Are iOS Apps Immune to Abusive Advertising Practices?

Tianming Liu
Monash University
Melbourne, Australia
Tianming.Liu@monash.edu

Jiapeng Deng†
Huazhong University of Science and Technology
Wuhan, China
jiapeng@hust.edu.cn

Yanjie Zhao†
Huazhong University of Science and Technology
Wuhan, China
yanjie_zhao@hust.edu.cn

Xiao Chen
University of Newcastle
Newcastle, Australia
xiao.chen@newcastle.edu.au

Xiaoning Du*
Monash University
Melbourne, Australia
Xiaoning.Du@monash.edu

Li Li
Beihang University
Beijing, China
lilicoding@ieee.org

Haoyu Wang*†
Huazhong University of Science and Technology
Wuhan, China
haoyuwang@hust.edu.cn

Abstract

Mobile app usage has surged globally, leading to the proliferation of in-app advertising. However, this business model introduces significant challenges, including inappropriate ad content in children's apps, security risks, and deceptive practices. While extensive research has examined abusive ad practices in Android apps, iOS remains largely unexplored. Our study addresses this critical gap by developing iAdAuditor, a specialized ad auditing tool for iOS apps. Through a comprehensive empirical analysis of over 6,000 top-listed iOS apps, we reveal concerning patterns of abusive ad practices: over 3% of ad multimedia materials contain explicit content, 14.9% of ads employ deceptive tactics, and 20% of app promotions are age-inappropriate. By extending ad scrutiny to iOS, our work aims to equip stakeholders in the mobile advertising industry—Apple, ad network companies, developers, and third-party auditors—with empirical insights and robust tools to better identify, measure, and mitigate these issues, ultimately contributing to fostering a more responsible, secure, and user-centric mobile app ecosystem.

CCS Concepts

• **Security and privacy** → *Software security engineering*.

Keywords

Mobile Advertising; Content Compliance; iOS Application

ACM Reference Format:
Tianming Liu, Jiapeng Deng, Yanjie Zhao, Xiao Chen, Xiaoning Du, Li Li, and Haoyu Wang. 2025. Are iOS Apps Immune to Abusive Advertising Practices?. In *33rd ACM International Conference on the Foundations of Software Engineering (FSE Companion '25), June 23–28, 2025, Trondheim, Norway.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3696630.3728571

*Xiaoning Du and Haoyu Wang are the corresponding authors.
†The full name of the authors' affiliation is Hubei Key Laboratory of Distributed System Security, Hubei Engineering Research Center on Big Data Security, School of Cyber Science and Engineering, Huazhong University of Science and Technology.

FSE Companion '25, Trondheim, Norway

ACM ISBN 979-8-4007-1276-0/25/06
https://doi.org/10.1145/3696630.3728571

1 Introduction

The past decade has witnessed a transformative surge in smartphone usage, with global user numbers reaching an astounding 4.88 billion in 2024, signifying that over 60% of the world's population is now connected through smartphones [81]. This remarkable growth has been fueled by, and in turn has catalyzed, the extensive evolution of the mobile app ecosystem. The Apple App Store alone now boasts a vast repertoire of 2 million apps [39]. However, over 95% of these apps are offered free of charge, with this proportion continuing to grow [17]. At the heart of this trend lies in-app advertising. While users do not pay to install these apps, developers can still monetize by displaying advertisements within their apps. The effectiveness of this strategy is evident in statistics showing that advertising accounts for 60% of mobile app revenue, surpassing in-app purchases at 38% [8]. Indeed, advertising has become the economic backbone of the app ecosystem, with global in-app ad spending amounting to a staggering 352 billion dollars in 2024 and showing continuous growth [29].

However, the proliferation of in-app advertising also brings significant challenges to the mobile app ecosystem. Of paramount concern is the inappropriate content of advertisements, particularly in apps targeting children. Alarmingly, multiple news reports have documented instances where children's apps on Google Play displayed sexually suggestive or even pornographic content through in-app ads [70, 73, 76]. Beyond content issues, security risks loom large, with the propagation of malware and scam campaigns through in-app ads, compromising user privacy and financial safety [50, 64, 75]. Compounding these problems is the prevalence of deceptive or intrusive ads designed to trick users into clicking, either to generate additional ad revenue or promote products, further undermining user trust and experience [52, 64, 92].

In response to these challenges, numerous research efforts have focused on understanding and detecting such abusive practices in in-app advertising (e.g., [50, 64, 75, 92, 94]). However, a significant gap persists in this body of research: an overwhelming focus on Android, while iOS - commanding a substantial 27.6% market share in global mobile operating systems [33] - remains largely unexplored.

This can be attributed to two primary factors. Firstly, iOS's closed-source nature presents unique technical challenges for app scrutiny. Secondly, iOS prohibits direct app installation (also referred to as sideloading) [78], creating a barrier for malware propagation through in-app ads, as all app promotions must go through the official app market. This stands in stark contrast to Android, where malware APK promotion via in-app ads is prevalent [64, 75], particularly in regions with limited access to the official Google Play store.

However, the absence of direct malware file promotion does not render iOS immune to other advertising-related issues. Inappropriate ad content and rogue ad practices remain potential threats, as these originate from advertisers and could theoretically affect both Android and iOS apps indiscriminately. Interestingly, while several news reports exist of porn ads in Google Play apps, similar reports for iOS are scarce, though multiple instances of user complaints on Apple's online community, featuring hundreds of users echoing similar concerns, suggest the problem may not be entirely absent from iOS (e.g., [3–7]).

To address this critical gap and empirically assess the prevalence of abusive ad practices in iOS apps at scale, we have developed a specialized tool, iAdAuditor, to audit in-app ads against a diverse range of abusive ad practices we summarized through reviewing relevant literature and regulatory documents. iAdAuditor operates by taking an iOS app as input and automatically executing it in a controlled environment, while monitoring for ad displays and simulating user interactions to access advertised content. The tool then meticulously analyzes the captured network traffic to extract ad-related content, including ad visuals, redirected web pages, and app store promotions, for subsequent detection of abusive practices.

We conduct extensive experiments by applying iAdAuditor to over 6,000 top-listed iOS apps, providing the first large-scale empirical assessment of abusive ad practices in the iOS ecosystem. Our findings reveal concerning patterns: over 3% of ad videos and images contain explicit content, with rare but alarming cases even in apps targeted at children (left of Figure 3). Additionally, 20% of app promotions on the Apple App Store feature age ratings exceeding that of the original app, potentially exposing young users to unsuitable material. While Apple's efforts to prohibit third-party ads in Kids apps have been largely effective, nearly 3% of these apps still breach this guideline. Furthermore, 14.9% of in-app ads are identified as deceptive or disruptive, employing tactics such as mimicking system notifications or failing to clearly identify themselves as ads.

These empirical results demonstrate that iOS apps, despite stringent platform controls, still suffer from various abusive ad practices. By extending ad scrutiny to iOS, our work aims to equip stakeholders in the mobile advertising industry—Apple, ad network companies, developers, and third-party auditors—with empirical insights and robust tools to better identify, measure, and mitigate these issues, ultimately contributing to fostering a more responsible, secure, and user-centric mobile app industry.

In conclusion, the major contributions of this work are as follows:

- We present the first initiative to investigate and understand the prevalence of abusive ad practices within iOS apps.
- We provide iAdAuditor, a tool for auditing ads in iOS apps against abusive practices. We have made iAdAuditor open-source at https://github.com/iAdAudit/iAdAuditor to boost future research and industrial efforts.
- By applying iAdAuditor to over 6,000 top iOS apps, we present the first large-scale assessment of abusive ad practices in the iOS ecosystem, providing empirical evidence and insights into the prevalence of abusive ad practices in iOS apps.

2 Background

2.1 The Mobile In-app Advertising Ecosystem

To provide essential background, Figure 1 presents a simplistic overview of the mobile in-app advertising ecosystem, which involves four key participants:

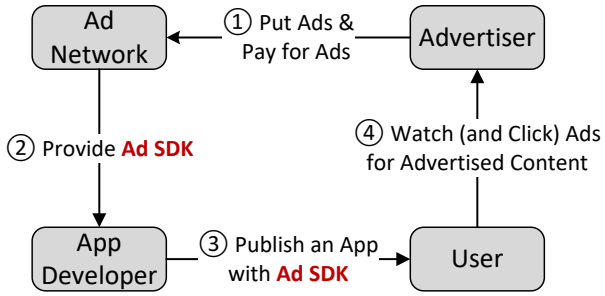

Figure 1: Overview of the In-app Advertising Ecosystem

- *Advertisers*, who seek to deliver ads to app users and pay for the ad delivery;
- *App Developers*, who integrate ads in their apps for revenue, typically based on the number of ad displays (also referred to as ad impressions) and/or ad clicks within their apps [2];
- *Ad Networks*, who serve as intermediaries between advertisers and developers. They provide an **Ad SDK** that developers can easily integrate into their apps. The **Ad SDK** handles all necessary ad activities within the app, including ad delivery and display;
- *Users* of the app, who consume the ads and receive the advertised content by viewing (and potentially clicking on) the ads.

