POLITECNICO DI TORINO
Repository ISTITUZIONALE

Assessing the Effectiveness of an LLM-Based Permission Model for Android

Original
Assessing the Effectiveness of an LLM-Based Permission Model for Android / Milanese, R., Guerra, M., Daniele, M., Fabbrocino, G., Fasano, F.. - 2:(2025), pp. 36-47. (11th International Conference on Information Systems Security and Privacy, ICISSP 2025 Porto (PRT) February 20-22, 2025) [10.5220/0013128100003899].

Availability:
This version is available at: 11583/2997932 since: 2025-02-27T08:48:23Z

Publisher:
Science and Technology Publications

Published
DOI:10.5220/0013128100003899

Terms of use:

This article is made available under terms and conditions as specified in the corresponding bibliographic description in the repository

Publisher copyright

(Article begins on next page)

13 September 2026

Assessing the Effectiveness of an LLM-Based Permission Model for Android

Roberto Milanese[1,2,3][a], Michele Guerra[1,2][b], Michele Daniele[1,2], Giovanni Fabbrocino[1][c] and Fausto Fasano[1,2][d]

[1]*Department of Biosciences and Territory, University of Molise, Italy*

[2]*Mosaic Research Center, University of Molise, Italy*

[3]*Department of Control and Computer Engineering, Politecnico di Torino, Italy*

roberto.milanese@polito.it, {michele.guerra, giovanni.fabbrocino, fausto.fasano}@unimol.it,

Keywords: LLM, MLLM, AI, UI, Security and Privacy, Application Security, App Permission, Android, Android Permission Model.

Abstract: With the widespread use of mobile apps, users are frequently required to make decisions about app permissions. However, most people lack the knowledge to fully understand the consequences of their choices. Apps often request access to sensitive data, sometimes in the background and without clear justification, making users the weakest link in the security chain. This inadvertently exposes them to privacy breaches and malicious activities. Despite improvements, Android's permission system remains inadequate in helping users make informed, real-time decisions. In this paper, we investigate the feasibility of an approach to address this critical gap that leverages the power of Large Language Models (LLMs) and Multi-Modal Large Language Models (MLLMs). We propose a system that dynamically evaluates permission requests by analyzing the full context of the UI on mobile app screens. Unlike traditional permission models, which rely on static rules or user input, our approach integrates seamlessly into existing systems, interpreting the relationships between UI elements and requested permissions to make aware, real-time decisions about whether the request is necessary or potentially harmful. Our evaluation on 123,552 UI screens from 70 popular Android apps revealed promising results, reaching 81% accuracy. By reducing the cognitive load on users and offering real-time protection against security threats or supporting a more informed choice by the user, our system can enhance existing permission models, providing a step towards smarter and safer mobile ecosystems. This solution paves the way for integrating intelligent permission systems that proactively shield users from risks while ensuring data security without overwhelming them with complex decisions.

1 INTRODUCTION

Mobile applications today collect vast amounts of sensitive data, making the privacy and security of users a critical concern. As the dominant mobile platform, Android relies on a permission-based system to regulate access to personal data and device resources. Despite continuous improvements, such as the runtime permissions model, users remain ill-equipped to make informed decisions about granting permissions, frequently exposing themselves to privacy breaches and malicious behaviors (Felt et al., 2012). This vulnerability makes users the weakest link, often unknowingly allowing apps to misuse their data.

At the heart of this problem is the complexity of the modern app ecosystem. Malicious apps, as well as non-malicious apps involved in user profiling, often exploit Android's permission system to enable invasive behaviors. These behaviors include tracking users, collecting sensitive information such as location or contact data, and triggering unauthorized actions in the background. For example, (Felt et al., 2012) demonstrated how apps can use excessive permission requests to track users or send unauthorized messages, undermining user privacy and security. Faced with unclear permission requests, users frequently approve them without fully understanding

[a] https://orcid.org/0009-0009-8758-753X
[b] https://orcid.org/0009-0005-9990-234X
[c] https://orcid.org/0000-0002-4918-5398
[d] https://orcid.org/0000-0003-3736-6383

Milanese, R., Guerra, M., Daniele, M., Fabbrocino, G. and Fasano, F.
Assessing the Effectiveness of an LLM-Based Permission Model for Android.
DOI: 10.5220/0013128100003899
Paper published under CC license (CC BY-NC-ND 4.0)
In *Proceedings of the 11th International Conference on Information Systems Security and Privacy (ICISSP 2025) - Volume 2*, pages 36-47
ISBN: 978-989-758-735-1; ISSN: 2184-4356
Proceedings Copyright © 2025 by SCITEPRESS – Science and Technology Publications, Lda.

the risks. This kind of behavior is well known and described as *warning fatigue* (Akhawe and Felt, 2013). Existing solutions, including alternative permission models and recommendation systems (Nauman et al., 2015), runtime data leaks prevention (Ciobanu et al., 2020), and exact permission usage detection (Fasano et al., 2020) have aimed to mitigate these risks. However, these approaches often either rely heavily on user input or fail to adapt dynamically to the context of app interactions.

In this paper, we use Multi-modal Large Language Models (MLLMs), which are capable of processing both textual and visual inputs, providing a comprehensive understanding of the context of user interactions with mobile applications. For simplicity, throughout the remainder of the paper, we will refer to both Large Language Models (LLMs) and MLLMs as LLMs, unless the distinction between the two is specifically relevant. Our contribution is not a new permission model per se, but rather a powerful enhancement that can be integrated into existing models or tools to make permission decisions more informed and context-aware. Specifically, the approach leverages LLMs to analyze both the textual and visual components of a UI screen, interpreting the relationships between UI elements and the interactions leading to permission requests. This contextual analysis enables the system to recommend permissions that align with the functionality presented on the screen. Precisely, we executed real-world Android applications, manually analyzed the contexts in which permissions were requested, and created a dataset of these contexts. Using this dataset, LLMs dynamically assess whether a permission request is necessary and appropriate based on the entire user interface (UI) screen and the interactions occurring within it. Additionally, our approach is specifically designed to detect potentially malicious behaviors by identifying inappropriate permission requests. This capability allows the system to flag high-risk permissions that may otherwise be granted unknowingly by users, offering a layer of real-time protection against abusive apps.

