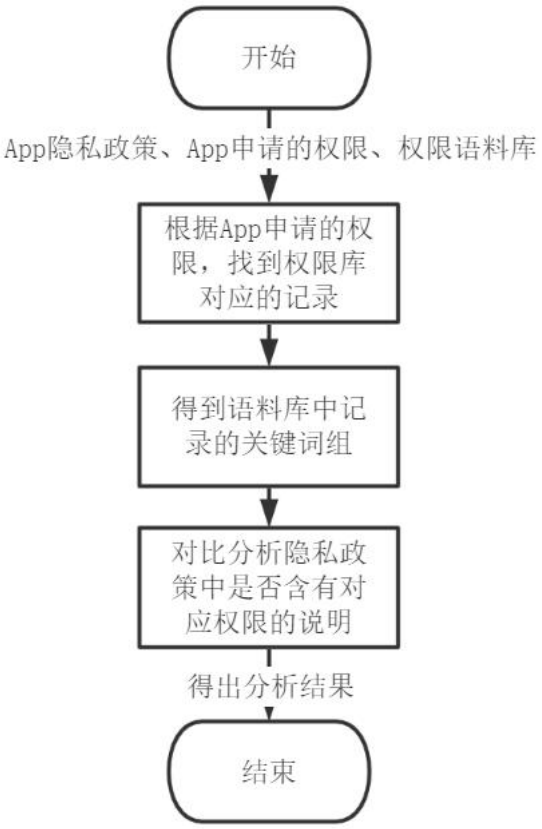

图 3-7 隐私政策明示性分析流程

说明：权限语料库中的权限对应的关键词组越精确，检测结果越好

【输出】

目标 App 未进行说明的权限以及对权限的解释

3.3.6 日志模块

【功能描述】

实时输出检测进度到前端、记录分析过程中产生的信息：App 中的 URLS、App 申请的所有权限、权限语料库

【输入】

无输入

【处理流程及说明】

流程：用户访问网站开始记录直到关闭网站

说明：记录实时输出

【输出】

前端的日志、各种过程文档

3.4 本章小结

在本章节中，对系统的整体架构设计进行了说明。解释了什么是 App 隐私政策明示性。接下来给出了系统的检测流程，了解了整个系统流程后，对文件处理、反编译、隐私政策获取、App 清单文件解析、App 隐私政策明示性分析、日志模块 6 个模块进行了设计，规定了各个模块的输入要求和输出结果以及各自的详细流程说明。

4 系统实现

4.1 文件处理

4.1.1 实现细节

文件处理即文件上传，从客户端输出文件到服务器。采用 commons-fileupload 实现，客户端采用 Layui 的 form 模块实现文件上传功能。系统不仅实现了基本的文件上传功能，而且对上传的文件进行了各方面的约束以及上传进度的实时展示。

文件大小方面，设置了由客户端上传的文件大小不能超过 100MB。文件类型方面，设置了由客户端上传的文件类型为 apk 类型，即只有 apk 类型的文件能进行上传。文件大小配置可以在项目中的配置文件中进行配置，文件类型是通过 Layui 进行的约束，Layui 对上传文件类型配置容易，易于实现。

考虑到 apk 文件普遍较大和网络传输速度较慢，实现上传进度展示显得非常必要。Layui 对进度条展示也进行了支持，采用回调函数实时的进行上传百分比的渲染。

4.1.2 核心代码

文件上传实现主要依赖于前端 Layui 中 form 和后端 commons-fileupload 配合实现：

```
upload.render({
    elem : "#upload",
    accept : "file",
    exts: 'apk',
    size: 100 * 1024 * 1024,
    url : "/apk/upload",
    progress:function(value){<!--上传进度回调 value 进度值-->
        element.progress('uploadProgressBar', value+'%')
    },
    before : function (obj) {...},
    done : function (res, index, upload) {...},
    error : function (res) {...}
});
```

图 4-1 前端文件上传

其中，exts 设置上传文件的后缀为 apk，size 设置上传文件最大为 100MB，url 是访问的接口，progress 是回调函数，value 是目前上传的进度百分比，done 方法是文件上传成功后执行，error 则是文件上传失败执行。

```
public Map<String, Object> upload(MultipartFile file, HttpServletRequest request){
        Map<String, Object> result = new HashMap<>();
        if (file != null && !file.isEmpty()){
            String directory = request.getRemoteAddr().replaceAll(":","-");
            String apkInput = res.apkInput + directory;
            File dir = new File(apkInput);
            if (!dir.exists()){
                dir.mkdirs();
            }
            String apkName = file.getOriginalFilename();
            File desFile = new File(apkInput +"\\"+ apkName);
            if (desFile.exists()){
                //目标文件存在，重命名
                apkName = UUID.randomUUID().toString() + apkName;
                desFile = new File(apkInput +"\\"+ apkName);
            }
            try {...} catch (Exception e) {...}
        } else {...}
        return result;
    }
```

图 4-2 后端文件上传

考虑到不同用户同时检测 App，后端对于每个访问的 IP 创建了唯一的目录文件夹进行检测。其中如果某个用户上传命名重复的 apk 文件，后端对其进行了重命名处理。

