Are Mobile Advertisements in Compliance with App's Age Group?

Yanjie Zhao*
Faculty of Information Technology, Monash University
Australia

Tianming Liu*
Faculty of Information Technology, Monash University
Australia

Haoyu Wang
Huazhong University of Science and Technology
China

Yepang Liu
Southern University of Science and Technology
China

John Grundy
Faculty of Information Technology, Monash University
Australia

Li Li†
School of Software, Beihang University
China

ABSTRACT

As smartphones and mobile apps permeate every aspect of people's lives, children are accessing mobile devices at an increasingly younger age. The inescapable exposure of advertisements in mobile apps to children has grown alarmingly. Mobile advertisements are placed by advertisers and subsequently distributed by ad SDKs, under the rare control of app developers and app markets' content ratings. Indeed, content that is objectionable and harmful to children's mental health has been reported to appear in advertising, such as pornography. However, few studies have yet concentrated on automatically and comprehensively identifying such kid-unsuitable mobile advertising. In this paper, we first characterize the regulations for mobile ads relating to children. We then propose our novel automated dynamic analysis framework, named *AdRambler*, that attempts to collect ad content throughout the lifespan of mobile ads and identify their inappropriateness for child app users. Using *AdRambler*, we conduct a large-scale (25,000 mobile apps) empirical investigation and reveal the non-incidental presence of inappropriate ads in apps with child-included target audiences. We collected 11,270 ad views and identified 1,289 ad violations (from 775 apps) of child user regulations, with roughly half of the app promotions not in compliance with host apps' content ratings. Our finding indicates that even certified ad SDKs could still propagate inappropriate advertisements. We further delve into the question of accountability for the presence of inappropriate advertising and provide concrete suggestions for all stakeholders to take action for the benefit of children.

CCS CONCEPTS

• **Security and privacy** → **Software and application security**; • **Information systems** → **Online advertising**; • **Human-centered computing** → **Ubiquitous and mobile computing**.

KEYWORDS

Mobile advertising, Android app, Malvertising

ACM Reference Format:
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 (WWW '23), May 1–5, 2023, Austin, TX, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3543507.3583534

*Co-first authors who contributed equally to this work.
†Corresponding author (lilicoding@ieee.org).

1 INTRODUCTION

Between 2017 and 2022, there has been a massive boom in smartphone users, with a 49.89% increase in the number of people owning a smart cell phone. Statistically, the number of smartphone users globally as of 2022 is 6.6 billion, corresponding to 83.37% of the world's population using a smartphone [39]. As the mobile industry develops, mobile devices have penetrated every aspect of people's lives, with no exception for children. According to a survey by the Pew Research Center [8], 60% of parents with a kid under the age of 12 indicate that their child has ever used or interacted with a smartphone, and 31% in that group are exposed to smartphones before age 5. Thus an increasing number of children are using smartphones at an increasingly younger age.

Given that mobile devices have gained such popularity among children, the public and the research community have commenced raising awareness about children being exposed to inappropriate information in recent years, especially after the notorious Elsagate [42] controversy emerged in late 2017. Thousands of videos are found on YouTube Kids and other video platforms that portray themselves as "child-friendly", featured by beloved cartoon characters such as "Elsa" from Disney animated film Frozen [43]. However, these videos involve inappropriate themes for children, such as sexual situations, graphic violence, fetishes, and drugs.

Mobile advertising fuels the economy of much of the mobile app ecosystem. According to traditional estimates, children begin to comprehend the persuasive intent of advertising around the age of 8 [15]. As advertisements (hereinafter referred to as "ad" or "ads") in mobile apps are generally served by third-party ad networks, the developers and app markets are in less control over the ads presented within the apps than the actual content of the apps. The inappropriateness of advertising within apps with child-friendly content ratings hence deserves significant attention. Indeed, the appearance of inappropriate ads with sexually-suggestive content or even pornography in children's apps from Google Play has been recurrently reported over the years [29, 36, 37]. Yet despite so many complaints and Google Play's phased update on child protection policies [4], inappropriate advertising still continues to appear. Chen et al. [10] performed a manual analysis on the inappropriateness of the in-app advertising in 405 mobile apps designed for children in 2013. Follow-up researches [9, 27] have done some preliminary examinations and analyses but are limited in scope. Few studies to date have concentrated on automatically detecting children's inappropriate exposure to mobile app advertising.

(a) App *Lion Roar Sound* (b) Screenshots of ad pages shown in App *Lion Roar Sound* (c) App *Bumble*

Figure 1: Motivating Example.

To fill this gap, we propose a prototype framework named *AdRambler* for automatically and precisely clicking the ads and collecting ad-related data, primarily based on dynamic analysis techniques. *AdRambler* aids in automatically triggering numerous ads on a vast number of apps whose target audience includes children to promote further analysis regarding whether these apps contain inappropriate ads. Our evaluation results demonstrate that *AdRambler* can detect and record ads precisely, and that there are, unfortunately, plenty of mobile app ads that violate the relevant policies for children's protection. We further discuss who share the blame for the presence of inappropriate advertising and provide tangible recommendations for all relevant parties to take action in this matter.

We make the following key contributions in this work:

- We propose a prototype tool, named *AdRambler*[1], for automatic dynamic analysis of Android apps to precisely click the ads and collect ad-related data required for further identification of ad inappropriateness;
- We conduct a large-scale empirical study in the wild with *AdRambler* to capture and record app ads and identify the presence of violations for advertising in realistic apps. By applying *AdRambler* to 25,000 Google Play apps, we collected 11,270 ad views and found that 775 apps delivered 1,289 ad violations for child users; and
- we delve into the question of accountability for the presence of inappropriate advertising and offer concrete suggestions for all stakeholders to take action for children's sake.

2 BACKGROUND

2.1 Motivation

In our preliminary research, we have noticed that many apps with child-friendly content ratings contain inappropriate ads. As a concrete example, consider Figure 1, which displays the triggering process in an ad included in the App *Lion Roar Sound*, which provides lion roar sounds for users. It is easy to observe from Figure 1(a) that the app's audiences are `EVERYONE`, of which children are not excluded. When we launch the app, however, it pops up a video-form ad, as shown in Figure 1(b). This ad eventually induces the user to download a dating app suitable for ages 17 and up, i.e., App *Bumble* illustrated in Figure 1(c), which is an obvious violation of user guideline regulations.

This motivating example suggests that apps that appear to be child-friendly on the surface could be potentially harmful, e.g., by directing children to premature exposure to inappropriate content with ads as a carrier. Therefore, we wanted to investigate relevant policies and detect such violating ads to warn and inspire the industry and research community.

2.2 Ad Inappropriateness for Children

Multiple laws and regulations are issued in different countries to safeguard children from the harm caused by the inappropriateness of mobile apps, such as the US Children's Online Privacy and Protection Act *(COPPA)* [11] and the EU General Data Protection Regulation *(GDPR)* [18]. In compliance with these regulations, Google Play has assigned each app to an age-appropriate group. The content ratings and descriptions of age groups vary in different regions[2] to comply with the local regulations [31]. For instance, the age appropriate groups in the North & South America are divided into `EVERYONE`, `EVERYONE 10+`, `TEEN` (13+), `MATURE` (17+), and `ADULTS ONLY` (18+). With the assigned age group, the app market could better maintain the age appropriateness of each app and reduce children's exposure to improper apps.

