Poster: *LLMalware*: An LLM-Powered Robust and Efficient Android Malware Detection Framework

Zijing Ma
The Hong Kong Polytechnic University
Hong Kong, China
zijing.ma@connect.polyu.hk

Leming Shen
The Hong Kong Polytechnic University
Hong Kong, China
leming.shen@connect.polyu.hk

Xinyu Huang
The Hong Kong Polytechnic University
Hong Kong, China
unixy-xinyu.huang@connect.polyu.hk

Yuanqing Zheng*
The Hong Kong Polytechnic University
Hong Kong, China
csyqzheng@comp.polyu.edu.hk

ABSTRACT

Android malware pose severe threats to the mobile application ecosystem. Although well-trained malware detection models can initially achieve satisfactory performance, they struggle with unseen Android apps constantly emerging over time, which is known as the concept drift problem. Previous methods frequently collect and label new apps to update the aging models. This process, however, necessitates domain knowledge and incurs prohibitive retraining overhead. To address this problem, this paper presents *LLMalware*, which integrates three novel technical components. First, we propose *full-spectrum automated feature extraction*, which automatically extracts diverse malware features from various detection models. Next, we develop *cohesive feature fusion*, which combines these features to build effective representations for robust malware detection. Lastly, we devise *agile knowledge update* to enable efficient online malware detection via an LLM-based automated agent and a dynamically maintained malware knowledge base. Extensive experiments demonstrate *LLMalware* can mitigate concept drift with an average improvement of approximately 10% in F1-score over state-of-the-art baselines.

CCS CONCEPTS

• **Security and privacy → Malware and its mitigation**.

KEYWORDS

Malware Detection, Metric Learning, Large Language Model

ACM Reference Format:
Zijing Ma, Leming Shen, Xinyu Huang, and Yuanqing Zheng. 2025. Poster: *LLMalware*: An LLM-Powered Robust and Efficient Android Malware Detection Framework. In *Proceedings of the 2025 ACM SIGSAC Conference on Computer and Communications Security (CCS '25), October 13–17, 2025, Taipei, Taiwan.* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/3719027.3760709

*Corresponding author.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
CCS '25, October 13–17, 2025, Taipei, Taiwan
© 2025 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-1525-9/2025/10.
https://doi.org/10.1145/3719027.3760709

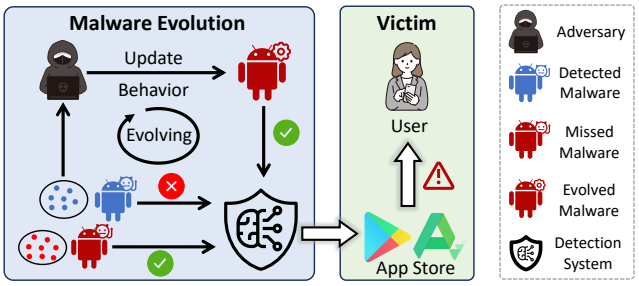


Figure 1: Limitations of existing malware detection methods. The unseen and evolved apps can evade the detection systems and appear in app stores, posing significant risks to users.

1 INTRODUCTION

Android malware pose severe threats to users with privacy leakage, data theft, and even financial losses. As reported by AV-TEST [1], nearly 35 million Android malware have been detected by June 2024, with a growth rate of nearly 100,000 per month. Manually analyzing and labeling such large amounts of malware requires substantial human efforts and incurs prohibitive costs. Existing malware detection methods train various machine learning models to extract and analyze features from static analysis and dynamic analysis.

Nonetheless, these machine learning models are inherently susceptible to the *concept drift* problem, as illustrated in Figure 1. As malware keep evolving in order to evade detection, current models quickly become obsolete with significant performance degradation over time. Moreover, an individual detection method can effectively identify a certain type of malware but may struggle with others. This is because different detection models tend to focus on distinct characteristics of known malware. Thus, solely relying on a trained detection model potentially suffers from the risk of overlooking unseen malware. *The evolving nature of malware motivates us to develop a detection system with enhanced robustness against the concept drift problem.*

To mitigate the concept drift problem, one possible approach is to fuse the static and dynamic features to fully exploit the orthogonal detection capabilities of diverse detection methods. This approach, however, requires substantial human efforts and domain expertise: First, the reports generated by dynamic analysis typically contain vast amounts of information about apps' routine operations, with only a small portion capturing essential malware-related behaviors. Moreover, identifying and extracting relevant features from these

