MAdScope: Characterizing Mobile In-App Targeted Ads

Suman Nath
Microsoft Research
suman.nath@microsoft.com

ABSTRACT

Advertising is the primary source of revenue for many mobile apps. One important goal of the ad delivery process is targeting users, based on criteria like users' geolocation, context, demographics, long-term behavior, etc. In this paper we report an in-depth study that broadly characterizes what targeting information mobile apps send to ad networks and how effectively, if at all, ad networks utilize the information for targeting users. Our study is based on a novel tool, called MAdScope, that can (1) quickly harvest ads from a large collection of apps, (2) systematically probe an ad network to characterize its targeting mechanism, and (3) emulate user profiles of specific preferences and interests to study behavioral targeting. Our analysis of 500K ad requests from 150K Android apps and 101 ad networks indicates that apps do not yet exploit the full potential of targeting: even though ad controls provide APIs to send a lot of information to ad networks, much key targeting information is optional and is often not provided by app developers. We also use MAdScope to systematically probe top 10 in-app ad networks to harvest over 1 million ads and find that while targeting is used by many of the top networks, there remain many instances where targeting information or behavioral profile does not have a statistically significant impact on how ads are chosen. We also contrast our findings with a recent study of targeted in-browser ads.

Categories and Subject Descriptors

C.4 [**Performance Of Systems**]: [Measurement techniques]

Keywords

In-app Advertising; Mobile Advertising; Targeted Advertising; User Profiles; Measurement

1. INTRODUCTION

In-app advertising is a key economic driver in mobile app ecosystem, funding a wide variety of free apps. Gartner predicts the mobile ad market to grow to $13.5 billion in 2015 [9]. The growth is primarily due to increasing engagement of users with apps—users are reported to spend significantly more time on mobile apps than on traditional web, and the gap is increasing [6].

Of central importance in the ad delivery process is *targeting*—the process of selecting specific ads to display to a given user of an app. In-app *ad controls*, which fetch ads from backend ad networks and display them to users, target users based on criteria such as app type, device attributes (e.g., screen size), user's long-term behavior, demographic, and geolocation. The presumption is that targeting benefits all parties in the ad ecosystem: users, app developers, ad networks, and advertisers. Note that targeting is not unique to in-app ads; it has been successfully used for in-browser ads as well [1, 8, 11, 30].

Despite its importance, in-app ad targeting is relatively ill-explored (compared to its in-browser counterpart). Existing results on in-browser ads (e.g., [1,3,7,8,11,23,30]) cannot be directly generalized to in-app ads for several reasons. First, unlike in-browser ads that cannot easily access user information because of the same origin policy [29], in-app ads run with the same permission as the app itself, making it easier for ad controls to exfiltrate user information. Therefore some targeting information (e.g., user's geolocation) is likely to be used more commonly by in-app ads than by in-browser ads, while some information (e.g., all or frequently used apps on the device) is unique to in-app ads. Second, ad controls often leave much targeting information optional to be provided by apps, and hence targeting behavior of even the same ad control may vary across apps depending on whether apps actually provide the optional targeting information. Third, some ad networks and advertisers such as mobile marketplaces are unique or more common in in-app ads and their characteristics have not been investigated before. Finally, the methodology for harvesting in-app ads is significantly different from that for harvesting in-browser ads.

In this paper, we present a first-of-its-kind study of targeted display ads delivered to mobile apps. Our objectives are to broadly characterize in-app display advertisement and to understand targeting mechanisms employed by apps and ad networks from an empirical perspective. The study seeks to understand several key questions including the following: *What targeting information do apps choose to send to ad* *networks? How effectively, if at all, do different ad networks utilize such information to target users? Do ad networks differentiate between different apps or different users using the same app, and if so by how much? What impact does targeting have on ad delivery process?*

Answers to the above questions are important not just for a general audience inquisitive into in-app ad ecosystem, but also for two important constituents of the ad ecosystem: app developers and ad networks. An app developer can use the answers to guide his decisions on what ad networks to choose to effectively target users. He can also reduce the targeting information that the app needs to send to only the ones that influence targeting decisions, thereby reducing bandwidth and exfiltrated user data. An ad network can use the answers to find how often app developers provide a specific optional targeting information to other ad networks and how much targeting opportunity it currently misses by not supporting the targeting information.

An empirical study of in-app ad ecosystem is challenging for various reasons: The vast number of mobile apps that show display ads, the large number of ad networks that serve ads to mobile apps, the diversity and dynamics of targeting mechanisms across ad networks and users all present significant challenges. Our first contribution in this paper is a novel tool called MAdScope that can address these challenges. Central to MAdScope are three key mechanisms to (1) quickly harvest ads from a large collection of apps, (2) systematically probe an ad network to characterize its targeting mechanism, and (3) emulate user profiles, i.e., interact with the ad ecosystem as if interactions were by users of specific preferences and interests, in order to understand if an ad network does behavioral targeting. Various aspects of these mechanisms are inspired by recent works on profile-based ad crawling [1], differential correlation [16], and ad classification [27], but adapted to in-app ads. All these capabilities are essential for understanding the full spectrum of targeting behavior of in-app ad ecosystem. We believe that MAdScope and its mechanisms will be valuable for future studies of in-app ad ecosystem as well.

Our second contribution is to characterize targeting behavior of in-app ad ecosystem by using MAdScope. We first study what targeting information apps send to ad networks. Our analysis of 500K unique ad requests harvested from 150K apps and 101 ad networks shows that apps do not yet exploit the full potential of targeting. Ad networks are aggressive in terms of allowing apps to provide targeting information of various categories: device, geolocation, demographic, keywords, and child flag—at least 76 of the 101 ad networks we studied can collect some of these information. However, much of the targeting information (41% on average per ad network) is left optional to developers, who do not often (in $> 90\%$ of the apps) provide them. For 17% of the apps, ad networks receive no targeting information since they leave all targeting information as optional and apps/developers do not provide any of them.

We next investigate how effectively top 10 ad networks utilize various targeting information to target users. We consider both *attribute-based targeting* that targets users based on various attributes encoded in ad requests, and *behavioral targeting* that targets based on user's historical activities. By analyzing over a million ad impressions, we found many instances where ad networks effectively utilize targeting information and behavioral profiles to target users, although the degree of utilization varies widely across networks. Our results show that in-app ads differ from in-browser ads in multiple ways—in-app advertising employs significantly less behavioral and demographic-based targeting but more location-based targeting, and it shows more entertainment-related ads.

Surprisingly, our study points out many instances where targeting information or behavioral profile does not have a statistically significant impact on how ads are chosen (within our observation window of one day). We are not certain why ad networks collect targeting information if not for selecting ads. Three plausible explanations for an ad network to collect such data could be: (1) forward-compatibility—the ad network wants keep the option open to select ads based on the information in future, and still be compatible to current or old versions of its ad controls used by existing apps, (2) backward-incompatibility—the targeting information is no longer used by the ad network but is still provided by old versions of its ad control, and (3) information brokerage—the ad network sells the information to other parties.

We would like to contrast our work with existing works on in-app ads. Most of the prior work has demonstrated the large extent to which apps are capable of collecting user's personal information and the potential implications to user's privacy [12,17,20,24,26] and device resources [4,15,19,28]. These works do not investigate user targeting. A recent work [27] investigates targeting, but in a limited scale and scope—it shows various categories of targeted ads shown to 100 apps by one ad network, but does not characterize how apps use optional targeting information and how effectively ad networks utilize targeting information and behavioral profile. To the best of our knowledge, no prior work investigates the key questions we investigate in the context of in-app ads.