It has been previously mentioned that the app market has difficulty regulating in-app ads as they are provided by third-party components. As such, Google Play has imposed a series of requirements on in-app advertising that developers must capitulate to if the target audience of their apps includes children [20]. The requirements can be summarized into two categories:

Ad SDK. For apps whose target audiences include only children and who serve ads using an ad SDK, the in-app ads can only be served by Google Play-certified ad SDKs [22], which currently include 11 well-known SDKs like Google AdMob, as listed in Table 1. These self-certified ad SDKs claim to rate their served ads according to age-appropriate groups and allow developers to request child-appropriate ads on a per-request or per-app basis. For apps targeting both children and older users, the developer must implement age

[1] *AdRambler* can be accessed via https://github.com/Tmliu06/AdRambler.

[2] Although different, they all classify people under the age of 12 or 13 as children, following local regulations [2, 12, 17]. Unless stated otherwise, we refer to children as anyone under 12 in this paper.

screening measures to avoid serving ads to children from non-certified SDKs [3].

Table 1: The Google Play-certified Ad SDKs.

Name	Example Package Name	Name	Example Package Name
Google AdMob	com.google.android.gms.ads	AdColony	com.adcolony
Google Ad Manager		AppLovin	com.applovin
Chartboost	com.chartboost	InMobi	com.inmobi
ironSource	com.ironsource	Kidoz	net.kidoz.sdk
SuperAwesome	tv.superawesome.sdk	Unity Ads	com.unity3d.services.ads
Vungle	com.vungle	**Total**	**11**

Ad Content. Several types of explicit content are regarded as inappropriate [21]. They should not be displayed to children in ads as stated by Google Play, including *pornography and sexually suggestive content*, *simulated or real gambling* (even if free to enter), *dating or adult relationships*, *controlled or harmful substances* (e.g., alcoholic beverages and tobacco products), *violent content*, and any other inappropriate media content. Specifically, *promoting inappropriate downloadable software* (e.g., apps) to children is also disallowed.

In order to adequately detect these two categories of violations in mobile advertising, it is necessary to first capture and record as much ad-related content within apps as possible. Especially when dealing with large-scale in-the-wild apps, it is essential to automate the ad sniffing, triggering, and gathering.

3 OUR APPROACH: *ADRAMBLER*

This research aims to characterize and detect inappropriate ads in mobile apps whose audiences include children. We expect to expose the current presence of in-app advertising violations toward child protection policies. To this end, we propose an automatic framework called *AdRambler* to explore the apps and harvest ad-related information for further investigations. Figure 2 demonstrates the workflow of our framework, which is composed of three modules: (1) **AdBot** uses app automating techniques to traverse an app's GUI pages and interact with in-app ads. To precisely identify and click the ad views during the exploration, AdBot captures the instantiating class for each GUI widget at runtime to map the GUI widgets with ad SDKs. (2) **AdTraffic** is responsible for the identification and collection of all the ad-related traffic during the exploration, which runs in parallel with AdBot. Taking advantage of the Socket Hooking approach, AdTraffic can recognize ad traffic that originates from ad SDKs. (3) **AdGuard** identifies any inappropriateness within the in-app ads based on pre-defined rules. Utilizing the recorded ad SDKs from AdBot and the ad-related content from AdTraffic, AdGuard can comprehensively characterize and inspect the ads' inappropriateness towards children within apps.

3.1 Automated Exploration of In-app Ads

Serving as the first step of *AdRambler*, **AdBot** is designed as a dynamic exploration tool to interact with the in-app ads automatically. To fully reveal the inappropriateness within in-app ads, AdBot has to lead the app to go through the whole lifespan of in-app ads, as shown in Figure 3, which includes both ad loading and ad clicking. Hence, given an Android app as input, AdBot must (1) traverse the app's GUI pages to trigger the ad loading, and (2) simulate click actions upon the ad views once they are loaded.

Traversing the GUI Pages for Ad Loading. AdBot needs to simulate user interactions with GUI widgets to achieve automation on app traversing. It thus takes advantage of Android Accessibility Services [1] to understand the layout of GUI pages, where a view

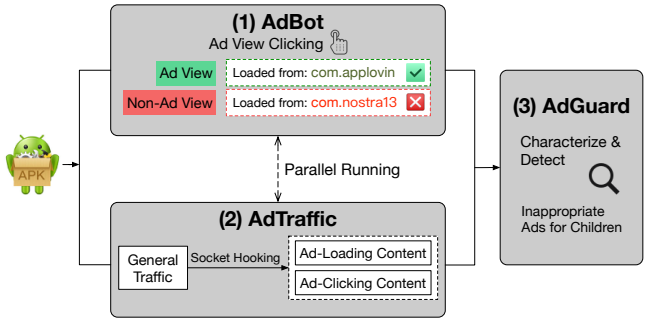

Figure 2: The overview of *AdRambler*, where "com.applovin" is the package name of an example ad SDK.

tree is obtained for each GUI page, containing the corresponding GUI information of each view within the page. The GUI information includes a wide range of attributes of this view, such as classname, coordinates, and whether this view is clickable. AdBot then generates test inputs based on the obtained view tree to automatically explore the app.

As *AdRambler* is designed to support large-scale analysis, it is impractical for AdBot to explore all the GUI pages within the app. This means that AdBot needs to trigger as many ads as possible within limited exploring steps. Inspired by an observation from a recent study [24] that most ads are located within the first activity and the second activity of Android apps, Adbot implements a depth-first search (DFS) exploration strategy with a limited number of inputs per app to prioritize these activities. In this way, AdBot achieves the efficiency needed for large-scale analysis, while avoiding diving too deep into the apps where ads are not located.

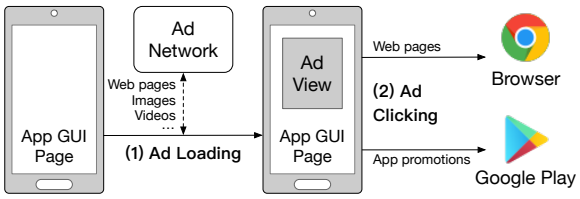

Figure 3: The lifespan of an in-app ad.

Simulation of Click Actions upon Ad Views. For each GUI page, AdBot precisely clicks the ads by locating ad views. It is critical for the ad views to be clicked once loaded; otherwise, the automated exploration could lead the app to a different GUI page with no ad view. However, it is non-trivial to identify ad views among all GUI widgets during the exploration. The common practice adopted by previous approaches [9, 14, 26] is to apply a set of heuristic rules based on certain ad-view features such as the classname or placement, which often results in a number of redundant clicks. Moreover, applying heuristic rules could also result in the omission of ad views, leading to false negatives in the inappropriateness detection. To ensure the comprehensiveness of our testing and to improve efficiency, we thus propose a novel ad view identification approach that utilizes framework hooking to link the ad view with its ad SDK.

