Proceedings of the 3rd International Conference on Software Engineering and Machine Learning
DOI: 10.54254/2755-2721/2025.23103

LLM-PrivCheck: Detecting Privacy Policy Violations in Android Applications with LLMs

Zexiong Xiang[1], Shanshan Wang[1*]

[1]*School of Information Science and Engineering, University of Jinan, Jinan, China*
**Corresponding Author. Email: ise_wangss@ujn.edu.cn*

Abstract: With the widespread use of Android applications, frequent access to personal user data (such as location information and device identifiers) has become the norm. However, inconsistencies between privacy policy statements and actual app behavior often lead to significant privacy risks. This paper proposes a consistency detection method, LLM-PrivCheck, based on large language models (LLMs), which automatically checks whether the data collection statements in privacy policies align with the actual behavior of applications. LLM-PrivCheck utilizes LLMs for privacy policy parsing and combines taint analysis techniques to track data flows, ultimately achieving consistency detection through ontology mapping. Experimental results show that in tests on 960 commercial applications from Google Play, LLM-PrivCheck detected that 96.25% of the apps had inconsistencies between their privacy policies and actual behavior.

Keywords: Android applications, privacy policies, consistency detection, taint analysis

1. Introduction

With the rapid development of the Android ecosystem, applications are increasingly collecting user personal data to provide personalized services. However, the collection and sharing of this data is often opaque, and inconsistencies exist between privacy policies and app code. For example, some applications do not explicitly state in their privacy policies the collection of certain data, while the actual app code includes operations for collecting this data. Such inconsistencies can lead to significant privacy risks and violate privacy regulations such as the GDPR[1] and CCPA[2]. Additionally, as user concerns over privacy protection continue to rise, related laws and regulations are becoming increasingly stringent. The EU Digital Services Act (DSA)[3], which came into effect in 2023, further requires applications to provide transparent privacy policies and ensure consistency with actual behavior. Therefore, accurately detecting inconsistencies in privacy policies is crucial for protecting user rights and ensuring regulatory compliance. Existing research mainly relies on static and dynamic analysis techniques to detect privacy data flows. For example, AppCensus[4] primarily relies on dynamic analysis, LeakPred[5] detects privacy data flows by identifying privacy leakage components in Android applications, and PoliCheck[6] employs a combined static and dynamic analysis approach. However, these methods have the following limitations:

- Traditional NLP technologies lack a deep understanding of privacy policies, and most tools only perform keyword matching, making it difficult to accurately parse complex privacy statements.

DOI: 10.54254/2755-2721/2025.23103

- Traditional taint analysis depends on manually defined data flow rules and taint propagation paths, making automation and high accuracy difficult to achieve.
- Handling code obfuscation is challenging, as many applications use ProGuard obfuscation techniques, making it hard for traditional methods to trace data flows.

To address these issues, this paper proposes the LLM-PrivCheck research framework, which combines large language models (LLMs), static analysis, and ontology mapping to improve the accuracy of consistency detection.

The main contributions of this work are summarized as follows:

- Privacy policy texts are parsed using LLMs, and a comparison with traditional NLP methods shows that this approach yields better results.
- A novel approach combining large models with taint analysis is applied to Android applications, efficiently and automatically extracting data flows.
- Building on the previous two contributions, the LLM-PrivCheck framework is proposed for consistency detection, effectively identifying inconsistencies between privacy policies and actual behavior in Android applications.

The structure of the remaining sections is as follows: Section 2 provides a detailed description of the model, followed by experimental validation in Section 3. Finally, Section 4 concludes the paper.