In the rest of the paper, we first discuss backgrounds and challenges of measuring in-app ads in Section 2. We then describe MAdScope's mechanisms and our results characterizing apps (Section 3) and ad networks (Section 4) from targeting point of view. We also contrast our findings with a recent study on in-browser ads in Section 5). Finally we discuss related works in Section 6) and conclude in Section 7.

2. BACKGROUND AND GOALS

2.1 In-app Ads

To display ads from a third-party mobile ad network such as AdMob[1] and MobClix[2] in an app, the app developer first registers with the ad network. The ad network provides the developer with an ad control (i.e. a library with some visual elements embedded within) that he includes in his app and assigns some screen "real estate". When a user runs the app, the ad control fetches ads from the ad network and displays it to the user. As mentioned in Section 1, unlike in-browser ads, in-app ads run with the same permission as the app itself.

The ad ecosystem consists of multiple entities including advertisers, ad agencies and brokers, ad networks, apps, and app users. Figure 1 shows a generic workflow of a mobile app requesting an ad and actions resulting from a click on the ad. First, ad control in the app requests for an ad from

[1] http://www.google.com/ads/admob
[2] https://developer.mobclix.com

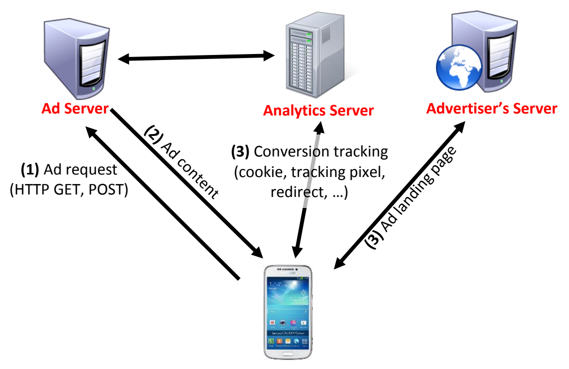

Ad Server
Analytics Server
Advertiser's Server
(1) Ad request (HTTP GET, POST)
(2) Ad content
(3) Conversion tracking (cookie, tracking pixel, redirect, …)
(3) Ad landing page

Figure 1: In-app ad serving workflow.

ad network. Second, the ad server serves an ad with associated contents, which the ad control displays in the app. The ad server may collect the ad offline from various brokers and ad agencies, or in real time from ad auction networks. Third, a click on the ad initiates connections to an ad analytics server, which tracks ad conversions, and to advertiser's server, which serves the landing page of the ad. Depending on the ad control, the click may generate only one connection to the analytics server, which can redirect the connection to the landing page.

▶ **Targeting.** An ad request, which is typically an HTTP GET or POST method, may include various targeting attributes such as app name, device properties, user's location, demographic, etc. An ad network can use these information to select targeted ads for the user. While some targeting information is automatically collected and sent by the ad control, some others are optionally provided by apps/developers (through ad control APIs). Targeting can also be based on a user's long-term behavior such as the types of apps he uses and the types of ads he clicks on. The ad server can obtain such statistics from analytics servers and maintain it for each user as his *profile*.

2.2 Goals

Our goal is to broadly characterize behavior of two important parties in mobile in-app ad ecosystem: apps and ad networks, and to understand their targeting mechanisms from empirical perspective. There are two key aspects of such understanding.

▶ **Client-side behavior.** *What various targeting information do mobile apps send to ad networks?* The goal is different from that in prior work [12, 26] that demonstrated, by analyzing ad controls, the large extent to which ad controls are *capable of* collecting user's personal information. Actually sending some of the information to ad networks, however, is often left optional to apps— depending on how app developers customize their use of ad controls within apps, some of the personal information the ad controls are capable of collecting may or may not actually be sent to ad networks. For example, AdMob, the most popular ad control in our datasets, allows apps to optionally send user's current location and demographic information such as his age and gender. How often apps actually send such information require analyzing ad requests made by apps—mere examination of individual ad controls is not sufficient.

Answer to the above question will illustrate the actual usage of these optional targeting information—whether developers really care about targeting users, whether they take additional efforts to provide these optional targeting information despite runtime overhead and privacy implications [12, 26], and if they do, what type of targeting information apps actually send to ad networks.

▶ **Ad network behavior.** Even if specific targeting information, such as a user's geolocation, is sent to ad networks, do the networks actually use the information for targeting users? If they do, how much do they rely on specific targeting information in selecting ads for a user? Answers to these questions can be useful for developers in choosing the right ad network and understanding whether it is worth providing optional targeting information. For example, as our results in Section 4 show, not all ad networks that collect geolocation actually use it to target users. Based on the results, a developer who wishes to show location-based ads in his app can choose an ad network that actually utilizes geolocation to target users. On the other hand, if the developer decides to use an ad network that can accept user's geolocation as an optional attribute but does not utilize the information for targeting users, he can choose not to provide the information to reduce exfiltrated user data, access permissions, and development and runtime overhead.

To the best of our knowledge, no previous works try to answer the above questions for mobile in-app ads. Answering the questions is challenging due to the following reasons.

2.3 Challenges

▶ **Scalability.** To understand what targeting information apps send to ad networks, we need to analyze a large enough sample of ad requests from real apps. One obvious way to collect such data is to capture app traffic in large network providers [28], but this requires access to such network providers. Another option is to recruit real users and to capture their ad traffic [27], but this is hard to scale to a large number of apps. Another option, which we explore in this paper, is to use an automation tool to automatically execute apps and to capture their ad traffic [18, 22]. Correctness and scalability of this approach, however, hinge upon answering several non-obvious questions: *Does poor coverage of existing automation tools introduce any bias to collected sample of ads requests? How long do we need to run an app to capture all its ad requests?*

▶ **Black-box ad networks.** Without knowing internal targeting algorithm of an ad network, we need to characterize its targeting mechanism only from its externally observable behaviors. Passively monitoring ad requests and fetched ads from existing apps may not be sufficient for this purpose. For example, to determine if an ad network targets users based on their locations, we need to compare ads from requests (1) with and without location information, (2) without any other optional targeting information (to isolate the impact of location), and (3) with the same mandatory information (e.g., IMEI number). The available sample of ad requests may not contain requests satisfying all these criteria. For instance, if all available ad requests to an ad network contain location information, we cannot know how the network would behave with a request without location information. Moreover, there are many variabilities such as available ad inventory and time of day that can affect targeting behavior of ad networks. One must use sound statistical tools to derive meaningful conclusions in the presence of such variability.

▶ **Profile-based targeting.** Behavioral targeting of display ads on the Web utilizes user's long term behavioral profiles [1] such as the sites he visits and the ads he clicks on. To study such targeting for in-app ads, we need to mimic multiple user profiles and study how ad networks target different profiles. For scalability, the profiles need to be built automatically. This can be tricky due to a *click-contamination problem* discussed in Section 4. For instance, to automatically build a profile of a "sports-loving" user, we need to automatically click on all and only sports-related ads. To automatically determine if an ad is about sports, we often need to analyze its landing page, which may be determined only after clicking the ad. If the ad is not about sports, this click can contaminate the "sports-loving" profile.