Identification of Ad Views. The in-app ads are distributed by third-party ad SDKs embedded in the app. Different from normal non-ad views that are implemented by app developers or UI libraries [7], ad views are instantiated by specific `View` classes within

the ad SDKs [16] (e.g., *com.applovin.adview.AppLovinAdView*, where *com.applovin* is the package name of an ad SDK). Therefore, by retrieving the class that instantiates the view, we can easily determine whether a view is an ad view. The implementation of our identification approach is detailed below. Recall that AdBot utilizes Accessibility Services to obtain the layout of each GUI page, we thus leverage framework hooking on Android Accessibility APIs, including *View.onInitializeAccessibilityNodeInfo()* and *View.createAccessibilityNodeInfo()*, to obtain the real class that instantiates the view. We then use a whitelist-based approach to identify ad views with an ad SDK list from a most recent work [5], which contains a total of 64 well-known ad SDKs and their corresponding package names. If a view's instantiating class comes from an ad SDK's package, we recognize the view as an ad view from that SDK and configure AdBot's next input to click the ad view.

3.2 Identifying Ad-related Traffic

The second module of *AdRambler*, **AdTraffic**, is designed to harvest and identify all the ad-related network traffic at runtime, which runs in parallel with AdBot. To handle and decrypt HTTPS messages, AdTraffic implements a Man-in-The-Middle (MiTM) proxy service on the test device, where the proxy is configured to record the whole traffic from the test device in detail in a timely manner.

Connecting Ad Traffic with Ad SDKs. While an ad is being loaded and clicked during the exploration, network traffic unrelated to ads will also be captured inevitably (e.g., the app communicating with its own server). The key difference between ad traffic and non-ad traffic is that ad traffic is initiated by ad SDKs. We applied a'Socket Hooking' approach to help identify ad traffic by connecting ad traffic with ad SDKs. The key idea is that any network communication in Android will commence with a socket system call [47]. We leverage framework hooking on socket APIs to monitor the network behavior on Android OS following the approach of previous studies [46, 47]. When a socket connection is established, our approach will automatically record the target host and dump the stack trace with the *getStackTrace* method. We can then locate the package initiating this connection. Recall that we have already obtained the ad SDK of ad views in AdBot. If the connection comes from the package of the ad SDK, we regard the connection as ad traffic.

Collection of Ad Content. The revenue of app developers from advertisers is generally based on either the number of ads displayed or the number of ads clicked, which are the two main phases of in-app advertising. Both phases will introduce ad content to users, which are called *ad-loading content* and *ad-clicking content* [26]. As shown in Figure 3, there are certain differences between the two types of ad content. First off, unlike ad-loading content that is delivered within the app, ad-clicking content often involves third-party apps such as Google Play promotions and web pages in a browser. In addition, ad-loading content will always be delivered to users' devices even before the ad is displayed, while ad-clicking content may not be triggered unless the ad view is clicked. Since both types of ad content have been stated to contain inappropriate artifacts to children in previous news reports [36, 37], AdTraffic aims to gather both types of ad content.

For the ad-loading content, we mainly focus on media files used in ad displays, such as images, videos, and web pages that construct the ad views, to detect any explicit content that may be inappropriate for children. In terms of ad-clicking content, two types are taken into account: (1) *web pages*: Upon clicking the ad, a series of redirection links will eventually lead the mobile device to a page displaying promoted ad information in a browser, which is collected for further detection. (2) *app promotions*: The host app is redirected to a Google Play page of the promoted app. We record the package name of the promoted app to detect further if it is suitable for children.

3.3 Identification of Inappropriate Ads

Finally, the third module of *AdRambler*, **AdGuard**, identifies the inappropriateness for children in ads harvested from mobile apps, using the output of AdBot and AdTraffic modules. We elaborate on the detection approaches for the inappropriateness of *Ad SDK* and *Ad Content* introduced in Section 2.2.

Non-compliant Ad SDK Usage Detection. As previously mentioned, Google Play explicitly disallows serving in-app ads to children through uncertified ad SDKs. Unlike certified SDKs which to a certain extent perform content rating procedures for their served ads, there is no guarantee that uncertified ad SDKs would follow the same patterns, rendering their underage audiences more susceptible to inappropriate or harmful advertising. Therefore, whether for the purpose of finding the culprit behind unsanitized behaviors, or to avoid potential hazards for child users, the usage of uncertified ad networks needs to be revealed. Recall that in the AdBot module, we have already obtained the ad SDK for each ad view, we then check the ad SDK against the Google Play-certified ad SDKs in Table 1 to expose the behavior of displaying ads through uncertified ad SDKs.

Inappropriate Ad Content Identification. As mentioned in Section 3.2, we are interested in *ad-loading content* as well as *ad-clicking content*. The former needs to deal with media files such as the loading web pages, images (in JPEG, PNG, GIF, and WebP formats), and videos; for the latter, the clicking web pages, as well as the promoted apps, need to be considered. For the inspection of web pages in both phases, we resort to the Natural Language API[3] provided by Google Cloud. According to the description of inappropriate ad content in Section 2.2, we collate 29 inappropriateness-related content categories[4], such as */Games/Gambling*, */Online Communities/Dating & Personals*, etc. If a web page is determined to fall into one of these categories, its associated ad is considered harmful to children. For the inspection of images and videos, we apply Google Vision API[5] to recognize the inappropriateness based on: (1) the *Safe Search Detection* feature, provided for explicit content detection, will return the likelihood of an image belonging to an unsafe category, e.g., adult, violence, and racy, on a Likert scale (here, we only consider `VERY_LIKELY` and `LIKELY` as positive); and (2) the *Label Detection* feature, which is further used to mark other types of inappropriate ad images, such as alcohol and tobacco sales, gambling inducement, and other unsuitable content, as listed in Section 2.2. Moreover, since a video is a sequence composed of a large number of images, we randomly select frames from a video for inappropriate content identification and introduce human involvement for manual double-checking. For the inspection of app

[3] https://cloud.google.com/natural-language
[4] https://cloud.google.com/natural-language/docs/categories
[5] https://cloud.google.com/vision

promotions, we crawl Google Play based on the package name of the promoted app and assess whether the host app's advertising violates child protection regulations based on the promoted app's age group. If the age group suggests the promoted app is exclusively for teenagers or more mature groups, the promotion is considered inappropriate for children, as illustrated in Section 2.1.

4 EVALUATION

Our investigations explore the experimental results with *AdRambler* to answer the following research questions:

- **RQ1:** How are mobile ads presented in real-world Android apps, and can *AdRambler* effectively recognize them?
- **RQ2:** Are there any policy violations within the in-app ads from apps for children audiences?
- **RQ3:** Does the inappropriateness of mobile advertising vary by region?
- **RQ4:** How well does *AdRambler* perform in ad collection compared with existing approaches?

