软件学报 ISSN 1000-9825, CODEN RUXUEW
2025,36(11):5102−5117 [doi: 10.13328/j.cnki.jos.007387] [CSTR: 32375.14.jos.007387]

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

# 基于语义分析的小程序代码与隐私声明一致性检测*

刘力沛[1], 毛 剑[1,2,3], 林其箫[1], 吕雨松[1], 李嘉维[1], 刘建伟[1]

[1](北京航空航天大学 网络空间安全学院, 北京 100191)

[2](北京航空航天大学杭州创新研究院, 浙江 杭州 310051)

[3](天目山实验室, 浙江 杭州 311115)

通信作者: 毛剑, E-mail: maojian@buaa.edu.cn

**摘 要**: 小程序需要为用户提供隐私声明, 告知要使用的隐私信息种类和目的. 代码与隐私声明不一致的小程序可能会欺骗用户导致用户隐私泄露. 现有一致性检测方法中, 将二者转为预设的标签进行一致性判断的方法会损失信息导致漏报, 而仅依靠代码分析的方法也难以应对混淆处理的小程序代码. 针对上述问题, 提出基于语义分析的小程序代码与隐私声明一致性检测方法, 根据定制化污点分析结果提取代码行为, 使用代码语言处理模型将敏感资源使用代码表示为自然语言, 结合隐私声明中资源使用目的, 人工检测与代码行为的一致性. 实验结果表明, 污点分析模块覆盖小程序接口的全部 3 种数据返回方式和 4 种常见数据流, 较同类方法提升小程序敏感行为发现能力; 在上万个小程序语义分析中, 发现高频调用接口的部分行为存在隐私泄露风险, 识别出真实环境中代码与隐私声明不一致的小程序.

**关键词**: 小程序; 代码与隐私声明一致性; 污点分析

**中图法分类号**: TP311

中文引用格式: 刘力沛, 毛剑, 林其箫, 吕雨松, 李嘉维, 刘建伟. 基于语义分析的小程序代码与隐私声明一致性检测. 软件学报, 2025, 36(11): 5102–5117. http://www.jos.org.cn/1000-9825/7387.htm

英文引用格式: Liu LP, Mao J, Lin QX, Lyu Y, Li JW, Liu JW. Code to Policy Consistency Detection for Mini Program Based on Semantic Analysis. Ruan Jian Xue Bao/Journal of Software, 2025, 36(11): 5102–5117 (in Chinese). http://www.jos.org.cn/1000-9825/7387.htm

## Code to Policy Consistency Detection for Mini Program Based on Semantic Analysis

LIU Li-Pei[1], MAO Jian[1,2,3], LIN Qi-Xiao[1], LYU Yu-Song[1], LI Jia-Wei[1], LIU Jian-Wei[1]

[1](School of Cyber Science and Technology, Beihang University, Beijing 100191, China)

[2](Hangzhou Innovation Institute of Beihang University, Hangzhou 310051, China)

[3](Tianmushan Laboratory, Hangzhou 311115, China)

**Abstract**: Mini programs are required to provide privacy policies to inform users about the types and purposes of the privacy data being collected and used. However, inconsistencies between the underlying codes and the privacy statements may occur, potentially deceiving users and leading to privacy leakage. Existing methods for detecting such inconsistencies typically rely on converting the code and policies into predefined labels for comparison. This approach introduces information loss during label conversion, resulting in underreporting. In addition, traditional code analysis methods are often ineffective against obfuscated mini program code. To address these limitations, a semantic-analysis-based method for code-to-policy consistency detection in mini programs is proposed. Customized taint analysis is utilized to capture code behaviors based on mini program coding paradigms, and a code language processing model is applied to represent these behaviors as natural language descriptions. By aligning the natural language representation of code behaviors with the stated purposes in privacy policies, expert reviewers can analyze the consistency between the two effectively. Experiments indicate that the proposed taint

* 基金项目: 国家自然科学基金 (62172027, U24B20117); 国家重点研发计划 (2020YFB1005601); 浙江省自然科学基金 (LZ23F020013)
收稿时间: 2024-06-06; 修改时间: 2024-08-29, 2024-10-14; 采用时间: 2024-12-24; jos 在线出版时间: 2025-07-17
CNKI 网络首发时间: 2025-07-18

analysis module covers all three data return methods and four common data flow patterns within mini programs APIs, achieving superior sensitivity compared to existing methods. Semantic analysis of tens of thousands of mini programs reveals privacy leakage risks associated with certain high-frequency API calls. Case studies using the MiniChecker tool further identify real-world instances of mini programs where inconsistencies between code and privacy policies are detected.

**Key words**: mini program; code to policy consistency; taint analysis

“小程序+宿主应用”是一种新型移动应用范式, 其中宿主应用运行在操作系统之上, 执行自身功能的同时, 还作为小程序的应用商店和运行环境, 汇总第三方开发的小程序供用户选择与运行; 小程序运行在宿主应用之上, 通过调用宿主应用为其封装的小程序接口获取系统资源 (如用户位置) 和宿主应用资源 (如电子钱包). 小程序在运行中会收集和使用用户隐私数据, 可能导致用户隐私泄露. 为保护用户隐私安全, 宿主应用要求小程序在发布前填写隐私声明, 向用户介绍收集和使用隐私数据的种类和目的[1]. 但小程序开发者可能故意在隐私声明中隐藏代码的恶意行为, 或者由不了解小程序具体功能的法务团队来制定隐私声明[2], 导致小程序的隐私声明与代码不一致. 宿主应用对小程序代码进行人工分析成本高且效率低, 且小程序用户大多不具备分析小程序代码的能力, 因此难以对小程序的代码与隐私声明一致性进行检测. 代码与隐私声明不一致的小程序可能会误导或迷惑用户并造成用户隐私泄露, 同时给开发者带来合规问题[3].

现有的小程序代码与隐私声明一致性检测方法存在以下问题: (1) 难以有效应对混淆处理后的小程序代码, 而现实中的小程序基本都采用了代码混淆技术, 导致现有方法[4,5]对代码行为分析的覆盖率低; (2) 将代码与隐私声明转化为标签进行对比的方法[2,6−8]的准确性高度依赖于预定义标签的粒度和广度, 需要大量的经验累计并不断更新, 且这种转化会损失代码的信息量, 影响信息的完整性; (3) 在判断标签的分类任务[5,7]中不可避免地会存在分类误差, 进一步影响了一致性分析的准确性. 这可能导致大量误报和漏报问题, 即方法检测出不一致的小程序实际是一致的, 而小程序中许多不一致的资源使用行为却未能检出.

准确且完整地实现小程序代码与隐私声明一致性检测, 关键在于小程序代码行为语义是否全面, 以及对比信息是否完整. 通过分析发现, 尽管可以通过污点分析技术提取数据流, 描述小程序使用隐私数据的代码行为, 但是小程序代码多经过混淆处理, 代码可读性差, 需要针对代码混淆特点制定污点分析规则来提升代码语义分析的完整性. 进一步地, 可以人工进行代码与隐私声明一致性对比, 充分结合小程序的使用场景等上下文信息以提升准确性.

本文提出了基于语义分析的小程序代码与隐私声明一致性检测方法. 以小程序代码包为输入, 经运行时数据收集提取小程序包和隐私声明, 对小程序包逆向解包, 设计污点分析规则提取隐私数据流并使用代码语言处理模型提取得到自然语言表征的代码行为语义, 人工对将代码行为语义与隐私声明的相关阐述的一致性进行分析, 得到代码与隐私声明的一致性关系.

## 1 相关工作

### 1.1 污点分析

污点分析是一种信息流分析技术, 用于分析特定数据的传播流. 污点分析可以抽象成一个二元组<source, sink>, 其中, source (污点源) 表示引入的要关注的数据, sink (污点汇聚点) 表示可能会产生安全问题的敏感函数. 污点分析可以分为两步: (1) 识别污点源和污点汇聚点, 识别方式可以根据具体应用程序调用的接口和重要数据类型进行手工标记, 或者使用统计或机器学习方式进行自动识别; (2) 污点传播分析, 利用特定的规则对程序进行分析以识别出从污点源出发并汇入污点汇聚点的数据流[9,10]. CodeQL[11]是一个开源的污点分析工具, 支持多种代码语言, 并允许分析人员编写查询语句来自定义污点分析规则.

TaintMini[4]是首个针对小程序设计的静态污点分析工具, 综合考虑了小程序的 UI 页面 (即 wxml 文件) 和逻辑代码 (即 JavaScript 文件) 构建了全局数据流图, 然后设计了污点传播规则, 并指定污点源为获取敏感数据的接口, 污点汇聚点为向外部服务器或小程序发送数据的接口.

