An Investigation into Android In-App Ad Practice: Implications for App Developers

Boyuan He*, Haitao Xu†, Ling Jin*, Guanyu Guo*, Yan Chen*† and Guangyao Weng*
{*heboyuan, ljin1995, guanyuguo, gyweng*}*@zju.edu.cn*, *Zhejiang University
{*hxu, ychen*}*@northwestern.edu*, †Northwestern University

***Abstract*—In-app advertising has served as the major revenue source for millions of app developers in the mobile Internet ecosystem. Ad networks play an important role in app monetization by providing third-party libraries for developers to choose and embed into their apps. However, developers lack guidelines on how to choose from hundreds of ad networks and various ad features to maximize their revues without hurting the user experience of their apps. Our work aims to uncover the best practice and provide app developers guidelines on ad network selection and ad placement.**

To this end, we investigate 697 unique APIs from 164 ad networks which are extracted from 277,616 Android apps, develop a methodology of ad type classification based on UI interaction and behavior, and perform a large scale measurement study of in-app ads with static analysis techniques at the API granularity. We found that developers have more choices about ad networks than several years before. Most developers are conservative about ad placement and about 71% apps contain at most one ad library. In addition, the likeliness of an app containing ads depends on the app category to which it belongs. The app categories featuring young audience usually contain the most ad libraries maybe because of the ad-tolerance characteristic of young people. Furthermore, we propose a terminology and classify mobile ads into five ad types: Embedded, Popup, Notification, Offerwall, and Floating. We found that embedded and popup ad types are popular with apps in nearly all categories. Our results also suggest that developers should embed at most 6 ad libraries into an app, which otherwise would anger the app users. Also, a developer should use at most one ad network when her app is still at the initial stage and could start using more (2 or 3) ad networks when the app becomes popular. Our research is the first to reveal the preference of both developers and users for ad networks and ad types.

I. Introduction

Android has increasingly become the dominant operating system for mobile devices today and its world wide market share has hit 86.1% in the first quarter of 2017 [1]. According to [2], the total number of apps in Google Play has reached 2.9 million in the Feb 2017, 92.5% of which are free apps. In-app advertising has become one of the major sources of income for free app developers, VisionMobile predicts that the in-app advertising market will be worth 62 billion US dollars by 2017 [3].

Most of free apps leverage third-party in-app advertising for monetization. To achieve this, app developers need to connect their own apps to an ad network, a intermediate platform used for ad delivery. This process usually requires code integration: merging a pre-compiled library (a.k.a. ad library) provided by a specific ad network with the original app. In practice, there are hundreds of ad networks available on the market, and even for a specific ad network, developers still have great flexibility in this integration process. As described in documentation, many ad networks provide different ad types, such as banner, interstitial, native, video and etc, varying in aggressiveness to users. Furthermore, there is no unified classification system available for in-app ads, making it easier for developers to get confused. Consequently, for a specific category of app, following decisions must be made by developers before integration:

- What is the trend in ad networks used by apps in recent years?
- Can in-app advertising jeopardize user growth and app ratings?
- Are there any patterns in different co-existed ad types within the same app?
- How many ad networks within an app can be considered as acceptable to users?
- What are the popular ad types and what are the ad types that user are inclined to tolerate?

All the questions above must be carefully considered by developers because excessive ads impression or improper ad type may ruin an app's user experience, thus causing user loss. Given the fact that there are hundreds of ad networks with different features on the market, it poses a great challenge for developers to choose the best ad networks and ad types for their own apps with the balance between revenue and apps' user experience. Although mobile advertising ecosystem has been a target of many recent research, most of them focus on security and privacy of ad networks [4] [5] [6] [7] [8] [9] [10] [11] , mobile ad fraud [12] [13], and ad targeting [14] [15] [16]. None of existing works can address this challenge.

To provide guidelines of choosing best ad networks and ad types, in this paper we study Android in-app advertising from a developer's perspective. In particular, we develop a system called `MAdLens`, a static analysis framework for Android apps, which extracts libraries of different ad networks from a large set of apps, map each of ad relevant APIs inside a SDK to a specific ad type and generates summary information (e.g. number of ad API calls, number of instructions, number of Android component and etc). Leveraging on `MAdLens`, we further perform a large scale measurement across the Android market and uncover the current trend in usage of mobile ad networks, aggressiveness difference of ad types and its impact on various properties of apps.

To summarize, this paper makes the following contributions:

- To the best of our knowledge, we are the first to provide practical guidelines for mobile developers to monetize their apps through third-party in-app advertising.
- We are the first to map APIs of ad network to specific ad types and measure third-party in-app advertising in Android apps at API granularity.
- We are the first to provide a unified classification system for mobile in-app ads, and measure the impact of different ad types on various properties of apps.

Overall, we successfully extract 697 unique APIs from 164 ad network, which are identified in a dataset of 277,616 apps. Our results reveal many implications for developers and here we list some major ones: 1) The number of apps using in-app ads has increased in recent years but 71% apps contains at most one ad network, indicating that a conservative strategy is widely accepted. 2) Developers are inclined to use two lowly aggressive ad types: *Embbedded* and *Popup* simultaneously. 3) Too many ad networks that placed in the same app will anger the users and therefore lead to bad rating. Empirically, no more than 6 ad networks should be integrated into a single app.

The remainder of this paper proceeds as follows: Section 2 provides relevant background in Android third-party ads and Section 3 provides the system design of `MAdLens` as well as detailed discussion of methodology we apply in `MAdLens`. Section 4 demonstrates the result of our large scale measurement study and gives explanations as well as implications based on the result. Section 5 discusses the limitation of our works. Section 6 presents related work and Section 7 concludes the paper.

II. Background

In this section, we give a brief introduction to the ecosystem of mobile advertising and explain how third-party ad networks currently work. We also discuss details of mobile ad classification as well as ad aggressiveness.