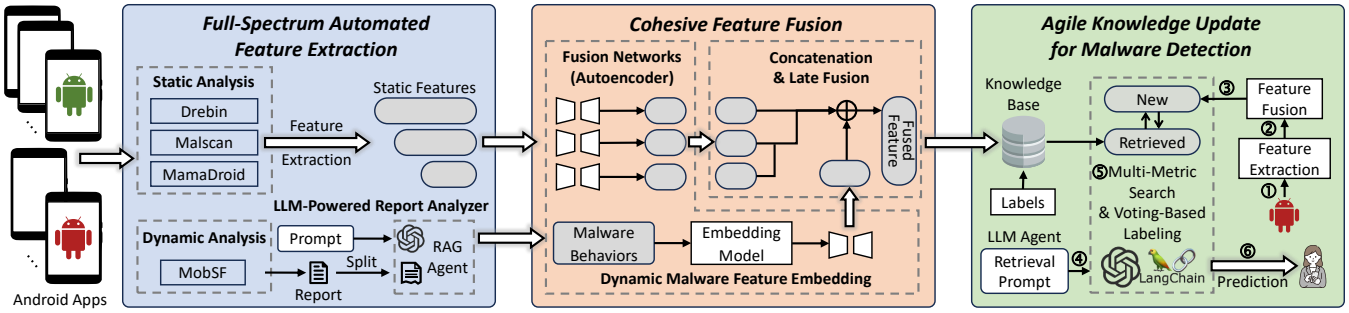


Figure 2: The overall workflow and the key technical components of *LLMalware*.

reports requires highly specialized domain expertise, making such a detection process prohibitive and impractical for large-scale real-world deployment. Second, the evolving nature of malware often introduces unforeseen behavioral mutations, necessitating adequate and timely adaptation of detection models to stay effective.

To tackle the concept drift problem, this paper presents *LLMalware*, an LLM-powered malware detection framework that leverages the latest advances of LLMs to improve the robustness and efficiency of malware detection. *LLMalware* automates analysis report interpretation using LLMs and extracts useful information without developer intervention. This approach substantially improves the efficiency of report analysis. Moreover, *LLMalware* develops a fusion network to integrate multiple features extracted from diverse detection models coherently. The feature fusion can effectively improve the detection robustness against the concept drift problem. *LLMalware* further transforms the conventional malware detection process to on-demand and agile knowledge updates of the malware knowledge base, which facilitates the continuous evolution of our detection models at significantly low costs.

We implement *LLMalware* and carry out extensive evaluations with more than 130,000 Android apps across 6 years. Results show that *LLMalware* significantly outperforms state-of-the-art (SOTA) baselines when facing new types of malware with substantial concept drift. Specifically, *LLMalware* achieves an 8.3% higher F1-score and a much lower aging rate, with cumulative gaps reduced by 210% compared to baselines.

2 SYSTEM DESIGN

Figure 2 illustrates the workflow of *LLMalware*, comprising three key technical components: *full-spectrum automated feature extraction*, *cohesive feature fusion*, and *agile knowledge update for malware detection*.

2.1 Full-Spectrum Automated Feature Extraction

LLMalware combines static analysis (*i.e.*, Drebin [3], MamaDroid [6], Malscan [7]) and dynamic analysis (*i.e.*, MobSF [2]) to extract diverse features from the apps. For dynamic analysis, MobSF records app behaviors during runtime and generates detailed reports. To automate the extraction of relevant behavior information from the reports, *LLMalware* leverages LLMs to interpret and analyze these reports, significantly reducing human efforts. Specifically, as shown in Figure 3, *LLMalware* first leverages a section-aware report chunking method to split the reports by sections, reducing the context length for LLM processing. It then applies a chain-of-thought

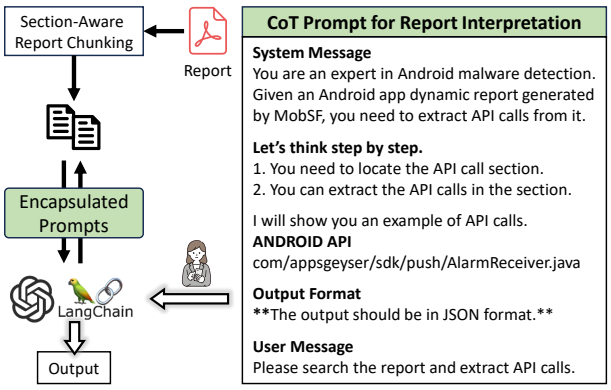


Figure 3: The workflow of LLM-powered automatic report analyzer.