### 1.2 代码与隐私声明一致性检测相关工作

在安卓应用代码与隐私声明一致性检测方面, Slavin 等人[12]构建了本体 (Ontology) 描述短语之间从属关系以归并隐私声明中对于同一概念的不同描述, 基于 Ontology 映射得到安卓应用实际使用的接口集合, 并计算与隐私声明中阐述的接口集合的差集, 最后通过 FlowDroid[13]分析差集中接口的数据流是否流向了对外发送数据的接口, 得到一致性检测结果.

许多一致性检测方法把代码和隐私声明都转为元组进行对比. Yu 等人[6]提出的一致性检测方法 PPChecker 对隐私声明中的句子进行词性分析, 得到二元组<隐私数据类型+使用行为>; 另一方面, PPChecker 对安卓应用代码进行污点分析查看应用是否会分享或者保存隐私信息, 最后使用预定义的规则对两种来源的分析结果进行匹配. Zimmeck 等人[7]使用信息增益和机器学习算法 TF-IDF 从隐私声明中提取出数据类型和使用行为的关键词, 得到二元组<隐私数据类型, 使用行为>; 通过 Androguard[14]对安卓接口进行污点分析以判断代码行为类型, 最后通过对比两种元组得到一致性结果. Andow 等人[8]提出的一致性检测工具 PoliCheck 使用 AppCensus 平台[15]对安卓应用进行动态污点分析以提取代码数据流并得到二元组<隐私数据类型, 接收方域名或 IP>; 使用 PolicyLint[16]提取隐私声明中的语义, 最后通过计算两种元组的关系得到一致性结果. Tan 等人[2]提出了检测第三方违反用户隐私声明的工具 PTPDroid, 基于构建的 Ontology 提取接口与短语的映射关系, 利用 PolicyLint[16]从隐私声明中提取三元组<实体, 行为, 隐私数据类型>, 利用 FlowDroid[13]对代码进行污点分析得到涉及敏感数据的数据流, 并利用构造的 Ontology 映射将污点流转为行为三元组<实体, 行为, 隐私数据类型>, 最后对比两种三元组得到一致性结果.

而在小程序代码与隐私声明一致性检测方面, Wang 等人[5]使用污点分析提取小程序敏感接口的数据流并得到代码行为二元组<隐私数据类型, 使用行为>, 构建数据集并训练分类器来提取隐私声明的关键词, 进而基于二元组计算一致性. Zhang 等人[17]把 Slavin 等人[12]的一致性对比思路迁移到了小程序中, 首先识别小程序代码使用与隐私声明阐述的接口的差集, 进而对差集中的接口进行污点分析判断是否存在危险行为.

表 1 汇总了上述一致性分析研究, 从表中可以看出大部分研究都将代码与隐私声明转为预定义的标签后进行一致性对比分析[2,5−8], 而也有研究从接口差集的角度进行一致性分析[12,17]. 但是这些分析方法都会不可避免地损失代码和隐私声明中的信息, 这可能影响二者一致性分析的准确性, 而且这种全自动化的一致性检测方法较为程式化, 无法充分考虑移动应用的功能、用户群体等上下文信息.

表 1 移动应用代码与隐私声明一致性检测方法对比表

| 应用 | 方法 | 一致性对比形式 | 代码分析方法 | 隐私声明分析方法 | 可检测的不一致使用行为 | | | |
|---|---|---|---|---|---|---|---|---|
| | | | | | 收集 | 使用 | 保存 | 发送 |
| 安卓应用 | 文献[12] | 分析代码与隐私声明的接口差集 | FlowDroid 污点分析 | 基于Ontology提取敏感数据与接口的映射 | | | | √ |
| | 文献[6] | 将代码与隐私声明转为二元组<隐私数据+使用行为>后规则匹配 | FlowDroid 污点分析 | 词性分析提取二元组 | | | √ | √ |
| | 文献[7] | 将代码与隐私声明转为二元组<隐私数据+使用行为>, 对比隐私数据对应的使用行为是否相同 | Androguard 污点分析 | 机器学习提取数据类型和使用行为的关键词构成特征向量, 分类器判断使用行为 | √ | | | √ |
| | 文献[8] | 将代码与隐私声明转为二元组<隐私数据类型, 接收方实体名>后对比 | AppCensus 污点分析 | PolicyLint提取四元组 | √ | | | √ |
| | 文献[2] | 将代码与隐私声明转为三元组<第三方实体名, 使用行为, 隐私数据类型>后对比 | FlowDroid 污点分析 | PolicyLint提取四元组 | √ | | √ | √ |
| 小程序 | 文献[5] | 将代码与隐私声明转为二元组<隐私数据类型, 使用行为>后对比 | 污点分析 | 分类器提取阐述并分词, 从预先构建的“数据类型”字典中计算相似单词得到二元组 | √ | √ | | √ |
| | 文献[17] | 对代码收集的与隐私声明阐述的隐私数据差集进行分析 | 污点分析 | NLP提取关键词 | √ | | | |

## 2 方法设计

为实现完整、准确的小程序代码与隐私声明一致性检测, 本文提出一种基于语义分析的小程序代码与隐私声明一致性检测方法, 其设计框架如图 1 所示. 以小程序代码包为输入, 经运行时数据收集提取小程序包和隐私声明, 对小程序包经逆向解包、敏感接口定位、污点分析和语义分析后得到自然语言表述的代码行为语义, 对隐私声明进行规则匹配得到对应的隐私声明阐述后, 人工对二者一致性进行分析, 得到代码与隐私声明的一致性关系.

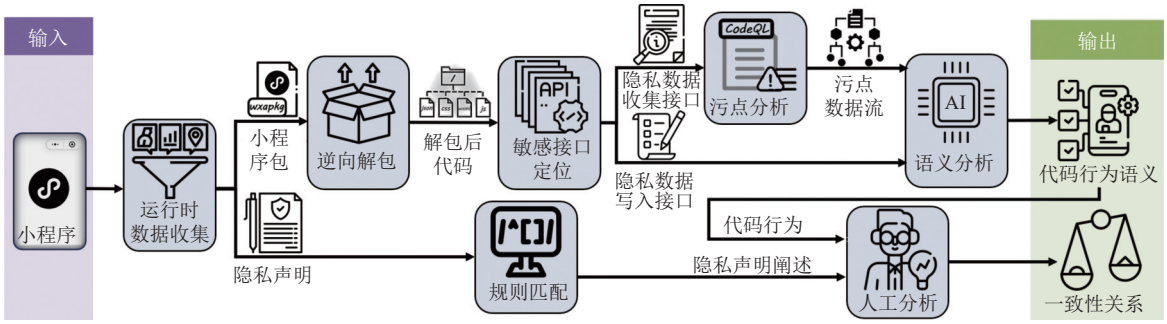



图 1 小程序代码与隐私声明一致性检测方法框架图

### 2.1 运行时数据收集

在宿主应用加载小程序时, 会从小程序服务器下载小程序包并缓存到本地. 为对小程序代码进行分析, 需要在宿主应用中运行对应的小程序, 并在宿主应用的本地文件目录中提取缓存的小程序包用于分析. 另一方面, 小程序隐私声明也需要在小程序运行时的设置页面中进行收集.

### 2.2 逆向解包

小程序开发者在发布小程序时, 小程序开发工具会对将小程序源代码进行打包处理, 即分别进行 Webpack 打包[18]、文件整合、加密处理后得到小程序包. 其中, Webpack 打包时会进行代码混淆压缩, 或者使用代码分割把代码分割成多个小的代码块 (bundle), 提高加载效率和利用率, 且 Webpack 打包不可逆.

根据小程序代码打包流程, 进行逆向解包时先对其进行解密, 之后根据文件整合规则, 将文件拆分并还原到不同页面或者全局的对应目录下. 由于 Webpack 打包操作是不可逆的, 即使对小程序包进行逆向解包, 得到的源代码也是经过混淆压缩、模块合并等多种转换后的代码, 代码可读性较低, 直接对代码进行人工分析难度较大, 因此需要对代码进行自动化分析以提取有效信息.

### 2.3 敏感接口定位

文献 [19] 用于帮助开发者编写隐私声明. 其中, 共包含 45 个涉及用户隐私的敏感小程序接口, 并将其分为 21 类, 其中既有位置信息等系统资源相关的敏感接口, 也有昵称头像等宿主应用资源相关的敏感接口. 此外, 本文还发现有 13 个接口涉及用户隐私但未被列在文献 [19] 中, 因此本文将这 13 个接口也纳入考虑中. 这些接口可以分为如下 3 类.