In this paper we address these challenges with a novel tool called MAdScope and use it later in the paper to understand targeted ads within mobile apps. At a high level, MAdScope has two functionalities (Figure 2): (1) passively harvesting ads from existing apps; we use this to understand what targeting information is sent to ad networks, and (2) actively probing ad networks; we use this to characterize targeting behavior of ad networks. We describe them in more detail in next two sections.

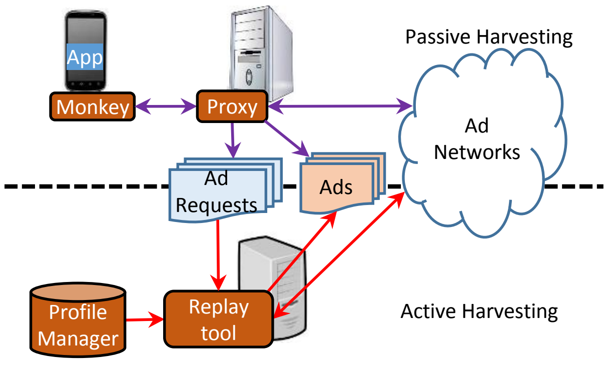


Figure 2: MAdScope architecture. Top half involves passive ad harvesting (Section 3), while the bottom half involves active ad harvesting (Section 4).

3. CHARACTERIZING CLIENT BEHAVIOR WITH PASSIVE AD HARVESTING

3.1 Passive Ad Harvesting in MAdScope

MAdScope harvests ad requests and impressions by running unmodified apps on a mobile device. To harvest ads (ad requests and corresponding responses) from an app, it launches the app on a mobile device, navigates to its pages, and captures ads shown in those pages with a network sniffer that runs on the same device or at a proxy sitting between the device and the Internet. The sniffer or the proxy captures all network traffic to and from the device, from which ad requests/responses are identified by looking for specific ad signatures (See MadFraud [5] for a principled approach of identifying ad requests from apps).

A recent work [27] used a similar methodology to harvest ads shown in 100 apps, but relied on humans to run the apps. Using human is not scalable. Since our goal involves ads from a large number of apps, MAdScope avoids the above scalability bottleneck by using a *Monkey*: a UI automation tool that can automatically launch an app in a real device and navigate through its pages by interacting with it, e.g., by clicking buttons, filling out text fields, swiping panels, etc. Previous work used Monkeys for various applications including app testing [22], ad frauds detection [5, 18], security bug detection [2], etc.

As mentioned before, one of our goals is to study what targeting information are sent to ad networks via ad requests. For our results to be sound, they need to be based on *all* or *an unbiased sample* of ad requests issued by a large sample of apps. All unique ad requests of an app can be collected by an ideal Monkey capable of visiting all runtime states (e.g., pages) of the app and thus triggering all ad requests the app can make. In practice, however, Monkeys have poor coverage. For instance, a Monkey may fail to visit some pages that can only be accessed by providing a correct username/password combination not known to the Monkey and hence miss the ad requests issued in those pages. (See [2] for a list of other reasons why a Monkey can have poor coverage.) If the missed ad requests are significantly different from the harvested ones, in terms of what targeting information is being sent out, the harvested sample can be a biased one. The amount of time a Monkey is given to explore an app can also affect the quality of harvested ad request sample: a longer exploration time may yield more ad requests, but at the cost of fewer explored apps within a given time. Therefore, before we analyze the ad requests collected by a Monkey, we need to first understand whether the ad request samples we collect are biased due to (a) poor coverage of the Monkey and (b) limited exploration time. We now investigate them empirically.

3.1.1 Understanding Ad Harvesting

To understand how apps make ad requests, we run top 3,000 ad-supported free Android apps in the Google Play market with a combination of human and an existing Monkey tool called PUMA [14]. We configured PUMA to launch each app on a Nexus 5 smartphone and to navigate through various app pages for at most 20 minutes per app. For around 700 apps, PUMA had low coverage—it explored fewer then 10 unique pages per app as the apps required login passwords and/or had custom controls that PUMA failed to interact with. We manually ran those apps (e.g., by supplying user name and password) to make sure that at least 10 or all unique pages are explored for each app. All network traffic to/from the device was captured with a proxy, where we identified ad requests by using techniques described in [5]. We harvested ad requests from a total of 31,203 unique app pages.

▶ **Observation 1.** *Most apps make the first ad request during launch time.*

Figure 3 shows the CDF of apps and the number of interactions PUMA made before an app issued its first ad request. As shown, for 81% of the apps, PUMA had to simply launch the app and make no interactions at all. These apps make ad requests during launch time so that the ads can be displayed on the first app page as soon as possible. Moreover, only 5 interactions were sufficient to harvest ad requests from > 95% of the apps, suggesting that even a basic Monkey that can interact with an app in a limited way

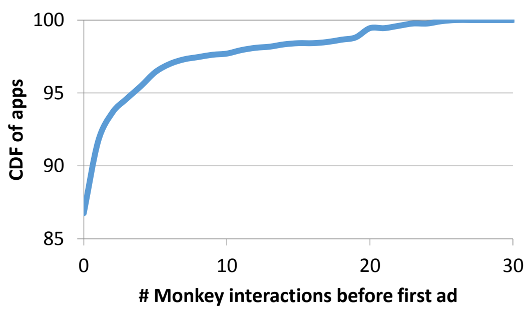


Figure 3: CDF of app count and number of Monkey interactions required to observe the first ad. No interaction is needed for 81% of the apps.

is able to capture first ad requests from a large fraction of apps.

▶ **Observation 2.** *For most apps, the first ad request is similar to all its ad requests.*

Let us first explain how we define two ad requests to be similar. A typical ad request is an HTTP GET or POST method call, consisting of a request page address (a host name and a path) and a query string with multiple attribute-value pairs separated by the '&' symbol. Attributes in a query string are either *variable* or *invariant*, depending on whether their values change or remain constant in two back-to-back ad requests from the same app page, over all apps.[3] For example, a timestamp attribute and a sequence number attribute are variable, while an IMEI number and a publisher ID are invariant. We consider two ad requests to be similar if they have the same request page address and the same values of invariant attributes. Intuitively, two similar ad requests convey the same or very similar targeting information.

Our results show that for 94.7% of the apps, the first ad request an app makes is similar to its subsequent requests.[4] Note that many of these apps required login and password and we had to manually run them to explore them well. We did not observe any instance where ad requests differ before and after user login. For 4.2% apps, subsequent ad requests differ from their first requests by only one attribute (e.g., `FirstPage=true` in first request and `false` in subsequent ones) and hence can be constructed from their first requests. Only ≈ 1% apps issue multiple different ad requests in different pages. Manual inspection reveals that these apps use keyword-based targeting and send different targeting keywords in different pages based on their contents.

3.1.2 Passive Harvesting in MAdScope

Insights from the above results guide us to the following scalable ad harvesting strategy in MAdScope: it launches an app and explores it with PUMA only until the app issues the first ad request or PUMA fails to explore further. We claim that the strategy yields:

[3]Note that attribute-value pairs encrypted with random salts look random. The above definition would treat such attributes as non-invariant, even if their unencrypted values are the same.

[4]Ad impressions returned by these similar requests, however, can be different.

Ad domain	Frequency
doubleclick.net	36%
mobclix.com	22%
appsgeyser.com	12%
adsmogo.com	12%
wapx.cn	3.8%
nend.net	1.1%
mopub.com	0.9%
mobfox.com	0.8%
applovin.com	0.7%
airpush.com	0.7%

Table 1: Top ten ad domains in the datasets.