(CoT) prompt to guide the LLM step-by-step in extracting detailed behavior-related information in the reports (*e.g.*, API calls). An example of behavior-related information is included in the prompt for LLM interpretation. This approach, combined with carefully designed prompts and one-shot in-context learning, enables the LLM to efficiently process complex malware analysis reports and transform them into numerical embeddings for further fusion, thus enhancing malware detection robustness without manual intervention.

2.2 Cohesive Feature Fusion

A naive fusion approach is to directly concatenate features from different methods. However, this approach results in feature misalignment, where longer feature vectors overshadow shorter ones, and the curse of dimensionality, which can cause overfitting. To overcome this, *LLMalware* designs a cohesive feature embedding network that employs an autoencoder to compress and align the feature. This ensures that each feature set contributes fairly during fusion. Specifically, the backbone of the network is a hybrid CNN-Transformer structure for the encoder to capture both local and global patterns. The encoder compresses the features into a low-dimensional latent space, while a decoder reconstructs the original features to ensure crucial information is retained. A hybrid loss function combining MSE loss

$$\mathcal{L}_{MSE} = \frac{1}{N}\sum_{i=1}^{N} \|f_i - \hat{f}_i\|_2^2 \tag{1}$$

and triplet loss

$$\mathcal{L}_{Tri} = \frac{1}{N}\sum_{i=1}^{N} \max(\|f_{a_i} - f_{p_i}\|_2^2 - \|f_{a_i} - f_{n_i}\|_2^2 + m, 0) \tag{2}$$

is used to minimize reconstruction errors while optimizing the distance of features, enhancing feature discrimination between malicious and benign apps. For the MSE loss, N is the number of training samples, f_i and $\hat{f}_i$ are the original input features and the reconstructed features, respectively. For the triplet loss, a_i, p_i, and n_i denote the i-th *anchor*, *positive*, and *negative* sample, respectively. After training, late fusion is applied to combine the compressed features from the different detection methods. This approach compensates for weaknesses in any single detection method, and enhances malware detection performance. The fused features are then stored in a knowledge base for future detection tasks.

2.3 Agile Knowledge Update

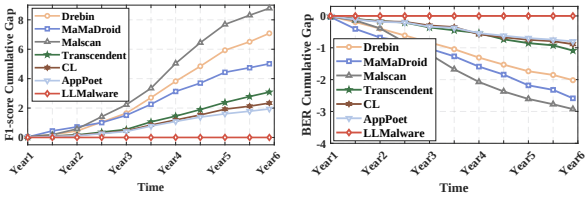
LLMalware introduces an agile knowledge update to tackle the rapidly evolving nature of Android malware, enabling frequent updates with minimal overhead. The system builds a feature-centric knowledge base that stores fused features and corresponding labels, acting as external memory for an LLM-based agent. The agent automatically uses a multi-metric search strategy to retrieve the most similar features from the knowledge base, improving detection without frequent model retraining. This allows for on-demand knowledge evolution, where the agent progressively incorporates new malware data, enhancing the detection performance of previously unseen threats. Specifically, the knowledge base is initialized with fused features from known apps and can be dynamically updated with new samples, enabling continuous malware detection evolution. To mitigate the issues of varying representation quality, the agent employs a multi-metric search using cosine similarity, Manhattan distance, and Euclidean distance to improve retrieval accuracy. A majority voting mechanism determines the final label based on the retrieved feature labels and returns to the agent, ensuring robust classification. The system selects the top K new samples with lower similarity features to update the knowledge base.

3 IMPLEMENTATION & EVALUATION

3.1 Experimental Setup

We implement *LLMalware* using the PyTorch 2.3.0 framework. We test *LLMalware* on a dataset that includes over 130,000 apps from the past six years. GPT-4 is selected to evaluate the impact on the performance of *LLMalware*. We compare *LLMalware* with 6 baselines, including Drebin [3], MamaDroid [6], Malscan [7], Transcendent [4], CL [5], AppPoet [8].