(1) 隐私数据收集接口. 隐私数据收集接口是指调用时会返回隐私数据的接口. 如接口“wx.getLocation”在调用时会通过回调函数返回用户当前的地理位置、速度等隐私数据.

(2) 隐私数据写入接口. 隐私数据写入接口是指调用时不返回隐私数据但会将隐私数据存入本地的接口. 如接口“wx.addPhoneContact”负责将联系人信息写入本地通讯录, 而不涉及通讯录读取.

(3) 隐私资源操作接口. 隐私资源操作接口是指调用时既不返回敏感数据也不涉及敏感数据写入的接口. 如接口“wx.stopCompass”负责停止监听罗盘数据, 即释放敏感资源, 也不会返回罗盘实时数据, 接口“wx.authorize”负责向用户申请使用某项资源或者访问某项数据的权限, 而不会真正执行使用或访问操作.

特别地, 小程序接口中存在管理器类接口. 这类接口在使用前需要先通过一个小程序接口获取管理器, 管理器

是一个拥有许多函数作为内部方法的对象. 以管理器类接口“RecorderManager.start”为例, 要调用此接口来开始录音, 需要先调用小程序接口“wx.getRecorderManager”获取管理器对象“RecorderManager”, 进而再调用此对象的“start”方法, 如图 2 所示. 由于管理器类接口内部方法才涉及隐私数据收集或者隐私数据写入, 因此本文不单独把管理器类接口作为一类进行研究, 而是根据其功能归类到隐私数据收集接口或隐私数据写入接口中.

```
//获取管理器
var recorderManager = wx.getRecorderManager();
//开始录音
recorderManager.start();
//录音5秒
setTimeout(5000);
//停止录音
recorderManager.stop();
//获取录音文件路径
recorderManager.onStop(function (res){
    console.log('录音文件路径:', res.tempFilePath);
})
```

图 2 管理器类接口调用示例

接下来, 对于隐私数据收集接口, 由于其返回数据的流向将影响用户隐私安全, 因此将对其进行污点分析提取接口返回的隐私数据的流向来分析其代码行为. 对于隐私数据写入接口, 其接口调用本身就会影响用户隐私数据, 因此直接使用接口调用语句进行代码语义分析.

### 2.4 面向隐私数据收集接口的污点分析

为分析隐私数据收集接口返回的数据是否存在泄露的风险, 需要对每次隐私数据收集接口调用所返回的数据进行污点分析. 分析发现, 隐私数据收集接口的数据返回方式主要有以下两种.

(1) 回调函数 (callback). 这类接口在调用时需要传入一个对象变量, 并在此对象变量中声明属性名为“success”“fail”和“complete”的回调函数作为属性值. 当接口查询到隐私数据时, 会将其作为形参传入并运行“success”回调函数; 如果查询失败, 会将错误信息作为形参传入并运行“fail”回调函数; “complete”回调函数无论接口调用成功或失败都会被执行. 如图 3(a) 所示, 小程序调用接口“wx.getLocation”后, 用户位置信息将作为回调函数“success”的参数值返回给小程序来分析处理. 除了以匿名函数的形式在接口调用时直接声明函数, 也可以通过函数变量的形式实现接口调用, 如图 3(b) 所示.

```
var data = {};
wx.getLocation({
    type: 'wgs84',
    success: function(res) {
        data.latitude = res.latitude;
        data.longitude = res.longitude;
        console.log(data);
    }
})
```

(a) 回调函数类接口的 obiect.success 为匿名函数

```
var handler = function (res) {
    const tempFilePaths = res.tempFiles;
    console.log(tempFilePaths)
}

wx.chooseMessageFile({
    count: 10,
    type: 'image',
    success: handler
})
```

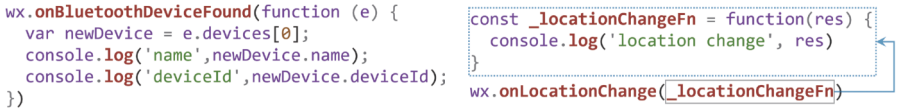
(b) 回调函数类接口的 obiect.success 为函数变量

图 3 回调函数类隐私数据收集接口代码示例

(2) 监听函数 (listener). 异步接口主要采用监听函数返回数据. 在调用时需要传入一个拥有形参的监听函数, 当调用此接口并监听到接口触发事件后, 接口将返回的隐私数据将作为参数传给监听函数并运行此监听函数. 如图 4(a) 所示, 小程序在调用负责监听搜索到新蓝牙设备的事件的接口“wx.onBluetoothDeviceFound”时会传入一个监听函数, 当“搜索到新蓝牙设备”这一事件发生时, 接口库会将新搜索到的蓝牙设备信息作为参数传递给这个监听函数并调用此监听函数. 监听函数的调用也可以通过函数变量的形式实现, 如图 4(b) 所示.

根据对数据返回方式的分析, 本文确定污点分析的污点源为隐私数据收集接口返回的数值, 即回调函数和监

听函数的参数值. 此外, 对管理器类接口进行污点分析时可以直接将管理器接口本身作为污点源, 在污点分析中可以进一步追踪到对管理器对象内部方法的调用.




(a) 监听函数类接口的监听函数为匿名函数 (b) 监听函数类接口的参数值为函数变量

图 4 监听函数类隐私数据收集接口代码示例

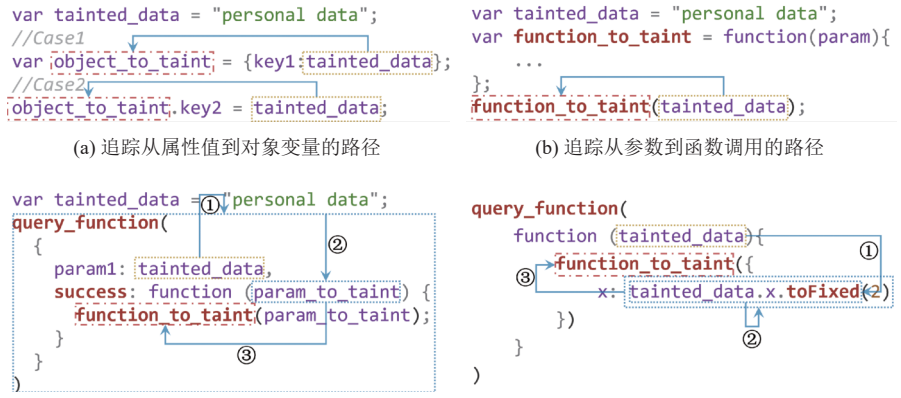
而污点分析的污点汇聚点为小程序接口. 本文注意到小程序的使用隐私数据的行为, 不论是发请求至服务器等存在隐私泄露风险的行为, 还是在页面中显式给用户的常规行为, 都需要调用小程序接口进行实现. 其他 JavaScript 内置函数或接口的调用无法有效表征小程序代码行为, 因此不予考虑.

本文进一步分析发现, 默认的污点分析规则无法对代码混淆处理后的 4 种数据流进行有效追踪, 因此本文对这 4 种数据流进行分析并添加了污点分析规则.

(1) 从对象变量的属性值到对象变量自身的数据流. 如图 5(a) 所示, 既包括在对象变量初始化时的值传递 (case1), 也包括直接对对象变量的某个属性进行赋值操作 (case2). 添加的污点流分析规则如公式 (1) 所示,

$$(n_a \rightarrow prop) \wedge (prop \in object\ n_b) \Rightarrow n_a \rightarrow n_b \tag{1}$$

其中, $\Rightarrow$ 之前表示要识别的数据流, $\Rightarrow$ 之后表示要添加追踪的数据流, $\rightarrow$ 表示数值传递关系, $n_a$ 表示污点数据结点, $n_b$ 表示应标记为污点但默认规则未标记的数据结点且在此规则中为 *object* 类型即对象变量, *prop* 为对象变量的属性.




(a) 追踪从属性值到对象变量的路径 (b) 追踪从参数到函数调用的路径

(c) 追踪从一属性值到另一属性回调函数参数的路径 (d) 追踪从函数名到函数调用的路径

图 5 默认污点分析规则无法追踪的数据流

(2) 从函数参数到函数调用的数据流. 如图 5(b) 所示, 如果缺少这种数据流, 污点分析的污点汇聚点难以明确到函数调用. 添加的污点流分析规则如公式 (2) 所示,

$$(n_a \rightarrow param) \wedge (param \in function\ n_b) \Rightarrow n_a \rightarrow n_b \tag{2}$$

其中, $n_b$ 为 *function* 类型即函数, $n_a$ 为污点数据结点且流向了函数 $n_b$ 的参数 *param*.

