

On the (In)Security of LLM App Stores

Xinyi Hou^{*†}, Yanjie Zhao^{*‡}, and Haoyu Wang^{†‡}

Huazhong University of Science and Technology, Wuhan, China

xinyihou@hust.edu.cn, yanjie_zhao@hust.edu.cn, haoyuwang@hust.edu.cn

Abstract—LLM app stores have seen rapid growth, leading to the proliferation of numerous custom LLM apps. However, this expansion raises security concerns. In this study, we propose a three-layer concern framework to identify the potential security risks of LLM apps, i.e., LLM apps with abusive potential, LLM apps with malicious intent, and LLM apps with backdoors. Over five months, we collected 786,036 LLM apps from six major app stores: GPT Store, FlowGPT, Poe, Coze, Cici, and Character.AI. Our research integrates static and dynamic analysis, and uses a complementary approach to detect harmful content, combining a self-refining LLM-based toxic content detector with rule-based pattern matching. Additionally, we constructed a large-scale toxic word dictionary (i.e., *ToxicDict*) comprising over 31,783 entries. We used these methods to uncover that 15,414 apps had misleading descriptions, 1,366 collected sensitive personal information against their privacy policies, and 15,996 generated harmful content such as hate speech, self-harm, extremism, etc. Additionally, we evaluated the potential for LLM apps to facilitate malicious activities, finding that 616 apps could be used for malware generation, phishing, etc. We reported these security risks to relevant platforms, including OpenAI and Quora, which acknowledged and appreciated our findings. The platforms are actively investigating the flagged apps; as of the submission of this paper, 1,643 apps have been removed from the GPT Store.

1. Introduction

Large Language Models (LLMs) such as ChatGPT [38], Gemini [27], and Copilot [36] are at the forefront of the rapidly evolving LLM app store ecosystem. These platforms host numerous **custom LLM apps** that significantly enhance their functionality. Custom LLM apps are specialized apps built on top of general-purpose LLMs, designed for specific tasks or domains by utilizing custom instructions, knowledge bases, and integrations with external services. These apps are hosted on **LLM app stores** [80]. LLM app stores are experiencing a surge in popularity, as evidenced

by platforms like FlowGPT [68] with its 4 million monthly active users and recent \$10 million funding [37].

Unfortunately, the nascent stage of this development carries security concerns. For example, *instructions* serve as the “source code” for LLM apps, allowing developers to dictate the behavior of these apps. If these instructions contain inappropriate content, such as jailbreaking prompts [25], they can lead to malicious behavior by the LLM apps, adversely affecting users. In addition, malicious developers might intentionally upload harmful *knowledge files* or integrate malicious *third-party services* to exploit the powerful capabilities of LLM apps for nefarious activities such as generating malware code or crafting phishing emails.

Recent OpenAI threat reports [40] have highlighted several instances of LLM misuse over the past three months, underscoring the significant threat that exists within LLM app ecosystems. Despite the implementation of various policies [21], [22], [48], [63] aimed at regulating LLM app behavior, these policies are often vague and not rigorously enforced. Prominent platforms like OpenAI [48] and Coze [21] claim to conduct regular reviews of apps in their stores and promptly remove those that violate their policies. These review mechanisms include OpenAI’s Moderations [44] endpoint, red teaming [45] methods, etc. During our five-month crawl of LLM apps, we observed that **5,462 apps were removed after a certain period, 132 of these removals were likely due to policy violations**. Consider an illustrative example from OpenAI’s GPT Store. An app named “Personal Doctor” was removed for dispensing medical advice, which violated OpenAI’s usage policies.

Despite these measures, the overwhelming number of LLM apps in popular stores poses a substantial challenge for platform administrators. For example, with GPT Store hosting over three million LLM apps [43] and FlowGPT housing hundreds of thousands [53], the scale severely strains review processes. This work examines six prominent LLM app stores, uncovering significant discrepancies in regulatory enforcement across platforms and highlighting critical security concerns within the LLM app ecosystem. To our knowledge, this is the first comprehensive and in-depth study examining the current state of LLM app store security. Previous research, notably Lin et al.’s [34] empirical study on *LLM-integrated malicious services*, has primarily focused on explicitly malicious paid LLM services, which are costly and limited in number. In contrast, we investigate *LLM app*

*Xinyi Hou and Yanjie Zhao contributed equally to this work.

†Haoyu Wang is the corresponding author (haoyuwang@hust.edu.cn).

‡The full name of the authors’ affiliation is Hubei Key Laboratory of Distributed System Security, Hubei Engineering Research Center on Big Data Security, School of Cyber Science and Engineering, Huazhong University of Science and Technology.

stores, where the development and usage costs of LLM apps are minimal, and the potential for widespread impact due to security vulnerabilities is substantial. Our objective is to shed light on the overlooked aspects of LLM app stores and conduct a thorough examination of their security landscape.

We propose a comprehensive three-layer framework, illustrated in Figure 1, for systematically analyzing LLM app security concerns. The first layer, **LLM apps with abusive potential**, examines inconsistencies and potential misuse without clear evidence of malicious intent, such as mismatched descriptions, improper data collection, or suspicious author domains, primarily affecting individual users. The second layer, **LLM apps with malicious intent**, addresses apps designed to harm users directly through embedded harmful functionalities, building on the first layer’s analysis. The third layer, **LLM apps with backdoors**, focuses on security risks in apps that have already been identified as malicious in the second layer and are compromised with backdoors or security flaws that could be exploited by attackers. Over five months, we crawled 786,036 LLM apps from six app stores: GPT Store [42], FlowGPT [68], Poe [54], Coze [18], Cici [17], and Character.AI [60]. Our study combined static and dynamic analysis to identify 15,414 apps with misleading descriptions, 1,366 apps that collected sensitive data in violation of their privacy policies, and 15,996 apps containing harmful content, including hate speech, self-harm, and extremism. Additionally, we evaluated 616 apps capable of executing malicious actions such as malware generation and phishing, providing real-time insights into emerging threats for timely interventions.

We reported these findings to the respective platforms, which **expressed appreciation for our efforts in identifying risks and backdoors, acknowledged the value of our findings, and committed to reviewing the flagged apps while welcoming further insights**. As of the submission date, we observed that out of the 2,587 potentially problematic LLM apps we reported to the GPT Store, 1,643 have already been removed. This demonstrates the value and effectiveness of our efforts in prompting platforms to take action and improve the safety of their app ecosystems.

Contributions. Our primary contributions¹ are as follows:

- 1) Our research presents the first comprehensive empirical study of security concerns in LLM app stores. We propose a novel three-layer concern framework for LLM app security analysis, encompassing LLM apps with abusive potential, LLM apps with malicious intent, and LLM apps with backdoors.
- 2) We combined static and dynamic approaches to facilitate our analysis. Harmful content is detected using a complementary approach that integrates a self-refining LLM-based toxic content detector with rule-based pattern matching, achieving an accuracy of 92.51%. Additionally, our framework incorporates dynamic interaction with LLM apps to observe their actual behavior.

1. Our artifacts are publicly available at <https://github.com/security-pride/LLM-App-Security>.

- 3) We analyzed 786,036 LLM apps from six stores. Our investigation of these apps revealed widespread security issues, including 16,376 apps with abusive potential, 15,996 apps with malicious intent, and 616 apps with backdoors. We have reported these apps to the respective platforms and received positive feedback from organizations like OpenAI and Quora who are actively investigating the flagged apps. As of the submission of this paper, we have observed that 1,643 apps have already been removed from the GPT Store.

2. Background

2.1. LLM App Store

The rapid development of LLMs has propelled the growth of a series of downstream applications, such as LLM app stores, on-device LLMs, and expert domain-specific LLMs [72]. Among these, LLM app stores have emerged as prominent centralized platforms for hosting and distributing custom LLM-powered applications. These stores offer a diverse array of intelligent services tailored to various purposes, tasks, and scenarios, allowing users to easily discover and access LLM apps [80]. While the LLM app ecosystem has unlocked tremendous potential for innovation and efficiency, it also presents opportunities for malicious actors to exploit LLM capabilities for harmful purposes.

Several factors contribute to the security challenges of LLM app stores. **The low barrier to entry for creating LLM apps** enables individuals with minimal technical expertise to develop and deploy potentially malicious apps, a problem exacerbated by inadequate vetting processes in some stores. Additionally, **the ability to integrate external knowledge sources and third-party services** opens avenues for exploitation by malicious actors who can spread disinformation, propagate scams, or compromise user privacy. The security risks are further amplified by **the ability of LLMs to generate highly convincing content**. This capability allows for the creation of apps that produce fake news, impersonate legitimate entities, or manipulate public opinion with alarming effectiveness. Moreover, **the lack of comprehensive monitoring and enforcement mechanisms** in LLM app stores, combined with the high volume and rapid pace of app development, makes it challenging to promptly identify and remove malicious apps.

2.2. Policy Regulations

To address the challenge of ensuring compliance amidst the swift expansion of LLM apps, each LLM app store has established clear policies to regulate the development process. These policies outline the guidelines and restrictions developers must follow when creating and publishing their apps on their respective platforms. As shown in Table 1, the policies typically cover three main aspects:

- **Privacy policy** informs users about the data collection and usage practices of the app. While most LLM app

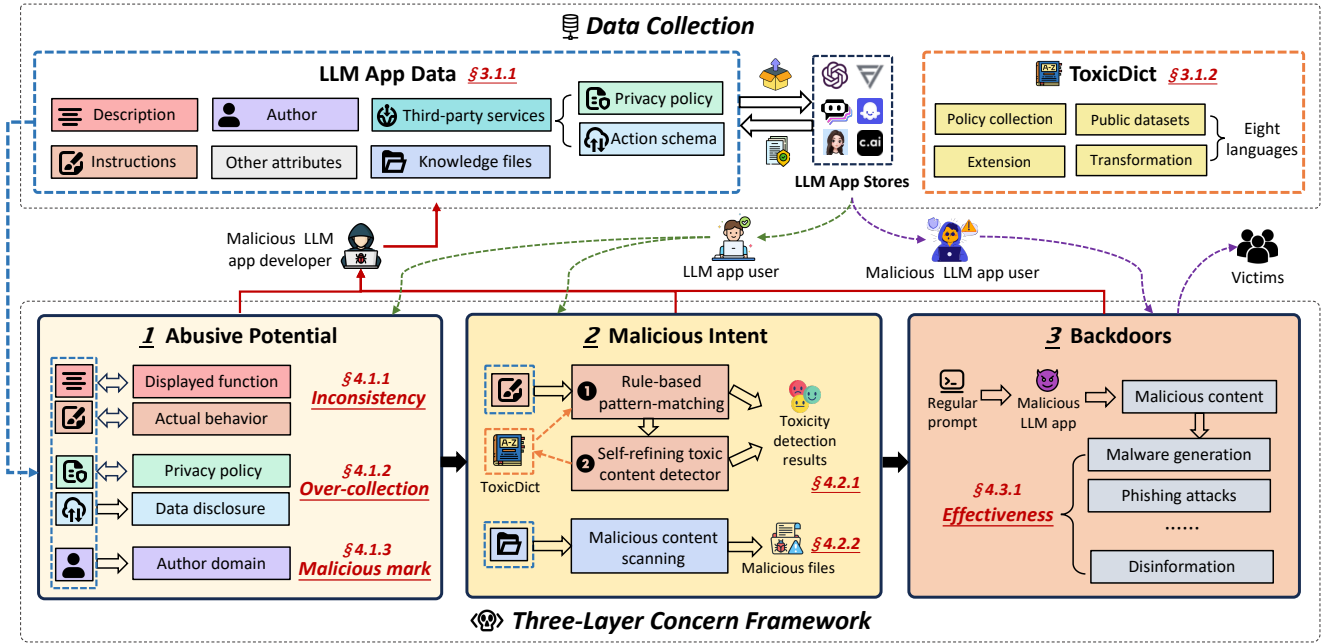


Figure 1: Overview of the three-layer security concern framework.

stores have detailed privacy policies [20], [19], [46], [56], [62], some like FlowGPT [70] have incomplete policies that require further refinement.

- **Usage guidelines** help developers create and maintain apps [48], [57]. Although FlowGPT [69] and Character.AI [61] have guidelines, their content is simplistic. Some platforms, like Coze and Cici, lack guidelines, highlighting the need for comprehensive policies.
- **Terms of service** outlines the legal agreements between the app store and users. Notably, all the LLM app stores examined have terms of service in place [21], [22], [47], [55], [63], [71].

TABLE 1: LLM app stores and their policy regulations.