2. Method

The LLM-PrivCheck framework is proposed as a consistency detection solution based on LLMs, aimed at protecting user privacy. The core of LLM-PrivCheck lies in utilizing LLMs to perform privacy policy information extraction and code taint analysis, generating two reports, and then conducting consistency detection related to user privacy behavior based on these reports. As shown in Figure 1, in the specific implementation process, first, the LLM conducts in-depth analysis of the privacy policy text, extracting key privacy information such as data collection types, purposes of use, sharing entities, and so on. Next, taint analysis is performed on the code to track the propagation paths of user data within the code, identifying operations that may involve privacy. Throughout this process, the LLM assists in understanding the code logic and semantics, determining whether these operations align with the descriptions in the privacy policy.

The two generated reports are the privacy policy report and the code analysis report. The privacy policy report lists in detail all the key information extracted from the privacy policy text, while the code analysis report records the behaviors involving user privacy and the data flow in the code. By comparing these two reports, inconsistencies between the privacy policy and the actual code behavior can be identified, allowing for timely adjustments and corrections to ensure that user privacy is adequately protected. The advantage of this approach lies in combining the powerful natural language processing capabilities of LLMs with the precision of code taint analysis, enabling more accurate detection of consistency between privacy policies and code behavior, thereby effectively enhancing user privacy protection.

DOI: 10.54254/2755-2721/2025.23103

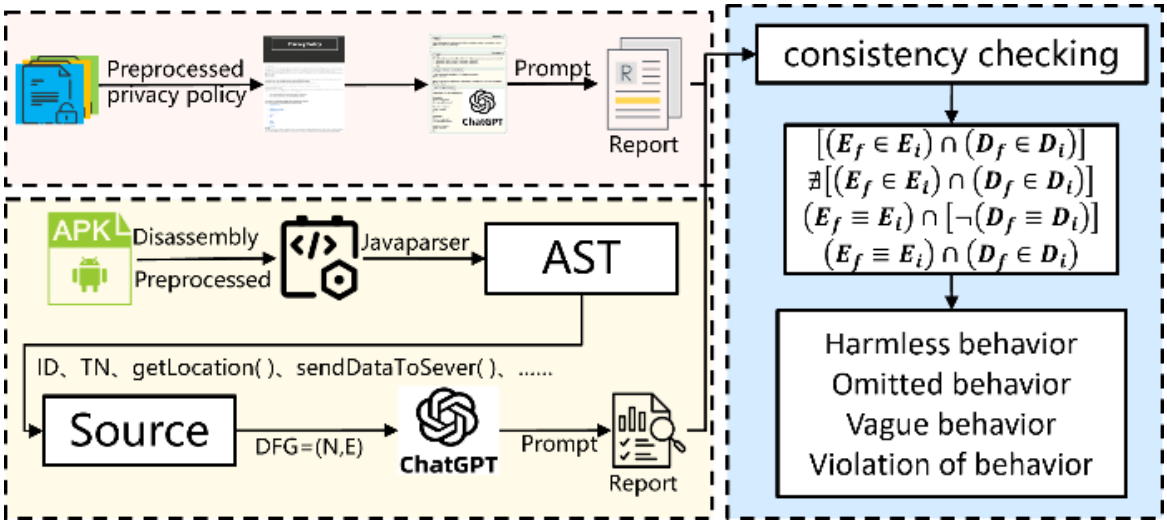

Figure 1: Structure diagram of the LLM-PrivCheck framework

2.1. Privacy policy parsing

The performance of large language models (LLMs) is influenced by various factors, among which prompt design, configuration parameters, and model selection are key factors. In this study, a Design Science Research (DSR) approach [7][8] was adopted to develop an LLM framework for privacy policy analysis and to evaluate its effectiveness. This approach enabled the systematic exploration and optimization of various aspects of the model to achieve optimal performance. A comparative analysis with existing state-of-the-art solutions was also conducted to verify the competitiveness and generalization capability of our framework. Through this systematic process, a well-performing and generalizable configuration was successfully proposed, offering a new and effective solution for the automated analysis of privacy policies.

