计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
2024,33(11):224−236 [doi: 10.15888/j.cnki.csa.009642] [CSTR: 32024.14.csa.009642]

E-mail: csa@iscas.ac.cn
http://www.c-s-a.org.cn
Tel: +86-10-62661041

小程序敏感数据收集行为检测①

花　楠, 杨哲慜

(复旦大学 计算机科学技术学院, 上海 200433)
通信作者: 花　楠, E-mail: huan19@fudan.edu.cn

摘　要: 小程序近年来被广泛应用, 因承载了大量的敏感用户数据而引发了广泛的隐私安全担忧. 现有的面向传统移动应用的隐私安全分析方法无法直接应用于小程序中. 一方面, 现有方法难以有效分析小程序闭源框架行为带来的隐私流转以及 JavaScript 闭包特性带来的跨作用域隐私流转, 造成分析结果的缺失. 另一方面, 小程序动态加载子包的机制导致不完整的分析范围, 进一步造成分析结果的缺失. 为此本文提出了动静态混合的小程序隐私收集行为分析方法. 首先, 该方法为小程序中的不同单元边界构建了基于控制流或数据依赖关系的数据传播路径, 即小程序隐私传播流图. 进一步地, 该方法通过学习并迁移传统移动应用端界面设计知识, 并利用 UI 事件与页面转换行为之间的控制流关联作为指引, 有效地对小程序界面进行探索, 从而触发子包加载过程. 相应的子包代码经分析后与已有分析结果融合, 形成更为全面的小程序隐私传播流图. 本文基于小程序隐私传播流图实现了对小程序内敏感数据的追踪. 本文基于上述方法实现了小程序隐私收集行为分析工具 MiniSafe. 评估结果表明, MiniSafe 在精确率与召回率上分别达到了 90.4% 与 87.4%, 均优于现有工作. 同时, MiniSafe 平均在每个小程序中检测出 7 项敏感数据收集行为, 通过考虑小程序子包中的敏感数据收集行为使整体检测效果提升了 42.9%, 具有较好的检测效果与实际可用性.

关键词: 小程序; 敏感数据收集; 数据流分析; 小程序隐私传播流图; UI 自动化探索

引用格式: 花楠,杨哲慜.小程序敏感数据收集行为检测.计算机系统应用,2024,33(11):224–236. http://www.c-s-a.org.cn/1003-3254/9642.html

Detection for Sensitive Data Collection Behaviors in Mini-programs

HUA Nan, YANG Zhe-Min

(School of Computer Science, Fudan University, Shanghai 200433, China)

Abstract: Mini-programs have been widely used in recent years, causing widespread privacy and security concerns for carrying a large amount of sensitive user data. Existing privacy and security analysis techniques for traditional mobile applications cannot be directly applied to mini-programs. On the one hand, it is difficult for existing methods to effectively analyze the privacy transfer caused by the closed-source mini-program framework and the cross-scope privacy transfer caused by the JavaScript closures, resulting in a lack of analysis results. On the other hand, the mechanism of dynamic sub-package loading leads to incomplete analysis scope, further resulting in a lack of analysis results. This study proposes a hybrid dynamic/static method for analyzing the privacy collection behaviors in mini-programs. First, this method constructs a data propagation path based on either control flow or data dependency for different unit boundaries in the mini-programs, namely the mini-program privacy propagation flow graph. Furthermore, this method effectively explores the mini-program UI by learning and transferring traditional mobile application UI design knowledge, and using the control flow association between UI events and page transition information as a guide, thereby triggering the sub-package loading process. The corresponding sub-package code is analyzed and integrated with existing analysis results to

① 基金项目: 工信部专项 (TC220H079)
收稿时间: 2024-04-02; 修改时间: 2024-04-29; 采用时间: 2024-05-09; csa 在线出版时间: 2024-09-24
CNKI 网络首发时间: 2024-09-25

form a more comprehensive mini-program privacy propagation flow graph. This study implements the tracking of sensitive data in mini-programs through the privacy propagation flow graph. Based on the above method, this study implements MiniSafe, a privacy collection behavior analysis tool for mini-programs. The evaluation results show that MiniSafe achieves 90.4% and 87.4% in precision and recall respectively, both of which outperform existing work. MiniSafe detects an average of 7 sensitive data collection behaviors in each mini-program. By considering sensitive data collection behaviors in mini-program sub-packages, the overall detection number has increased by 42.9%, demonstrating good detection performance and practical usability.

Key words: mini-program; sensitive data collection; data flow analysis; mini-program privacy propagation graph; automated UI exploration

近年来, 小程序凭借易于开发与使用的优势吸引了大量开发者与用户, 成为当下最流行的应用形态. 小程序被广泛应用于人们日常生活的各个方面, 例如用户可以通过小程序完成在线购物支付、医院挂号、政务服务等. 据统计[1], 截至 2022 年末, 互联网小程序数量已超 780 万, 每日活跃用户数量突破 8 亿.

随着近年来国内《中华人民共和国个人信息保护法》等相关法律法规的颁布以及用户隐私保护意识的提升, 保护用户敏感数据变得愈发重要. 小程序承载了大量的用户敏感数据, 进而引发了广泛的隐私安全担忧. 一项针对国内银行类小程序的安全调查[2]显示, 超过 60% 的小程序对用户敏感数据的 保护措施不到位, 进而导致存在隐私安全隐患, 敲响了小程序隐私安全的警钟. 尽管小程序平台积极地采取相应措施以保障用户隐私安全, 然而这些方法存在较大的局限性. 例如, 人工审核制度难以发现深层代码中的敏感数据泄露行为. 因而有效的小程序行为分析方法对保障用户隐私安全有重要意义.

隐私收集是隐私生命周期管理中一个重要环节, 而对用户敏感数据不合理的收集行为是造成隐私安全隐患的一项重要问题. 先前学术界对该问题已经有所关注, 提出了一系列分析技术, 并将其应用于隐私泄露、隐私过度收集等问题的检测. 然而先前研究工作提出的分析方法往往面向传统的移动应用, 而无法直接应用在小程序相关问题的检测上. 面对该现状, 本文旨在提出一种适用于小程序的分析方案, 以实现小程序敏感数据收集行为检测. 通过对小程序工作原理的深入分析, 本文提出了动静态混合的分析方案: 静态分析过程结合静态程序分析技术与小程序代码加载流程等信息, 为不同单元边界生成基于控制流或数据依赖关系的数据传播路径, 本文称之为小程序隐私传播流图. 而动态分析过程通过对小程序用户界面 (user interface, UI) 进行探索, 有效地触发了小程序子包的加载过程. 新加载的子包代码经过分析后会与已有的数据传播路径结合, 形成更全面的小程序隐私传播流图. 对敏感数据的追踪可以通过分析小程序隐私传播流图中的控制流或数据依赖信息来实现. 实验结果证明了本文提出的方法能够有效地检测真实世界小程序中的敏感数据收集行为, 从而实现更好的小程序平台隐私保护.

1 相关工作

小程序是传统移动应用的延伸, 其在架构与机制设计方面一定程度上参考了传统移动应用, 因而小程序在相关问题的研究上可以借鉴传统移动应用的研究思路. 本节将分别介绍移动平台隐私收集行为检测方面的研究进展以及小程序在隐私安全方面的研究进展.

1.1 移动平台敏感数据收集问题研究

当前对移动平台应用程序的研究工作主要集中在发现安全和隐私问题上[3]. 敏感数据收集行为检测作为一种分析用户隐私数据泄露情况的重要手段, 获得了学术界的广泛关注. 当前与敏感数据收集行为检测有关的研究主要面向移动端应用, 包括敏感数据的识别与敏感数据追踪两方面.