(3) 从函数调用到回调函数或监听函数参数的数据流. 如图 5(c) 所示, 函数“query_function”接受的参数为一个对象变量, 此对象变量包括记录污点数据的“param1”属性和作为回调函数的“success”方法. 由于函数“query_function”的运行依赖于污点数据“tainted_data”, 因此 CodeQL 默认污点分析规则会把“query_function”函数调用标记为污点 (数据流①). 但进一步考虑到“query_function”执行后返回的结果可能受到输入的污点数据的影响, 因此需要把其回调函数返回的参数值也标记为污点数据, 即添加从函数调用到函数回调函数的参数的路径 (数据流②). 类似地,

也需要添加从函数调用到函数监听函数参数的路径. 添加的污点流分析规则如公式 (3) 所示,

$$(n_a \rightarrow object\ obj) \wedge (callback\ n_b \in obj) \Rightarrow n_a \rightarrow n_b \tag{3}$$

其中, 污点数据结点 $n_a$ 流向了对象变量 *obj*, 回调函数 $n_b$ 作为 *obj* 的属性也应被标记为污点结点.

(4) 从函数名到函数调用的数据流. 如图 5(d) 所示, 在调用隐私数据收集接口“query_function”时传入一个监听函数, 此监听函数的形参“tainted_data”为需要关注的污点数据, 但 CodeQL 只能追踪到对此污点数据内部属性的访问 (数据流①), 而无法进一步追踪到内部属性的函数调用 (数据流②), 导致无法追踪到此污点数据被传入需要标记的威胁函数“function_to_taint” (数据流③), 因此需要添加从函数名到函数调用的路径追踪 (数据流②). 添加的污点流分析规则如式 (4) 所示,

$$(n_a \rightarrow function) \wedge (function \rightarrow callnode\ n_b) \Rightarrow n_a \rightarrow n_b \tag{4}$$

其中, 污点数据结点 $n_a$ 流向了一个函数, 且此函数在结点 $n_b$ 被调用, 因此需要把 $n_b$ 标记为污点结点.

### 2.5 基于代码语言处理模型的语义分析

敏感数据最终的流向是判断数据是否存在泄露风险的主要依据, 因此本文提取污点汇聚点所在的代码; 另一方面, 对于隐私数据写入接口, 由于此类接口调用本身就会影响用户隐私数据, 因此直接提取该接口的调用语句. 上述两类接口对应的隐私行为代码片段都是函数调用, 其语句的元素依次为接口名、括号、参数、括号. 进行接口语句调用提取时, 本文通过判断括号的闭合即可提取出语义完整的代码判断. 考虑到参数内可能也有括号, 因此在进行括号闭合判断时, 利用队列数据结构, 对接口调用语句进行遍历时, 将所有遇到的左括号入队, 在遇到右括号时执行一次出队, 当队列为空时, 说明已定位到接口调用语句的句尾, 此过程见算法 1.

**算法 1**. 敏感接口调用语句提取算法.

输入: 代码文本 *codeText*, 敏感接口名 *targetAPI*;
输出: 敏感接口调用语句 *targetSnippet*.

```
cursorIndex = codeText.find(targetAPI) //定位游标为敏感接口所在的位置
cursorStart = cursorIndex //记录敏感接口语句开始位置
bracketQueue = ["("] //将接口调用语句的第 1 个左括号入队
cursorIndex += len(targetAPI) //更新游标为第 1 个左括号后的第 1 个字符
//遍历之后字符, 直到括号闭合
while bracketQueue do
    if codeText [cursorIndex] == "(": //遇到左括号, 左括号入队
        bracketQueue.enqueue("(")
    elif codeText [cursorIndex] == ")": //遇到右括号, 出队一个左括号
        bracketQueue.dequeue()
    cursorIndex +=1
return codeText [cursorStart: cursorIndex]; //返回敏感接口调用语句
```

提取得到的接口调用语句中, 不只是接口名包含代码行为信息, 参数也将影响代码行为. 如对于后文图 6 中的代码片段, 单从代码片段的接口名只可以得知接口调用所拍摄的照片被用于向外部服务器发送请求, 但如果结合接口调用语句中参数的 URL 值 (虚线框出的部分), 就可以得知接口调用所拍摄的照片将被发送至外部服务器来执行图像分类任务. 因此, 为更好利用代码片段中参数、代码结构、代码逻辑等代码上下文所包含的信息, 且方便与隐私声明进行一致性对比分析, 采用代码语言处理模型来分析代码片段并输出代码行为的自然语言描述.

### 2.6 隐私声明规则匹配与一致性检测

本文将小程序代码与隐私声明的一致性关系分为 4 类: (1) 隐私声明的阐述与代码使用敏感数据的行为相一

致; (2) 代码中使用了某种敏感数据, 但隐私声明没有进行阐述或阐述不全面; (3) 隐私声明阐述了代码没有使用的敏感数据; (4) 隐私声明既存在不全面的阐述, 也存在多余阐述, 即同时存在 (2) 和 (3) 问题.

```
wx.request({
    url: "https://**.com/rest/2.0/image-classify/v2/advanced_general?access_token=" + e.access_token,
    method: "POST",
    header: { "content-type": "application/x-www-form-urlencoded" },
    data: { image: e.img, baike_num: 0 },
    success: function (a) { t.setData({ src: e.src, imgList: a.data.result }) };
});
```

图 6　参数中包含代码行为信息的代码片段

由于隐私声明的编写方式较为主观, 小程序的隐私声明和代码行为之间仍可能存在一定差异, 比如同样是收集用户账户昵称, 小程序 A 声明的目的是“为了用户登录”, 而小程序 B 声明的目的是“为了分辨用户”, 二者目的相似, 但描述角度截然不同, 难以直接计算二者的自然语言的语义距离来分析一致性. 基于上述问题, 现有方法[2,5]把代码和隐私声明都转为标签进行对比以进行一致性检测, 但转化过程中不可避免地会损失信息, 而且完全自动化的检测方法难以利用小程序使用场景等上下文信息进行一致性判断.

针对上述问题, 为更好地结合小程序的使用场景等上下文信息, 本文基于手工分析代码与隐私声明的一致性. 代码语义方面, 通过之前的污点分析、语义分析等方法, 小程序调用敏感接口的代码行为已转为自然语言描述, 之后根据敏感接口与资源的映射关系将其转为小程序使用敏感资源的语义, 帮助分析人员了解代码的敏感行为. 隐私声明方面, 可以对小程序隐私声明进行正则匹配, 自动化提取出小程序使用的资源和目的. 这样, 分析人员可以直接对比小程序使用某项敏感资源的代码行为和隐私声明阐述, 判断得到每项资源的代码与隐私声明一致性关系.

## 3　方法测试与分析

### 3.1　方法实现与测试环境

MiniChecker 采用 Python 语言编程实现并运行于 Ubuntu 系统中, 硬件配置为 Intel Core i9-13900K CPU 和 32 GB RAM. 各模块的实现细节如下.

(1) 运行时数据收集: 利用 Android 调试桥 (ADB)[20]连接安卓系统并进入 ROOT 账户后, 从小程序包缓存文件夹中获取“.wxapkg”后缀文件即为小程序包, 隐私声明文件在小程序运行时的设置页面中进行收集.

(2) 逆向解包: 使用开源工具 unveilr[21]实现, 可以对小程序包进行解密和分离, 并根据分离得到的代码文件的 AST 结构来调整代码结构以提升代码可读性.

(3) 敏感接口定位: 根据本文设计的敏感接口分类原则, 定位的敏感接口数据已在仓库 (https://gitee.com/GODKID/MiniChecker-privacy-api) 中公开.

(4) 污点分析: 采用开源工具 CodeQL[11]实现, CodeQL 支持对小程序采用的 JavaScript 语言进行污点分析, 且可以自定义数据流追踪规则来进行污点分析, 根据 CodeQL 语法规则, 对本文发现默认规则无法追踪的 4 种数据流的伪代码见算法 2.

