On the Usage-scenario-based Data Minimization in Mini Programs

Shenao Wang
Huazhong University of Science and Technology
Wuhan, Hubei, China
shenaowang@hust.edu.cn

Yanjie Zhao
Monash University
Melbourne, Victoria, Australia
Yanjie.Zhao@monash.edu

Kailong Wang
Huazhong University of Science and Technology
Wuhan, Hubei, China
wangkl@hust.edu.cn

Haoyu Wang
Huazhong University of Science and Technology
Wuhan, Hubei, China
haoyuwang@hust.edu.cn

ABSTRACT

Mini programs, or MiniApps, have become prevalent in the digital landscape, offering convenience but raising privacy concerns, particularly in data minimization. Existing coarse-grained privacy measures fall short in ensuring effective data minimization due to the complex structure of MiniApps and the specificities of data usage scenarios. This work proposes an innovative end-to-end hybrid analysis framework, comprising three key modules, to analyze fine-grained usage-scenario-based data minimization within MiniApps. The framework constructs the page-transition structure, aligns data collection with specific purposes, and detects violations of data minimization principles. We also outline our plan to evaluate the framework through a large-scale study involving 120K MiniApps. This research represents a significant advancement in the pursuit of responsible data practices within MiniApps, contributing to the broader field of computer science and digital security.

CCS CONCEPTS

• **Security and privacy** → *Software security engineering*.

KEYWORDS

Mini-programs; Privacy; Data Minimization

ACM Reference Format:
Shenao Wang, Yanjie Zhao, Kailong Wang, and Haoyu Wang. 2023. On the Usage-scenario-based Data Minimization in Mini Programs. In *Proceedings of the 2023 ACM Workshop on Secure and Trustworthy Superapps (SaTS '23), November 26, 2023, Copenhagen, Denmark*. ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3605762.3624435

1 INTRODUCTION

Mini programs, or MiniApps hereafter, have rapidly gained popularity in the digital ecosystem, becoming a prominent feature within larger hosting platforms, such as WeChat, AliPay, TikTok, etc. These small yet powerful applications provide users with streamlined access to diverse services, from shopping to social networking, without requiring separate installations. However, their capability to access and utilize extensive user data, often in ways that transcend conventional app permissions, has drawn significant attention. As Miniapps continue to flourish, the complex and sometimes opaque nature of their data practices has led to mounting privacy concerns. In particular, the very features that contribute to their flexibility and functionality also introduce challenges in data management and control, raising significant questions regarding user privacy, consent, and security.

In recent years, there has been a global emphasis on privacy protection, giving rise to stringent regulations and a collective consciousness towards responsible data handling [1–4, 6, 8, 10, 14]. Hosting platforms have responded by implementing privacy measures designed to guard permissions and oversee data collection practices within MiniApps [5]. Yet, these measures often prove insufficient to guarantee true data minimization. The underlying issue resides in the current privacy vetting process, which tends to operate at a coarse-grained level, examining overall data and permissions requested for a MiniApp. This approach overlooks the fact that many data collections and permissions are tailored to very specific tasks within a MiniApp. Therefore, even when overall privacy-related practices appear reasonable, they do not necessarily ensure that each individual practice complies with the principle of data minimization. This disconnect reveals a significant gap in our understanding and ability to safeguard user privacy within the increasingly complex world of MiniApps, necessitating a more refined and task-specific approach to privacy analysis.

Analyzing fine-grained usage-scenario-based data minimization within MiniApps presents intricate technical challenges. The first challenge emanates from the current deficiency in our understanding of the exact structure of a MiniApp at runtime. Despite the abundance of static analysis tools focusing on data flow analysis of sensitive data, these tools often fall short in addressing the complexity introduced by sub-packages and dead code within MiniApps. Sub-packages where certain libraries are only executed at runtime, and dead code where certain parts of the source code are not reachable during execution, add layers of intricacy that make conventional static analysis methods inadequate.