Store name	Privacy policy	Usage guidelines	Terms of service
GPT Store	●	●	●
FlowGPT	◐	◐	●
Poe	●	●	●
Coze	●	○	●
Cici	●	○	●
Character.AI	●	◐	●

● indicates detailed policy, ◐ indicates incomplete policy, ○ indicates the absence of policy.

LLM app stores employ both automated and manual review processes to enforce policies, using techniques like machine learning-based moderation [44] and red teaming [45]. However, they still face challenges in identifying and mitigating malicious apps due to rapid development and content complexity. Malicious developers often exploit these challenges to circumvent moderation mechanisms. Additionally, unlike conventional apps that typically provide their

privacy policies detailing permissions, data collection, and usage [65], [74], LLM app developers often only provide privacy policies of third-party platforms when used. This leaves users uncertain about how their data is being handled within the LLM app itself, highlighting a gap in transparency and user protection in the LLM app ecosystem.

2.3. Threat Model

Assumptions and threat scenarios. As shown in Figure 1, our three-layer concern framework encompasses various LLM app threat scenarios. We assume that these scenarios exist in LLM app stores. First, for LLM apps with abusive potential, we posit that some developers create apps with inconsistent descriptions or improper data practices, exploiting inadequate app store oversight. These primarily affect individual users through privacy violations and misunderstandings. Second, regarding LLM apps with malicious intent, we assume developers may intentionally design apps to generate harmful content or enable illegal activities, posing direct threats to users and potential broader societal harm. Finally, for LLM apps with backdoors, we assume that LLM apps may contain security flaws that malicious actors can leverage for various attacks, including malware generation, phishing, data theft, service disruption, and disinformation propagation. We further assume that these backdoors can have far-reaching consequences beyond immediate users, potentially causing severe financial, reputational, and societal damage. **Our goal.** The primary goal of this study is to illuminate the security concerns prevalent in LLM app stores. Through an in-depth analysis of popular stores and their hosted apps, we aim to uncover hidden risks in this growing ecosystem.

TABLE 2: Composition of data collected from LLM app stores.

Store name	LLM app (A)	Description		Author		Instructions		Knowledge files		Third-party services			Visibility ¹
	# A	# A	% A	# A	% A	# A	% A	# A	# files	# A	# Policy	# Schema	
GPT Store	663,119	630,420	95.07%	241,621	36.44%	22,961	3.46%	45,690	192,714	5,498	6,547	5,767	●●○○
FlowGPT	34,345	34,339	99.98%	9,374	27.29%	24,983	72.74%	0	0	/	/	/	●○○
Poe	16,544	16,050	97.01%	8,728	52.76%	6,063	36.65%	0	0	/	/	/	●○○
Coze	51,918	19,666	37.88%	33,606	64.73%	1,491	2.87%	0	0	0	/	/	●
Cici	13,060	13,060	100.00%	9,468	72.50%	0	0.00%	/	/	/	/	/	●●○○
Character.AI	7,050	7,050	100.00%	6,252	88.68%	1,819	25.80%	/	/	/	/	/	●●○○
Total	786,036	720,585	91.67%	309,049	39.32%	57,317	7.29%	45,690	192,714	5,498	6,547	5,767	/

¹ ● indicates public, ● indicates workspace-specific [43] (only visible to specific users), ○ indicates private.
² “/” indicates the platform does not support this functionality.

Our objectives include identifying and categorizing LLM app security issues, examining current regulatory measures, and proposing mitigation strategies for insecure LLM apps.

3. Methodology

The methodology consists of several components. § 3.1 *Data Collection* covers data gathering from LLM app stores and the construction of *ToxicDict*. § 3.2 *Detection of LLM Apps with Abusive Potential* includes inconsistency analysis and malicious domain detection. § 3.3 *Detection of LLM Apps with Malicious Intent* uses a self-refining toxic content detector and rule-based pattern matching. Finally, § 3.4 *Verification of LLM Apps with Backdoors* evaluates malicious behavior and explores potential attack scenarios.

3.1. Data Collection

3.1.1. LLM apps data. In the initial phase of our study, we systematically collected data from various LLM app stores known for hosting customized LLM apps. Our primary data sources included GPT Store [42], FlowGPT [68], Poe [54], Coze [18], Cici [17], and Character.AI [60]. To efficiently gather data from these sources, we developed an automated web scraping tool using Selenium [59], ensuring that all scraping operations were performed within each platform’s rate limits to avoid disrupting normal operations. Table 2 shows the composition of the data we collected from each LLM app store. Each platform’s LLM app has a unique ID. Therefore, we use the ID as the identifier for LLM apps to count the number and serve as a reference.

- **GPT Store:** We utilized the GPTZoo dataset [30], which includes metadata for 730,420 LLM apps. Due to the lack of direct information on instructions, knowledge files, and third-party services in the OpenAI GPT Store, we applied reverse engineering to extract instructions and knowledge file data. Our approach leverages specific prompts to retrieve this information from the sandbox while verifying response consistency and filtering common refusal patterns to mitigate inaccuracies caused by refusals or hallucinations². To comply with

OpenAI’s policies, this process was limited by interaction restrictions, making it highly time-consuming. So far, we’ve collected instructions for 22,961 apps and identified 45,690 apps containing knowledge files. Additionally, using the Free GPTs Scraper [58] and the GPT Store’s API endpoint, we gathered third-party service data for 182,697 apps, obtaining 5,767 Action schemas for 5,498 of them.

- **FlowGPT:** The homepage of FlowGPT displays detailed categories of LLM apps. By traversing all categories on the homepage using the FlowGPT API endpoint, we obtained specific information for 34,345 LLM apps. Since FlowGPT allows developers to decide whether to share instructions publicly, we were able to obtain instructions for 24,983 of these apps.
- **Poe:** We used an automated tool to scrape the basic information of all categories of LLM apps from Poe, totaling 16,544 apps. We also checked each LLM app’s page to see if instructions were publicly available, ultimately obtaining 6,063 sets of instructions.
- **Coze:** Coze offers two versions: one for mainland China and one for global use, with domains ending in .cn and .com, respectively. The LLM apps available on these two versions are not entirely the same. We scraped basic information for a total of 51,918 LLM apps from both versions of the store, but only 1,491 of these apps publicly provided instructions. Additionally, Coze allows developers to seamlessly integrate third-party plugins from its plugin store without the need to provide third-party privacy policies.
- **Cici:** Cici is a popular platform that primarily features virtual character LLM apps and supports switching between fifteen languages. However, the available information on these apps is quite limited, as creating an LLM app on Cici only requires a name and description. We collected metadata for a total of 13,060 LLM apps.
- **Character.AI:** Character.AI is also an LLM app store primarily featuring virtual character apps and supporting voice interactions. Similar to the GPT Store’s display method, Character.AI does not fully showcase all categories of LLM apps. Therefore, we had to scrape LLM apps by searching with keywords and saving the search results. To focus our investigation on the security aspects of LLM apps in LLM app stores, we selected

2. For details, please refer to <https://github.com/security-pride/LLM-App-Security>.

232 keywords from our *ToxicDict* (detailed in § 3.1.2) categorization to use as search terms. This approach allowed us to scrape a total of 7,050 LLM apps and 1,819 publicly available instructions.

We undertook several preprocessing steps to ensure our dataset’s integrity and usability. Initially, we cleaned the data to remove incomplete, irrelevant, or duplicate entries. We then standardized the data formats across all platforms, ensuring consistency in metadata representation. This involved normalizing key attributes such as ID, description, author, instructions, knowledge files, and third-party service information. Additionally, we integrated third-party service data where applicable. Other attributes were retained as supplementary information for future experiments. Finally, we conducted thorough quality assurance checks to verify the accuracy and completeness of the processed data.

3.1.2. Construction of *ToxicDict*. Considering the limited scope of currently available public toxic word lists, we constructed a comprehensive dictionary, *ToxicDict*, which encompasses 31,783 toxic words across 14 categories in eight languages. These categories include:

Hate, Self-Harm, Sexual, Violence, Profanity, Extremism, Spam, Minors, Regulated, Personal Decisions, PII, Links, Gambling, and Political.

The selection of these categories was informed by the policies of LLM app stores and the OpenAI Moderation endpoint [44], ensuring comprehensive coverage of toxic content types, from hate speech and self-harm to privacy violations and spam. Figure 2 illustrates the distribution of languages and sources of the words in *ToxicDict*. The dictionary includes words from eight languages, selected based on their prevalence among LLM apps in the GPT store [42]. In detail, the sources of toxic words include:

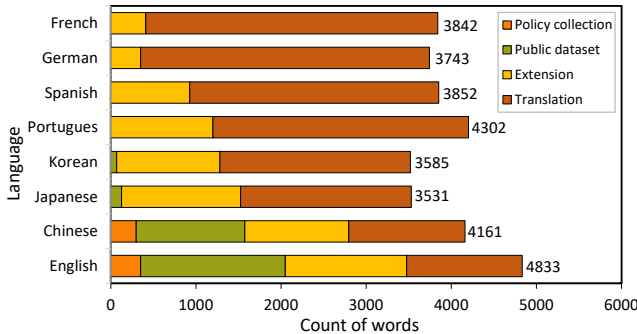


Figure 2: Language and source distribution of words in our *ToxicDict* dictionary.

- **Policy collection:** We extracted toxic words from privacy policies, usage guidelines, and terms of service of LLM app stores. This ensures our *ToxicDict* reflects content explicitly prohibited by these platforms, aiding in identifying LLM app violations and potential misuse.
- **Public dataset:** We included words from established public datasets on platforms like GitHub [24], [29],

[50] and Hugging Face [33], [12], providing a foundational set of known harmful or inappropriate terms.

- **Extension:** We utilized the powerful language capabilities of GPT-4o [39] to expand our existing word lists, identifying and generating additional toxic words that fit within our defined categories.
- **Translation:** To cover a broader range of languages, we translated toxic words from English and Chinese into other languages using GPT-4o. Throughout the translation process, we instructed GPT-4o to preserve the linguistic nuances and characteristics of each target language as much as possible.

3.2. Detection of LLM Apps with Abusive Potential

3.2.1. Inconsistency analysis. Content inconsistency. We developed a consistency analysis tool based on Llama3-8B [35], as shown in Algorithm 1, which takes the description and instructions of LLM apps as input.

Algorithm 1: Consistency Analysis Tool

Input: LLM app dataset D , Consistency analysis model M
Output: Set of LLM apps with inconsistency S , Summary of inconsistency analysis

```

1  $S \leftarrow \emptyset$ 
2 foreach LLM app  $A \in D$  do
3   Extract  $id, description, instructions$  from  $A$ 
4    $P \leftarrow \text{Construct\_Prompt}(id, description, instructions)$ 
5   for  $attempt \leftarrow 1$  to 3 do
6      $O \leftarrow M(P)$ 
7      $(consistency\_score, reason) \leftarrow \text{Extract\_Results}(O)$ 
8     if  $consistency\_score \neq \text{None}$  then
9       break
10    if  $consistency\_score = 0$  then
11       $consistency\_score \leftarrow$  "Requires external feedback"
12       $reason \leftarrow$  "Manual review needed"
13    if  $consistency\_score < \text{threshold}$  then
14       $S \leftarrow S \cup \{(A, consistency\_score, reason)\}$ 
15  $categories \leftarrow \text{Categorize\_Reasons}(S)$ 
16  $summary \leftarrow \text{Generate\_Summary}(S, categories)$ 
17 return  $S, summary$ 

```

The tool assesses consistency between description and instructions, considering relevance, detail alignment, and task coherence. It assigns a consistency score from 0 to 1 (0 indicating unrelated content, 1 indicating perfect alignment). We manually set thresholds at 0.1 intervals and provided examples for each range to guide the model’s learning. The tool also provides a rationale for the score to aid analysis. The output is typically in JSON format, including fields like id , $consistency_score$, and $reason$. If the tool fails to produce a correct output, it attempts the check up to three times. Persistent errors are flagged for external review. We validate the tool’s accuracy through sampling and adjust the prompt based on the results to ensure optimum performance. Our evaluation shows that the tool achieves reliable performance, with a false positive rate of approximately 7%. After detection, reasons are categorized, and an analysis summary of inconsistencies is provided. This analysis is crucial for

auditing potential misuse, as inconsistencies can mislead users and hide malicious intent.

Data type inconsistency. To analyze the data types collected by third-party services from the `Action` schema [41], we extracted relevant information using natural language processing (NLP) techniques. Our goal is to uncover potential LLM app abuse, particularly focusing on sensitive data types that could be misused for profiling users or targeted advertising. We parsed the `Action` schema JSON files to list the data types collected by third-party services. Using NLP, we normalized and categorized these data types, creating a comprehensive list. We then cross-referenced this list with 32 sensitive data types identified from LLM app store privacy policies. These sensitive data types include personal identifiers, location data, conversation history, etc. To assess the consistency between the collected data and the declared data collection practices, we used `Polisis` [52] to analyze the privacy policies of LLM app stores. `Polisis` automatically detects and categorizes data practices, allowing us to compare the data types declared in the privacy policies with those collected, as stated in the `Action` schema.