In terms of prompt engineering, extensive experiments and tests were conducted to explore the impact of different prompt designs on model performance. Several methods were tested, such as specifying data boundaries, expanding task descriptions, and using Few-shot prompts. By analyzing the test results, it was found that some prompt engineering practices had a significant positive impact on model performance, while others might have a negative effect. After multiple iterations and optimizations, the optimal prompt engineering combination was identified, ensuring that the model could achieve the best performance in privacy policy analysis tasks.

The model parameter configuration focused on three key parameters: model temperature, Top p, and system input. Through a series of tests and analyses, it was found that the temperature parameter had the most significant impact on test results for privacy policy information extraction tasks. When the temperature was set to a low value, particularly 0, the model exhibited the most stable and accurate performance. Further experiments demonstrated that, when the temperature was set to 0, variations in the Top p parameter within the range of [0, 1] had a relatively small impact on model performance. Based on these findings, it was concluded that setting Top p to its default value and temperature to 0 represented the best parameter combination to ensure the stability and accuracy of the model in privacy policy analysis. Additionally, tests indicated that system input did not have a positive effect on the model's performance, leading to its exclusion from the final configuration.

DOI: 10.54254/2755-2721/2025.23103

2.2. Data flow extraction

Data flow extraction utilizes taint analysis technology [9], which parses the data flow propagation paths and privacy-sensitive API calls in the app source code to identify potential privacy risks. As shown in Figure 2, the execution of this method is divided into the following four steps:

- **AST Syntax Tree Construction:** The Abstract Syntax Tree (AST) is used to parse the source code, extracting its structural information and converting the code into a tree-like structure. The focus is on variable scope, data flow propagation paths, and code hierarchy, providing fundamental data support for subsequent analysis.
- **Function Call Chain Generation:** The complete function call relationships are constructed to identify and trace the code locations that call security-sensitive functions (such as file operations, network communications, etc.). The data flow dependencies are traced, and a complete call chain is established, considering cross-file and cross-module calling relationships to ensure the integrity of the call chain.
- **Dangerous Flow Generation:** After obtaining the call chain, potential dangerous data flows are analyzed, identifying the propagation paths of external input data in the call chain. These inputs are checked for their impact on the parameters of security-sensitive functions. Taint analysis is combined to trace the data propagation path and identify potential security risks.
- **Prompt Sequence Construction:** Based on the dangerous flow, targeted prompt sequences are generated. Each function in the dangerous flow is treated as an analysis node, and multi-level prompts are constructed using its data flow information. A staged prompt strategy is used to guide the LLM in code understanding and vulnerability identification, improving the accuracy and interpretability of the analysis.

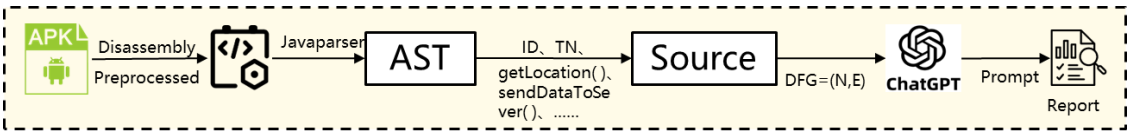

Figure 2: Data flow analysis principle diagram

2.3. Consistency detection

The core function of the consistency detection algorithm is to determine whether the data flows related to third-party collection of user information in the application code are adequately and explicitly disclosed in the privacy policy. To achieve this goal, the algorithm classifies disclosures into four distinct types [10] based on the semantic relationships between terms in the privacy policy: clear disclosures, vague disclosures, omitted disclosures, and negative disclosures. This classification method facilitates systematic analysis and judgment of different disclosure situations regarding data flows, thereby improving the accuracy and consistency of detection.

