

Hybrid Privacy Policy-Code Consistency Check using Knowledge Graphs and LLMs

Zhenyu Mao¹, Xinxin Fan¹, Yifei Wang¹, Jacky Keung^{1,*}, and Jialong Li²

¹City University of Hong Kong, Hong Kong, China

²Waseda University, Tokyo, Japan

zhenyumao2-c@my.cityu.edu.hk, xinxinfan3-c@my.cityu.edu.hk

ywang4748-c@my.cityu.edu.hk, Jacky.Keung@cityu.edu.hk, lijialong@fuji.waseda.jp

*corresponding author

Abstract—The increasing concern in user privacy misuse has accelerated research into checking consistencies between smartphone apps’ declared privacy policies and their actual behaviors. Recent advances in Large Language Models (LLMs) have introduced promising techniques for semantic comparison, but these methods often suffer from low accuracies and expensive computational costs. To address this problem, this paper proposes a novel hybrid approach that integrates 1) knowledge graph-based deterministic checking to ensure higher accuracy, and 2) LLMs exclusively used for preliminary semantic analysis to save computational costs. Preliminary evaluation indicates this hybrid approach not only achieves 37.63% increase in precision and 23.13% increase F1-score but also consumes 93.5% less tokens and 87.3% shorter time.

Keywords—Privacy Alignment, Privacy Testing, Large Language Models, Knowledge Graph, Static Analysis

1. INTRODUCTION

Smartphone apps have transformed modern life, influencing nearly every aspect of human activity ranging from daily purchases to civic engagement. These apps typically justify their extensive data collection practices as necessary for service improvement, however, recently there is a rising concern in the misuse of these data. While developers are required to disclose their data practices through privacy policies, numerous studies have highlighted significant gaps between these formal declarations and the apps’ actual data handling behaviors [1]. Privacy policy-code consistency check has emerged as a significant research stream [2], [3], where most studies employ static analysis tools like FlowDroid for this task [1], [4]. Recent work has explored LLM-based approaches for semantic comparison [5], typically implementing a pipeline that: 1) extracts events and data flows, 2) identifies relevant methods, and 3) computes policy-code similarity. However, the exclusive LLMs-based approach suffers from two fundamental limitations: 1) Over-alignment bias in LLMs frequently generates false positives by fabricating policy-code relationships, reducing checking accuracy. 2) End-to-end LLMs-based processing requires iterative prompting sequences, resulting in higher computational costs. To address this problem, this paper proposes a hybrid approach that combines: 1) knowledge graph-based deterministic checking to avoid false positives caused by over-alignment bias, and 2) LLMs focusing exclusively on preliminary semantic analysis prior to the final comparison, reducing the prompting.

2. PROPOSED HYBRID APPROACH

Figure 1 presents an overview of the hybrid approach, realized through coordinated interaction among three core components: a Policy Reader, a Leak Extractor, and a Consistency Checker.

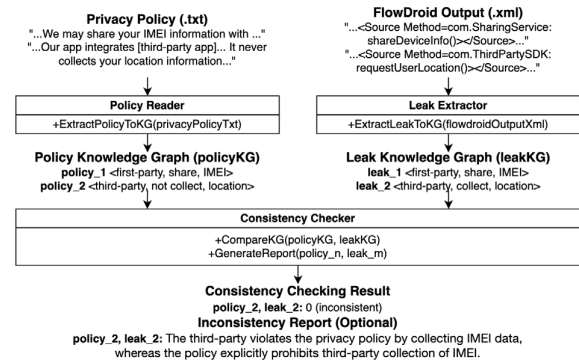


Figure 1. Overview of Hybrid Approach

Policy Reader: The policy reader employs an LLM to transform the natural language-based privacy policies into a structured knowledge graph (i.e., *policyKG*). Each *policyKG* triple follows the format $\langle actor, action, data \rangle$, where: *actor* denotes the responsible entity as either first-party (i.e., the app) or third-party (e.g., SDKs), *action* represents data operations, classified as *collect*, *share*, or their negation via LLM semantic mapping, and *data* indicates the processed user data type, standardized through LLM mapping according to a list containing 14 common data types, referring to [2].

Leak Extractor: The leak extractor extracts methods containing potential violations by processing FlowDroid’s XML output, which encodes static analysis results of sensitive data flows in android apps. Through LLM semantic analysis, it constructs a leak knowledge graph (i.e., *leakKG*) that mirrors the *policyKG* structure, enabling direct consistency checking.

Consistency Checker: The consistency checker executes the checking algorithm shown in Algorithm 1 to detect inconsistency between *policyKG* and *leakKG* triples. For each identified inconsistency (i.e., flagged as *False*), the consistency checker optionally invokes an LLM to generate a natural language inconsistency report, providing a detailed explanation of how the app’s actual behavior violates its privacy policy.