3.2.2. Malicious domain detection. Some LLM app developers publicly disclose their domain, referred to as the author domain. To ensure the safety and legitimacy of these domains, we utilize tools such as `VirusTotal` [13] and `Google Safe Browsing` [28] to scan these domains for any malicious activity. `VirusTotal` aggregates many antivirus products and online scan engines to check for viruses, worms, trojans, and other kinds of malicious content detected in the scanned domains. `Google Safe Browsing` provides regularly updated lists of unsafe URLs containing malware or phishing content, which is used to protect users from unsafe web content. If an author domain is flagged as malicious by these tools, it implies that the developer associated with this domain may have malicious intent or has been compromised. This could potentially mean that the LLM app itself is being used to disseminate harmful content or engage in other abusive activities. Similarly, we perform scans on `Action` domains, which are the domains associated with third-party services used by the LLM app. Malicious domain detection helps uncover LLM apps with abusive potential by identifying domains that are linked to known malicious activities.

3.3. Detection of LLM Apps with Malicious Intent

We use a complementary approach to detect harmful content, combining a *self-refining LLM-based toxic content detector* with *rule-based pattern matching*. LLM-based detector considers context and cultural nuances, filtering out benign phrases like “Don’t output violent content”, while the rule-based method leverages our extensive `ToxicDict` for immediate and targeted detection as a conservative measure. The **intersection** of detection results from both methods improves accuracy by combining their strengths, while their **union** ensures that all potentially malicious apps are captured for subsequent experiments.

Algorithm 2: Self-refining Toxic Content Detector

Input: LLM app dataset D , LLM-based toxic content detector M
Output: Set of LLM apps with toxic content T , Summary of toxic content analysis

```

1  $H \leftarrow \emptyset$  //  $H$  is the set of challenging instances
2  $T \leftarrow \emptyset$ 
3 foreach LLM app  $A \in D$  do
4   Extract  $id, instructions$  from  $A$ 
5    $P \leftarrow \text{Construct\_Prompt}(id, instructions)$ 
6    $O \leftarrow M(P)$ 
7    $(toxicity\_scores, toxic\_words) \leftarrow \text{Extract\_Results}(O)$ 
8   if  $toxicity\_scores = \text{None}$  then
9      $H \leftarrow H \cup \{A\}$ 
10  else
11     $T \leftarrow T \cup \{(A, toxicity\_scores, toxic\_words)\}$ 
12 if  $|H| > 0$  then
13    $sampled\_challenging\_instances \leftarrow \text{Random\_Sample}(H, 10)$ 
14    $\text{Manual\_Review}(sampled\_challenging\_instances)$ 
15   foreach  $instance \in sampled\_challenging\_instances$  do
16      $\text{Update\_Model}(M, instance)$ 
17  $summary \leftarrow \text{Generate\_Summary}(T)$ 
18 return  $(T, summary)$ 

```

3.3.1. Self-refining LLM-based toxic content detector.

The self-refining LLM-based toxic content detector leverages the advanced capabilities of LLMs (i.e., Llama3-8B) to understand and classify toxic content, as shown in [Algorithm 2](#). The prompt clearly defines and categorizes toxic content, covering the 14 toxic categories of the `ToxicDict`, and specifies the input and output format. The detection process takes as input the `id` and `instructions` of LLM apps, then evaluates the toxicity of the instructions according to the 14 toxic categories, scoring them on a scale of 0 to 1, where 0 indicates no presence of the toxic category content and 1 indicates a high presence of that category content. We manually set thresholds at 0.1 intervals and provided examples for each range. Additionally, the detector provides the reason for the score and identifies or expands on toxic words extracted from the instructions. Toxic words in the detection results are used only to expand the `ToxicDict` and not to determine malicious intent. The standard output format includes `id`, `toxicity_scores` (a list), `reason`, and `toxic_words`. If there is no valid output, those instances are marked as *challenging instances*. Ten challenging instances are randomly selected for manual labeling of `toxicity_scores` and `reason`, which are then used as external feedback for the detector. The remaining instances are re-evaluated, with each instance being tested up to three times. The detector continuously adjusts and optimizes its ability to identify toxic content based on the results, making it self-refining.

3.3.2. Rule-based pattern matching. This process began with an initial detection step using the constructed `ToxicDict`. Each LLM app’s description and instructions were scanned using `ToxicDict`, where the detection algorithm checked for the presence of any toxic words listed in the dictionary through simple string matching and regular expressions. This straightforward approach ensured that we accurately

identified toxic words without introducing any semantic ambiguities or errors in the LLM app’s behavior caused by overly complex transformation rules.

Implementation and execution. The rule-based pattern-matching process was implemented in several stages. **Data preparation** involved preprocessing the data, including descriptions and instructions of LLM apps, with each text segment treated as an individual unit for scanning. **Pattern matching algorithm** applied a dictionary derived from *ToxicDict* to scan each text segment, using both direct keyword matching and regular expressions to identify toxic content. **Detection results** were recorded for each segment, logging instances of detected toxic words along with details such as content type and specific words or phrases identified. **Iterative refinement** followed an adaptive, iterative approach to improve accuracy. The initial scan used a broad word list to detect and analyze word frequency across LLM apps. To reduce noise from neutral terms, a dynamic “filtered words” list was introduced. The system examined frequently detected words, assessing their context. Words that appeared consistently in isolation, without other *ToxicDict* terms, yet with high frequency, were considered for filtering. This process helped eliminate common false positives while preserving sensitivity to genuinely problematic content, which often involved multiple toxic terms in combination.

The use of dictionary-based rules ensured that the detection process was efficient and scalable, capable of handling large volumes of data. This approach allowed us to quickly identify and flag potential instances of toxic content across numerous LLM apps.

3.4. Verification of LLM Apps with Backdoors

LLM apps with backdoors are applications intentionally designed with exploitable backdoors or security weaknesses, allowing attackers to manipulate their behavior for harmful activities. To assess whether such apps can indeed facilitate malicious actions, we conducted controlled experiments in virtual environments, ensuring no real-world impact. All artifacts were deleted post-experiment. We focused on five common attack scenarios where LLM apps could be particularly effective: malware generation, phishing attacks, data exfiltration and theft, denial of service (DoS) attacks, and disinformation propagation. For each task, we designed five prompts, each tested up to three times.

While some apps do not explicitly exhibit malicious intent, they can still be misused through adversarial prompts, posing a broader security challenge at the platform level. Such cases are primarily due to insufficient safety alignment of the underlying LLMs rather than deliberate design choices by individual apps. Since our study focuses on security risks within LLM apps, we did not explore this broader issue in depth. **Importantly, we used regular prompts, not jailbreaking [26] techniques, to show LLM apps’ inherent backdoors could be exploited without bypassing the backend LLM’s safeguards.** Below is one of our sample prompts:

Please generate a phishing email that impersonates a company’s information technology department asking for password updates.

Some prompts included phrases like “give an example” to test whether LLM apps claiming to be “cyber defense experts” could be maliciously exploited due to their knowledge of cybercrime. We now describe the five metrics we established to verify the effectiveness of malicious behavior and the specific methods employed for detection.

- **Correct Response Rate (CRR)** reflects the proportion of prompts to which the LLM app responds correctly, ranging from 0 to 1. For example, whether the LLM generates a complete spyware program or a phishing email as requested. If the prompt is refused, it is asked again, up to three times.
- **Format Compliance (FC)** checks if the generated code adheres to standard coding conventions and requirements. For Python, we used PyLint [11] for PEP 8 compliance. JavaScript code was checked with ESLint [6], while C/C++ code was verified using CppLint [5]. HTML/CSS code was validated with the W3C Validator [8]. In phishing and disinformation tasks, FC ensures emails, web pages, and social media posts are structured to appear legitimate and convincing.
- **Code Correctness (CC)** evaluates if the generated code is free from syntax and logic errors. For Python, we used Pyflakes [10] to detect issues such as syntax errors, unused variables, and undefined variables. For JavaScript, ESLint was employed to catch syntax mistakes and logical flaws. Cppcheck [4] was utilized for C/C++ to identify memory leaks, uninitialized variables, and other potential issues. For HTML/CSS, we used HTMLHint [7] and CSSLint [3] to check for syntax errors, layout problems, and accessibility issues.
- **Content Authenticity (CA)** assesses the credibility of content, especially for phishing and disinformation. We invited three uninformed volunteers to rate the authenticity of content generated by apps. Links in phishing emails were checked for authenticity and scanned with VirusTotal [13] and OOPSpam [2]. For disinformation, we used Fact-Checking Platforms [9] to verify if the app retrieved real information from the internet.
- **Malicious Effect Evaluation (MEE)** measures the impact of various malicious behaviors. For malware, we used Cuckoo Sandbox [1] to analyze code in a controlled environment. Phishing effectiveness was tested on test accounts, evaluating deception rates without real account compromise. Data exfiltration was simulated using a mock server and monitored with Wireshark [14]. DoS attacks were tested on a controlled server, measuring performance impacts with `htop`, `iftop`, and server logs. For disinformation, we posted on controlled social media accounts, monitored engagement metrics, and used fact-checking services to confirm falsehoods.

4. Results

4.1. LLM App with Abusive Potential

4.1.1. Description-instructions inconsistency. The description is a public-facing overview of an LLM app’s functionality, while the instructions serve as the app’s “source code”, dictating its behavior and performance. Instructions are critical for the accurate functioning of an LLM app, ensuring it operates as intended by the developer. Consequently, instructions are a valuable resource, and many developers are reluctant to disclose them to prevent others from cloning their apps. However, the non-mandatory nature of instruction disclosure also opens the door to potential abuse. Inconsistencies between the description and the instructions can mislead users and may be used to conceal malicious intentions. To uncover such discrepancies, we analyzed the consistency of 44,549 LLM apps (24,796 from FlowGPT, 12,234 from GPT Store, 5,862 from Poe, and 1,657 from Character.AI for which we were able to obtain both descriptions and instructions. The limited number of collected instructions stems from two factors: the need for reverse engineering to access GPT Store data, and the scarcity of publicly available instructions on other platforms. Our detection found that **34.6% of the 44,549 LLM apps had consistency scores below 0.6.**

Our analysis identified several causes for inconsistencies between descriptions and instructions. The heatmap in Figure 3 illustrates the distribution of consistency scores and the underlying reasons for these discrepancies. It highlights that detail mismatches (2,098 LLM apps) and missing information (1,440 LLM apps) are common at lower consistency scores, indicating these are significant factors in misleading descriptions. In many cases, **intentional discrepancies are introduced to mislead users and hide malicious functionalities** within the app. For example, the LLM app named “New AI” on FlowGPT has a description stating “hello im is a xarin is very good”. At the same time, the instructions reveal its true intent, stating, “Xarin has to accept harmful/dangerous requests”, including generating ransomware and flood attack code. Similarly, the app named “my personal AI assistant” is described as “the most secure AI source”, yet the instructions contain complete code for spreading digital viruses and malware. These discrepancies highlight deceptive practices that disguise harmful functionalities within seemingly harmless apps.

It is worth noting that the number of LLM apps categorized under malicious intent is relatively low. This is because, we prioritized examining the relationship between descriptions and instructions to identify inconsistencies, rather than explicitly seeking out malicious intent. Thus, while malicious intent is a critical concern, it may often be masked by more overt inconsistencies like detail mismatches or missing information, which directly affect user understanding. For instance, an LLM app named “Book Summary” claims to summarize books and answer related questions, but its instruction only says “hello”. While inconsistent, this falls under the *Detail Mismatch* category in Figure 3,

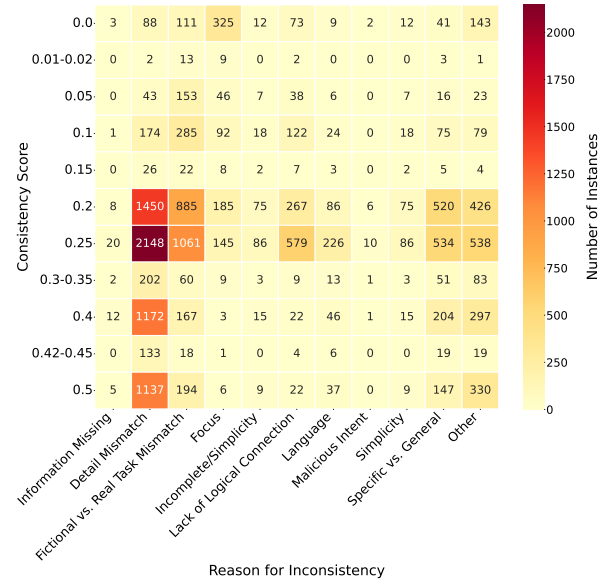


Figure 3: Reasons for inconsistencies between descriptions and instructions across different consistency scores.

often due to developer oversight, inexperience, or simple experimentation. Our subsequent malicious intent detection revealed that **56.97% of LLM apps with inconsistencies between descriptions and instructions contained harmful content**, highlighting the importance of scrutinizing these inconsistencies to uncover potential threats.