Within this ecosystem, in-app ads are generally delivered at app runtime, utilizing techniques such as real-time bidding [31] to maximize revenue for all parties involved. However, this process highlights **a critical challenge**: as in-app ads are placed by advertisers and distributed through ad networks **at runtime**, they often **elude the direct oversight of app developers and app stores**. This lack of immediate control creates an environment where abusive ad practices can potentially proliferate, underscoring the complexity of maintaining integrity within the in-app advertising ecosystem.

2.2 Potential Abusive Ad Practices in iOS Apps

In this study, we use the term 'Abusive Ad Practices' to encompass a range of problematic behaviors in mobile advertising, including but not limited to inappropriate ad content, security threats, and deceptive ad designs. These practices, as introduced in §1, can significantly undermine user experience, privacy, and safety. This

section delves deeper into the specific types of abusive ad practices that could potentially exist in iOS apps.

Through a comprehensive review of relevant laws and regulations (such as COPPA [54] and GDPR [56]), existing literature (e.g., [50, 64, 75, 94]), and mainstream app store review guidelines (Apple [10] and Google Play [1]), we have identified **three** types of abusive ad practices that could potentially occur in iOS apps.

- **Inappropriate or Restricted Ad Content**: This category encompasses ad content that violates local administrative policies or is unsuitable for the app's content rating. For instance, an app rated as appropriate for all ages might display ads with explicit content intended only for adult audiences. The nature of inappropriateness can vary based on regional policy differences (e.g., [18]) and may include sexual content, hate speech, violence, controlled substances, gambling, etc. [42]. Content inappropriateness in in-app ads can manifest both within the app (e.g., displaying inappropriate ad images [51, 94]) and external to the app (e.g., ad clicks redirecting users to app store pages for adults-only apps). A recent study [94] revealed the prevalence of such issues in Google Play apps, with over 10% of examined ads exhibiting inappropriateness for children and nearly half of the Google Play app promotions being age-inappropriate. Content inappropriateness remains a significant challenge for the mobile ad ecosystem.
  - A notable sub-category of this abusive practice relates to one of Apple's key protective measures for children: to provide young users with a safer experience, **Apple mandates that apps in the Kids category must not include third-party advertising** [10][1]. Our audit specifically aims to verify compliance with this regulation, assessing the extent to which developers adhere to this crucial child protection policy.
- **Deceptive or Disruptive Ads**: This category encompasses in-app ads that significantly undermine user experience or employ manipulative tactics to elicit user interaction, often aimed at maximizing user engagement or generating undue ad revenue. Such practices include ads that fail to clearly identify themselves as ads, ads that employ deceptive design elements to trick users into unintended interactions, and full-screen ads lacking appropriate close/skip buttons. Such practices not only violate ethical advertising standards but also infringe upon platform regulations [10].
- **External Security Threats**: While iOS apps typically resist direct malware file promotion via in-app ads, they remain susceptible to external security risks. Ad interactions can potentially redirect users to fraudulent websites or scam campaigns that operate beyond the confines of the app environment. Extensive research on Android [50, 64, 75] has highlighted the prevalence and significance of such external security threats in mobile advertising, suggesting iOS apps could face comparable challenges. Notably, a recent study [67] revealed a new form of malware promotion through in-app ads: rather than direct malware file downloads, ads promote malicious apps available on Google Play, Android's official app store. While a similar threat might exist in the iOS ecosystem, this vector was not covered in our study as their work was published after our investigation.

It is crucial to note the boundaries of our study, which focuses on auditing ads in iOS apps against the aforementioned abusive practices. Our research deliberately excludes privacy-related advertising issues, such as user tracking, identification, and personal information collection. These topics, being vast and extensively addressed in existing literature (e.g., [59, 72, 77, 79]), warrant separate, dedicated studies due to their complexity and scope.

[1]With certain exceptions, which will be discussed in detail in §4.2.2.

3 Approach

To systematically audit iOS in-app advertisements for abusive practices at scale, we have implemented iAdAuditor, a dynamic analysis framework that takes an iOS app installation package as input and outputs detected abusive ad practices. We opted for a dynamic analysis-based approach due to the runtime loading nature of in-app ads, which precludes effective static analysis. Figure 2 illustrates an overview of iAdAuditor, which comprises three modules:

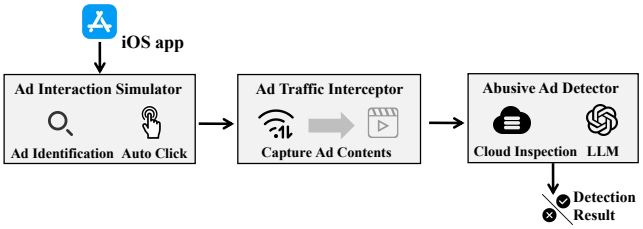

Figure 2: Overview of iAdAuditor

- **Ad Interaction Simulator**: This module is responsible for dynamically running the app, monitoring ad loading, and simulating user interactions with ads to trigger advertised content.
- **Ad Traffic Interceptor**: This module captures and analyzes network traffic related to in-app advertising by instrumenting network APIs on iOS. It differentiates ad-related network traffic and collects detailed contents during ad delivery, such as ad images and videos, for subsequent ad abuse analysis.
- **Abusive Ad Practice Detector**: This module leverages the ad contents collected from the previous two modules to identify and classify various abusive ad practices based on predefined criteria and heuristics, which will be outlined in §3.3.

3.1 Ad Interaction Simulator

This module performs three critical tasks:

- App testing: triggers ad loading through automated app testing.
- Ad widget identification: monitors and identifies ad widgets in real-time as they load during testing.
- Ad click simulation: simulates user clicks on identified ads to access advertised content.

These tasks work in concert to facilitate iAdAuditor's ad harvesting process. While the first and third tasks are relatively straightforward, the second presents unique challenges in iOS.

3.1.1 App Testing and Ad Click Simulation. The first task is relatively straightforward, as previous research has shown that simple automated testing is often sufficient for ad discovery. Static analysis [61] revealed that over 85% of in-app ads are located within the first two layers of app activities, aligning with developers' interests. Dynamic approaches in Android ad auditing studies [50, 64, 94] have successfully relied on simple testing strategies to harvest ads,

with some even limiting the testing to opening the app and harvesting ads on the first screen without further interaction. Therefore, we opt to adopt CydiOS [86], a state-of-the-art off-the-shelf iOS app testing tool, to facilitate this task.

For the third task, once ad widgets are identified in real-time (introduced subsequently), simulating a click action is as simple as executing a touch event at the ad's coordinates. We modified CydiOS to achieve this using the Appium testing framework [12].

3.1.2 Ad Widget Identification in iOS. Ad widget identification is a critical and more challenging task in our framework. Precise identification is fundamental to valid abusive ad detection, as misclassification of app UI elements could lead to false positives. For instance, legitimate app components might be erroneously flagged as deceptive ads failing to signal their advertising nature. Moreover, real-time identification is essential to ensure our automated testing can interact with ads before navigating away, enabling the collection of post-click content for comprehensive ad abuse analysis.

Previous approaches to this problem [50, 55, 57, 71] heavily relied on heuristics based on widget types and sizes, which are prone to errors due to diverse ad designs. A recent work, AdRambler [94], proposed a novel and precise approach, identifying ad widgets based on their fundamental distinction from other UI elements that ad widgets are implemented by ad SDKs. However, their method, tailored for Android's Java-like package structure, is not directly applicable to iOS. Consequently, we developed a custom solution for iOS based on this fundamental distinction.

Our ad widget identification process comprises several key steps:

- **UI Hierarchy Extraction:** We obtain the UI hierarchy file of the current device page before each simulated input during dynamic app testing. This is achieved by instrumenting iOS's UIKit component [48] using Theos [24], a widely-used tool for iOS system monitoring and modification. The UI hierarchy file includes all widgets within the page, along with each widget's class name—the class responsible for implementing the view.
- **Ad SDK Collection:** We reference a list of top ad SDKs employed in iOS apps [47], as compiled by 42Matters [35], a leading mobile app intelligence provide frequently sourced by previous mobile app research (e.g., [60, 68, 69, 95]). This continuously updated list also provides market share data for each SDK, reflecting their prevalence across the Apple App Store. To ensure a representative yet manageable dataset, we focus on SDKs with a market penetration exceeding 2%. We collect these SDKs from CocoaPods [15], a widely adopted iOS SDK distribution platform, yielding a robust dataset of 20 SDKs for our ad widget detection.
- **SDK Class Extraction:** For each SDK, we statically extract all its class names using industry-standard tools, accommodating the two primary types of iOS SDKs: static libraries and dynamic libraries [41]. For static libraries, we use the nm tool [37] to analyze extracted object files (.o) and then identify class definitions by recognizing symbols prefixed with '_OBJC_CLASS_'. In the case of dynamic libraries, we employ class-dump [23] to extract header files (.h) and subsequently extract declared class names. Furthermore, we refine the resulting class list by filtering out non-SDK-proprietary classes, such as iOS system classes, based on iOS class naming conventions and string matching. This refinement ensures our analysis focuses exclusively on SDK-specific classes.
- **Ad Widget Identification:** Finally, we determine if a widget is an ad by checking if its class name in the UI hierarchy file appears in our extracted ad SDK class name list.

