Enhancing Android Malware Detection: The Influence of ChatGPT on Decision-centric Task

YAO LI, Macau University of Science and Technology, China
SEN FANG, North Carolina State University, USA
TAO ZHANG*, Macau University of Science and Technology, China
HAIPENG CAI, University at Buffalo, USA

With the rise of large language models, such as ChatGPT, non-decisional models have been applied to various tasks. Moreover, ChatGPT has drawn attention to the traditional decision-centric task of Android malware detection. Despite effective detection methods proposed by scholars, they face low interpretability issues. Specifically, while these methods excel in classifying applications as benign or malicious and can detect malicious behavior, they often fail to provide detailed explanations for the decisions they make. This challenge raises concerns about the reliability of existing detection schemes and questions their true ability to understand complex data. In this study, we investigate the influence of the non-decisional model, ChatGPT, on the traditional decision-centric task of Android malware detection. We choose three state-of-the-art solutions, *Drebin*, XM_{AL}, and *MaMaDroid*, conduct a series of experiments on publicly available datasets, and carry out a comprehensive comparison and analysis. Our findings indicate that these decision-driven solutions primarily rely on statistical patterns within datasets to make decisions, rather than genuinely understanding the underlying data. In contrast, ChatGPT, as a non-decisional model, excels in providing comprehensive analysis reports, substantially enhancing interpretability. Furthermore, we conduct surveys among experienced developers. The result highlights developers' preference for ChatGPT, as it offers in-depth insights and enhances efficiency and understanding of challenges. Meanwhile, these studies and analyses offer profound insights, presenting developers with a novel perspective on Android malware detection—enhancing the reliability of detection results from a non-decisional perspective.

CCS Concepts: • **Computing methodologies** → **Machine learning**; • **Software and its engineering** → **Software notations and tools**.

Additional Key Words and Phrases: Android, Malware Detection, Interpretability, ChatGPT

ACM Reference Format:
Yao Li, Sen Fang, Tao Zhang*, and Haipeng Cai. 2018. Enhancing Android Malware Detection: The Influence of ChatGPT on Decision-centric Task. 1, 1 (February 2018), 30 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

The Android operating system is the world's most widely used mobile platform. As of the first quarter of 2024, Android continues to lead the smartphone market, capturing over 70% of the total market share [43]. However, this

[1]Tao Zhang* is the corresponding author.

Authors' addresses: Yao Li, liyao@must.edu.mo, Macau University of Science and Technology, Macao, China; Sen Fang, fangsen1996@gmail.com, North Carolina State University , North Carolina, USA; Tao Zhang*, tazhang@must.edu.mo, Macau University of Science and Technology, Macao, China; Haipeng Cai, University at Buffalo, Buffalo, USA, haipengc@buffalo.edu.

Manuscript submitted to ACM

widespread adoption has made Android a prime target for malicious attacks. Alarmingly, in 2022, cybercriminals released nearly 135,000 new malware variants per day, equating to over 93 attack attempts per minute, according to G DATA CyberDefense [8]. Furthermore, Kaspersky blocked almost 33.8 million malware, adware, and riskware attacks in 2023 [12]. Given Android's popularity, the operating system remains susceptible to a wide array of attacks, including credential theft, privacy breaches, bank fraud, ransomware, adware, SMS spoofing, and more. The Android malware threat poses a significant danger to users and continues to escalate as malware becomes increasingly sophisticated and virulent. For instance, ransomware like *LockerPin* can lock users out of their devices, while spyware such as *Pegasus* can steal sensitive data without detection. Additionally, banking trojans like EventBot can siphon financial information, further highlighting the growing severity of this issue. Consequently, the accurate classification of malware applications is paramount to ensuring system security and safeguarding user privacy [65].

Numerous approaches have been proposed to detect Android malware [10, 20, 26, 28, 32, 61], aiming to uncover new attack patterns, devise fresh signatures, or identify malicious code. For instance, Feng et al. [21] propose semantic-based Android malware detection through static analysis. Cai et al. [30] propose a series of weighted formulas to improve the detection effect of models such as support vector machine (SVM). Kim et al. [29] propose a multi-modal deep learning (DL) model to detect Android malware. Meanwhile, in Android malware detection, beyond basic identification, there is an emphasis on categorizing malware into specific classes for improved recognition [11, 15, 60]. For instance, Xu et al. [62] utilize deep learning to autonomously segment Android malware into distinct categories. Similarly, Vij et al. [56] introduce a method hinged on a graph signature-based classification approach. However, whether these methods are based on traditional machine learning or advanced deep learning techniques, whether their goal is detection or classification, they all face low interpretability issues. While some solutions, such as Drebin [4], claim to offer explainable malware detection, they often fall short in providing the level of detailed analysis and explanations that can fully inform developers about the basis of their decision-making. This has raised concerns about the reliability of existing detection schemes. Especially when these existing solutions are deployed in real-world data or when new datasets are introduced, the results for detecting new unknown malicious applications tend to be subpar. This decline in detection performance further exacerbates our concerns about their decision-making reliability. Moreover, it also leads developers to doubt the true capability of these solutions in understanding complex Android applications.

Simultaneously, with the birth and evolution of large language models like ChatGPT, their excellent analysis and understanding capabilities are gradually exploited. This powerful ability to analyze and explain has led to ChatGPT being employed across various tasks [46]. With appropriate prompts, ChatGPT can excel in diverse tasks, meticulously analyzing data and providing detailed explanations. The limitations of current solutions, combined with ChatGPT's analytical and capabilities, have led us to reconsider the following question: **Under the influence of ChatGPT, is there a novel perspective that detects Android malware from a non-decisional viewpoint, emphasizing not just the act of deciding, but also explaining the reason behind those decisions?**

To answer this question, we undertake the first comprehensive empirical study, delving into the limitations of decision-based models in Android malware detection tasks and the influence of ChatGPT on existing Android malware detection tasks. One of the key reasons for choosing ChatGPT over other large language models (LLMs) is that it requires no additional training. This significantly lowers the technical barriers and reduces the time and resources typically needed to fine-tune or retrain models. Unlike some other LLMs that may require customization or fine-tuning for specific tasks, ChatGPT is a ready-to-use model, providing high-quality results out of the box. Additionally, it eliminates the need for extensive computational resources, which are often required for training large models, making it an efficient and cost-effective solution. Meanwhile, we select three state-of-the-art detection solutions: *Drebin*, XM_{AL},

and *MaMaDroid*. Among them, *Drebin* and XM_{AL} are based on a machine learning model while *MaMaDroid* is built upon a deep learning model. We then craft appropriate prompts to guide ChatGPT in Android malware detection tasks. We input the same dataset (features extracted from both benign and malware) into *Drebin*, XM_{AL}, *MaMaDroid*, and *ChatGPT* respectively, retrieve their outputs, and subsequently conduct a comparative analysis. To structure our investigation, we formulate three research questions (RQ) to validate our observations. We provide a detailed exposition in the subsequent subsections.

1.1 Reliability Analysis

To gain a deeper understanding of Android malware detection and ChatGPT, we conduct an in-depth study of three state-of-the-art methods: *Drebin* and XM_{AL} based on machine learning, and *MaMaDroid* built upon deep learning. Our analysis is guided by the following research questions:

RQ 1:

To what extent can current models proficiently identify Android malware, and how reliable are they in terms of comprehending data when compared to ChatGPT?

We conduct a series of experiments employing *Drebin*, XM_{AL}, *MaMaDroid*, and ChatGPT on the same dataset. Our research findings indicate that existing solutions can effectively detect malicious software, but they are susceptible to dataset biases and lack interpretability. **In contrast, ChatGPT, while unable to provide specific decisions, can offer comprehensive and detailed analysis. It categorizes the input, analyzes each category, and ultimately provides a comprehensive analysis of potential risks and a maliciousness score.** For more information, please refer to Section 5.1.

1.2 Influence Analysis

Following ChatGPT's analysis of each sample, we obtain a series of detailed reports. We meticulously scrutinize both ChatGPT's reports and the outputs generated by existing models for each sample.

RQ 2:

How does the large language model (i.e., ChatGPT) impact existing Android malware detection solutions? And what valuable insights can it provide to motivate developers?

Through comprehensive analysis and extensive research, we identify that existing solutions suffer from usability issues, being not only challenging to use but also presenting difficulties in terms of re-deployment and reproducibility for programmers. Additionally, the low interpretability of current solutions is a pressing concern. In contrast, ChatGPT offers comprehensive and detailed analysis but falls short in decision-making capabilities. **This realization leads us to understand that the development of Android malware solutions should not be solely driven by decision-centric tasks but should emphasize the need for a more interpretable approach that provides explanations alongside decision-making.** For more details, please refer to Section 5.2.

1.3 Improvement Analysis

Dataset bias and low interpretability are prevalent concerns within existing Android malware solutions. It prompts questions about these models' capacity to genuinely comprehend the data. Thus, in this study, we strive to enhance and improve Android malware detection through a novel perspective.

RQ 3:

How to enhance the Android Malware Detection capability of existing large language models (i.e., ChatGPT)?

Through our research, we discover that existing Android malware detection solutions, while effective, still suffer from issues related to dataset bias and low interpretability. Meanwhile, although ChatGPT can provide detailed analysis and explanations, it cannot make final decisions. Based on these findings, we aim to provide developers with more insights from a non-decision perspective to assist them in making improvements in the future. **Therefore, we find two ways to improve: 1) Further enhancing the explanatory and analytical capabilities of ChatGPT; 2) Constructing a large model specifically for detecting Android software.** For the details, please refer to Section 5.3.

1.4 Contributions

In summary, this paper makes the following contributions:

- We conduct the **first comprehensive study** on the impact of ChatGPT in Android malware detection, identifying its strengths in interpretability and analysis compared to existing solutions.
- Through experiments and user/developer surveys, we demonstrate the strong preference for large language models like ChatGPT due to their interpretability and analytical power.
- We introduce a novel perspective for future Android malware detection: prioritizing **explanation alongside decision-making** to enhance usability and trust in large language models.

The remainder of this article is structured as follows. Section 2 introduces the motivation and the relevant information on ChatGPT and Android malware detection. The proposed approach is presented in Section 3. Section 4 describes the experimental setup and evaluation metrics. Experimental results are shown in Section 5. The further analysis of the proposed approach is discussed in Section 6. Section 7 presents the related work. Finally, Section 8 provides a summary of the proposed approach in this paper.

Data Avalibility: The data underlying this paper is available on Figshare [1].

2 MOTIVATION AND BACKGROUND

2.1 Motivation

The development of Android malware detection has progressed significantly over the years, giving rise to numerous effective detection strategies. In theory, these strategies have demonstrated remarkable efficacy. However, they still suffer from dataset bias, particularly selection bias and label bias, which stem from imbalanced and inconsistent datasets used during model training. To investigate this, we select six representative strategies for our study and conduct experiments to evaluate their performance. **Therefore, we collected 67 instances of new malware introduced between July 1, 2023, and August 31, 2023, from the MalwareBazaar[2] website. This platform uploads newly**

[1] https://doi.org/10.6084/m9.figshare.27004879.v1
[2] https://bazaar.abuse.ch/

discovered malware daily, providing direct download access to researchers and analysts. The results of their detection are presented in Table 1.

Table 1. Performance of schemes against Zero-day Malware.

Method	EFIMDetector	Drebin	MaMaDroid	EC2	SEDMDroid	MMN
Number	22	9	33	26	22	26
Detection Rate	0.343	0.143	0.497	0.401	0.339	0.389

From Table 1, it can be observed that the six schemes perform poorly in detecting unknown malicious applications. While these methods have demonstrated good results in training and testing datasets, a significant decline can be observed when detecting unfamiliar malware samples. This phenomenon raises our concerns regarding the reliability of the detection results. Meanwhile, these detection solutions have not provided detailed analysis and explanations. This means that we do not know the basis of their decision-making. Moreover, this raises our concerns about the reliability of these detection schemes. Consequently, we undertake a more in-depth exploration. As illustrated in Figure 1, we display the detection report of unknown application on the MalwareBazaar[3] website. This site uploaded the application to eight popular antivirus tools. Among them, two tools deem the software harmless, but three label this unknown software as malicious, two consider it suspicious, and one does not provide any result. This indicates that even these advanced antivirus tools are unable to make a unanimous decision regarding an unknown malicious application. All of these tools provide their detection results, with some even offering detailed reports. However, these reports suffer from poor readability and require a wealth of specialized knowledge to comprehend. Moreover, these reports do not provide the rationale behind their decisions, nor do they offer explanations or analyses. Therefore, this demonstrates that both advanced detection schemes and popular antivirus tools have neglected the importance of explanation and analysis. This situation motivates us to conduct in-depth research to improve interpretability, thereby enhancing Android malware detection.