在敏感数据识别方面, 早期的研究工作主要关注了操作系统集中管控的敏感数据. 研究者通常基于特定的系统 API 来识别此类敏感数据. 如安卓系统在 LocationManager 类下提供了 getLastKnownLocation 方法用于获取用户最近地理位置信息, 相关研究通过识别安卓应用程序中对该方法的调用来识别敏感数据源.

Nan 等人[4]以及 Huang 等人[5]揭示了移动应用中另一种常见且重要的敏感数据类型, 即用户输入型敏感数据. 相较于能通过 API 调用来识别的系统集中管控的敏感数据, 用户输入型敏感数据需要通过分析输入数据的上下文以及相关语义信息来识别. 相关工作通过对与用户输入组件相关的静态文本语义分析, 有效识别了该类型的敏感数据. Nan 等人[6]进一步关注了移动应用代码中的语义信息, 结合自然语言分析、机器学习等技术, 从代码中识别出与敏感数据相关的变量、常量、方法名等, 进一步扩展了敏感数据识别范围.

在敏感数据追踪方面, 静态污点分析作为一种常用且有效的分析技术, 被广泛用于对安卓平台的研究工作中. 早期的研究[7,8]利用静态污点分析方法检测移动平台中潜在的隐私数据泄露等问题. 然而因为静态分析技术的限制, 这些研究工作的方法存在误报率高、分析开销大等问题. Arzt 等人[9]首次提出了面向安卓移动应用组件精准的生命周期建模方法, 能够有效处理安卓应用中各种回调函数对污点分析产生的影响. 其基于 IFDS (interprocedural finite distributive subset) 框架实现的污点分析算法能够实现上下文、流、域以及对象敏感的精度, 具有较高准确性. 此外, 该研究工作还提出了需求驱动的别名分析技术, 在精度与效率方面取得了较好的平衡. 后续的研究工作关注了安卓应用污点分析的不同维度: 一些研究工作关注了安卓系统中的特殊机制对污点分析的影响, 如 Li 等人[10]与 Wei 等人[11]提出了对安卓系统组件间通信分析的方法, 而 Gordon 等人[12]提出了精确分析桩技术, 解决了安卓系统框架中一些基于非 Java 环境的运行时数据流语义对污点分析的影响. 另外一些工作[13–16]关注了安卓应用污点分析的效率问题, 通过不同的方法有效减少了静态分析过程的时间开销. 此外, 还有一些研究工作[17,18]关注了跨越语言架构对移动应用污点分析的影响, 并提出了相应的解决方案. 这些研究工作提出的方法极大提升了安卓应用污点分析技术的能力.

总体而言, 尽管已有的敏感数据收集行为检测思路对小程序上相关问题的研究有所启发, 但先前工作提出的许多分析技术仅考虑了移动平台特性对数据流产生的影响, 而无法将其直接应用于分析或追踪小程序内的敏感数据流.

1.2 小程序隐私安全问题研究

小程序作为近年来新兴的流行移动平台, 其特有的机制给其分析带来了独有挑战, 成为学术界研究的热点方向.

最早关于小程序安全问题的研究工作往往是对小程序安全机制或生态的评估, 或是某一种具体的安全漏洞. Lu 等人[19]主要关注了小程序生态中的资源管理风险问题, 揭示了小程序中一种允许攻击者提升权限进而访问相机、相册、麦克风等重要资源的漏洞. Zhang 等人[20]关注了小程序身份混淆问题导致的越权风险, 其研究对象主要是小程序宿主应用, 因此分析方法主要也是针对安卓应用, 并未涉及对小程序代码行为的分析. Zhang 等人[21]通过对微信应用的逆向实现了微信小程序代码包的爬虫工具, 并通过分析大量微信小程序代码, 统计了微信小程序的资源消耗、API 与第三方库使用、代码混淆率、小程序分类以及评分等情况. Yang 等人[22]关注了跨小程序通信伪造漏洞, 并通过分析小程序抽象语法树代码中的静态数据流, 实现对该类型漏洞的检测. 叶瀚等人[23]提出了融合指针分析和关系图谱的小程序函数调用图构建方法, 能够有效提升现有研究在敏感 API 调用检测方面的不足. 然而该方法仅关注了小程序中函数调用相关的控制流信息, 而忽视了普遍存在的数据流信息. 尽管上述研究关注的问题可能会影响到小程序中的数据安全, 然而这些研究提出的分析方法并非面向小程序的一般数据流分析方法.

Wang 等人[24]首次关注了小程序中的敏感数据追踪问题, 提出了面向小程序的通用污点分析框架 TaintMini. 其提出的通用数据流图 (universal data flow graph) 结构将小程序中各种类型的数据表示为图中的节点, 将数据之间的关系表示为节点之间的边, 基于图搜索的方式实现小程序中污点数据的追踪. 然而该技术分析数据流转行为的精度不足, 其仅追踪了 JavaScript 对象层面的污点数据传播, 而难以区分对象下的不同属性, 由此造成污点数据传播的精确性有限. Li 等人[25]就 TaintMini 存在的问题做出了改进, 提出了新的分析框架 MiniTracker. MiniTracker 提出了一种新的图结构, 称作赋值流图 (assignment flow graph). 该图结构能够区分与追踪 JavaScript 对象下的不同属性, 使得污点传播过程更加精确. 另外该工作考虑了更多 JavaScript 语言机制带来的影响, 如 Promise 机制产生的异步数据流等, 进一步提升了分析效果.

然而本文通过对小程序特性以及先前工作的深入

分析发现, 先前的工作仍然存在较大的改进空间. 一方面, 先前工作对小程序闭源框架行为的不合理的分析方法形成的不完整的控制流信息, 造成了数据流传播分析的中断, 而忽视了从 JavaScript 语言中继承的闭包 (closure) 机制导致了先前研究工作难以克服函数作用域对数据流分析的限制, 造成了跨作用域数据流的中断. 另一方面, 先前研究工作中提出的批量化小程序包收集方法忽视了小程序按需加载的分包机制, 因而难以应用于小程序子包的收集. 这导致了先前研究工作的分析范围仅限于小程序主包, 而忽视了小程序子包中存在的敏感数据收集行为, 由此造成了分析范围不全面.

2 案例分析与核心挑战总结

本节通过一个敏感数据收集行为的完整过程来说明检测该行为所面临的具体挑战. 图 1 展示了一个小程序中涉及敏感数据收集行为的代码, 其中包含了应用模块 (包括其所需的外部模块)、入口页面模块、数据收集页面模块以及数据确认与提交页面模块. 其中应用模块与入口页面模块存在于小程序主包中, 而数据收集页面模块以及数据确认与提交页面模块存在于小程序子包中. 图 1 中左上方淡黄色底的框表示外部模块代码, 淡绿色底的框表示小程序视图层语言 WXML 代码, 无色底的框表示小程序逻辑层语言 JavaScript 代码.