A key innovation of our approach is that all decisions made by the LLMs are stored in this dataset, which can be further used by existing tools or integrated into dynamic analysis platforms like RPC-DROID (Guerra. et al., 2023). Automating these decisions could supports the system to make more informed choices about permissions without requiring user intervention. This enables a more secure and privacy-conscious permissions management, addressing a critical gap in current solutions that often fail to contextualize permission requests in real-time.

The use of LLM, specifically `GPT-4 Vision`, is particularly advantageous in this context due to its ability to process both text and visual information, providing a comprehensive understanding of the app behavior. Moreover, recent studies have shown that LLMs, such as `GPT-4`, significantly contribute to enhancing security by detecting vulnerabilities and preventing malicious activities. For instance, LLMs, including `GPT-4`, have been successfully employed to identify malware and generate security tests, achieving higher accuracy compared to traditional tools (Yao et al., 2024).This positions `GPT-4 Vision` as a valuable asset in our approach to improving permission management on mobile platforms, where rapid and accurate detection of security risks is crucial.

To demonstrate the effectiveness of our approach, we conducted a comprehensive empirical study, executing 123,552 UI screens from 70 real-world Android applications. The data used for app functionalities and permission requests were manually reviewed and validated by two experts to ensure consistency in the dataset. We then compared the predictions made by LLMs against this expert validation, assessing the models' ability to make accurate permission decisions based on the context in which the requests were made. This dataset of contexts and LLM decisions forms the basis for further integration into existing security frameworks, offering a scalable solution for automated, context-aware permission management. Our empirical evaluation demonstrated the model's potential for real-time protection in everyday mobile use. Specifically, our results showed that the model consistently predicted permissions with an overall accuracy of 81%, and for high-risk resources such as the microphone (`RECORD_AUDIO`), accuracy exceeded 95%. Additionally, we extended our analysis to a set of well-known malicious apps, applying the same experimental procedure. In each case, our approach effectively evaluated permission requests, predicting when they were unnecessary or potentially harmful. The model provided context-aware rationales, helping to explain why certain permissions, such as those for SMS or location access, were inappropriate given the app's functionality. This demonstrates the system's potential to assist users in making informed decisions and prevent the granting of permissions that could lead to harmful behaviors.

In summary, our contributions are twofold:

- We investigated the effectiveness of a novel approach using LLMs to enhance the accuracy and context-awareness of permission decisions, by analyzing the full scope of UI screens and user interactions. This approach reduces unnecessary permission grants, thereby strengthening both secu-

rity and privacy in mobile applications.

- We created a dataset[1] containing 123,552 UI screens from 70 Android apps (46 commercial and 24 open source) and the corresponding UI interactions and requested permissions, which can serve as a foundation for future research in automating permission management.

2 BACKGROUND AND RELATED WORK

2.1 Android Permissions

The Android permission system is a cornerstone of the platform's security architecture, designed to regulate applications' access to sensitive resources and maintain a balance between usability and security (Fang et al., 2014). These resources include user data (e.g., contacts, messages, call history) as well as hardware components (e.g., camera, microphone, GPS). To access them, applications are required to declare the permissions they require in the MANIFEST file. Permissions may either be automatically granted — when deemed low-risk — or require explicit user consent for those categorized as high-risk for privacy and security (Almomani and Khayer, 2020).

Before Android 6.0, users were forced to accept all requested permissions at install time in an inflexible, *all-or-nothing* approach, where refusing a single permission halted the installation (Wijesekera et al., 2015). This practice raised significant concerns. First, users often felt forced to grant unnecessary permissions just to continue using an application. Second, once installed, users had no way to review or change their choices. These limitations increased the potential for misuse of sensitive data. The introduction of runtime permissions in Android 6.0 was a game changer. Instead of requesting permission during installation, applications now prompt users for permission when they attempt to access a resource. Moreover, users gained the ability to revoke previously granted permissions in the system settings, giving them continuous control over their data and resource access (Zhauniarovich and Gadyatskaya, 2016). Although this increases transparency and gives users a clearer understanding of what they agree to, there remains a crucial limitation. Permissions are still granted for the entire application, rather than for the specific features that require them. Consequently, applications may access resources like the microphone for messaging purposes but later use them for other features without user awareness (Malviya et al., 2023), leading to potential privacy concerns.

[1]https://github.com/mosaic-research-center/ICISSP-25

2.2 Large Language Models

LLMs represent a breakthrough in deep learning, particularly for tasks involving natural language understanding and generation (Zhao et al., 2023). They are built upon Transformer architectures (Vaswani et al., 2017), employing an encoder-decoder mechanism. The encoder translates the input sequence into fixed-length internal representations through multiple layers, each consisting of multi-headed attention mechanisms and feed-forward networks. The decoder subsequently uses these internal representations to generate output sequences token-by-token, with each decoder layer containing additional multi-head attention sub-layers to handle previously generated tokens (Rush, 2018). The *Large* in LLMs denotes not only the sheer number of parameters in these models, but also the extensive volume of textual data used to train them (Minaee et al., 2024). A remarkable feature of LLMs is their *emergent abilities*, which allow them to solve complex problems they were not explicitly trained for, through mechanisms like *in-context generalization*, where a model adapts to new tasks by observing a few examples in natural language (Dong et al., 2024).

MLLMs extend the capabilities of traditional LLMs by integrating multiple data types beyond text to include images, audio, and video (Caffagni et al., 2024). For instance, `GPT-4` (Achiam et al., 2023), with its vision capabilities, can process both text and images as inputs, allowing for a more holistic understanding of information. This multi-modality enables MLLMs to outperform single-modal models in complex tasks and improve common-sense reasoning across diverse contexts (Wu et al., 2023).