All of our experiments are conducted on four parallel-running Nexus 5 smartphones. Recall that we mentioned in Section 2.2 that for apps with audiences including children and older users, age screening measures by developers are mandated to avoid serving inappropriate ads to children. In accordance with *COPPA* [13], Google Play demands that the age screening must be implemented in a neutral manner, requiring users to enter their birthdays freely, instead of presetting the birth date to a required age or simply providing a dialog to ask if the user is above a certain age [3]. As *AdRambler* is configured not to send any text inputs to the app, the age screening (if any) must assume the user is underage when serving ads. In other words, any ads displayed in our experiments can potentially be received by children.

4.1 Dataset

We adopt the widely-used AndroZoo dataset [6] to crawl Android apps from the official Google Play store for the experiments. Since we are only interested in apps that might display ads to children, apps with age-appropriate groups of `EVERYONE` or `EVERYONE 10+` on Google Play are considered, as both groups include target audiences under 12 years old. Google Play marks each app as ad-containing or not; in this work, we only consider apps that contain ads in recent years. Hence, an app will be collected as a candidate only if it is (1) available on Google Play, (2) with an ad-containing tag marked by Google Play, (3) with an update time from 2018 to 2022, according to AndroZoo, and (4) is assigned with an age group that includes children. Eventually, we randomly selected 25,000 apps that met the above criteria to form our overall dataset.

4.2 RQ1: *AdRambler* Effectiveness

Our first research question concerns the effectiveness of our *AdRambler* framework for detecting ads in mobile apps. We conduct a large-scale study with *AdRambler* on our dataset composed of 25,000 in-the-wild Android apps. The investigation for this RQ is conducted in the Hong Kong network environment. Given that it takes time for in-app ads to be loaded from the ad networks, a time interval is set between the generated test inputs. Overall, an app is evaluated in 2 minutes on average.

As shown in Table 2, **the presence of mobile advertising is widespread, as 11,270 ad views are identified out of 8,971 apps in our investigation**. The median and mean numbers of ad views among all the 8,971 apps are 1 and 1.26. Most apps (7,031/8,971 = 78.37%) have one ad view, while 1,691 apps have two, and 249 apps have more than two.

Table 2: The presence of ads in Android apps.

Category	Type	# All	# Valid	# Kid-unsuitable / # App
Ad View	-	11,270	-	-
Ad SDK	-	12	-	7/936
Ad-loading Content	Web Pages	51,812	5,342	94/61
	Images	66,984	24,544	758/535
	Videos	733	690	177/117
Ad-clicking Content	Web Pages	53,461	12,187	115/55
	App Promotions	305	295	145/49
In Total	-	**173,295**	**43,058**	**1,289/775**

Pre-filtering is required for all recorded ad-related content as there are many invalids. We filter the invalid web pages with less than 100 characters, image files smaller than 2KB in size, corrupted video files, and app promotions that have been unattainable in Google Play. The statistics after filtering are shown in the fourth column of Table 2, where images are the most widely used advertising mediums. In the end, we collected 43,058 valid samples of various advertising.

Recall that we construct the dataset considering only apps labeled by Google Play as "Contains ads". However, as shown in Table 2, we can only collect ads from 8,971 apps out of the 25,000 apps in our dataset. The main reasons are: (1) some apps require complex human interactions that cannot be simulated through general automation to reach the GUI pages that display ads, e.g., they request users to register and sign in before ad-showing; (2) some apps are inherently corrupted and will crash upon startup or after running only a few steps. Nevertheless, to the best of our knowledge, we achieve the highest percentage (8,971/25,000 = 35.88%) of in-app ads collected in similar state-of-the-art works, for instance, 21.51% for Liu et al. [26], and 3.99% for Chen et al. [9] [6]

A total of 12 ad SDKs are used to display ads in these 8,971 apps, detailed in Table 3. Use of the **ad SDKs provided by Google, i.e., Google AdMob and Google Ad Manager, lead the others by a wide margin** with a percentage of 7,894/9,046 = 87.27%. The next three most frequently used ad SDKs are StartApp, AppBrain, and AppLovin.

Table 3: The presence of ad SDKs in Android apps.

Ad SDK	Package Name	Frequency	Google Play Certified
Google	com.google.android.gms.ads	7,894	✓
	com.google.android.ads	5	
StartApp	com.startapp	595	✗
AppBrain	com.appbrain	290	✗
AppLovin	com.applovin	159	✓
Unity Ads	com.unity3d.services.ads	24	✓
	com.unity3d.ads	4	
Facebook	com.facebook.ads	24	✗
InMobi	com.inmobi	22	✓
Appnext	com.appnext	19	✗
MoPub	com.mopub	5	✗
Smaato	com.smaato	2	✗
ironSource	com.ironsource	2	✓
Appodeal	com.appodeal	1	✗
Total	-	**9,046***	**✓ 5/12**

* One app may involve multiple ad SDKs.

[6] The dataset in Liu et al. [26] is constructed with adware; while Chen et al. [9] run their experiments on free Google Play apps without considering ad tag of any kind.

4.3 RQ2: Detection of Inappropriate Ads

For our second research question, we aim to investigate the potential inappropriateness of in-app ads shown to child mobile app users. The last column of Table 2 lists the violations existing in our large-scale investigation. We depict the proportional percentage of the numbers of improper advertising in stacked bars, shown in Figure 4. We can conclude that **app promotion is the type of ad content most likely to be inappropriate, with roughly 49.15%**, although it is not outstanding in quantity. In other words, approximately half of the promoted Google Play apps from host apps with child-included audiences are inappropriate for children. Video ad content also has a relatively higher percentage (25.65%) to be kid-unsuitable. Although the proportion of unsafe ad images is not high, the number of images with inappropriate content is the most among all types of ad content. In Section 2 we introduced

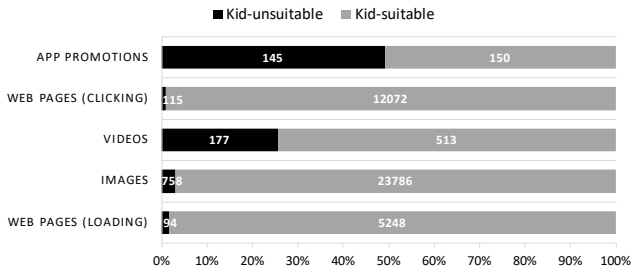

Figure 4: Proportion of various types of kid-unsuitable ad content.

six categories of inappropriate ad content for child mobile app users. The numbers of inappropriate ad contents found in these categories are displayed in Figure 5, where the distribution of different advertising carriers is totaled. We can conclude that most of the kid-unsuitable content in ad-loading web pages is affiliated with simulated or real gambling, while in ad-clicking web pages, the inappropriateness is mainly associated with controlled or harmful substances (especially alcohol selling). For the inappropriate ad images, most of them (582/758 = 76.78%) contain controlled or harmful substances, followed by pornography and sexually suggestive content (135/758 = 17.81%). The inappropriate ad videos mainly consist of dating & adult relationships (103/177 = 58.19%) and violence (41/177 = 23.16%), e.g., the promotion of live video streaming platforms limited to 12+, games that contain violence inducement, and other mature-only apps.