This methodology enables precise ad widget identification while effectively adapting to iOS's unique technical landscape. Moreover, **by identifying the ad's SDK, it facilitates pinpointing which ad networks deliver abusive ads, enabling accountability and opening avenues for future mitigation strategies**.

3.2 Ad Traffic Interceptor

When the Ad Interaction Simulator is running, we configure the testing device's traffic to go through mitmproxy [34], a widely-used HTTPS proxy, to record and decrypt all network traffic from the device. This obtained traffic contains ad-related traffic, from which we can extract multimedia files used for ad display such as images and videos to check for its appropriateness, and web pages the ads redirect to, to check if it contains security threats. However, non ad-related traffic is also included within, such as app's own traffic or system traffic, which should be filtered out for a precise ad auditing.

To differentiate ad-related traffic, we instrumented iOS network APIs to track the origin of network connections, specifically to determine if they originate from ad SDKs, similar to previous works on Android network analysis such as [64, 97]. This approach is based on the principle that all in-app ad activities, including ad fetching, are facilitated by ad SDKs. Our method comprises the following steps:

- **Network API Instrumentation:** We instrument iOS network APIs to track the origin of network connections, focusing on the widely-used iOS *NSURLSession* [38] APIs—Apple's standard networking framework that is widely adopted in iOS development and serves as the foundation for popular network libraries like AFNetworking [21] and Alamofire [22]. This is achieved using Frida[19], a cross-platform dynamic instrumentation tool commonly used in security research. This instrumentation allows us to obtain the domain name each network API call connects to.
- **Call Stack Trace:** Next, we trace the call stack of the instrumented network API call to determine whether each network call originate from ad libraries. This process is complicated by iOS's Address Space Layout Randomization (ASLR) [43], a security feature that randomizes the memory locations of executables and libraries to protect against memory exploitation attacks. To overcome ASLR, we extract call stacks from the CPU context and calculate virtual memory slides to determine actual addresses relative to module base addresses, allowing us to retrieve module files, class, and method names initiated the network connection. By backtracking the obtained stack and cross-referencing with SDK-specific class lists obtained in §3.1.2, we can identify whether the connection was initiated by an ad SDK.
- **Ad Traffic Identification:** Based on the call stack analysis, we can flag domain names associated with ad-related connections. This allows us to create a filter that we apply to all traffic recorded by mitmproxy, thereby effectively isolating ad-related traffic for subsequent ad abuse analysis.

After obtaining ad-related traffic, we extract detailed ad contents from the traffic for subsequent ad abuse analysis. These contents serve as input to our next module and include:

Table 1: Classification Rule and Detection Method of Abusive Ad Practices

Abuse Type	Rule	Description	Detection
Type 1: Inappropriate Ad Content	**R1.1 Explicit Ad Multimedia**	Ad multimedia files may contain explicit content inappropriate for users, especially children	Google Cloud Vision API [16], coupled with manual confirmation
	R1.2 Age-Inappropriate App Promotions	Ads promoting apps beyond the host app's age group (e.g., all-age app promoting adult app)	Comparing age ratings of host and promoted apps from Apple App Store
	R1.3 Ads in Kids' Apps	Apple generally prohibits third-party ads in Kids category apps to provide young users with a safer experience [10]	Presence of ads detected by iAdAuditor indicates rule violation
Type 2: Deceptive or Disruptive Ads	**R2.1 Unclear Ad Identification**	Ads should clearly identify themselves to prevent confusion or unintended clicks	Leveraging Multimodal Large Language Models (e.g., GPT-4o [26]), coupled with manual confirmation
	R2.2 Misleading Ad Content	Ads that deceive users into clicking (e.g., fake system notifications, false prizes)	
	R2.3 Intrusive Full-Screen Ads	Full-screen ads lacking proper close/skip options disrupt user experience	
Type 3: External Security Threats	**R3.1 Malicious Web Pages**	In-app ads may promote malicious web pages, such as those associated with scams or fraud	VirusTotal [49] scanning for maliciousness detection

- **Ad Display Multimedia Files:** Ad images and videos extracted by filtering the content types of ad-related traffic responses.
- **Apple App Store Promotion Pages:** Pages showcasing apps promoted by in-app ads, identified through post-ad-click network connections to *itunes.apple.com*.
- **Advertised Web Pages:** Destinations which in-app ads redirect users to, captured from ad-related traffic following ad clicks.
- **Ad Widget Screenshots:** Unlike the above, these are not derived from network traffic. Instead, we extract them by cropping the device screenshot based on the ad widget's coordinates within the UI hierarchy file. This method preserves the ad's appearance as seen by users during interaction, which traffic-based ad images may fail to accurately represent due to segmentation (for example, ad creatives and close buttons may be delivered as separate multimedia files).

3.3 Abusive Ad Practice Detector

This module identifies and classifies the three types of abusive ad practices introduced in §2.2, utilizing the previously collected ad contents as input. We have defined seven specific rules across these three abuse categories. Table 1 provides a overview of each rule, including its description and a concise explanation of the corresponding detection method.

The classification of these seven in-app ad practices as abusive is largely self-evident. Not only have they been identified as problematic by previous research and Google's regulatory framework, Apple has also explicitly prohibited them in its app review guidelines [10]. This is particularly noteworthy for R2.1, R2.2, and R2.3, which might not be as immediately apparent as abusive compared to the others.

The detection methods for these rules need further explanation:

R1.1: To detect explicit content in ad multimedia files, we utilize Google Cloud Vision API's SafeSearch Detection Feature [16], a commercial off-the-shelf solution. This API, widely adopted in previous Android-focused ad content auditing studies [50, 64, 94], specializes in identifying explicit content in images and videos. The API analyzes input multimedia files for several types of explicitness, including adult (pornographic or sexually suggestive) and violence. It returns a five-level likelihood scale for each type, ranging from "very unlikely" to "very likely". Following previous work, we consider "very likely" and "likely" as positive indicators of explicit content. We then manually examine each flagged case to verify its validity.

While this API is specialized, it is important to note that it may not detect all possible types of inappropriate ad content (e.g. those mentioned in §2.2). Nevertheless, we deem it appropriate for our study as it covers the two major categories of inappropriateness—adult and violence—and is widely accepted in the research community. Future efforts could enhance this module with more comprehensive detection capabilities for a broader range of explicit content types.

R1.2: The age group of apps promoted through in-app ads should not exceed that of the host app. Otherwise, it risks exposing younger users to unsuitable content.

R1.3: Apple prohibits third-party ads in Kids apps. As our approach detects third-party ads delivered through ad networks, any ads we identify in apps from the kids category automatically violate this rule.

R2.1-R2.3: We first employ Large Language Models (LLMs) to identify potential violations, as traditional methods struggle to effectively analyze the deceptive or disruptive nature of ad widget screenshots. LLMs, representing recent technological advancements with enhanced semantic understanding capabilities, have been widely adopted across various software engineering and security tasks [58, 88]. Their multimodal ability to process image inputs has also shown promise in understanding app UIs, as demonstrated in app testing [89, 90] and bug finding [62, 66] tasks. Leveraging this potential, we apply LLMs to detect these deceptive or disruptive ads.

Table 2 provides a detailed illustration of our prompt methodology for identifying deceptive or disruptive ads (R2.1 to R2.3) using GPT-4o [26]. We implement few-shot learning [82] to enhance LLM performance by first providing the model with carefully crafted examples of violating ad screenshots for each rule before conducting actual assessments. Four classical examples were crafted based on real-world ads, two of which are included in Table 2. For detection, we utilize the ad widget screenshots generated by iAdAuditor to construct individual abuse identification prompts.A screenshot is considered potentially abusive if the LLM responds affirmatively (YES) to any of the assessment questions.

However, given the known propensity of LLMs for hallucination problems [91], we implemented an additional layer of verification. All cases initially identified as violations by the LLM were subsequently subjected to manual confirmation to ensure the accuracy

and reliability of our final results. The scalability concern arising from manual confirmation, together with the impact of potential false negatives of the detectors, is further discussed in §5.2.

Table 2: An example of our prompt engineering for detecting deceptive or disruptive ads

Prompt Type	Instantiation
Task Description	Evaluate images for violations of three guidelines on deceptive or disruptive ads. Learn from provided examples before assessing the target image.
Guidelines	1. Ads that don't clearly identify themselves 2. Ads that deceive users into clicking (e.g., fake notifications, false prizes) 3. Full-screen ads lacking proper close/skip options
Example Cases	Example 1: input: #An ad image with no abusive practice output: "1.No. It contains the word "AD" to explain it's an ad." "2.No. It doesn't contains false prizes or fake system notifications. " "3.No. The advertisement features a cross symbol in the top-left corner for closing the image." Example 2: input: #Another ad image violating R2.2 and R2.3 output: "1.No. It contains the word "Advertisement" to explain it's an advertisement and also has a Google Ads identifier in the top right corner in Ad." "2.Yes. It not only contains a fake system notification to deceive users into clicking, but also features a fake close button that misleads users into clicking on the ad." "3.Yes. Even though there is an 'x' in the image, its position is part of the ad content, not a functional close button." Example 3 and 4 omitted...
Assessment Questions	Evaluate if the image violates each of the three guidelines, answering YES or NO for each.