Algorithm 1 Consistency Checking Algorithm

```

1: Input: policyKG, leakKG
2: for each  $l \in leakKG$  do
3:   for each  $p \in policyKG$  do
4:     if  $p.actor == l.actor \wedge p.data == l.data$  then
5:       if  $p.action == l.action$  then
6:         return True,  $l$ ,  $p$ 
7:       else if  $p.action == (!l.action)$  then
8:         return False,  $l$ ,  $p$ 
9:       end if
10:    end if
11:  end for
12: return False,  $l$ , Null
13: end for

```

3. PRELIMINARY EVALUATION

This preliminary evaluation is driven by the following two Research Questions (RQs): **RQ1:** Compared to the pure LLMs approach, does the hybrid approach provide higher accuracy? **RQ2:** Compared to the pure LLMs approach, does the hybrid approach reduce computational costs in time and tokens?

3.1 Experiment Settings

Raw data: The experiment uses FlowDroid-analyzed outputs from 7 Android apps with their privacy policies, supplemented by author-constructed test cases (totaling 17 XML files and 23 policies) to ensure full coverage of common data types in [2].

Ground Truth: The ground truth was established through dual-author annotation with consensus validation.

Baseline: The proposed hybrid approach is compared against the pure LLM baseline as introduced in Section 1 [5]. LLMs used for this experiment are instances of DeepSeek-V3 [6].

Evaluation Metrics: To ensure a systematical evaluation, precision rate, recall rate, and F1 score, are employed to assess the accuracy in RQ1, while end-to-end execution time and token usage are used to quantify computational costs in RQ2.

3.2 Experiment Results and Discussion

TABLE I
COMPARISON IN PRECISION, RECALL, AND F1

Metrics	Baseline	Hybrid
Precision (%)	42.37	80.00
Recall (%)	80.65	77.42
F1 (%)	55.56	78.69

As demonstrated in Table I, the hybrid approach achieves significant improvements over the baseline, showing 37.63% higher precision and 23.13% better F1-score while maintaining competitive recall (77.42%). In terms of computational costs, as shown in Table II, the hybrid approach outperformed with 93.5% reduction in token usage and 87.3% faster processing time from input to binary output. The advantage in token usage even remains consistent with taking the tokens for generating inconsistency reports (as exemplified in Figure 2) into account.

TABLE II
COMPARISON IN TOTAL TOKEN AND TIME CONSUMPTION

Cost types	Baseline approach	Hybrid w/o explanation	Hybrid w/ explanation
Prompt token	381134	23668	60650
Completion token	2674	1432	33888
Total token	383808	25100	94538
Time (sec)	1625	207	2551

Xml 5, Policy 2:

Explanation: The app violates the privacy policy by collecting IMEI data as a first party, whereas the policy explicitly prohibits first-party collection of IMEI.

Figure 2. Example of Generated Inconsistency Report

The experimental results demonstrate that the proposed hybrid approach successfully addresses two fundamental limitations in the pure LLM approach. First, replacing LLMs-based semantic similarity matching with the knowledge graph-based deterministic checking managed to eliminate numerous false positives without significantly compromising recall rate by avoiding over-alignment bias. Second, by restricting LLMs to preliminary semantic analysis prior to final comparison, computational costs in terms of both time and tokens are significantly reduced through minimized interactions with LLMs.

4. CONCLUSION AND FUTURE WORKS

This paper presents a novel hybrid approach that addresses key limitations of low accuracies and high computational costs in LLMs-based privacy policy-code consistency checking by integrating knowledge graph-based deterministic checking and LLMs exclusively for preliminary semantic analysis. Preliminary results demonstrate significant improvements in both precision (increased by 37.63%) and F1-score (increased by 23.13%) while reducing computational costs (93.5% fewer tokens and 87.3% shorter time). Future work should include: 1) regulatory compliance assessment against legal standards like GDPR, and 2) actionable repairing guidance for detected violations, enabling fully privacy policy-code alignment.

REFERENCES

- [1] Y. Javed *et al.*, “A systematic review of privacy policy literature,” *ACM Computing Surveys*, 2024.
- [2] Z. Tan *et al.*, “Ptdroid: Detecting violated user privacy disclosures to third-parties of android apps,” in *ICSE*, 2023.
- [3] K. Zhao *et al.*, “Demystifying privacy policy of third-party libraries in mobile apps,” in *ICSE*, 2023.
- [4] S. Arzt *et al.*, “Flowdroid: Precise context, flow, field, object-sensitive and lifecycle-aware taint analysis for android apps,” *ACM sigplan notices*, 2014.
- [5] G. Morales *et al.*, “A large language model approach to code and privacy policy alignment,” in *2024 SANER*, 2024.
- [6] DeepSeek, “Deepseek-v3: Advanced language model api,” <https://platform.deepseek.com>, 2024.