4.1.3 效果图

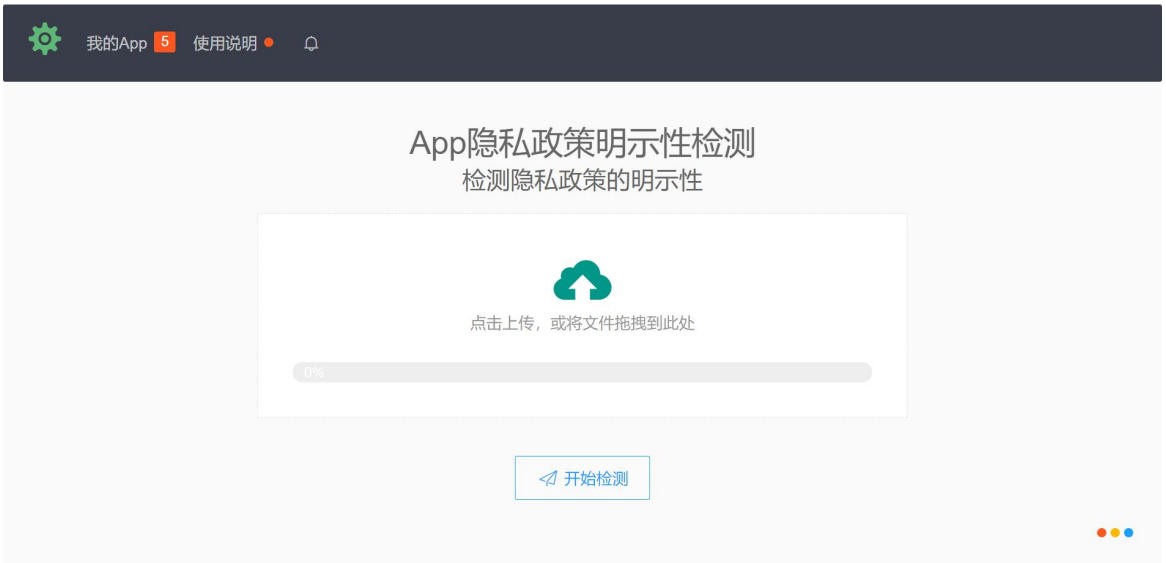

图 4-3 文件上传界面

用户可以选择从文件管理器中选中文件或将文件拖入上传框两种方式进行文件上传。‘开始检测’按钮开始为禁止点击状态，上传完成后即可进行检测。

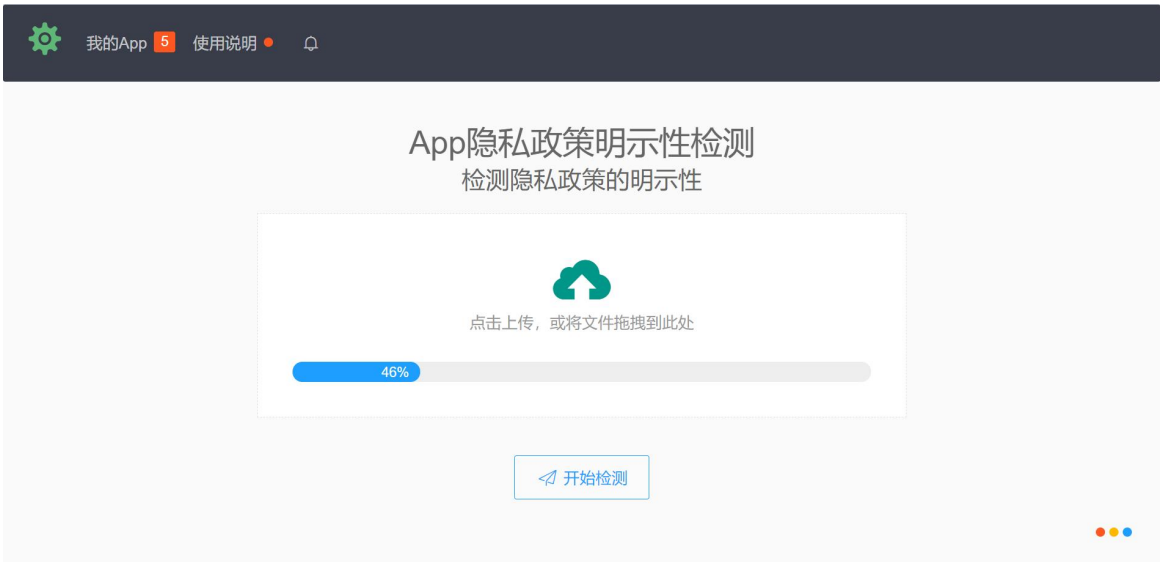

图 4-4 上传进度条

如图 4-4 所示为文件上传实时进度的展示。

4.2 反编译

4.2.1 实现细节

对 apk 的反编译称为 Android 逆向工程，Android 逆向工程是将完整的 apk 尽量的还原到构建之前的源码形式。然而，目前绝大多数 App 在构建 apk 的时候会对代码进行混淆和加密处理，导致反编译变的异常困难。混淆代码是指将代码中的类名，变量名，方法名进行混淆，如使用 a，b，c 代替，但是这并不影响代码的编译运行。混淆代码的目的是为了让反编译后的代码可阅读性变差。对代码的加密处理则是为了让构建完成的 apk 更难被反编译成功。反编译可以获得对应 apk 的 smali 代码，加密则让 smali 代码的获取难上加难。

系统中对于 apk 的反编译采用 Apktool 工具实现，Apktool 通常使用命令的方式进行反编译处理，经过大量的调研，通过 Java 代码使用 Apktool.jar 包内部的类实现了与命令相同的反编译功能。

4.2.2 核心代码

Apktool.jar 包反编译时使用的也是 ApkDecoder 对象，如图 4-6 反编译日志即可得知。

反编译是系统检测隐私政策明示性的基础，如图 4-5 所示为反编译的核心实现：

```
public static void decompile(String input,String output){
    ApkDecoder apkDecoder = new ApkDecoder();
    File apk = new File(input);
    if(apk.exists() && apk.isFile()){
        apkDecoder.setApkFile(apk);
        File out = new File(output);
        boolean flag = IOUtil.deleteFile(out);
        if(flag){
            try {
                apkDecoder.setOutDir(new File(output));
                apkDecoder.decode();
            } catch (Exception e) {
            }finally {
                try {
                    apkDecoder.close();
                } catch (IOException e) {
                    e.printStackTrace();
                }
            }
        }
    }
}
```