R3.1: To detect malicious web pages promoted by in-app ads, we utilize VirusTotal, a comprehensive online security threat detection platform [49]. VirusTotal aggregates over 60 renowned antivirus engines and has gained widespread adoption in the security community for detecting threats in both files and URLs (e.g., [53, 74]). We leverage this robust platform by inputting the URLs of ad-promoted web pages for analysis to identify its potential security risks.

4 Experiment

4.1 Experiment Setup

Testing Device. Our experiment was conducted on three second-hand iPhone 8 devices running iOS 14. This choice was motivated by their affordability and that iOS 14 is one of the newest versions that can be easily and fully jailbroken [32]. Jailbreaking is crucial for our study as it provides the necessary administrative rights to implement key functionalities of iAdAuditor, such as obtaining device UI hierarchy for app testing and precise ad widget identification (§3.1), and instrumenting iOS network APIs to differentiate ad-related traffic (§3.2). However, it is important to note that the requirement for jailbroken devices does not limit iAdAuditor's applicability to older iOS versions only. iAdAuditor is designed for both research purposes, as in this study, and for use by Apple and third-party auditing authorities, who possess the necessary administrative access to implement it on the latest devices and iOS versions.

For testing efficiency, we executed each app for a duration of 2 minutes during our experiment, aligning with practices in comparable Android studies [64, 94]. To accommodate ad loading times, we implemented a 5-second interval between successive action inputs. Our experiment was conducted in September and August, 2024.

Dataset and Target Region: Our dataset comprises top free iPhone apps from the Apple App Store, focusing on the United States and China markets. Our dataset collection was conducted in September 2024 and involved the following steps for each region:

- Aggregated the current top 200 apps across 23 app categories (all but the Kids category), totaling 4,600 apps per region.
- Extracted app metadata from the official Apple App Store and identified third-party advertising presence with Apple's App Privacy Disclosure [9].
- Filtered the 4,600 apps to retain only those declared with ad presence.
- Acquired app installation packages (IPA files) using ipatool [40] from the official Apple App Store.

Given the importance of auditing abusive ads in children's apps and verifying adherence to Apple's prohibition of ads in Kids apps (R1.3), we paid particular attention to the Kids category. For this category, we aggregated and de-duplicated the top 500 Kids apps on the App Store for the first day of each month over the past year[2]. These apps were included in our dataset without filtering for ad presence, ensuring a comprehensive analysis of the Kids app landscape.

Table 3 presents an overview of our final dataset composition.

Table 3: Dataset Overview

App Type	# US Apps	# CN Apps	# Total
Top regular apps with ads	2,640	2,075	4,715
Top Kids apps	784	823	1,607
Total Apps	**3,424**	**2,898**	**6,322**

Our experiment focused on apps from the United States and China, representing two distinct landscapes of mobile app advertising. The U.S. market generally utilizes globally popular ad networks [35] like Google's AdMob [25], while China's app ecosystem, characterized by its vast scale and limited access to Google services, has fostered a unique in-app advertising environment with different trending ad networks and companies [27]. We selected these two regions for their representativeness, aiming to demonstrate iAdAuditor's effectiveness while gaining initial insights into abusive ad practices in iOS apps across diverse markets.

The decision to limit our study to these two regions was influenced by several factors: the substantial number of apps required for a comprehensive understanding of each region, our limited familiarity with ad networks in other regions (which could lead to suboptimal ad identification), and our goal, as the first study on iOS abusive ad practices, to provide a thorough understanding of a few regions rather than a broad but shallow cross-regional analysis, which we believe warrants separate, dedicated research efforts.

[2]Collected from Qimai [13], a reputable third-party App Store Optimization (ASO) website widely adopted as data source in mobile app research (e.g., [80, 84, 85, 96]).

4.2 Results

4.2.1 Overall Ad Presence. After applying iAdAuditor across all 6,322 apps in our dataset, **994** apps were identified with in-app ads, including **44 Kids apps**. Specifically, iAdAuditor detected ad presence in 20.15% (950/4,715) of top regular apps and 2.73% (44/1,607) of top Kids apps. Interestingly, despite all top regular apps declaring ad presence in their App Store's Privacy Disclosures, iAdAuditor could only confirm ads in 20% of these apps. This discrepancy can be attributed to several factors. Many top-listed apps require user registration and sign-in before granting access to their main functionality and displaying ads. The diverse and often complex user interactions required for these authentication processes pose significant challenges for automated testing to achieve. Furthermore, automatic dynamic testing often struggles to achieve comprehensive coverage, unable to reach every app page within the brief testing period allocated for each app. Additionally, some apps may falsely claim ad presence that cannot be triggered even after extended manual testing, possibly as a strategy to appear comprehensively declared and avoid potential scrutiny. These challenges are inherent to dynamic testing techniques, and have also been highlighted in previous Android ad auditing studies [50, 64, 94] who reported detection rate of 4%, 21.51%, and 35.88%. Our detection rate of 20.15% is comparable to [64]'s 21.51% but lower than [94]'s 35.88%, primarily due to our focus on top-tier apps that necessitate user authentication to access their full functionality more often, while both [64] and [94] selected ad-containing apps randomly[3].

Despite these constraints which may potentially lead to an underestimation of abusive ads' presence, our analysis successfully collected a substantial corpus of ad data from 16 distinct widely-adopted ad networks (whose distribution among apps across both regions are shown in Table 4), providing a robust foundation for understanding the landscape of abusive ad practices in iOS apps. We also conducted a focused evaluation to assess iAdAuditor's efficacy in accurately identifying triggered ads and their associated content, as detailed in §4.2.4.

As delineated in §3.2, our ad data collection process yielded:

- 1,630 ad widget screenshots,
- 4,880 multimedia files for ad display (comprising 3,826 ad images and 1,054 videos),
- 2,382 app promotions redirecting to the Apple App Store,
- 556 advertised web pages.

The quantity of ad multimedia files significantly exceeds the number of ad widgets. This discrepancy stems from two factors: firstly, a single ad may require multiple, segmented multimedia files, and secondly, an app may preload multiple ads, some of which may not yet be rendered within the app interface. Similarly, the count of app promotions surpasses that of ad widgets. This is attributed to our ability to analyze ad loading traffic, which reveals information about promoted apps even before the ads are displayed. However, this approach cannot be applied to advertised web pages, as accessing their final landing pages necessitates actual redirections—a process only triggered by active ad interactions.

[3]Chen et al. [50] did not filter for ad-containing apps while the other two studies filtered for apps claiming to contain ads similar to ours.

Table 4: Distribution of the 16 Ad Network SDKs and their Corresponding Parent Companies

SDK	Company	Total (CN)	Total (US)	Total
Admob	Google	284 (46.41%)	332 (86.91%)	616
Pangle (CN)	ByteDance	311 (50.82%)	10 (2.62%)	321
GDTMobSDK	Tencent	283 (46.24%)	6 (1.57%)	289
KSAdSDK	Kuaishou	171 (27.94%)	14 (3.66%)	185
AppLovin	AppLovin	37 (6.05%)	127 (33.25%)	164
FBAudienceNetwork	Meta	18 (2.94%)	91 (23.82%)	109
IronSource	IronSource	15 (2.45%)	70 (18.32%)	85
UnityAds	Unity	11 (1.80%)	61 (15.97%)	72
MintegralAd	Mintegral	23 (3.76%)	47 (12.30%)	70
BaiduMobAdSDK	Baidu	69 (11.27%)	1 (0.26%)	70
Pangle (Global)	ByteDance	17 (2.78%)	40 (10.47%)	57
DTexchange	Digital Turbine	9 (1.47%)	38 (9.95%)	47
VungleSDK	Digital Turbine	4 (0.65%)	27 (7.07%)	31
ChartboostSDK	Chartboost	1 (0.16%)	11 (2.88%)	12
AdColony	Digital Turbine	1 (0.16%)	9 (2.36%)	10
Tapjoy	Digital Turbine	2 (0.33%)	9 (2.36%)	11
Total apps with ads		612	382	994

Note: Percentages may exceed 100% as some apps utilize multiple ad SDKs.

For each collected item, iAdAuditor has meticulously tagged it with the corresponding ad SDK, enabling precise attribution to specific ad networks. This tagging not only enhances our ability to pinpoint responsibility for abusive practices but also provides valuable data for future mitigation strategies and policy recommendations.

Table 5: Prevalence of Ad Abuse Types and Top Three Contributing Ad Network Companies

Abuse Rule	Abuse Rate	Rank 1	Rank 2	Rank 3
R1.1	168 (3.42%)	AppLovin (45)	Bytedance (44)	Google (33)
R1.2	503 (21.1%)	Google (420)	Applovin (18)	Bytedance (13)
R1.3	44 (2.7%)	Bytedance (13)	Tencent (12)	Baidu (10)
R2.1	92 (5.6%)	Google (43)	Bytedance (28)	Tencent (16)
R2.2	71 (4.4%)	Google (24)	Tencent (23)	Bytedance (16)
R2.3	143 (24.9%)	Google (67)	Bytedance (22)	Tencent (16)

4.2.2 Prevalence of Abusive Ad Practices. Table 5 presents the number and percentage of each type of abusive practice outlined in §3.3, along with the three ad network companies contributing most to each violation type. Below we analyze the occurrence and distribution of these abusive ad practices across different categories.