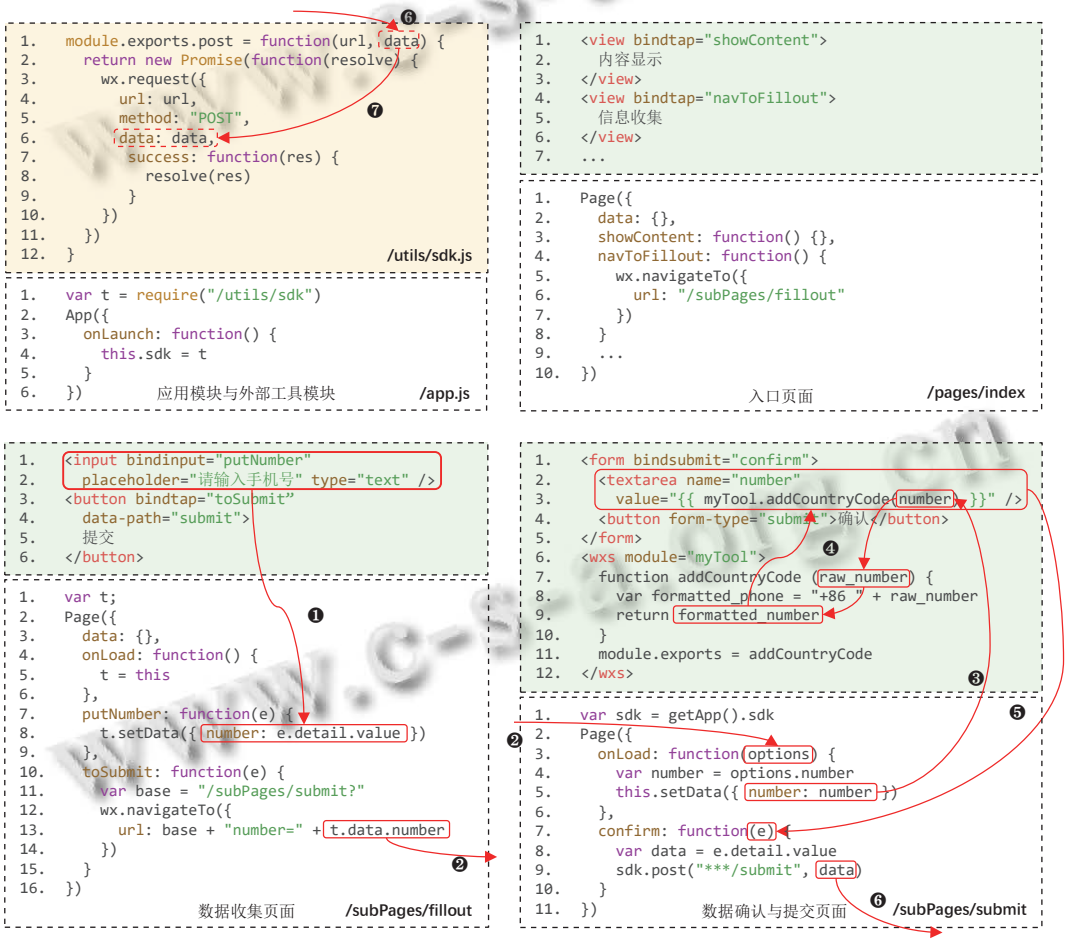

图 1 案例分析

● 小程序启动阶段. 当小程序初次开始运行时, 小程序主包代码会被加载. 首先小程序从文件路径为/app.js 的应用模块代码开始运行. 该应用模块通过 require 关键词引用了外部模块, 而该外部模块中定义了用于发送网络请求的匿名函数. 应用模块引入该外部模块后, 通过调用 APP 函数完成小程序的注册.

● 入口页面运行阶段. 小程序启动阶段完成后, 路径为/page/index 的入口页面被加载. 入口页面中定义

了不同功能的入口. 当用户点击信息收集功能对应的组件后, 当前页面中的 navToFillout 方法被触发, 该方法会通过小程序路由 API 打开路径为/subPages/fillout 的数据收集页面. 由于数据收集页面位于小程序子包中, 因此小程序框架会触发子包的加载. 当子包加载完成后, 小程序会转换到数据收集页面.

● 数据收集页面运行阶段. 数据收集页面中存在一项敏感数据收集行为. 页面最上方红色实线框包围的界面元素表示一个用户输入框, 要求用户向其中输入手机号码, 属于用户输入型敏感数据. 当用户输入数据时, 输入内容通过图中标号为❶的路径由视图层流向了逻辑层. 当用户点击按钮提交之后, toSubmit 函数被调用. 该函数通过路由 API 转换到路径为/subPages/submit 的数据确认与提交页面中, 且数据通过小程序框架以图中标号为❷的路径传输到数据确认与提交页面.

● 数据确认与提交页面运行阶段. 当数据流转到数据确认与提交页面之后, 首先由逻辑层 onLoad 函数的参数接收, 并通过 setData 函数调用, 经图中标号为❸的传播路径由逻辑层流转到了视图层. 数据在视图层中通过图中标号为❹的传播路径进行格式处理后呈现在页面上供用户确认. 当用户点击确认按钮后, 数据通过图中标号为❺的传播路径由视图层流转到了逻辑层, 传入到 confirm 函数中. 数据进一步通过图中标号为❻的传播路径传入到 sdk.post 表示的函数中. 该函数实际调用的是小程序应用模块引用的外部模块中定义的函数. 数据进一步通过图中标号为❼的传播路径由外层作用域流动到内层作用域中, 最终通过小程序网络接口 wx.request 发送至网络, 完成收集流程.

上述分析说明了在小程序敏感数据收集过程中, 数据会跨越不同的代码单元边界流动, 具体体现为跨视图与逻辑层数据流、跨页面数据流、跨函数数据流以及跨作用域数据流. 有效的小程序敏感数据追踪应当能够覆盖这些不同类型的数据流. 然而不同原因导致了小程序代码中敏感数据传播路径的缺失, 使得传统的数据流分析方法难以有效地识别与追踪小程序中的敏感数据, 为检测小程序中的敏感数据收集行为带来了挑战, 具体可以概括为以下两方面.

● 挑战 1: 敏感数据传播路径的缺失使得现有的方法难以有效追踪小程序中的数据.

一方面, 小程序闭源框架控制的不同单元交互行为导致了小程序控制流的中断, 其中包括跨视图与逻辑层交互以及跨页面交互. 由于视图层与逻辑层基于不同编程语言实现, 对跨层数据流的分析需要克服语法差异统一地分析不同语言中的数据流. 对跨页面数据流的分析则需要对小程序代码的预处理准确识别存在数据通信行为的不同页面, 以及页面内的具体函数. 而小程序动态特性与广泛使用的 JavaScript 匿名函数进一步为小程序控制流的识别带来了困难. 小程序中存在多种具体使用方式不同的匿名函数: 一些函数以同步调用的方式使用, 另一些则以异步调用的方式使用. 异步函数又可以进一步分为不同模块的生命周期函数、UI 事件处理函数、异步 API 的回调函数, 由小程序框架运行时调用. 准确识别不同匿名函数的作用能够帮助确定小程序中数据的流向, 对敏感数据的追踪过程十分重要. 然而 JavaScript 支持的一级函数 (first-class function) 特性允许这些匿名函数在小程序中不同空间内流动, 极大地增加了控制流分析的复杂性.

另外一方面, 小程序继承了 JavaScript 中的闭包作用域. 闭包是编程语言中的一个常用概念, 而 JavaScript 对闭包的实现同时允许函数读取与修改外部作用域中变量的值, 在不同作用域之间产生了双向数据流动. 这种数据流动行为在代码中并没有连续的控制流关系, 而是通过基于词法规则的数据依赖关系体现. 然而小程序中存在许多同名的变量, 这种情况在经过混淆技术处理的小程序代码中尤为突出, 这要求跨作用域的数据流分析能够区分不同的变量. 而现有的分析方法往往基于函数内变量的定义与使用关系实现数据流分析, 难以实现这种跨作用域的数据流追踪.

● 挑战 2: 按需加载的分包机制造成目标小程序的代码加载不完全, 进一步导致敏感数据收集行为检测结果的缺失.