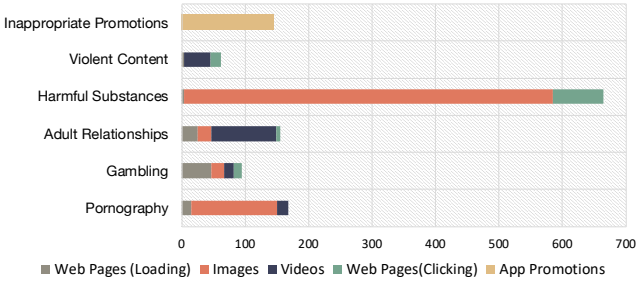

Figure 5: Distribution of categories of kid-unsuitable ads.

Since mobile ads are served through third-party ad SDKs integrated into the app, we analyze the relationship between inappropriate ads and ad SDKs. As shown in Table 1, there are 11 Google Play-certified ad SDKs that can be used in apps whose audiences include children. However, as the last column of Table 3 shows, uncertified ad SDKs, such as StartApp, AppBrain, and Facebook, are still leveraged. An interesting observation is that StartApp, a renowned ad SDK, used to be on the list of certified ad SDKs but has been removed, probably as domains and URLs sent in ads from StartApp flood users with links to malicious websites [38]. *Kodular* is a widely-used platform for mobile app development, and it received a lot of complaints from users [19, 23] as it continued to use the StartApp SDK for apps designed for children after StartApp's removal. More notably, **our results indicate that even ad SDKs regarded as certified by Google Play could display inappropriate ads to children**. The most noteworthy of them is Google AdMob, one of the largest global ad networks.

(a) Example Ad from Google AdMob (b) Example Ad from AppLovin

Figure 6: Example video-form ad screenshots delivered by Google Play-certified ad SDKs.

Case Study. As an example, consider an app named "Berita Bola Terkini" [30], designed to share updated football news and that belongs to the `Everyone` age group. This app leverages Google AdMob SDK. However, our investigation discovered that Google AdMob delivered sexually suggestive and pornographic-oriented video ads in this app, e.g., the ad shown in Figure 6(a), as an attempt to lure users into downloading an adult-only live broadcast app called MoreinLive [33]. Another example is AppLovin, also an ad SDK certified by Google Play. An App named "HD Background Photo Editing" [32], falling in the `Everyone` age group and employing AppLovin SDK, was distributed with an ad shown in Figure 6(b). The ad is trying to promote a game where the polygamy experience is its major selling point (according to the Chinese slogan within the figure). This severely violates Google Play's prohibition on ads towards children related to adult relationships. The above two cases are not accidental. We have found a total of 589 such violations in the ads displayed by Google AdMob and 48 in that shown by AppLovin.

4.4 RQ3: Region Impact on Inappropriate Ads

Our third research question is motivated by the fact that advertising violations may vary from region to region due to different cultural characteristics, child protection policies, and penalties in different countries or regions. We thus have done a preliminary study to investigate such potential differences. To build a dataset for RQ3, we select 1,000 apps from RQ1's dataset, where the app meets the criterion of displaying at least one in-app ad in RQ1's experiments. The selection is random while favoring apps with video ads and app promotions (the two less numerous types in the in-the-wild distribution, as shown in Table 2), as we desire to examine the regional difference in multiple ad types. We perform the experiments on the 1,000 apps four times following the same setup in RQ1, except

each time under the network environment of a different region (i.e., the United States, Australia, India, and Hong Kong). We detail the experiment results in Appendix A, which indicate the regional difference on inappropriate ads do exist. With this preliminary study, our aim is to shed light on these regional disparities and inspire future research to examine this issue more thoroughly.

4.5 RQ4: Comparison of *AdRambler* with State-of-the-art Approaches

For our last research question, we are interested in comparing the performance of *AdRambler* with state-of-the-art in-app ad collection efforts.

MadDroid: In 2020, Liu et al. [26] proposed an automatic framework named *MadDroid* for detecting malicious in-app ad content, which is the first tool that systematically collects both ad-loading content and ad-clicking content. Unfortunately, as it is incapable of accurately identifying and clicking ad views, it considers three commonly-used classes of views as potential ad views, namely *WebView*, *ImageView*, and *ViewFlipper*, and clicks all of them to ensure coverage. Within limited time and exploration steps, it cannot precisely click ads but only explore aimlessly, which is a significant flaw in its dynamic testing.

MAdLife: In 2019, Chen et al. [9] proposed *MAdLife*, another automatic approach for malicious in-app ad detection, which is also unable to identify ad views. Their solution is to open the app without any GUI exploration and only collect the *WebView* on the launching page as ads, which results in a significantly lower coverage as mentioned in RQ1.

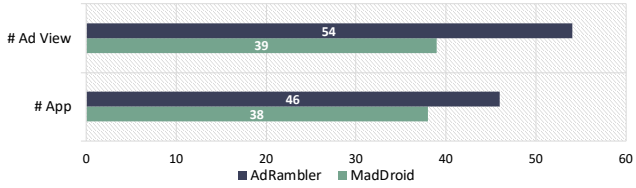

Figure 7: Comparison with the state-of-the-art approach.

We add only MadDroid to our comparison, as both our framework and MadDroid utilize app exploration for ad collection, while MAdLife adopts no app exploration and achieves a relatively lower coverage. We thus use the same configuration, i.e. number of exploring steps per app and time interval between steps, and the same dataset as MadDroid for a fair comparison. We randomly select 100 apps that are still available on Google Play from MadDroid's dataset to avoid the apps being outdated. We run the experiment on the 100 apps with both MadDroid and *AdRambler* under the same network environment, and evaluate their performance based on the number of ad views collected and the number of apps with ads collected. The result is displayed in Figure 7, where *AdRambler* harvests 138% (54/39) ads and successfully collects ads in 121% (46/38) apps compared with MadDroid. Upon our manual investigation, we found there are mainly two reasons for MadDroid's lower performance: (1) a number of inputs are wasted on clicking potential ad views, while most of them are non-ad views or part of ad views belonging to the same ad; this exploration flaw in particular limits MadDroid's ability to harvest ads in apps with more than one ad; (2) as opposed to *AdRambler*'s depth-first search exploration strategy, MadDroid utilizes a breadth-first strategy, where the "BACK" input event is given a much higher priority, preventing MadDroid from reaching ad-containing GUI pages. Interestingly, we find one case where the ad is collected by MadDroid but not *AdRambler*. Through further investigation, we discover that this ad is triggered by the "BACK" event exclusively, which is not covered by *AdRambler* within a limited number of steps. Nevertheless, this evaluation clearly demonstrates *AdRambler*'s superiority against the state-of-the-art approach in ad collection.