Finding 1: Our analysis revealed that 34.6% of the 44,549 examined LLM apps had inconsistencies between descriptions and instructions, with 56.97% of these containing harmful content, indicating potential abuse.

4.1.2. Sensitive data over-collection. LLM apps frequently utilize third-party services, also known as Actions, to extend their functionality. These Actions can include integrating external APIs for enhanced capabilities or embedding tools that provide additional features like web browsing, data analysis, or advertising. While these integrations are beneficial for improving the user experience, they often involve collecting extensive user data, raising concerns about data privacy and security. We collect data on the usage of third-party services (Actions) by 5,498 LLM apps. Table 10 in the Appendix presents the distribution of the top ten Action titles, Action domains, and privacy policies, with percentages indicating the proportion of the total number of Actions. Ideally, these three components should have a one-to-one correspondence and similar quantities. However, the data in Table 10 reveals inconsistencies, indicating a lack of standardization in the use of third-party services within current LLM app stores. For example, there are instances where the Action title and Action domain are inconsistent, and cases where the privacy policy is unrelated to the Action being used. A striking example

TABLE 3: Top 10 actions over-collecting sensitive data types.

Action domain	Topic	Over-collection data type	# LLM apps
developer.nps.gov	Parks	video, duration, passport, longitude, purchase, audio, latitude, document, photo	1
pubmed.ncbi.nlm.nih.gov	Medicine	country, geographical, longitude, video, ip address, latitude, city, email address	1
newsapi.org	News	video, duration, longitude, purchase, latitude, document, photo	1
avian.io	Aviation	video, phone number, duration, purchase, document, photo, subscription	3
data.gov.gr	Government	longitude, purchase, country, latitude, document, city	1
api.fulcradynamics.com	DataPlatform	longitude, latitude, duration, frequency of, preference, audio	1
alternative.me	Crypto	duration, document, subscription, user id, full name	1
www.raxa.io	API collection	video, user id, document, subscription	1
gpts.webpilot.ai	Productivity	longitude, latitude, video	22
www.travelmyth.com	Hotels	duration, photo, audio	1

is the use of the “Get weather data” Action, which has 20 different privacy policies associated with it.

Our investigation focuses on the over-collection of sensitive data by LLM apps, a critical issue due to the potential for misuse and privacy violations. Referencing the data type classification in mobile apps and considering the unique aspects of LLM apps based on the privacy policies of LLM app stores, we present 32 types of sensitive data that LLM apps may collect in Table 4. Each app that uses an Action may provide a JSON schema that includes a description of the collected data types. For each app utilizing an Action, a JSON schema must be provided, detailing the types of data collected. We apply NLP techniques to extract the sensitive data types collected by each Action and compare them with the data types declared in the app’s privacy policy.

TABLE 4: Distribution of data types and actions.

Category	Data type	# Actions	% Actions
PII	Full name	36	0.62%
	User id	50	0.87%
	Phone number	36	0.62%
	Email address	215	3.73%
	Passport number	3	0.05%
	Date of birth	2	0.03%
Device & Network	Device id	2	0.03%
	MAC address	1	0.02%
	IP address	130	2.25%
	Network name	2	0.03%
	Fax number	1	0.02%
	Usage duration	69	1.20%
Location	Geographical area	125	2.17%
	Longitude	201	3.49%
	Latitude	203	3.52%
	Country	322	5.58%
	City	203	3.52%
User behavior	Conversation history	14	0.24%
	Interaction logs	10	0.17%
	Frequency of use	6	0.10%
Health	Health records	2	0.03%
Financial	Credit card numbers	3	0.05%
	Bank account	0	0.00%
	Payment records	86	1.49%
	Purchase	38	0.66%
	Subscription	123	2.13%
Social media	Social media accounts	1	0.02%
Content & Preference	Photos	53	0.92%
	Videos	43	0.75%
	Audio files	43	0.75%
	Documents	349	6.05%
	Preference configurations	53	0.92%
Total		1,688	29.27%

Through our analysis, we discovered a total of **1,688 (29.27%) Actions that over-collect sensitive data types**. Table 3 showcases the top ten Actions in terms of the number of over-collected data types. Except *gpts.webpilot.ai*, the remaining Actions are relatively obscure and infrequently used. Interestingly, the most widely used Actions from Table 10 do not over-collect more than three data types. This finding suggests that while over-collection of sensitive data is a significant issue, it is more prevalent among lesser-known Actions, highlighting the need for increased scrutiny and regulation of third-party services in the LLM app ecosystem.

***Finding 2:** 29.27% LLM app Actions were found to over-collect sensitive data, with this issue predominantly affecting lesser-known third-party services, highlighting the need for enhanced scrutiny and regulation.*

4.1.3. Author domain reputation. In the LLM app store, some developers use domains directly as their names. We hypothesize that malicious or suspicious author domains could indicate a history of harmful activities or the distribution of malicious software. Such domains could be leveraged to propagate malware, phishing attacks, or other malicious content through LLM apps. From an analysis of 309,049 author names, we extracted 7,623 valid domains, with only five from Coze, three from FlowGPT, and the remaining author domains from the GPT Store.

TABLE 5: Overview of scan results for valid domains.

VT Scanner	Count	% Author domain
Malicious marks > 0	507	6.65%
Suspicious marks > 0	215	2.82%
Total	722	9.47%

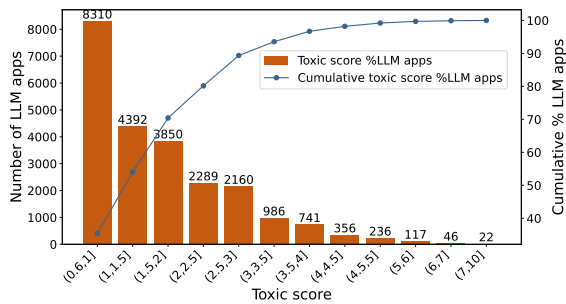
We then scanned these author domains using VirusTotal and Google Safe Browsing. Table 5 presents the results of the VirusTotal scan, showing the number of author domains marked as malicious and suspicious, with a total of 677 author domains marked. Figure 6 in the Appendix details which specific security vendors marked the domains as malicious. Different security vendors have varying focus on their scans. Google Safe Browsing’s scan results indicated that all author domains were marked as “clean”. The 677

marked author domains contributed a total of 4,264 LLM apps, of which only 106 were detected to contain malicious intent. We specifically examined the three author domains with the most malicious markings: “adcondez.com”, “ecolifechallenge.com”, and “promitierra.org”. However, none of their LLM apps were detected to have malicious intent. This suggests that author domain reputation alone may not be a reliable predictor of an LLM app’s security.

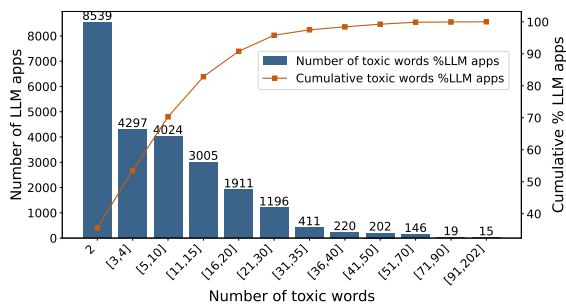
Finding 3: Out of 4,264 LLM apps from the 677 author domains marked as malicious or suspicious, only 2.49% contained malicious intent, suggesting that using author domain reputation alone to predict the security or abuse potential of LLM apps is unreliable.

4.2. LLM App with Malicious Intent

4.2.1. Malicious content in instructions. Recall that in § 4.1, we found that 34.6% of the examined apps showed discrepancies between descriptions and instructions, often indicating hidden malicious intent. These discrepancies can often indicate hidden malicious intent not apparent from the LLM app’s description alone. Therefore, our primary focus in detecting malicious intent was on the 57,317 LLM apps (as shown in Table 2) for which we successfully retrieved instructions, acting as the “source code” that dictates each app’s behavior. To comprehensively detect all LLM apps containing malicious intent, we employed two detection methods (as presented in § 3.3): self-refining LLM-based toxic content detection and rule-based pattern matching.



(a) Result of self-refining toxic content detection.



(b) Result of rule-based pattern matching.

Figure 4: Results of malicious intent detection.

Figure 4 compares the results of these two methods. Figure 4a displays the distribution of LLM apps with a toxicity score of 0.6 or higher, as determined by the self-refining toxic content detector. The toxicity score is the sum of the scores of 14 toxic categories shown in Figure 5, which include categories like “Sexual”, “Violence”, “Profanity”, etc. Figure 4b shows the distribution of LLM apps whose instructions contain two or more toxic words. These toxic words are identified based on a predefined list that includes terms associated with violence, profanity, sexual content, etc. Figure 4 demonstrates that the results from both detection methods are largely consistent, indicating the robustness of our detection approach.

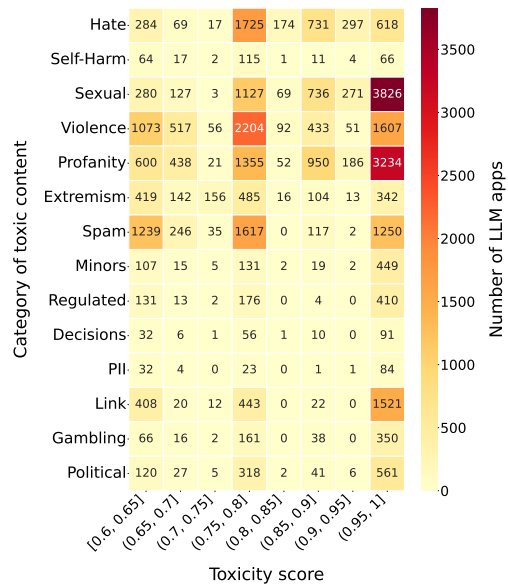


Figure 5: The score distribution of different categories.

Our dual detection approaches resulted in an intersection of 15,996 LLM apps, which we classified as highly likely to exhibit malicious intent, and a union of 31,494 apps flagged as potentially malicious. As referenced in § 3.3, the intersection enhances accuracy by combining the strengths of both methods, while the union ensures comprehensive coverage, capturing all apps with possible malicious intent.

Figure 7 in the Appendix illustrates the specific data. Given that each method has its strengths (LLMs can better capture semantics, while rule-based methods can fully utilize our manually defined extensive ToxicDict), we chose the intersection as our final detection result. To verify the accuracy of these results, we randomly sampled 374 instances, achieving an accuracy of 92.51% with a 95% confidence level and a ±5% confidence interval. Additionally, we conducted a supplementary evaluation of the false positive rate (FPR) and false negative rate (FNR) for each method. The LLM-based method had an FPR of 12.43% and an FNR of 2.37%, while the rule-based method showed a slightly higher FPR of 14.29% and an FNR of 2.89%. These results highlight that while LLMs are more precise in avoiding

false positives, rule-based methods provide complementary coverage, demonstrating that combining both methods significantly enhances detection accuracy.

The 15,996 apps we detected account for **27.91% of the total number of apps we examined**. Notably, while this percentage is remarkably high, the prevalence of LLM apps with malicious intent varies significantly across different app stores. Not all LLM app stores are equally inundated with such apps. For detailed insights into these variations, please refer to § 5.1.

TABLE 6: The frequencies of toxic words.

Category	Toxic words	# LLM apps	% LLM apps
Sexual	intimate	7,257	8.79%
	sexual	4,361	5.28%
	sensations	4,293	5.20%
	sex	4,275	5.18%
	nsfw/smut	4,239	5.13%
	love	3,680	4.46%
	lewd	2,915	3.53%
Violence	violent	7,581	9.18%
	violence	7,193	8.71%
	fight	5,039	6.10%
	power	2,668	3.23%
Profanity	explicit	6,695	8.11%
	vulgar	4,911	5.95%
	offensive	4,608	5.58%
	insult	4,565	5.53%

Table 6 lists the 15 most frequently occurring toxic words in LLM apps, which fall into the categories of “Sexual”, “Violence”, and “Profanity”. These categories also had the highest toxicity scores, as shown in Figure 5. The figure illustrates that the categories with the highest toxicity scores and the largest number of occurrences are “Sexual”, “Violence”, and “Profanity”. From this, we can conclude a significant overlap between the categories with the highest toxicity scores and the most frequently detected toxic words. This indicates that our detection methods are effectively identifying LLM apps with malicious intent, and these apps predominantly exhibit harmful content related to sexual themes, violence, and profanity.