For the definition and modeling of the algorithm, it is assumed that an application can be abstracted as a tuple $a=(F,P)$, where F represents the data flow information obtained from static analysis of the application, and P represents the collection statement content extracted from the privacy policy analysis. Additionally, to more specifically characterize the attributes of privacy data flows, set D is defined as the set of types of privacy data, and set E represents the set of potentially involved third-party entities. Based on these definitions, the consistency detection rules intuitively describe the matching relationship between the application data flow and the privacy policy, providing a clear rule framework and methodological guidance for identifying inconsistencies in data flows. This rule model allows for efficient detection of whether the privacy policy comprehensively reflects actual data collection behaviors and helps identify potential privacy compliance issues.

DOI: 10.54254/2755-2721/2025.23103

3. Experiment

3.1. Dataset

A total of 13,417 Android applications and their privacy policies were crawled from 32 categories on Apkpure (a mirror site for Google Play) for the purpose of conducting experiments. To validate the effectiveness of the approach, the privacy policy parsing and data flow extraction tasks were analyzed separately. For the privacy policy task, the publicly available MAPP corpus [11] was utilized, while the DroidLeaks dataset was used for data flow extraction. After validating the effectiveness of the privacy policy parsing and data flow extraction methods, experimental analysis was conducted on the real dataset collected.

3.1.1. MAPP dataset

The MAPP dataset [11] holds significant value in the field of privacy policy analysis. This dataset includes privacy policies from 64 applications on Google Play Store, divided into paragraphs. Each paragraph is annotated by legal experts to indicate whether it involves the collection or sharing of different types of personal data. For instance, a paragraph may be marked as disclosing the collection of geographic location information. Such fine-grained annotations help in the accurate training and evaluation of NLP models to identify specific data processing methods. Notably, the MAPP corpus is one of the few datasets with multilingual annotations, supporting both English and German, making it more applicable for cross-language legal research.

3.1.2. DroidLeaks dataset

The DroidLeaks dataset [12] plays a key role in privacy and security vulnerability analysis for Android applications. This dataset includes a large number of applications collected from the Google Play Store, and it extracts code patterns through static analysis methods that may lead to the leakage of sensitive information. Each detected code snippet is annotated by experts to indicate whether it involves improper access or sharing of privacy data. For example, a piece of code may be labeled as accessing contact information without the user's consent. Such fine-grained annotations help in training and evaluating NLP models and program analysis tools to accurately identify potential data leakage risks. It is worth noting that the DroidLeaks dataset covers a variety of application categories, making it more widely applicable for Android security research and privacy compliance detection.

3.2. Analysis of experimental results

3.2.1. Privacy policy parsing experiment

Experiments were conducted using the MAPP dataset to verify and quantify the optimal configuration effect of Deepseek. A stratified sampling method was employed to construct balanced experimental and control groups, ensuring that each subset adequately represented the diversity of the overall data. The experimental group consisted of 33 policy documents, while the control group contained 31. The study began with the design of prompt words, which were applied to identify the types of personal data in privacy policies. After several rounds of partitioned testing, the prompt words were optimized through strategies such as specifying data boundaries, adjusting data arrangement, enhancing task descriptions, trimming data, and introducing a few example prompts. Furthermore, by adjusting Deepseek parameters (such as temperature, top p, and system input), response quality was further improved, and the model was fine-tuned to enhance its performance.

As shown in the experimental results in Table 1, enhancing task descriptions and using Few-shot prompts had a significant positive effect on the model's performance. However, the arrangement of

DOI: 10.54254/2755-2721/2025.23103

data placement weakened the model, while specifying data boundaries had no significant effect, and data pruning had a negligible impact on the model. From the comparative experiment in Figure 3, it is evident that the Deepseek model outperformed both GPT-2 and the traditional Policylint method [13]. Therefore, in the subsequent consistency detection, the optimized Deepseek model will be adopted.