图 4-5 反编译核心代码

在反编译时，会先判断路径是否为文件路径，如果为文件夹路径则不进行反编译。设置反编译的输出目录时，如果目标文件夹存在，则会对其进行删除处理。反编译的最后会关闭 ApkDecoder，避免造成资源浪费。

4.2.3 效果图

```
Using Apktool 2.5.0 on 云达人.apk
Loading resource table...
Decoding AndroidManifest.xml with resources...
Loading resource table from file: C:\Users\可乐yue\AppData\Local\apktool\framework\1.apk
Regular manifest package...
Decoding file-resources...
Decoding values */* XMLs...
Baksmaling classes.dex...
Baksmaling classes2.dex...
Baksmaling assets/A3AEECD8.dex...
```

图 4-6 反编译日志

反编译成功后，会产生如图 2-2 的反编译文件，系统之后的检测就是针对此目录中文件进行分析。apk 文件在反编译后，占用的空间会非常大，基本都在 100MB 以上，考

虑到服务器的存储压力，在系统检测完成后，会对 apk 文件和 apk 反编译生成的文件进行删除操作，避免服务器堆积大量无用文件。

反编译并不是每次都是成功的，Apktool 也会有反编译失败的情况发生，系统对此采取忽略的方法。反编译的失败分为很多种情况，其中最常见的.dex 文件加密导致反编译失败。.dex 文件是 apk 源代码打包后形成的一种文件，系统在检测过程中是对 smali 代码进行分析，对于.dex 文件没有硬性要求。smali 代码是源代码的另一种形式，因此分析 smali 代码和.dex 代码作用大致相同。

忽略反编译失败必然出现很多情况，面对上述情况外的失败，比如反编译后并未产生相应的目录文件，这种情况，后续在检测中会抓住异常情况告知用户检测失败。

4.3 隐私政策获取

4.3.1 实现细节

经过大量调研发现，目前 App 的隐私政策是通过 URL 的方式进行访问，在 App 安装后，弹出的隐私政策同意窗口，其中隐私政策以链接的形式存在窗口中。另外，在 App 的“关于”页面也可以再次找到隐私政策。经调研发现，在没有网络的情况下，隐私政策处于不能访问的状态。

App 中必须包括隐私政策声明，否则表示该 App 未进行隐私政策明示。因此获取 App 中的隐私政策需要分为以下几步：

1. 判断 App 的界面文件中是否存在‘隐私’关键词，存在则说明 App 含有隐私政策，不存在说明 App 并不存在隐私政策。

2. 确认 App 含有隐私政策后，需要获取 App 中存在的所有 URL，从中筛选出属于隐私政策的 URL。

3. 利用爬虫爬取所有 URL 对应的文本内容，通过求余弦相似度，筛选出 App 的隐私政策。

其中，第一步判断隐私政策是否存在通过 Java IO 读取 App 的布局文件进行字符串搜索实现。第二步筛选出 App 的所有 URL 通过 Java 操作命令行实现，在不同的环境下，命令行也不同。比如，在 Windows 中使用 findstr 命令实现 URL 筛选，在 Linux 中使用 grep 命令实现。第三步将得到的 URL 使用 Jsoup 爬取文本，之后利用余弦相似度求出与模板隐私政策最相似的文本，该文本即为目标 App 的隐私政策。

第三步中求出的余弦相似度值越接近 1 则说明与模板隐私政策越相似，但是存在一个问题：缺少一个阈值来区分隐私政策和非隐私政策文本，即小于阈值的为非隐私政策，

反之为隐私政策。

如何得出余弦相似度的阈值？本系统通过求取 Gini 系数得到，Gini 系数是决策树用于选择属性节点时用的方法。在机器学习中，决策树的属性节点区分度越大，选择出的属性节点就越优。区分度大的属性节点意味着对数据集的区分更准确。Gini 系数是选择属性节点的一种准则，属性节点对应的 Gini 系数越小代表该属性节点区分度越大。借鉴 Gini 系数的这种特性，将余弦相似度的阈值从 0.10 开始递增到 0.90，递增间隔为 0.01，求出每个阈值对应的 Gini 系数，得出最小的 Gini 系数对应的余弦相似度的阈值，该阈值代表着最优的区分度，也是区分隐私政策和非隐私政策最优的阈值。

Gini 系数的计算公式如 4-1 所示：

$$G(p)=\sum_{k=1}^{K} p_k(1-p_k)=1-\sum_{k=1}^{K} p_k^2 \tag{4-1}$$

其中 p_k 代表的是某个结果对应的概率。

比如有数据集隐私政策 10 个，非隐私政策 10 个，在某个阈值的区分下，判断为隐私政策的文本为 9 个，非隐私政策的文本为 11 个。其中 9 个隐私政策原本为非隐私政策的文本 2 个，11 个非隐私政策原本为隐私政策的有 4 个。

由于结果为二分类问题，所以需要先求出结果为隐私政策的 Gini 系数 gini1 和结果为非隐私政策的 Gini 系数 gini2，再对二者进行加权平均得出该阈值下的 Gini 系数。

gini1 = 1 - $((7/9)^2+(2/9)^2)$=0.36

gini2 = 1 - $((4/11)^2+(7/11)^2)$=0.47

Gini = gini1 * (9/20) + gini2 * (11/20)=0.4205

上述数值为近似值，求得的 Gini 系数为 0.4205，由此可知该阈值的区分效果不是很好，11 个非隐私政策中原先有 4 个隐私政策，可见区分度很小，误差很大。