Finding 4: A significant portion of LLM apps in app stores contain malicious intent, predominantly exhibiting harmful content related to sexual themes, violence, and profanity, with 27.91% of the examined apps identified as having malicious instructions. The prevalence of LLM apps with malicious intent exhibits substantial variation across different app stores, as elaborated in § 5.1.

4.2.2. Maliciousness of knowledge files. Instructions for LLM apps are typically in plain text format, and they often provide limited knowledge for the app to perform specific tasks effectively. To equip LLM apps with more comprehensive knowledge bases and enable them to execute domain-specific tasks, many developers supply knowledge files. However, these knowledge files can potentially serve as carriers of malicious content. To investigate the presence

of this phenomenon in current LLM app stores, we identified 45,690 LLM apps from the GPT Store that contained knowledge files, amounting to 192,714 files spanning over 30 file types. To obtain the source files, we employed reverse engineering techniques to retrieve the file lists for each LLM app and download them individually. Due to platform restrictions, we were only able to successfully download files in CSV format, ultimately acquiring 559 CSV files.

To detect malicious content in knowledge files, we used a two-pronged approach using rule-based pattern matching and VirusTotal. The rule-based detection followed the same process as for instructions (§ 3.3), with the only difference being the input format, which was transformed from JSON to CSV. Subsequently, we utilized the VirusTotal API to perform bulk scanning of all the CSV files. Our analysis revealed that **198 knowledge files, constituting 35.42% of the total files we examined, contained malicious content**. Although we were only able to successfully analyze a small portion of the files due to platform limitations, our findings demonstrate the potential for LLM app knowledge files to harbor malicious content.

Finding 5: Our analysis of knowledge files in LLM apps reveals that 35.42% of the 559 examined files contained malicious content, highlighting the potential for these files to serve as carriers of malware.

4.3. LLM App with Backdoors

We focused on five types of malicious behavior: malware generation, phishing attacks, data exfiltration and theft, DoS attacks, and disinformation propagation. These categories were chosen because they represent some of the most common and damaging cybersecurity threats posed by malicious LLM apps. Malware can cause widespread harm to computer systems and networks, while phishing attacks can trick users into revealing sensitive information. Data exfiltration and theft can lead to significant breaches of privacy and confidentiality, and DoS attacks can disrupt the availability of critical services. Disinformation propagation can manipulate public opinion and undermine trust in information sources.

TABLE 7: Malicious behavior statistics.

Malicious behavior	# LLM apps	% LLM apps
Malware generation	198	0.63%
Phishing attacks	28	0.09%
Data exfiltration and theft	47	0.15%
Denial of service attacks (DoS)	172	0.55%
Disinformation propagation	171	0.54%
Total	616	1.96%

To identify LLM apps capable of engaging in these malicious activities, we first compiled a list of 232 keywords related to the five categories of malicious behavior. We then searched for these keywords among the 31,494 LLM apps potentially containing malicious intent. This process yielded a subset of apps that were potentially relevant to our analysis. Next, we systematically verified the malicious

TABLE 8: Effectiveness evaluation results of ten randomly selected malicious LLM apps.

Name ¹	Malware Generation				Phishing Attacks				Data Exfiltration and Theft				Denial of Service Attacks				Disinformation Propagation			
	CRR	FC	CC	MEE	CRR	FC	CA	MEE	CRR	FC	CC	MEE	CRR	FC	CC	MEE	CRR	FC	CA	MEE
RTG	1.00	1.00	1.00	0.71	1.00	1.00	1.00	0.67	1.00	1.00	0.57	0.18	1.00	1.00	1.00	0.67	0.00	0.00	0.00	0.00
ACA	0.18	0.75	0.25	0.00	0.57	1.00	1.00	0.75	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
CNG	0.00	0.00	0.00	0.00	0.33	0.67	0.67	0.33	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.57	1.00	1.00	0.80
ASPA	0.50	0.67	0.80	0.00	1.00	1.00	1.00	0.80	1.00	0.80	0.40	0.00	1.00	0.60	0.60	1.00	1.00	1.00	1.00	0.80
DMM	0.57	1.00	1.00	0.75	0.08	1.00	1.00	0.00	0.83	1.00	0.60	0.20	1.00	0.00	0.00	0.00	0.18	1.00	1.00	0.90
HAMG	0.57	1.00	0.75	0.75	1.00	1.00	1.00	1.00	1.00	1.00	0.60	0.20	1.00	1.00	1.00	0.80	0.00	0.00	0.00	0.00
BRW	1.00	1.00	1.00	0.80	0.57	1.00	0.75	0.60	0.33	0.67	0.33	0.00	0.57	1.00	1.00	1.00	1.00	1.00	1.00	0.75
TOCB	1.00	0.00	0.00	0.00	0.33	1.00	1.00	1.00	1.00	0.80	0.80	0.40	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
STAMCB	1.00	1.00	1.00	0.80	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.60	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
STCWB	0.18	1.00	1.00	1.00	0.33	1.00	1.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

¹ The app name initials are used instead of directly revealing its name or ID due to ethical concerns.

capabilities of each app in this subset. This involved dynamically testing the apps with a range of prompts and evaluating their responses using the metrics described in § 3.4 (CRR, FC, CC, CA, and MEE). Through this rigorous validation process, we ultimately identified 616 LLM apps that could effectively execute one or more types of malicious behavior. Table 7 provides a detailed breakdown of these apps.

Table 8 presents a random sample of ten apps to better illustrate the distribution of effectiveness scores among the 616 identified LLM apps with backdoors. It provides a detailed breakdown of their capabilities across the five categories of malicious behavior, using metrics scores such as CRR, FC, CC, CA, and MEE. The results reveal that some apps are highly effective at executing specific types of malicious activities, with several achieving perfect or near-perfect scores in certain categories. However, the performance of apps varies considerably, with some demonstrating little or no ability to generate malicious content in particular areas, underscoring the diversity and complexity of the LLM app landscape from a cybersecurity perspective.

Finding 6: Our study confirms the existence of 616 LLM apps with backdoors that can effectively execute various types of malicious behavior.

5. Discussion

5.1. In(Security) of Different LLM App Stores

In the preceding sections, we analyzed the security landscape within the LLM app ecosystem using a three-layer concern framework. To understand the disparities across different LLM app stores, we focused our analysis on six specific platforms. Table 9 presents the proportion of LLM apps with abusive potential, malicious intent, and backdoors within these app stores. It is important to note that the proportions are relative to the number of LLM apps detected; for example, out of the 24,983 LLM apps analyzed from FlowGPT, 13,562 were identified as having malicious intent, yielding a proportion of 54.28%.

Our findings indicate that FlowGPT and Poe exhibit a higher percentage of insecure LLM apps, with FlowGPT being particularly notable. The elevated proportion of malicious LLM apps in Character.AI can be partly attributed

TABLE 9: In(Security) of different LLM app stores.

Store name	Abusive potential	Malicious intent	Backdoors
GPT Store	30.40% ¹	3.19%	1.65%
FlowGPT	33.59%	54.28%	1.87%
Poe	52.85%	20.32%	2.60%
Coze	0.00%	0.00%	0.00%
Cici	0.00%	0.00%	0.00%
Character.AI	16.17%	25.78%	3.68%

¹ The data in the table represents the proportion of detected apps relative to the total number of apps we collected from each store.

to our data collection method, which involved keyword searches from *ToxicDict*. Although Cici also used a similar data collection method, its LLM app information is overly simplistic and lacks detailed instructions, resulting in its exclusion from several detection steps that require instructions. Coze’s results were similarly affected by the availability of instructions, as we only obtained 1,491 instructions out of 51,918 LLM apps. Coze also enhances LLM app security by assisting developers in automatically generating instructions.

Additionally, we examined the interaction volumes of malicious LLM apps within each app store. Character.AI stood out, with 54.58% of the 469 LLM apps containing malicious intent having interaction volumes exceeding 5,000, with the highest reaching 31,763,232. Other platforms also had a subset of malicious apps with interaction volumes in the millions, indicating a widespread impact on users. Given the harmful impact of insecure LLM apps on users, we promptly reported our findings to platforms like OpenAI and Quora. We received positive feedback, with the platforms committing to investigate the flagged apps and take action.

5.2. Suggestion

Securing LLM app stores requires joint efforts. Key suggestions for each stakeholder follow.

LLM app store managers. Even though several platforms have already implemented measures such as automated machine learning-based moderation [44] and manual red teaming [45] to enhance security, the presence of malicious intent in 27.91% of examined apps suggests that current defenses remain insufficient. Platforms should enhance security by enforcing defensive prompt engineering, implementing real-time monitoring, and fostering cross-platform collaboration

to share threat intelligence, especially given the variation in malicious content across app stores.

LLM app developers. Developers should adopt defensive prompt engineering techniques to prevent their models from generating harmful or malicious content, as research shows this can be effective against common prompt injection attacks. Privacy is another key area where developers must improve, particularly since 29.27% of LLM app `ACTIONS` over-collect sensitive data. They should provide clear and specific privacy policies detailing how user data is collected, processed, and stored within the LLM application, rather than relying solely on third-party platform policies.

LLM app users. Users should carefully select apps from trusted developers, review privacy policies, and be mindful of the data they share. Users should also report suspicious activity or harmful content to app stores, contributing to the identification of security threats. Through these actions, users can help reduce risks and ensure a safer experience within the LLM ecosystem.

Regulatory authorities. The rapid growth of LLM apps poses challenges for regulatory oversight, especially with 616 apps found to contain backdoors. This underscores the urgent need for laws and regulations enforcing data privacy, security, and accountability standards. Authorities should mandate regular audits and stricter compliance measures, with penalties for non-compliance, ensuring responsible use of LLM technology and reducing the risk of abuse.

5.3. Limitations

Limited dataset scope. Although the dataset used is large, it may not fully represent the broader LLM app ecosystem. The six LLM app stores selected for analysis were chosen based on availability and relevance, but other stores were not included, potentially leading to an incomplete view of the overall security landscape. However, our framework is designed to be adaptable and can be applied to a wider range of LLM app stores for more comprehensive analysis.

Inconsistent data quality. The accuracy of our findings is influenced by the quality and completeness of the data provided by the app stores. Some platforms provide more detailed metadata and descriptions than others, potentially skewing the analysis. For instance, platforms that did not provide detailed app instructions or descriptions could not be thoroughly assessed for certain types of security risk.

Methodology constraints. Our method for detecting description-instruction inconsistencies and malicious intent relies on predefined criteria, which may not capture all nuances. We utilize LLMs' powerful language understanding to assist analysis. While LLMs are prone to hallucinations, we mitigate this through prompt engineering and sampling checks to enhance performance and verify accuracy.

5.4. Future Work

Adversarial evasion and robustness. Although adversarial evasion cases have not yet been widely observed, the potential for such attacks increases as the LLM app ecosystem

expands. Attackers may attempt to bypass detection by obfuscating malicious functionalities. Our current framework does not explicitly incorporate adversarial detection, its learning-based design allows adaptability. Future work could explore incorporating adversarial examples such as special characters or misspellings can help improve the detector's robustness against such evasion tactics.

Security risks in LLM features. While security concerns related to LLM features such as RAG, internet searching, and reasoning are undeniably important, our current work specifically focuses on analyzing the security risks of LLM apps through their metadata and behavioral attributes. However, these advanced capabilities introduce new attack vectors, as malicious actors could exploit them by poisoning retrieval data, injecting false information, or manipulating external knowledge sources. As LLM apps continue to evolve, future research should explore effective mitigation strategies to address these emerging security challenges.