为了保障更好的用户体验, 小程序采用了分包机制, 通过限制每个安装包的大小来保证加载速度. 小程序初次运行时仅加载主包, 仅当特定功能被触发之后才会从服务器加载对应的子包. 当前主流的小程序包收集方法[21]仅能够实现小程序主包的收集, 因而现有的关于小程序用户隐私的研究方法仅面向小程序主包, 而往往难以发现并追踪小程序子包中的敏感数据. 由于小程序的子包中往往也存在着影响用户敏感数据安全的行为, 如图 1 案例中敏感数据行为收集行为主要

在子包页面中完成. 尽管先前工作提出了通过主动调用小程序宿主应用内加载小程序子包的函数来实现子包加载, 然而该方法过于依赖人工逆向分析. 小程序宿主应用高度混淆的代码、高频率的版本迭代导致该方法的稳定性与扩展性不够强, 因而难以得到有效应用.

另一种思路则是通过模拟用户与小程序的交互行为来触发小程序子包的加载. UI 自动化探索是学术界常用的触发应用功能的手段, 其通过识别用户界面上的控件并与之产生交互行为来模拟用户对程序的操作. 然而现有的 UI 自动化探索方法与工具无法对小程序进行有效探索, 原因在于其无法稳定地识别小程序 UI 控件. 一方面, 小程序往往采用了与安卓系统不同的 UI 渲染核心 (例如微信使用了 Chromium Webview 内核[26]), 因而传统的移动应用自动化测试工具无法直接应用于小程序. 另一方面, 尽管小程序开发环境提供了 UI 自动化测试接口, 然而其往往只面向开发者自己的应用. 本文在实际测试过程中发现真实小程序可能关闭对小程序 UI 布局的获取接口, 使得分析工具无法稳定地通过该接口获取小程序 UI 布局信息.

3 方法设计与实现

基于第 2 节分析过程不难发现, 面向小程序的敏感数据收集行为检测不仅需要有效分析小程序中跨不同单元边界的数据流, 同时要有效考虑到小程序子包动态加载机制对分析范围的影响. 为此, 本文在深入分析上述挑战所对应的小程序特性基础上, 提出了动静态混合的小程序敏感数据收集行为检测方法. 该方法的整体技术方案如图 2 所示, 其中主要包含了两个重要阶段, 即静态分析过程小程序隐私传播流图构建, 以及动态分析过程小程序 UI 自动化探索. 两个阶段互相依赖: 前一阶段在构建小程序敏感数据传播路径的同时能够为后一阶段的动态探索过程提供指导信息; 而后一阶段触发加载的小程序子包能够进一步完善前一阶段构建的小程序敏感数据传播路径. 其具体说明如下.

● 小程序隐私传播流图构建. 该阶段的作用在于为小程序中不同单元建立基于控制流 (其中隐含了数据流信息) 或者显式数据依赖关系的数据传播路径以方便小程序中敏感数据的追踪, 本文将其称作小程序隐私传播流图. 小程序隐私传播流图的生成由 3 个分析过程构成, 分别是模块控制流分析、页面转换分析以及跨作用域数据依赖分析. 模块控制流分析首先利用程序分析技术以及小程序加载与执行流程信息识别出不同匿名函数的具体作用, 在此基础上为小程序中不同类型的模块构建了内部控制流信息, 服务于模块内部的数据流分析. 页面转换分析在模块控制流分析的基础上, 通过分析页面模块中的路由行为识别出跨页面的数据传播路径. 跨作用域数据依赖分析通过为不同作用域下的同名变量建立联系, 突破了函数作用域对现有数据流分析方法的限制, 使分析过程能够在不同作用域下快速转换, 有效实现了跨作用域的数据流分析.

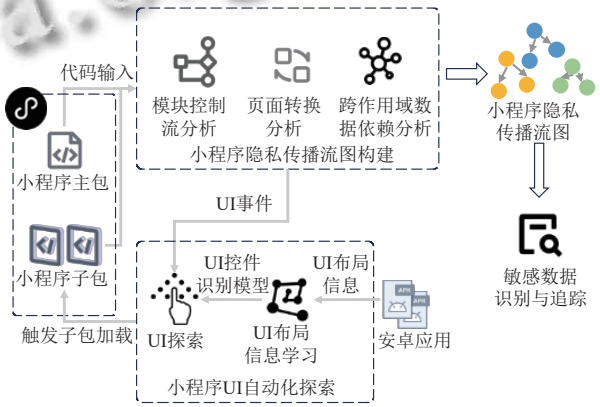

图 2 小程序敏感数据收集行为检测整体技术方案

● 小程序 UI 自动化探索. 该阶段的作用在于利用 UI 自动化探索技术触发小程序子包加载行为, 进而保障分析范围的全面. 本文基于传统移动应用与小程序在布局上具有相似性这一观察, 提出了借鉴传统移动应用的 UI 布局视觉信息来识别小程序 UI 控件的方法, 并将其应用于小程序 UI 的自动化探索. 同时由于小程序子包的加载行为通常伴随着页面的转换行为发生, 而页面的转换行为通常由特定的 UI 事件触发, 该模块利用先前小程序隐私传播流图构建过程中提取的事件与页面转换行为的控制依赖关系指导小程序 UI 探索. 该模块基于这两点方法有效地触发小程序子包的加载, 而子包中的代码则会作为前一阶段分析的输入形成更为全面的小程序隐私传播流图.

3.1 小程序隐私传播流图构建

如前文所述, 面向小程序的敏感数据收集行为检测应当能够覆盖小程序中跨越不同代码单元边界传递的数据流, 而数据传播路径的缺失为数据流分析带来了挑战. 因此本文结合了多种程序分析技术, 通过为小

程序构建包含一般控制流信息与数据依赖关系的小程序隐私传播流图, 以支撑小程序敏感数据收集行为检测. 具体而言, 本文首先结合小程序不同模块的生命周期信息以及小程序框架的控制信息, 构建了融合视图层与逻辑层信息的模块控制流图. 接着本文利用静态程序分析方法识别出所有的页面转换行为以及转换的目标页面, 构建了页面转换图. 最后, 对于分析过程中所有加载的模块, 本文基于 JavaScript 作用域链规则为其中嵌套的作用域构建了数据依赖关系, 形成了跨作用域数据依赖图. 上述 3 种图结构共同构成了小程序隐私传播流图.

3.1.1 模块控制流分析

小程序中存在 3 种模块类型, 分别是应用模块、页面模块与自定义组件模块. 应用模块是小程序执行的入口, 定义了小程序应用的生命周期, 同时提供了一些全局信息初始化功能. 页面是小程序中主要的可视化模块, 负责处理与用户交互的业务逻辑. 自定义组件模块则是小程序对页面中一些常用的组件及其组合, 以及其相关的逻辑进行封装以便复用. 其中页面模块最具代表性与广泛性, 因而本文以页面模块为例详细说明模块控制流图的构建方法, 具体分为逻辑层内部的控制流构建方法, 以及融合视图层的控制流构建方法.

页面模块逻辑层的执行流程可以概括为两个主要的阶段, 分别是环境初始化阶段与生命周期执行阶段. 在环境初始化阶段, 页面模块会完成当前页面所需的信息初始化, 比如定义页面生命周期函数、定义页面 UI 事件处理函数以及引入该页面用到的外部模块等. 在环境初始化阶段的最后, 页面会通过小程序框架提供的 Page 函数来注册该模块. Page 函数接收一个 JavaScript 对象作为参数, 本文将该对象简称为页面注册对象. 页面注册对象中包含了当前页面的初始化数据、生命周期函数、UI 事件回调函数等信息. 页面注册完成之后会转入生命周期执行阶段. 模块的主要逻辑往往在页面生命周期函数与 UI 事件处理函数内定义, 而这些函数属于由小程序框架控制的异步函数, 在模块代码中并没有显式的调用. 由于这些函数涉及许多数据流传播路径, 因而模块控制流图构建过程需要有效反映框架对这些函数的控制. 结合上述页面模块的执行流程信息, 本文方法将构建页面控制流信息的过程分为两阶段, 分别用于识别由框架控制的异步函数与构建页面生命周期.