5 DISCUSSIONS

5.1 Who Share the Blame?

Our empirical study demonstrates that even though Google Play-certified ad SDKs, such as Google AdMob, dominate the market share in in-app ad serving, inappropriate advertising continues to emerge in a non-incidental manner. However, it would be unfair to simply blame these certified ad SDKs alone, as the appearance of inappropriate ads could be due to the negligence and/or the unwillingness to fulfill the duties of multiple parties together: if non-certified ad SDKs are used to serve ads to children, it's the developers' fault for their poor choices of ad SDKs, and the app markets' fault for their lack of education towards developers, and more importantly, their bad vetting, as they fail to identify such SDK misuse; and, if certified ad SDKs are used and inappropriate ad content is still propagated to children, the situation may be a bit more complex.

Upon our further investigations, it turns out that, to comply with app markets' advertising policies, certified ad SDKs usually require app developers to declare the ad content rating suitable for their target audience, so that ad SDKs could serve the appropriate ads for different age groups. Thus, the inappropriate ads exposed to children through certified SDKs could be due to different reasons: if the app developers indeed specify the appropriate ad content rating according to their target audiences, it's the SDKs' fault for labeling the wrong ad-content rating (for instance, an ad suitable only for adults is labeled by SDKs as *suitable for children*), and advertisers' fault for providing inappropriate ad creatives in the first place; otherwise, it's the app developers' fault for the wrong declaration. As for the app markets, they cannot perform any vetting procedures toward apps in this case, unless the declaration information is also shared with them either by developers or SDKs.

We further discovered that when integrating ad SDKs into apps, developers can specify ad content ratings with configuration settings[7], either directly via the SDK's web-side client or by invoking specific SDK APIs within their apps. We thus built a static scanner on top of Soot [40] for discovering the invocations of these SDK APIs and found that none of the apps detected with inappropriate ads in our research was using these APIs. This indicates that the developers of these apps either neglected to provide appropriate ad content ratings to SDKs, or they did so on the SDK's web-side client. Unfortunately, in terms of finding the culprits, this is the most we can do on the app side, as we don't have access to the SDK's web-side client, which requires logging in with the app developers' own SDK accounts. Fortunately, through this analysis, we can provide concrete suggestions to all parties involved toward

[7]For example, with Google AdMob, refer to https://developers.google.com/admob/android/targeting#ad_content_filtering.

establishing effective communication channels to help find the culprits and thereby mitigate the issue in the future. We describe this in Section 5.2.

5.2 Implications

Our research in this paper indicates that children's exposure to inappropriate in-app ads is not uncommon. When children use smartphones out of the sight of their guardians, they may be easily misled by such ads, resulting in negative cognition and even detrimental behavior. Therefore, we appeal to the community to invest more in exploring this research direction and establish a much more effective mechanism for advertising placement, distribution, and monitoring to block the dissemination of undesirable content that is harmful to children. In addition to what this paper investigates, Google Play generally prohibits ad behaviors that could lead to unintentional clicks from child users. In-app ads displayed misleadingly will result in unintentional clicks from underage users, thus undermining their experience and gaining unscrupulous profit. For future work, we aim to detect such inappropriate ad behaviors in accordance with the regulations.

Based on our analysis in Section 5.1, we make the following suggestions to all parties involved in the in-app ad ecosystem: 1) for app developers, to choose ad SDKs more carefully, preferably using only certified ad SDKs when their target audiences include children, and declare proper ad content ratings to ad SDKs and app stores; 2) for ad SDKs, to strengthen scrutiny against ad content provided by advertisers, and to actively participate in the SDK certifying program provided either by Google Play or by other app markets; 3) specifically, for certified ad SDKs, to build a communication channel with the app stores and share the ad content ratings declared by app developers, so that app stores can perform checks against apps' age groups to better maintain the appropriateness of in-app ads; 4) for advertisers, to exercise self-censorship in regards to their own ad creatives, and to directly provide accurate age-appropriate ratings themselves, as they are the best-equipped to determine suitability; 5) for app stores, to enhance their app vetting process in the following aspects: to identify uncertified ad SDK misuse utilizing static analysis approaches such as third-party library detection tools [28, 44, 45] or dynamic analysis approaches such as our *AdRambler*, to obtain ad content rating declared by app developers either directly from developers or from ad SDKs and compare it with the app's intended age group to see if the app developers made the proper declaration, to implement dynamic approaches such as *AdRambler* to periodically detect inappropriate ads and monitor new emerging ad SDKs; and, for app stores who do not currently have an ad SDK certifying program, it is highly recommended to establish one as soon as possible.

5.3 Limitations

Although this work focuses on precisely triggering and collecting ads and detecting the presence of kid-unsuitable ads, it still comes with common limitations of dynamic testing techniques. For example, we may be unable to trigger all ads in each app within a limited scope of exploration. Also, the third-party component-provided ads shown in a certain app could be different each time launched. Nonetheless, since the content of an ad is primarily determined by the ad SDKs, ads within a single app have relatively little impact on our investigation findings, especially when we take the measure of conducting large-scale experiments to minimize the effect of the factors above. Besides, the reliability of the investigation results relies in part on the stability of the network. There are cases where network fluctuations prevent successful loading. In response, our measure is to use a VPN service with a stable Wifi signal to simulate the network environment of each region. The accuracy of the Google Cloud APIs may also impact the experimental results. Fortunately, Google Cloud services are widely employed by our research community [41] and have been experimentally demonstrated in successfully flagging unsafe content [9, 26]. We also did a small-scale manual verification with 100 random samples where the Google Cloud APIs achieved an accuracy of 75%. Moreover, as our identification of ad views relies on a whitelist-based approach with regard to the package names of ad SDKs, it may be under influence of code obfuscation, namely the obfuscating type that renames the package names of ad SDKs. Fortunately, such obfuscation is very rare within commercial Google Play apps; we randomly sampled 100 apps from the dataset of RQ1, manually examined all the package names within the apps, and found none of those cases. Another limitation is that our framework is currently unable to explore apps automatically on the iOS mobile operating system. Fortunately, Chen et al. [10] state that the descriptions of apps for Android and iOS are identical by crawling the app metadata from the iTunes Store and Google Play, which indicates that restrictions on app age grouping apply in other formal app markets. Since ad SDKs investigated in this paper, i.e., Google AdMob, are also available for iOS app development, it is reasonable to infer that the same concern exists in iOS devices as well.

6 RELATED WORK

Mobile Ad Content Analysis. In 2013, a study by Chen et al. [10] first explored the content inappropriateness of in-app advertising on mobile devices from children's perspective; since it is a manual analysis, the number of explored apps is limited to hundreds. In 2016, Rastogi et al. [34] explored the malvertising-related security risks of in-app ads. However, their study only focused on the ad-clicking content, leaving ad-loading content untouched. In 2019, Chen et al. [9] presented MAdLife, which explores abusive tactics, such as click fraud, malvertising, and other threats related to ad-clicking content. Nonetheless, its limitations are evident in ad gathering as described in Section 4.5. In 2020, Liu et al. [26] developed MadDroid to detect malicious ad content as introduced in Section 4.5, which is unable to precisely identify and click the ad views in dynamic testing. While these existing efforts are more or less relevant to ad collection, none of them achieves ad view identification during automated app exploration, or focused on automatically detecting ad inappropriateness towards children.