A. Overview of Mobile Ad Network

Generally, the current ecosystem of mobile advertising involves three major participants: a publisher who shows ads in her app to make revenue, an advertiser who pays to get impression of her own ads and an ad network that serves as a intermediate platform between a publisher and an advertiser. In practice, an ad network connects the supply side platform (SSP) to the demand side platform (DSP), exchanging advertising impression inventory and revenue across the ecosystem.

As a publisher, most developers monetize their apps with third-party ad networks. This is usually implemented by importing an ad library provided by an ad network into an app, registering on the ad network's website to set up an ad account for payment, setting preferences in the ad library to specify credentials, suitable ad contents, desired ad frequency, layout, format and etc, and finally building and publishing the app. Unless explicitly stated, the two terms *ad network* and *ad library* are used interchangeably in this paper since all ad networks that we discussed have their correspondent ad libraries for integration. As shown in Figure 1, an ad request is triggered by ad library when a user browses specific parts of the app. The ad network server then receives the request, authenticates the developer's account, checks the parameters and responds with a desired ad. Any impression or click on this ad will be counted by ad network for revenue share.

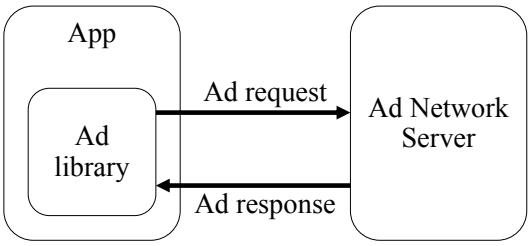


Fig. 1: Overview of mobile ad network.

B. Mobile Ad Network Integration in Android Apps

Ad networks often provide developers with both documentation and SDK for easy integration. The SDK usually consists of a pre-compiled ad library and its necessary dependencies. For Android apps, ad libraries are generally implemented in Java and provided in compiled jar files. Developers are required to import the ad library into the project of their own app and interact with the ad library with specific APIs.

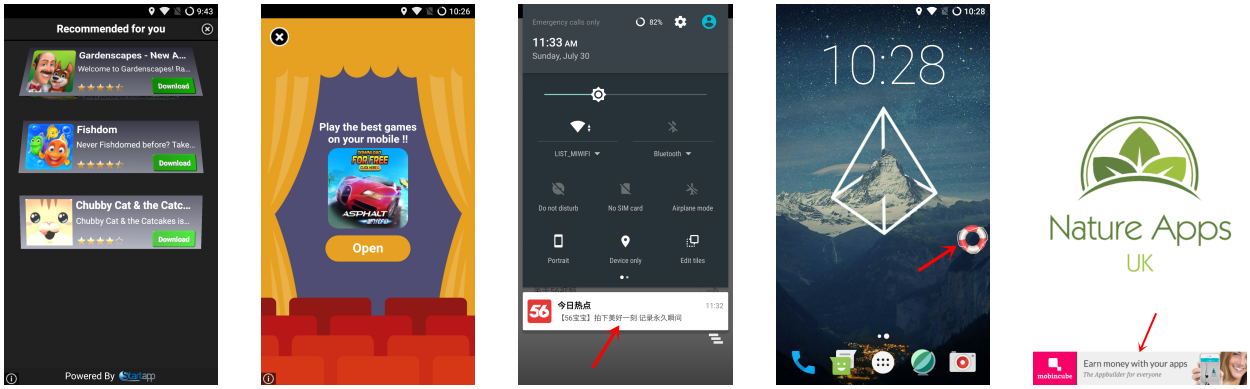
```
1  <?xml version="1.0" encoding="utf-8"?>
2  [...]
3  <com.google.android.gms.ads.AdView
4      xmlns:ads="http://schemas.android.com/apk/
           res-auto"
5      android:id="@+id/adView"
6      android:layout_width="wrap_content"
7      android:layout_height="wrap_content"
8      android:layout_centerHorizontal="true"
9      android:layout_alignParentBottom="true"
10     ads:adSize="BANNER"
11     ads:adUnitId="MY-UNIT-ID">
12 </com.google.android.gms.ads.AdView>
13 [...]
```

Listing 1: Banner ad implementation in XML

Most of ad networks provide a rich API surface, which allows the developer considerable latitude in manipulating the ad impression. However, this interaction is done either by changing layout files or calling specific Java API method. Listing 1 and Listing 2 give two examples of a banner ad implementation with Google Admob [17]. Both examples place an `AdView` to the app's GUI for the banner ad by changing app's XML layout files and using Java code respectively. Then a banner ad can be loaded where the developer wants by calling `loadAd()` method of the `AdView` class.

```
1  import com.google.android.gms.ads.AdView;
2  [...]
3  AdView adView = new AdView(this);
4  adView.setAdSize(AdSize.BANNER);
5  adView.setAdUnitId("MY-UNIT-ID");
6  [...]
7  }
```

Listing 2: Banner ad implementation in Java



(a) Offerwall (b) Popup (c) Notification (d) Floating (e) Embedded

Fig. 2: Examples of different mobile ad types.

C. Mobile Ad Classification and Aggressiveness

An ad network usually supports more than one ad format and developers should choose those that fit best with the design and user flow of their app. By randomly picking up 10 popular mobile ad networks (Table I), we can easily draw conclusion that there is no unified classification system for mobile ads. For research purpose, in this paper we propose a methodology of mobile ad classification based on the behaviour, UI layout and UI interaction of an ad.

Generally, we classify mobile ad into 5 different types that are described below:

1) **Offerwall** is the type of ad that uses a page appearing within the app to offer users rewards (e.g. Unlocking new features) in exchange for completing some specified actions (e.g. downloading other apps). An *Offerwall* ad often completely interrupts app's behaviour and cannot be ignored by users. Figure 2(a) is a typical *Offerwall* ad that we identified in an Android app.
2) **Popup** is the type of ad that uses a new pop-up windows in front of current app's GUI to display its contents. Some popup ads are full screen like interstitial ad, while others are not. While *Popup* ads interrupt app's behaviours, most of them can be closed by users by clicking on the close button. A typical *Popup* ad is shown in Figure 2(b).
3) **Notification** is the type of ad that uses the notification mechanism to display its contents (Shown in Figure 2(c)). It is displayed in the system's notification area rather than inside the app. Technically, an *Notification* ad are still active after the app is closed since it uses notification pushing mechanism. Like any other daily used notifications (e.g. SMS), by default *Notification* ads are displayed while the same ringtone is played. Besides, users often have no idea from which app a *Notification* ad is pushed, therefore have troubles to turn it off.
4) **Floating** is the type of ad that appears in front of the GUI of current app with a floating window (Shown in Figure 2(d)). It usually just occupies a small part of the screen, but can still be seen outside the app.
5) **Embedded** (Shown in Figure 2(e)) stands for a type of ad which embeds its contents into current window of the app. It covers a variety of different ad types listed in Table I, such as banner, video and etc. In general, embedded ads are user friendly because they rarely interrupt app's user flow or only interrupt for a short time and usually can be ignored, skipped or closed.