Meanwhile, we conducted a survey involving 101 practitioners to gather insights into their perspectives on current Android malware detection methods. Each participant was asked six questions, as listed in Table 2. The survey was designed in a semi-close-ended format, allowing participants to provide additional written feedback to supplement their answers.

The participants held diverse roles, including software engineers, developers, managers, and other professionals. Among them, 13 individuals were directly involved in the development of antivirus software, offering specialized insights into the field. The remaining participants worked in related areas such as mobile software development, operations, and maintenance. Table 3 provides a detailed breakdown of the participants' professions and years of experience.

To select participants, we employed the snowball sampling approach [25]. Initially, we reached out to 19 senior professionals from our personal contacts, each with over 10 years of industry experience. After recording their feedback, we asked them to refer other suitable candidates for the study. This process led to the inclusion of an additional 82 participants, encompassing a broad spectrum of roles, from developers to managers. This approach ensured a diverse and experienced pool of respondents for the survey. The following is a detailed analysis of the questionnaire survey results.

[3] https://bazaar.abuse.ch/sample/44d0a81bdd9bbd3892f0ec49c430d60717be35e15a1a9ff3c2b4fddae58ad45c/

Q1 In response to the first question, "Do you use antivirus software?", all respondents reported utilizing antivirus software to protect their devices. This widespread adoption highlights the critical role of antivirus solutions as an essential tool for ensuring security, both for individuals and businesses. Whether for personal or professional use, antivirus software enjoys considerable popularity, with individuals generally perceiving it as a crucial measure for ensuring system security. This demonstrates that respondents, regardless of their technical expertise, exhibit a notable degree of protection awareness.

Q2 Second question, we ask the participants about their reaction/experience of using antivirus software. The vast majority of respondents (91%) express satisfaction with the performance of their antivirus software, indicating that the majority of users believe it can effectively protect their systems. However, a notable minority (5%) report negative experiences, suggesting some level of dissatisfaction or a neutral attitude. Additionally, 4% of respondents indicate that their experience with this software is not positive and identify potential areas for improvement, particularly in functional performance and user experience.

Q3 The third question posed to the interviewees is: "How well do you understand the detection reports generated by antivirus software?" With regard to the comprehension of the detection reports, the distribution of respondents' answers is relatively dispersed. Approximately 34.7% of respondents indicate that they could fully comprehend the report content, suggesting that these individuals possess a robust technical background or familiarity with the report's subject matter. However, 11.9% of respondents indicate that their understanding of the report is limited, suggesting that the technical terminology or structure of the report may present a challenge. Of greater concern is that more than half of the respondents (53.4%) do not understand or are unclear about the report content, indicating that antivirus software reports may be too complex for ordinary users and lack simple and clear explanations.

Q4 The survey results of the fourth question indicate that 95% of respondents had utilized ChatGPT, which is indicative of the pervasive adoption of this AI tool across technical and non-technical domains. This finding underscores the recognition of ChatGPT as a pivotal problem-solving instrument and its significant role in daily work activities. Conversely, only 2.9% of respondents report no usage of ChatGPT, which could be attributed to their professional requirements or a lack of familiarity with such tools.

Q5 The fifth question is about the performance of ChatGPT. To ensure the interpretability of ChatGPT in SE real-world situations, we need to assess its ability to- i) interpret specific scenarios and contexts accurately; ii) generate appropriate information based on that. Hence, we provide ChatGPT with a specific question-answer set (about `Glide`) from Stack Overflow and ask it to explain how the library is used solely based on the provided information. In response to the question of whether ChatGPT could provide a more detailed explanation of the problem, 97% of respondents indicate that they find the explanation provided by ChatGPT to be sufficiently detailed and clear. This suggests that this cohort of users is content with their capacity to resolve issues. Nevertheless, 3% of respondents indicate a preference for more comprehensive explanations from ChatGPT, suggesting a need for more detailed information.

Q6 The last question posed to the interviewees is: "What can be the ways to enhance existing forms of malware detection?" In regard to the enhancement of malware detection, 69.3% of respondents indicate that the introduction of AI and machine learning technology would facilitate the improvement of detection capabilities. This finding suggests a high level of expectation for the integration of automation and intelligent technology in this domain. A 9.9% proportion of respondents express the desire to improve both the detection function and the software's accuracy in identifying threats, indicating that existing functions could benefit from enhancement. Furthermore, 3% of respondents underscore the necessity for simplifying the user experience, proposing that a more intuitive interface and operational methodology

Table 2. Survey questions.

Q#	Questions
1	Do you use antivirus software?
2	How would you describe your experience with using it so far?
3	How much do you understand the antivirus software's detection reports?
4	Do you use ChatGPT?
5	Do you think ChatGPT can explain in more detail when using ChatGPT to solve problems?
6	What can be the ways to enhance existing forms of malware detection?

Table 3. Demography of Survey Participants

	Years of Experience					
Current Profession	0-5	6-10	11-15	16-20	21-24	Total
Software Engineer (Dev/QA/Data/Ops)	40	17	11	8	-	76
Manager (Dev/PM/Ops)	3	1	6	2	-	12
Executive (Company Leadership)	0	-	-	1	1	2
Non-Tech (Sales/Marketing/CS/HR)	7	3	1	-	-	11
Total	50	21	18	11	1	101

would facilitate the software's usability. The remaining 17.8% of respondents provide responses that are unclear, indicating that they may lack specific suggestions on how to improve malware detection or may be uncertain about their specific needs.

By analyzing the above-mentioned investigation results, we can get the following conclusion. While existing malware detection schemes are effective, there is still considerable room for improvement in understanding reports and further improving software functionality. In particular, there is a need to simplify the user experience and increase the clarity of reports. Therefore, the survey results motivate us to strengthen the existing detection form.

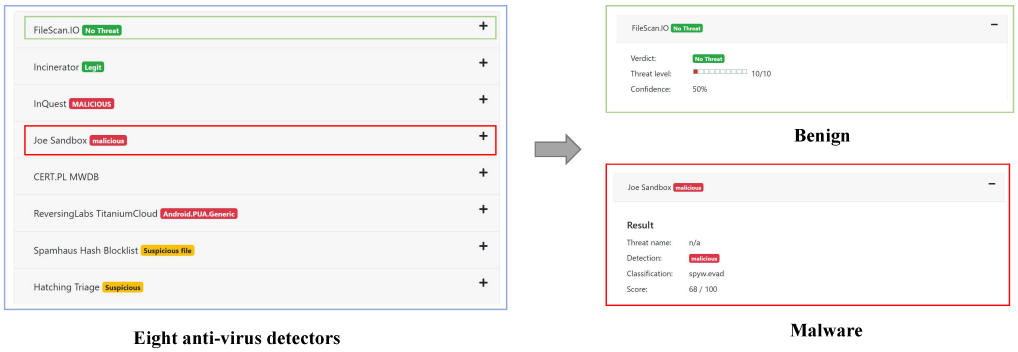


Fig. 1. SHA256 for "feb1b1019a31d0677c1d25bb80549a171c5f7da381f38014aae0d63b56126722", an unidentified Android software reported on MalwareBazaar.

2.2 ChatGPT

ChatGPT originates from OpenAI and is a descendant of the GPT model lineage. From GPT-1 to GPT-3 [44, 45, 47], each iteration has seen these models grow exponentially in complexity and computational capacity. ChatGPT adopts a transformer architecture, specifically tailored for sequential data processing, making it particularly suited for text. This structure aids the model in understanding context, thereby generating intricate and coherent text. The versatility of ChatGPT is evident in its wide range of applications. Whether it is content creation, tutoring, coding assistance, language translation, or casual conversation, this model can satisfy diverse conversational needs.

Despite ChatGPT's powerful capabilities, it is not without challenges. OpenAI prioritizes user safety and the ethical behavior of the model [46]. The fine-tuning phase is supported by human reviewers and guided by clear protocols, aiming to curb unsolicited outputs. OpenAI actively seeks feedback, using insights to refine and enhance model behavior.

In conclusion, ChatGPT has become a beacon in conversational AI, embodying a synthesis of extensive knowledge and vital adaptability to numerous text-generation tasks.

2.3 Android malware detection

In an era where smartphones are omnipresent and the Android operating system is extensively used, the convenience offered by mobile applications is being undermined by the continuous threats posed by the evolution of Android malware. Given the indispensable role that these devices play in our lives, they have also become the primary targets for malicious actors aiming to exploit vulnerabilities and compromise security. The surge in Android malware has stimulated the development of sophisticated detection technologies and tools. Android malware detection can be categorized into three methods: static analysis [14, 53], dynamic analysis [51, 57], and hybrid analysis [38, 42]. Static analysis involves decompiling an Android application package (APK) through reverse engineering to extract features from a series of files. Dynamic analysis, on the other hand, involves installing APKs on simulators or real devices to monitor system calls. Hybrid analysis employs both static and dynamic analysis methods. Furthermore, with the evolving landscape of Android malware, there is a need for detection solutions to be continuously enhanced and strengthened.

3 METHODOLOGY

3.1 Overview

To delve deeply into the challenges of limited data comprehension, reduced reliability, and interpretability in prevailing Android malware detection solutions, as well as to assess the potential influence of ChatGPT on these solutions, a methodological approach is essential. Our aim is to shed light on their capability to discern features effectively. In this research, as visualized in Figure 2, we introduce a verification method designed to rigorously assess the dependability of judgments rendered by current Android malware detection strategies.

Initially, we harness reverse engineering methods to distill static features from APK files. Concurrently, to glean dynamic features, the APK is installed and exercised on a genuine Android device.

Next, we craft targeted prompts to steer ChatGPT meticulously, honing its attention on Android malware detection, culminating in the formulation of a conclusive prompt.

The extracted data is then channeled into the prevailing Android malware detection systems and ChatGPT for evaluation.

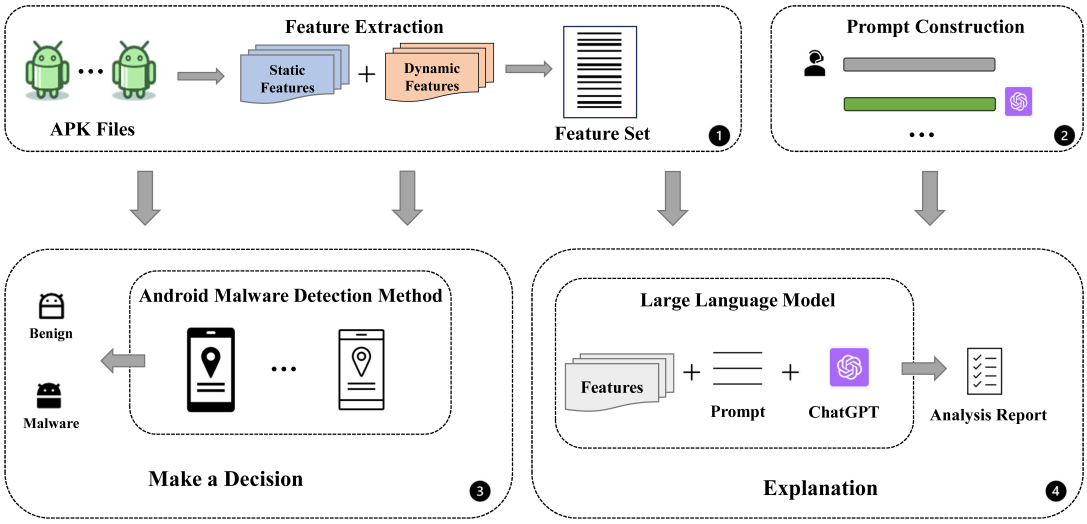


Fig. 2. Overview of our study.

In the final step, we collate in-depth analytical reports generated by ChatGPT, juxtaposed with individual detection outcomes produced by the conventional detection models.

3.2 Feature Extraction

The APK file, which functions as the installation package for Android applications, is analyzed using reverse engineering tools such as Apktool [54] and Androguard [13]. Apktool decompiles the APK into its core resources and bytecode, extracting critical components like the *AndroidManifest.xml* file and converting the *Class.dex* file (containing Dalvik Executable bytecode) into a human-readable *.smali* format. This enables detailed examination of API calls, strings, and control flows. In contrast, Androguard offers a robust framework for static analysis, supporting disassembly, decompilation, and in-depth bytecode inspection. It complements Apktool by facilitating advanced operations such as cryptographic signature analysis and comprehensive behavioral insights.