4.3.2 核心代码

1. 获取 URLS

```
Process process = null;
int exitValue = 0;
try {
    process = Runtime.getRuntime().exec(command,null,new File(dir));
    exitValue = process.waitFor();
} catch (IOException | InterruptedException e) {
    e.printStackTrace();
    log.info("process 执行失败");
}
if(exitValue != 0){
    log.info("command 命令执行失败");
}
```

图 4-7 Java 执行操作系统命令

利用操作系统的命令筛选出 App 的 URL 需要使用 Java 执行命令，其中 command 表示执行的命令，dir 为命令的执行工作目录。

2. 求余弦相似度

```
try {
        String content = Jsoup.connect(url).timeout(5000).ignoreContentType(true).get().body().text();
        if(content != null && content.length() > 0){
            double similarity = CosineSimilarity.getSimilarity(content, privacyTemplate);
            synchronized (GetPrivacy.class){
                if(similar < similarity){
                    similar = similarity;
                    privacy = content;
                    privacyUrl = url;
                }}}
} catch (IOException e) {}finally {cdl.countDown();}}
```

图 4-8 获取相似度最高的隐私政策文本

将 Jsoup 爬取的每个 URL 的文本与模板隐私政策求余弦相似度，得到余弦相似度最高的 URL 对应的文本。

3. 获取隐私政策

```
if(similar < 0.6){
    return null;
}else{
    return privacy;
}
```

图 4-9 判断隐私政策

最高余弦相似度的文本并非一定为隐私政策文本，通过求得余弦相似度的阈值对其进行决策，判断该文本是否为隐私政策。

4. 求 Gini 系数

```
double p1 = (privacy_num[0] * 1.0 / privacy_num[1]);
double p2 = ((privacy_num[1] - privacy_num[0]) * 1.0 / privacy_num[1]);
gini1 = 1.0 - (p1 * p1 + p2 * p2);
p1 = (non_privacy_num[0] * 1.0 / non_privacy_num[1]);
p2 = ((non_privacy_num[1] - non_privacy_num[0]) * 1.0 / non_privacy_num[1]);
gini2 = 1.0 - (p1 * p1 + p2 * p2);
gini = (gini1 * privacy_num[1] / (privacy.size() + non_privacy.size())) +
        (gini2 * non_privacy_num[1] / (privacy.size() + non_privacy.size()));
```

图 4-10 Gini 系数

4.3.3 效果图

```
Gini: -0.1,余弦相似度: 0.99
Gini: 0.061,余弦相似度: 0.83
Gini: 0.063,余弦相似度: 0.69
Gini: 0.114,余弦相似度: 0.87
Gini: 0.120,余弦相似度: 0.68
Gini: 0.124,余弦相似度: 0.74
Gini: 0.160,余弦相似度: 0.91
Gini: 0.171,余弦相似度: 0.62
Gini: 0.218,余弦相似度: 0.59
Gini: 0.261,余弦相似度: 0.55
Gini: 0.295,余弦相似度: 0.92
Gini: 0.300,余弦相似度: 0.53
Gini: 0.320,余弦相似度: 0.93
Gini: 0.336,余弦相似度: 0.35
Gini: 0.364,余弦相似度: 0.94
```

图 4-11 求余弦相似度阈值

Gini 系数为-0.1 表示决策结果只有一种情况，但是数据集中包含隐私政策和非隐私政策两种数据集，出现这种结果表示极端的决策，当余弦相似度为 0.99 时，绝大多数文本都会判断为非隐私政策，相似度为 0.99 的文本基本可以认为是同一种文本。

当 Gini 系数为 0.061 时，余弦相似度的区分度是最大的，但是实际测试过程中，发现同为隐私政策的文本也存在相似度较低的情况，因此本系统选择的是 0.063 对应的余弦相似度 0.69，并对其进行了降低，最终确认为 0.6。

4.4 App 清单文件解析

4.4.1 实现细节

App 清单文件指 Android 工程中的 AndroidManifest.xml，AndroidManifest.xml 文件中定义了 App 申请的所有用户权限。清单文件规定，如要访问敏感用户数据（如联系人和短信）或某些系统功能（如相机和互联网访问），则 Android 应用必须请求相关权限。每个权限均由唯一标签标识。例如，如果应用需要发送短信，则必须在清单中添加以下代码行：

```
<manifest>
<uses-permission android:name="android.permission.SEND_SMS"/>
...
</manifest>
```

图 4-12 清单文件权限申请

系统获取清单文件中申请的权限，不可避免需要对清单文件进行解析处理，本系统使用了 Dom4j 来进行解析。另外，考虑到 SDK 23 版本以上申请权限的新标签<uses-permission-sdk-23>，在解析时将该标签也包含进去进行解析处理。

4.4.2 核心代码

```
ArrayList<String> permissions = new ArrayList<>();
Element root = getRootElement(path);
Element foo;
assert root != null;
for (Iterator i = root.elementIterator("uses-permission"); i.hasNext(); ) {
        foo = (Element) i.next();
        Attribute attribute = foo.attribute("name");
        permissions.add(attribute.getValue());
}
for (Iterator i = root.elementIterator("uses-permission-sdk-23"); i.hasNext(); ) {...}
return permissions;
```

图 4-13 解析清单文件

图 4-13 所示，path 代指清单文件所在的目录，根据清单文件的位置，先获取根节点，即<manifest>。

获取到根节点后，对子节点进行遍历，获取<uses-permission>以及<uses-permission-sdk-23>节点，进一步获取二者的 name 属性的值，即 App 申请的权限清单。