**算法 2**. CodeQL 定义污点分析类以添加 4 种污点追踪数据流的伪代码.

输出: 定义的可以追踪 4 类数据流的污点分析类 *DataTrackingConfiguration*.

```
    //定义由污点数据进行追踪得到污点汇聚点的污点分析类.
class DataTrackingConfiguration extends TaintTracking::Configuration:
    override predicate isAdditionalTaintStep (DataFlow::Node pred, DataFlow::Node succ):
        //添加要追踪的数据流 a) 从属性值到对象变量的数据流
        exists (DataFlow::PropWrite prop_node| //定义 prop_node 为写入对象属性的结点
            pred = prop_node.value and
```

```
            succ = prop_node.object
        ) or
        //添加要追踪的数据流 b) 从参数到函数调用的数据流
        exists (DataFlow::CallNode call_node| //定义 call_node 为函数调用结点
          pred = call_node.argument and
          succ = call_node
        ) or
        //添加要追踪的数据流 c) 从一属性值到另一属性回调函数参数的数据流
        exists (DataFlow::CallNode call_node, DataFlow::FunctionNode function_node| //定义 call_node 为函数调用结点, function_node 为函数结点
          pred = call_node and
          (
            function_node = call_node.callback
            or
            function_node = call_node.listener
          ) and
          succ = function_node.parameter
        ) or
        //添加要追踪的数据流 d) 从函数名到函数调用的数据流
        exists (DataFlow::CallNode call_node| //定义 call_node 为函数调用结点
          pred = call_node.name
          succ = call_node
        )
```

(5) 语义分析: 代码语言处理模型采用 ProphetNet-X[22]用于代码语义分析任务的下游模型 ProphetNet-Code[23], 该模型在 CodeXGLUE[24]平台发布的 JavaScript 代码语义分析任务排行榜[25]中排名第 1.

(6) 规则匹配: 根据小程序隐私声明固定的阐述范式[19]进行正则匹配, 提取出小程序使用敏感资源的类别与使用目的.

(7) 人工分析: 针对每个敏感资源调用将代码行为与隐私声明阐述进行对比, 得到一致性关系.

测试选用的数据集包括 MiniCrawler[26]公布的小程序包和我们从本地文件系统读取的缓存的小程序包, 共计 33 770 个. 本文从以下 4 个方面评估我们的方法: (1) 相比于现有方法, 本文方法污点分析模块追踪小程序敏感接口返回数据的有效性; (2) 污点分析和语义分析模块不同参数选择对小程序敏感代码行为分析的影响; (3) 小程序频繁调用敏感接口的种类及其调用的主要代码行为; (4) 代码与声明不一致的小程序发现.

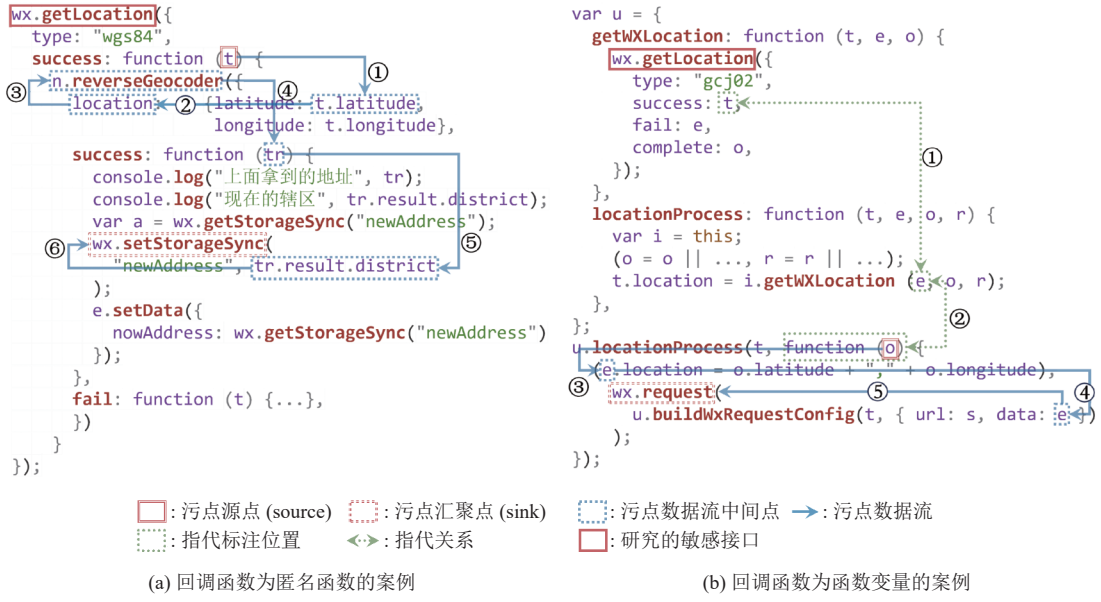
### 3.2 污点分析模块有效性测试

通过污点分析提取代码数据流是本文方法开展代码与隐私数据声明一致性检测的关键. 通过对小程序代码的分析, 小程序具有 3 种隐私数据返回方式 (即, 回调函数、监听函数、管理器) 以及 4 种默认污点分析规则无法追踪的数据流. 本节测试本文方法污点分析模块针对上述小程序独有数据流的追踪识别有效性, 并与现有小程序污点分析工具 TaintMini[4]和 Wang 等人[5]的方法对比, 验证了本文方法污点分析模块的有效性.

为测试污点分析模块的有效性, 本文分别针对回调函数、监听函数、管理器这 3 种隐私数据返回方式分别选取真实小程序代码作为测试环境进行有效性测试.

对于回调函数类接口 (callback) 返回数据的情形, 针对回调函数为匿名函数返回的数据, MiniChecker 的污点

分析效果如图 7(a) 所示, 接口“wx.getLocation”将用户位置信息通过 success 回调函数的参数 *t* 进行返回, 因此参数 *t* 为污点源. 污点源 *t* 作为参数值传入对象变量 *location* (数据流①②) 后, 这一对象变量又作为函数 *n.reverseGeocoder* 的参数参与了函数调用 (数据流③). 这一函数调用的返回值会通过传入函数的 success 回调函数的参数 *tr* 返回 (数据流④), 进而 MiniChecker 将 *tr* 也标记为污点数据. 之后 *tr* 的数值参与了小程序接口“wx.setStorageSync”的调用 (数据流⑤⑥), 即接口“wx.setStorageSync”为污点分析的污点汇聚点. 针对回调函数为函数变量返回的数据, MiniChecker 的污点分析效果如图 7(b) 所示, 接口“wx.getLocation”的 success 回调函数 *t* 是函数“getWXLocation”的参数 (数据流①), 并在调用“locationProcess”函数时声明了实际函数 (数据流②), 因此 MiniChecker 第 1 次污点分析定位到此匿名函数, 并将其参数 *o* 标记为污点源. 之后第 2 次污点分析 (数据流③④⑤) 得到污点汇聚点为向服务器发送请求的接口“wx.request”. 通过上述两个代码片段分析发现, MiniChecker 的污点分析模块可以对回调函数类隐私数据收集接口进行有效追踪. 为方便起见, 图 7(b) 中删除了源文件中部分无关代码, 并调整了部分变量名.




(a) 回调函数为匿名函数的案例 (b) 回调函数为函数变量的案例

图 7 对回调函数形式返回数据接口的污点分析测试案例

针对监听函数类接口 (listener) 返回数据的情形, MiniChecker 的污点分析效果如图 8 所示. 图中传入监听罗盘数据变化的接口“wx.onCompassChange”的监听函数为一个匿名函数, 将匿名函数的参数 *n* 标记为污点源, 经过污点分析 (数据流①②) 后得到污点汇聚点为接口“this.setData”, 即在页面进行显示. 针对监听函数为函数变量返回的数据, 污点分析流程与回调函数相同, 因此这里不再赘述. 上述分析证明 MiniChecker 的污点分析模块可以对监听函数类隐私数据收集接口进行有效追踪.

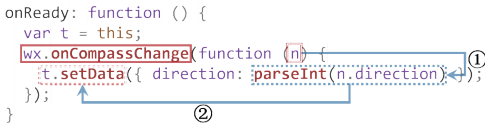


图 8 对监听函数形式返回数据接口的污点分析测试案例

针对管理器类接口 (manager) 内部方法返回数据的情形, 调用管理器类接口后会返回管理器实例, 开发者需要进一步调用管理器实例的内部方法来获取隐私数据. 管理器实例的内部方法返回数据的方式也主要是回调函数和

监听函数, 因此 MiniChecker 只需要可以由管理器实例追踪到其内部方法调用, 即可实现对管理器类接口的污点分析, 对应的分析效果如图 9 所示, 调用管理器类接口“wx.getFileSystemManager”后返回管理器示例 *r* (数据流①), 在此过程中 MiniChecker 首先将变量 *r* 标记为污点源, 之后由于小程序调用了 *r* 的内部方法“readFile” (数据流②), MiniChecker 追踪到“readFile”的 success 回调函数的调用, 将其参数 *i* 标记为污点. 根据上述测试分析, MiniChecker 的污点分析模块可以对管理器类接口进行有效的污点分析.