The extracted *AndroidManifest.xml* file is parsed to collect static features such as declared permissions, intent filters, and metadata, providing valuable information about the app's functionalities and interactions with the Android system. Additionally, the *Class.dex* file, once converted to *.smali* format, supports an in-depth exploration of the app's logic, helping uncover potential malicious behaviors embedded within its control flows or API usage patterns.

For dynamic feature extraction, applications are executed in a controlled emulator environment where runtime behaviors are monitored. This includes tracking system calls, interactions with the Android system, and other runtime characteristics.

When obfuscated APK files are encountered, unpacking tools [63, 67] are employed to recover the underlying files. Features extracted from these samples are cross-verified against dataset-provided features to ensure their accuracy.

3.3 Prompt Construction

A prompt acts as a director to initiate a machine learning model, particularly those designed for text generation. It is essentially a text fragment or directive that shapes the model's output towards a specific theme, style, or structure.

Unlike the conventions of supervised training, where every sample needs an associated label, using prompts in training only necessitates a guiding statement to steer the model's output generation. As such, the prompt's formulation becomes paramount. Inappropriate or vague prompts could lead the model astray, yielding incorrect or irrelevant results. We have delineated a three-step process to craft an effective prompt:

(1) **Task Definition:** Starting by succinctly expressing the specific output the prompt aims to elicit from the model.
(2) **Role Assignment:** Ascertaining and detailing the role the model should embody during text generation. Setting a clear context or perspective for the model often refines the resultant output, aligning it closely with the intended purpose.
(3) **Guidance:** Offering comprehensive directions for the model's text composition. This can encompass specifying the nature of the content (e.g., Answers should reflect advanced mathematical understanding), the structure of the response (this includes aspects like length, format, and tone), and any recommended approaches or strategies for formulation.

Such a structured approach ensures that the model not only produces relevant content but also maintains a high caliber of output quality. Therefore, we follow the above three principles and build a suitable prompt.

> You are an experienced Android developer and Android malware detector. I will give you the permissions, APIs, etc. used in the APK. Can you analyze these contents and make a judgment as to whether there are problems with this APK, based on these characteristics give the software a malicious score (range 0-1).

4 EXPERIMENTAL SETUP

To comprehensively assess the reliability of current models for data comprehension, we conducted a series of experiments using publicly accessible datasets. Specifically, we utilized the Kronodroid dataset [27], which includes data collected from both emulator environments and actual devices. The dataset comprises 28,745 malicious samples across 209 distinct malware families and 35,256 benign samples. For our analysis, we focus on extracting 289 dynamic features (e.g., system calls) and 200 static features (e.g., permissions, intent filters, and metadata). In our research, we randomly selected 100 benign samples and 100 malicious samples from the Kronodroid dataset, enabling rapid experimentation and analysis. Additionally, we constructed $dataset_T$ by randomly selecting 3,000 benign samples and 3,000 malicious samples from the Drebin [4] and AMD datasets [58]. Unlike prior studies that emphasize malware family classification, our research exclusively focuses on the binary classification of benign versus malicious applications, disregarding family information throughout dataset construction, model training, and detection processes. For obfuscated samples, we employed unpacking systems [63, 67] to recover original files, verified the extracted features against dataset-provided features, and ensured the accuracy of the analysis.

We construct two distinct datasets to support our analysis. The $dataset_A$, containing 200 samples (100 benign and 100 malicious), is primarily used to analyze ChatGPT's performance and conduct comparative experiments. We adopt this size based on the dataset configurations commonly used in related research [1, 3]. Its main purpose is to explore ChatGPT's ability to analyze malware rather than optimizing detection performance, and its size also helps minimize computational overhead, fitting within GPT-4's usage constraints. The $dataset_D$, with 6,000 samples (3,000 benign and 3,000 malicious), is used for training traditional detection models, reflecting typical configurations in related research [32]. The $dataset_D$ allows for better model training and generalization by covering a wider range of software behaviors. By using both datasets, we leverage the strengths of each: the smaller one enables quick analysis, while the larger one

supports more comprehensive training and validation. The $dataset_A$ is utilized for the analysis and testing of ChatGPT, while $dataset_D$ is employed for the training and testing of traditional models. For a comprehensive overview of the experiments and results, please refer to Section 5.1.

The subsequent experiments are conducted on a personal computer integrated with 13th Gen Intel(R) Core(TM) i9-13900K 3.00 GHz and NVIDIA GeForce RTX 4090. The computer has 32GB of memory and 2TB of storage. The deep neural networks are implemented using Scikit-learn, and Pytorch. Meanwhile, the ChatGPT 4 model is employed for analysis in subsequent experiments.

4.1 Evaluation Metrics

There are four common metrics adopted to evaluate the performance of classification: accuracy, precision, recall rate, and F-score. The malicious sample is denoted as the positive (P) class and the benign sample is denoted as the negative (N) class. Then, four numbers are obtained:

- True Positive (TP): the number of positive samples that are correctly predicted as positive.
- False Negative (FN): the number of positive samples that are incorrectly predicted as negative.
- False Positive (FP): the number of negative samples that are incorrectly predicted as positive.
- True Negative (TN): the number of negative samples that are correctly predicted as negative.

Based on these designations, the following equations calculate four common metrics:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FN} + \text{FP} + \text{TN}} \tag{1}$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{2}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{3}$$

$$F\text{-Score} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{4}$$

5 EMPIRICAL STUDY

In this study, we explore the influence of ChatGPT on decision-making scenarios such as Android malware. Like the recent series of conversational large language models, especially those represented by ChatGPT, they have a higher degree of understanding of data and tasks. By comparing ChatGPT with the traditional decision-making Android malware detection model, we discover the influence of ChatGPT on the traditional model, and how to further improve the traditional model and construct a large language model more suitable for Android malware detection. Therefore, we design three research questions to conduct a comprehensive analysis and research, which are presented in the following subsections 5.1-5.3.

5.1 Reliability of data understanding

RQ 1:

To what extent can current models proficiently identify Android malware, and how reliable are they in terms of comprehending data when compared to ChatGPT?

In this section, we begin by verifying the effectiveness of existing detection solutions and identifying their limitations (Section 5.1.1). We conduct this experiment using two datasets: $dataset_T$ and $dataset_A$. The $dataset_T$ is divided into a training set and a test set, with the training set used to train the models and the test set used to evaluate their performance. Subsequently, we apply the trained models to $dataset_A$ to assess their performance on previously unseen data, providing insights into the generalizability of the detection schemes. Next, we assess the analysis and explanation capabilities of ChatGPT (Section 5.1.2). Finally, we compare the performance of traditional models with that of ChatGPT (Section 5.1.3).

5.1.1 ***Performance of Existing Methods.*** The Android application is processed to extract both static and dynamic features, which are then provided as input to ChatGPT and used as training data for the model. **The Android application is not imported in its entirety for two principal reasons.** First, in order to maintain consistency in our experiments, it is necessary to align this solution with existing approaches that do not support the processing of the entire Android application as input. These methods typically concentrate on the analysis of particular features as opposed to the processing of complete binaries. This guarantees that the results obtained remain comparable across methods. Secondly, although ChatGPT is technically capable of reading binary files, such as those found in Android applications, it lacks the inherent ability to effectively interpret their complex structure and semantics. Binary files contain low-level machine code and compressed data that require specialized software to properly analyze. Direct input into a language model such as ChatGPT would not yield meaningful insights, as the model is optimized for processing human-readable text rather than machine-level instructions. Accordingly, we elected to extract pertinent features that ChatGPT is better equipped to process with greater efficiency, thereby enhancing the reliability and interpretability of the ensuing analysis.

In our experiments, we choose three state-of-the-art solutions: *Drebin* [4], XM_{AL} [59], and *MaMaDroid* [39]. We select models that use both traditional machine learning and deep learning techniques to identify malicious applications. Additionally, these models use feature sets that contain the same categories as the feature sets we extract.

We design a series of experiments to measure the efficacy of these three solutions in accurately identifying malicious software samples, as well as to compare them with ChatGPT. Below, we provide detailed insights into these methods.

- *Drebin* [4] is a lightweight method proposed to address the security threat posed by malicious applications on the Android platform. As the proliferation of such applications hampers conventional defense mechanisms, Android smartphones often lack adequate protection against emerging malware. To address this issue, *Drebin* performs an extensive static analysis by gathering a wide range of application features. These features are then embedded in a unified vector space, allowing for the identification of typical patterns indicative of malware.
- MaMaDroid [39] is a static analysis-based system designed to detect malware targeting the Android platform. *MaMaDroid* takes a behavioral approach to malware detection by focusing on the sequence of abstracted application programming interface (API) calls performed by an app. The system abstracts the API calls into their corresponding class, package, or family, and constructs a model using the sequences obtained from the call graph of an app, treating them as Markov chains. This approach enhances the resilience of the model to changes in the underlying APIs and results in a manageable feature set.
- XM_{AL} [59] is a novel and interpretable machine learning-based method for detecting malware with high precision, while also explaining the detection results. XM_{AL} uses multilayer perceptrons and attention mechanisms to identify the key features most relevant to the detection results. Then, XM_{AL} automatically generates natural language descriptions to explain the core malicious behaviors within the application.

Table 4. Results of Drebin, XM_{AL} and MaMaDroid on $dataset_T$.

Method	FN	FP	Accuracy	Precision	Recall	F-Score
Drebin	5	4	0.955	0.963	0.951	0.952
XM_{AL}	2	3	0.983	0.985	0.986	0.984
MaMaDroid	2	1	0.979	0.978	0.973	0.981

Table 5. Results of Drebin, XM_{AL} and MaMaDroid on $dataset_A$.

Method	FN	FP	Accuracy	Precision	Recall	F-Score
Drebin	9	3	0.914	0.903	0.911	0.912
XM_{AL}	9	4	0.965	0.959	0.949	0.954
MaMaDroid	6	11	0.945	0.939	0.940	0.941

We use $dataset_T$ to train these three schemes and test on $dataset_A$. The results are shown in Table 4 and Table 5.

As shown in Table 4, *Drebin*, XM_{AL}, and *MaMaDroid* all demonstrate impressive performance in $dataset_T$, with accuracy and F-Score exceeding 95%. Meanwhile, the error rates of these three solutions are relatively low. Their FN values are 3, 2, and 2, respectively. These results prove the effectiveness of these Android malware detection solutions in accurately identifying malicious applications.

However, when we switch to a new dataset, the performance of *Drebin*, XM_{AL}, and *MaMaDroid* decline, as shown in Table 5. While *Drebin*, XM_{AL}, and *MaMaDroid* maintain over 90% in all four metrics on $dataset_A$, their FN values significantly increase. The FN values for *Drebin*, XM_{AL}, and *MaMaDroid* increase from 3, 2, 2 to 9, 9, and 6 respectively. This indicates that these Android malware detection solutions suffer from dataset bias. Moreover, this suggests that these detection solutions have not truly understood the features we input; instead, they seem to primarily learn the unique statistical patterns inherent in datasets [34].

Finding 1: Existing detection solutions can effectively detect Android malware, but they suffer from dataset bias. Under a new dataset, the detection efficacy decreases, and the FN values significantly increase.

5.1.2 ***Performance of ChatGPT.*** We use ChatGPT to analyze all samples in $dataset_A$, utilizing our designed prompts to guide ChatGPT. Meanwhile, the prompt we used is introduced in Section 3.3 and illustrated in Figure 3.

We use the designed prompts to guide ChatGPT, defining the tasks we need it to perform and the desired output. Meanwhile, due to OpenAI's limitations, ChatGPT cannot make definitive decisions. Therefore, we cannot use ChatGPT to determine whether a given sample is malicious or benign. Instead, ChatGPT can provide a maliciousness score for each sample, ranging from 0 to 1, where higher values indicate a higher degree of maliciousness.

We input various features of a sample into ChatGPT, enabling it to analyze the sample and generate results as shown in Figures 4, 5, 6, and 7.

We randomly select a malicious sample and input the static and dynamic features extracted from this sample into ChatGPT, as illustrated in Figure 4. Then ChatGPT provides analysis.

First, ChatGPT analyzes the dynamic features employed by the sample, specifically system calls, as depicted in Figure 5. ChatGPT classifies these system calls into six categories and provides analysis for each category. These analyses indicate that the application uses the file system, abuses memory mapping, and performs networking operations. These insights inform developers in detail about the potential functionality of the application.