The second challenge revolves around the difficulty in obtaining the dedicated functionality for each usage scenario within a

SaTS '23, November 26, 2023, Copenhagen, Denmark

ACM ISBN 979-8-4007-0258-7/23/11…$15.00
https://doi.org/10.1145/3605762.3624435

MiniApp. Given the diverse functionalities and the inherent heterogeneity of these applications, discerning the specific purpose and context for each data collection practice becomes a convoluted task. This heterogeneity impedes the precise alignment of data collection with the minimal requirements for fulfilling a specific task, undermining the ability to ensure that the principle of data minimization is consistently adhered to. Together, these challenges underscore the complexity of ensuring data minimization within MiniApps and point to the necessity for novel approaches that can accurately navigate the multifaceted environment in which these applications operate.

Our Work. In response to the challenges, we propose an end-to-end hybrid analysis framework. This novel framework is composed of three key modules:

Hybrid Analysis Module: The first module employs a hybrid analysis technique, utilizing static analysis to parse and construct the page-transition structure of the MiniApp. This facilitates an easy identification of the usage scenarios for the user, where each usage scenario can be considered as a specific page within the MiniApp. By leveraging dynamic analysis, we can derive the exact data collected and permissions requested, adding depth and specificity to our understanding of each scenario.

Data Minimization Benchmarking Module: The second module focuses on data minimization benchmarking. Using the information related to the specific page, we deduce the purpose of that particular page, and then map this purpose to a corresponding set of privacy-related permissions. This task-level alignment is crucial in ascertaining the precise requirements for data access within each context, enhancing our ability to ensure data minimization.

Comparison Module: The third module encompasses a comparison mechanism, contrasting the results derived from the previous modules to detect inconsistencies and potential overreach. It raises warnings on the pages that violate the data minimization principle, providing actionable guidance for rectifying these breaches.

Building upon the foundation of our innovative hybrid analysis framework, we plan to conduct a large-scale study to evaluate its effectiveness in ensuring data minimization within MiniApps. The evaluation will be both comprehensive and representative, leveraging a dataset that consists of a total number of 120,000 MiniApps. This extensive collection will enable us to probe the nuanced aspects of MiniApp behavior, uncovering insights that may be generalized across various contexts and platforms. Through the examination, we aim to not only validate its efficacy but also refine its methodology, ensuring that it stands as a robust, adaptable solution for the pressing challenges of data privacy in the fast-evolving world of MiniApps.

2 METHODOLOGY

2.1 Methodology Overview

Recognizing the complexity and specificity of data usage scenarios, our approach goes beyond traditional methods, offering a fine-grained, scenario-based solution. As shown in Figure 1, the framework consists of three key modules, each serving a critical function in the analysis and enforcement of data minimization principles:

- Hybrid Analysis Module: Constructs the page-transition structure and identifies usage scenarios.
- Data Minimization Benchmarking Module: Aligns data collection with specific purposes.
- Comparison Module: Detects violations of data minimization principles.

2.2 Hybrid Analysis Module

The hybrid analysis model serves as the foundational module of our framework, focusing on the intricate topology of the MiniApp and translating it into a discernible functional structure. The complex nature of MiniApps often results in a diversity of functionalities interwoven within the same application. By employing a hybrid analysis approach, we are able to dissect this complexity and provide a structured view that aligns with the unique characteristics of MiniApps.

2.2.1 Static Analysis.

UI State Transition Analysis. We initiate the analysis by leveraging the UI state transition, using it to construct a topology that authentically reflects the functional structure of the MiniApp. By mapping the UI state transitions, we create a network that mirrors the navigation patterns within the application. This step will also assist in identification of the dead code inside the MiniApp.

Given that a MiniApp may harbor diverse functionalities and that each page typically focuses on a specific function (e.g., booking a ticket, editing a document), it is crucial to isolate these functions and examine them individually. We thus extract the usage scenarios on a per-page basis. From the UI state transition model, we can effectively separate different pages and categorize the diverse usage scenarios, allowing for targeted analysis of each function.