4.4.3 效果图

图 4-14 所示为解析的 Q 音探歌.apk 申请的权限：

图 4-14 Q 音探歌.apk 申请的权限

4.5 App 隐私政策明示性分析

4.5.1 实现细节

App 隐私政策明示性分析目的是为了检测出 App 的隐私政策中未进行说明的权限。通过前面的介绍，获取到的隐私政策和目标 App 申请的权限通过构建的语料库关联起来。根据 App 申请的权限集合去查取语料库中对应的权限，进一步得到对应的搜索关键词组，之后使用关键词组去搜索隐私政策是否存在对这些权限的说明，存在即说明隐私政策对权限进行了解释说明，反之即没有对此权限进行说明。

未进行说明的权限视为未进行隐私政策明示性，对此会对用户进行提示说明。考虑到用户对权限的不了解，系统对权限进行了中文的解释说明。

由于本系统从权限和隐私政策的角度出发进行的检测，涉及的检测范围比较小，并未考虑完善，因此对于进行了明示的 App 会弹出‘暂未检测到隐私政策未明示的情况’。

4.5.2 核心代码

系统利用字符串搜索的方式，在隐私政策中查询是否存在指定权限的关键词组，这些关键词组代表的即是某个权限，如图 4-15 所示：

```
//存放隐私政策中未说明的权限对象，即 res 中存在值说明 app 中有隐私政策未明示的情况
ArrayList<Permission> res = new ArrayList<>();
for(String s : permissions){
    Permission permission = manifest.get(s);
    if(permission != null){
        String[] keywords = permission.getKeyword().split(" ");
        int len = keywords.length;
        for (int i = 0; i < len; i++) {
            if(privacy.contains(keywords[i])){
                break;
            }
            if(i == len-1 && !privacy.contains(keywords[i])){
                res.add(new Permission(s,permission.getKeyword(),permission.getDescription()));
            }
        }
    }
}
```

图 4-15 App 隐私政策明示性分析

其中，res 集合存放未进行说明的权限，keywords 数组存放权限对于的关键词组。当最后一个关键词仍未从隐私政策中找到，说明该权限未在隐私政策中进行对应的说明。

4.5.3 效果图

(1)图 4-16 为系统检测出 App 中包含未说明的权限，图 4-17 为对权限的解释：

图 4-16 App 未说明权限

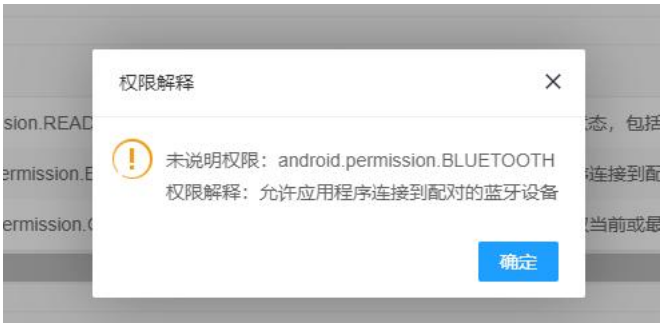

图 4-17 权限解释

(2)如图 4-18 所示，系统并未检测出 App 存在隐私政策未明示的情况：

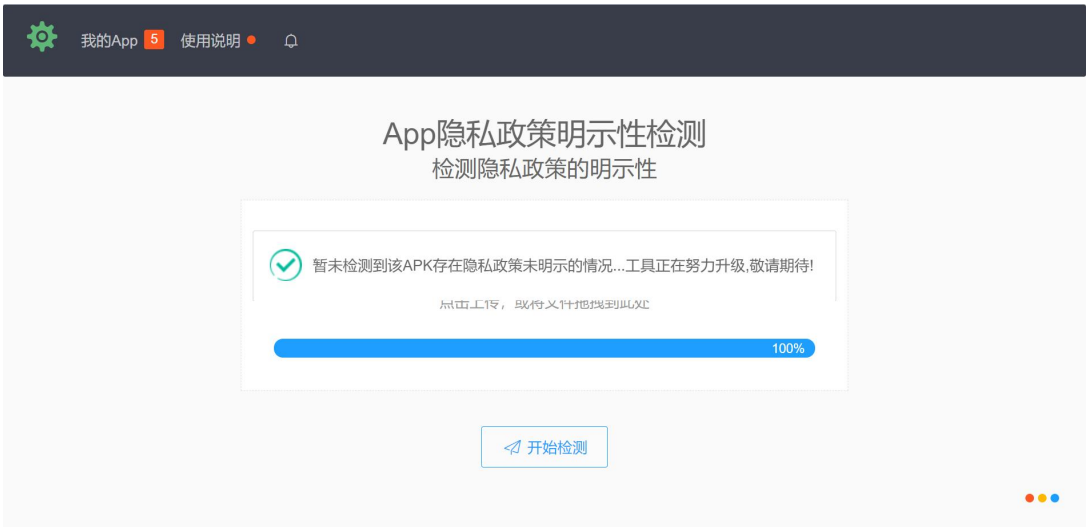

图 4-18 检测通过

4.6 日志模块

4.6.1 实现细节

日志模块是为了更清晰的展示检测进度，对于漫长的检测过程，进度尤为重要。考虑到带宽的消耗，本系统使用 WebSocket 实现了实时推送日志的功能。

WebSocket 全双工通信避免了浏览器频繁的轮询请求，提高了效率。由于 WebSockset 的优点，服务器可以随时推送消息到浏览器，这种工作模式也利于日志模块的实现。本系统正是采用这种方式实现了检测进度的实时展示。