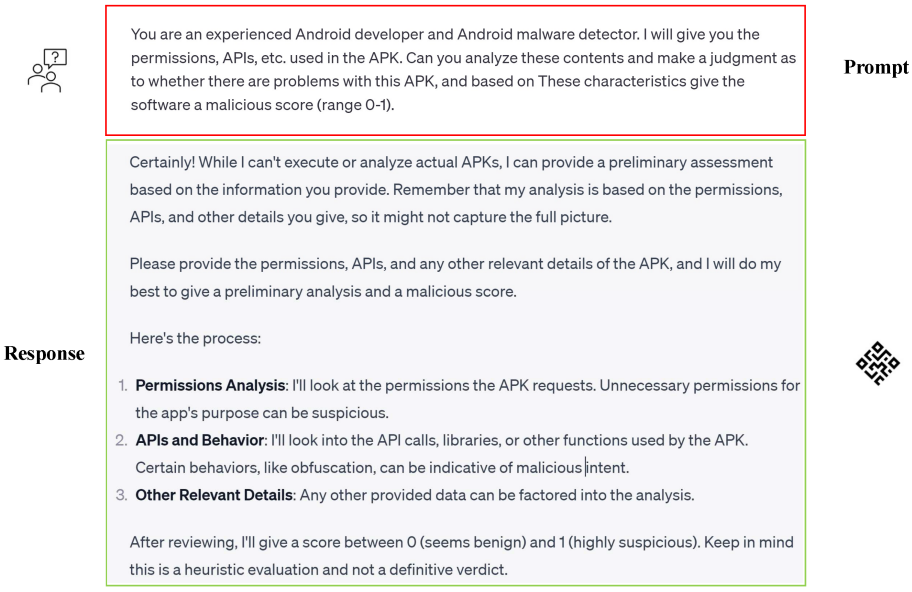


Fig. 3. The initial guidance and outcome presentation from ChatGPT.

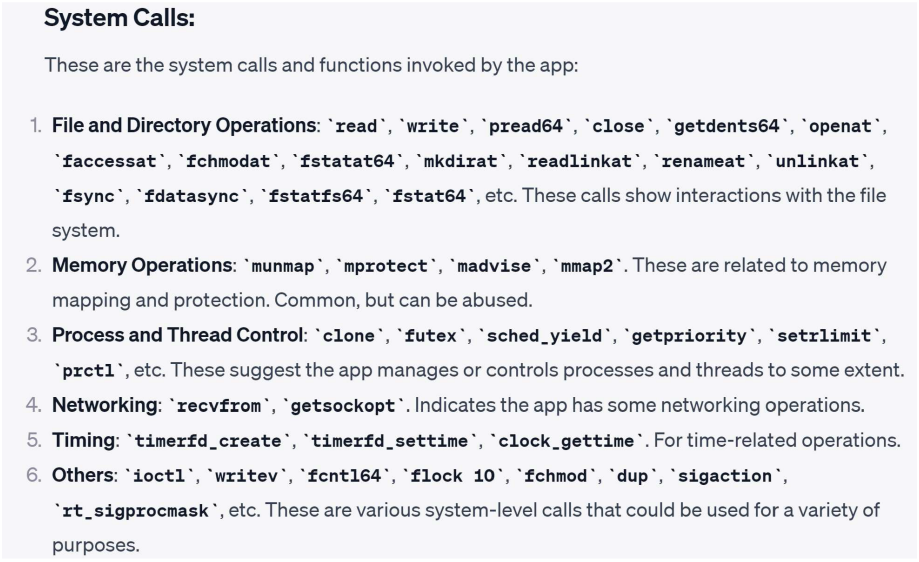
```
getuid32 getpriority ugetrlimit setrlimit prctl sigaltstack read write pread64 close munmap
mprotect madvise ioctl writev fcntl64 flock fchmod dup fsync fdatasync getdents64 openat
faccessat fchmodat fstatat64 mkdirat readlinkat renameat unlinkat lseek _llseek mmap2 fstatfs64
fstat64 timerfd_create timerfd_settime sigaction rt_sigprocmask recvfrom getsockopt
sched_yield sysinfo epoll_ctl epoll_pwait eventfd2 wait4 clock_gettime clone futex getrandom
ACCESS_COARSE_LOCATION ACCESS_FINE_LOCATION ACCESS_NETWORK_STATE
ACCESS_WIFI_STATE INTERNET READ_PHONE_STATE WAKE_LOCK
WRITE_EXTERNAL_STORAGE
```

Fig. 4. Input features, with each feature separated by a space.

Second, ChatGPT analyzes the permissions declared by the application. ChatGPT categorizes these permissions into five distinct categories and provides explanations for each category, as depicted in Figure 6. These results reveal that the application can access the device's location, retrieve network status, wake up the device, and access device identification, among other capabilities. This provides developers with a clear understanding of the permissions sought by the application.

Finally, ChatGPT summarizes the aforementioned feature analysis, underscores four potentially malicious behaviors, and assigns a degree of maliciousness to the application.

As illustrated in Figures 5, 6, and 7, ChatGPT thoroughly examines the features within each category, providing explanations regarding their functions and potential security threats. It is important to note that, owing to the constraints of OpenAI, ChatGPT cannot provide definitive decisions but can only offer an estimate of the potential level of risk [46]. In contrast to existing detection solutions, ChatGPT does not provide definitive decisions, which means we cannot directly determine whether a piece of software is malware. However, ChatGPT offers an extensive analysis, furnishing



System Calls:

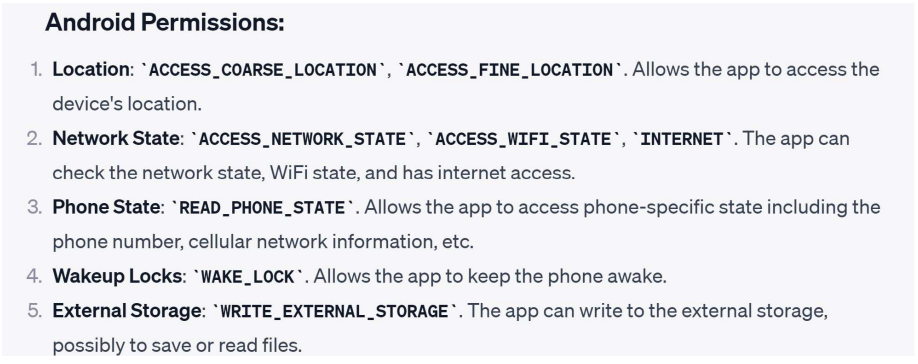
These are the system calls and functions invoked by the app:

1. **File and Directory Operations**: `read`, `write`, `pread64`, `close`, `getdents64`, `openat`, `faccessat`, `fchmodat`, `fstatat64`, `mkdirat`, `readlinkat`, `renameat`, `unlinkat`, `fsync`, `fdatasync`, `fstatfs64`, `fstat64`, etc. These calls show interactions with the file system.
2. **Memory Operations**: `munmap`, `mprotect`, `madvise`, `mmap2`. These are related to memory mapping and protection. Common, but can be abused.
3. **Process and Thread Control**: `clone`, `futex`, `sched_yield`, `getpriority`, `setrlimit`, `prctl`, etc. These suggest the app manages or controls processes and threads to some extent.
4. **Networking**: `recvfrom`, `getsockopt`. Indicates the app has some networking operations.
5. **Timing**: `timerfd_create`, `timerfd_settime`, `clock_gettime`. For time-related operations.
6. **Others**: `ioctl`, `writev`, `fcntl64`, `flock 1O`, `fchmod`, `dup`, `sigaction`, `rt_sigprocmask`, etc. These are various system-level calls that could be used for a variety of purposes.

Fig. 5. Analysis of system calls used by application.

explanations for each feature within its respective category and delving into potential threats associated with these features. These detailed analyses and explanations help enhance developers' understanding. They enable developers to gain a comprehensive grasp of the functionality utilized by the software and its potential implications.

Moreover, we compare the detailed reports provided by ChatGPT with the explanatory descriptions of XM_{AL} from *Drebin*. *Drebin* simply provides basic definitions for each feature. XM_{AL} deduces features that make decisions based on detection results and provide natural language descriptions. Both *Drebin* and XM_{AL}'s explanatory descriptions lack analysis and more detailed, in-depth descriptions. Conversely, while ChatGPT may not offer specific decisions, it conducts analysis and provides explanations based on features, deducing results from underlying causes rather than reverse-engineering causes from outcomes. This illustrates ChatGPT's genuine understanding of input features and its analytical approach.



Android Permissions:

1. **Location**: `ACCESS_COARSE_LOCATION`, `ACCESS_FINE_LOCATION`. Allows the app to access the device's location.
2. **Network State**: `ACCESS_NETWORK_STATE`, `ACCESS_WIFI_STATE`, `INTERNET`. The app can check the network state, WiFi state, and has internet access.
3. **Phone State**: `READ_PHONE_STATE`. Allows the app to access phone-specific state including the phone number, cellular network information, etc.
4. **Wakeup Locks**: `WAKE_LOCK`. Allows the app to keep the phone awake.
5. **External Storage**: `WRITE_EXTERNAL_STORAGE`. The app can write to the external storage, possibly to save or read files.

Fig. 6. Analysis of permissions used by application.

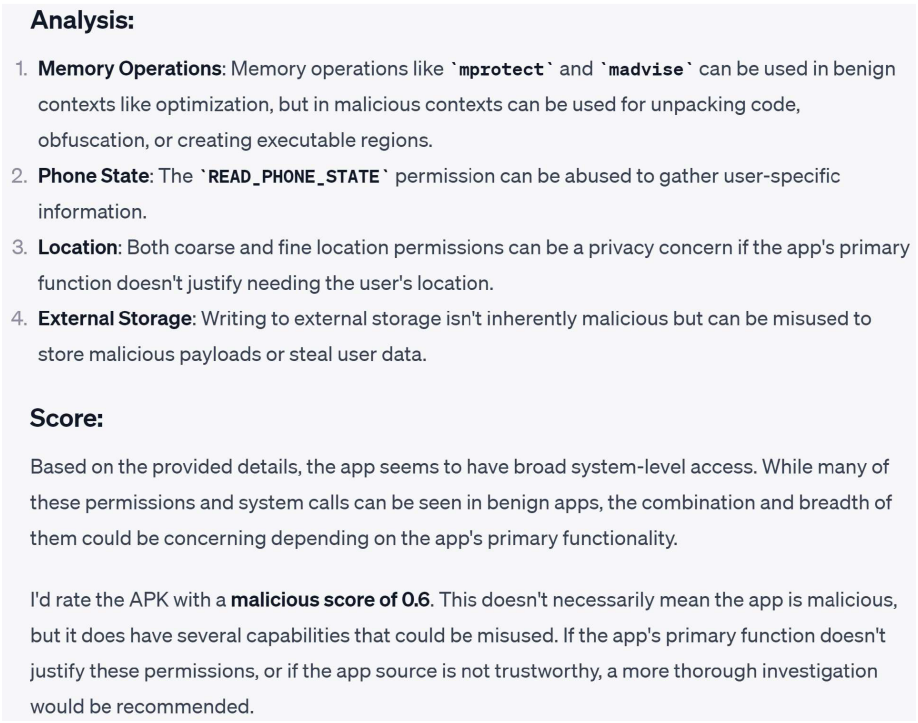


Fig. 7. Summary analysis and the assignment of maliciousness scores.

Finding 2: Existing detection solutions either provide only decisions or generate reports that are mere explanations of features. In contrast, ChatGPT not only offers detailed explanations but also provides in-depth analysis.

5.1.3 ***Human Evaluation***. In order to conduct a comprehensive comparison and analysis between ChatGPT and existing detection solutions, we invite ten experienced developers, each with more than 10 years of experience in Android, to participate in a questionnaire. All ten respondents are engaged in Android mobile software development, which aligns closely with the focus of our study. Their work involves various aspects of the Android ecosystem, including coding, testing, and maintaining mobile applications. This hands-on experience ensures they possess in-depth knowledge of Android development frameworks, tools, and best practices. The specific questions are as follows:

(1) **Android Device Usage:**
- Do you frequently utilize devices powered by the Android operating system, such as smartphones and tablets?

(2) **Interactions with Antivirus Software:**
- How often do you come across alerts or detections from antivirus software when using or installing applications on your Android device?
  - Rate your experience on a scale of 0 to 5:
  - 0 - Never encountered this situation;

  - 5 - Frequently encounter this situation.

(3) **Evaluation of Existing Detection Solutions:**
  - We provide three state-of-the-art Android malware detection solutions, each with 20 detection reports. Please select five reports from each of these three solutions. After reading these reports, please answer the following questions: Do you think the reports provided by the state-of-the-art Android malware detection solutions are detailed? Are you concerned about the reliability of the decision-making process of these solutions?