TABLE I: Supported ad format of 10 popular mobile ad network.

Ad Network	Ad Type Supported
Admob	Native, Video, Interstitial.
AdColony	Rich Media, Video.
Airpush	App Icon, Messaging, Notifications, Offerwall.
Chartboost	Content Lock, Interstitial, OfferWall, Video.
Fyber	Banner, Interstitial, Native, Video.
Inmobi	Banner, Native, Video, Interstitial, Rich Media.
Leadbolt	Native, Video, Interstitial.
RevMob	Interstitial, Video, Pop-up, Rich Media.
Startapp	App Icon, Full Page Ads, InApp Ads, Interstitial, Video.
Tapjoy	Content Lock, Interstitial, Offerwall, Rewards.

By taking all the factors mentioned above into consideration, we intuitively give different levels of aggressiveness to each type of mobile ad based on its user experience (Table II). For those types of ad that can be easily ignored, skipped or closed by user, we consider them as user tolerable and their aggressiveness as "Low". Otherwise, we give "High" as its aggressiveness level. The only exception is the notification ad. Although it can be closed or even disabled by user, it still cause interference to user reading other notifications because most of notification ads will eventually turn into a notification flood. We will demonstrate the impact of different ad types on various categories of Android applications in Section IV.

III. Methodology

In this section, we present our methodology that we apply in `MAdLens`.

A. System Overview

As mentioned earlier, we design and implement a system called `MAdLens`, which identifies internal third-party ad network modules from a large Android app dataset, maps APIs of ad networks to specific ad types, and finally generates detailed report of all the ad information and summaries for each app in the dataset using static analysis techniques. Our overall

TABLE II: Aggressiveness of ad type.

Ad Type	Possible to Ignore, Skip or Close	Aggressiveness
Embedded	Yes[1]	Low
Popup	Yes	Low
Float	Yes	Low
Notification	Yes	High[2]
Offerwall	No	High

[1] Banner ad only takes a small part of the screen, so it can be considered as ignorable. Other embedded ads like video ad usually can be skipped or closed.
[2] Even though notification ad can be closed or disabled in Android notification bar, we still consider its aggressiveness as "High" because it often affects users reading other notifications.

approach is summarized in Figure 3. Specifically, MAdLens works in 3 steps: module analysis, API mapping and static analysis, which are discussed in detail in the following subsections.

B. Data Collection

Before MAdLens could get to work, here we introduce our approach for dataset collection first. For fairness, instead of directly crawling from the web pages of Android market, we take another approach by using Android package name enumeration techniques based on a customized pre-defined dictionary. The crawler randomly picks up words from the dictionary, concatenates them into any possible package name of an app and downloads it if it exists in the market. Our *Android App Dataset* covers all 48 categories of apps in Google Play. Along with an app's apk file, we also collect its meta info including description, user rating and reviews. Since paid apps rarely have third-party ads, it is unnecessary to include any of them in the dataset. Overall, we collected 277,616 free Android apps from Google Play market in March, 2017.

C. Ad Network Identification

The first goal of MAdLens is to automatically identify all existing ad networks from each app in the dataset. Since most of ad networks are widely used in Android apps, we can assume common ad libraries are shared by many applications and thus code clones can be detected in these applications. Besides, an ad library, which is provided by a certain ad network, is a relatively independent piece of code in apps because there are usually only a few API calls across the boundaries between an ad library and other parts of the app. Based on the two important observations, we leverage the technique described in [18] [19] to detect these frequently used but relatively independent modules in apps. Clustering techniques are employed when measuring the coupling of different modules and finally, clusters are mapped to ad libraries, which are associated with ad networks. We refer the process described above as module analysis in our system.

In total, we identify over 200 unique ad networks from our dataset, however, only 164 of them have documentation available. Our goal is to perform a comprehensive third-party in-app ads analysis which requires semantic information at API level, so we only add the 164 unique ad networks into our *Ad Network Dataset*.

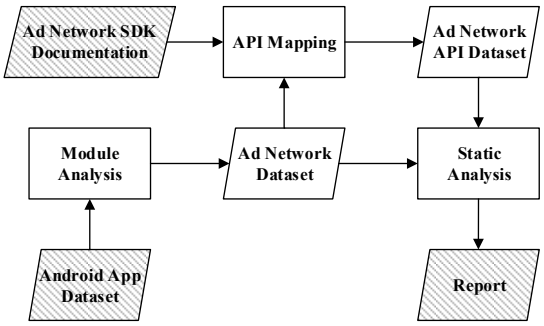


Fig. 3: Overview of MAdLens.

TABLE III: Feature statistics in ad network API dataset.

	Embedded Ad	Popup Ad	Floating Ad	Notification Ad	Offerwall Ad	Total
Number of APIs	407[1]	200	29	10	51	697

[1] APIs of embedded ad are found in both in Java code (299 APIs) and XML (108 APIs) layout resource files in apps (Listing 1 and Listing 2).