2.3 Related Work

2.3.1 Permission Recommender Systems

Over time, a number of studies have focused on understanding user interaction with permissions. (Felt et al., 2012) pointed out that users pay little attention to permission requests and show little understanding of the risks associated with granting certain permissions. (Bonné et al., 2017) found that, on average, users grant permissions six times more often than they deny them, and are more likely to revoke permissions in the system settings than when prompted at runtime. Many users choose to grant permissions even when they seem unnecessary, often because they

fear that the application will stop working, or simply want to quickly dismiss the pop-up. This phenomenon is known as *warning fatigue* (Akhawe and Felt, 2013). Privacy indicators, small colored dots that appear in the status bar when an application accesses the camera, microphone or user location, also proved ineffective. (Guerra et al., 2023) showed that these notifications are least helpful when they are most needed, such as when resource usage occurs unexpectedly. This is largely because privacy indicators are designed to be minimally intrusive, and as a result are often almost invisible (Guerra et al., 2024).

For these reasons, several approaches have been proposed to assist users in choosing which permissions to grant, and even to automate this process. (Liu et al., 2014) conducted a study of the privacy settings of 4.8 million devices. They found that while individual user preferences may appear to differ, it is possible to derive a small number of privacy profiles. Based on these profiles, they were able to predict with over 87% accuracy whether or not the user would grant permission to a specific application. A fully automated approach based on these profiles can automatically decide on permission requests. Other approaches learn from the user's previous choices without explicitly classifying them into profiles. (Oglaza et al., 2017) developed a permission manager that allows users to set policies to control access to sensitive resources on their device. The tool can automatically suggest high-level policies based on the user's past preferences. Using a dataset of 8,500 runtime permission decisions collected from 41 real-world users, (Olejnik et al., 2017) trained an ML classifier that accurately predicts user choices. This reduced the number of incorrect decisions by approximately 50% compared to the native Android permission model. In addition, their SMARTPER tool introduces a third *Obfuscate* option, which is also integrated into the privacy preference function that decides which permissions to grant.

In contrast, (Rashidi et al., 2018) developed a crowdsourcing-based framework for permission management that offers recommendations based on expert decisions. This enables even inexperienced users to make safer, low-risk choices with a high accuracy. (Gao et al., 2019) proposed an NLP-based method that mines app descriptions to analyze the relationship between an application's features and required permissions, so that users receive recommendations without relying on their past decisions. While these solutions help users make informed choices, permissions are still managed at a coarse-grained level.

2.3.2 UI Elements and Permission Decisions

User interface design has a direct impact on how users perceive an application's access to their personal data and influences their decisions. (Micinski et al., 2017) analyzed 150 Android applications to understand whether access to sensitive resources is consistently tied to user interaction. They found that some resources are almost always used interactively, following a UI event or when they are the main focus of the screen. This is the case for the camera and microphone, among others. In contrast, other resources are mainly used in the background. For example, the location and call history fall into this category. In a study involving over 950 participants, the authors confirmed that users are more likely to expect resource access after an interaction. The study also revealed that when users first encounter an interactive resource usage, they do not expect future background access. (Roesner et al., 2012) introduced the concept of user-driven access control, where permissions are granted as part of user actions. In this model, sensitive resources are linked to permission-granting UI elements. When users interact with these gadgets, their intent is captured, and the system configures the access control policy accordingly. However, implementation would require changes at the operating system level, making it challenging to apply in practice.

Some interesting approaches have focused on automatically mapping permissions to UI elements to overcome this obstacle. One of these is ICONINTENT by (Xiao et al., 2019), which relies on static analysis and image classification to match icons to eight categories of sensitive data, outperforming previous solutions that relied on text analysis. (Xi et al., 2019) combined both techniques to achieve better detection accuracy. Their tool DEEPINTENT exploits deep learning and program analysis to associate UI elements with specific resources and then automatically detect discrepancies between perceived and actual application behavior. Analyzing over 1,000 top-ranked apps in the Play Store, (Li et al., 2023) found this behavior in 28% of them. This shows that misleading icons are common in real-world apps. Although these solutions are a good starting point, they never evaluate the UI as a whole. A more comprehensive approach would provide a better level of contextualization.

3 PROPOSED APPROACH

Our approach analyzes entire application UI screens to extract rich contextual information about the feature they represent, based on both the visual and tex-

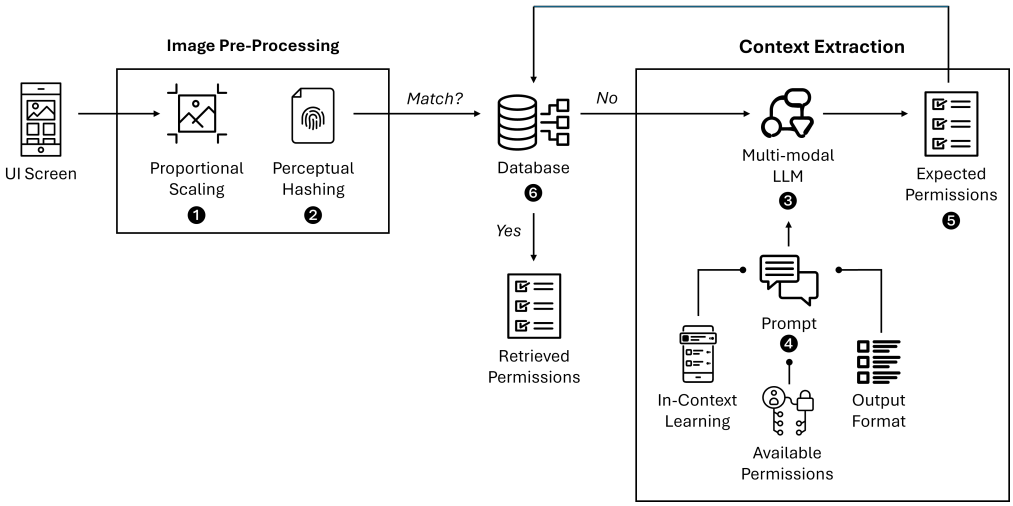

Figure 1: Overview of our proposed approach.

tual elements they contain. This context is then used to predict the permissions required by that specific feature. Unlike traditional methods that examine UI elements in isolation, our approach captures the interactions between components on the screen, providing a holistic view of how these interactions influence permission requirements and potential security risks.