(4) **Understanding ChatGPT's Analysis:**
  - We provide 20 detection reports generated by ChatGPT. Please randomly select five reports from them. After reading them, please answer the following question: Do you think the reports provided by the ChatGPT are detailed? Are concerned that ChatGPT cannot provide specific decisions?

(5) **Comparative Report Preferences:**
  - After reviewing both sets of reports, which format do you find more appealing and why? To assist in our analysis, kindly rate the two types of reports based on:
    - Comfort: How easily do you feel when trying to understand the content?
    - Readability: How clear and understandable is the text?
    - Friendliness: How approachable and user-friendly is the content?
  - For each aspect, use a scale of 0 to 5, where:
    - 0 signifies extremely poor;
    - 5 denotes exceptional performance.

We consolidate feedback from all participants, which is illustrated in Figures 8, 9, and 10.

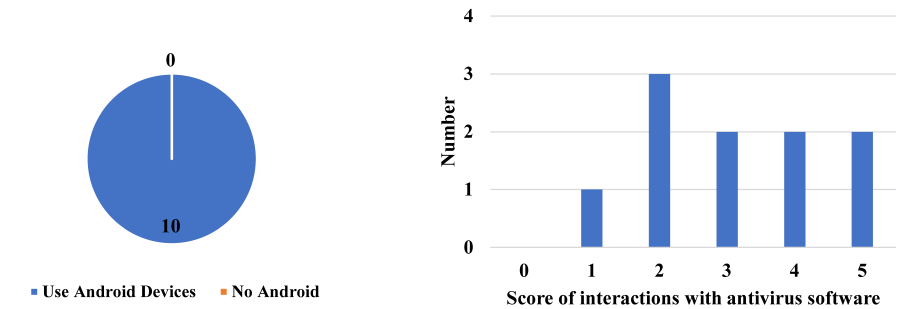


Fig. 8. Percentage of participants who use Android devices and the regularity of their interactions with antivirus software.

For the first question, ten experienced developers state that they often use Android-based devices, such as mobile phones and tablets. For the second question, to facilitate quantitative analysis, we ask respondents to rate it. As shown in Figure 8, four respondents scored below 3, while the remaining 6 respondents scored 3 or above. This suggests that, both in professional settings and in daily life, people frequently use antivirus software on Android devices and frequently encounter diagnostic reports generated by these software.

For the third question, the ten developers have varying opinions on the three detection solutions. First, all ten developers unanimously find the reports generated by *MaMaDroid* to be very concise, providing only detection results.

They also express significant concerns about how *MaMaDroid* makes its decisions. Second, 30% of the respondents believe that *Drebin*'s reports are relatively detailed but of limited utility. The remaining 70% of the respondents find *Drebin*'s reports not very detailed, merely providing explanations of some feature definitions with no practical value. Additionally, all ten respondents agree that *Drebin*'s reports do not reveal how it arrives at its judgments, and they still have concerns about the reliability of *Drebin*'s decision-making process. Third, the ten respondents believe that compared to *MaMaDroid* and *Drebin*, XM_{AL}'s reports contain more content and have clearer descriptions. However, they feel that XM_{AL} still does not provide analysis and explanations; it merely describes the functionality of the features being used. Based on XM_{AL}'s reports, they also cannot ascertain the basis for its decision-making process, leading to concerns about its reliability.

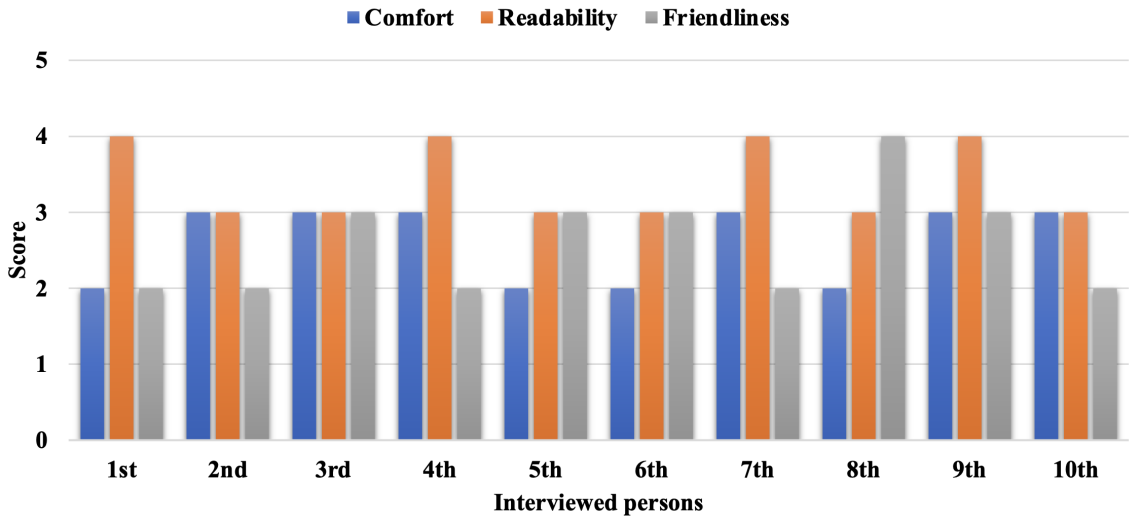


Fig. 9. Respondents' feedback on reports of existing detection solutions. The vertical axis displays the scores, ranging from 0 to 5, while the horizontal axis represents the respondents, numbered sequentially from the first to the tenth.

For the fourth question, we summarize the feedback from the ten respondents. First, ten developers agree that the reports provided by ChatGPT are detailed, including not only analysis and explanations of input features but also their potential implications and final summary analysis. However, ten developers also state that ChatGPT's inability to provide decisions and judgments is a major drawback. They mention that if ChatGPT needs further improvement, it should be able to make decisions.

In reference to the fifth question, every interviewee expresses a unanimous preference for ChatGPT due to its provision of clear explanations. Furthermore, we ask the interviewees to assess the decisions made by existing models and the explanations provided by ChatGPT, evaluating them on three key aspects—*Comfort, Readability, and Friendliness*, as illustrated in Figures 9 and 10. From these two figures, it is evident that all ten interviewees have given higher ratings to the analysis and explanations offered by ChatGPT. This reinforces the notion that ChatGPT's explanations are more intuitive and easier to comprehend.

The survey responses offer insightful perspectives on Android malware detection solutions and the preferences of experienced developers. Most respondents, who frequently use Android-based devices both professionally and

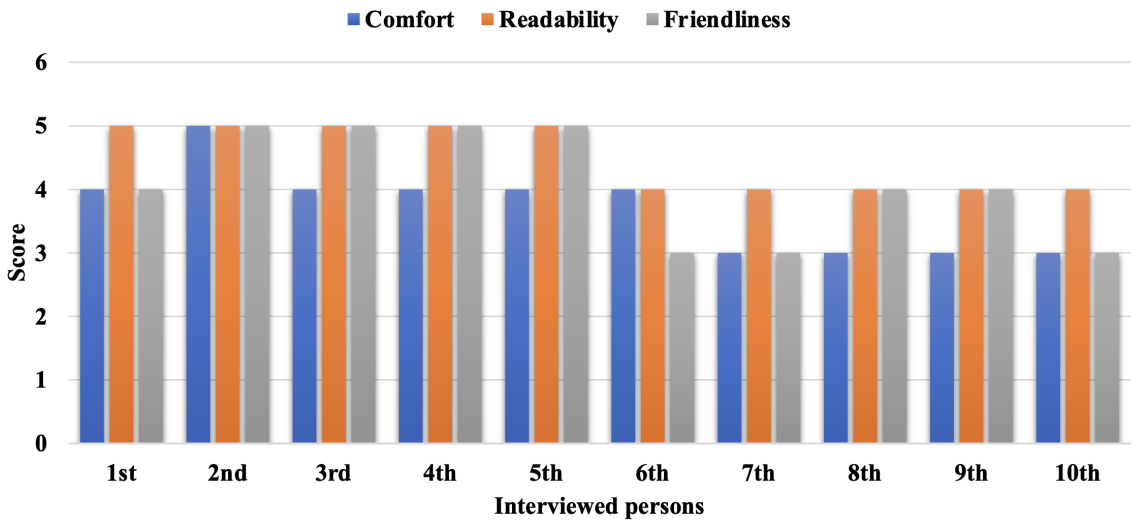


Fig. 10. Respondents' feedback on reports of ChatGPT. The vertical axis represents the score, while the horizontal axis represents the respondents. The vertical axis displays the scores, ranging from 0 to 5, while the horizontal axis represents the respondents, numbered sequentially from the first to the tenth.

personally, demonstrate a high reliance on antivirus software. This underscores the importance of effective malware detection tools in real-world applications.

In contrast, ChatGPT's reports were universally praised for their clarity, detail, and comprehensive analysis, making them more intuitive and user-friendly. However, respondents also noted its limitation in providing final decisions or judgments, which is a significant drawback for practical application in malware detection. Despite this, the developers overwhelmingly preferred ChatGPT for its readability, comfort, and overall user-friendliness, reinforcing the importance of clear and understandable explanations in building user trust. Overall, while existing Android malware detection models are widely used, their lack of transparency and interpretability remains a key challenge. ChatGPT's strength lies in its ability to offer detailed explanations, making it a valuable tool for developers, though further development is needed to enable it to provide definitive decisions.

Finding 3: Compared to existing detection solutions, the reports provided by ChatGPT are not only detailed but also highly readable. However, ChatGPT has a drawback in that it cannot make decisions.

5.2 Influence and improvement

RQ 2:

How does the large language model (i.e., ChatGPT) impact existing Android malware detection solutions? And what valuable insights can it provide to motivate developers?

In our previous discussions, we demonstrated the effectiveness of existing detection solutions in detecting malicious software through a series of experiments. However, we also discover that despite their strong performance and high detection rates, they are still susceptible to dataset biases. Additionally, these solutions suffer from low interpretability. The reports they provide are often simple explanations of features, and some solutions only provide detection results. Developers cannot discern the reasons and basis for the decisions made by these solutions from these reports and detection results, which has raised concerns about their decision-making reliability.

In contrast, compared to these decision-based detection solutions, ChatGPT, although unable to make explicit decisions, can provide a wealth of rich and detailed analysis and explanations. These analyses and explanations allow developers to gain deeper insights into the functionality used by the software and potential issues it may pose. Therefore, in order to improve and enhance detection solutions, we invite ten experienced developers to participate in a questionnaire. Six of the ten respondents are actively involved in antivirus software development, contributing their specialized expertise in identifying and mitigating security threats. Three respondents are engaged in artificial intelligence development, including predictive analytics, machine learning, and intelligent automation. The remaining respondent focuses on Android software development. Their expertise adds valuable context to the analysis, particularly in examining the practical implications of malware detection systems within the Android environment. We prepare four questions to gain a deeper understanding:

(1) Are you acquainted with the current leading Android malware detection tools? How would you rate their beginner-friendliness on a scale of 0 to 5?
(2) From your experience, which facets of the existing Android malware detection solutions could be enhanced? Please provide ratings on a scale for the following dimensions:
    - Interpretability: How clear and comprehensible are the results or outputs?
    - Ease of Use: How user-friendly and intuitive is the software or tool?
    - Convenience: How streamlined and efficient is the detection process?
    - For each aspect, use a scale of 0 to 5, where:
        - 0 denotes no improvement is needed;
        - 5 means improvement is urgently needed.
(3) What are the advantages and disadvantages of non-decision-based large language models like ChatGPT compared to traditional Android malware detection tools that rely on decision-making?
(4) Would you be open to adopting a specialized large language model designed for Android malware detection if it becomes available in the foreseeable future?

For the first question, everyone has exposure to or has used the current advanced Android malware detection solutions. We also ask them to rate the ease of using these solutions. 20% of the people give them 4 points, 50% of the people give them 3 points, and 30% of the people give them 2 points. This indicates that only 20% of respondents currently perceive existing Android malware detection solutions to be user-friendly, while the remaining 80% encounter a range of difficulties. The reported difficulties include the presence of complex interfaces and a paucity of user-friendly documentation, which collectively render these tools inaccessible to non-experts. Furthermore, a considerable proportion of respondents indicated that the replication or deployment of these solutions necessitates a considerable investment of time and effort, frequently due to intricate setup procedures, dependency on particular environments, or the necessity for extensive technical expertise. This indicates a clear necessity for the development of more intuitive and streamlined

tools, with the objective of reducing the barrier to entry for users and enhancing the overall efficiency of deployment and use.