D. Ad Network API Mapping

Since there is no unified ad classification system available at present, we propose methodology of mobile ad classification based on the behavior, UI layout and UI interaction of an ad. To investigate various ad types adopted by apps at a large scale, it is necessary to have the "API mapping" information which includes all the correspondent ad network APIs to specific ad types. Obviously, it is impossible to automate this task since ad APIs are not always well documented, so we have to manually test all relevant APIs from 164 ad networks and map them to the 5 ad types we defined in Section II-C. This is a non-trivial task that requires huge effort. For those ad libraries with detailed documentation, we write test cases to verify each APIs to make sure that it maps to the a correct ad type. For those undocumented ad libraries we identified from apps, we do API mapping by setting hooks to suspicious APIs with reverse engineering techniques and interacting with the host app to observe its behavior. Overall we get 697 unique ad APIs (Shown in Table III) and add them into our *Ad Network API Dataset*. On average, each ad network has 4.25 ad APIs.

E. Ad Detection

With the two datasets that generated from last two steps, now it is possible to detect different types of ads from different ad networks in Android apps. Recall that there are two approaches for ad APIs to integrated into an app, either in resource file (e.g. xml) or in Java code (See Section II-B). To get a complete result, both resource files and Java code need to be analyzed thus we leverage on Androguard [20], an lightweight opensource reverse engineering tool for Android for in static analysis. For resources files, we first decompile the apk file with Androidguard to get all the Android layout files, then parse all the XML nodes and match them with our 2 datasets. Similarly, for java code, smali code is generated by Androguard after decompiling the apk file and we again match the smali code with the 2 datasets. Note that the code of ad library itself must be filtered before the code matching

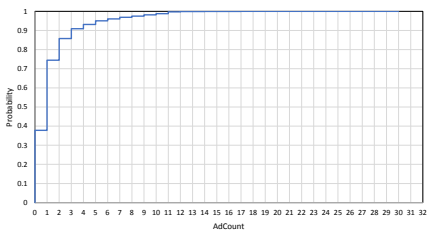


Fig. 4: CDF of the number of ad libraries embedded in an app.

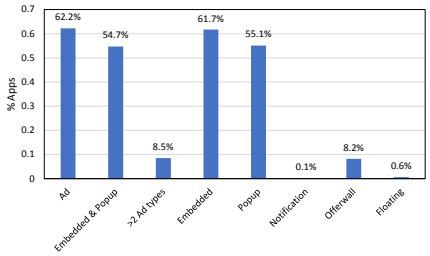


Fig. 5: Breakdown of apps by the ad type.

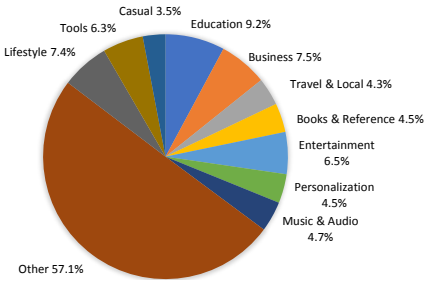


Fig. 6: Top 10 categories with the most apps.

process because self-reference to ad APIs may exist inside the code of ad libraries.

In comparison with other Android reverse engineering tools, one of Androguard's major advantages is its resistance to apk obfuscation and hardening. Androguard successfully decompiles 98% of Android apps in our measurement. For those apps which try to hide their code behaviours by adopting app hardening techniques, we characterize them with summary information (e.g. numbers of Android activities in code and mainifest file, numbers of Java class and instructions, etc) that extracted from the apk file, and eliminate them from our dataset.

IV. Data Analysis and Results

To understand the status quo of how in-app ads are involved in the real-world apps, we conducted a large-scale measurement study of in-app ad libraries among the apps from Google Play. Specifically, we collected a total number of 277,616 apps during March 2017. We then perform both manual and static analysis on those real-world apps across tens of app categories to better understand the preference of developers in in-app ad selection from various angles.

A. The Trend in the Number of Ad Libraries Embedded in an App

We estimate the trend in the number of ad libraries embedded in an app. The last update time of an app indicates the year in which the latest version of the app comes onto the market. We examine whether there exists a positive correlation between the number of ad libraries hosted by an app and its last update year. In Figure 7, we use boxplots to demonstrate the correlation. For each box, its bottom corresponds to the number of ad libraries per app on the 25th percentile, its top corresponds to the ad library number on the 75th percentile, and the line across the box corresponds to the ad library number in the median. These boxplots show that the apps in the years 2008 and 2009 barely have ads, 25% apps released in the years from 2010 to 2014 host at least one ad library, and 50% apps and 25% apps released in the recent years from 2015 to 2017 host at least one or two ad libraries, respectively. Thus the figure presents a clear trend that the number of ad libraries embedded in an app increases with the year.

One interesting question is to see how many ad libraries a developer usually places in her app *noways*. Figure 4 shows

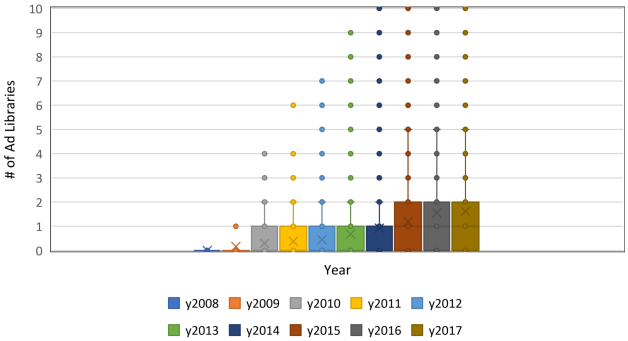


Fig. 7: The trend of the number of ad libraries along with year.

the cumulative distribution function (CDF) of the number of ad libraries embedded in an app. Up to 37.8% apps do not contain any ads, which is surprising given the fact that developers profit from their apps mainly by accommodating ads in their apps. The percentages of apps partnering with one ad library, two ad libraries, and three libraries are about 33%, 10%, and 5%, respectively. This indicates that most apps only involve one or two ad libraries. One possible explanation is that developers avoid to place too many ads[1] in apps since displaying ads may affect user experience, hurdle the promotion of the app, and finally decrease the revenue of the developers.

Implication 1: Developers have more choices about ad networks than several years before. Most developers are conservative about ad placement in their apps given the observation that about 71% apps contain at most one ad library.