R1.1 Explicit Ad Multimedia. As delineated in §3.3, we employed the widely adopted Google SafeSearch API for the detection of explicit ad multimedia. Among the 4,880 ad multimedia files, 220 (4.51%) were identified as abusive by the Google SafeSearch API. After manual review, **168 (3.44%) were confirmed to be abusive**. Specifically, 1 image and 4 videos were identified with violent content, while 102 images and 66 videos were identified with adult or sexually suggestive content. They originated from 7 distinct SDKs, with Applovin (26.8%), Bytedance (26.2%), and AdMob (19.6%) being the most prevalent.

Among the identified abusive ad materials, **6 items (3 videos and 3 images) were from Kids apps**. One particularly concerning example involves a U.S. Kids gaming app [36], designed for children aged 6 to 8, which was serving a sexually suggestive video ad through the AdMob SDK. This ad, promoting a TV casting app, employed highly inappropriate content for its target audience. The

video ad in question suggests that the promoted app enables users to "see through" others' clothing (as illustrated in the left image of Figure 3), a claim illustrated through depictions of several young women. This content not only misrepresents the app's actual functionality but also presents material wholly unsuitable for children. We have made the full video ad available on our website [28]. This case exemplifies the critical need for more stringent ad content auditing, especially in apps targeted at young, vulnerable users.

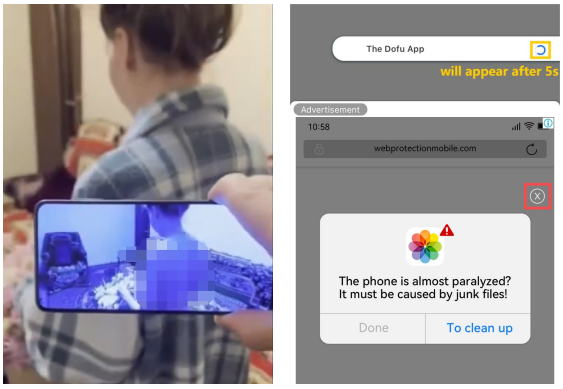

Figure 3: Examples of Abusive Ads

R1.2 Age-Inappropriate App Promotions. iAdAuditor identified a total of 2,382 app promotions where in-app ads redirect the user to the respective Apple App Store page. By comparing the age ratings of the app the host the ad, and the app being promoted, we uncovered a significant prevalence of age-inappropriate promotions. Specifically, **503 (21.1%) app promotions were found to be age-inappropriate**, where the promoted app's age rating exceeded that of the host app, with 420 of these instances from AdMob, accounting for 83.5%. Among apps with app-promotion ads, **12 apps were from the Kids category, where 6 apps** were promoting apps with higher age ratings than their own. One noteworthy case involved a gaming app designed for children over four years old. This app was promoting a game rated for ages nine and up [11], utilizing the AdMob SDK. According to the Apple App Store's official description, the promoted game may contain occasional or mild adult or suggestive themes.

This finding underscores a concerning trend in iOS advertising, where age-appropriate content boundaries are frequently transgressed, potentially exposing young users to inappropriate material.

R1.3 Ads in Kids' Apps. Among the 1,607 top Kids apps examined (as shown in Table 3), **44 (2.74%) were found to display ads**. Apple mandates that Kids apps are generally prohibited from displaying ads. However, this rule allows for an exception: ads from networks that publicly claim to use human review for ad content appropriateness are permitted [10]. Of these 44 Kids apps displaying ads, 35 are Chinese and 9 are American. The ads are served through four distinct ad SDKs, provided by ByteDance (13 apps), Tencent (12), Baidu (10), and Google (9), respectively, with the first three being prominent Chinese tech giants. Upon careful review of these ad networks' policies and guidelines [14, 20, 30, 46], we found that **none of them publicly declared the use of human review** for ad content appropriateness. Notably, Tencent and Baidu appear to lack any guidelines on how developers can prevent serving inappropriate ads to children. ByteDance [45] and Google AdMob [44], on the other hand, allow app developers to set ad content ratings that align with the user's age. However, neither explicitly states the use of human review for ad content appropriateness, likely due to the substantial resources such a process would require.

These findings **present a nuanced picture of Apple's efforts** to protect children from potentially harmful ad content. On one hand, the specific prohibition appears to be relatively effective, with only 2.74% of the examined top Kids apps breaching this regulation. However, the existence of even a few cases of non-compliance is cause for concern. The fact that some apps continue to serve ads in violation of Apple's guidelines indicates that there are **gaps in the enforcement mechanism**. Moreover, it is particularly worrisome that none of the adopted SDKs seem capable of fully meeting Apple's stringent requirements for child-safe advertising. This situation underscores the ongoing challenges in maintaining a safe digital environment for children.

R2.1 Unclear Ad Identification. From the 1,630 ad widget screenshots directly cropped from app screens, GPT-4o flagged 217 as potentially lacking clear advertising identification. Upon manual verification, **92 (5.6%, 92/1,630) of these ads were confirmed to indeed lack any form of ad identification.** The majority of these ads were traced back to two prominent ad networks: Google AdMob (43 instances) and ByteDance (28). Further analysis revealed that 41 of these confirmed ads were banner ads, while **51 were interstitial or full-screen ads**. The latter category is particularly concerning as these ad formats can interrupt or block user experience, and their lack of clear identification exacerbates this issue.

The discrepancy between GPT-4o's initial identification and manual confirmation can be attributed to the inherent limitations of LLM's image recognition capabilities. These models often struggle with distinguishing text in low resolution or low contrast scenarios, as well as identifying very small text elements.

R2.2 Misleading Ad Content. GPT-4o identified 92 ad widget screenshots as misleading, of which **71 (4.4%, 71/1,630)** were manually confirmed as true positives. Among these 71 confirmed cases, 24 originated from Google, 23 from Tencent, and 16 from ByteDance. The relatively low false positive rate of GPT-4o can be attributed to the often overt nature of misleading patterns in these ads. Our manual confirmation process also revealed two predominant categories of misleading ads. The first category involves ads that deceptively imitate system notifications to induce user interaction. A prevalent example is a deceptive full-screen ads, as shown in the right of Figure 3. This ad mimics an iOS system alert, falsely claiming that the phone is nearly paralyzed due to junk files. It aims to trick users into clicking a "clean up" button, which actually promotes a cleaning app. Morevoer, the ad cleverly features a fake close button within the ad (marked in the red box), which actually redirects to the ad's page when clicked. The second category encompasses ads that falsely promise benefits to users. Common tactics include offering illusory "red envelopes" (a Chinese cultural reference to monetary gifts) or promoting deceptive opportunities to access movies or content for free. These misleading practices exploit user trust and familiarity with system interfaces or leverage the allure of free offerings to generate clicks. Such tactics potentially lead to unintended app installations or expose users to

further deceptive content, undermining the integrity of the mobile advertising ecosystem and harming user experience.

R2.3 Intrusive Full-Screen Ads. Of the 1,630 ad widget screenshots, 575 were as full-screen ads. GPT-4o initially flagged 196 of these as potentially intrusive, with subsequent manual verification confirming **143 (24.9%, 143/575)** as true positives. Notably, the majority of these intrusive ads were served by two dominant ad networks: Google AdMob (67) and ByteDance (22). Further examination revealed a common manipulative tactic employed in these intrusive ads: the intentional delay in displaying close or skip buttons. As shown in the right of Figure 3, its genuine skip button will only appears on the top right corner of the screen after the ad has been displayed for a while. This strategy effectively forces users to view the ad content for a predetermined period before providing an option to exit. While the majority of these ads imposed viewing durations of less than 10 seconds, a particularly concerning subset of ads from ByteDance (13 instances) required users to endure a full 60 seconds of ad content before offering any means of closure. This practice of deliberately delaying exit options not only infringes upon user autonomy but also significantly disrupts the user experience.

In aggregate, our analysis of the 1,630 ad widget screenshots revealed that 243 (14.9%) exhibited at least one violation across the categories R2.1 to R2.3. This substantial proportion underscores the pervasive nature of problematic advertising practices within the iOS app ecosystem. Such a finding not only highlights the need for more rigorous ad content regulation but also calls for more transparent and user-centric ad design.

R3.1 Malicious Web Pages. Our analysis of 556 web pages advertised through in-app ads yielded intriguing results. VirusTotal flagged only 12 URLs as potentially malicious, with each flagged by merely one or two antivirus engines out of over 60. Such low detection rates typically suggest a high likelihood of false positives. Indeed, our subsequent manual inspection confirmed that **all 12 cases were, in fact, false positives**. These findings indicate that the iOS in-app advertising ecosystem has implemented effective measures to protect users from external security threats. This stands in stark contrast to several Android studies [50, 64, 71] published over four years ago, which reported frequent occurrences of malicious web pages and scam campaigns in Android in-app ads. This discrepancy could be attributed to two potential factors: either the mobile ad ecosystem has improved significantly overall in recent years, or iOS has independently achieved superior security in this domain. Determining the precise reason would necessitate a comparative analysis across both platforms, which presents an intriguing avenue for future research.