第 1 阶段分析以页面文件作为入口构建控制流图, 并沿着控制流图查找页面注册函数的调用行为. 分析过程会追踪所有匿名函数定义的流转. 当分析过程识别到页面注册函数 Page 调用后, 基于对函数追踪的结果, 本文通过检查页面注册对象中 onLoad 等属性指向的函数, 识别出页面中生命周期、UI 事件回调函数等重要函数.

而第 2 阶段基于页面生命周期信息为第 1 阶段识别的重要异步函数构建新的控制流关系. 具体而言, 本文通过创建一个虚拟的生命周期入口函数来模拟小程序框架对页面生命周期的控制流, 使其成为页面生命周期执行阶段的入口点. 本文从该虚拟生命周期入口函数调用处开始第 2 次分析. 页面模块在生命周期执行阶段还会调用大量的 API, 其中包括了许多异步的 API. 这些异步 API 会注册异步执行的回调函数, 因此在第 2 阶段本文以虚拟生命周期入口函数调用为起点构建控制流图, 并沿着该控制流图查找所有 API 的调用行为. 对于过程中识别出的所有异步 API 调用, 本文根据页面生命周期模型将其注册的异步回调函数连接到已经构造的页面模块控制流图中. 本文将该虚拟生命周期入口为起点的控制流连接到页面注册函数 Page 调用之后, 初步构建了页面模块控制流图.

另外, 页面作为小程序中一种主要的 UI 视图模块, 和用户之间存在着大量的交互, 而这些交互常常伴随着数据在视图层与逻辑层之间流动. 用户提交表单时, 小程序框架会将用户输入的数据通过参数传递给逻辑层的 UI 事件处理函数. 而当逻辑层的数据更新之后, 可以通过页面注册对象调用 setData 函数, 更新的数据通过该函数的参数传递到视图层中. 本文利用微信小程序本地编译环境的能力将 WXML 文件转化为渲染逻辑, 基于程序切片分析技术从中提取与数据流传播相关的代码片段, 并将其融入先前构建的生命周期中, 形成了完整的页面控制流图. 图 3 展示了图 1 案例中数据确认与提交页面基于上述方法构建的页面控制流图.

3.1.2 页面转换分析

小程序中往往包含多个页面. 每个页面都是一个独立的视图模块, 负责不同的业务逻辑. 不同页面之间可能存在数据交互关系, 而跨页面的数据交互通常与页面转换行为有关. 因此准确分析页面转换关系有助

于实现面向小程序的敏感数据收集行为检测. 本节通过两个阶段构建页面转换图, 即转换行为识别与转换路径分析. 本节以图 1 案例中的数据收集页面为例来说明该分析流程.

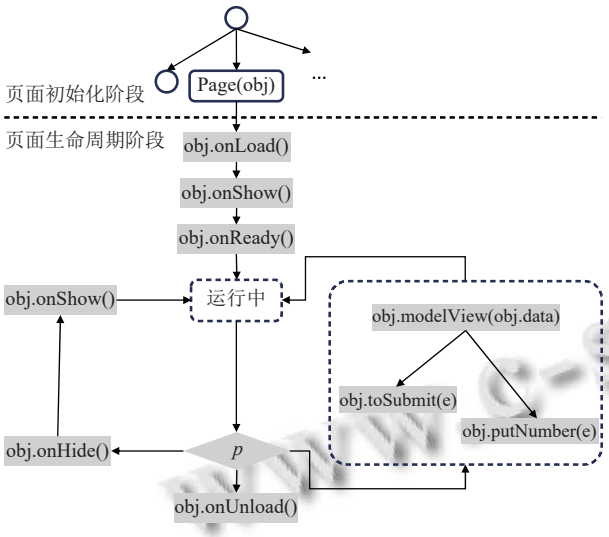

图 3 模块控制流图示例

本文首先基于前一节实现的模块控制流图来分析页面间转换的关系. 由于页面转换行为通常在页面的生命周期过程中发生, 因此本文以先前构建的虚拟的页面生命周期入口函数为起点, 通过模块控制流图来识别页面转换行为. 大多数页面转换行为通过小程序路由 API 实现. 本文通过发现控制流图中这些路由 API 的调用行为来识别页面转换行为. 通过对数据收集页面控制流的追踪能够能识别 toSubmit 函数的调用, 进而识别该函数对 wx.navigateTo 路由 API 的调用.

接着本文结合了常量传播分析、后向数据流分析以及字符串技术实现转换路径分析. 其中常量传播分析用于计算常量字符串拼接后生成的新的常量字符串; 后向数据流分析从路由 API 调用处反向地识别路由路径的生成过程, 直到能够追踪到字符串常量; 而字符串分析则用于进一步处理字符串常量, 识别出其中符合页面模块路径的格式. 在 toSubmit 函数中, 通过对路由 API 参数的识别, 分析过程会查找到其 url 属性的定义. 由于其并非常量, 分析过程接着从该表达式反向地分析其生成过程. 当追踪到第 13 行表达式 base+“number=”+t.data.number 表示的路径拼接行为时, 分析方法通过常量传播信息识别出左侧表达式 base+“number=”的结果为字符串常量“/subPages/submit?number=”. 通过对该字符串进行格式化分析得到符合文件路径格式的字符串“/subPages/submit”, 其对应了数据确认与提交页面, 因此分析过程在数据收集页面和数据确认与提交页面之间添加转换关系.

本文基于上述方法为所有能够识别的页面构建转换关系, 形成了页面转换图. 进一步地, 本文基于页面转换图可以为不同页面间构建数据流传播路径.

3.1.3 跨作用域数据依赖分析

小程序的主要逻辑基于 JavaScript 实现. JavaScript 的闭包机制使得嵌套的内层作用域可以直接访问其外层作用域中的变量, 因而产生了跨作用域的数据流. 本文在实际小程序应用中发现了大量该类型的数据流, 因此实现准确的跨作用域数据流分析对本文检测小程序中的敏感数据收集行为有重要作用.

跨作用域数据流与一般数据流不同, 无法通过函数内与函数间的控制流分析. 一方面, 在 JavaScript 程序中, 不同作用域下常常存在同名变量. 而小程序代码往往经过混淆技术处理, 使得同名变量的现象更为常见. 内层作用域中声明的变量可以覆盖外层作用域已经声明的变量, 这要求跨作用域数据流分析能够有效区分不同作用域下的同名变量是否指向同一处声明. 另一方面, 内部作用域可以同时对外部作用域中的变量进行读取与修改. 当一个作用域下修改了其外层某个作用域声明的变量, 这种影响应当能有效地反映至变量声明所在的作用域. 如果变量声明所在的作用域内部某个作用域读取了该变量的值, 对该变量的影响应当能有效地反映至读取行为所在的作用域. 跨作用域数据流分析能够有效识别这种双向的数据流动行为.

本文通过构建跨作用域数据依赖图解决跨作用域的数据流分析问题. 该方法基于 JavaScript 语言的作用域链机制为不同作用域下的变量建立关系. 具体而言, 作用域链机制遵循一种由内而外的解析方式. 当分析过程在一个作用域中发现了对某个变量的访问行为(包括读取与修改), 其首先会在当前作用域中查找该变量的声明. 如果当前作用域中找不到声明, 分析过程会逐层地从外层作用域中查找, 一直到顶层作用域. 在小程序中该顶层作用域为文件作用域. 当分析过程在外层某个作用域中找到了同名的声明, 即停止查找过程. 本文为不同作用域建立基于树状结构表示的嵌套关系, 并为不同作用域下的变量建立基于定义或使用关系的数据依赖路径. 本文即可通过这种数据依赖路径实现

跨作用域的数据流分析.