- **Unbiased sample of ads:** Poor coverage or limited exploration time of PUMA does not introduce any significant bias on the harvested sample of ad requests. As Observation 1 shows, even if it explores zero or very small number of app pages, it still can harvest ad requests from most apps. Moreover, as Observation 2 shows, collecting only one ad request per app is sufficient, since that represents *all* ad requests made by 99% of the apps (i.e., not many unique ad requests are missed from unexplored pages).

- **Scalable harvesting:** As Observation 1 shows, MAdScope needs to make a small number of interactions to harvest its ad, enabling it to quickly harvest ads from a large number of apps. In our experiments, we could harvest ads from around 4000 apps per device per day.

3.2 Empirical Results

3.2.1 Datasets and Basic Statistics

We use the following two datasets:

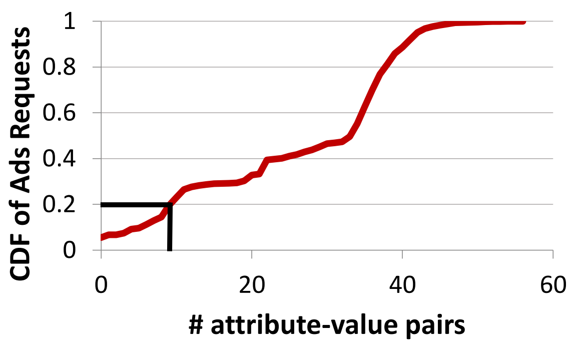
Dataset 1. This dataset contains 211,231 unique ad requests collected by MAdScope from 22,324 most popular free apps in Google Play market.

Dataset 2. The dataset is shared by the authors of the MadFraud system [5]. It consists of ad requests from 130,339 Android apps, selected randomly from a pool of 669,591 apps authors crawled from 19 markets including Google Play. Unlike Dataset 1, this dataset contains only ad requests that apps made during launch time: each app is launched, kept running for 60 seconds without any user interaction, and ads are captured. It consists of 353,181 unique ad requests. Note that, due to our Observation 1 and Observation 2, this dataset represents an almost unbiased sample of ad requests.

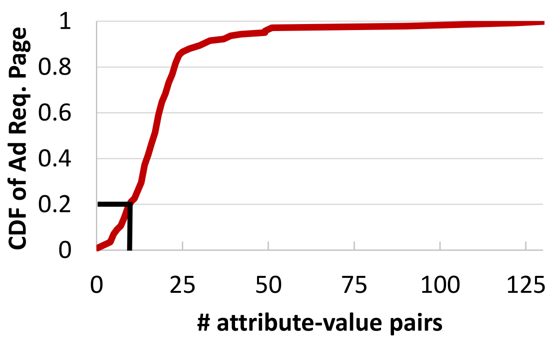
Overall, both the datasets contain ∼500K unique ad requests from ∼150K apps. They contain 143 unique ad request pages from 101 unique top level ad network domains. Table 1 shows ten most frequent top level domains appearing in the ad requests.

3.2.2 How much information is sent to ad networks?

To approximate the magnitude of information sent by apps to ad networks, we count the number of attribute-value pairs in each ad request. Figure 4(a) shows the cumulative distributions of the number of attribute-value pairs in all ad requests. As shown, ad requests contain a large number of attribute-value pairs: 26.3 pairs on average per re-



(a) Attr-value pairs in ad requests



(b) Attr-value pairs in request pages

Figure 4: CDF of the number of attribute-value pairs appearing (a) in all ad requests and (b) in all ad request pages. *$> 80\%$ of ad requests and $> 80\%$ of ad request pages contain > 10 attribute-value pairs.*

quest. More than 80% of all ad requests contain more than 10 attribute-value pairs.

We also aggregate the counts per ad request page address: for each address, we take union of all attribute-value pairs that appear in any ad request to that address and count them. Figure 4(b) shows the CDF of the number of attribute-value pairs of all 143 ad request pages in our datasets. Overall, ad networks are aggressive in collecting various attributes. The average number of attribute-value pairs per ad request page address is 20.4, with $> 80\%$ of the ad request pages supporting > 10 attribute-value pairs.

The results above demonstrate that ad networks are aggressive in collecting a large number of attributes via ad requests. Note that many of the attributes are automatically collected by ad controls (and hence are transparent to developers), while others are left optional to developers who need to explicitly provide their values. Moreover, not all these attributes are relevant to targeting users. We characterize these aspects next.

3.2.3 Mandatory and Optional Attributes

We define an attribute to be mandatory for a request page address if it appears in *all* ad requests to that page address from all apps; otherwise, the attribute is optional. The intuition is that if an attribute is optional, then it is highly likely for *some* developers to not use it and hence ad requests from their apps to not include the attribute. For example, the ad control requesting ads from googleads.g.doubleclick.com/mads/gma allows developers to explicitly provide value of the optional attribute `cust_age`; however, only a subset of the developers actually provide the attribute value. One limitation of this classification is that a mandatory attribute will deem optional if it does not appear in *all* versions of the ad control. For instance, if a mandatory attribute is introduced in a later version of the ad control, all ad requests from the latest version of the ad control will have the attribute, while none from an earlier version will have it.

We observed that, surprisingly, a large fraction (55%) of attributes are optional. On average, each ad request page address has 9.1 mandatory attributes and 11.3 optional attributes.

The result highlights the significance of our data-driven analysis of ad clients. By manually examining the usage of an ad client (as in previous works [12, 26]), one could identify extent to which ad clients are capable of collecting user's personal information. But since a large fraction of such information is optional, this gives only the upper bound, not the actual magnitude, of information leak. In contrast, examining ad requests from real apps gives the true magnitude of the information being sent to ad networks.

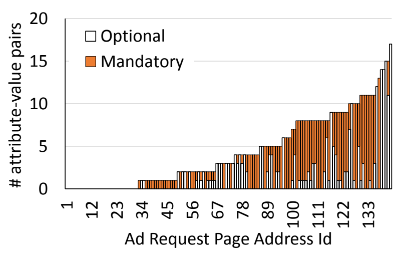
Category	Example values
Location	Latitude and Longitude, City, Zipcode
Keywords	Keywords
Device	IMEI number, MAC address, Android ID, IP address, CPU speed, RAM size, Screen size, Device model, Carrier name
Demographic	Age, Gender, Language
Child flag	True/False

Table 2: Targeting attribute categories and values.

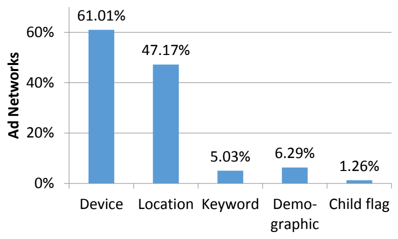
3.2.4 Identifying Targeting Attributes

To analyze targeting behavior, we must first identify various targeting information that are sent from apps to ad networks in ad requests. In other words, we need to identify which attributes in an ad request contain targeting information. Since there are many ad providers for Android (101 in our datasets), it would be prohibitive to manually reverse engineer each ad provider's ad serving protocol to determine which attributes in an ad request correspond to what targeting information. Instead, we develop a semi-automated method for such identification purely based on ad requests appearing in our datasets.