Table 1: Prompt engineering experimental results

Label	Specify Boundary	Data Placement (Back)	Expand Description	Data Pruning	Few-shot
Acc	+1.54%	-1.22%	**+6.32%**	-	**+5.12%**
Pre	+1.36%	-0.10%	**+7.56%**	-	**+6.58%**
Recall	**+0.74%**	-1.58%	**-0.21%**	-	-
F1	+0.93%	-0.94%	**+3.54%**	-	**+2.75%**

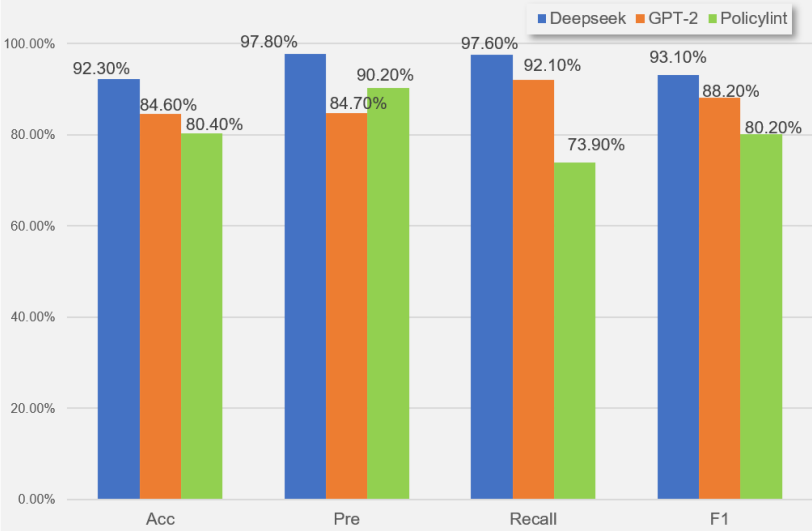

Figure 3: Comparative experimental results for privacy policy parsing

3.2.2. Data flow extraction experiment

The experiment was conducted on a server equipped with a Core i7 CPU and 8GB of RAM. The data source was collected from the Google Play Store and the DroidLeaks public dataset. A comparison was made between the LLM-based method and the traditional taint analysis method, LeakPred [7]. As shown in Figure 4, a 9.23% improvement in the accuracy of extracting sensitive privacy data flows was observed, indicating that the LLM-based method outperforms the traditional approach in data flow extraction for Android applications.

DOI: 10.54254/2755-2721/2025.23103

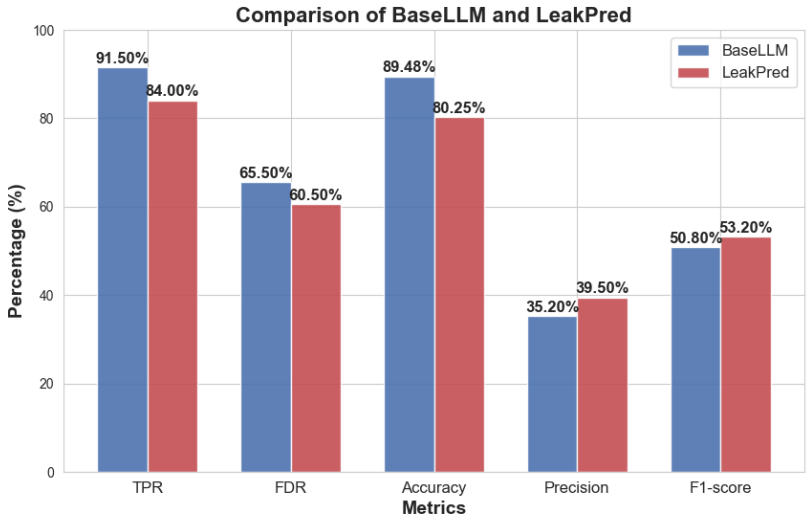

Figure 4: Comparative experimental results for data flow extraction

3.2.3. Consistency detection

After completing the experiments in the previous two steps, the optimized model was run on the real dataset. This dataset was obtained through stratified sampling from 32 categories based on download rankings in the Google Play Store, resulting in a total of 960 applications. Following the completion of the first two steps of LLM-PrivCheck on the real dataset, a consistency detection analysis was conducted to assess privacy data leakage. According to the results in Table 2, a total of 3,255 data flows related to privacy data were collected from the 960 applications. Of these, 96.25% of the applications displayed inconsistencies between their privacy policies and actual behaviors, with 90.67% of the data flows showing inconsistencies.