B. Prevalence of Ad Types in an App

We also examine the popularity of each of the five mobile ad types among 277,616 apps. Figure 5 shows a breakdown of the apps by the ad type involved. We can see that 62.2% apps in our dataset contain at least one ad type, which is reasonable since after all advertising is one of the most important monetization ways for ad developers. An app could contain multiple types of ads, and 8.5% apps accommodate more than two out of the five types of ads. *Embedded* and *Popup* are among the most popular ad types, and up to 61.7%

[1]Note that the number of ad libraries included in an app is proportional to the number of ads displaying on the app.

and 55.1% apps contain those two types of ads, respectively. Furthermore, these two ad types usually appear in one app simultaneously, and more than a half (54.7%) of apps are found to contain both ad types. *Offerwall* is the third popular ad type, which is observed in 8.2% apps. *Notification* and *Floating* ad types are the two least popular, with less than 1% apps containing them.

Implication 2: Statistically, developers include two lowly aggressive ad types, *Embedded* and *Popup*, simultaneously in more than a half of their apps. In contrast, the two highly aggressive ad types including *Offerwall* and *Notification* are not popular.

C. Prevalence of Ad Types in the Apps of the Top 10 Categories

Based on the category tag associated with each app, the apps we collected fall into 48 categories. The top 10 categories with the most apps are shown in Figure 6. It can be seen that apps are quite dispersed across categories. The number of apps in the top 3 three categories, `Education`, `Business`, and `Lifestyle`, only occupy less than 10% each, respectively. The rest popular categories include `Entertainment`, `Tools`, `Music & Audio`, `Personalization`, `Books & Reference`, `Travel & Local`, and `Casual`. Up to 57.1% apps fall under the categories out of the top 10 ones.

We further study the prevalence of the five ad types in the apps of the top 10 categories, which is helpful for developers to learn the best practices for placement of different kinds of ads in the corresponding popular categories. Figure 8 provides the details about the ad placement in the top categories. Note that for each app category, the figure provides the percentages of apps in this category which host: 1) ads, 2) ads of more than two different ad types, 3) both *Embedded* and *Popup* ads, 4) *Embedded* ads, 5) *Popup* ads, 6) *Notification* ads, 7) *Offerwall* ads, and 8) *Floating* ads. Please check the legend of the figure for the meaning of each column.

Percentage of apps with ads. Among the top 10 categories, `Business`, `Education` and `tools` apps are the least likely to contain ads, with 39.6%, 48.0%, and 48.5% having ads, respectively. IV-B shows that 62.2% apps contain ads in overall. One reason why these three categories of apps have the below average percentages is that those apps are mainly utility tools and too many ads may distract or even annoy users. In contrast, about 74% to 87% of the apps in the four categories, `Entertainment`, `Music & Audio`, `Personalization`, and `Casual`, contain ads.

Implication 3: The likeliness of an app containing ads depends on the app category to which it belongs, to some extent. The business or utility apps are much less likely to contain ads than the entertainment or casual apps. The app categories featuring young audience usually contain the most ad libraries maybe because of the ad-tolerance characteristic of young people. Thus, developers may decide the ad placement issue based on the category of their apps.

Percentage of apps accommodating at least three types of ads. App developers could place multiple types of ads in their apps for maximizing the profit. The figure shows that the percentage of such apps is not high. Only 1.1% `Business` apps are embedded with at least three types of ads, while the `Casual` apps have the largest likeliness, with 23.6% accommodating at least three ad types.

Embedded ads and Popup ads. *Embedded* ads turn out to be the most popular ad types, which is true across all categories. 39.6% to 85.7% apps in the top 10 categories contain *Embedded* ads. *Popup* ads are quite popular too. Interestingly, the figure shows that a significant proportion (from 36.8% to 71.5%) of apps across all the top categories contain both *Embedded* ads and *Popup* ads.

Notification, Offerwall, and Floating ads. Comparatively, these three kinds of ad types are much less popular than *Embedded* and *Popup* ad types. *Offerwall* ad type is observed across all top categories, and up to 23.8 `Casual` apps contain *Offerwall* ad type. In contrast, the *Notification* and *Floating* ads are trivial. `Casual` apps have the largest percentage to contain *Notification* ads, with a value of 0.4%, *Floating* ads mostly appear in `Lifestyle` apps, occupying only 2.9%.

Implication 4: The two lowly aggressive ad types, *Embedded* and *Popup*, are popular with apps in nearly all categories. The highly aggressive ad type, *Offerwall*, is somewhat popular. Developers could consider placing these three types of ads on their apps.

D. Correlation between in-app Ads and User App Ratings

Intuitively, a user could be annoyed by an app embedded with too many ads. Thus it is interesting to examine whether there indeed exist the cause-effect relationships between in-app ads and user ratings. We explore the question by studying the correlation between the number of ad libraries contained in an app and the number of users who give bad ratings for the app. Figures 9 depicts such a correlation. It clearly shows that the number of bad user ratings increase significantly along with the number of ad libraries. Bad comments received on apps with 7 ad libraries or more increase sharply compared to those received by apps with 6 ad libraries or fewer.

Implication 5: Developers should avoid embedding too many ad libraries into an app, empirically no more than 6, which otherwise would anger the app users and result in bad ratings.

E. Ad Network Choice for Apps in the Different Stages of Their Lifecycles

To maximize the revenue, apps in the different stages of their lifecycles may consider different ad networks. Among the metadata information associated with an app, the *downloads* of an app could best indicate the stage of the lifecycle in which the app currently is, at least to some extent. A newly released app typically has few downloads and an app on the app market for a while usually has more downloads.