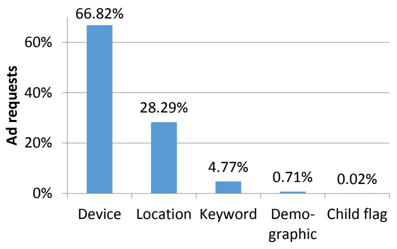
Our method consists of two steps. First, we keep only the attributes that satisfy either of the two criteria (and discard the others): (1) the attribute contains a relatively small number (we used a threshold of five) of distinct values in the entire datasets, or (2) its values are English words. Intuitively, since the ad requests in our datasets are harvested from a small set of similarly configured identical devices from the same location, a targeting attribute should contain one or a very small number of values across all requests. For example, if the apps are all run from the same location, a location attribute should always have the same value. Attributes containing English words include target-



(a) Targeting attributes



(b) Categories in ad networks



(c) Categories in ad requests

Figure 5: (a) shows distribution of identified mandatory and optional targeting attributes in all ad networks. *A significant number of targeting attributes are optional* (b) and (c) show frequencies of various categories of targeting attributes in all ad networks and in all ad requests, respectively.

ing keywords. The process discards attributes with arbitrary values such as timestamps or hashed values.

Second, for each of the remaining attributes, we manually inspect its name and values in our datasets and identify if the name and values are semantically meaningful (for example, `cust_gender=male` has a semantically meaningful interpretation but `q3=5` does not). We select all attributes with meaningful interpretation. Note that the first step discards attributes that are unlikely to have meaningful interpretation, and thus reduces laborious manual effort. Finally the selected attributes for each request page are categorizes into five different categories: Location, Keywords, Device, Demograhic, and Child Flag. Example values of attributes of these categories are shown in Table 2.

▶ **Targeting Dictionary.** The targeting attributes, their categories, and whether they are mandatory or optional are encoded in MAdScope as a *Targeting Dictionary.* This is a static dictionary that maps each ad request page address to a set of targeting attributes that may appear in corresponding query strings and whose semantics are known. For instance, for the ad request page address googleads.g.doubleclick.com/mads/gma, the dictionary contains the information that the attribute `cust_age` is optional and its value is age of the user in plain text. MAdScope uses this dictionary for systematically probing various ad networks to learn their targeting behavior (Section 4).

Our datasets contain 143 unique request page addresses and 2921 unique attributes for these requests. By using the above methodology, we could recognize 626 targeting attributes (21% of the unique attributes). Note that the above methodology may miss some targeting information sent by an app, e.g., when they are sent as hashed or encrypted values or codes (e.g., “v=1“ instead of “gender=male”). Therefore our results on extent of targeting is actually a lower bound.

3.2.5 *Targeting Attribute Counts*

Figure 5(a) shows the distribution of identified targeting attributes in all ad request page addresses. (Page addresses are sorted based on increasing order of the number of identified targeting attributes.) 112 of total 143 ad request page addresses contain at least one targeting attributes. On average, each address contains 4.4 targeting attributes.

Figure 5(a) also distinguishes between mandatory and optional targeting attributes. On average, 41% of targeting attributes are optional—there are 1.8 optional and 2.6 mandatory targeting attributes per ad request page address. The

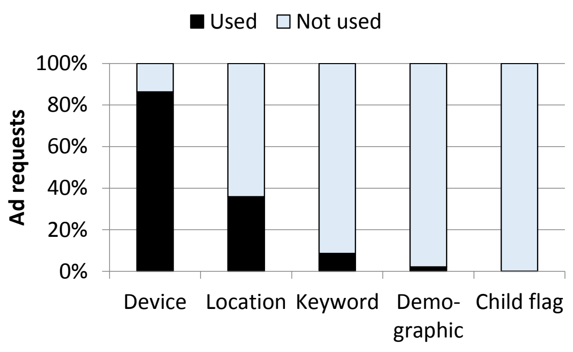


Figure 6: Frequencies of ad requests that do not use a category of targeting attributes even though the ad network support that category.

high fraction of optional targeting attributes, again, highlights the importance of data-driven analysis of ad clients.

We also categorize targeting attributes into various categories shown in Table 2. Figure 5(b) and (c) show distributions of various categories of targeting attributes in all ad request page addresses and in all ad requests, respectively. As shown, two most common categories of attributes are device-related attributes such as IMEI number or device Android ID and location attributes such as latitude and longitude. Keyword-based (also known as contextual) and demographics-based targeting is less common. Few ad networks collect a child flag, to serve child-appropriate ads.

3.2.6 *Omission of Optional Attributes*

Figure 6 shows how often apps/developers actually provide the optional targeting attributes of various categories. As shown, a large fraction of the ad requests do not include attributes of a targeting category even though their ad networks support that category. For instance, 65% of the ad requests do not provide location attributes even though their ad networks allow location attributes. The fractions of ad requests not utilizing the opportunity of providing targeting attributes are higher (> 90%) for other categories, except for the Device category that is often automatically collected by ad controls. A developer may omit providing an optional attribute for many reasons: to reduce exflitrated user information, to avoid asking for additional permission (e.g., for location attribute), to avoid additional programming efforts to include the attribute, to avoid runtime overhead of sending additional attributes over the network, or even due to his lack of awareness of the targeting attribute.

A few ad networks leave all targeting information optional. For 17% of ad requests in our datasets, corresponding ad networks do not receive any targeting information since they leave all targeting information as optional and apps/developers do not provide any of them.

3.2.7 Static Targeting Attributes

Some targeting attributes such as location are usually specified dynamically, e.g., based on user's current geolocation. Interestingly, we observed that some developers specify some of the attributes statically. For example, the app `Virtual Table Tennis 3D` requests ads from ad.where.com, with `city=ADELAIDE` irrespective of the user's current location. Similarly, we observed a few apps requesting ads from doubleclick.net by setting `Age=10` and `Gender=female`. We suspect that most users of these apps are kids and girls, and without an easy way to determine the true age/gender of the users, app developers statically target the dominant user group.

To quantify such static targeting attributes, we counted all ad requests that specify location or demographics different from the Monkey's profile. We found that at least 3.3% of all ad requests contain one or more such statically defined attribute values.

4. CHARACTERIZING AD NETWORKS WITH ACTIVE AD HARVESTING

The goal of active probing in MAdScope is to issue ad requests to a target ad network, with carefully chosen combinations of targeting attributes and values, in order to facilitate understanding of targeting mechanism of the ad network. For instance, to understand how much an ad network targets users based on their locations or whether its ad serving rate changes when targeting is enabled, we would like to compare responses from many ad requests without location and with various locations. MAdScope's active probing can issue such customized ad requests. Similarly, to understand if an ad network does behavioral targeting, we would need to build long term behavioral profiles and compare ads received by various profiles. MAdScope can build such profiles and issue ad requests from them.

One way to issue customized ad requests is to build a custom app with the desired ad control and to customize it with various optional attributes (such as location). However, this allows only to customize the parameters that ad controls allow developers to set; other parameters that ad controls collect automatically (such as Android ID or IMEI number, or location for some ad controls) cannot be customized.

To address this, MAdScope relies on modifying and replaying passively harvested ad requests. This is supported by the following two observations.

► **Observation 3.** *A passively harvested ad request can be replayed later to harvest more ads.*

► **Observation 4.** *Targeting attributes in ad requests can be modified before replaying.*

To validate Observation 3, we selected a sample of 50 random apps. For each app, we performed the following three tasks until we got 100 ads from each: (1) launched the app on a mobile device and kept it running while it made periodic ad requests; (2) issued the first ad request of the app (passively harvested as described before) each time the app requested an ad in task 1, from a replay tool running on a desktop PC; (3) replayed the ad request 100 times after a week. We also wrote parsers for the ads returned by these requests, to determine if responses from the ad requests were valid.