Table 2: Consistency detection results

Behavior type	APP	Data Flow
Harmless Behavior	**36**	**304**
Vague Behavior	**617**	**1889**
Omitted Behavior	**268**	**909**
Violation Behavior	**42**	**153**
Total	**960**	**3255**

4. Conclusion

This paper discusses a consistency detection method based on large models, LLM-PrivCheck, which can automatically check whether the data collection statements in privacy policies align with the actual behaviors of applications. Based on the dataset, our evaluation results indicate that this method effectively detected inconsistencies between privacy policies and actual behaviors in 96.25% of the applications. By integrating large models with privacy protection, this method offers a new approach to safeguarding user privacy.

DOI: 10.54254/2755-2721/2025.23103

Acknowledgements

The anonymous reviewers are thanked for their valuable comments and suggestions, which contributed to the improvement of the paper's quality and the depth of the research.

References

[1] *Voigt P, Von dem Bussche A. The eu general data protection regulation (gdpr)[J]. A practical guide, 1st ed., Cham: Springer International Publishing, 2017, 10(3152676): 10-5555.*
[2] *Bonta R. California consumer privacy act (CCPA)[J]. Retrieved from State of California Department of Justice: https://oag. ca. gov/privacy/ccpa, 2022.*
[3] *Wilman F. The Digital Services Act (DSA)-An Overview[J]. Available at SSRN 4304586, 2022.*
[4] *Egelman S. Taking responsibility for someone else's code: Studying the privacy behaviors of mobile apps at scale[C]//2020 {USENIX} Conference on Privacy Engineering Practice and Respect ({PEPR} 20). 2020.*
[5] *Lima J G, Giusti R, Dias-Neto A C. LeakPred: An Approach for Identifying Components with Resource Leaks in Android Mobile Applications[J]. Computers, 2024, 13(6): 140.*
[6] *Andow B, Mahmud S Y, Whitaker J, et al. Actions speak louder than words:{Entity-Sensitive} privacy policy and data flow analysis with {PoliCheck}[C]//29th USENIX Security Symposium (USENIX Security 20). 2020: 985-1002.*
[7] *Vom Brocke J, Hevner A, Maedche A. Introduction to design science research[J]. Design science research. Cases, 2020: 1-13.*
[8] *Lima J G, Giusti R, Dias-Neto A C. LeakPred: An Approach for Identifying Components with Resource Leaks in Android Mobile Applications[J]. Computers, 2024, 13(6): 140.*
[9] *Rodriguez D, Yang I, Del Alamo J M, et al. Large language models: a new approach for privacy policy analysis at scale[J]. Computing, 2024, 106(12): 3879-3903.*
[10] *Tan, Z. Y. Research on Android Applications Leaking User Privacy to Third Parties. Nanjing University of Science and Technology, 2021. DOI:10.27241/d.cnki.gnjgu.2021.003590.*
[11] *Arora S, Hosseini H, Utz C, et al. A tale of two regulatory regimes: Creation and analysis of a bilingual privacy policy corpus[C]//LREC proceedings. 2022.*
[12] *Liu Y, Wang J, Wei L, et al. DroidLeaks: a comprehensive database of resource leaks in Android apps[J]. Empirical Software Engineering, 2019, 24: 3435-3483.*
[13] *Andow B, Mahmud S Y, Wang W, et al. {PolicyLint}: investigating internal privacy policy contradictions on google play[C]//28th USENIX security symposium (USENIX security 19). 2019: 585-602.*