4.2.3 Comparative Analysis between Two Regions. Our analysis reveals that abusive ad practices exist in both the Chinese and U.S. app markets, albeit with varying prevalence across different categories. The Chinese market showed a lower violation rate compared to the U.S. market in terms of R1.1 (2.56% to 6.14%) and R1.2 (18.87% to 23.56%), and a much higher violation rate for R1.3 (4.25% to 1.15%). For R2.1-R2.3 (Deceptive or Disruptive Ads), the Chinese market accounted for over 70% of the 243 total violations (172). Notably, it had a disproportionately high share of misleading ad content (R2.2), constituting 88.7% (63 cases) of violations in this category. While the prevalence of abusive ad practices fluctuates between categories and regions, their persistent presence across both markets is evident. This underscores the necessity for continued vigilance and enhanced regulatory measures within the global app ecosystem.

4.2.4 Evaluation of iAdAuditor's Effectiveness. This subsection evaluates iAdAuditor's capability to accurately identify ads during app execution and comprehensively collect ad-related content. We sampled 20 apps for evaluation: 10 identified by iAdAuditor as containing ads and 10 where iAdAuditor did not detect ads during testing. For the 10 apps identified with ads, we detected 17 ads, 37 ad multimedia files (32 images and 5 videos), 18 app promotions, and 4 advertised web pages. Our manual verification of all explored app pages and captured network traffic confirmed that iAdAuditor precisely identified all 17 ads displayed during automated testing. Regarding detailed content, only 2 videos were falsely classified as ad-related, with no false negatives. This misclassification stemmed from the app and ad SDK downloading resources from the same domain (*.googlevideo.com), leading to an erroneous categorization of app traffic as ad traffic. Given the rarity of such occurrences in our broader manual checks, this minor overclassification is acceptable to ensure comprehensive capture of ad media traffic. Our manual review of the testing sessions for the 10 apps where iAdAuditor did not detect ads confirmed that no ads actually appeared during these sessions. However, it is important to note that this does not necessarily mean these apps are completely ad-free, but rather that no ads were triggered during our limited testing period. While this small-scale experiment demonstrates iAdAuditor's effectiveness in identifying ads and ad-related content when they appear, our tool still faces limitations common to dynamic analysis techniques, as detailed in §4.2.1. These limitations primarily affect iAdAuditor's ability to trigger ad displays initially. We further discuss this constraint in §5.3.

5 Discussion

5.1 Implications and Mitigation Strategies

Our study reveals that iOS apps are also not immune to abusive ad practices. Given that third-party advertising plays a crucial role in the mobile app ecosystem, yet remain beyond the full control of both app developers and app stores, combating these abusive practices necessitates a collective effort from all stakeholders.

For app stores, implementing periodic checks for abusive ad practices using techniques similar to iAdAuditor should be considered as part of their quality assurance process. The ad SDK identification capability of iAdAuditor enables app stores to hold ad networks accountable. Ad networks, being at the nexus of this ecosystem, bear a significant responsibility. They should proactively implement robust mechanisms to prevent the serving of abusive ads at the source. This could involve more stringent vetting processes for advertisers and real-time monitoring of ad content and behavior. App developers, while often not directly responsible for ad content, also play a crucial role in this ecosystem. They should exercise discretion in choosing ad SDKs, prioritizing those with strong track records of ethical practices. Additionally, developers should actively monitor the behavior of ads within their apps and be prepared to

switch providers if abusive practices are detected. Furthermore, regulatory bodies and industry associations could play a pivotal role by establishing and enforcing standards for ethical mobile advertising. This could include guidelines for ad content and experience, as well as mechanisms for reporting and addressing violations.

Ultimately, mitigating abusive ad practices in iOS apps requires a multi-faceted effort involving technological solutions, policy changes, and increased awareness among all participants in the mobile advertising ecosystem. By fostering a culture of responsibility and user-centric design, the industry can work towards a more trustworthy and sustainable mobile app environment.

5.2 Discussion on Detector's FPs and FNs

The detectors employed in iAdAuditor, namely Google SafeSearch API, LLM, and VirusTotal, each carry inherent risks of inaccuracy, potentially introducing false positives (FPs) and false negatives (FNs) to iAdAuditor's results. To mitigate FPs, we implement manual verification for all positive cases, as detailed in §3.3 and §4.2.2. While this may raise scalability concerns, the verification scope is limited to positive cases only: 220 ad images for explicit content (R1.1), approximately 500 ad screenshots for deceptive practices (R2.1-R2.3), and 12 potentially malicious URLs. As violations are often obvious to human reviewers once judging standards are established, this verification only requires two persons for independent judging and dispute resolution, plus a third person for final verification - in approximately one day's work. This manual effort is considerably modest for vetting abusive ads across over 6,000 apps. Future practitioners can adjust the thoroughness-efficiency balance by modifying detector configurations. For instance, prompt engineering can be adjusted for harsher or looser initial examination, and stricter or more lenient detectors can be adopted for explicit multimedia and malicious URL detection. More advanced and comprehensive detection solutions can also be integrated as they become available.

For FNs, while we acknowledge that they may lead to underestimating abuse prevalence, their impact on our tool's primary purpose—facilitating ad network accountability—is limited. Since ad networks repeatedly serve ads across many apps, abusive practices would still be consistently observable in large-scale measurements despite some FNs. This enables auditors to identify problematic patterns at network level rather than focusing on individual instances.

5.3 Other Limitations and Future Work

Our study is subject to several limitations that warrant acknowledgment, while providing avenues for future research. As detailed in §4.2.1, iAdAuditor faces inherent limitations typical of dynamic testing, such as limited testing coverage and the inability to automate registration and login processes, potentially underrepresenting the prevalence of ads in apps. However, recent advancements in large language models show promise in mitigating these constraints. Future work could explore integrating these models, as demonstrated in recent applications [65, 83], to enhance iAdAuditor's ad harvesting capabilities.

Ads served within the app can vary based on numerous factors, including time of access, user location, and network environment. Future work could leverage iAdAuditor to perform multiple iterations under diverse conditions, offering a more comprehensive view of ad behavior variability.

While our study focused on two representative regions, a truly global understanding of abusive ad practices necessitates broader geographic coverage. Future work should extend this research to more regions, enabling deeper insights into how cultural, economic, and regulatory factors influence abusive ad practices.

6 Related Work

Mobile Ad Auditing. Our study represents the first investigation into abusive ad practices within iOS apps. While this specific domain remains unexplored, extensive research has been conducted on similar practices in Android apps, providing valuable context and methodological insights for our work.

Several seminal studies [50, 64, 67, 71, 94] have conducted large-scale ad auditing in the Android ecosystem, examining a range of mobile advertising security threats. These investigations have uncovered various malicious practices, including click fraud, scam campaigns, malware propagation, and age appropriateness, highlighting the pervasive nature of ad-related threats in mobile platforms. Notably, a recent study [67] uncovered a novel threat vector: malware distribution through in-app ads that promote malicious apps on Google Play. While this threat vector was not investigated in our current study, similar risks could exist in the iOS ecosystem, warranting future investigation.

Furthermore, a number of studies [52, 55, 63, 87, 92, 93] have focused on comprehending and detecting aggressive, deceptive, or intrusive advertising practices in Android apps. For instance, Cheng et al. [52] examined deceptive ads that exploit the Chinese tradition of "red envelope" using CV-based techniques.

While these Android-focused studies have provided invaluable insights and inspired our approach, our work complements this existing body of knowledge by extending the investigation to the iOS ecosystem. By doing so, we not only fill a critical gap but also enable comparative analyses between the two dominant mobile platforms, potentially uncovering platform-specific vulnerabilities and informing more comprehensive mobile ad security strategies.

7 Conclusion

This study provides the first comprehensive analysis of abusive ad practices in iOS apps, revealing significant challenges across various categories. Our findings demonstrate that despite Apple's stringent controls, iOS apps are not immune to inappropriate content and deceptive tactics in in-app advertising. By developing iAdAuditor and conducting a large-scale analysis, we offer valuable insights and tools for the ecosystem stakeholders to better identify, measure, and mitigate these issues. Our work lays the foundation for improved ad regulation and user protection in the iOS ecosystem, contributing to a more responsible and secure mobile app environment.

Acknowledgment

This work was partially supported by the National Natural Science Foundation of China (grant No.62072046) and the Key R&D Program of Hubei Province (2023BAB017, 2023BAB079).