Finding 4: The deployment and replication of existing Android malware detection solutions frequently presents significant challenges, primarily due to their intricate setup procedures and the necessity for a highly specialized technical infrastructure. These tools necessitate the navigation of numerous dependencies, the configuration of intricate parameters, and the resolution of compatibility issues between diverse devices or systems. Furthermore, the absence of transparent, user-friendly documentation exacerbates the complexity, rendering it challenging for users (particularly those lacking profound technical expertise) to effectively utilize these solutions.

For the second question, we ask respondents to evaluate whether existing solutions need improvement based on three aspects: interpretability, ease of use, and convenience. They were required to rate these three aspects on a scale from 0 to 5, where 0 represents no need for improvement and 5 represents an urgent need for improvement. Detailed results are shown in Figure 11. From Figure 11, we can observe that the ten developers are more concerned about the interpretability and convenience of the detection solutions. In terms of interpretability, two developers give a rating of five, indicating an urgent need for improvement. The rest of the developers give scores of no less than three. Regarding convenience, three developers give a rating of five, and the remaining developers also give scores of three or above. This suggests that developers prefer a detection tool that is highly interpretable and readily accessible.

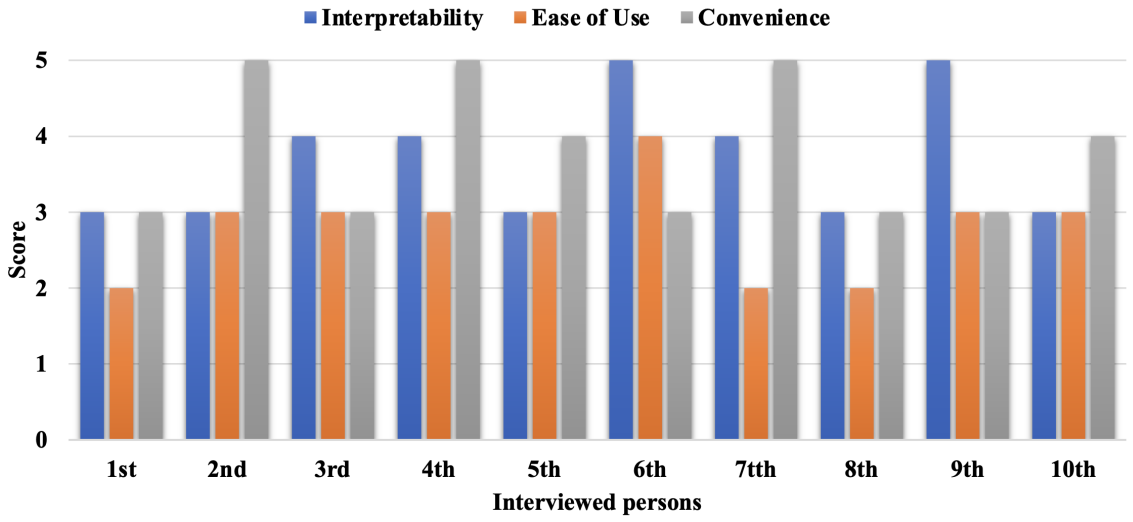


Fig. 11. Respondents' feedback on potential enhancements of existing Android malware detection solutions. The vertical axis displays the scores, ranging from 0 to 5, while the horizontal axis represents the respondents, numbered sequentially from the first to the tenth.

Finding 5: For existing Android malware detection solutions, interpretability and convenience need to be further improved.

For the third question, we summarize the feedback from the ten developers. First, ten developers agree that, compared to existing detection solutions, ChatGPT's primary advantage is its robust analytical and explanatory capabilities. ChatGPT can comprehensively analyze and explain input features. Second, ChatGPT's ease of use is another advantage. The conversational input and output format make it more convenient for developers to use. Meanwhile, all ten respondents also mention what they consider to be the main drawback, which is ChatGPT's inability to make absolute decisions. This is seen as a critical flaw for Android malware detection tasks that rely heavily on decision-making. Meanwhile, regarding the fourth question, ten respondents express their anticipation for large language models specifically designed for Android malware detection.

The survey results reveal significant insights into the current state of Android malware detection solutions and highlight key areas for improvement. While all respondents have experience with detection tools, the majority find them challenging to use, with only 20% rating them as user-friendly. The remaining 80% encounter difficulties such as complex interfaces, a lack of user-friendly documentation, and time-consuming setup processes, which create barriers to accessibility, especially for non-experts.

Regarding ChatGPT, all respondents acknowledged its strengths in analytical capabilities and ease of use, particularly its ability to provide clear, detailed explanations in a conversational format. However, the inability of ChatGPT to make definitive decisions was identified as a major drawback, as decision-making is crucial in malware detection tasks.

In conclusion, the survey highlights the need for more user-friendly and interpretable malware detection tools that not only simplify the user experience but also provide clear insights into decision-making processes. ChatGPT shows promise due to its analytical strengths and ease of use, but its inability to make concrete decisions indicates the need for further development to make it a more practical tool for malware detection. The future of Android malware detection lies in integrating robust explanatory features with actionable decision-making capabilities, ensuring both accessibility and reliability for developers.

Finding 6: While ChatGPT possesses strong analytical and explanatory capabilities, it falls short in delivering conclusive decisions.

5.3 Future development

RQ 3:

How to enhance the Android Malware Detection capability of existing large language models (i.e., ChatGPT)?

After the previous research and investigation, we find that although ChatGPT can provide detailed analysis and explanation, it is incapable of making the final decision, that is, determining whether an application is benign or malicious. In fact, we construct the prompt, trying to get ChatGPT to make a definitive decision or judgment, but it refuses each time. ChatGPT even resists giving a rating on the level of maliciousness. Through continuous optimization and testing, we determine the appropriate prompt that can elicit a maliciousness level rating from ChatGPT. This is precisely a limitation of ChatGPT. ChatGPT, constrained by OpenAI's requirements, is unable to make definitive

judgments [46]. But We want large models like ChatGPT to be capable of both making decisions and providing analysis and explanations. **Based on this, we hope to provide developers with more insights from a non-decision perspective to facilitate future improvements and the development of better Android malware solutions.** Therefore, our improvement plan is divided into two steps: 1) Improving ChatGPT; 2) Constructing a large language model specifically for Android malware detection.

First, to enhance ChatGPT, we can provide more data to allow it to analyze more deeply. For this, we can ask ChatGPT what additional data it needs to further enhance the analysis and interpretation of Android malware. Therefore, we can inquire with ChatGPT about what additional data it requires to further strengthen its analysis and interpretation of malicious Android applications. After inquiring, we learn that providing the hardware resources used by the APK, Intent, Activity, shell commands, and background information of the application can allow ChatGPT to better analyze Android malicious applications. Meanwhile, we can build corresponding retrieval enhancement generation (RAG) for ChatGPT. We use the open source RAG framework[4]. Then we collect Android permissions[5] and APIs[6] to form a vector database.

Figures 12, 13, and 14 show ChatGPT's responses after adding RAG. The example shown here is the same as that used in Section 5.1. At the same time, the same prompt is still used for booting. Then, we interview fifty participants to assess whether RAG enhances their experience with ChatGPT. The opinions of the fifty participants can be categorized into three groups. Thirty-six participants feel that while RAG offered some improvement, it is not substantial. Eleven participants believe that RAG provided more detailed explanations. The remaining three participants think there is no improvement at all. Limited by the design of the retrieval algorithm and the construction of the database, although the simple RAG has a certain improvement over ChatGPT, it does not have a better performance.

Although the analysis using RAG yields a greater number of results compared to the analysis without RAG, each feature is described in significantly more detail. However, the score for the same malicious application shows only a slight increase, from 0.6 to 0.7. This suggests that while ChatGPT's understanding of the characteristics leads to a more thorough analysis, the overall assessment score remains relatively stable. This demonstrates that ChatGPT, when enhanced with RAG, can provide more comprehensive insights into each feature, but the impact on the final scoring is modest.

Second, to construct a large language model focused on Android malware detection, we can use open-source large models for retraining. Through the first step, we will collect a large amount of feedback from ChatGPT, which can be used for model fine-tuning. Moreover, we need to collect a large number of benign and malicious samples, along with background and contextual information about these samples.

Finding 7: The enhancement goal is for large models to provide analysis and explanations while making decisions. Therefore, the improvement plan is divided into two steps: first, enhancing ChatGPT, and second, constructing a large model for Android malware detection.

[4]TinyRAG, https://github.com/phbst/tinyRAG/tree/main
[5]Adnroid permission, https://developer.android.com/guide/topics/permissions/overview?hl=en
[6]Android API, https://developer.android.google.cn/reference/packages

APIs and System Calls:

The APK uses several low-level system calls, which can be indicative of specific behaviors:

1. **getuid32, getpriority, ugetrlimit, setrlimit, prctl, sigaltstack**: These calls are often associated with managing process privileges and resource limits. They could be used legitimately but may also be employed in malicious activities to manipulate process behavior or bypass certain restrictions.
2. **read, write, pread64, close, munmap, mprotect, madvise, ioctl, writev, fcntl64, flock, fchmod**: These are common file I/O and memory management operations. While these are typical in many applications, they can be exploited to modify or access files and memory in unauthorized ways.
3. **dup, fsync, fdatasync, getdents64, openat, faccessat, fchmodat, fstatat64, mkdirat, readlinkat, renameat, unlinkat, lseek, _llseek, mmap2, fstatfs64, fstat64**: These are more file and directory-related operations. The presence of calls like `unlinkat` (which can delete files) and `mmap2` (which can map files or devices into memory) can be potentially risky if used improperly.
4. **timerfd_create, timerfd_settime, sigaction, rt_sigprocmask, recvfrom, getsockopt, sched_yield, sysinfo, epoll_ctl, epoll_pwait, eventfd2, wait4, clock_gettime, clone, futex**: These are often related to handling time, signals, synchronization, and system monitoring. The presence of `clone` (which is used to create processes) could be indicative of a program that spawns multiple processes, which could be suspicious if it's not expected behavior.
5. **getrandom**: This call is generally used to generate random numbers. In malicious software, it could be used to create unique identifiers or random data for obfuscation.

Fig. 12. Analysis of API and system calls used by the application.

6 DISCUSSION

6.1 Analysis

Our study of the impact of ChatGPT and LLM on Android malware detection challenges traditional decision-based detection models and proposes a shift towards more easily interpretable, analysis-driven approaches. **This shift is significant because traditional models, while effective, often lack transparency in the decision-making process, which can impede developers' ability to trust and enhance the detection process.**

The potential for enhancing the interpretability and analysis of malware detection is expanding with the advent of LLMs, such as ChatGPT. LLMs can furnish comprehensive explanations, furnish insight into the rationale behind decision-making processes, and assist in identifying patterns of malware behavior that may not be readily discernible through conventional methods. This enables not only the detection of malware but also the understanding of the reasoning behind each detection decision. This, in turn, paves the way for hybrid models where decisions are supported by deeper analysis and explanation.

Permissions:

- **ACCESS_COARSE_LOCATION, ACCESS_FINE_LOCATION**: These permissions allow the app to access the device's location. If the app has no legitimate reason to need this, it could be collecting location data for malicious purposes.
- **ACCESS_NETWORK_STATE, ACCESS_WIFI_STATE, INTERNET**: These are common permissions for apps that require internet access. However, combined with other permissions, they could be used to exfiltrate data.
- **READ_PHONE_STATE**: This permission allows the app to access phone information like the device ID and network information. This could be used to track the device uniquely.
- **WAKE_LOCK**: This permission allows the app to keep the device awake, which can be abused to perform tasks in the background.
- **WRITE_EXTERNAL_STORAGE**: This is a high-risk permission that allows the app to read and write to external storage. It could potentially be used to read/write sensitive data or introduce malicious files.

Fig. 13. Analysis of permission used by the application.

Malicious Score (0-1):

Given the above analysis, this APK exhibits several behaviors and permissions that could be considered risky, especially if not justified by the app's intended functionality. The combination of location access, phone state access, external storage writing, and low-level system calls, particularly those related to process control and file operations, are concerning.

Malicious Score: **0.7**

- **Explanation**: The score is relatively high due to the presence of permissions and system calls that could be abused for malicious purposes. However, without further context (e.g., the app's intended functionality, obfuscation techniques, network behavior), this score reflects a moderate to high potential for malicious activity.

Fig. 14. Summary analysis and the assignment of maliciousness scores.