We examined the correlation between downloads of an app and the number of ad networks that the app is partnering with. Figure 10 depicts such a correlation for all 277,616 apps in our dataset. The x axis denotes the logarithm of the downloads of apps, and the y axis denotes the number of ad networks

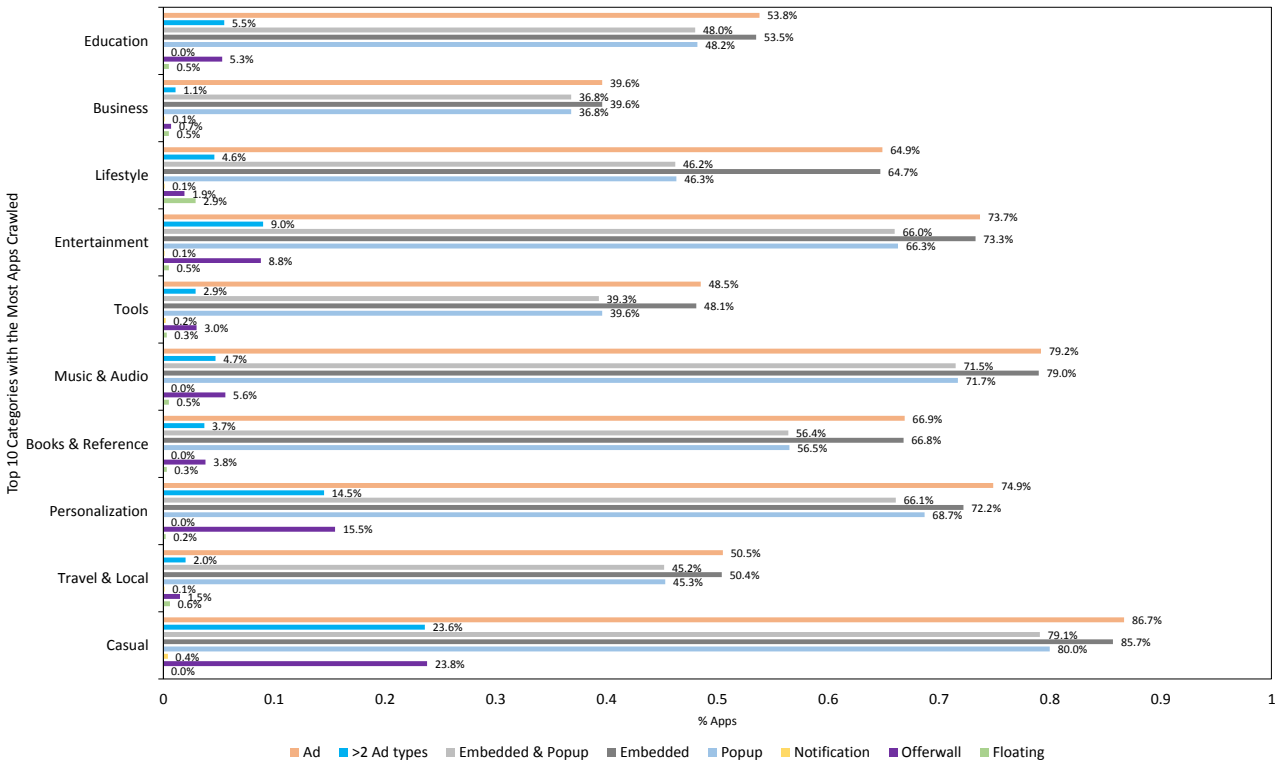


Fig. 8: Breakdown of the apps in the top 10 categories by the ad type.

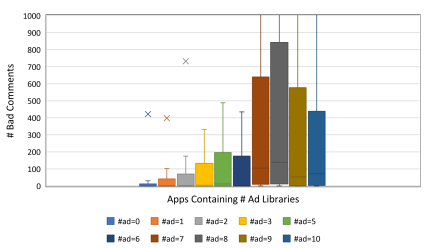


Fig. 9: Correlation between the ad library number in an app and the number of bad ratings for the app.

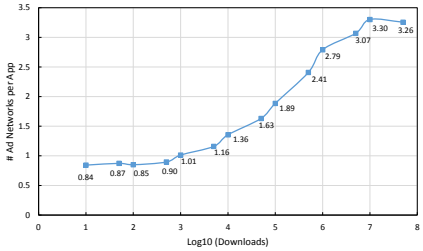


Fig. 10: Ad network choice for apps in different lifecycle stages.

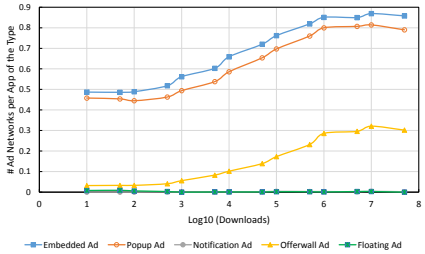


Fig. 11: Ad type choice for apps in different lifecycle stages.

used per app on average. The curve in the figure shows that statistically the average number of ad networks used by an app has a positive correlation with the downloads of the app. Specifically, an app with 10 downloads uses 0.84 ad networks on average and an app with 1 million downloads adopts 2.79 ad networks on average. That is, a new app usually uses less ad networks and an relatively old app tends to use more ad networks. Since the number of ad networks is reasonably proportional to the number of ads displayed on the app, we could say that a new app tends to display less ads than an old app.

Implication 6: A developer should use at most one ad network when her app is still at the initial stage and could start using more (2 or 3) ad networks when the app becomes popular, reflected by its downloads.

F. Ad Network Choice for Apps in the Different Stages of Their Lifecycles

Similarly, we examined the correlation between downloads of an app and the number of ad networks of each ad type that the app is partnering with. Figure 11 depicts such a correlation for each of the 5 ad types. One observation is that statistically *Embedded* and *Popup* are the two most popular ad types for apps with any number of downloads; in contrast, *Notification* and *Floating* are the two least popular ad types, with negligible percentages; *Offerwall* is only popular with the apps with tremendous downloads. Overall, the average number of ad networks of each ad type used per app increases with the downloads of apps.

Implication 7: As the statistical results of the large dataset suggest, a developer should mainly place *Embedded* and *Popup* ads on her app in whatever lifecycle stage, and could start to place *Offerwall* ads when her app has many enough

TABLE IV: Pearson correlation coefficients

	# ad networks	# low-aggressiveness ad types	# high-aggressiveness ad types	Downloads	Ratings
# low-aggressiveness ad types	0.568934	1	0.999974	0.004088	0.038284
# high-aggressiveness ad types	0.686385	0.999974	1	0.004583	0.03201

downloads.