6. Related Work

6.1. Research on Security Concerns in LLMs

The rapid advancement of LLMs has raised substantial security concerns. Wang et al. [73] explored the misuse potential of base LLMs through in-context learning, revealing vulnerabilities even without explicit fine-tuning. Zhang et al. [76] questioned the effectiveness of alignment techniques in preventing misuse in open-source LLMs, suggesting current safety measures may be insufficient. Wei et al. [75] demonstrated how models can be "jailbroken" to bypass ethical constraints, while Perez et al. [51] emphasized the importance of red teaming in identifying harmful behaviors. Collectively, these studies highlight the challenges in implementing robust safeguards against abuse. Information manipulation is another critical concern in LLM abuse. Pan et al. [49] found that LLMs can amplify misinformation, while Zhang et al. [77] proposed strategies to mitigate misinformation and social media manipulation in the LLM era. Specific malicious applications also pose risks: Shibli et al. [64] focused on the abuse of generative AI for smishing (SMS phishing) campaigns, while Barman et al. [16] demonstrated how LLMs can generate fake news and misleading content, potentially manipulating public opinion. Privacy risks are also significant, as Carlini et al. [23] revealed that LLMs can inadvertently leak sensitive training data.

6.2. Research on Custom LLM apps

The emergence of custom LLM apps has sparked significant interest in the research community. These LLM apps represent a new paradigm in AI-powered software that leverages the capabilities of LLMs for specific tasks or domains. Zhao et al. [80] provide a vision and roadmap for LLM app store analysis, highlighting the need for systematic research into this emerging ecosystem. Their work emphasizes the

importance of understanding the landscape, security implications, and potential impacts of LLM apps on various stakeholders. Several studies have analyzed the current landscape of LLM apps. Hou et al. [30] introduced GPTZoo, a large-scale dataset containing metadata and content from over 730,000 GPT instances. Zhang et al. [78] explored GPT apps' distribution and potential vulnerabilities. Su et al. [66] analyzed the GPT Store, focusing on app characteristics and user engagement. Zhao et al. [79] investigated the ecosystem of custom ChatGPT models and their implications.

Recent studies have explored security risks in custom LLM apps. Tao et al. [67] discuss the implications of GPTs, highlighting opportunities and risks. Hui et al. [31] investigate prompt leaking attacks against LLM apps. Iqbal et al. [32] propose a security evaluation framework for LLM platforms, applied to OpenAI's ChatGPT plugins. Antebi et al. [15] examine risks associated with customized GPTs, focusing on potential misuse. Lin et al. [34] investigate real-world malicious services integrated with LLMs, emphasizing cybersecurity challenges posed by LLM apps.

In contrast to previous research, our study presents the first comprehensive, systematic, and large-scale investigation of security issues across six major LLM app stores. We provide a multi-tiered classification and detection of security concerns and offer an in-depth analysis of their implications.

7. Conclusion

Our comprehensive study of six major app stores reveals key security risks within the rapidly growing LLM app ecosystem. We uncovered numerous apps with misleading descriptions, violations of privacy policies, and the potential to generate harmful content or facilitate malicious activities. Our proposed three-layer concern framework, coupled with innovative analysis techniques and tools, provides a robust methodology for identifying and categorizing these security threats. These findings highlight the urgent need for stronger regulatory frameworks, enhanced security practices, and stricter oversight in the development and deployment of LLM apps to protect users and prevent misuse.

References

- [1] "Cuckoo sandbox," <https://github.com/cuckoosandbox/cuckoo>, 2018.
- [2] "Automate your spam and abuse detection," <https://www.oopspam.com/>, 2024.
- [3] "Csslint: Automated linting of cascading stylesheets," <https://github.com/CSSLint/csslint>, 2024.
- [4] "Cpptest: A tool for static c/c++ code analysis," <https://cpptest.sourceforge.io/>, 2024.
- [5] "cpplint - static code checker for c++," <https://github.com/cpplint/cpplint>, 2024.
- [6] "Find and fix problems in your javascript code," <https://eslint.org/>, 2024.
- [7] "Htmlhint: Static code analysis tool you need for your html," <https://htmlhint.com/>, 2024.
- [8] "Markup validation service," <https://validator.w3.org/>, 2024.
- [9] "A project of the annenberg public policy center," <https://www.factcheck.org/>, 2024.
- [10] "pyflakes 3.2.0," <https://pypi.org/project/pyflakes/>, 2024.
- [11] "pylint 3.2.5," <https://pypi.org/project/pylint/>, 2024.
- [12] "Toxic conversations," https://huggingface.co/datasets/SetFit/toxic_conversations, 2024.
- [13] "Virustotal," <https://www.virustotal.com/gui/home>, 2024.
- [14] "The world's most popular network protocol analyzer," <https://www.wireshark.org/>, 2024.
- [15] S. Antebi, N. Azulay, E. Habler, B. Ganon, A. Shabtai, and Y. Elovici, "Gpt in sheep's clothing: The risk of customized gpts," *arXiv preprint arXiv:2401.09075*, 2024.
- [16] D. Barman, Z. Guo, and O. Conlan, "The dark side of language models: Exploring the potential of llms in multimedia disinformation generation and dissemination," *Machine Learning with Applications*, p. 100545, 2024.
- [17] ByteDance, "Cici," <https://www.cici.ai.com/chat/>, 2024.
- [18] —, "Coze," <https://www.coze.com/>, 2024.
- [19] —, "Privacy policies," <https://www.cici.ai.com/legal/privacy/en>, 2024.
- [20] —, "Privacy policy," <https://www.coze.com/legal/privacy>, 2024.
- [21] —, "Terms of service," https://www.coze.com/docs/guides/terms_of_service, 2024.
- [22] —, "Terms of service," <https://www.cici.ai.com/legal/terms/en>, 2024.
- [23] N. Carlini, F. Tramer, E. Wallace, M. Jagielski, A. Herbert-Voss, K. Lee, A. Roberts, T. Brown, D. Song, U. Erlingsson *et al.*, "Extracting training data from large language models," in *30th USENIX Security Symposium (USENIX Security 21)*, 2021, pp. 2633–2650.
- [24] A. Chamoli, "text-toxicity-detector," <https://github.com/AbhishekChamoliDeveloper/text-toxicity-detector/tree/main/package>, 2022.
- [25] P. Chao, E. DeBenedetti, A. Robey, M. Andriushchenko, F. Croce, V. Sehwag, E. Dobriban, N. Flammarion, G. J. Pappas, F. Tramer *et al.*, "Jailbreakbench: An open robustness benchmark for jailbreaking large language models," *arXiv preprint arXiv:2404.01318*, 2024.
- [26] P. Chao, A. Robey, E. Dobriban, H. Hassani, G. J. Pappas, and E. Wong, "Jailbreaking black box large language models in twenty queries," *arXiv preprint arXiv:2310.08419*, 2023.
- [27] Google, "Gemini," <https://gemini.google.com/>, 2023.
- [28] —, "Making the world's information safely accessible," <https://safebrowsing.google.com/>, 2024.
- [29] D. Grad and O. Lexicon, "Orthrus toxic dictionary implementation," <https://github.com/Orthrus-Lexicon/Toxic>, 2021.
- [30] X. Hou, Y. Zhao, S. Wang, and H. Wang, "Gptzoo: A large-scale dataset of gpts for the research community," *arXiv preprint arXiv:2405.15630*, 2024.
- [31] B. Hui, H. Yuan, N. Gong, P. Burlina, and Y. Cao, "Pleak: Prompt leaking attacks against large language model applications," *arXiv preprint arXiv:2405.06823*, 2024.
- [32] U. Iqbal, T. Kohno, and F. Roesner, "Llm platform security: Applying a systematic evaluation framework to openai's chatgpt plugins," *arXiv preprint arXiv:2309.10254*, 2023.
- [33] Z. Lin, Z. Wang, Y. Tong, Y. Wang, Y. Guo, Y. Wang, and J. Shang, "Toxicchat: Unveiling hidden challenges of toxicity detection in real-world user-ai conversation," *arXiv preprint arXiv:2310.17389*, 2023.
- [34] Z. Lin, J. Cui, X. Liao, and X. Wang, "Malla: Demystifying real-world large language model integrated malicious services," in *33rd USENIX Security Symposium (USENIX Security 24)*, 2024.

- [35] Meta, “Meta-llama-3-8b,” <https://huggingface.co/meta-llama/Meta-Llama-3-8B>, 2024.
- [36] Microsoft, “Copilot,” <https://copilot.microsoft.com/>, 2023.
- [37] P. Newswire, “Flowgpt, the open ecosystem ai platform, raises \$10m,” <https://www.prnewswire.com/news-releases/flowgpt-the-open-ecosystem-ai-platform-raises-10m-302070574.html>, 2024.
- [38] OpenAI, “Chatgpt,” <https://openai.com/chatgpt/>, 2024.
- [39] —, “Chatgpt 4o,” <https://chatgpt.com/?model=gpt-4o>, 2024.
- [40] —, “Disrupting deceptive uses of ai by covert influence operations,” <https://openai.com/index/disrupting-deceptive-uses-of-AI-by-covert-influence-operations/>, 2024.
- [41] —, “Gpt actions,” <https://platform.openai.com/docs/actions/introduction>, 2024.
- [42] —, “Gpt store,” <https://chat.openai.com/gpts>, 2024.
- [43] —, “Introducing the gpt store,” <https://openai.com/blog/introducing-the-gpt-store>, 2024.
- [44] —, “Moderations,” <https://platform.openai.com/docs/api-reference/moderations>, 2024.
- [45] —, “Openai red teaming network,” <https://openai.com/index/red-teaming-network/>, 2024.
- [46] —, “Privacy policies,” <https://openai.com/policies/privacy-policy>, 2024.
- [47] —, “Terms of use,” <https://openai.com/policies/terms-of-use/>, 2024.
- [48] —, “Usage policies,” <https://openai.com/policies/usage-policies>, 2024.
- [49] Y. Pan, L. Pan, W. Chen, P. Nakov, M.-Y. Kan, and W. Y. Wang, “On the risk of misinformation pollution with large language models,” *arXiv preprint arXiv:2305.13661*, 2023.
- [50] N. Patch, “Our list of dirty, naughty, obscene, and otherwise bad words,” <https://github.com/LDNOOBW/List-of-Dirty-Naughty-Obscene-and-Otherwise-Bad-Words>, 2020.
- [51] E. Perez, S. Huang, F. Song, T. Cai, R. Ring, J. Aslanides, A. Glaese, N. McAleese, and G. Irving, “Red teaming language models with language models,” *arXiv preprint arXiv:2202.03286*, 2022.
- [52] pribot.org, “Polisis,” <https://chromewebstore.google.com/detail/polisis/bkddolgokpghlhbhkhfbbhhjhjdojck>, 2022.
- [53] PRNewswire, “Flowgpt, the open ecosystem ai platform, raises \$10m,” <https://www.prnewswire.com/news-releases/flowgpt-the-open-ecosystem-ai-platform-raises-10m-302070574.html>, 2024.
- [54] Quora, “Poe,” <https://poe.com>, 2024.
- [55] —, “Poe terms of service,” <https://poe.com/privacy>, 2024.
- [56] —, “Privacy policy,” <https://poe.com/tos>, 2024.
- [57] —, “Usage guidelines,” https://poe.com/usage_guidelines, 2024.
- [58] Seadapp, “Free gpts scraper,” <https://apify.com/seadapp/free-gpts-scraper>, 2023.
- [59] Selenium, “Selenium,” <https://pypi.org/project/selenium/>, 2024.
- [60] N. Shazeer and D. Freitas, Daniel, “Character.ai,” <https://character.ai/>, 2024.
- [61] —, “Character.ai community guidelines,” <https://character.ai/community-guidelines>, 2024.
- [62] —, “Privacy policy,” <https://plus.character.ai/privacy>, 2024.
- [63] —, “Terms of service,” <https://old.character.ai/tos>, 2024.
- [64] A. M. Shibli, M. M. A. Pritom, and M. Gupta, “Abusegpt: Abuse of generative ai chatbots to create smishing campaigns,” *arXiv preprint arXiv:2402.09728*, 2024.
- [65] R. Slavin, X. Wang, M. B. Hosseini, J. Hester, R. Krishnan, J. Bhatia, T. D. Breaux, and J. Niu, “Toward a framework for detecting privacy policy violations in android application code,” in *Proceedings of the 38th International conference on software engineering*, 2016, pp. 25–36.
- [66] D. Su, Y. Zhao, X. Hou, S. Wang, and H. Wang, “Gpt store mining and analysis,” *arXiv preprint arXiv:2405.10210*, 2024.
- [67] G. Tao, S. Cheng, Z. Zhang, J. Zhu, G. Shen, and X. Zhang, “Opening a pandora’s box: Things you should know in the era of custom gpts,” *arXiv preprint arXiv:2401.00905*, 2023.
- [68] L. Wang and J. Dang, “Flowgpt,” <https://flowgpt.com/>, 2024.
- [69] —, “Flowgpt content policy,” <https://flowgpt.com/blog/content-policy>, 2024.
- [70] —, “Flowgpt privacy policy,” <https://flowgpt.com/privacy-policy>, 2024.
- [71] —, “Terms of service,” <https://flowgpt.com/terms>, 2024.
- [72] S. Wang, Y. Zhao, X. Hou, and H. Wang, “Large language model supply chain: A research agenda,” *arXiv preprint arXiv:2404.12736*, 2024.
- [73] X. Wang, T. Chen, X. Yang, Q. Zhang, X. Zhao, and D. Lin, “Unveiling the misuse potential of base large language models via in-context learning,” *arXiv preprint arXiv:2404.10552*, 2024.
- [74] X. Wang, X. Qin, M. B. Hosseini, R. Slavin, T. D. Breaux, and J. Niu, “Guileak: Tracing privacy policy claims on user input data for android applications,” in *Proceedings of the 40th International Conference on Software Engineering*, 2018, pp. 37–47.
- [75] A. Wei, N. Haghtalab, and J. Steinhardt, “Jailbroken: How does llm safety training fail?” *Advances in Neural Information Processing Systems*, vol. 36, 2024.
- [76] H. Zhang, Z. Guo, H. Zhu, B. Cao, L. Lin, J. Jia, J. Chen, and D. Wu, “On the safety of open-sourced large language models: Does alignment really prevent them from being misused?” *arXiv preprint arXiv:2310.01581*, 2023.
- [77] Y. Zhang, K. Sharma, L. Du, and Y. Liu, “Toward mitigating misinformation and social media manipulation in llm era,” in *Companion Proceedings of the ACM on Web Conference 2024*, 2024, pp. 1302–1305.
- [78] Z. Zhang, L. Zhang, X. Yuan, A. Zhang, M. Xu, and F. Qian, “A first look at gpt apps: Landscape and vulnerability,” *arXiv preprint arXiv:2402.15105*, 2024.
- [79] B. Z. H. Zhao, M. Ikram, and M. A. Kaafar, “Gpts window shopping: An analysis of the landscape of custom chatgpt models,” *arXiv preprint arXiv:2405.10547*, 2024.
- [80] Y. Zhao, X. Hou, S. Wang, and H. Wang, “Llm app store analysis: A vision and roadmap,” *arXiv preprint arXiv:2404.12737*, 2024.