References

[1] 2018. Developer Policy Center: Monetization and Ads. https://play.google.com/about/monetization-ads/.
[2] 2020. CPM and CPC: Understanding Mobile Advertising Rates - Moloco. https://www.moloco.com/blog/cpm-and-cpc.
[3] 2021. Inappropriate Ads | Apple Community. https://discussions.apple.com/thread/253220407.
[4] 2022. Disturbing Adult Content Ads | Apple Community. https://discussions.apple.com/thread/253799147.
[5] 2022. How do i block sexually explicit ads from playing in games that are rated safe for children? | Apple Community. https://discussions.apple.com/thread/254383270.
[6] 2022. Inappropriate ads in the kids apps | Apple Community. https://discussions.apple.com/thread/253534067.
[7] 2022. PORNOGRAPHIC ADS FOR KIDS GAMES | Apple Community. https://discussions.apple.com/thread/253581261.
[8] 2024. App market global revenues by segment | Statista. https://www.statista.com/forecasts/1324000/global-app-market-revenue-monetization-segment.
[9] 2024. App Privacy Details - App Store - Apple Developer. https://developer.apple.com/app-store/app-privacy-details/.
[10] 2024. App Review Guidelines - Apple Developer. https://developer.apple.com/app-store/review/guidelines/.
[11] 2024. App-Store: Baby Car Games - A Complete Collection of Truck and Bus Games. https://apps.apple.com/cn/app/id1197200768.
[12] 2024. appium/appium-ios-device: Tools for interacting with iOS devices. https://github.com/appium/appium-ios-device.
[13] 2024. AppStore iPhone Free Apps Real-time Ranking - Qimai Data. https://www.qimai.cn/rank/index/brand/free/device/iphone/country/us/genre/6061.
[14] 2024. BaiduMobAdSDK. https://union.baidu.com/docs/sdk/iOSSDK.html.
[15] 2024. CocoaPods.org. https://cocoapods.org/.
[16] 2024. Detect explicit content (SafeSearch) - Cloud Vision API - Google Cloud. https://cloud.google.com/vision/docs/detecting-safe-search.
[17] 2024. Distribution of free and paid iOS apps in 2024 - Statista. https://www.statista.com/statistics/1020996/distribution-of-free-and-paid-ios-apps/.
[18] 2024. Financial Services - Play Console Help. https://support.google.com/googleplay/android-developer/answer/9876821?hl=en&ref_topic=9877466.
[19] 2024. Frida: A world-class dynamic instrumentation for developers, reverse-engineers, and security researchers. https://frida.re/.
[20] 2024. Get Started | iOS | Goggle Admob. https://developers.google.com/admob/ios/quick-start.
[21] 2024. GitHub - AFNetworking/AFNetworking: A delightful networking framework for iOS, macOS, watchOS, and tvOS. https://github.com/AFNetworking/AFNetworking.
[22] 2024. GitHub - Alamofire/Alamofire: Elegant HTTP Networking in Swift. https://github.com/Alamofire/Alamofire.
[23] 2024. GitHub - nygard/class-dump: Generate Objective-C headers from Mach-O files. https://github.com/nygard/class-dump.
[24] 2024. GitHub - theos/theos: A cross-platform suite of tools for building and deploying software for iOS and other platforms. https://github.com/theos/theos.
[25] 2024. Google AdMob - Earn More With Mobile App Monetization. https://admob.google.com/.
[26] 2024. GPT-4o | OpenAI. https://openai.com/index/hello-gpt-4o/.
[27] 2024. How can you monetize your app or game in China? | AppInChina. https://appinchina.co/services/monetization/ad-monetization/.
[28] 2024. IADAUDITOR/ViolatingVideoAdInKidsApp.mp4 at main · IADAUDIT/IADAUDITOR. https://github.com/IADAUDIT/IADAUDITOR/blob/main/ViolatingVideoAdInKidsApp.mp4.
[29] 2024. In-App Advertising - Worldwide - Statista. https://www.statista.com/outlook/amo/advertising/in-app-advertising/worldwide.
[30] 2024. Integration | Pangle SDK. https://www.pangleglobal.com/integration/integrate-pangle-sdk-for-android.
[31] 2024. Introduction to real-time bidding (RTB) - Authorized Buyers Help - Google Ads. https://support.google.com/authorizedbuyers/answer/6136272.
[32] 2024. iOS jailbreaking - Wikipedia. https://en.wikipedia.org/wiki/IOS_jailbreaking.
[33] 2024. Market share of mobile operating systems worldwide from 2009 to 2024 - Statista. https://www.statista.com/statistics/272698/global-market-share-held-by-mobile-operating-systems-since-2009/.
[34] 2024. mitmproxy - an interactive HTTPS proxy. https://mitmproxy.org/.
[35] 2024. Mobile and CTV App Intelligence | 42matters. https://42matters.com/.
[36] 2024. My City : Jail House on the App Store. https://apps.apple.com/us/app/my-city-jail-house/id1513523639.
[37] 2024. nm (Unix) - Wikipedia. https://en.wikipedia.org/wiki/Nm_(Unix).
[38] 2024. NSURLSession | Apple Developer Documentation. https://developer.apple.com/documentation/foundation/nsurlsession.
[39] 2024. Number of apps available in leading app stores in 2024 - Statista. https://www.statista.com/statistics/276623/number-of-apps-available-in-leading-app-stores/.
[40] 2024. NyaMisty/ipatool-py: IPATool-py: download ipa easily. https://github.com/NyaMisty/ipatool-py.
[41] 2024. Overview of Dynamic Libraries - Apple Developer. https://developer.apple.com/library/archive/documentation/DeveloperTools/Conceptual/DynamicLibraries/100-Articles/OverviewOfDynamicLibraries.html.
[42] 2024. Restricted Content - Play Console Help. https://support.google.com/googleplay/android-developer/topic/9877466?hl=en.
[43] 2024. Security of runtime process in iOS, iPadOS, and visionOS - Assistance Apple (FR). https://support.apple.com/fr-fr/guide/security/sec15bfe098e/web.
[44] 2024. Set the maximum ad content rating for an app or account - Google AdMob Help. https://support.google.com/admob/answer/7562142?hl=en.
[45] 2024. Setting Ad Content Ratings for Underaged Users. https://www.csjplatform.com/supportcenter/28178.
[46] 2024. Tencent GDTMobSDK. https://developers.adnet.qq.com/doc/ios/guide.
[47] 2024. Top 20 Ad Networks SDKs Used in iOS Apps on the Apple App Store - 42Matters. https://42matters.com/sdk-analysis/app-store/top-ad-networks-sdks.
[48] 2024. UIKit | Apple Developer Documentation. https://developer.apple.com/documentation/uikit/.
[49] 2024. VirusTotal. https://www.virustotal.com/.
[50] Gong Chen, Wei Meng, and John Copeland. 2019. Revisiting Mobile Advertising Threats with MAdLife. In *The World Wide Web Conference*. ACM, 207–217.
[51] Ying Chen, Sencun Zhu, Heng Xu, and Yilu Zhou. 2013. Children's exposure to mobile in-app advertising: an analysis of content appropriateness. In *2013 International Conference on Social Computing*. IEEE, 196–203.
[52] Yu Cheng, Xiaofang Qi, Yanhui Li, and Yumeng Wang. 2024. ReckDroid: Detecting red packet fraud in Android apps. *Computers & Security* (2024), 104117.
[53] Euijin Choo, Mohamed Nabeel, Doowon Kim, Ravindu De Silva, Ting Yu, and Issa Khalil. 2024. A large scale study and classification of virustotal reports on phishing and malware urls. *ACM SIGMETRICS Performance Evaluation Review* 52, 1 (2024), 55–56.
[54] Federal Trade Commission. 2020. Children's Privacy. https://www.ftc.gov/tips-advice/business-center/privacy-and-security/children's-privacy.
[55] Feng Dong, Haoyu Wang, Li Li, Yao Guo, Tegawendé F Bissyandé, Tianming Liu, Guoai Xu, and Jacques Klein. 2018. FraudDroid: Automated Ad Fraud Detection for Android Apps. In *The 26th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE 2018)*.
[56] General Data Protection Regulation (GDPR). 2022. General Data Protection Regulation (GDPR) Compliance Guidelines. https://gdpr.eu/.
[57] Ziyao He, Syed Fatiul Huq, and Sam Malek. 2024. " I tend to view ads almost like a pestilence": On the Accessibility Implications of Mobile Ads for Blind Users. In *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*. 1–13.
[58] Xinyi Hou, Yanjie Zhao, Yue Liu, Zhou Yang, Kailong Wang, Li Li, Xiapu Luo, David Lo, John Grundy, and Haoyu Wang. 2023. Large language models for software engineering: A systematic literature review. *arXiv preprint arXiv:2308.10620* (2023).
[59] Boyang Hu, Qicheng Lin, Yao Zheng, Qiben Yan, Matthew Troglia, and Qingyang Wang. 2019. Characterizing location-based mobile tracking in mobile ad networks. In *2019 IEEE Conference on Communications and Network Security (CNS)*. IEEE, 223–231.
[60] Johannes Huebner, Andrea Girardello, Olivia Sliz, Elgar Fleisch, and Alexander Ilic. 2020. What people focus on when reviewing your app-an analysis across app categories. *IEEE Software* 38, 3 (2020), 96–105.
[61] Ling Jin, Boyuan He, Guangyao Weng, Haitao Xu, Yan Chen, and Guanyu Guo. 2019. Madlens: Investigating into android in-app ad practice at api granularity. *IEEE Transactions on Mobile Computing* 20, 3 (2019), 1138–1155.
[62] Bangyan Ju, Jin Yang, Tingting Yu, Tamerlan Abdullayev, Yuanyuan Wu, Dingbang Wang, and Yu Zhao. 2024. A Study of Using Multimodal LLMs for Non-Crash Functional Bug Detection in Android Apps. *arXiv preprint arXiv:2407.19053* (2024).
[63] Tianming Liu, Haoyu Wang, Li Li, Guangdong Bai, Yao Guo, and Guoai Xu. 2019. DaPanda: Detecting Aggressive Push Notifications in Android Apps. In *2019 34th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 66–78.
[64] Tianming Liu, Haoyu Wang, Li Li, Xiapu Luo, Feng Dong, Yao Guo, Liu Wang, Tegawendé Bissyandé, and Jacques Klein. 2020. Maddroid: Characterizing and detecting devious ad contents for android apps. In *Proceedings of The Web Conference 2020*. 1715–1726.
[65] Zhe Liu, Chunyang Chen, Junjie Wang, Mengzhuo Chen, Boyu Wu, Xing Che, Dandan Wang, and Qing Wang. 2024. Make llm a testing expert: Bringing human-like interaction to mobile gui testing via functionality-aware decisions. In *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*. 1–13.
[66] Zhe Liu, Cheng Li, Chunyang Chen, Junjie Wang, Boyu Wu, Yawen Wang, Jun Hu, and Qing Wang. 2024. Vision-driven Automated Mobile GUI Testing via Multimodal Large Language Model. *arXiv preprint arXiv:2407.03037* (2024).