Permission Request Analysis. In tandem with the functional analysis, we also examine the requested permissions in the source code. By scrutinizing the permissions, we obtain a preliminary view of the access rights required by each page or function. This information is vital and serves as a basis that will be further confirmed and refined during the dynamic analysis phase.

2.2.2 Dynamic Analysis.

In the dynamic analysis component of our hybrid approach, we place a particular emphasis on identifying subpackages within the MiniApp, which are crucial as they are loaded only upon execution. Ignoring these subpackages would hinder comprehensive coverage during dynamic analysis. To navigate this complexity, we design algorithms specifically crafted to thoroughly traverse the MiniApp and trigger potential subpackage downloads. This method ensures that no functional aspect of the MiniApp is overlooked. Simultaneously, we diligently record triggered permission requests during this process, which serves to validate and corroborate the results obtained from the static analysis. Through these intricate measures, the dynamic analysis offers a more exhaustive perspective on the MiniApp's structure and behavior, enhancing the overall reliability of our framework.

2.3 Data Minimization Benchmarking Module

In the data minimization benchmarking module, we undertake a synergistic approach by leveraging both static and dynamic analysis to deduce the purpose of each usage scenario. This process is carried out in two key steps. Firstly, we identify the textual contents

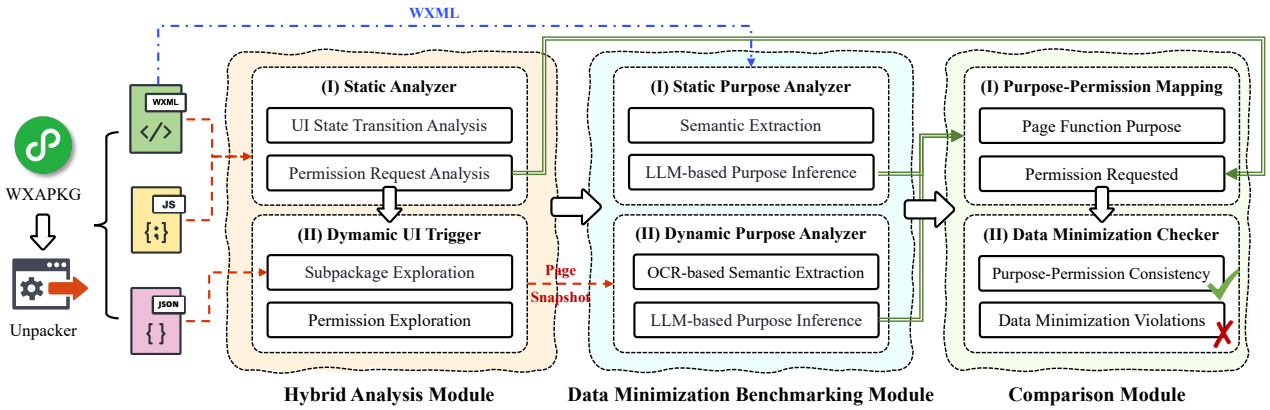

Figure 1: The Architecture and Workflow of Our Framework

displayed for each page. For the static analysis, we ascertain the page's purpose by referencing the WXML file, a crucial source that describes the structural layout of the MiniApp's pages. To further complement this, we screenshot each page during the dynamic traversal process and employ Optical Character Recognition (OCR) technology to identify the textual content. Secondly, with the aid of state-of-the-art Natural Language Processing (NLP) techniques, including the utilization of various large language models (LLMs), we deduce the specific purpose of each page. This intricate blend of methods ensures a nuanced and precise understanding of the functionality and intention behind every page in the MiniApp, crucial for evaluating compliance with data minimization principles.