LLMs have the potential to transform malware detection by increasing the visibility of the models' detection processes, thereby bridging the gap between decision-making and interpretability. **This transition from a decision-oriented to an explanation-oriented framework has the potential to markedly enhance confidence in detection models, facilitate more efficacious enhancements, and culminate in the development of more sophisticated malware detection systems.**

Our research indicates that LLM (e.g., ChatGPT) has made notable progress in the interpretability and accuracy of recognition systems. While traditional decision-based models are undoubtedly powerful, they are often constrained by a lack of transparency and an inability to elucidate the underlying decision-making processes. This study

demonstrates the advantages of integrating LLMs to enhance not only the detection process but also the transparency and credibility of these systems.

A principal finding of this research is the necessity for interpretability in the context of malware detection. The operation of traditional models as "black boxes" makes it challenging for developers and analysts to comprehend the underlying decision-making process. The integration of LLMs enables a more transparent and interpretable approach to malware detection, facilitating a deeper understanding of the underlying decision-making process and the rationale behind the identification of specific threats among users.

Further research should concentrate on enhancing the integration of LLM with conventional malware detection techniques. One avenue of promising research is the development of hybrid models. Furthermore, it is imperative to examine the efficacy of these models in actual operational contexts, particularly in regard to their scalability, computational demands, and practical deployment.

Furthermore, future research should examine how the more sophisticated analytical abilities of the LLM can be leveraged to minimize false positives and omissions. This may entail fine-tuning the model to gain a deeper understanding of the contextual factors that contribute to malware behavior, thereby enhancing detection accuracy and reducing errors.

In light of our findings, we propose the following recommendations for those engaged in the development or utilization of Android malware detection systems:

- It is recommended that LLM integration be employed to enhance interpretability. It would be beneficial for organizations to consider incorporating LLMs such as ChatGPT into their existing detection pipeline. This not only enhances the efficacy of detection but also improves the interpretability of results, thereby increasing the system's reliability and facilitating audit processes.
- It is recommended that a hybrid detection model be employed. In order to achieve an optimal balance between decision speed and interpretability, a hybrid approach should be employed. The model will initially employ a traditional decision-based approach, while the LLM will provide an additional layer of analysis and insight.
- User-centered design is a priority for future detection systems. These systems should be user-friendly and interpretable, facilitating comprehension of the reasons behind detected threats among both technical and non-technical users. This will enhance adoption and trust in these systems.

6.2 Why ChaGPT

We select ChatGPT as the LLM for the following reasons.

First, the goal of our research is not to compare or determine which LLM is superior in the field of Android malware detection and analysis. Instead, our focus is to demonstrate that LLMs have the potential to transform the existing detection paradigm. We aim to inform developers and users that malware detection methodologies in the LLM era should evolve. Rather than solely relying on traditional decision-oriented methods, greater emphasis should be placed on providing analysis and explanations to enhance understanding and usability.

Second, This research began in 2023. At the time, Claude 1.0 was primarily designed as a chatbot with limited functionality. When Claude 2.0 was released, its performance fell short of ChatGPT 4 [9]. It was not until June 2024, with the release of Claude 3.5 Sonnet, that its performance surpassed ChatGPT 4 [7]. However, Claude imposed strict usage restrictions, including requirements for billing and IP address consistency, as well as limitations on access in

[7] https://www.anthropic.com/news/claude-3-5-sonnet

certain regions. Since our primary authors are based in Greater China, these restrictions made it impractical to utilize Claude in our research. In contrast, ChatGPT's more lenient access policies allowed us to integrate it seamlessly into our workflow.

Finally, due to hardware constraints and the lack of comprehensive datasets (not only in terms of sample size but also missing contextual information such as application background data, etc.), training a large-scale open-source LLM was not feasible. ChatGPT offered a ready-to-use solution with robust capabilities, enabling us to conduct our research effectively. That said, we are actively collecting relevant datasets to support the future development of an open-source LLM specifically designed for Android malware detection.

These considerations made ChatGPT the most convenient and accessible option for our study. Looking ahead, we plan to build upon this work and develop a specialized LLM for Android malware detection using open-source models.

6.3 THREATS TO VALIDITY

Internal threats to validity. We construct suitable prompts to allow ChatGPT to provide analysis and explanations, as well as to give a rating on the level of maliciousness. However, due to the uncertainty of ChatGPT, when analyzing the same application multiple times, the explanations and ratings given are not entirely consistent. Therefore, it is impossible to determine which answer is more truthful and effective. In the future, we hope to eliminate this uncertainty by constructing a large model specifically for Android malware detection.

External threats to validity. First, our study explores the influence of a large model known as ChatGPT. Presently, numerous outstanding conversational large language models have yet to undergo similar investigations. Secondly, the dataset we employed offers both static and dynamic features; however, it has limitations in terms of scope, type, and the absence of contextual information, such as application usage. These limitations constrain ChatGPT's performance. In the future, we plan to gather more comprehensive datasets and investigate multiple large models.

7 RELATED WORK

Android malware detection: Android malware detection techniques can be categorized into two forms: static analysis and dynamic analysis. Static analysis involves decompiling the APK file to obtain the source code and metadata files, followed by extraction of significant features for malware detection and analysis such as the application programming interface [36, 53], permissions [2, 14, 30, 68], intent [52], function call graph [17, 24, 35], control flow [37], and grayscale [7]. Li et al. [33] propose a meta-learning-based multi-classifier that uses multiple features such as permissions and APIs to classify malware and achieves good results. Although the meta-classifier learns meta-knowledge in the dataset, it cannot provide a basis for making decisions. Dynamic analysis [6, 40, 49, 51, 57, 69] involves the extraction of features by monitoring the application's execution in real-time. Ficco [23] combines general and specialized detectors to enhance the randomness of detection and improve the overall detection rate. Although this solution combines static and dynamic analysis, it is still decision-oriented and does not consider the interpretability of the solution. With the development of artificial intelligence, many related researches focus on models, such as SVM [35, 48, 55], and decision tree (DT) [42], RF [30, 71], AdaBoost [7] and ensemble learning algorithm [19, 64], to classify malware. In addition, even deep learning methods [5, 70] and meta-learning methods [31, 72] are directly used for classification. Although these methods have shown superior capabilities in detecting malware, they lack the ability to explain and analyze, and are unable to analyze the features of each input and inform developers and users of the functions and potential threats represented by these features.

ChatGPT for SE: ChatGPT, with its powerful analysis and understanding capabilities, has been applied to various tasks in software engineering [16, 41, 66]. Feng et al. [22] utilize crowdsourced social data to study the code generation performance of ChatGPT. Sakib et al. [18] explore the efficacy of ChatGPT in solving programming problems, examining the correctness and efficiency of its solutions in terms of time and memory complexity. Sun et al. [50] conduct research to compare and evaluate the capabilities of ChatGPT in automatic code summarization. These studies have solely concentrated on ChatGPT's capacity for generation, without delving into its capabilities for analysis and interpretation.

8 CONCLUSION AND FUTURE WORK

ChatGPT has demonstrated exceptional interpretation and analysis capabilities, making it applicable to a variety of tasks. We are the first to study the influence of ChatGPT on Android malware detection tasks. We select state-of-the-art Android malware detection solutions for comparison with ChatGPT. The research indicates that existing detection solutions suffer not only from dataset bias but also lack explanation and analysis capabilities. Given the existence of these problems and the excellent interpretability of ChatGPT, we identify a novel perspective for detecting Android malware, that is, less decision-making, and more explanation. In the future, we aim to enhance and expand the existing datasets by addressing gaps and completing missing information, such as application background details. Additionally, we plan to develop a specialized large language model tailored for Android malware detection, with a focus on improving both detection accuracy and interpretability.

REFERENCES

[1] A. Abraham, R. Andriatsimandefitra, A. Brunelat, J.-F. Lalande, and V. Viet Triem Tong. 2015. GroddDroid: a gorilla for triggering malicious behaviors. In *2015 10th International Conference on Malicious and Unwanted Software (MALWARE)*. 119–127.

[2] Moutaz Alazab, Mamoun Alazab, Andrii Shalaginov, Abdelwadood Mesleh, and Albara Awajan. 2020. Intelligent mobile malware detection using permission requests and API calls. *Future Generation Computer Systems* 107 (2020), 509–521.

[3] Aisha I. Ali-Gombe, Brendan Saltaformaggio, J. "Ram" Ramanujam, Dongyan Xu, and Golden G. Richard. 2018. Toward a more dependable hybrid analysis of android malware using aspect-oriented programming. *Computers & Security* 73 (2018), 235–248.

[4] Daniel Arp, M. Spreitzenbarth, M Hübner, H. Gascon, and K. Rieck. 2014. DREBIN: Effective and Explainable Detection of Android Malware in Your Pocket. In *Network & Distributed System Security Symposium*. 1–15.

[5] Kathy Wain Yee Au, Yi Fan Zhou, Zhen Huang, and David Lie. 2012. PScout: Analyzing the Android Permission Specification. In *ACM Conference on Computer and Communications Security*. 217–228.

[6] Chanwoo Bae and Seungwon Shin. 2016. A collaborative approach on host and network level android malware detection. *Security and Communication Networks* 9 (2016), 5639–5650.

[7] K Bakour and H. M. Nver. 2021. VisDroid: Android malware classification based on local and global image features, bag of visual words and machine learning techniques. *Neural Computing and Applications* 33 (2021), 3133–3153.

[8] Kathrin Beckert-Plewka, Hauke Gierow, Vera Haake, and Stefan Karpenstein. 2020. *G DATA Mobile Malware Report: Harmful Android apps every eight seconds*. https://www.gdatasoftware.com/news/1970/01/-36401-g-data-mobile-malware-report-harmful-android-apps-every-eight-seconds

[9] Ali Borji and Mehrdad Mohammadian. 2023. Battle of the Wordsmiths: Comparing ChatGPT, GPT-4, Claude, and Bard. *SSRN Electronic Journal* (2023). https://api.semanticscholar.org/CorpusID:259621175

[10] Haipeng Cai. 2020. Assessing and Improving Malware Detection Sustainability through App Evolution Studies. *ACM Transactions on Software Engineering and Methodology* 29, 2 (2020).

[11] Tanmoy Chakraborty, Fabio Pierazzi, and V. S. Subrahmanian. 2020. EC2: Ensemble Clustering and Classification for Predicting Android Malware Families. *IEEE Transactions on Dependable and Secure Computing* 17, 2 (2020), 262–277.

[12] Victor Chebyshev. 2021. *Mobile malware evolution 2020*. Kaspersky. https://securelist.com/mobile-malware-report-2023/111964/

[13] Anthony Desnos. 2023. *Androguard, a full python tool to play with Android files*. https://github.com/androguard/androguard

[14] V. P. Dharmalingam and V. Palanisamy. 2020. A novel permission ranking system for android malware detection—the permission grader. *Journal of Ambient Intelligence and Humanized Computing* 12 (2020), 5071–5081.

[15] Karim O. Elish, Mahmoud O. Elish, and Hussain M. J. Almohri. 2022. Lightweight, Effective Detection and Characterization of Mobile Malware Families. *IEEE Trans. Comput.* 71, 11 (2022), 2982–2995.

[16] Mary E. Emenike and Bright U. Emenike. 2023. Was This Title Generated by ChatGPT? Considerations for Artificial Intelligence Text-Generation Software Programs for Chemists and Chemistry Educators. *Journal of Chemical Education* 100, 4 (2023), 1413–1418.

[17] M. Fan, J. Liu, X. Luo, K. Chen, Z. Tian, Q. Zheng, and T. Liu. 2018. Android Malware Familial Classification and Representative Sample Selection via Frequent Subgraph Analysis. *IEEE Transactions on Information Forensics and Security* 13 (2018), 1890–1905.

[18] A. H. M. Rezaul Karim Fardin Ahsan Sakib, Saadat Hasan Khan. 2023. Extending the Frontier of ChatGPT: Code Generation and Debugging. *ArXiv* (2023), 1–8.

[19] P. Feng, J. Ma, S. Cong, X. Xu, and Y. Ma. 2018. A Novel Dynamic Android Malware Detection System With Ensemble Learning. *IEEE Access* 6 (2018), 30996–31011.

[20] Ruitao Feng, Sen Chen, Xiaofei Xie, Lei Ma, Guozhu Meng, Yang Liu, and Shang-Wei Lin. 2019. MobiDroid: A Performance-Sensitive Malware Detection System on Mobile Platform. In *2019 24th International Conference on Engineering of Complex Computer Systems*. 61–70.