We observed that all replay requests (including the ones replayed a week later) returned valid ads and on average, the overlap between ads received by apps and real-time replay was $> 82\%$. This implies that ad networks treat requests from the app and from the replay tool in a similar way. This also suggests that passively harvested ad requests can be quickly replayed from multiple desktops to harvest a large number of ads.

Observation 4 is motivated by the fact that most targeting attributes appear in plain text in ad requests. Ad providers do not typically encrypt ad requests because of the extra overhead that is required on the ad server to support a large number of SSL connections [26]. To validate Observation 4, we took ad requests from the previous experiment. For each ad request, we created a modified request by changing the value of one random targeting attribute, or dropping it if the attribute is optional. For example, if the original request contains `gender=male`, the modified request contained `gender=female`. We then replayed both versions of the requests 50 times each and observed that ad networks responded to the modified requests with valid ads (i.e., they could be successfully parsed), validating Observation 4.

4.1 Active Probing in MAdScope

Figure 2 shows the overall architecture of MAdScope and its active probing components. The key component is a replay tool that uses insights from the above results to actively probe an ad network: It starts with ad requests passively harvested from real apps, selectively modifies various targeting attributes in the HTTP GET/POST requests, replays the modified HTTP requests with the same request header as the original ad request, and analyzes the responses (i.e., ads). What attributes in ad requests are modified depends on what targeting behavior to emulate.

Other than these basic components, MAdScope has several additional components that are crucial for understanding ad targeting. They involve how ad networks are probed, how targeted ads are identified, and how ads are classified. We describe them next, starting with the probing mechanism to understand targeting.

4.1.1 Probing for Attribute-based Targeting

From MAdScope's perspective, there are two broad categories of targeting. *Attribute-based* targeting is stateless where ad networks select ads solely based on various attribute-values within ad requests. Examples include targeting based on keywords (contextual ads), location, demographic information encoded in ad requests. *Behavioral targeting*, on the other hand, is stateful where ad networks select ads based on historical behavior of the user, e.g., what types of ads he clicked on and what types of apps he used in the recent past. We start with attribute-based targeting.

To construct ad requests with targeting enabled for category C to an ad network N, MAdScope samples an ad request to N from its passively harvested pool of ad requests. It then consults its targeting dictionary to identify targeting attributes of category C in the request and discards any other optional targeting attributes from it (to isolate the impact of C from other targeting categories). Finally it enumerates a few values of C, possibly sampled from passively harvested ad requests, and for each value, it produces one modified ad request by setting the attribute of category C to that value (other attributes keep the same value across all requests). The ad requests are then replayed by MAdScope. To avoid any potential impact of behavioural targeting based on device ID, different ad requests are issued with different device IDs.

For instance, to construct ad requests to adfonic.net with location-based targeting enabled, MAdScope samples an ad request with hostname adfonic.net from its passively harvested pool of ad requests. Using the targeting dictionary described in Section 3.2.4, it identifies location attributes `u.latitude`, `u.longitude`, and `u.postalCode` and discards other optional targeting attributes such as `u.gender`, `u.ageLow`, and `u.ageHigh` from the request. It then produces multiple requests by setting values of location attributes to latitude/longitude and postal codes of top US cities.

In a very small number of cases, however, we found that targeting values (e.g., location in ad requests to googleads.g.doubleclick.net) appear encrypted or hashed. For them, MAdScope uses encrypted/hashed targeting values from its passively harvested request pool. For example, to harvest encrypted location values for googleads.g.doubleclick.net, MAdScope uses the *mock location* feature of popular mobile OS such as Android, iOS, and Windows Phone that allows MAdScope to set the location of the device to a target location before harvesting ad requests. MAdScope then identifies encrypted location attributes in those harvested ads and uses them in ad requests to be replayed.

Note that the above procedure tries to identify the effect of one category of targeting attribute at a time, in isolation from other optional attributes. In practice, it is possible for an ad network to take targeting decisions based on multiple, rather than a single, categories of attributes. For example, an ad network may target women in a specific city, but such targeting would not be revealed by probing with only gender or with only location attributes. Identifying impact of combinations of attributes is challenging due to an exponential number of combinations. Recent work has shown how to explore such combinations in a scalable way [16]; incorporating such techniques to MAdScope is part of our future work.

4.1.2 Probing for Behavioral Targeting

Unlike attribute-based targeting, behavioral targeting is done based on a user's profile, representing his historical behavior such as types of apps he uses or ads he clicks on. MAdScope can automatically build profiles of specific interests and preferences and can issue ad requests with those profiles.

There are several ways to create user profiles. One can mimic actions of a particular user (e.g., [3]). Characteristics of profiles built in this fashion can be arbitrary close to the profiles observed in reality. However, the approach is not scalable. MAdScope instead generates user location- and interest-based profiles by mimicking actions that reflect the target location and interest. (A similar approach was used in [1] to create profiles for in-browser ads.) The strategy enables MAdScope to generate profiles for virtually *any* location and interest.

Given a target location L and interest I, we would like MAdScope to mimic various user actions that are related to L and I and that an ad network may observe, e.g., through targeting attribute values in ad requests or by correlating app and web usage data. More specifically, given L and I, we would like MAdScope to emulate the following tasks for an extended period of time.

1. Set the user's mobile device's location to L before making any ad request.

2. Use various apps related to interest I only. Apps related to interest I are chosen by matching I with Google Play app category [27].

3. Browse various web sites related to interest I. Web sites related to interest I are chosen by matching I to website categorization service such as Alexa[5] [1]. An ad network may correlate web and app usage to deliver ads to app based on the type of web page visited [21].

4. Click on ads of category I. Category of an ad is determined by a classifier we describe later.

By emulating the above tasks, MAdScope lets ad networks observe (a subset of) the activities and build a user's profile related to L and I. Step (1) is needed to make sure that an ad network can infer L as a significant location of the user and show location-based ad for L even when he is not at L. Steps (2), (3), and (4) are needed because they reflect user's interest, which an ad network can use to target ad. Among them (4) is particularly important since a click is a strong signal for a user's need or interest.

Interestingly, MAdScope can perform actions (1), (2), and (3) easily with its replay tool, without using any real device. Since we only care about the interactions between apps and ad networks, instead of actually launching and using the target apps, MAdScope can simply replay ad requests made by those apps. To enforce location L and interest I, MAdScope replays ad requests harvested from apps of category I with location attributes set to L. Finally, it repeatedly visits top Alexa web sites of category I.

However, it is challenging to faithfully emulate clicks only on ads relevant to the profile category due to the following *click contamination* problem.

▶ **Click Contamination.** While building a user's profile of interest category I, MAdScope receives a sequence of ads, of which it must click on all and only ads of category I. Clicking on ads not related to I may contaminate the profile (i.e., confuse the ad networks about user's true interest).