2.4 Comparison Module

The final component in our framework is the comparison module, a critical stage that blends the insights gleaned from both the functional purpose extracted from each page and the derived permissions requested. This fusion enables us to pinpoint instances of data over-collection beyond what is necessary for the given function. A key aspect of this process involves mapping the functional purpose to a predefined set of permissions, an essential step that facilitates the comparison. By employing this mapping, we are able to systematically analyze the alignment between the function of each page and the permissions requested, thereby identifying any discrepancies. Should a page request permissions that are inconsistent with its functional purpose, this serves as a warning sign of potential data over-collection.

3 EVALUATION

3.1 Research Questions

We plan to conduct a large-scale evaluation on our framework, aiming at answering the following research questions (RQs):

• **RQ1 (Effectiveness): How effective is the framework in deducing page purpose and identifying permission requested?** This question evaluates the capability of our framework in accurately deducing the functional purpose of each MiniApp page and identifying the corresponding permissions requested. The evaluation includes assessing the precision in understanding page purposes through static and dynamic analysis, confirming the completeness and accuracy of captured permissions.

• **RQ2 (Ablation Study): How do static and dynamic analyzers contribute to the performance separately?** This research question targets an ablation study to dissect the individual contributions of static and dynamic analyzers within the framework.

• **RQ3 (A Large-scale Study in the Real World): How prevalent are data minimization violations in MiniApps ecosystem? What are their characteristics?** This research question probes into the frequency and nature of data minimization violations within MiniApp ecosystem. By utilizing the proposed framework to analyze a substantial dataset of MiniApps, RQ3 aims to ascertain the extent of over-collection of permissions and data that goes beyond the necessity of specific functions or pages. The study will also uncover specific patterns and characteristics of these violations, shedding light on potential systemic weaknesses or trends.

3.2 Dataset Collection

To collect MiniApps, we utilize `MiniCrawler` [24] to download MiniApp packages from the WeChat App Market. We have collected a total of 127,460 MiniApps, with 289 GB of total size. The crawler works on a server running Ubuntu Linux of 22.04 version with two 64-core AMD EPYC 7713 and 256 GB RAM. To load subpackage at runtime, the dynamic analysis is run on an Android Virtual Device (AVD) with a system version of Android 8.1.0 and API version Level 27. The version of WeChat used is 8.0.37, and the WebView kernel version is 107.0.5304.141.

4 RELATED WORK

In recent years, MiniApps have emerged as a new application paradigm and have garnered significant scholarly interest. Previous research in this area can be classified into two main aspects.

4.1 MiniApp Security

Several investigations have delved into the security aspects of Miniapps, shedding light on various vulnerabilities and threats [11, 12, 16, 17, 20–22]. For example, one particular research effort collected 83 real-world MiniApp bugs and gave rise to WeDetector, a

tool aimed at identifying WeBugs by following three distinct bug patterns [17]. Another study probed into problems such as system resource exposure, subwindow spoofing, and subroutine hijacking within the Mini-Program ecosystem, conducting evaluations on 11 prominent platforms to highlight the pervasive nature of these security issues [12]. Moreover, a novel issue regarding privacy leaks in MiniApps has been explored [22], potentially leading to private data theft by the MiniApp platform, with the researchers detailing an attack process that exploits this vulnerability. Additionally, the discovery of a Cross Mini-Program request forgery vulnerability (CMRF) [20] has been documented, along with the development of the CMRFScanner tool for its detection.

4.2 MiniApp Privacy

A series of studies have emphasized the importance of privacy in MiniApp ecosystem [7, 9, 13, 15, 18, 19, 23, 25]. TaintMini [15] introduced a framework for detecting flows of sensitive data within and across mini-programs using static taint analysis. Another work MiniTracker [9] constructed assignment flow graphs as common representation across different host apps and performed a large-scale study on 150k MiniApps, which revealed the common privacy leakage patterns. Moreover, several studies [7, 13, 25] have focused on taint analysis technique to detect AppSecret leaks. In particular, another work [18] focused on the consistency of data collection and usage in MiniApps. They crawled 2,998 MiniApps and detected 89.4% of them violated their privacy policies. More recently, Zhang et al. introduced SPOChecker and performed the first systematic study of privacy over-collection in MiniApps. Despite these significant contributions to understanding privacy dimensions, there remains a noticeable gap in the literature concerning the privacy compliance and data minimization of Miniapps, indicating a crucial area for further exploration and research.