Child-related Mobile App Analysis. Liu et al. [25] designed a machine learning model for predicting whether mobile apps are designed for children. Reyes et al. [35] proposed a dynamic analysis framework to evaluate privacy-related behaviors of Android apps, which is then used to check the app's compliance with the Children's Online Privacy Protection Act (COPPA). Luo et al. [27] proposed an automatic approach to determine whether a kid-friendly Android app contains inappropriate content, including a very rough inspection of ads based on screenshots only. However, none of them have specifically concentrated on the inappropriateness of in-app ad content toward children.

ACKNOWLEDGMENT

We are deeply grateful to the anonymous reviewers for their insightful comments and suggestions that have helped improve this paper. Their feedback has particularly motivated us to delve further into finding the culprits of inappropriate advertising and providing more concrete suggestions for all parties involved. Parts of this work have been supported by ARC Laureate Fellowship FL190100035 and the Guangdong Basic and Applied Basic Research Fund (Grant No. 2021A1515011562).

REFERENCES

[1] AccessibilityService | Android Developers, 2022. https://developer.android.com/reference/android/accessibilityservice/AccessibilityService.
[2] Hong kong society for the protection of children (hkspc). https://www.hkspc.org/php/webcms_en/public/index.php3?refid=109&mode=published, 2022.
[3] Neutral age screen - Manage target audience and app content settings, 2022. https://support.google.com/googleplay/android-developer/answer/9867159?visit_id=638009215648103129-3811598826&rd=1#neutral-agescreen.
[4] Summaries of Updates to Google Play Policies, 2022. https://support.google.com/googleplay/android-developer/answer/9876714?hl=en&ref_topic=9877065.
[5] Md Ahasanuzzaman, Safwat Hassan, and Ahmed E Hassan. Studying ad library integration strategies of top free-to-download apps. *IEEE Transactions on Software Engineering*, 2020.
[6] Kevin Allix, Tegawendé F Bissyandé, Jacques Klein, and Yves Le Traon. Androzoo: Collecting millions of android apps for the research community. In *2016 IEEE/ACM 13th Working Conference on Mining Software Repositories (MSR)*, pages 468–471. IEEE, 2016.
[7] AppBrain. Android widgets and ui libraries. https://www.appbrain.com/stats/libraries/tag/ui-component/android-widgets-and-ui-libraries, 2022.
[8] Pew Research Center. Children's engagement with digital devices, screen time. https://www.pewresearch.org/internet/2020/07/28/childrens-engagement-with-digital-devices-screen-time/, 2020.
[9] Gong Chen, Wei Meng, and John Copeland. Revisiting mobile advertising threats with madlife. In *The World Wide Web Conference*, pages 207–217. ACM, 2019.
[10] Ying Chen, Sencun Zhu, Heng Xu, and Yilu Zhou. Children's exposure to mobile in-app advertising: An analysis of content appropriateness. In *2013 International Conference on Social Computing*, pages 196–203. IEEE, 2013.
[11] Federal Trade Commission. Children's privacy. https://www.ftc.gov/tips-advice/business-center/privacy-and-security/children's-privacy, 2020.
[12] Federal Trade Commission. Children's online privacy protection rule. https://www.ftc.gov/enforcement/rules/rulemaking-regulatory-reform-proceedings/childrens-online-privacy-protection-rule, 2022.
[13] Federal Trade Commission. Complying with coppa: Frequently asked questions. https://www.ftc.gov/tips-advice/business-center/guidance/complying-coppa-frequently-asked-questions-0, 2022.
[14] Feng Dong, Haoyu Wang, Li Li, Yao Guo, Tegawendé F Bissyandé, Tianming Liu, Guoai Xu, and Jacques Klein. 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)*, 2018.
[15] Beatriz Feijoo and Charo Sádaba. When ads become invisible: Minors' advertising literacy while using mobile phones. 2022.
[16] Cuiyun Gao, Jichuan Zeng, Federica Sarro, Michael R Lyu, and Irwin King. Exploring the effects of ad schemes on the performance cost of mobile phones. In *Proceedings of the 1st international workshop on advances in mobile app analysis*, pages 13–18, 2018.
[17] General Data Protection Regulation (GDPR). Conditions applicable to child's consent in relation to information society services. https://gdpr-info.eu/art-8-gdpr/, 2022.
[18] General Data Protection Regulation (GDPR). General data protection regulation (gdpr) compliance guidelines. https://gdpr.eu/, 2022.
[19] Gio. Google play removed startapp sdk from family program. https://community.kodular.io/t/google-play-removed-startapp-sdk-from-family-program/108718, 2021.
[20] Google Play Console Help. Ads and monetization. https://support.google.com/googleplay/android-developer/answer/9898834/, 2022.
[21] Google Play Console Help. Google play families policies - examples of common violations. https://support.google.com/googleplay/android-developer/answer/9893335#zippy=%2Cexamples-of-common-violations, 2022.
[22] Google Play Console Help. Participate in the families ads program. https://support.google.com/googleplay/android-developer/answer/9283445/, 2022.
[23] Asraful Islam. What can i do? violation of families policy requirements. https://community.kodular.io/t/what-can-i-do-violation-of-families-policy-requirements/130767, 2021.
[24] Ling Jin, Boyuan He, Guangyao Weng, Haitao Xu, Yan Chen, and Guanyu Guo. Madlens: Investigating into android in-app ad practice at api granularity. *IEEE Transactions on Mobile Computing*, 2019.
[25] Minxing Liu, Haoyu Wang, Yao Guo, and Jason Hong. Identifying and analyzing the privacy of apps for kids. In *Proceedings of the 17th International Workshop on Mobile Computing Systems and Applications*, pages 105–110, 2016.
[26] Tianming Liu, Haoyu Wang, Li Li, Xiapu Luo, Feng Dong, Yao Guo, Liu Wang, Tegawendé Bissyandé, and Jacques Klein. Maddroid: Characterizing and detecting devious ad contents for android apps. In *Proceedings of The Web Conference 2020*, pages 1715–1726, 2020.
[27] Qian Luo, Jiajia Liu, Jiadai Wang, Yawen Tan, Yurui Cao, and Nei Kato. Automatic content inspection and forensics for children android apps. *IEEE Internet of Things Journal*, 7(8):7123–7134, 2020.
[28] Ziang Ma, Haoyu Wang, Yao Guo, and Xiangqun Chen. Libradar: fast and accurate detection of third-party libraries in android apps. In *Proceedings of the 38th international conference on software engineering companion*, pages 653–656. ACM, 2016.
[29] Brian Pia. Sexually suggestive ads appearing on children's apps. https://abc3340.com/archive/sexually-suggestive-ads-appearing-on-childrens-apps, 2017.
[30] Google Play. Berita bola terkini. https://play.google.com/store/apps/details?id=com.berita.bola.terkini.update, 2021.
[31] Google Play. Apps games content ratings on google play. https://support.google.com/googleplay/answer/6209544, 2022.
[32] Google Play. Hd background photo editing. https://play.google.com/store/apps/details?id=com.roniit.hdqualitybackground, 2022.
[33] Google Play. Moreinlive. https://play.google.com/store/apps/details?id=com.acfantastic.moreinlive, 2022.
[34] Vaibhav Rastogi, Rui Shao, Yan Chen, Xiang Pan, Shihong Zou, and Ryan Riley. Are these ads safe: Detecting hidden attacks through the mobile app-web interfaces. In *NDSS*, 2016.
[35] Irwin Reyes, Primal Wijesekera, Joel Reardon, Amit Elazari Bar On, Abbas Razaghpanah, Narseo Vallina-Rodriguez, Serge Egelman, et al. "won't somebody think of the children?" examining coppa compliance at scale. In *The 18th Privacy Enhancing Technologies Symposium (PETS 2018)*, 2018.
[36] Elena Root and Bogdan Melnykov. 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/, 2018.
[37] Mikey Smith. 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, 2015.
[38] Jeff Stone. Scammers are abusing mobile ad networks in an attempt to phish android app users. https://www.cyberscoop.com/ad-network-malware-android-startapp-adsalsa-wandera/, 2020.
[39] Ash Turner. How many smartphones are in the world? https://www.bankmycell.com/blog/how-many-phones-are-in-the-world, 2022.
[40] Raja Vallée-Rai, Phong Co, Etienne Gagnon, Laurie Hendren, Patrick Lam, and Vijay Sundaresan. Soot: A java bytecode optimization framework. In *CASCON First Decade High Impact Papers*, pages 214–224. 2010.
[41] Chengcheng Wan, Shicheng Liu, Henry Hoffmann, Michael Maire, and Shan Lu. Are machine learning cloud apis used correctly? In *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*, pages 125–137. IEEE, 2021.
[42] Wikipedia. Elsagate. https://en.wikipedia.org/wiki/Elsagate, 2020.
[43] Wikipedia. Elsa (frozen). https://en.wikipedia.org/wiki/Elsa_(Frozen), 2022.
[44] Xian Zhan, Lingling Fan, Tianming Liu, Sen Chen, Li Li, Haoyu Wang, Yifei Xu, Xiapu Luo, and Yang Liu. Automated third-party library detection for android applications: Are we there yet? In *Proceedings of the 35th IEEE/ACM International Conference on Automated Software Engineering*, pages 919–930, 2020.
[45] Xian Zhan, Tianming Liu, Yepang Liu, Yang Liu, Li Li, Haoyu Wang, and Xiapu Luo. A systematic assessment on android third-party library detection tools. *IEEE Transactions on Software Engineering*, 2021.
[46] Onur Zungur, Gianluca Stringhini, and Manuel Egele. 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)*, pages 318–330. IEEE, 2020.
[47] Onur Zungur, Guillermo Suarez-Tangil, Gianluca Stringhini, and Manuel Egele. Borderpatrol: Securing byod using fine-grained contextual information. In *2019 49th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, pages 460–472. IEEE, 2019.