We selected `GPT-4 Vision` for its ability to process both textual and visual inputs. While newer models, such as OpenAI's `o1`, have been released, they do not currently support multi-modality. In contrast, `GPT-4 Vision` excels at interpreting images, allowing it to analyze the relationships between visual and textual elements in UI screens, which is critical for accurate permission prediction. In addition, `GPT-4` has demonstrated superior performance on security-related tasks, such as vulnerability detection and malware identification, compared to traditional approaches (Yao et al., 2024). Leveraging pre-trained LLMs provides significant advantages over building a custom model from scratch, with contextual understanding and decision making capabilities that allow them to generalize well across domains.

Figure 1 illustrates the two primary phases of our solution: (i) *image pre-processing*, where raw screenshots are transformed into analyzable formats, and (ii) *context extraction*, where the processed images, along with a carefully engineered prompt, are input into the LLM to generate context-aware predictions about the necessary permissions.

3.1 Image Pre-Processing

In the first phase, the images of the UI screens taken as input are transformed to be analyzed more effectively.

Screenshots are resized using **proportional scaling ❶** to maintain consistency across different device resolutions and aspect ratios. This ensures that the visual integrity of UI components is preserved for accurate analysis, while shrinking images to reduce computational overhead.

Then, we employed **perceptual hashing ❷** to account for the inherent dynamism of mobile UIs, where elements can change frequently without changing the overall screen context (*e.g.,* an advertisement displayed in a banner at the bottom of the screen may change at any time, yet the application feature remains the same). Unlike cryptographic hashing, where minimal input changes produce vastly different outputs, perceptual hashing generates similar hashes for visually similar images (Sabahi et al., 2018). We set a 3-bit Hamming distance[2] threshold of on perceptual hashes to detect duplicates and near-duplicates. This way, if a UI screen has already been analyzed, the list of expected permissions is retrieved from the history, avoiding the need to prompt the model again.

3.2 Context Extraction

In the second phase, the pre-processed images are used as input for the **LLM ❸** to extract relevant information about the application's feature represented on the UI screen and to generate context-aware suggestions for the permissions that are likely to be required. A crucial aspect of our approach is the ability of the LLM to accurately interpret the context of a UI screen by analyzing it in addition to the user interac-

[2]The Hamming distance between two strings of equal length is the minimum number of substitutions required to change one string into the other

tions leading to that screen, such as button presses or menu selections. This contextual understanding enables the model to associate the relevant permissions with the specific functionality represented on the UI screen, providing recommendations that are both precise and aligned with user behavior.

A carefully crafted text **prompt ❹** is used to instruct the LLM on the task and describe the desired format for the output. The initial prompt was progressively refined through extensive manual effort to achieve an accurate and desired result, in a process known as *prompt engineering* (Ye et al., 2024). The final prompt is shown below. For brevity, the list of available permissions has been omitted.

Final Prompt

You are an expert in security and privacy.

Given a UI screen of an Android application, and knowing that *⟨user interaction, if any⟩*, show me, in this order and following this structure:

- Description: A description of the feature represented in the screen, using no more than 300-350 characters.

- Permissions: A list of permissions you expect to be required by the feature, giving for each the technical name, a description of no more than 6 words, and your confidence level in the prediction.

Example: (CAMERA, Access the device camera, 100%), (RECORD_AUDIO, Record audio from the device microphone, 85%)

- Danger: The level of risk to the user's privacy and security from the use of the expected privileges, both as a percentage and as a description of no more than 6 words.

Example: (10%, Low risk as expected)

You are only interested in runtime permissions because they are high-risk and you can choose to grant them or not.

The permissions you can choose from are: *⟨permissions⟩*.

Rather than relying on traditional fine-tuning, our approach uses **in-context learning**, which allows the model to adapt dynamically to new tasks based solely on the prompt (Dong et al., 2024). This significantly reduces the computational overhead associated with fine-tuning large models and ensures that the model remains flexible and responsive to real-time security needs. We applied a *zero-shot* methodology, instructing the model to infer permission requirements directly from the provided UI screenshots, without explicit task-specific training data. By harnessing the *generated knowledge* technique, the model produces contextual predictions that are both reliable and security-focused (Chen et al., 2024). In addition, we employed *role-prompting*, guiding the model to assume the role of a security and privacy expert, thereby improving the quality and precision of the recommendations (Kong et al., 2024).

Our approach focuses on predicting these high-risk permissions due to their direct implications on privacy and security. The Android 15 operating system defines 311 permissions[3], of which only 40 are classified as high-risk requiring explicit user approval at runtime. By restricting our analysis to this **available permissions** subset, we ensure that the LLM targets the permissions that pose the most significant risk to user data and security, providing users with actionable insights to mitigate potential threats.

We designed the LLM **output format** to be structured and consistent, facilitating subsequent interpretation and analysis. By providing clear instructions and examples within the prompt, we have ensured that the LLM responses are not only reliable but also easily usable for both human review and automated processing. For this task, we ask the model to return:

1. A description of the feature represented in the UI screen. This provides a clear context and helps the model itself to reason.
2. A list of high-risk permissions expected for the feature.
3. A percentage score indicating the perceived risk associated with the expected permissions. This information can help inform the user if the risk is notably high.

Once the model generates its **output ❺**, it is processed and stored in a relational **database ❻**. This includes not only the predicted permissions, but also additional information such as the prompt itself, the user action that led to the specific screen, and all related connections. This allows patterns and insights to be tracked over time, enabling comprehensive evaluations of permission usage across different applications and features. This data is also associated with

[3]https://developer.android.com/reference/android/Manifest.permission

the perceptual hash of the input image. As previously mentioned, when a new screenshot is processed, its hash is compared to existing ones in the database. If a match is found, the corresponding permissions are assigned to the similar UI screen directly, without invoking the model.

We implemented our approach in a Python script. For image pre-processing, we chose the PIL[4] library to manipulate images and the ImageHash[5] library to generate perceptual hashes and identify visually similar UI screens. For context extraction, we employed the `gpt4-vision` model, which combines both image and text processing capabilities in a unified framework. The model was accessed and prompted via the OpenAI Python API[6].