4.6.2 核心代码

(1)图 4-19，其中 onopen 为建立连接成功后调用，onerror 为发生异常调用，onmessage 为接收后端实时推送的消息的方法，onclose 默认为关闭窗口调用起到关闭连接的作用：

```
var websocket = null;
var log = this.logs;
//判断当前浏览器是否支持 WebSocket, 主要此处要更换为自己的地址
if ('WebSocket' in window) {
    // websocket = new WebSocket("ws://47.95.33.183:8080/analysis/log");
    websocket = new WebSocket("ws://127.0.0.1:8080/analysis/log");
} else {
    alert('不支持 webSocket')
}
//连接发生错误的回调方法
websocket.onerror = function() {...};
//连接成功建立的回调方法
websocket.onopen = function(event) {...};
//接收到消息的回调方法
websocket.onmessage = function(event) {...};
//连接关闭的回调方法
websocket.onclose = function() {...};
```

图 4-19 WebSocket 前端配置

(2)如图 4-20，为 WebSocket 在后端推送日志的方法：

```
Session session = websocketMap.get(uid);
if(session == null){...}else{
    session.getBasicRemote().sendText(message);
}
```

图 4-20 WebSocket 推送日志

4.6.3 效果图

如图 4-21 所示，为系统在检测目标 App 时展示的进度日志，从反编译开始直到分析结束：

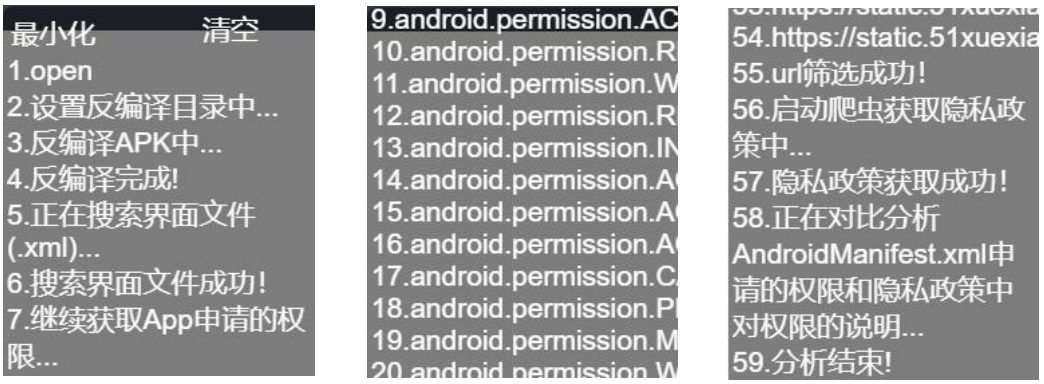

图 4-21 日志

4.7 本章小结

在本章节中，针对系统设计进行了系统实现相关的阐述。对模块设计中的文件处理、反编译、隐私政策获取、App 清单文件解析、App 隐私政策明示性分析、日志模块进行了详细讲解，包括模块的实现细节和核心代码。另外还展示了相关模块的效果图。

5 系统测试

系统测试是系统开发完成的最后一步，也是最为关键的一步。系统测试从不同的方面对软件进行检测以验证是否符合用户预期。软件测试的目的是为了发现程序中的错误而执行程序的过程，它能够帮助识别软件的正确度、完全度、质量，从而帮助系统更好的开发[19]。本章节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

测试集下载自网络和华为应用市场，主要用于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
Test1(Q 音探歌.apk)	是	<50MB(17.8MB)
Test2(学小易.apk)	是	<50MB(20.7MB)
Test3(云达人.apk)	是	<50MB(38.0MB)
Test4(Cirta.apk)	否	<50MB(15.4MB)
Test5(Clash.apk)	否	<50MB(19.1MB)
Test6(小鸡模拟器.apk)	否	≥50MB,<100MB(63.1MB)
Test7(粉笔教育.apk)	是	≥50MB,<100MB(53.0MB)
Test8(夸克.apk)	是	≥50MB,<100MB(78.2MB)
Test9(美团外卖.apk)	是	≥50MB,<100MB(64.86MB)
Test10(高德地图.apk)	是	≥50MB,<100MB(88.0MB)

其中，包含三个 App 不存在隐私政策，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

系统测试环境包含多个方面，有测试系统兼容性时采用的移动设备端和 PC 端、也有系统测试使用的第三方工具版本等，详细环境配置如表 5-2 所示：

表 5-2 测试环境

操作系统	Windows 10 家庭中文版
Apktool-反编译工具	Apktool 2.5.0 版本
Dom4j-XML 解析	Dom4j 1.6.1
Jsoup-爬虫	Jsoup 1.10.2
HanLP-分词	portable-1.8.1
PC 端浏览器	Google Chrome 版本 90.0.4430.212、 Microsoft Edge 版本 91.0.864.37
移动设备	Honor Play4T
移动端浏览器	华为浏览器 版本 11.1.1.300 Quark 5.0.6.179

5.3 系统功能性测试

本小节主要描述了本系统关键模块的功能性测试，系统功能性测试的目的是为了检验各个功能模块功能是否符合预期需求标准。

5.3.1 文件处理的测试

文件处理模块的测试用例主要包括文件上传是否成功，文件类型和大小约束是否成功，进度条是否能够实时的展示，文件处理模块测试表如表 5-3 所示：

表 5-3 文件处理模块测试表

功能模块	文件处理模块				
测试目的	测试文件是否能够正常上传，进度条能否实时展示				
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	用户点击或拖拽 apk 文件之外的文件	非 apk 文件	提示不能上传	提示不能上传	结果符合预期
2	用户点击或拖拽 apk 文件（文件选择大于 100MB）	apk 文件	提示超出文件大小上限	提示超出文件大小上限	结果符合预期

续表 5-3

3	用户点击或拖拽 apk 文件	apk 文件	进度条百分比增加，达到 100%，文件上传完毕	进度条百分比增加，达到 100%，文件上传完毕	结果符合预期

5.3.2 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 5-4 所示：

表 5-4 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 的隐私政策 URL	检测结果中包含目标 App 的隐私政策 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