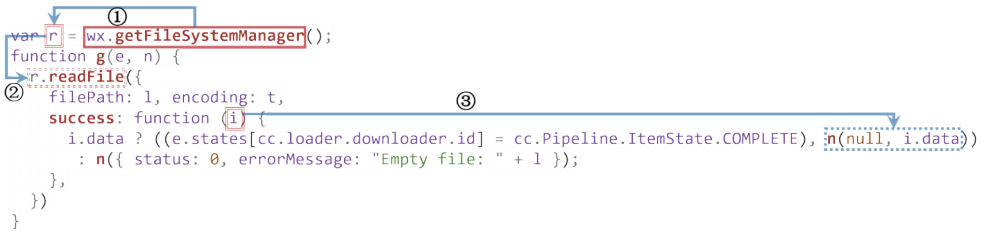



图 9 对管理器类接口的污点分析测试案例

此外, 对于默认污点分析规则无法追踪的 4 种数据流, 上述测试案例也可以证明实现的污点分析模块的有效性. 具体地, 从对象变量的属性值到对象变量自身的数据流在图 7(a) 中的数据流②得到了验证; 从函数参数到函数调用的数据流在图 8 中的数据流①得到了验证; 从函数调用到回调函数或监听函数参数的数据流在图 9 中的数据流③得到了验证; 从函数名到函数调用的数据流在图 9 中的数据流②得到了验证.

(2) 本文方法污点分析模块与现有小程序污点分析工具有效性对比测试

本文将本文方法污点分析模块与现有的小程序污点分析工具 TaintMini[4]和 Wang 等人[5]的方法对上述小程序独有的数据流进行追踪分析, 并对随机选取的 50 个真实环境的小程序进行对比测试, 分析本文根据小程序数据流特点设计污点分析规则对敏感行为发现能力的增益, 测试的结果见表 2. 其中, √表示可追踪; ×表示无法追踪.

表 2 MiniChecker 污点分析模块与现有小程序污点分析工具有效性对比表

| 类型 | 子类型 | TaintMini[4] | Wang等人[5] | MiniChecker |
|---|---|---|---|---|
| 回调函数类接口 | 匿名函数 | √ | √ | √ |
| | 函数变量 | × | √ | √ |
| 监听函数类接口 | 匿名函数 | √ | × | √ |
| | 函数变量 | × | × | √ |
| 管理器类接口 | 直接调用内部方法 | × | × | √ |
| | 变量调用内部方法 | × | × | √ |
| 默认污点分析规则无法追踪的4种数据流 | 1) 从对象变量的属性值到对象变量自身的数据流 | × | √ | √ |
| | 2) 从函数参数到函数调用的数据流 | √ | √ | √ |
| | 3) 从函数调用到回调函数或监听函数参数的数据流 | × | × | √ |
| | 4) 从函数名到函数调用的数据流 | × | √ | √ |
| 在50个被测试小程序中发现的敏感行为数量 | | 332 | 1 045 | 1 533 |

从表 2 中可以看出, TaintMini[4]只考虑了匿名函数的回调函数或监听函数传入情况, 难以追踪以函数变量传入的情况, 这导致 TaintMini[4]追踪的覆盖面不足且极易被绕过检查; Wang 等人[5]的方法虽然考虑了函数变量, 但却不支持对监听函数类接口的追踪. 还发现现有的两种方法都无法追踪管理器内部方法的调用, 这可能会影响对录音、相机、本地文件等敏感资源使用的代码分析. 与此同时, TaintMini[4]对小程序代码常见的 4 种数据流 (即表中“默认污点分析规则无法追踪的四种数据流”) 的追踪效果也较为薄弱, Wang 等人[5]的方法也难以追踪从函数调用到回调函数或监听函数参数的数据流, 这导致难以分析函数中嵌套函数的代码行为. 这使得本文方法在所测试的小程序中, 分别比 TaintMini[4]和 Wang 等人[5]的方法多发现了 361.75% 和 46.70% 个敏感行为.

### 3.3 消融研究: 污点分析和语义分析模块不同参数选择的影响

本节通过消融研究分别评估污点分析和语义分析模块不同参数选择对小程序敏感代码行为分析的影响.

(1) 污点分析模块对敏感行为发现的影响

为测试污点分析模块对提取隐私数据收集接口调用的代码行为的作用, 本研究还随机选取了 50 个小程序, 对比使用污点分析后数据流进行语义分析和直接使用接口调用语句进行语义分析对代码语义分析的差别. 经测试, 选取的 50 个小程序直接使用敏感接口调用语句进行语义分析得到了 928 个代码行为, 其中仅有 440 个有效敏感代码行为, 其余 488 个代码行为只描述了敏感数据收集行为而未描述其收集后的使用目的; 而对这 50 个小程序经污点分析可以直接追踪到对敏感数据的使用行为, 之后进行语义分析得到了 1 533 个有效敏感代码行为. 说明污点分析模块将所测小程序的敏感行为发现能力提升了 248.4%, 证明污点分析模块可以有效提升小程序敏感代码行为发现能力.

(2) 不同代码语义分析模型对小程序代码语义分析能力的对比

本节测试 ProphetNet-Code[23]作为语义分析模型进行代码语义提取的效果, 并选用 CodeBERT 模型[27]作为基准模型用于对比, 测试所用的代码片段来自真实环境小程序, 部分对比结果见表 3. 从表 3 中可以看出 ProphetNet-Code 比 CodeBERT 对于代码片段的信息提取能力更强, 比如对序号 1 的代码片段, ProphetNet-Code 可以从代码中的“url”字段分析出发送请求目的的关键词“distance”; 同样地, 对于序号 2 的代码片段, ProphetNet-Code 除了可以分析出创建 camera context 之外, 还分析出了会执行上传. 因此本文认为 ProphetNet-Code 可以更好地捕捉并解释小程序代码片段中关于敏感数据的行为. 表 3 中, 加粗字段表示执行敏感行为的代码及语义分析提取的内容.

表 3 CodeBERT 与 ProphetNet-Code 对小程序代码片段分析对比

| 序号 | 污点源接口 | 代码片段 | CodeBERT | ProphetNet-Code |
|---|---|---|---|---|
| 1 | wx.getLocation | wx.**request**(<br>u.**buildWxRequestConfig**(t, {<br>url:<br>"https://apis.map.qq.com/ws/distance/v1/",<br>data: e, })<br>) | HTTP request | Request the WX **distance** |
| 2 | wx.createCameraContext | wx.**createCameraContext**().**takePhoto**({<br>**success**: function (t) {<br>(e.takePhone =),<br>(e.identing = !0),<br>(e.photoUrl = t.tempImagePath),<br>e.**uploadImgToQiNiu**(e.photoUrl),<br>e.**$apply**();<br>},<br>}); | create a photo context | Creates a new camera context and **upload** it to the QiNiu |

### 3.4 小程序敏感接口使用频率与代码行为统计分析

本研究基于 MiniCrawler[26]收集的小程序包数据集进行测试. 经逆向解包, 共得到 12 872 个小程序的有效源代码. 下面将基于这些小程序分别分析敏感接口的使用频率和使用行为.

(1) 敏感接口使用频率分析

在隐私数据收集接口方面, 获取微信昵称、头像、性别等个人信息的接口“wx.getUserInfo”被 65.2% 的小程序调用, 是小程序最常调用的接口; 此外 51.1% 的小程序调用了获取用户位置信息的接口“wx.getLocation”, 50.7% 的小程序调用了收集用户选中照片或视频的接口“wx.chooseImage”, 以上 3 个调用频率最高的接口均涉及用户隐私, 一旦泄露将严重影响用户隐私安全.

在隐私数据写入接口方面, 小程序调用频率最高的两个接口分别是保存照片到相册的接口“wx.saveImageToPhotosAlbum”, 占比 35.1%, 和向剪切板写入数据的接口“wx.setClipboardData”, 占比 30.9%. 虽然小程序恶意调

用隐私数据写入接口不会直接泄露用户隐私, 但也将对用户正常使用带来干扰, 如保存大量照片占用磁盘空间等.