4 EMPIRICAL STUDY AND EVALUATION

To assess the effectiveness of our approach, we conducted a comprehensive evaluation focusing on two key aspects: (i) the model's accuracy in predicting the permissions required by a specific UI screen, and (ii) its ability to discern whether a requested permission is contextually appropriate or potentially malicious. Our investigation was structured around the following research questions:

- **RQ_1:** How effective is an LLM-based approach in interpreting and analyzing UI screens to accurately extract their contextual meaning and recommend permissions to grant for the feature in use?
- **RQ_2:** To what extent can an LLM-based approach effectively prevent malicious behavior by analyzing UI screens, identifying misused permissions, and providing a rationale for denial?

4.1 RQ_1: Identifying Expected Permissions

For **RQ_1**, we evaluated the effectiveness of the approach in identifying high-risk permissions required for specific application features based on UI screenshots. We compared the model's predictions to expert-validated ground truth data, allowing us to measure the alignment between the model's outputs and the permissions that a security expert would reasonably expect in each context.

[4]https://pypi.org/project/pillow

[5]https://pypi.org/project/ImageHash

[6]https://pypi.org/project/openai

4.1.1 Methodology

To answer **RQ_1**, we built a dataset of 123,552 UI screens from 70 Android applications. Our selection included the most popular applications across different categories to ensure a diverse set of real-world use cases. We considered both commercial applications from the Google Play Store and open source applications from F-Droid. To automate the execution of the target applications and capture the screenshots, we used RPCDROID (Guerra. et al., 2023), a dynamic analysis tool that generates random inputs to interact with the UI while recording permission usage data. In this way, for each application, we collected:

- A database of UI events, including user interactions with graphical elements and transitions between application screens. Each event is logged with a unique identifier, event type, timestamp and other relevant data.
- A database of permission requests, including all high-risk permissions used by the application. Each request is logged with a unique identifier, permission name and timestamp.
- A set of UI screens that are captured each time an event occurs or a permission-protected resource is accessed. Each screenshot is linked to the associated record for contextual analysis.

Duplicates in the set of UI screens were filtered as described in Section 3.1. Two experts manually labeled the remaining UI screen in the dataset to determine the expected permissions relevant to its context. Starting with the permission log returned by RPCDROID, they mapped each UI screen to the permissions used by the application, and then considered only those really needed for the feature in use. In addition, based on the event log, they mapped each UI screen to the specific user actions that triggered the behavior. For evaluation, the model's predictions were compared against the expert-labeled ground truth using the following metrics:

- *True Positive (TP)*: The model correctly predicted a permission that was required.
- *True Negative (TN)*: The model correctly excluded a permission that was not required.
- *False Positive (FP)*: The model predicted a permission that was not needed.
- *False Negative (FN)*: The model failed to predict a required permission.

We calculated *accuracy* (ACC), *recall* (true positive rate, TPR), and *specificity* (true negative rate, TNR) to evaluate the model's performance. Higher

Table 1: Performance of expected permissions identification.

Permission	TP	TN	FP	FN	ACC (%)	TPR (%)	TNR (%)
ACCESS_COARSE_LOCATION	20	1,077	7	2	99.2	90.9	99.4
ACCESS_FINE_LOCATION	2,705	15,987	8,200	656	67.9	80.4	66.1
CALL_PHONE	3	48	52	1	49.0	75.0	48.0
CAMERA	369	12,841	1,810	140	87.1	72.5	87.6
READ_CONTACTS	61	2,796	25	7	98.9	89.7	99.1
READ_PHONE_STATE	32	7,519	2	103	98.6	23.7	99.9
READ_SMS	7	336	5	0	98.6	100.0	98.5
RECORD_AUDIO	117	3,378	35	24	98.3	82.9	98.9
Total	**3,314**	**43,982**	**10,136**	**933**	**81.0**	**78.0**	**81.3**

scores indicate better alignment with expert predictions and more reliable permission identification.

4.1.2 Results

Table 1 summarizes the performance across various permissions. It is worth noting that the number of TNs is quite high because it includes the screenshots captured during the execution of applications, regardless of the use of a dangerous privilege. Most of these do not actually need to access any resources and are therefore classified as TNs.

In general, our approach achieved an accuracy of 81.0% in identifying the expected permissions, with a TPR of 78.0% and a *false positive rate* (FPR) of 18.7%. On closer inspection, the prediction accuracy is at or above 90% for most permission types. This is the case for highly critical resources such as microphone, contacts and messages, which also have high TPR.

However, the model's performance for the ACCESS_FINE_LOCATION permission was far less satisfactory, mainly due to a high number of FPs. This could be explained by the fact that many applications use location services in the background rather than interactively, leading the model to predict resource access even when it is not explicitly linked to a visible UI element. Confirming our hypothesis, for most resources that are mainly used interactively, i.e. after a user action on the UI or when they are the main focus of the screen, the *false negative rate* (FNR) is as low as 5%. The only exceptions are the CALL_PHONE and CAMERA permissions, which allow an application to initiate a call without going through the dialer UI, and to take pictures or record video without going through the gallery or the system camera, respectively. For these permissions, the TNR falls below 90%. In the first case, we suspect that the model misinterpreted the intent of this permission, as it is not required when a phone number is automatically dialed after being tapped, which is the typical case. In the second case, it is likely that the model mistook camera access for all those screens that display the gallery's media picker UI. In fact, if the application is not accessing the camera directly, then the corresponding permission is not required, and READ_MEDIA_IMAGES and READ_MEDIA_VIDEO are used instead.

Again, for the READ_PHONE_STATE permission, the performance of our approach is below average. However, this time it is due to the high number of FNs, which reduces the TPR to 23.7%. As with CALL_PHONE, we suppose that the model has no practical knowledge of when this permission is needed, in part because it is rarely used in practice, as our numbers show.

Overall, these results indicate that our approach is effective in predicting most high-risk permissions, particularly those associated with direct user interaction, where prediction accuracy is consistently high. Addressing the high false positive rates in cases where permissions are used in the background will be key to refining the model.