5 CONCLUSION

This research has illuminated the pressing challenge of data minimization within the complex domain of MiniApps, a challenge that has been compounded by existing coarse-grained privacy measures. In response, we introduced an end-to-end hybrid analysis framework designed to address this issue at a fine-grained level. Comprising three key modules, the framework offers a nuanced, usage-scenario-based approach to data privacy. The planned large-scale study, encompassing 120K MiniApps, will further validate and potentially refine this groundbreaking solution. Our work signifies a vital step towards transparent and responsible data practices in MiniApps, contributing to broader advances in digital security and computer science.

ACKNOWLEDGEMENT

The authors would like to thank all the anonymous shepherd and reviewers for improving this manuscript. We also thank Moxuan Wang, Lizheng Wang, Zhengyang Xiao, Qian Huang for their insightful discussions and feedbacks. This work was supported in part by National Key R&D Program of China (2021YFB2701000), the National Natural Science Foundation of China (grant No.62072046), Knowledge Innovation Program of Wuhan-Basic Research and HUST CSE-HongXin Joint Institute for Cyber Security.

REFERENCES

[1] 2022. Act on the Protection of Personal Information. https://www.ppc.go.jp/.
[2] 2022. California Consumer Privacy Act. https://oag.ca.gov/privacy/ccpa.
[3] 2022. General Data Protection Regulation. https://commission.europa.eu/law/law-topic/data-protection_en.
[4] 2022. Personal Data Protection Act. https://www.pdpc.gov.sg/.
[5] 2023. WECHAT POLICY. https://www.wechat.com/en/privacy_policy.html.
[6] Benjamin Andow, Samin Yaseer Mahmud, Justin Whitaker, William Enck, Bradley Reaves, Kapil Singh, and Serge Egelman. 2020. Actions speak louder than words:{Entity-Sensitive} privacy policy and data flow analysis with {PoliCheck}. In *29th USENIX Security Symposium (USENIX Security 20)*. 985–1002.
[7] Supraja Baskaran, Lianying Zhao, Mohammad Mannan, and Amr Youssef. 2023. Measuring the Leakage and Exploitability of Authentication Secrets in Super-apps: The WeChat Case. *arXiv preprint arXiv:2307.09317* (2023).
[8] Duc Bui, Yuan Yao, Kang G Shin, Jong-Min Choi, and Junbum Shin. 2021. Consistency analysis of data-usage purposes in mobile apps. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*. 2824–2843.
[9] Wei Li, Borui Yang, Hangyu Ye, Liyao Xiang, Qingxiao Tao, Xinbing Wang, and Chenghu Zhou. 2023. MiniTracker: Large-Scale Sensitive Information Tracking in Mini Apps. *IEEE Transactions on Dependable and Secure Computing* (2023).
[10] Yuxi Ling, Kailong Wang, Guangdong Bai, Haoyu Wang, and Jin Song Dong. 2022. Are they toeing the line? diagnosing privacy compliance violations among browser extensions. In *Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering*. 1–12.
[11] Yi Liu, Jinhui Xie, Jianbo Yang, Shiyu Guo, Yuetang Deng, Shuqing Li, Yechang Wu, and Yepang Liu. 2020. Industry practice of javascript dynamic analysis on wechat mini-programs. In *Proceedings of the 35th IEEE/ACM International Conference on Automated Software Engineering*. 1189–1193.