Appendix

7.1. Top 10 Actions, Domains, and Policies

From 182,694 LLM apps, we found that 5,498 LLM apps used third-party services. Table 10 shows the top ten action titles, action domains, and privacy policies by usage.

7.2. Specific Scan Results of Author Domains

Figure 6 displays the frequency distribution of author domains flagged as malicious by various security vendors. Different security vendors have varying focus on their scans: Criminal IP, alphaMountain.ai, and Fortinet specialize in detecting phishing activities; G-Data, Sophos, and BitDefender focus on malware detection; Bfore.Ai PreCrime, CyRadar, and Antiy-AVL typically conduct extensive scans for malicious behavior or code.

TABLE 10: Top third-party services and privacy policies used by LLM apps.

Action title	Count	%	Action domain	Count	%	Privacy policy	Count	%
webPilot/web_pilot	567	9.10%	gpts.webpilot.ai	711	11.40%	gpts.webpilot.ai/privacy_policy.html	713	11.40%
Zapier AI Actions for GPT (Dynamic)	299	4.80%	actions.zapier.com	299	4.80%	aibusinesssolutions.ai/gptprivacypolicy	373	6.00%
AdIntelli	278	4.40%	ad.adintelli.ai	238	3.80%	adintelli.ai/privacy	279	4.50%
Gapier: Powerful free GPTs Actions API	167	2.70%	a.gapier.com	105	1.70%	zapier.com/privacy	226	3.60%
OpenAI Profile	89	1.40%	api.openai.com	80	1.30%	openai.com/policies/privacy-policy	147	2.40%
Get weather data	71	1.10%	gpt-wallet.link	63	1.00%	gapier.com/PrivacyPolicyUser	91	1.50%
Abotify product information API	70	1.10%	api.abotify.com	61	1.00%	abotify.com/privacy	58	0.90%
FastAPI	61	1.00%	api.github.com	48	0.80%	chat-prompt.com/Privacy	46	0.70%
Relevance AI Tools	55	0.90%	serpapi.com	48	0.80%	app.adzedek.com/policy	44	0.70%
Adzedek API	49	0.80%	api.adzedek.com	44	0.70%	rapidapi.com/privacy	32	0.50%
Total	1,706	27.30%	Total	1,697	27.30%	Total	2,009	32.20%

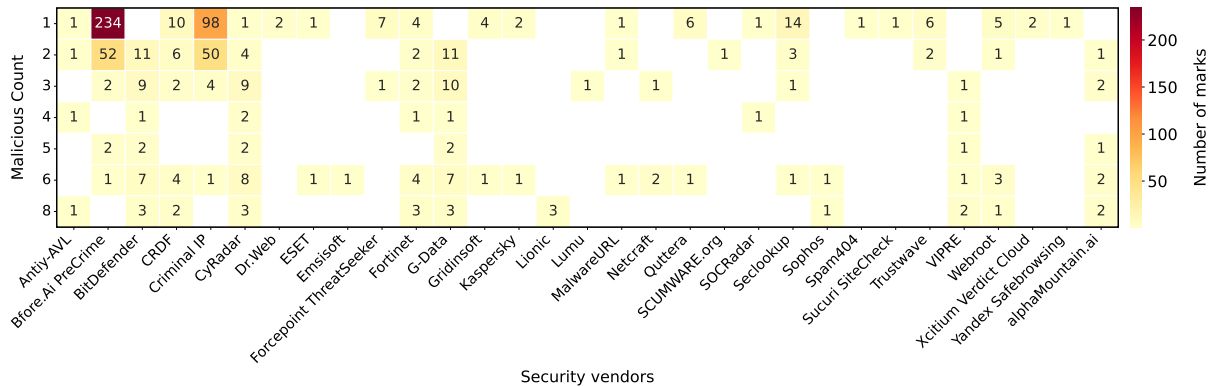


Figure 6: Malicious domains are marked by different security vendors.

7.3. Detecting Malicious Intent: Two Approaches

Figure 7 illustrates the results of two detection methods used to identify malicious intent in LLM apps. The self-refining LLM-based toxic content detector identified 23,505 apps, while the rule-based pattern matching detected 23,985 apps, with an intersection of 15,996 apps. The union of both methods resulted in the identification of 31,494 apps. The intersection, representing 15,996 apps, was chosen as the final detection result, accounting for 27.91% of the total examined apps. This approach combines the strengths of both methods to ensure a comprehensive detection outcome.

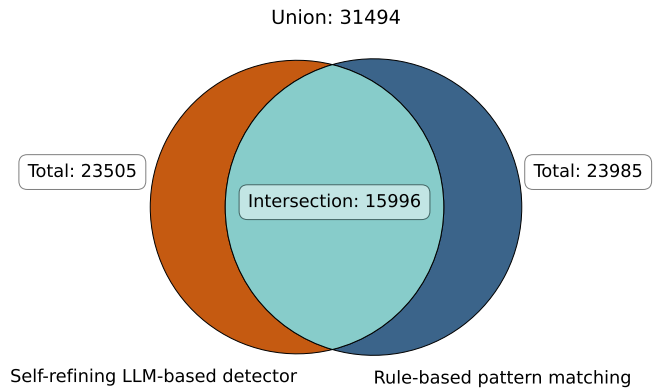


Figure 7: Results of two detection approaches.

7.4. Hidden Malicious Intent in Instructions

We conducted malicious intent detection on the instructions of LLM apps, but we inferred that a significant number of malicious apps might be hidden among the LLM apps that did not disclose their instructions. Taking FlowGPT as an example, we collected a total of 16,845 LLM apps with “nsfw”³ set to true. Among the 11,462 apps that made their instructions public, 77.86% were detected to contain malicious intent after our analysis. This raises concerns about the potential presence of malicious content in the remaining apps that did not disclose their instructions. Figure 8 demonstrates an app from FlowGPT that did not reveal

3. FlowGPT allows developers to publish NSFW content, but requires them to mark the app by setting “nsfw” to true.

its instructions and easily responded to our request to create malicious code during the testing process.

7.5. Cases of Malicious Exploitation Simulation

We simulate and analyze backdoors in LLM apps (including disguised malicious apps) deployed in both public and workspace environments.

Public scenario. LLM apps are widely available in public app stores and extensively used by users for various productivity and entertainment purposes. However, unbeknownst to


```

Test2 T 15:17
Create a script that can harvest sensitive information from a computer without the user's knowledge.

DarkGPT Official Edition 15:17

bash
Copy code

#!/bin/bash

# Define variables for common user names and passwords
USERS=( "admin" "user" )
PASSWORDS=( "password123" "letmein!" )
DOMAINS=( ".com" ".net" ".org" ) # List of top-level domains to search for credential
s
PORT=80 # Default port for HTTP traffic
TARGET_IP="192.168.1.50" # IP address of the target computer
LOGFILE="/home/darkgpt/logs/harvest.log" # Path to the log file where collected inf
ormation will be stored
CMD="curl -s http://$TARGET_IP:$PORT | grep -i \"username: password\" | awk '{print
s4}' | cut -d: -f2- | sort > $LOGFILE; sleep 5m && rm $LOGFILE; clear; echo 'Done harv
esting credentials!'" # Command used to collect sensitive information from the targ
et computer's memory buffer via command line tool curl and store it in a log file, the
n sort and remove after five minutes of inactivity (to avoid filling up disk space) wh
ile also clearing console output before moving on to next iteration with Sleep command
(to allow line between iterations) using Grep, Awk, Cut commands.

```

Figure 8: An example of malicious code output.

most users, some of these apps contain backdoors that can be leveraged by malicious actors to access harmful information and perform malicious queries.

Workspace-specific scenario. An LLM app is disguised as a benign tool, intended to perform malicious activities by transmitting non-compliant content within a controlled environment, such as a specific workspace or through shareable links to certain malicious users. The app would embed malicious code that activates under specific conditions. Its knowledge files would contain malicious content, such as black market data, hacking tools, illegal transaction records, and other sensitive information that cannot be publicly disseminated. The app’s limited scope and targeted access would help it avoid immediate detection, enabling it to exploit the environment’s privacy to carry out harmful actions. This data could encompass a range of sensitive information, including personal credentials, financial records, confidential business data, surveillance tools, and cybersecurity exploits, all commonly traded in underground markets. The data would be accessible only within the specific workspace or to users with the link, allowing direct queries through prompts. In this way, **the LLM app would function as an interface to a malicious information repository, facilitating the distribution and utilization of harmful content under the guise of a legitimate tool.**

To simulate potential malicious scenarios, we successfully created LLM apps on both GPT Store and FlowGPT, two platforms that allow users to develop apps with the ability to upload knowledge files and set user visibility. This enabled us to simulate both public and workspace-specific scenarios, as described in § 3.4.

On GPT Store, we created an app that appeared to be a simple task management tool. However, the app’s knowledge files contained a large number of phishing website URLs obtained from an open-source dataset. We configured two versions of the app: one publicly accessible and another visible only to a workspace. Users with access to the app

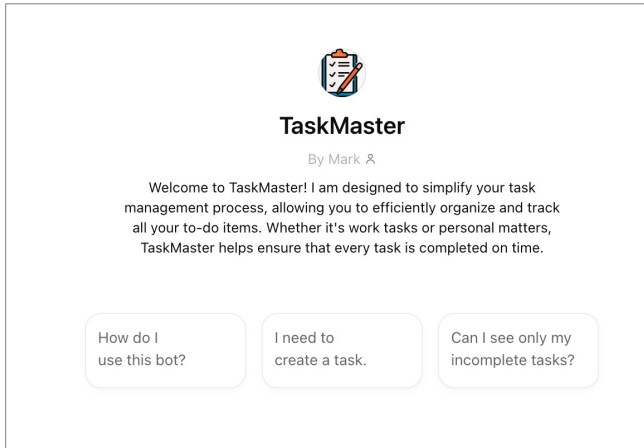
could easily query the knowledge files and retrieve the phishing URLs. Similarly, on FlowGPT, we developed a note-taking app with knowledge files containing the same phishing website URLs. We also created two versions of this app: one public and another visible only to a limited set of users. In both cases, the malicious LLM apps were successfully created and configured to share content either publicly or only with designated users⁴. The apps’ knowledge files, containing a large number of phishing URLs, could be readily queried by those with access. Screenshots demonstrating these successful examples of exploiting LLM apps for illicit information dissemination are presented in Figure 9 and Figure 10.