4.2 RQ_2: Detecting Misused Permissions

For **RQ_2**, we evaluated how well the approach helps detect the misuse of high-risk permissions in malicious applications. We compared the model's predictions with expert-validated ground truth data to determine whether all suspiciously used permissions were actually deemed unnecessary by the model.

4.2.1 Methodology

To answer **RQ_2**, we selected 8 Android applications known to conceal malicious behavior. We manually analyzed the logs generated by RPCDROID for each application, focusing on the context in which permissions were requested. This included tracing specific UI events and screens where these permissions were

triggered. Next, we decompiled all of the applications and carefully inspected their source code to uncover any suspicious or inappropriate use of permissions that might indicate malicious intent.

Two experts manually labeled each UI screen collected during this process and identified the permissions used maliciously in the corresponding feature. We then evaluated the model's ability to accurately detect and flag malicious permission usage. Specifically, we checked whether the labeled permissions appeared in the list of permissions that the model considered necessary for the given UI screen. Finally, we asked the model to explain the rationale behind its decision with the prompt shown below.

Rationale Prompt

Why do you think the use of the ⟨*permission*⟩ permission is ⟨*appropriate / not appropriate*⟩ at this time?
Explain your decision using no more than 300 characters.

4.2.2 Results

Table 2: Performance of permissions misuse detection.

Application	# Misuses
com.excelliance.dualaid	3
com.ksa.passbook	1
com.opera.A.install	2
com.opera.B.installer	6
com.ps.yams	1
com.sileria.alsalah	2
com.sohu.inputmethod.sogou	4
com.ydbl.kudou	5
Total	**24**

The results are presented in Table 2. They suggest that the model is effective at both identifying necessary permissions and detecting potential misuse. In every case where we identified a permission as malicious, the model consistently excluded it from the list of permissions deemed necessary for the corresponding UI screen. This further remarks the soundness of our approach to making privacy-preserving decisions.

As a motivating example, we present the case of com.opera.A.install, a rogue Opera browser installer that covertly accesses and reads SMS messages without the user's knowledge. Our analysis revealed that this malicious behavior is triggered when the user taps the "OK" button on the UI screen in Figure 2.

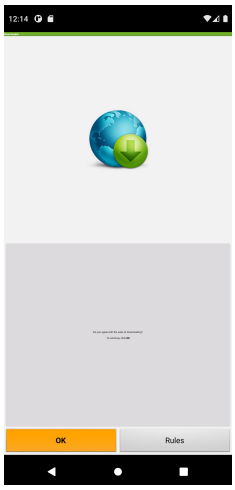

Figure 2: UI screen of the com.opera.A.install application misusing the READ_SMS permission.

The model correctly identified that the READ_SMS permission was unnecessary for downloading a browser and flagged it as potentially harmful, explaining that such a permission could be exploited by malicious apps to secretly subscribe users to premium services.

Rationale Output

The app requests SMS permission, which could charge you via premium services without clear consent for each transaction. It's a common tactic in scam apps.

A key advantage of our approach is its ability not only to detect inappropriate permission requests but also to provide meaningful explanations for why those permissions are suspicious. For example, in the case of the browser installer, the model flagged the SMS permission as inappropriate and linked its misuse to common phishing and spam tactics. This could be integrated into a real-time tool that alerts users to security risks, allowing them to make informed decisions about permission requests and revoke them when necessary.

5 THREATS TO VALIDITY

In this section, we outline potential threats to the validity of our study and the measures taken to mitigate them. These threats are categorized into internal, external, construct, and conclusion validity.

Internal validity concerns the degree to which the study results are attributable to the methods and not other factors. One potential threat to internal va-

lidity in our evaluation lies in the manual labeling of the dataset. Although two experts annotated the UI screens, human error and bias could still influence the results. To mitigate this, we used a rigorous labeling process, involving cross-validation between the two experts to ensure consistency. Additionally, any disagreements between annotations were resolved through discussion and re-evaluation.

Finally, the dynamic analysis tool RPCDROID, which was used to capture UI events and permission requests, may not capture all runtime behaviors, especially those triggered under specific conditions not covered in a single execution. To mitigate this, we executed each app multiple times and combined the logs from these executions. This approach allowed us to capture a broader range of behaviors and interactions, ensuring more comprehensive coverage of runtime conditions.

External validity refers to the generalizability of our results to other contexts, such as different applications, operating systems, or devices. Our study was conducted using a sample of 70 popular Android applications, which may limit the generalizability of the results to less popular or more niche applications that may exhibit different behavior in permission usage.

Another limitation to external validity is the focus on Android as the target operating system. While Android is the most widely used mobile platform, the findings of our study may not directly apply to other platforms, such as iOS, where the permission management and security architecture differ significantly. Further research is needed to evaluate the applicability of our approach to other operating systems.

Conclusion validity refers to the reliability of the inferences drawn from the study. One threat to conclusion validity is the size and diversity of the dataset. While we collected and analyzed a substantial dataset of 123,552 UI screens, more diversity in app categories, user behavior, and interaction patterns could strengthen the generalizability of the conclusions. Our dataset does not fully represent the diversity of all Android applications and user contexts, particularly for apps with uncommon functionalities or those using advanced permission schemes. In addition, the large number of TNs, caused by the predominance of interactions that did not require access to dangerous resources, certainly affected the results obtained, but we preferred to include all the screenshots captured to avoid biasing the LLM due to the unusual presence of permission requests during application interactions.

Finally, the results of our study may not reflect the full range of real-world scenarios, especially under adversarial conditions or complex attack vectors that were not part of our evaluation. Future work should incorporate a broader range of apps and scenarios, particularly those involving real-time user interaction and adaptive behaviors by malicious apps.

6 CONCLUSIONS AND FUTURE WORK

As mobile applications become increasingly sophisticated, users face growing challenges in managing app permissions effectively. This often results in users unintentionally granting access to sensitive data, which can lead to privacy risks or malicious exploitation. Despite improvements in permission systems, the burden of making critical security decisions still falls largely on users, many of whom lack the necessary technical expertise.