3.2 小程序 UI 自动化探索

本文基于传统移动应用与小程序在布局上具有相似性这一观察提出了借鉴传统移动应用的 UI 布局知识来识别小程序 UI 布局的方法. 该方法能够从大量的移动应用 UI 布局样本与数据中学习一般的移动端 UI 布局知识, 并将其用于对移动端小程序 UI 布局中不同控件的识别, 以及与这些控件的交互. 另外, 小程序子包的加载行为通常伴随着页面的转换行为发生, 而页面的转换行为通常由特定的 UI 事件触发. 因而本文利用先前构建的小程序隐私传播流图识别 UI 事件与页面转换行为之间的控制流关联, 从而高效地引导小程序 UI 探索过程. 图 4 展示了该方法整体的工作流程, 包含了通过学习与迁移 UI 布局知识识别小程序 UI 控件以及利用控制流信息指导 UI 探索两部分.

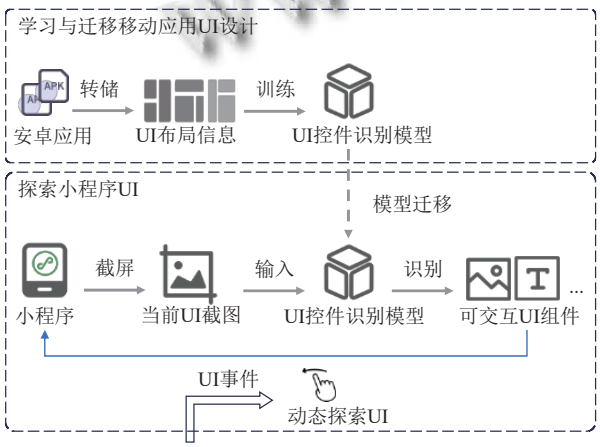

图 4 基于 UI 自动化探索的小程序子包加载方案

3.2.1 UI 布局知识学习与迁移

UI 自动化探索是触发程序中不同功能以保证动态分析代码覆盖率的常用技术手段, 其需要 UI 自动化探索工具能够理解应用的 UI 布局并识别 UI 上的控件. 小程序的 UI 布局信息无法与传统移动应用一样通过常规方式稳定地获取. 但是本文注意到, 小程序在 UI 设计上通常与传统移动应用十分相似. 一方面, 小程序与传统移动应用在 UI 呈现方面共享相同的显示布局, 即它们都显示在智能移动设备的屏幕上. 另一方面, 小程序的 UI 设计旨在帮助用户从移动应用平滑地过渡到小程序, 因而在布局设计上会参考移动应用. 本文基于该观察提出了 UI 布局知识学习与迁移方法, 其核心想法是利用从传统移动应用中获取的大量 UI 布局数据来学习移动设备上的一般 UI 布局信息, 进而用于识别小程序 UI 中的控件.

具体而言, 本文方法使用了 Monkey[27]、Droidbot[28]以及 Stoat[29]等学术界现有的方案对移动应用 UI 进行自动化探索. 本文方法在探索过程中利用安卓调试桥 (ADB) 动态地获取 UI 屏幕截图并转储其对应的布局信息, 即通过“adb shell screencap -p path/to/save/img”命令获取当前 UI 的截图, 以及通过“adb shell uiautomator dump/path/to/save/layout”命令调用安卓系统提供的 UiAutomator[30]工具转储当前 UI 的控件布局信息. 本文通过上述方法从预先收集的传统移动应用程序中累计自动收集了大量训练 UI 样本, 并且利用 UI 控件布局信息对 UI 截图中不同的空间进行标注. 接下来本文采用了对象检测框架 YOLO[31]对这些数据进行训练, 从自动标记的 UI 样本中学习 UI 设计模型, 进而为识别小程序 UI 视图中的控件服务.

3.2.2 控制流信息指导的 UI 探索

当 UI 自动化探索工具能够正确识别小程序 UI 中的控件后, 本文将其应用于对小程序的 UI 自动化探索. UI 自动化探索过程往往基于一定的策略完成一系列 UI 交互动作, 以触发应用中的不同功能. 尽管本文可以采取穷举策略对 UI 上的不同控件实施各种交互行为, 然而其会产生极大的 UI 探索空间, 因而难以保证分析效率. 由于小程序代码中定义了 UI 交互产生的各种响应行为信息, 因而本文利用小程序静态代码中的行为信息来指导 UI 探索.

小程序子包的加载行为通常伴随着页面的转换行为发生, 而页面的转换行为通常由特定的 UI 事件触发. 因而对于能够触发小程序子包加载行为的 UI 事件, 其处理函数与页面转换行为之间应当存在控制依赖关系. 如图 5 所示, 其表示一个简单的 UI 布局以及其对应的静态代码. 在当前 UI 中, 当用户点击文本为“信息收集”的控件时, 其绑定的 navToFillout 函数会被调用. 以 navToFillout 为入口的控制流会调用 wx.navigateTo 函数. 该函数使小程序框架提供的路由函数, 会执行一个转换页面的操作, 因而有可能会加载小程序子包. 本文以该路由函数为起点, 基于先前构建的模块控制流信息后向地寻找触发该控制流的 UI 事件, 最终能够确认该控制流绑定的视图组件属性为 bindTap, 其对应一个用户点击操作. 本文会进一步将该交互操作作为探索当前页面中对应控件的动作.

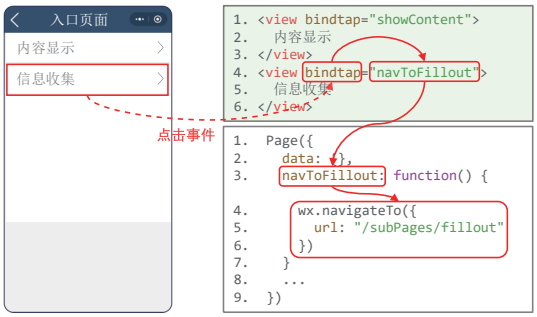

图 5 基于控制依赖关系的 UI 交互行为识别

4 实验评估

本节对本文提出的小程序敏感数据收集行为检测方法进行系统的实验评估. 实验对本文分析方法中两个阶段, 即小程序隐私传播流图构建与小程序 UI 自动化探索的效果进行评估, 并与先前相关工作的方法结果进行对比, 证明本文方法的有效性与优越性.

4.1 实验环境

本文方法中的小程序隐私传播流图构建过程运行在一台服务器上, 其环境与各项参数如下: 操作系统为 Ubuntu 18.04 LTS, CPU 为 Intel Xeon Gold 5218, 内核版本为 4.15.0, 主频 2.30 GHz, 内存大小为 211 GB. 运行的 Java 版本为 1.8.0_201-b09, Python 版本为 3.9.7. 而小程序 UI 自动化探索过程除了需要上述服务器环境之外还需要移动设备. 本文配置了一台一加 9 手机, 其内存为 12 GB.

4.2 实验数据集

由于在所有小程序平台中微信小程序受众最多, 且本文方法的实现基于微信小程序平台, 因此本文在微信小程序上进行评估. 本文通过两个不同的数据集评估本文方法以及工具的有效性.

数据集 1: 公开的小程序基准测试集. 先前关于小程序的研究工作[25]提出了追踪小程序中敏感数据的方法. 为了验证其方法的有效性, 研究人员构建了基准数据集, 用来评估该方法的实际效果. 该数据集包含了由研究者人工构造的 65 个微信小程序, 其中有 54 个小程序包含了一个敏感数据泄露行为, 另外 11 个小程序不包含敏感数据泄露行为.