G. Impact Analysis of the Aggressiveness of the Ads Hosted by an App

As mentioned before, ad types are classified into two categories: low-aggressiveness and high-aggressiveness. It would be interesting to evaluate the impact of different levels of aggressiveness of the ads on other performance metrics of apps. We choose Pearson correlation coefficient to perform correlation analysis.

As a measure of the linear correlation between two variables, the value of Pearson correlation coefficient is between +1 and -1, where 1 is total positive linear correlation, 0 is no linear correlation, and -1 is total negative linear correlation [21]. Table IV lists the computed Pearson correlation coefficients.

The table shows that the number of low-aggressiveness ad types and the number of high-aggressiveness ad types included in an app have extremely strong positive correlation between each other, which suggests that an app usually includes these two aggressiveness-levels of ads simultaneously. In addition, both of them have relatively strong positive correlation with the number of ad networks used of the app, which is reasonable given that the two ad types of ad networks combined make the total number of ad networks. Surprisingly, the levels of aggressiveness have no correlations with other metrics of the apps including downloads and ratings. One possible reason is that apps in the different stages could host both aggressiveness-level ads for maximum profit, and app ratings could be affected by many factors and not only by the aggressiveness of the ads in the app.

Implication 8: Developers may use both low-aggressiveness ads and high-aggressiveness ads at the same time for maximizing their revenues.

V. Limitation

We acknowledge the following limitations of our methodology.

Ad Number Approximation. Our system `MAdLens` takes a static approach to analyze Android apps, so it has inherited limitations of static analysis. `MAdLens` measures the number of ads inside a certain app by counting the occurrence of ad APIs. However, this is an approximation due to code control flows and not every ad API can be necessarily triggered at runtime. While dynamic approaches are capable of capturing an app's runtime behaviors, it is still very challenging to analyze apps at a large scale. Besides, our experiment shows that most ad networks have encrypted their network traffic with servers, making it even more difficult for dynamic analysis to monitor ad behaviors at network level. One possible dynamic solution is to apply image recognition techniques in identifying ad activities on the user interface, and we leave it for future work.

Android App Hardening. Although we have apply some countermeasures to app hardening techniques by characterizing these apps with code summary information (Section III-E), in practice this approach could not cover all the cases and thus may lead to false negatives, which affects the accuracy of our measurement.

VI. Related works

A. Mobile advertising

Mobile advertising has become a hot topic for research recently. The privacy issues related to ad networks and associated ad libraries have been the focus of many existing works. Grace et al. [7] study the potential risks (e.g. privacy leakage) posed by embedded or in-app ad libraries. Demetriou et al. [6] and Meng et al. [9] estimate the risk associated with user data exposure to advertising libraries in Android apps and show that malicious ad libraries can infer sensitive information. To address the privacy concerns, many solutions are proposed to isolate advertising from application. Pearce et al. [4] use privilege separation to identify advertising-related over-privilege. Shekhar et al. [5] implement privilege separation by extracting ad services from recompiled apps. Zhang et al. [10] provide a general approach to isolate third-party ad libraries into a separate process and implement privilege, display and input isolation. Liu et al. [11] first use machine-learning to detect ad libraries and then use code instrumentation to de-escalate their privileges. Besides, to balance user privacy and mobile advertising, Leontiadis et al. [8] propose a privacy protection framework to "achieve an equilibrium" between the developer's revenue and the user's privacy based on the establishment of a feedback control loop that adjusts the level of privacy protection.

In addition to user privacy, Crussell et al. [12] study mobile ad fraud perpetrated by Android apps and identify two fraudulent ad behaviors in apps. One is requesting ads while the app is in the background and the other is clicking on ads without user interaction. Liu et al. [13] study a kind of mobile ad fraud called "placement fraud" and design a system for automated detection. Nath [14] characterize user targeting strategies of top ad networks and measure their effectiveness by developing a tool called MAdScope. Ad targeting has also been discussed in [15] and [16].

Besides, some of the recent works study the impact of ad networks on an app's rating. Ruiz et al. [22] find that integrating multiple ad networks can lead to negative impact on user experience. Fu et al. [23] collect user feedback to explain why people dislike a given app.

B. Online advertising

Online advertising has been studied for decades from many perspectives with focus on security and privacy. Stone-Gross et al. [24] describe how online advertising ecosystem works,

study the associated security issues from network level and introduce known types of fraud, including impression spam, click spam, competitor clicking, conversion (action) spam and misrepresentation. Similarly, Xu et al. [25] study click fraud in online advertising and provide a novel approach for advertiser to detect and evaluate click frauds against their campaigns.

Apart from ad fraud, other various topics have also been discussed. Li et al. [26] study malicious activities behind online advertising and provide mitigation using prominent features they identify from malicious advertising nodes and their related content delivery paths. Apostolis et al. [27] analyze to what extent users are exposed to malicious content through online advertisements. Malicious advertising known as malvertising, exhibit different behaviors: drive-by downloads, deceptive downloads and link hijacking. Phillipa et al. [28] characterizing the value of user information and privacy to advertising revenue by measuring network traffic.

None of existing works cited in this section analyzes the potential impact of certain ad networks or ad types on app popularity at API granularity. To the best of our knowledge, we are the first to provide a unified classification of mobile in-app ads and practical guidelines for mobile developers to monetize their apps with best ad networks and ad types.

VII. Conclusion

App monetization could be the ultimate goal of most app developers. However, app developers lack the guidelines on how to maximize their app revenues. In this work, we aim to provide insights from developers about how to optimize their app monetization with optimal ad placement choices. To this end, we study the in-app advertising ecosystem from a developer's perspective. We collected 277,616 Android apps and developed a static analysis framework to extract ad libraries of different ad networks from those apps. We also abstract ad relevant APIs inside a SDK to ad types. With the extracted information by both manual labeling and static analysis, we further perform a large scale measurement study and uncover the current practice about ad placement. We found that most developers are conservative about ad placement and about 71% apps contain at most one ad library. In addition, the likeliness of an app containing ads depends on the app category to which it belongs. Furthermore, embedded and popup ad types are found to be quite popular with apps in nearly all categories. Our results also suggest that developers should embed at most 6 ad libraries into an app to avoid not to affect the app user ratings. Also, a developer should use at most one ad network at the initial stage of her app and could use 2 or 3 ad networks later. Our research is the first to reveal the preference of both developers and users for ad networks and ad types.