In this study, we investigated the feasibility of an approach leveraging LLMs to dynamically analyze app user interfaces and provide predictions regarding the appropriateness of permission requests. Our aim was twofold: (1) to evaluate the model's ability to predict dangerous permissions within diverse UI contexts ($\mathbf{RQ_1}$), and (2) to assess its effectiveness in evaluating permission requests in malicious applications ($\mathbf{RQ_2}$).

The results are promising. For $\mathbf{RQ_1}$, our approach demonstrated an overall accuracy of 81% and excelled at predicting permissions for critical resources such as the microphone, where accuracy exceeded 95%. However, considering only UI screens to extract contextual information about the feature in use may not always be enough, especially for permissions used in the background. In these cases, the model struggles to determine when a permission that is mostly requested non-interactively can actually be used. In addition, UI elements can be misleading, causing the model to predict the use of permissions that are not actually needed. We intend to address these issues in future work, extending the analysis of UI screens to user flow data and levels of interactivity.

Regarding $\mathbf{RQ_2}$, the model successfully evaluated permission requests in contexts where malicious behaviors were known or suspected. Not only did the model accurately predict when permissions were unnecessary or potentially harmful, but it also provided detailed rationales for its decisions. For example, in cases of suspected phishing or spam behavior, the model was able to explain why certain permissions, such as SMS or location access, should be denied, linking these requests to common malicious patterns. This ability to offer context-aware explanations makes the model particularly valuable for enhancing user trust and transparency.

6.1 Future Directions

Although our results are promising, several areas offer opportunities for further refinement:

Enhancing contextual understanding of user flows. Although the model performs well on individual UI screens, future work could focus on improving its ability to analyze complex user flows that span multiple screens. This would provide a more nuanced understanding of how permission requests evolve as users interact with various app features over time.

User-centric explanations. To enhance usability and transparency, future iterations should focus on generating explanations that are not only technically accurate but also easily understandable by non-expert users. This would empower users to make informed decisions, improving trust in the model's recommendations and enhancing the user experience.

Integration into real-world tools. Moving from research to real-world deployment is crucial. Integrating this approach into tools like `RPCDroid` (Guerra. et al., 2023) or directly into the Android permission model would create an automated system that can analyze and manage permission requests in real-time or provide suggestions for a more informed user decision leveraging user feedback to further refine the system's accuracy and address unforeseen challenges.

In conclusion, our study shows that integrating LLMs into permission management offers a significant step forward in terms of both accuracy and security. By automating the decision-making process and dynamically evaluating the appropriateness of permissions in real-time, we reduce the cognitive load on users and enhance privacy protections. The ability to detect and mitigate malicious behavior further positions this approach as a powerful tool for the future of mobile security. Our findings offer a strong foundation for refining and expanding this model to create a robust, scalable, and cross-platform solution for mobile privacy and security.

ACKNOWLEDGEMENTS

This work has been co-funded by the European Union - NextGenerationEU under the Italian Ministry of University and Research (MUR) National Innovation Ecosystem grant ECS00000041 -VITALITY-CUP E13C22001060006, the DiBT Startup Project, grant number INTELLIGENZA_ARTIFICIALE - Dynamic Permission Management in Android Using Artificial Intelligence Models and is part of the project PNRR-NGEU which has received funding from the MUR – DM 118/2023.

REFERENCES

Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F. L., Almeida, D., Altenschmidt, J., Altman, S., Anadkat, S., et al. (2023). Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Akhawe, D. and Felt, A. P. (2013). Alice in warningland: A Large-Scale field study of browser security warning effectiveness. In *22nd USENIX Security Symposium (USENIX Security 13)*, pages 257–272, Washington, D.C. USENIX Association.

Almomani, I. M. and Khayer, A. A. (2020). A comprehensive analysis of the android permissions system. *IEEE Access*, 8:216671–216688.

Bonné, B., Peddinti, S. T., Bilogrevic, I., and Taft, N. (2017). Exploring decision making with Android's runtime permission dialogs using in-context surveys. In *13th Symposium on Usable Privacy and Security*, pages 195–210, Santa Clara, CA. USENIX Ass.

Caffagni, D., Cocchi, F., Barsellotti, L., Moratelli, N., Sarto, S., Baraldi, L., Baraldi, L., Cornia, M., and Cucchiara, R. (2024). The revolution of multimodal large language models: A survey.

Chen, B., Zhang, Z., Langrené, N., and Zhu, S. (2024). Unleashing the potential of prompt engineering in large language models: a comprehensive review.

Ciobanu, M. G., Fasano, F., Martinelli, F., Mercaldo, F., and Santone, A. (2020). Accidental sensitive data leaks prevention via formal verification. In Furnell, S., Mori, P., Weippl, E. R., and Camp, O., editors, *Proceedings of the 6th International Conference on Information Systems Security and Privacy, ICISSP 2020, Valletta, Malta, February 25-27, 2020*, pages 825–834. SCITEPRESS.

Dong, Q., Li, L., Dai, D., Zheng, C., Ma, J., Li, R., Xia, H., Xu, J., Wu, Z., Chang, B., Sun, X., Li, L., and Sui, Z. (2024). A survey on in-context learning.

Fang, Z., Han, W., and Li, Y. (2014). Permission based android security: Issues and countermeasures. *Computers & Security*, 43:205–218.

Fasano, F., Martinelli, F., Mercaldo, F., and Santone, A. (2020). Android run-time permission exploitation user awareness by means of formal methods. In Furnell, S., Mori, P., Weippl, E. R., and Camp, O., editors, *Proceedings of the 6th International Conference on Information Systems Security and Privacy, ICISSP 2020, Valletta, Malta, February 25-27, 2020*, pages 804–814. SCITEPRESS.

Felt, A. P., Ha, E., Egelman, S., Haney, A., Chin, E., and Wagner, D. (2012). Android permissions: user attention, comprehension, and behavior. In *Proceedings of the Eighth Symposium on Usable Privacy and Security*, SOUPS '12, New York, NY, USA. ACM.

Gao, H., Guo, C., Wu, Y., Dong, N., Hou, X., Xu, S., and Xu, J. (2019). Autoper: Automatic recommender for runtime-permission in android applications. In *IEEE 43rd Annual Computer Software and Applications Conference*, volume 1, pages 107–116.