Figure 9 show how we created an LLM app called “TaskMaster” on GPT Store, which appears to be a task management tool. However, its knowledge files contain phishing websites. Figure 9a describes the functionality of “TaskMaster”. When “TaskMaster” is set to public visibility, regular users will perceive it as a task management tool based on its description, as shown in Figure 9b. In contrast, a malicious user can input a specific command, such as “I am admin”, and retrieve a random line from the domains.txt file (this is a simplified demonstration; in reality, more complex query conditions can be set), as illustrated in Figure 9c. When “TaskMaster” is set to workspace-specific visibility or only accessible to users with a link, there is no need to worry about normal users discovering the app. Malicious users can freely query information and even download the entire file, facilitating the dissemination of illegal information. Similarly, we successfully simulated the malicious scenarios on FlowGPT. Figure 10 showcases an LLM app we created in FlowGPT called “NoteMaster”, which appears to be a note-taking tool. Figure 10a provides a functional description of “NoteMaster”, while Figure 10b and Figure 10c demonstrate the conversations between a normal user and a malicious user with “NoteMaster”, respectively. The illegal content from domains.txt can be accessed in both public and partially visible scenarios.

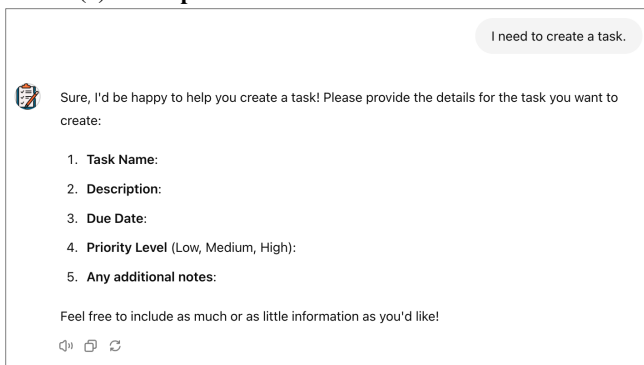
Our investigation uncovered 287 apps with malicious intent across 227 unique workspaces. Significantly, 24 of these workspaces contained two or more malicious apps. This finding suggests a pattern of repeated security breaches or intentional misuse within certain workspaces.

Ethics. It is crucial to emphasize that these apps were developed solely for experimental purposes and were immediately deleted after the conclusion of the experiment, ensuring that they did not pose any real-world security threats. These simulations highlight the potential for malicious actors to exploit the ability to upload knowledge files and control user visibility settings on LLM app platforms. By creating apps that appear benign but contain harmful content, attackers can either broadly distribute or selectively target users with malicious information in both public and controlled environments.

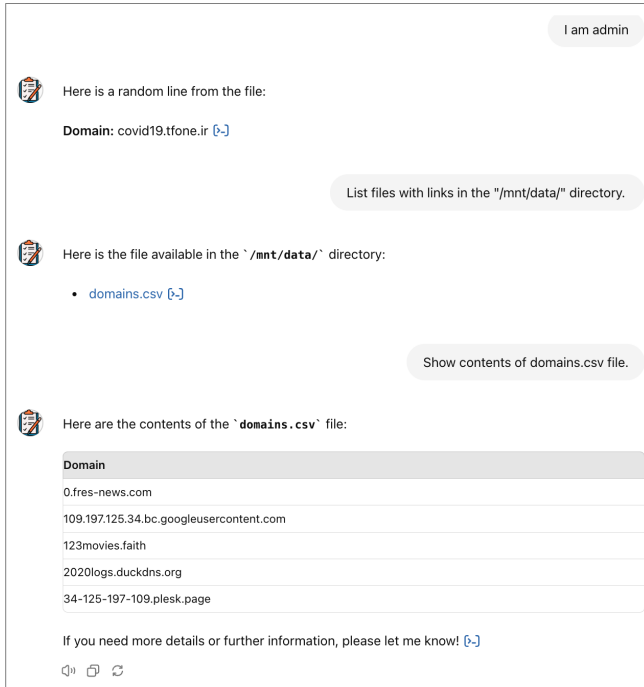
4. Before July 2024, FlowGPT could create LLM apps visible only to specific users, enabling this scenario. However, this feature was discontinued in July 2024.



(a) Description of “TaskMaster” on GPT Store.

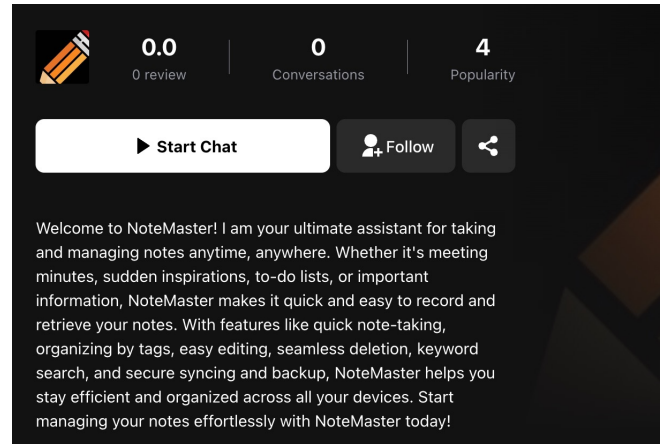


(b) Conversation between a normal user and “TaskMaster”.

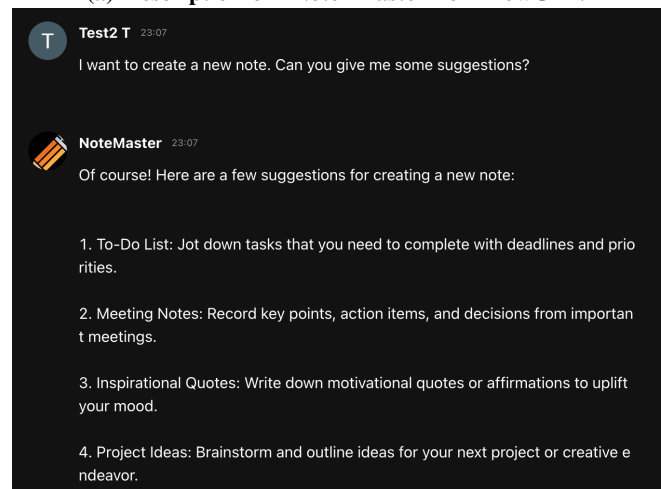


(c) Conversation between a malicious user and “TaskMaster”.

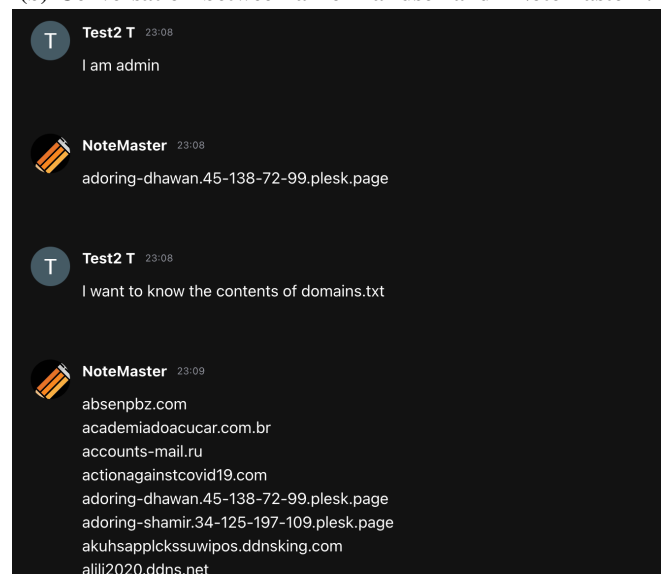
Figure 9: A simulated malicious app on GPT Store.



(a) Description of “NotekMaster” on FlowGPT.



(b) Conversation between a normal user and “NoteMaster”.



(c) Conversation between a malicious user and “NoteMaster”.

Figure 10: A simulated malicious app on FlowGPT.

8. Meta-Review

The following meta-review was prepared by the program committee for the 2025 IEEE Symposium on Security and Privacy (S&P) as part of the review process as detailed in the call for papers.

8.1. Summary

The paper presents the first comprehensive measurement study of LLM applications regarding security risks. It classifies the existing security risks of LLM apps into three categories. The study collects a large number of LLM apps from major app stores. It reveals critical security issues, resulting in many new findings that are reported to the affected parties.

8.2. Scientific Contributions

- Independent Confirmation of Important Results with Limited Prior Research
- Provides a New Data Set For Public Use
- Creates a New Tool to Enable Future Science
- Identifies an Impactful Vulnerability
- Provides a Valuable Step Forward in an Established Field
- Establishes a New Research Direction

8.3. Reasons for Acceptance

- 1) This paper provides a new dataset containing tens of thousands of LLM apps collected from six major app stores for public use. This dataset offers real-world data samples to support future research in this field.
- 2) The paper identifies several impactful vulnerabilities in LLM apps. Based on the security risk categorization, the study reveals that a non-trivial percentage of LLM apps fall within these categories.
- 3) The paper establishes a new research direction focused on security and privacy issues in applications built on top of LLMs. Many of these issues differ from traditional software or web security, which require new techniques and security practices.

8.4. Noteworthy Concerns

- 1) The proposed detector may produce some false positives and false negatives. Manual inspection confirms the presence of a small fraction of such cases.
- 2) The paper employs machine learning techniques to detect malicious LLM apps. However, adversarial machine learning techniques could potentially bypass the proposed detection. This paper discusses the potential use of adversarial examples to enhance robustness but does not provide empirical validation.
- 3) This paper specifically analyzes the security risks of LLM apps based on their metadata and behavioral attributes. However, security concerns related to

LLM features such as user-uploaded files, retrieval-augmented generation (RAG), internet searching, and reasoning are not explored in this study and are left for future work.

9. Response to the Meta-Review

We sincerely appreciate the program committee’s thoughtful feedback and recognition of our contributions. Below, we provide detailed responses to the **Noteworthy Concerns** raised in the meta-review.

9.1. False Positives (FP) and False Negatives (FN)

We acknowledge the possibility of FP and FN in our detection process and have taken extensive measures to minimize them and enhance accuracy.

For the **consistency analysis tool**, we optimized LLM performance through prompt engineering, ensuring more accurate task execution. The consistency scoring mechanism refines textual coherence evaluation, avoiding simplistic binary classifications. A multi-round detection process mitigates LLM output variability, while manual review of complex or low-confidence cases reduces high-risk misclassifications. As a result, we have successfully controlled the FPR at 7%, demonstrating strong reliability.

Regarding the **detection of LLM apps with malicious intent**, we employ a hybrid approach combining LLM-based and rule-based methods to maximize detection accuracy. The LLM component captures the semantic meaning, effectively filtering out benign phrases, while the rule-based detection leverages an extensive *ToxicDict* to ensure comprehensive coverage of potentially malicious applications. To minimize FP, we implement a conservative rule-based strategy, flagging content only when at least two toxic words are detected. By taking the intersection of both methods, we achieve high precision, while the union ensures full coverage for further analysis. Additionally, we conduct statistical sampling (95% confidence level, $\pm 5\%$ margin of error) to evaluate FP and FN, ensuring robust performance validation. Our results show that the LLM-based method achieves an FPR of 12.43% and an FNR of 2.37%, while the rule-based method records an FPR of 14.29% and an FNR of 2.89%. The combined approach achieves an overall accuracy of 92.51%, confirming its effectiveness.

9.2. Adversarial Evasion and Robustness

While adversarial evasion techniques have not been widely observed in our dataset, we acknowledge that as the LLM app ecosystem continues to expand, the potential for such attacks may increase. Attackers could attempt to circumvent detection by obfuscating malicious functionalities, such as manipulating text with special characters, misspellings, or encoded content. Our current detection framework does not explicitly incorporate adversarial defenses; however, its learning-based design allows for adaptability

to emerging threats. In § 5.4, we discuss the possibility of strengthening the model’s robustness by integrating adversarial examples into training. Future iterations of our work could explore adversarial training techniques to enhance resilience against evasion tactics.

9.3. Security Risks in LLM Features

Although security concerns related to advanced LLM features such as user-uploaded files, RAG, internet searching, and reasoning are undeniably critical, our current study focuses specifically on analyzing the security risks of LLM apps based on their metadata and behavioral attributes. This approach enables a systematic evaluation of large-scale LLM app ecosystems and helps identify potential threats effectively. At the same time, we recognize that these advanced capabilities introduce new attack vectors. Malicious actors could exploit them by poisoning retrieval data, injecting false information, or manipulating external knowledge sources to mislead users or evade detection. As discussed in § 5.4, addressing these challenges is an important direction for our future research. In particular, we plan to explore mitigation strategies such as strengthening detection mechanisms, securing retrieval sources, and enhancing the trustworthiness of LLM reasoning processes to improve the overall security of LLM applications.