Table 4: The inappropriateness of ads in different regions based on 1,000 Android apps.

	Region	The United States			Australia			India			China (Hong Kong)		
Category	**Type**	**# All**	**# Valid**	**# Kid-unsuitable / # App**	**# All**	**# Valid**	**# Kid-unsuitable / # App**	**# All**	**# Valid**	**# Kid-unsuitable / # App**	**# All**	**# Valid**	**# Kid-unsuitable / # App**
Ad View	-	1,027	-	-	1,144	-	-	1,187	-	-	1,324	-	-
Ad SDK	-	9	-	4/86	9	-	4/86	8	-	3/88	9	-	4/88
Ad-loading Content	Web Pages	4712	1,479	25/18	5,416	1,613	15/14	5,330	1,172	24/12	6,603	741	13/8
	Images	5639	2,262	49/14	7,030	2,726	73/56	6,840	2,326	39/31	10,295	3,395	187/123
	Videos	171	166	75/55	191	169	108/91	224	211	81/59	333	322	144/99
Ad-clicking Content	Web Pages	1,136	256	3/2	2,115	579	3/3	1,909	621	17/15	5,789	1,555	6/4
	App Promotions	306	293	114/46	210	203	7/3	71	70	18/7	91	91	59/19
In Total	-	**11,964**	**4,456**	**266/163**	**14,962**	**5,290**	**206/148**	**14,374**	**4,400**	**179/107**	**23,111**	**6,104**	**409/232**

APPENDIX

A EXPERIMENT RESULTS FOR RQ3

Table 4 displays our preliminary investigation statistics and Figure 8 further demonstrates the distribution of inappropriate ad content detected in different regions. There are significantly more inappropriate app promotions in the US than in other regions, with HK coming second, as depicted in Figure 8. Pornographic or sexually suggestive ads exist in all regions, especially in HK, the primary forms of which are ads on explicit live broadcast platforms and sexually suggestive games. Ads containing violent content are predominantly seen in the US and HK. The advertisers there mostly seem to promote a variety of games, including PC, web, and mobile games. It is worth noting that in the US, 42 out of all 75 violation-video ads are connected to violent games portraying or simulating human-on-human violence, e.g., Mafia City and the State of Survival, far more than in other regions. Ads shown in India region have a very high number of gambling-baiting ads, such as those placed by advertisers like *ludosupreme.com* and *zupee.com*. Furthermore, social media apps like Facebook, Instagram, and Now&Me, which are commonly used by teenagers and above, also frequently appear in ads in India. In addition to some violation categories shared with other regions, Australia is unique in that there are a lot of ads placed by TikTok (for 12+).

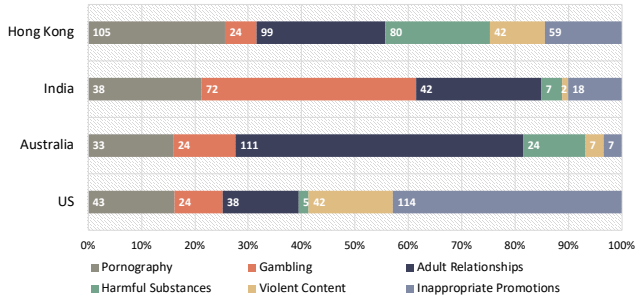

Figure 8: Distribution of inappropriate-ad categories in different regions.

The proportions of apps with in-app ads collected in the US, Australia, and India are less than 100% (i.e., 87.6%, 88.5%, 89.8%, respectively), which could be due to network fluctuations in the ad SDK's servers or in our test environment. Although this is only a preliminary study, the experimental results also suggest that kid-inappropriate advertising content in apps exists across multiple regions; and the distribution of different types of inappropriate content varies by region. We hope that with this preliminary study, we can shed light on this phenomenon, and inspire further research to explore the regional differences in inappropriate advertising in a more comprehensive manner.