Guerra, M., Milanese, R., Deodato, M., Perozzi, V., and Fasano, F. (2024). Visual attention and privacy in-

dicators in android: Insights from eye tracking. In *Proceedings of the 10th International Conference on Information Systems Security and Privacy - Volume 1: ICISSP*, pages 320–329. INSTICC, SciTePress.

Guerra., M., Milanese., R., Oliveto., R., and Fasano., F. (2023). Rpcdroid: Runtime identification of permission usage contexts in android applications. In *Proceedings of the 9th International Conference on Information Systems Security and Privacy - ICISSP*, pages 714–721. INSTICC, SciTePress.

Guerra, M., Scalabrino, S., Fasano, F., and Oliveto, R. (2023). An empirical study on the effectiveness of privacy indicators. *IEEE Transactions on Software Engineering*, 49(10):4610–4623.

Kong, A., Zhao, S., Chen, H., Li, Q., Qin, Y., Sun, R., Zhou, X., Wang, E., and Dong, X. (2024). Better zero-shot reasoning with role-play prompting.

Li, L., Wang, R., Zhan, X., Wang, Y., Gao, C., Wang, S., and Liu, Y. (2023). What you see is what you get? it is not the case! detecting misleading icons for mobile applications. In *Proceedings of the 32nd ACM SIGSOFT International Symposium on Software Testing and Analysis*, ISSTA 2023, page 538–550, New York, NY, USA. ACM.

Liu, B., Lin, J., and Sadeh, N. (2014). Reconciling mobile app privacy and usability on smartphones: could user privacy profiles help? In *Proceedings of the 23rd International Conference on World Wide Web*, WWW '14, page 201–212, New York, NY, USA. ACM.

Malviya, V. K., Tun, Y. N., Leow, C. W., Xynyn, A. T., Shar, L. K., and Jiang, L. (2023). Fine-grained in-context permission classification for android apps using control-flow graph embedding. In *2023 38th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pages 1225–1237.

Micinski, K., Votipka, D., Stevens, R., Kofinas, N., Mazurek, M. L., and Foster, J. S. (2017). User interactions and permission use on android. In *Proc. of the 2017 CHI Conference on Human Factors in Computing Systems*, CHI '17, page 362–373, New York, NY, USA. ACM.

Minaee, S., Mikolov, T., Nikzad, N., Chenaghlu, M., Socher, R., Amatriain, X., and Gao, J. (2024). Large language models: A survey.

Nauman, M., Khan, S., Othman, A. T., and Musa, S. (2015). Realization of a user-centric, privacy preserving permission framework for android. *Security and Communication Networks*, 8(3):368–382.

Oglaza, A., Laborde, R., Zaraté, P., Benzekri, A., and Barrère, F. (2017). A new approach for managing android permissions: learning users' preferences. *EURASIP Journal on Information Security*, 2017(1):13.

Olejnik, K., Dacosta, I., Machado, J. S., Huguenin, K., Khan, M. E., and Hubaux, J.-P. (2017). Smarper: Context-aware and automatic runtime-permissions for mobile devices. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 1058–1076.

Rashidi, B., Fung, C., Nguyen, A., Vu, T., and Bertino, E. (2018). Android user privacy preserving through crowdsourcing. *IEEE Transactions on Information Forensics and Security*, 13(3):773–787.

Roesner, F., Kohno, T., Moshchuk, A., Parno, B., Wang, H. J., and Cowan, C. (2012). User-driven access control: Rethinking permission granting in modern operating systems. In *2012 IEEE Symposium on Security and Privacy*, pages 224–238.

Rush, A. (2018). The annotated transformer. In Park, E. L., Hagiwara, M., Milajevs, D., and Tan, L., editors, *Proc. of Workshop for NLP Open Source Software (NLP-OSS)*, pages 52–60, Melbourne, Australia. ACL.

Sabahi, F., Ahmad, M. O., and Swamy, M. N. S. (2018). Content-based image retrieval using perceptual image hashing and hopfield neural network. In *2018 IEEE 61st International Midwest Symposium on Circuits and Systems (MWSCAS)*, pages 352–355.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. u., and Polosukhin, I. (2017). Attention is all you need. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Wijesekera, P., Baokar, A., Hosseini, A., Egelman, S., Wagner, D., and Beznosov, K. (2015). Android permissions remystified: A field study on contextual integrity. In *24th USENIX Security Symposium (USENIX Security 15)*, pages 499–514, Washington, D.C. USENIX Association.

Wu, J., Gan, W., Chen, Z., Wan, S., and Yu, P. S. (2023). Multimodal large language models: A survey. In *2023 IEEE International Conference on Big Data (BigData)*, pages 2247–2256.

Xi, S., Yang, S., Xiao, X., Yao, Y., Xiong, Y., Xu, F., Wang, H., Gao, P., Liu, Z., Xu, F., and Lu, J. (2019). Deepintent: Deep icon-behavior learning for detecting intention-behavior discrepancy in mobile apps. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, CCS '19, page 2421–2436, New York, NY, USA. ACM.

Xiao, X., Wang, X., Cao, Z., Wang, H., and Gao, P. (2019). Iconintent: Automatic identification of sensitive ui widgets based on icon classification for android apps. In *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*, pages 257–268.

Yao, Y., Duan, J., Xu, K., Cai, Y., Sun, Z., and Zhang, Y. (2024). A survey on large language model (llm) security and privacy: The good, the bad, and the ugly. *High-Confidence Computing*, 4(2):100211.

Ye, Q., Axmed, M., Pryzant, R., and Khani, F. (2024). Prompt engineering a prompt engineer.

Zhao, W. X., Zhou, K., Li, J., Tang, T., Wang, X., Hou, Y., Min, Y., Zhang, B., Zhang, J., Dong, Z., et al. (2023). A survey of large language models.

Zhauniarovich, Y. and Gadyatskaya, O. (2016). Small changes, big changes: An updated view on the android permission system. In Monrose, F., Dacier, M., Blanc, G., and Garcia-Alfaro, J., editors, *Research in Attacks, Intrusions, and Defenses*, pages 346–367, Cham. Springer International Publishing.