Overall performance. We evaluate the performance of *LLMalware* across different periods and measure the F1-score and BER cumulative gap of different malware detection methods, denoted as $CG_{baseline}(t) = \sum_{i=1}^{t}(F_{LLMalware}(i) - F_{baseline}(i))$, where $F_{LLMalware}(i)$ and $F_{baseline}(i)$ are the F1-score of *LLMalware* and baselines. Specifically, we use data collected in Year 0 as the training set and data in Year 1 - Year 5 as the test set. The test set is divided into multiple subsets by months to assess the influence on the performance over different periods. The results are shown in Figure 4. As shown in Figure 4(a), the cumulative gaps of baselines like Drebin and MaMaDroid increase to 54% and 76%, while the SOTA baselines, including Transcendent, CL, and AppPoet, reach gaps of around 20%, indicating a significant model deterioration compared



(a) F1-score cumulative gap. (b) BER cumulative gap.

Figure 4: The overall performance of *LLMalware* and the baselines over time.

to *LLMalware*. By Year 5, the gaps of Transcendent, CL, and AppPoet further expand to around 210%, highlighting their declining robustness and the model aging problem over time. Moreover, as shown in Figure 4(b), the BER cumulative gaps of all baselines increase. For example, the gaps of baselines such as MaMaDroid and CL increase from 70% and 20% in Year 0 to 230% and 79% in Year 5, indicating significant misclassification.

4 CONCLUSION AND FUTURE WORK

We propose *LLMalware*, an LLM-powered robust and efficient Android malware detection framework against the concept drift problem. The key insight of *LLMalware* is to fully exploit the orthogonal detection capabilities of diverse detection methods. To achieve this, *LLMalware* develops and incorporates three key technical components: ① *full-spectrum automated feature extraction* to extract static and dynamic features, ② *cohesive feature fusion* to constructively integrate features of distinct detection methods, and ③ *agile knowledge update* for malware detection to continuously update the knowledge of LLM for unseen app detection. Experimental results demonstrate the effectiveness of the proposed feature extraction and fusion approach. Such technical components jointly help mitigate the urgent issue of concept drift in malware detection. In the future, we aim to realize finer-grained malware family classification such that defenders can process the malware strategically.

ACKNOWLEDGMENTS

We sincerely thank the anonymous reviewers for their constructive comments and invaluable suggestions. This paper is supported by Hong Kong GRF under Grant No. 15206123 and 15211924.

REFERENCES

[1] AV-TEST. https://www.av-test.org/en/.
[2] MobSF. https://github.com/MobSF/Mobile-Security-Framework-MobSF.git.
[3] Daniel Arp, Michael Spreitzenbarth, Malte Hubner, Hugo Gascon, Konrad Rieck, and CERT Siemens. 2014. Drebin: Effective and explainable detection of android malware in your pocket. In *21st Annual Network and Distributed System Security Symposium (NDSS)*. 23–26.
[4] Federico Barbero, Feargus Pendlebury, Fabio Pierazzi, and Lorenzo Cavallaro. 2022. Transcending transcend: Revisiting malware classification in the presence of concept drift. In *2022 IEEE Symposium on Security and Privacy (S&P)*. 805–823.
[5] Yizheng Chen, Zhoujie Ding, and David Wagner. 2023. Continuous learning for android malware detection. In *32nd USENIX Security Symposium (USENIX Security 23)*. 1127–1144.
[6] Enrico Mariconti, Lucky Onwuzurike, Panagiotis Andriotis, Emiliano De Cristofaro, Gordon J. Ross, and Gianluca Stringhini. 2017. MaMaDroid: Detecting Android Malware by Building Markov Chains of Behavioral Models. In *24th Annual Network and Distributed System Security Symposium (NDSS)*.
[7] Yueming Wu, Xiaodi Li, Deqing Zou, Wei Yang, Xin Zhang, and Hai Jin. 2019. Malscan: Fast market-wide mobile malware scanning by social-network centrality analysis. In *34th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. 139–150.
[8] Wenxiang Zhao, Juntao Wu, and Zhaoyi Meng. 2025. Apppoet: Large language model based android malware detection via multi-view prompt engineering. *Expert Systems with Applications* 262 (2025), 125546.