Acknowledgments

This research was supported by the National Natural Science Foundation of China under Grant No.61472209 and the U.S. National Science Foundation under Grant CNS-1408790. The authors would like to thank all the anonymous reviewers for their thoughtful comments.

References

[1] "Gartner Says Worldwide Sales of Smartphones Grew 9 Percent in First Quarter of 2017." http://www.gartner.com/newsroom/id/3725117.
[2] AppBrain, "Free vs. paid Android apps." http://www.appbrain.com/stats/free-and-paid-android-applications, 2017.
[3] VisionMobile, "App Economy Forecasts 2014 - 2017." December 2014.
[4] P. Pearce, A. P. Felt, G. Nunez, and D. Wagner, "Addroid: Privilege separation for applications and advertisers in android," in *Proc. of ASIACCS*. Acm, 2012, pp. 71–72.
[5] S. Shekhar, M. Dietz, and D. S. Wallach, "Adsplit: Separating smartphone advertising from applications." in *USENIX Security Symposium*, vol. 2012, 2012.
[6] S. Demetriou, W. Merrill, W. Yang, A. Zhang, and C. A. Gunter, "Free for all! assessing user data exposure to advertising libraries on android," in *Proc. of NDSS*, 2016, pp. 1–15.
[7] M. C. Grace, W. Zhou, X. Jiang, and A.-R. Sadeghi, "Unsafe exposure analysis of mobile in-app advertisements," in *Proc. of the WiSec*. ACM, 2012, pp. 101–112.
[8] I. Leontiadis, C. Efstratiou, M. Picone, and C. Mascolo, "Don't kill my ads!: Balancing privacy in an ad-supported mobile application market," in *Proc. of HotMobile*. ACM, 2012, p. 2.
[9] W. Meng, R. Ding, S. P. Chung, S. Han, and W. Lee, "The price of free: Privacy leakage in personalized mobile in-apps ads." in *Proc. of NDSS*, 2016.
[10] X. Zhang, A. Ahlawat, and W. Du, "Aframe: Isolating advertisements from mobile applications in android," in *Proc. of ACSAC*. ACM, 2013, pp. 9–18.
[11] B. Liu, B. Liu, H. Jin, and R. Govindan, "Efficient privilege de-escalation for ad libraries in mobile apps," in *Proc. of MobiSys*. ACM, 2015, pp. 89–103.
[12] J. Crussell, R. Stevens, and H. Chen, "Madfraud: Investigating ad fraud in android applications," in *Proc. of MobiSys*. ACM, 2014, pp. 123–134.
[13] B. Liu, S. Nath, R. Govindan, and J. Liu, "Decaf: Detecting and characterizing ad fraud in mobile apps." in *Proc. of NSDI*, 2014, pp. 57–70.
[14] S. Nath, "Madscope: Characterizing mobile in-app targeted ads," in *Proc. of MobiSys*, 2015, pp. 59–73.
[15] J. Lee and D. H. Shin, "Targeting potential active users for mobile app install advertising: An exploratory study," *International Journal of Human-Computer Interaction*, 2016.
[16] T. Book and S. W. Dan, "An empirical study of mobile ad targeting," *Computer Science*, 2015.
[17] "Mobile Ads SDK for AdMob," https://developers.google.com/admob/.
[18] V. Rastogi, R. Shao, Y. Chen, X. Pan, S. Zou, and R. Riley, "Are these ads safe: Detecting hidden attacks through the mobile app-web interfaces." in *Proc. of NDSS*, 2016.
[19] W. Zhou, Y. Zhou, M. Grace, X. Jiang, and S. Zou, "Fast, scalable detection of piggybacked mobile applications," in *Proc. of CODASPY*. ACM, 2013, pp. 185–196.
[20] "Androguard," https://github.com/androguard/androguard.
[21] "Pearson correlation coefficient," https://en.wikipedia.org/wiki/Pearson_correlation_coefficient.
[22] I. J. M. Ruiz, M. Nagappan, B. Adams, T. Berger, S. Dienst, and A. E. Hassan, "Impact of ad libraries on ratings of android mobile apps," *IEEE Software*, vol. 31, no. 6, pp. 86–92, 2014.
[23] B. Fu, J. Lin, L. Li, C. Faloutsos, J. Hong, and N. Sadeh, "Why people hate your app: Making sense of user feedback in a mobile app store," in *Proc. of KDD*. ACM, 2013, pp. 1276–1284.
[24] B. Stone-Gross, R. Stevens, A. Zarras, R. Kemmerer, C. Kruegel, and G. Vigna, "Understanding fraudulent activities in online ad exchanges," in *Proc. of IMC*. ACM, 2011.
[25] H. Xu, D. Liu, A. Koehl, H. Wang, and A. Stavrou, "Click fraud detection on the advertiser side," in *European Symposium on Research in Computer Security*. Springer, 2014, pp. 419–438.
[26] Z. Li, K. Zhang, Y. Xie, F. Yu, and X. Wang, "Knowing your enemy: understanding and detecting malicious web advertising," in *Proc. of CCS*. ACM, 2012, pp. 674–686.
[27] A. Zarras, A. Kapravelos, G. Stringhini, T. Holz, C. Kruegel, and G. Vigna, "The dark alleys of madison avenue: Understanding malicious advertisements," in *Proc. of IMC*. ACM, 2014, pp. 373–380.
[28] P. Gill, V. Erramilli, A. Chaintreau, B. Krishnamurthy, K. Papagiannaki, and P. Rodriguez, "Follow the money: understanding economics of online aggregation and advertising," in *Proc. of IMC*. ACM, 2013, pp. 141–148.