数据集 2: 本文基于真实小程序应用构建的数据集. 本文研究发现先前基于人工构造的小程序数据集相对简单, 并不能有效反映真实世界中小程序的行为. 例如, 该数据集没有体现复杂小程序中的跨作用域数据流、小程序子包加载等机制. 因此本文进一步在真实小程序中评估本文整体的敏感数据收集行为检测的表现. 本文从微信小程序平台中累计收集了 7862 个小程序, 其中有 785 个小程序因为反编译失败导致配置文件无法识别. 本文保留了剩余的 7077 个小程序作为本文用于实验评估的真实小程序数据集.

4.3 小程序隐私传播流图构建模块评估

首先本节在数据集 1 上将本文方法与先前工作的结果进行比较. 具体而言, 本文考虑了先前的小程序污点分析研究工作 TaintMini 与 MiniTracker. 这两个研究工作本质上都是通过提出面向小程序的污点分析方法来实现敏感数据追踪, 其分析工具都提供了 source 与 sink 的配置文件. 本文并未修改这两个工具的源代码, 仅针对数据集 1 中小程序的实际敏感数据源以及敏感数据泄漏点做了统一配置.

分析结果如表 1 所示, 可以看到 TaintMini 的精确率达到 83.3%, 但是召回率仅能够达到 18.5%. 其分析过程中遇到多个异常导致分析过程中断, 而所有异常情况都会算入到 TN 或者 FN 中. 经本文分析, TaintMini 极高的漏报率主要由 4 个原因导致: 1) 其构建了一个相对粗糙的数据流图, 这限制了其处理复杂表达式的能力. 其对于抽象语法树结构的分析过程过于粗糙, 缺少了对不同结构的解析过程而导致该工具极容易崩溃. 2) TaintMini 仅能够在对象级别追踪数据, 而无法区分对象下的不同属性, 是一种较粗粒度的数据追踪方法. 3) TaintMini 对小程序中常见的 JavaScript 功能缺乏足够的关注, 例如异步调用中经常使用的 Promise 机制等. 4) 本文发现 TaintMini 的代码完成度相对较低, 如其文中声称的对跨页面数据流追踪的支持并未体现在其开源代码中, 这进一步导致其对小程序中数据流追踪的效果受到限制. 相对而言, MiniTracker 的精确率达到 89.8%, 召回率达到了 81.5%. 而本文的工具精确率达到了 90.4%, 召回率达到了 87.4%, 在三者中取得了最好效果. 本文通过进一步分析发现, 本文方法构建的模块控制流图有助于实现常量追踪分析, 例如 obj[prop] 格式的对象属性操作, 本文能够识别 prop 变量对应的具体属性名, 进而能够分析这类方括号表示法表示的对象属性操作. 另外本文方法能够用于追踪跨小程序的敏感数据, 因而在相较于 MiniTracker 能够分析出数据集中更多的敏感数据泄露行为.

接着本节在数据集 2 上将本文方法与先前工作的

结果进行比较. 由于本文通过前一步的实验分析发现 TaintMini 工具的实现完整度较低, 其在分析真实世界小程序时效果欠佳, 因此本实验仅将本文方法的分析效果和 MiniTracker 进行了对比. 本实验关注了小程序应用中两种常见类型的敏感数据, 即用户输入数据和敏感 API 返回的结果. 对于用户输入型敏感数据, 本文采用了先前工作[4]的识别方法, 利用与用户输入组件有关的文本语义信息来判断该用户输入是否为敏感数据. 对于敏感 API, 本文通过人工审核微信开发者文档, 确认了 27 个涉及用户隐私的 API, 具体可以分为系统与设备信息、支付信息、设备媒体信息、地理位置与地址信息、用户个人信息以及其他类型的敏感数据共 6 类. 对于敏感收集行为的判定, 本文认为仅当敏感数据经过网络 API 发送至互联网, 收集行为才算作真正实施, 因而本文仅将 5 个与网络请求相关的 API 作为收集行为的入口. 本实验通过追踪敏感数据源到信息收集入口的路径来识别敏感数据收集行为.

表 1 不同工具在开源基准数据集上的分析效果对比 (%)

工具	精确率	召回率	*F*1-score
TaintMini	83.3	18.5	30.3
MiniTracker	89.8	81.5	85.4
MiniSafe	90.4	87.4	88.7

在数据集 2 上的实验结果如表 2 所示. 实验数据表明, MiniTracker 平均在每个小程序中检测出 5.69 个敏感数据收集行为, MiniSafe 平均在每个小程序中检测出 7.07 个敏感数据收集行为, 平均比 MiniTracker 多 1.38 个.

表 2 不同工具在真实应用中的敏感数据收集行为检出情况

工具	检出总量	平均检出数量
MiniTracker	40233	5.69
MiniSafe	50039	7.07

本节进一步从数据集 2 中随机挑选 100 个较为流行的小程序用于验证本文的实际分析效果. 本文通过人工分析的方式确认标注了这些小程序中敏感数据收集情况. 人工标注的结果显示其中存在 741 个敏感数据收集行为. 本文利用本文工具 MiniSafe 以及先前研究的工具 MiniTracker 分别检测这些敏感数据收集行为. MiniSafe 识别出了 662 个, 召回率达到了 89.34%. 而 MiniTracker 仅能识别出 594 个, 召回率为 80.16%, 相较于本文的方法产生了更多漏报. 本文通过进一步人工分析发现, 造成两个工具共同漏报的一个主要原因是第三方框架开发的小程序. 由于这类小程序与基于一般标准开发的小程序在项目结构、代码结构上都有较大差异, 本文工具以及 MiniTracker 都无法识别这类小程序中逻辑的入口, 因而难以识别其中的敏感数据传播过程. 在其余的小程序中本文方法能够识别出更多的敏感数据收集行为, 其原因主要包括 3 点: 1) 本文方法中的模块控制流分析能够有效识别更多的匿名函数的具体使用方式, 构建出更加完整的控制流信息进而方便了敏感数据追踪; 2) 本文方法中的页面转换分析能够有效识别跨页面的数据流向; 3) 本文方法中的跨作用域数据依赖分析能够帮助识别小程序中普遍存在的跨作用域数据流. 该结果能够证明, 本文构建的小程序隐私传播流图结构在小程序敏感数据收集行为检测方面具有良好效果, 且效果优于现有工作.

4.4 小程序 UI 自动化探索模块评估

接着本文对小程序子包整体加载情况进行统计. 本文用于实验的 7077 个小程序中有 3717 个小程序在其配置文件中声明了使用子包, 且所有小程序声明的子包总数为 69984. 经本文 UI 探索模块触发功能后, 其中成功加载的子包数量为 68882, 占小程序声明的总子包数量的 98.4%. 此结果能够说明本文提出的小程序 UI 自动化探索方法在整体上能够有效触发小程序子包加载, 保证了分析代码的范围.

本文进一步评估小程序 UI 自动化探索模块对小程序敏感数据收集行为检测的效果. 统计结果如表 3 所示, 在 MiniSafe 识别出的 50039 项敏感数据收集行为中, 有 35015 个敏感数据源产生自小程序主包中, 剩余 15024 个产生自小程序子包中. 相较于前文工作, 本文通过设计 UI 自动化探索模块触发子包的加载, 能够识别的敏感数据收集行为总量增加了 42.9%. 可以看到, 考虑了小程序子包代码后, 小程序敏感数据收集行为检出的数量有较大上升. 由此证明了本文小程序 UI 自动化探索方法的必要性和实用性.