Category of an ad may be determined based on its textual content. However, many ads display images and videos and contain no or very little texts and hence cannot be classified with sufficient confidence. A more foolproof strategy is to classify based on content or Alexa category of the landing page of the ad (the page that a click on the ad leads to). Unfortunately, landing pages are not often mentioned in the ad itself; rather the ad contains a redirection URL, a click on which is redirected through an analytics server, typically used for accounting purpose, to a final landing page. Thus, to determine landing page of an ad, we need to click on it

[5] http://www.alexa.com

$$\underbrace{\texttt{http://ads.lfstmedia.com/click}}_{RedirectionServer}\underbrace{\texttt{/cmp10183}}_{AdId}\underbrace{\texttt{/XXX/2?\_ads=XXX\&adkey=XXX\&\_androididsha1=XXX\&slot=XXX\&impts=XXX\&...}}_{DeviceParams}$$

Figure 7: Anatomy of a click URL of a MobClix ad. XXX denotes encrypted and numeric values.

even if we later find that its category does not match the profile category. This can contaminate the profile.

To address this, we use the following mechanism.

► **Profile isolation via a dummy profile.** MAdScope uses a dummy profile whose only purpose is to click on ads in order to determine their landing page. A redirecting click url consists of three components shown in Figure 7: a Redirection Server that receives the click and eventually redirects it to advertiser's landing page, Ad Id that maps the ad to landing page and that the redirection server uses to decide which landing page to redirect to, and Device Params that contains various device-specific information that redirection server uses to identify the source of the click. The same ad shown in two different devices have the same redirection server and ad id, but different device attributes.

We utilize the above structure of click urls to click an ad without contaminating the profile. Given an ad request r, we issue it twice, once with the original profile and again with the dummy profile, to receive ads a and a' respectively. By parsing the ads, we identify their click urls u and u'. We then generate a click url u'' by concatenating the Redirection Server and the Ad Id from u but Device Params from u' and click (i.e., issue an HTTP GET request to) it. The effect is a click on the ad a, but pretending the source of the click to be the dummy profile, which gives us the landing page without any click from the original profile.

4.1.3 *Ad Classifier*

MAdScope provides a classifier that given an ad can determine its Alexa category[6]. This classifier is useful for various analysis and for building behavioral profile.

Category of an ad can be determined based on its landing page or its content. To determine category based on the ad's landing page, we rely on multiple sources, since we did not find an authoritative source that would assign a category to all of the landing pages in our data set. We used site categorization services from Alexa and WebPulse[7] from Blue Coat. However, as we show later, a significant number of the advertisers do not appear in Alexa (or WebPulse), and hence we train a machine learning classifier that predicts the category of a web page based on its content. The same classifier can be used for content-based classification if an ad contains enough textual contents.

► **Classification features.** For each top-level Alexa category, we consider top 1000 web sites. For each site, we take English words appearing in its title and keyword metadata in its HTML document header and discard frequent words such as 'the' and 'and'. We then normalize each word by stemming it[8]. Finally, for each category, we construct a "bag of words" with all stemmed words and their 2-grams in all 1000 web sites of that category. The bag is used as features for the category.

► **Classification algorithm.** Given the feature sets for all categories, we train a multi-class logistic regression classifier [25], which is commonly used for text classification. We use a min-max normalizer for normalizing features, 20 random sweeps with L1 and L2 weights of 0, 0, 1, and 1 (for a linear combination of L1 and L2 regularization), and optimization tolerances of 10^{-4} and 10^{-7}.

► **Classification accuracy.** To evaluate the performance of the classifier, we apply cross validation to the ground truth. We split the ground truth into 3 folds, train a classifier on 2 of those folds and then evaluate its performance on the remaining fold. Overall, the classifier has an average accuracy of 0.76 across all categories. The Log-loss reduction or the Reduction in Information Gain (RIG) is 78%, which can be interpreted as the advantage of the classifier—the probability of a correct prediction is 78% better than random guessing.

4.1.4 *Identifying Targeted Ads*

To identify whether an ad is targeted, we use the intuition that a targeted ad is shown more frequently to some specific value of a targeting attribute than others. For each ad, we count how many times it is been shown to different values of a targeting attribute. This gives us empirical distribution of the ad over all targeting values. If the ad is not targeted to specific values, it is fair to assume that the observed distribution should be close to uniform. To compare the observed distribution with uniform distribution, we use Pearson's χ^2 test with the null hypothesis being "observed distribution is uniform". The test involves computing p-value, whose value is small (less than 0.05) if the null hypothesis is rejected and close to 1 if the hypothesis is accepted. We say that an ad is targeted if the p-value of its observed distribution is less than 0.05.

The statistical analysis is meaningful only for ads that are shown at least a few times; we use a threshold of 10.

4.2 Limitations

Before we present our results, we would like to point out a few limitations of our methodology to set the expectation right. Targeting behaviour of an ad network may depend on many variabilities such as its available ad inventory, demands of ads, its targeting algorithm which may evolve over time, time of the day, etc. Although our attribute-based targeting results are based on a week long probe, the window may not be sufficiently large to capture all variabilities. Our results are based on a small number of targeting values (e.g., locations of five big US cities); it is possible that an ad network employs custom targeting strategies for specific values which we might miss. For example, if an ad network used a different location-based targeting algorithms for USA and China, we would miss its behavior for China since we used USA locations only.

[6]http://www.alexa.com/topsites/category

[7]http://sitereview.bluecoat.com/sitereview.jsp

[8]Stemming is the process for reducing inflected (or sometimes derived) words to their base or root form; e.g., "photos" to "photo".

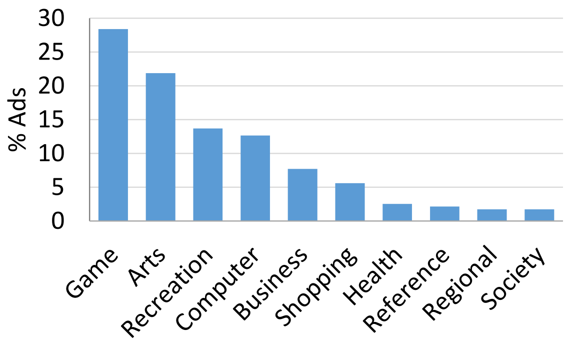


Figure 8: Distribution of ads over top ten Alexa categories

It is also possible that we failed to identify behavioral targeting of some ad network. This can happen for a few reasons. First, we use a window of one day to build profiles, after which we test for behavioral targeting. It is possible that our profile-building window of one day is too small to capture behavioral targeting of the network (although it is unlikely based on results from [1]). Second, the ad network may rely on other actions than what MAdScope emulates (Section 4.1.2) to build behavioral profiles. Third, MAdScope's ad classifier is not perfect, and hence it can click on unrelated ads to contaminate profiles. The ad network under study may be sensitive to such contaminated profiles.

4.3 Empirical Results

4.3.1 *Ads and Advertisers*

We start with some results about ads and advertisers, which are based on ad impressions received by replaying 100,000 randomly selected ad requests from our datasets.

► **Ad serving rate.** Ad serving rate indicates the fraction of ad requests returning valid ads. We say an ad network fails to return an ad when in response to an ad request, it returns an unsuccessful HTTP response (e.g., error 503: Service Unavailable) or an empty ad (likely due to lack of enough ads to serve).

We observed various degrees of ad serving rates for different ad networks. While most of the ad networks we tried had good ($> 95\%$) ad serving rate, we experienced poor serving rates for two ad networks: adsmogo.com and mobclix.com. Ad serving rates for these two networks were 43% and 81% respectively. Such a poor ad serving rate is bad for app developers since they waste opportunities to show ads to users.