(2) 敏感接口使用行为分析

为了直观地统计分析小程序调用敏感接口的行为, 本文对上述敏感接口进行行为语义分析并聚类. 选用可以表示两个句子中相同单词或单词序列的指标 $B_{LEU}$(BiLingual evaluation understudy)[28], $B_{LEU}=BP\cdot\exp\left(\sum_{n=1}^{N}w_n\log P_n\right)$, 来计算两个代码语义间的距离, 其中 $BP$ 是一个惩罚因子, 可以防止一句话的长度过短而导致距离计算倾向于短单词序列; $N$ 表示考虑的单词序列的最大长度, 本测试中设置为 2; $w_n$ 表示为长度为 $n$ 的单词序列重要性赋予的权重, 本测试中将其设置为均匀权重; $p_n$ 表示长度为 $n$ 的单词序列相同的概率. 聚类算法采用不需要预先指定簇数量的自下而上的层次聚类法, 经数据分析选定簇距离边界为 2 000.

基于上述聚类算法, 对接口“wx.getUserInfo”“wx.getLocation”和接口“wx.chooseImage”的行为语义分析结果见图 10, 其中每个矩形均表示一个接口的使用行为的次数. 从图 10 中可以看出接口调用最常见的行为是用于前端显示, 此外“存储”“发至服务器”“重定向页面”等也是这些敏感接口被调用的主要行为, 而这些行为有潜在的泄露用户隐私的风险, 需要小程序开发者在隐私声明中进行详细解释.

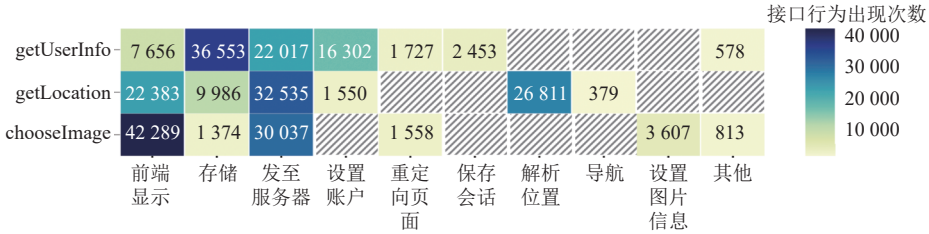



图 10 对部分敏感接口的行为聚类分析热力图

### 3.5 代码与隐私声明不一致的小程序案例分析

本文方法发现了真实环境中存在代码与隐私声明不一致的小程序. 以某体育运动服务小程序 A 为例, 其可以定位用户周边的体育场馆, 并提供多种体育运动服务. 使用本文方法提取该小程序的代码行为, 将代码行为与隐私声明进行对比, 得到的对比结果见表 4. 表 4 中, 对于存在阐述缺失的操作, 表中将未阐述的代码行为加粗显示.

表 4 小程序 A 代码行为与隐私声明对比表

| 敏感数据操作 | 隐私声明的阐述 | MiniChecker分析的代码行为 | 结论 |
|---|---|---|---|
| 收集你的位置信息 | 提供附近距离最近的场馆 | Fetch there geocode;<br>Get the weather information;<br>Get the geo code from amap;<br>Fetch the place around;<br>Set the user location | ☑ |
| 收集你的微信运动步数 | 个人运动记录 | (有相关接口调用, 但无有效sink) | □ |
| 访问你的摄像头 | 拍摄照片 | Creates a new camera context;<br>Creates a new camera context and **upload it to the Qi Niu** | ☒ |
| 收集你的微信昵称、头像 | 用户登录 | (无相关接口调用) | 🖂 |
| 使用你的相册(仅写入)权限 | 办卡 | Save image to Photos Album | □ |
| 收集你的手机号 | 手机号授权登录 | (无相关接口调用) | 🖂 |
| 访问你的麦克风 | 联系场馆 | (无相关接口调用) | 🖂 |
| 收集你选中的照片或视频信息 | 上传照片 | (有相关接口调用, 但无有效sink) | □ |
| 获取你选择的位置信息 | 提供附近距离最近的场馆 | (无相关接口调用) | 🖂 |
| 读取你的剪切板 | 复制订单号 | Set the clipboard data | ☑ |
| 收集你选中的文件 | NA | **Get the filesystem manager** | ☒ |

注: ☑: 代码与隐私声明一致; ☒: 隐私声明存在阐述缺失; 🖂: 隐私声明存在多余阐述; □: 无法仅基于MiniChecker输出判断.

本文方法发现了隐私声明阐述缺失的情况, 如“访问你的摄像头”敏感操作的代码行为包含有上传操作, 但在隐私声明中并未阐述, 且还有调用了相关接口但未在隐私声明中阐述的敏感操作. 本文方法也可以发现多余的隐私声明阐述, 比如隐私声明中阐述“为了手机号授权登录, 开发者将在获取你的明示同意后, 收集你的手机号”, 但该小程序的代码中未调用收集用户手机号的相关接口.

此外, 分析中发现本文方法得到的代码行为可能与隐私声明的阐述存在一定粒度和维度差距, 这种情况对于隐私数据写入接口更加明显, 比如表中的“使用你的相册 (仅写入) 权限”对应的代码行为只是保存照片到相册, 但是隐私声明中介绍的目的是“办卡”. 这种情况的代码与隐私一致性分析需要结合小程序的功能和代码上下文等进行综合分析, 很难只凭借自动化的代码片段分析来判断一致性. 因此本研究只将代码语义分析实现自动化, 更加灵活地帮助分析人员结合实际情况去进行安全检测, 更好保护用户隐私安全.

## 4 总结与展望

针对小程序的代码与隐私声明不一致可能导致的用户隐私泄露问题, 本文提出了一种基于语义分析的小程序代码与隐私声明一致性检测方法. 该方法以小程序代码包为输入, 经运行时数据收集、逆向解包后提取小程序代码和隐私声明文件. 之后将小程序敏感接口分为隐私数据收集接口和隐私数据写入接口, 设计污点分析模块来追踪隐私数据流向, 并使用代码语言处理模型分析得到自然语言表征的代码语义. 基于代码行为语义和提取的隐私声明阐述可辅助人工分析二者的一致性. 测试发现, 其污点分析模块可对隐私数据收集接口的 3 种不同的数据返回方式进行追踪分析, 并可对本文发现的小程序代码常见的 4 种数据流进行追踪分析, 覆盖率优于同类的污点分析方法. 本文对小程序经常调用的敏感接口和使用这些接口的代码行为进行统计分析, 并围绕发现的存在代码与隐私声明不一致的真实环境小程序进行了案例分析.

本文所提出的一致性检测方法主要关注微信小程序, 在后续研究中, 该方法可在对小程序敏感接口定位与污点分析规则调整的基础上进一步拓展至运行在其他宿主应用中的小程序. 此外, 也可考虑自动化实现代码行为和隐私声明的一致性判别 (例如, 机器学习算法、自然语言处理等). 本文方法也可扩展至小程序以外的平台, 基于目标平台的代码特点定制污点分析规则, 提取数据流, 再进行代码与隐私声明一致性分析.

**References:**

[1] Wang Y, Fan M, Tao JJ, Lei JY, Jin WX, Han DQ, Liu T. Compliance detection method for mobile application privacy policy statement. Ruan Jian Xue Bao/Journal of Software, 2024, 35(8): 3668–3683 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/7121.htm [doi: 10.13328/j.cnki.jos.007121]

[2] Tan ZY, Song W. PTPDroid: Detecting violated user privacy disclosures to third-parties of Android apps. In: Proc. of the 45th Int’l Conf. on Software Engineering (ICSE). Melbourne: IEEE, 2023. 473–485. [doi: 10.1109/ICSE48619.2023.00050]

[3] Information and Communication Administration Bureau of the Ministry of Industry and Information Technology of the People’s Republic of China. APP (SDK) Circular on Acts of Infringing on the Rights and Interests of Users (Batch 3 of 2023, Total Batch 29). 2023 (in Chinese). https://wap.miit.gov.cn/jgsj/xgj/fwjd/art/2023/art_c62fe7c7fa2446b69b8906f8be75af9e.html

[4] Wang C, Ko R, Zhang Y, Yang YQ, Lin ZQ. Taintmini: Detecting flow of sensitive data in mini-programs with static taint analysis. In: Proc. of the 45th Int’l Conf. on Software Engineering (ICSE). Melbourne: IEEE, 2023. 932–944. [doi: 10.1109/ICSE48619.2023.00086]

[5] Wang Y, Fan M, Liu JF, Tao JJ, Jin WX, Wang HJ, Xiong Q, Liu T. Do as you say: Consistency detection of data practice in program code and privacy policy in mini-App. IEEE Trans. on Software Engineering, 2024, 50(12): 3225–3248. [doi: 10.1109/TSE.2024.3479288]

[6] Yu L, Luo XP, Liu XL, Zhang T. Can we trust the privacy policies of Android Apps? In: Proc. of the 46th Annual IEEE/IFIP Int’l Conf. on Dependable Systems and Networks (DSN). Toulouse: IEEE, 2016. 538–549. [doi: 10.1109/DSN.2016.55]

[7] Zimmeck S, Wang ZQ, Zou LY, Iyengar R, Liu B, Schaub F, Wilson S, Sadeh N, Bellovin SM, Reidenberg J. Automated analysis of privacy requirements for mobile Apps. In: Proc. of the 2017 Network and Distributed System Security (NDSS) Symp. San Dieg: ISOC, 2017. [doi: 10.14722/ndss.2017.23034]

[8] Andow B, Mahmud SY, Whitaker J, Enck W, Reaves B, Singh K, Egelman S. Actions speak louder than words: Entity-sensitive privacy policy and data flow analysis with PoliCheck. In: Proc. of the 29th USENIX Conf. on Security Symp. Berkeley: USENIX Association, 2020. 985–1002.