表 3 敏感数据收集行为在小程序主包和子包中的分布

工具	主包中的数量	子包中的数量
MiniSafe	35015	15024

5 结语

小程序因其易于传播和使用的特性, 广泛使用于人们生活与工作的方方面面. 但是小程序中敏感数据

使用与管理不当导致的数据泄露问题层出不穷, 由此引发了广泛的隐私安全担忧. 本文聚焦于小程序敏感数据收集行为检测这一用户隐私数据保护过程中的重要问题. 本文通过对具体案例的深入分析, 揭示了检测小程序敏感数据收集行为所面临的挑战, 即如何有效分析小程序中跨不同单元边界的数据流以及如何规避小程序分包机制带来的代码分析范围不完整问题. 本文就上述挑战提出动静态结合的解决方法, 并实现了对应的分析工具 MiniSafe. 实验评估结果表明, 本文的方法检测小程序中敏感数据收集行为的能力优于现有工作, 对保障用户隐私安全具有重要意义.

参考文献

1 阿拉丁研究院. 2022 年度小程序互联网发展白皮书. https://eguayn27bz.feishu.cn/file/boxcnv3PNaH8LvxYHd0sM6O20Ag. (2023-01-11)[2024-03-11].

2 国家互联网应急中心. 2020 年中国互联网网络安全报告. https://www.cert.org.cn/publish/main/upload/File/2020%20Annual%20Report.pdf. [2024-03-11].

3 Li L, Bissyandé TF, Papadakis M, *et al*. Static analysis of Android APPs: A systematic literature review. Information and Software Technology, 2017, 88: 67–95. [doi: 10.1016/j.infsof.2017.04.001]

4 Nan YH, Yang M, Yang ZM, *et al*. Uipicker: User-input privacy identification in mobile applications. Proceedings of the 24th USENIX Security Symposium. Washington: USENIX, 2015. 993–1008.

5 Huang JJ, Li ZC, Xiao XS, *et al*. SUPOR: Precise and scalable sensitive user input detection for Android APPs. Proceedings of the 24th USENIX Security Symposium. Washington: USENIX, 2015. 977–992.

6 Nan YH, Yang ZM, Wang XF, *et al*. Finding clues for your secrets: Semantics-driven, learning-based privacy discovery in mobile APPs. Proceedings of the 25th Annual Network and Distributed System Security Symposium. San Diego: NDSS, 2018.

7 Yang ZM, Yang M. LeakMiner: Detect information leakage on Android with static taint analysis. Proceedings of the 3rd World Congress on Software Engineering. Wuhan: IEEE, 2012. 101–104.

8 Lu L, Li ZC, Wu ZY, *et al*. CHEX: Statically vetting Android APPs for component hijacking vulnerabilities. Proceedings of the 2012 ACM Conference on Computer and Communications Security. Raleigh: ACM, 2012. 229–240.

9 Arzt S, Rasthofer S, Fritz C, *et al*. 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]

10 Li L, Bartel A, Bissyandé TF, *et al*. IccTA: Detecting inter-component privacy leaks in Android APPs. Proceedings of the 37th IEEE International Conference on Software Engineering. Florence: IEEE, 2015. 280–291.

11 Wei FG, Roy S, Ou XM, *et al*. Amandroid: A precise and general inter-component data flow analysis framework for security vetting of Android APPs. ACM Transactions on Privacy and Security, 2018, 21(3): 14.

12 Gordon MI, Kim D, Perkins JH, *et al*. Information flow analysis of Android applications in DroidSafe. Proceedings of the 22nd Annual Network and Distributed System Security Symposium. San Diego: NDSS, 2015.

13 Arzt S, Bodden E. StubDroid: Automatic inference of precise data-flow summaries for the Android framework. Proceedings of the 38th IEEE/ACM International Conference on Software Engineering (ICSE). Austin: IEEE, 2016. 725–735.

14 Wu DY, Gao DB, Deng RH, *et al*. When program analysis meets bytecode search: Targeted and efficient inter-procedural analysis of modern Android APPs in backdroid. Proceedings of the 51st Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN). Taipei: IEEE, 2021. 543–554.

15 Zhang J, Tian C, Duan ZH. FastDroid: Efficient taint analysis for Android applications. Proceedings of the 41st IEEE/ACM International Conference on Software Engineering: Companion Proceedings. Montreal: IEEE, 2019. 236–237.

16 Do LNQ, Ali K, Livshits B, *et al*. Cheetah: Just-in-time taint analysis for Android APPs. Proceedings of the 39th IEEE/ACM International Conference on Software Engineering Companion (ICSE-C). Buenos Aires: IEEE, 2017. 39–42.

17 Lee S, Dolby J, Ryu S. HybriDroid: Static analysis framework for Android hybrid applications. Proceedings of the 31st IEEE/ACM International Conference on Automated Software Engineering (ASE). Singapore: IEEE, 2016. 250–261.

18 Alam S, Qu ZY, Riley R, *et al*. DroidNative: Automating and optimizing detection of Android native code malware variants. Computers & Security, 2017, 65: 230–246.

19 Lu HR, Xing LY, Xiao Y, *et al*. Demystifying resource management risks in emerging mobile APP-in-APP

ecosystems. Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security. ACM, 2020. 569–585.

20 Zhang L, Zhang ZB, Liu AC, *et al*. Identity confusion in WebView-based mobile APP-in-APP ecosystems. Proceedings of the 31st USENIX Security Symposium. Boston: USENIX, 2022. 1597–1613.

21 Zhang Y, Turkistani B, Yang AY, *et al*. A measurement study of WeChat mini-APPs. Proceedings of the 2021 ACM on Measurement and Analysis of Computing Systems, 2021, 5(2): 14.

22 Yang YQ, Zhang Y, Lin ZQ. Cross miniapp request forgery: Root causes, attacks, and vulnerability detection. Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security. Los Angeles: ACM, 2022. 3079–3092.

23 叶瀚, 杨哲慜. 面向小程序的函数调用图构建方法. 小型微型计算机系统. https://kns.cnki.net/kcms2/detail/21.1106.TP.20230606.1805.018.html. (2023-06-07).

24 Wang C, Ko R, Zhang Y, *et al*. TaintMini: Detecting flow of sensitive data in mini-programs with static taint analysis. Proceedings of the 45th IEEE/ACM International Conference on Software Engineering (ICSE). Melbourne: IEEE, 2023. 932–944.

25 Li W, Yang BR, Ye HY, *et al*. MiniTracker: Large-scale sensitive information tracking in mini APPs. IEEE Transactions on Dependable and Secure Computing, 2023, 21(4): 2099–2114. [doi: 10.1109/TDSC.2023.3299945]

26 微信官方文档. 小程序简介. https://developers.weixin.qq.com/miniprogram/dev/framework/quickstart/. [2024-03-11].

27 Google. UI/application exerciser monkey. https://developer.android.com/studio/test/other-testing-tools/monkey. [2024-03-11].

28 Li YC, Yang ZY, Guo Y, *et al*. Droidbot: A lightweight UI-guided test input generator for Android. Proceedings of the 39th IEEE/ACM International Conference on Software Engineering Companion (ICSE-C). Buenos Aires: IEEE, 2017. 23–26.

29 Su T, Meng GZ, Chen YT, *et al*. Guided, stochastic model-based gui testing of Android APPs. Proceedings of the 11th Joint Meeting on Foundations of Software Engineering. Paderborn: ACM, 2017. 245–256.

30 Google. Android UiAutomator. https://developer.android.com/training/testing/ui-automator. [2024-03-11].

31 Redmon J, Divvala S, Girshick R, *et al*. You only look once: Unified, real-time object detection. Proceedings of the 2016 Conference on Computer Vision and Pattern Recognition (CVPR). Las Vegas: IEEE, 2016. 779–788.

(校对责编: 张重毅)