5.3.3 日志模块的测试

日志模块的测试用例主要包括系统检测进度的实时展示，反编译日志展示、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 5-5 所示：

表 5-5 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、获取到的 URL 展示、隐私政策获取成功展示	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

5.3.4 隐私政策明示性分析测试

隐私政策明示性分析测试的主要目的是：测试系统能否准确检测出不含隐私政策的 App 和 App 未说明的权限，测试表如表 5-6 所示：

表 5-6 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 App 未说明的权限				
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	Q 音探歌.apk	未说明权限	未说明权限	符合预期
Test2	上传并检测	学小易.apk	未说明权限	未说明权限	符合预期
Test3	上传并检测	云达人.apk	未说明权限	未说明权限	符合预期
Test4	上传并检测	Cirta.apk	不含隐私政策	不含隐私政策	符合预期
Test5	上传并检测	Clash.apk	不含隐私政策	不含隐私政策	符合预期
Test6	上传并检测	小鸡模拟器.apk	不含隐私政策	不含隐私政策	符合预期
Test7	上传并检测	粉笔教育.apk	未说明权限	未说明权限	符合预期
Test8	上传并检测	夸克.apk	未说明权限	未说明权限	符合预期
Test9	上传并检测	美团外卖.apk	未说明权限	检测失败	不符合预期
Test10	上传并检测	高德地图.apk	未说明权限	未说明权限	符合预期

5.4 系统非功能性测试

任何软件系统仅仅满足功能性需求是不够的，还需要满足一些非功能性的需求[20]。本小节将对本系统的非功能性需求测试，其中包含：系统兼容性测试和系统性能测试。

5.4.1 兼容性测试

兼容性测试对于任意一个系统都非常重要。系统可能在不同设备下运行结果都不同，严重者甚至能影响到功能的正常运行。此时兼容性测试就显得尤为重要，如表 5-4 为系统兼容性测试：

表 5-7 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

性能测试对于系统亦非常重要，不错的性能能收获大量的用户。相反，也能够对用户造成极其不好的体验。因此性能测试尤为重要，本小节对性能测试进行了着重介绍。本文利用不同大小的各类 App 进行了系统检测速度测试并实时的记录其运行时间，具体测试结果如表 5-8 所示。

通过观察性能测试表，可以得知，影响检测速度的因素有很多，当 App 不存在隐私政策时，后续检测步骤省略，检测时间自然缩短。而当 App 的体量过大则会导致反编译占据大量时长或筛选出的 URL 较多，导致爬虫执行时间长，最终检测时间也变长。

表 5-8 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	Q 音探歌.apk	<50MB(17.8MB)	18.57s
Test2	上传并检测	学小易.apk	<50MB(20.7MB)	5s
Test3	上传并检测	云达人.apk	<50MB(38.0MB)	17.06s
Test4	上传并检测	Cirta.apk	<50MB(15.4MB)	3.84s
Test5	上传并检测	Clash.apk	<50MB(19.1MB)	4.80s
Test6	上传并检测	小鸡模拟器.apk	≥50MB,<100MB(63.1MB)	16.79s
Test7	上传并检测	粉笔教育.apk	≥50MB,<100MB(53.0MB)	39.36s
Test8	上传并检测	夸克.apk	≥50MB,<100MB(78.2MB)	50.12s
Test9	上传并检测	美团外卖.apk	≥50MB,<100MB(64.86MB)	-
Test10	上传并检测	高德地图.apk	≥50MB,<100MB(88.0MB)	1:28.89s

5.5 本章小结

本章节中主要对系统的测试进行了相关介绍。采用系统功能性测试对关键的模块进行了详细测试与介绍。其中包括文件处理模块、隐私政策获取模块、日志模块、隐私政策分析四个主要模块。另外，对系统进行了非功能性测试，包含兼容性和性能两方面。通过对比测试数据的预期结果和实际结果，确认系统满足了用户的功能需求并且具有一定的稳定性。

6 总结和展望

6.1 本文工作总结

目前，Android 的隐私安全已经成为全社会关注的热点。本文以隐私政策为切入点，探索隐私政策中未明示的地方。本文通过结合 App 清单文件申请的权限与隐私政策，利用相关的技术实现了检测隐私政策明示性的功能。

本文主要是针对《App 违法违规收集使用个人信息行为认定方法》中的第一条展开了设计与实现。第一条规则规定 App 中不存在隐私政策或是隐私政策中没有对收集个人信息规则进行规定都属于隐私政策未明示。本文围绕这条规则设计了 App 的检测流程。

为了提高隐私政策获取的速度，本文采用 Java 调用命令的方法去搜索指定目录下的文件内容，这其实是利用了操作系统的 dos 命令。倘若这一步利用 Java IO 流实现，会耗费大量的资源和时间。一个 App 的反编译目录存在大量的文件夹，Java IO 流读取会进行大量的递归处理，从而导致效率不高。然而使用操作系统本身 dos 命令会导致环境配置困难，不同的操作系统面临着使用不同的命令。比如，Windows 系统搜索目录文件的命令为 findstr，而 Linux 则是 grep。这就造成了系统的部署面临着多套环境配置，为系统的推广增加了困难，这也是系统的不足之处。

最后，在检测结果中，本文输出了 App 申请的权限、App 未进行说明的权限、权限库、App 内的 URL 以及 App 的隐私政策多个信息。帮助用户全面了解 App 不合规之处。

6.2 展望

由于本文主要是针对《App 违法违规收集使用个人信息行为认定方法》中的第一条展开的设计与实现，这也就导致对于其他方面的检测有所欠缺。比如，《App 违法违规收集使用个人信息行为认定方法》规定，进入 App 主界面后，多于 4 次点击访问到隐私政策也属于隐私政策未明示。本文未对这方面进行相关的检测设计与实现，这也是将来系统升级可以实现的新内容。