[9] Wang R, Feng DG, Yang Y, Su PR. Semantics-based malware behavior signature extraction and detection method. Ruan Jian Xue Bao/Journal of Software, 2012, 23(2): 378–393 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/3953.htm [doi: 10.3724/SP.J.1001.2012.03953]

[10] Wang L, Zhou Q, He DJ, Li L, Feng XB. Multi-source taint analysis technique for privacy leak detection of Android apps. Ruan Jian Xue Bao/Journal of Software, 2019, 30(2): 211–230 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/5581.htm [doi: 10.13328/j.cnki.jos.005581]

[11] Kristensen EK, White G, Larsen RW, Pedersen MV. CodeQL: The libraries and queries that power security researchers around the world, as well as code scanning in GitHub advanced security. 2023. https://github.com/github/codeql

[12] Slavin R, Wang XY, Hosseini MB, Hester J, Krishnan R, Bhatia J, Breaux TD, Niu JW. Toward a framework for detecting privacy policy violations in Android application code. In: Proc. of the 38th Int'l Conf. on Software Engineering. Austin: ACM, 2016. 25–36. [doi: 10.1145/2884781.2884855]

[13] Arzt S, Rasthofer S, Fritz C, Bodden E, Bartel A, Klein J, Le Traon Y, Octeau D, McDaniel P. FlowDroid: Precise context, flow, field, object-sensitive and lifecycle-aware taint analysis for Android Apps. ACM SIGPLAN Notices, 2014, 49(6): 259–269. [doi: 10.1145/2666356.2594299]

[14] Androguard: Reverse engineering and pentesting for Android applications. 2024. https://github.com/androguard/androguard

[15] AppCensus Inc. AppCensus-Mobile App Privacy Assurance. 2024. https://appcensus.io/

[16] Andow B, Mahmud SY, Wang WY, Whitaker J, Enck W, Reaves B, Singh K, Xie T. PolicyLint: Investigating internal privacy policy contradictions on google play. In: Proc. of the 28th USENIX Conf. on Security Symp. Santa Clara: USENIX Association, 2019. 585–602.

[17] Zhang XH, Wang Y, Zhang X, Huang ZQ, Zhang L, Yang M. Understanding privacy over-collection in wechat sub-app ecosystem. arxiv:2306.08391, 2023.

[18] Liu Y, Xie JH, Yang JB, Guo SY, Deng YT, Li SQ, Wu YC, Liu YP. Industry practice of javascript dynamic analysis on WeChat mini-programs. In: Proc. of the 35th IEEE/ACM Int'l Conf. on Automated Software Engineering (ASE). Virtual Event: ACM, 2020. 1189–1193. [doi: 10.1145/3324884.3421842]

[19] WeChat Official Document. Privacy protection guidelines for small program users. 2021 (in Chinese). https://developers.weixin.qq.com/miniprogram/dev/framework/user-privacy/miniprogram-intro.html

[20] Android Open Source Project. Android debug bridge (ADB). 2023. https://source.android.google.cn/docs/setup/build/adb

[21] r3x5ur. unveilr. 2023. https://github.com/r3x5ur/unveilr

[22] Qi WZ, Gong YY, Yan Y, Xu C, Yao BL, Zhou B, Cheng B, Jiang DX, Chen JS, Zhang RF, Li HQ, Duan N. ProphetNet-X: Large-Scale pre-training models for English, Chinese, multi-lingual, dialog, and code generation. In: Proc. of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th Int'l Joint Conf. on Natural Language Processing: System Demonstrations. Online: ACL, 2021. 232–239. [doi: 10.18653/v1/2021.acl-demo.28]

[23] Yan Y, Gou Z, Zhen W, Liu D. ProphetNet-Code: A Research Project for Natural Language Generation, Containing the Official Implementations By MSRA NLC Team. 2023. https://github.com/microsoft/ProphetNet/tree/master/ProphetNet/ProphetNet_Code

[24] Lu S, Guo DY, Ren S, Huang JJ, Svyatkovskiy A, Blanco A, Clement C, Drain D, Jiang DX, Tang DY, Li G, Zhou LD, Shou LJ, Zhou L, Tufano M, Gong M, Zhou M, Duan N, Sundaresan N, Deng SK, Fu SY, Liu SJ. CodeXGLUE: A machine learning benchmark dataset for code understanding and generation. arXiv:2102.04664, 2021.

[25] Microsoft. CodeXGLUE Leaderboard. 2023. https://microsoft.github.io/CodeXGLUE/

[26] Zhang Y, Turkistani B, Yang AY, Zuo CS, Lin ZQ. A measurement study of wechat mini-apps. In: Abstract Proc. of the 2021 ACM SIGMETRICS/Int'l Conf. on Measurement and Modeling of Computer Systems. Virtual Event: ACM, 2021. 19–20. [doi: 10.1145/3410220.3460106]

[27] Feng ZY, Guo DY, Tang DY, Duan N, Feng XC, Gong M, Shou LJ, Qin B, Liu T, Jiang DX, Zhou M. CodeBERT: A pre-trained model for programming and natural languages. In: Proc. of the 2020 Findings of the Association for Computational Linguistics: EMNLP 2020. 2020. 1536–1547. [doi: 10.18653/v1/2020.findings-emnlp.139]

[28] Papineni K, Roukos S, Ward T, Zhu WJ. Bleu: A method for automatic evaluation of machine translation. In: Proc. of the 40th Annual Meeting on Association for Computational Linguistics (ACL). Philadelphia: ACL, 2002. 311–318. [doi: 10.3115/1073083.1073135]

## 附中文参考文献:

[1] 王寅, 范铭, 陶俊杰, 雷靖薏, 晋武侠, 韩德强, 刘烃. 移动应用隐私权声明内容合规性检验方法. 软件学报, 2024, 35(8): 3668–3683.

http://www.jos.org.cn/1000-9825/7121.htm [doi: 10.13328/j.cnki.jos.007121]

[3] 工业和信息化部信息通信管理局. 关于侵害用户权益行为的 APP(SDK) 通报 (2023 年第 3 批, 总第 29 批). 2023. https://wap.miit.gov.cn/jgsj/xgj/fwjd/art/2023/art_c62fe7c7fa2446b69b8906f8be75af9e.html

[9] 王蕊, 冯登国, 杨轶, 苏璞睿. 基于语义的恶意代码行为特征提取及检测方法. 软件学报, 2012, 23(2): 378–393. http://www.jos.org.cn/1000-9825/3953.htm [doi: 10.3724/SP.J.1001.2012.03953]

[10] 王蕾, 周卿, 何冬杰, 李炼, 冯晓兵. 面向 Android 应用隐私泄露检测的多源污点分析技术. 软件学报, 2019, 30(2): 211–230. http://www.jos.org.cn/1000-9825/5581.htm [doi: 10.13328/j.cnki.jos.005581]

[19] 微信官方文档. 小程序用户隐私保护指引内容介绍. 2021. https://developers.weixin.qq.com/miniprogram/dev/framework/user-privacy/miniprogram-intro.html

**刘力沛**(1999－), 男, 硕士生, 主要研究领域为移动安全.

**吕雨松**(1999－), 男, 硕士生, 主要研究领域为物联网安全.

**毛剑**(1978－), 女, 博士, 教授, 博士生导师, 主要研究领域为物联网安全, Web 安全, 移动安全, 社交网络安全.

**李嘉维**(1997－), 男, 博士生, 主要研究领域为移动安全, Web 安全, 物联网安全.

**林其箫**(1995－), 男, 博士生, 主要研究领域为物联网安全, Web 安全, 移动安全.

**刘建伟**(1964－), 男, 博士, 教授, 博士生导师, 主要研究领域为密码学, 5G 网络安全, 移动通信网络安全.