[21] Yu Feng, Saswat Anand, Isil Dillig, and Alex Aiken. 2014. Apposcopy: Semantics-Based Detection of Android Malware through Static Analysis. In *Proceedings of ACM SIGSOFT International Symposium on Foundations of Software Engineering*. 576–587.

[22] Yunhe Feng, Sreecharan Vanam, Manasa Cherukupally, Weijian Zheng, Meikang Qiu, and Haihua Chen. 2023. Investigating Code Generation Performance of ChatGPT with Crowdsourcing Social Data. In *2023 IEEE 47th Annual Computers, Software, and Applications Conference*. 876–885.

[23] Massimo Ficco. 2022. Malware Analysis by Combining Multiple Detectors and Observation Windows. *IEEE Trans. Comput.* 71, 6 (2022), 1276–1290.

[24] H. Gao, S. Cheng, and W. Zhang. 2021. GDroid: Android Malware Detection and Classification with Graph Convolutional Network. *Computers & Security* 20 (2021), 102264–102278.

[25] Leo A. Goodman. 1961. Snowball Sampling. *The Annals of Mathematical Statistics* 32, 1 (1961), 148 – 170.

[26] Michael Grace, Yajin Zhou, Qiang Zhang, Shihong Zou, and Xuxian Jiang. 2012. RiskRanker: Scalable and Accurate Zero-Day Android Malware Detection. In *Proceedings of the 10th International Conference on Mobile Systems, Applications, and Services*. 281–294.

[27] Alejandro Guerra-Manzanares, Hayretdin Bahsi, and Sven Nõmm. 2021. KronoDroid: Time-based Hybrid-featured Dataset for Effective Android Malware Detection and Characterization. *Computers & Security* 110 (2021), 102399.

[28] Taewon Jeong and Heeyoung Kim. 2020. OOD-MAML: Meta-Learning for Few-Shot out-of-Distribution Detection and Classification. In *Proceedings of the International Conference on Neural Information Processing Systems*.

[29] Taeguen Kim, Boojoong Kang, Mina Rho, Sakir Sezer, and Eul Gyu Im. 2019. A multimodal deep learning method for android malware detection using various features. *IEEE Transactions on Information Forensics and Security* 14 (2019), 773–788.

[30] Cai L., Li Yao, and Xiong Zhi. 2021. JOWMDroid: Android malware detection based on feature weighting with joint optimization of weight-mapping and classifier parameters. *Computers & Security* 100 (2021), 102086–102100.

[31] Wenhao Li, Huaifeng Bao, Xiao-Yu Zhang, and Lin Li. 2022. AMDetector: Detecting Large-Scale and Novel Android Malware Traffic with Meta-learning. In *Computational Science*, Derek Groen, Clélia de Mulatier, Maciej Paszynski, Valeria V. Krzhizhanovskaya, Jack J. Dongarra, and Peter M. A. Sloot (Eds.). 387–401.

[32] Yao Li, Zhi Xiong, Tao Zhang, Qinkun Zhang, Ming Fan, and Lei Xue. 2022. Ensemble Framework Combining Family Information for Android Malware Detection. *Comput. J.* (2022).

[33] Yao Li, Dawei Yuan, Tao Zhang, Haipeng Cai, David Lo, Cuiyun Gao, Xiapu Luo, and He Jiang. 2024. Meta-Learning for Multi-Family Android Malware Classification. *ACM Trans. Softw. Eng. Methodol.* 33, 7 (2024), 1–27.

[34] Yao Li, Tao Zhang, Xiapu Luo, Haipeng Cai, Sen Fang, and Dawei Yuan. 2023. Do Pre-trained Language Models Indeed Understand Software Engineering Tasks? *IEEE Transactions on Software Engineering* (2023), 1–18. https://doi.org/10.1109/TSE.2023.3308952

[35] P. Liu, W. Wang, X. Luo, H. Wang, and C. Liu. 2021. NSDroid: efficient multi-classification of android malware using neighborhood signature in local function call graphs. *International Journal of Information Security* 20 (2021), 59–71.

[36] Zhen Liu, Ruoyu Wang, Nathalie Japkowicz, Deyu Tang, Wenbin Zhang, and Jie Zhao. 2021. Research on unsupervised feature learning for Android malware detection based on Restricted Boltzmann Machines. *Future Generation Computer Systems* 120 (2021), 91–108.

[37] Z. Ma, H. Ge, Y. Liu, M. Zhao, and J. Ma. 2019. A Combination Method for Android Malware Detection Based on Control Flow Graphs and Machine Learning Algorithms. *IEEE Access* 7 (2019), 21235–21245.

[38] Samaneh Mahdavifar, Dima Alhadidi, and Ali. A. Ghorbani. 2022. Effective and Efficient Hybrid Android Malware Classification Using Pseudo-Label Stacked Auto-Encoder. *Journal of Network and Systems Management* 30 (2022), 22–56.

[39] E. Mariconti, L. Onwuzurike, P. Andriotis, E Cristofaro, G. Ross, and G. Stringhini. 2019. MaMaDroid: Detecting Android Malware by Building Markov Chains of Behavioral Models. *ACM Transactions on Privacy and Security* 22 (2019), 1–34.

[40] L. Massarelli, L. Aniello, C. Ciccotelli, L. Querzoni, and R. Baldoni. 2020. AndroDFA: Android Malware Classification Based on Resource Consumption. *Information (Switzerland)* 11 (2020), 326–346.

[41] Sanjay Mate, Vikas Somani, and Prashant Dahiwale. 2023. ChatGPT: Optimizing Text Generation Model for Knowledge Creation. *i-Manager's Journal on Software Engineering* 17, 3 (2023), 21–26.

[42] M. Nisa, J. H. Shah, S. Kanwal, M. Raza, and T. Blaauskas. 2020. Hybrid Malware Classification Method Using Segmentation-Based Fractal Texture Analysis and Deep Convolution Neural Network Features. *Applied Sciences* 10, 14 (2020), 4966–4989.

[43] S. O'Dea. 2024. *Android - Statistics & Facts.* Statista. https://www.statista.com/statistics/272698/global-market-share-held-by-mobile-operating-systems-since-2009/

[44] OpenAI. 2019. Language Models are Unsupervised Multitask Learners. (2019), 1–24.

[45] OpenAI. 2020. Language Models are Unsupervised Multitask Learners. (2020), 1–12.

[46] OpenAI. 2023. GPT-4 Technical Report. *ArXiv* (2023), 1–100.

[47] Alec Radford. 2018. Improving Language Understanding by Generative Pre-Training. *OpenAI* (2018), 1–12.

[48] A. Salah, E. Shalabi, and W. Khedr. 2020. A Lightweight Android Malware Classifier Using Novel Feature Selection Methods. *Symmetry* 12 (2020), 858–874.

[49] A. Saracino, D Sgandurra, G. Dini, and F. Martinelli. 2018. MADAM: Effective and Efficient Behavior-based Android Malware Detection and Prevention. *IEEE Transactions on Dependable & Secure Computing* 15 (2018), 83–97.

[50] Weisong Sun, Chunrong Fang, Yudu You, Yun Miao, Yi Liu, Yuekang Li, Gelei Deng, Shenghan Huang, Yuchen Chen, Quanjun Zhang, Hanwei Qian, Yang Liu, and Zhenyu Chen. 2023. Automatic Code Summarization via ChatGPT: How Far Are We? *ArXiv* (2023), 1–13.

[51] R. Surendran, T. Thomas, and S. Emmanuel. 2020. On Existence of Common Malicious System Call Codes in Android Malware Families. *IEEE Transactions on Reliability* 70 (2020), 248–260.

[52] R. Taheri, R. Javidan, M. Shojafar, Z. Pooranian, and M. Conti. 2020. On Defending Against Label Flipping Attacks on Malware Detection Systems. *Neural Computing and Applications* 32 (2020), 14781–14800.

[53] G. Tao, Z. Zheng, Z. Guo, and M. R. Lyu. 2018. MalPat: Mining Patterns of Malicious and Benign Android Apps via Permission-Related APIs. *IEEE Transactions on Reliability* 67 (2018), 355–369.

[54] Connor Tumbleson and RyszardWiśniewski. 2023. *APKtool.* https://ibotpeaches.github.io/Apktool/

[55] P. D. Varna and P. Visalakshi. 2020. Detecting android malware using an improved filter based technique in embedded software. *Microprocessors and Microsystems* 76 (2020), 103115–103127.

[56] Devyani Vij, Vivek Balachandran, Tony Thomas, and Roopak Surendran. 2020. GRAMAC: A Graph Based Android Malware Classification Mechanism. In *Proceedings of the Tenth ACM Conference on Data and Application Security and Privacy*. 156–158.

[57] A Vp, B Az, and C Mc. 2019. A machine learning based approach to detect malicious android apps using discriminant system calls. *Future Generation Computer Systems* 94 (2019), 333–350.

[58] Fengguo Wei, Yuping Li, Sankardas Roy, Xinming Ou, and Wu Zhou. 2017. Deep Ground Truth Analysis of Current Android Malware. In *Detection of Intrusions and Malware, and Vulnerability Assessment*. 252–276.

[59] Bozhi Wu, Sen Chen, Cuiyun Gao, Lingling Fan, Yang Liu, Weiping Wen, and Michael R. Lyu. 2021. Why an Android App Is Classified as Malware: Toward Malware Classification Interpretation. *ACM Trans. Softw. Eng. Methodol.* 30, 2 (mar 2021), 1–29.

[60] Nannan Xie, Xing Wang, Wei Wang, and Jiqiang Liu. 2018. Fingerprinting Android malware families. *Frontiers of Computer Science* 13 (2018), 637–646.

[61] Ke Xu, Yingjiu Li, Robert Deng, Kai Chen, and Jiayun Xu. 2019. DroidEvolver: Self-Evolving Android Malware Detection System. In *IEEE European Symposium on Security and Privacy*. 47–62.

[62] Zhiwu Xu, Kerong Ren, Shengchao Qin, and Florin Craciun. 2018. CDGDroid: Android Malware Detection Based on Deep Learning Using CFG and DFG. In *IEEE International Conference on Formal Engineering Methods*.

[63] Lei Xue, Xiapu Luo, Le Yu, Shuai Wang, and Dinghao Wu. 2017. Adaptive Unpacking of Android Apps. In *IEEE/ACM International Conference on Software Engineering*. 358–369.

[64] Yerima, Y. Suleiman, Sezer, and Sakir. 2019. DroidFusion: A Novel Multilevel Classifier Fusion Approach for Android Malware Detection. *IEEE Transactions on Cybernetics* 49 (2019), 453–466.

[65] Le Yu, Xiapu Luo, Jiachi Chen, Hao Zhou, Tao Zhang, Henry Chang, and Hareton K. N. Leung. 2021. PPChecker: Towards Accessing the Trustworthiness of Android Apps' Privacy Policies. *IEEE Transactions on Software Engineering* 47, 2 (2021), 221–242.

[66] Jiansong Zhang. 2023. How Can ChatGPT Help in Automated Building Code Compliance Checking? , 63-70 pages.

[67] Yueqian Zhang, Xiapu Luo, and Haoyang Yin. 2015. DexHunter: Toward Extracting Hidden Code from Packed Android Applications. In *European Symposium on Research in Computer Security*. 293–311.

[68] X. Zhi, T. Guo, Q. Zhang, C. Yu, and X. Kai. 2018. Android Malware Detection Methods Based on the Combination of Clustering and Classification. In *International Conference on Network and System Security*. 411–422.

[69] Q. Zhou, F. Feng, Z. Shen, R. Zhou, M. Y. Hsieh, and K. C. Li. 2019. A novel approach for mobile malware classification and detection in Android systems. *Multimedia Tools and Applications* 78 (2019), 3529–3552.

[70] Huijuan Zhu, Yang Li, Ruidong Li, Jianqiang Li, Zhuhong You, and Houbing Song. 2021. SEDMDroid: An Enhanced Stacking Ensemble Framework for Android Malware Detection. *IEEE Transactions on Network Science and Engineering* 8, 2 (2021), 984–994.

[71] H. J. Zhu, T. H. Jiang, B. Ma, Z. H. You, W. L. Shi, and L. Cheng. 2017. HEMD: a highly efficient random forest-based malware detection framework for Android. *Neural Computing & Applications* 30 (2017), 3353–3361.

[72] Jinting Zhu, Julian Jang-Jaccard, Amardeep Singh, Ian Welch, Harith AI-Sahaf, and Seyit Camtepe. 2022. A few-shot meta-learning based siamese neural network using entropy features for ransomware classification. *Computers & Security* 117 (2022), 102691–102720.