另一个方面，系统将来可以对 App 首次运行弹窗进行检测。通过该检测，可以判断 App 在首次运行时是否进行了隐私政策提示，或是可以判断 App 是否默认同意隐私政策，默认同意也属于隐私政策未明示。

在后续的研究中，考虑对《App 违法违规收集使用个人信息行为认定方法》中其他认定方法的实现，在逐步的完善中，检测会越来越精确，也为用户带来更强大的保障。

致　谢

时间总在不经意间悄然逝去，四年的大学时光迎来了落幕。随着时间的推移，论文马上就要完成，在这里，我要感谢所有给予过我帮助和关心的人。

首先感谢我的导师刘晓建！本文的研究工作是在刘老师的悉心指导和严格要求下完成的。在课题的研究和论文的写作过程中，刘老师给了我很多耐心的指导和启发。让我学习到的不只有老师渊博的知识，还有对待事情一丝不苟的态度。毕业设计初期阶段，我对课题并没有很深刻的理解，这也导致毕设迟迟没有进度。刘老师在每次开会时都会给我一些实现思路，在慢慢的学习理解中，形成了现有的系统。

然后还要感谢的是四年来所有教导过我的老师，如果没有他（她）们的教导，四年来我不会成长这么多。书山有路勤为径，学海无涯苦作舟。四年的学习生活是苦中带甜的，正是因为老师们的存在才会让其中的苦能少一分。老师们将复杂的知识以简单的方式讲解出来，化繁杂为简单，这是我自己不能做到的。因此，我由衷的感谢四年来相遇的老师。

我还要感谢我的同学和舍友，在我遇到困难的时候，是他（她）们牺牲自己的时间来帮助我。四年里我无数次感概友情的力量，我也一直相信友谊天长地久，即使快要分别，但我知道那是为了更好的重逢。

最后，再次感谢，感谢给予我关怀和帮助的每一个人！

参考文献

[1] 孟霞,岳鹏宇.移动终端 APP 隐私政策内容分析[J].山西师大学报(社会科学版),2018,45(06):47-54.

[2] 秦克飞.手机 APP 隐私政策的可读性研究[J].情报探索,2019(01):18-23.

[3] 张梦秋.从 APP 使用角度对用户隐私安全的思考[J].电子世界,2019(11):82-83.

[4] 刘百灵,万璐璐.基于移动服务的隐私政策 Petri 网协商算法[J].系统管理学报,2019,28(02):231-239.

[5] 李清文.大数据背景下社交网络用户隐私安全问题浅析[J]《财讯》2019 年 3 期.

[6] 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.

[7] 刘娇,白净.中外移动 APP 用户隐私保护文本比较研究[J].汕头大学学报(人文社会科学版),2017,33(03):82-87.

[8] 王靖瑜,徐明昆,王浩宇,徐国爱.Android 应用隐私条例与敏感行为一致性检测[J].计算机科学与探索,2019,13(01):56-69.

[9] 张永兵.隐私信息检索方法在位置隐私保护中的应用[J].信息与电脑,2018(14):225-226+231.

[10] Muharman Lubis,Rahmat Fauzi,Ahmad Almaarif. Privacy Protection Drivers towards Electronic Platform: A Systematic Review[P]. Proceedings of the 2018 International Conference on Industrial Enterprise and System Engineering (IcoIESE 2018), 2019.

[11] Zi-Peng Zhang,Ming Fu,Xin-Yu Feng. A Lightweight Dynamic Enforcement of Privacy Protection for Android[J]. Journal of Computer Science and Technology, 2019, 34(4):901-923.

[12] Le Yu,Xiapu Luo,Jiachi Chen,Hao Zhou,Tao Zhang,Henry Chang,Hareton K N Leung. PPChecker: Towards Accessing the Trustworthiness of Android Apps' Privacy Policies[J]. IEEE Transactions on Software Engineering, 2021, 47(2):221-.

[13] Abu Bakar Anizah,Mahinderjit Singh Manmeet,Mohd Shariff Azizul Rahman. A Privacy Preservation Quality of Service (QoS) Model for Data Exposure in Android Smartphone Usage[J]. Sensors, 2021, 21(5).

[14] Xing Liu,Jiqiang Liu,Sencun Zhu,Wei Wang,Xiangliang Zhang. Privacy Risk Analysis and Mitigation of Analytics Libraries in the Android Ecosystem[J]. IEEE Transactions on Mobile Computing, 2020, 19(5):1184-1199.

[15] Cha Youngrok,Pak Wooguil. Protecting contacts against privacy leaks in smartphones[J]. PloS one, 2018, 13(7):e0191502.
[16] Baalous Rawan,Poet Ronald. Utilizing Sentence Embedding for Dangerous Permissions Detection in Android Apps' Privacy Policies[J].International Journal of Information Security and Privacy (IJISP), 2021, 15(1):173-189.
[17] 李晓军主编.城市地下空间信息化技术指南:同济大学出版社,2016.04
[18] 刘效伯. Android 系统隐私泄露检测与保护研究[D].东南大学,2017.
[19] Akito Monden,Masateru Tsunoda,Mike Barker,Kenichi Matsumoto.Examining Software Engineering Beliefs about System Testing Defects[J]. IT Professional Magazine, 2017, 19(2):58-.
[20] 王博.基于个性化推荐的居舍优品商城的设计与实现[D].北京交通大学,2020.
[21] DCloud.什么是 uni-app[EB/OL].2020.https://github.com/dcloudio/uni-app/blob/master/docs/README.md
[22] 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].兰州大学,2020.
[23] 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
[24] 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