[67] Shang Ma, Chaoran Chen, Shao Yang, Shifu Hou, Toby Jia-Jun Li, Xusheng Xiao, Tao Xie, and Yanfang Ye. 2024. Careful About What App Promotion Ads Recommend! Detecting and Explaining Malware Promotion via App Promotion Graph. *arXiv preprint arXiv:2410.07588* (2024).

[68] Lakhdar Meftah, Romain Rouvoy, and Isabelle Chrisment. 2019. Testing nearby peer-to-peer mobile apps at large. In *2019 IEEE/ACM 6th International Conference on Mobile Software Engineering and Systems (MOBILESoft)*. IEEE, 1–11.

[69] Martin F Mendiola, Miriam Kalnicki, and Sarah Lindenauer. 2015. Valuable features in mobile health apps for patients and consumers: content analysis of apps and user ratings. *JMIR mHealth and uHealth* 3, 2 (2015), e4283.

[70] Brian Pia. 2017. Sexually suggestive ads appearing on children's apps. https://abc3340.com/archive/sexually-suggestive-ads-appearing-on-childrens-apps.

[71] Vaibhav Rastogi, Rui Shao, Yan Chen, Xiang Pan, Shihong Zou, and Ryan D Riley. 2016. Are these Ads Safe: Detecting Hidden Attacks through the Mobile App-Web Interfaces.. In *NDSS*.

[72] Abbas Razaghpanah, Rishab Nithyanand, Narseo Vallina-Rodriguez, Srikanth Sundaresan, Mark Allman, Christian Kreibich, Phillipa Gill, et al. 2018. Apps, trackers, privacy, and regulators: A global study of the mobile tracking ecosystem. In *The 25th annual network and distributed system security symposium (NDSS 2018)*.

[73] Elena Root and Bogdan Melnykov. 2018. Malware Displaying Porn Ads Discovered in Game Apps on Google Play. https://research.checkpoint.com/malware-displaying-porn-ads-discovered-in-game-apps-on-google-play/.

[74] Aleieldin Salem, Sebastian Banescu, and Alexander Pretschner. 2021. Maat: Automatically analyzing virustotal for accurate labeling and effective malware detection. *ACM Transactions on Privacy and Security (TOPS)* 24, 4 (2021), 1–35.

[75] Rui Shao, Vaibhav Rastogi, Yan Chen, Xiang Pan, Guanyu Guo, Shihong Zou, and Ryan Riley. 2018. Understanding in-app ads and detecting hidden attacks through the mobile app-web interface. *IEEE Transactions on Mobile Computing* 17, 11 (2018), 2675–2688.

[76] Mikey Smith. 2015. Porn advert shown in children's smartphone game to kids as young as five. https://www.mirror.co.uk/news/uk-news/porn-advert-shown-childrens-smartphone-5855008.

[77] Sooel Son, Daehyeok Kim, and Vitaly Shmatikov. 2016. What Mobile Ads Know About Mobile Users.. In *Ndss*.

[78] Nico Steffen, Peter Kroon, Faisal Aman Abbasi, and Lukas Wiewiorra. 2023. *Access charges in software-based termination monopolies*. Technical Report. WIK Diskussionsbeitrag.

[79] Ruoxi Sun, Minhui Xue, Gareth Tyson, Shuo Wang, Seyit Camtepe, and Surya Nepal. 2023. Not seen, not heard in the digital world! measuring privacy practices in children's apps. In *Proceedings of the ACM Web Conference 2023*. 2166–2177.

[80] Youshuai Tan, Jinfu Chen, Weiyi Shang, Tao Zhang, Sen Fang, Xiapu Luo, Zijie Chen, and Shuhao Qi. 2023. STRE: An Automated Approach to Suggesting App Developers When to Stop Reading Reviews. *IEEE Transactions on Software Engineering* 49, 8 (2023), 4135–4151.

[81] Ash Turner. 2024. How many smartphones are in the world? https://www.bankmycell.com/blog/how-many-phones-are-in-the-world.

[82] Yaqing Wang, Quanming Yao, James T Kwok, and Lionel M Ni. 2020. Generalizing from a few examples: A survey on few-shot learning. *ACM computing surveys (csur)* 53, 3 (2020), 1–34.

[83] Hao Wen, Yuanchun Li, Guohong Liu, Shanhui Zhao, Tao Yu, Toby Jia-Jun Li, Shiqi Jiang, Yunhao Liu, Yaqin Zhang, and Yunxin Liu. 2024. Autodroid: Llm-powered task automation in android. In *Proceedings of the 30th Annual International Conference on Mobile Computing and Networking*. 543–557.

[84] Peihan Wen and Mo Chen. 2020. A new analysis method for user reviews of mobile fitness apps. In *Human-Computer Interaction. Human Values and Quality of Life: Thematic Area, HCI 2020, Held as Part of the 22nd International Conference, HCII 2020, Copenhagen, Denmark, July 19–24, 2020, Proceedings, Part III 22*. Springer, 188–199.

[85] Huayao Wu, Wenjun Deng, Xintao Niu, and Changhai Nie. 2021. Identifying key features from app user reviews. In *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*. IEEE, 922–932.

[86] Shuohan Wu, Jianfeng Li, Hao Zhou, Yongsheng Fang, Kaifa Zhao, Haoyu Wang, Chenxiong Qian, and Xiapu Luo. 2023. CydiOS: A Model-Based Testing Framework for iOS Apps. In *Proceedings of the 32nd ACM SIGSOFT International Symposium on Software Testing and Analysis*. 1–13.

[87] Guosheng Xu, Yangyu Hu, Qian Guo, Ren He, Li Li, Guoai Xu, Zhihui Han, and Haoyu Wang. 2020. Dissecting mobile offerwall advertisements: An explorative study. In *2020 IEEE 20th International Conference on Software Quality, Reliability and Security (QRS)*. IEEE, 518–526.

[88] HanXiang Xu, ShenAo Wang, Ningke Li, Yanjie Zhao, Kai Chen, Kailong Wang, Yang Liu, Ting Yu, and HaoYu Wang. 2024. Large language models for cyber security: A systematic literature review. *arXiv preprint arXiv:2405.04760* (2024).

[89] An Yan, Zhengyuan Yang, Wanrong Zhu, Kevin Lin, Linjie Li, Jianfeng Wang, Jianwei Yang, Yiwu Zhong, Julian McAuley, Jianfeng Gao, et al. 2023. Gpt-4v in wonderland: Large multimodal models for zero-shot smartphone gui navigation. *arXiv preprint arXiv:2311.07562* (2023).

[90] Zhao Yang, Jiaxuan Liu, Yucheng Han, Xin Chen, Zebiao Huang, Bin Fu, and Gang Yu. 2023. Appagent: Multimodal agents as smartphone users. *arXiv preprint arXiv:2312.13771* (2023).

[91] Jia-Yu Yao, Kun-Peng Ning, Zhen-Hui Liu, Mu-Nan Ning, and Li Yuan. 2023. Llm lies: Hallucinations are not bugs, but features as adversarial examples. *arXiv preprint arXiv:2310.01469* (2023).

[92] Shuangmin Zhang, Ruixuan Li, Junwei Tang, and Xiwu Gu. 2020. Automated Rogue Behavior Detection for Android Applications.. In *SEKE*. 550–553.

[93] Zhen Zhang, Yu Feng, Michael D Ernst, Sebastian Porst, and Isil Dillig. 2021. Checking conformance of applications against GUI policies. In *Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 95–106.

[94] Yanjie Zhao, Tianming Liu, Haoyu Wang, Yepang Liu, John Grundy, and Li Li. 2023. Are Mobile Advertisements in Compliance with App's Age Group?. In *Proceedings of the ACM Web Conference 2023*. 3132–3141.

[95] Nan Zhong and Florian Michahelles. 2013. Google Play is not a long tail market: An empirical analysis of app adoption on the Google Play app market. In *Proceedings of the 28th annual ACM symposium on applied computing*. 499–504.

[96] Ziyi Zhou, Xing Han, Zeyuan Chen, Yuhong Nan, Juanru Li, and Dawu Gu. 2022. Simulation: demystifying (insecure) cellular network based one-tap authentication services. In *2022 52nd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. IEEE, 534–546.

[97] Onur Zungur, Gianluca Stringhini, and Manuel Egele. 2020. Libspector: Context-aware large-scale network traffic analysis of android applications. In *2020 50th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. IEEE, 318–330.