[12] Haoran Lu, Luyi Xing, Yue Xiao, Yifan Zhang, Xiaojing Liao, XiaoFeng Wang, and Xueqiang Wang. 2020. Demystifying resource management risks in emerging mobile app-in-app ecosystems. In *Proceedings of the 2020 ACM SIGSAC conference on computer and communications Security*. 569–585.
[13] Shi Meng, Liu Wang, Shenao Wang, Kailong Wang, Xusheng Xiao, Guangdong Bai, and Haoyu Wang. 2023. WeMinT: Tainting Sensitive Data Leaks in WeChat Mini-Programs. In *Proceedings of the 38th IEEE/ACM International Conference on Automated Software Engineering*. IEEE.
[14] Faysal Hossain Shezan, Zihao Su, Mingqing Kang, Nicholas Phair, Patrick William Thomas, Michelangelo van Dam, Yinzhi Cao, and Yuan Tian. 2023. CHKPLUG: Checking GDPR Compliance of WordPress Plugins via Cross-language Code Property Graph.. In *NDSS*.
[15] Chao Wang, Ronny Ko, Yue Zhang, Yuqing Yang, and Zhiqiang Lin. 2023. Taintmini: Detecting flow of sensitive data in mini-programs with static taint analysis. In *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*. IEEE, 932–944.
[16] Chao Wang, Yue Zhang, and Zhiqiang Lin. 2023. Uncovering and Exploiting Hidden APIs in Mobile Super Apps. *arXiv preprint arXiv:2306.08134* (2023).
[17] Tao Wang, Qingxin Xu, Xiaoning Chang, Wensheng Dou, Jiaxin Zhu, Jinhui Xie, Yuetang Deng, Jianbo Yang, Jiaheng Yang, Jun Wei, et al. 2022. Characterizing and detecting bugs in WeChat mini-programs. In *Proceedings of the 44th International Conference on Software Engineering*. 363–375.
[18] Yin Wang, Ming Fan, Junfeng Liu, Junjie Tao, Wuxia Jin, Qi Xiong, Yuhao Liu, Qinghua Zheng, and Ting Liu. 2023. Do as You Say: Consistency Detection of Data Practice in Program Code and Privacy Policy in Mini-App. *arXiv preprint arXiv:2302.13860* (2023).
[19] Yuqing Yang, Chao Wang, Yue Zhang, and Zhiqiang Lin. 2023. SoK: Decoding the Super App Enigma: The Security Mechanisms, Threats, and Trade-offs in OS-alike Apps. *arXiv preprint arXiv:2306.07495* (2023).
[20] Yuqing Yang, Yue Zhang, and Zhiqiang Lin. 2022. Cross Miniapp Request Forgery: Root Causes, Attacks, and Vulnerability Detection. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*. 3079–3092.
[21] Jianyi Zhang, Leixin Yang, Yuyang Han, Zixiao Xiang, and Xiali Hei. 2023. A Small Leak Will Sink Many Ships: Vulnerabilities Related to mini-programs Permissions. In *2023 IEEE 47th Annual Computers, Software, and Applications Conference (COMPSAC)*. IEEE, 595–606.
[22] Lei Zhang, Zhibo Zhang, Ancong Liu, Yinzhi Cao, Xiaohan Zhang, Yanjun Chen, Yuan Zhang, Guangliang Yang, and Min Yang. 2022. Identity Confusion in {WebView-based} Mobile App-in-app Ecosystems. In *31st USENIX Security Symposium (USENIX Security 22)*. 1597–1613.
[23] Xiaohan Zhang, Yang Wang, Xin Zhang, Ziqi Huang, Lei Zhang, and Min Yang. 2023. Understanding Privacy Over-collection in WeChat Sub-app Ecosystem. *arXiv preprint arXiv:2306.08391* (2023).
[24] Yue Zhang, Bayan Turkistani, Allen Yuqing Yang, Chaoshun Zuo, and Zhiqiang Lin. 2021. A measurement study of wechat mini-apps. *ACM SIGMETRICS Performance Evaluation Review* 49, 1 (2021), 19–20.
[25] Yue Zhang, Yuqing Yang, and Zhiqiang Lin. 2023. Don't Leak Your Keys: Understanding, Measuring, and Exploiting the AppSecret Leaks in Mini-Programs. *arXiv preprint arXiv:2306.08151* (2023).