► **Advertiser rank analysis.** We observed a total of 4037 unique advertisers. We determined their ranks by using Alexa's top 1 million web sites. About 55% of the advertisers appear in Alexa's top 1 million web sites, with an average ranking of 217K. Remaining 45% of the advertisers do not appear in Alexa's top 1 million web sites and hence we could not determine their global web rank. Ads from these unranked constitute 75% of the total ads in our corpus. This shows that most of the mobile ads come from obscure advertisers who are not among the top 1 million web sites listed by Alexa.

► **Ad categories.** We categorized 1734 of 4037 advertisers using Alexa, 836 using WebPulse, and the rest with our classifier. Figure 8 shows ad distribution for the top 20 root level Alexa categories. The top advertiser category is Game, where the landing page is a Google Play page for downloading Android games. Some other major categories are Arts, Recreation, Computer, Business, and Shopping.

4.3.2 *Attribute-based Targeting Analysis*

To characterize attribute-based targeting of ad networks, we use MAdScope to actively probe top ten ad networks in our dataset. To capture temporal variability, each ad network was probed once every two hours, over a period of a week starting from October 10, 2014. To probe an ad network N, MAdScope first uses the targeting dictionary to identify categories of targeting information collected by N. The top ad network, doubleclick.net, allows targeting based on six different categories of attributes: location, keywords, device, demographics, child flag, and app name. Other networks allow a subset of these categories. We identified a total of 26 {ad network, targeting category} combinations. For each network N and category C, MAdScope then creates up to 5 ad requests, with different values of targeting attributes of category C plus one request without any optional attribute of category C. (Note that MAdScope isolates C by omitting targeting attributes of other categories in ad requests.) Each ad request is then issued 100 times and the responses are captured. The overall process generated slightly over a million ad impressions over a week.

► **Utilization of targeting attributes.** Our key results on attribute-based targeting are shown in Table 3 that shows how effectively ad networks utilize various targeting information for targeting users. We compare ads received with different targeting values. For each {ad network, targeting category} combination, we calculate the fraction of ads whose distribution across targeting values does not follow a uniform distribution with any statistical significance (i.e., p-value of the χ^2 test is less than 0.05). The closer the value to 1, the more targeted the ad. A value of 1 means 100% of the ads are distributed in a non-uniform way (i.e., they are targeted) across all targeting values. The same metric was used in [1] and [27] to quantify targeted ads.

We observed many instances where targeting information has statistically significant impact on how an ad network selects ads (shown by cells with values close to 1 in Table 3). For example, we found doubleclick.net to select ads based on targeting keywords every time such keywords are available, while it targets users based on their locations 80% of the cases when location information is available.

We also observed many instances where ad networks collect many different types of targeting information, but do not use the information to target users. This is shown as cells in Table 3 with a zero values. For example, even though appsgeyser.com allows apps to send location information, the information does not have a statistically significant impact on how the network selects ads.

• **Location.** We expected location-based ads to be common and location information to be utilized by all top ad networks. Surprisingly, three of the seven ad networks that collect location information do not utilize it for targeting user. The fourth network, applovin.com, utilizes location information only marginally.

• **Keywords.** We found three networks to accept targeting keywords and two of them to actually use them.

Ad network	Location	Keywords	Device	Demographics	App	Child flag
doubleclick.net	0.80	1.00	0	0.17	1.00	0
mobclix.com	×	0	0.10	×	×	×
appsgeyser.com	0	×	×	×	1.00	×
adsmogo.com	0	×	×	×	0	×
wapx.cn	0.08	×	0	×	0	×
nend.net	1.00	×	0	×	0.23	×
mopub.com	×	1.00	0	×	×	×
mobfox.com	×	×	0	×	0	×
applovin.com	0.25	×	0.3	×	1.00	×
airpush.com	0	×	×	×	0.75	×

Table 3: Ratio of ads shown by different (ad network, targeting category) combinations whose distribution across profiles does not follow uniform distribution with statistical significance. A cell marked with × indicates that the ad network does not collect targeting information of that category.

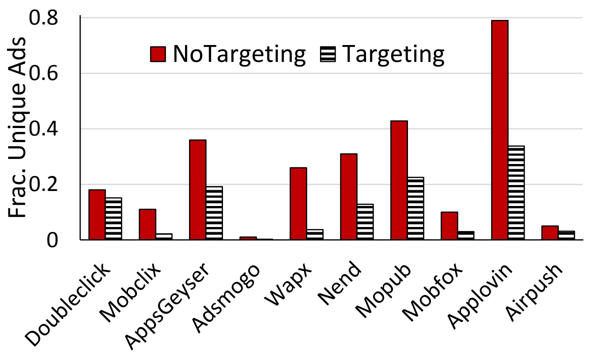


Figure 9: Arrival rate of unique ads. Without targeting, average rate for all top ten ad networks is 26%, with targeting it is 11%.

• **Device.** Most (7 out of 10) ad networks collect device-specific information, but most ignore it for targeting purpose.

• **Demographics and Child flag.** Only one top ad network collects these information, but use them only marginally.

• **App name/ID.** Eight of the top ten ad networks collect app names and only half of them use the information to select ads.

Note that we identified targeting information from ad requests made by real apps and some of the targeting information is optional, which implies that app developers explicitly provide these optional targeting information without getting any perceived targeting benefit. As mentioned in Section 1, ad networks most likely collect such targeting information for forward-compatibility and for information brokerage.

Also note that the results in Table 3 should not be interpreted as a way to decide whether one ad network is necessarily "better" than another. An ad network may not collect or effectively use targeting information, but yet it can be lucrative to developers in terms of payouts and to advertisers in terms of conversion rates (and vice versa). Evaluating ad networks in terms of these metrics is part of our future work.

▶ **Impact of targeting on average distinct ad arrival rate.** Figure 9 shows the fraction of distinct ads of all ads harvested from each of the top ten ad providers. Two ads are considered distinct if they have the same text or image content and have the same click url. For each ad network, we report the fraction of distinct ads for two scenarios: when targeting is enabled and when targeting is disabled (i.e., when ad requests do not contain any targeting attributes).

Figure 9 shows several key points. First, all ad networks serve a large fraction of duplicate ads. On average, only 26% of the ads were distinct, suggesting that ad networks could benefit from caching ads on the client, as suggested by prior works [15, 19, 27]. The fraction of distinct ads vary widely across ad networks, ranging from 1% to 80%.

Second, targeting reduces the fraction of distinct ads. We observed this for all ad networks. When averaged over all ten top ad networks, the average fractions of distinct ads reduces from 26% to 11% when targeting is enabled. This is most likely because the pool of distinct ads applicable to specific targeting scenario is substantially smaller than those applicable to any or no targeting. This suggests that developers are likely to see more distinct ads when they do not overly target their ad requests. This might be one reason of why many developers do not provide optional targeting attributes even if ad networks allow them (Figure 6).

4.3.3 Behavioral Targeting Analysis

We use the following two-step methodology to detect if top ten ad networks use behavioral targeting. In the training step, we let MAdScope create five profiles with (1) different device ids, (2) different locations L: New York, Los Angeles, Houston, Atlanta, and Seattle (five big cities in USA), and (3) different interests I: Game, Arts, Recreation, Computer, and Business (top five ad categories shown in Figure 8). We then use MAdScope to issue 5000 ad requests for each profile, over a period of one day. Then in the testing step, we use MAdScope to issue 1000 ad requests for each profile, with the ad requests containing only device information but no location or specific interest information (i.e., we select ad requests from a set of 100 apps from all categories, remove their optional targeting attributes, and replay them for each profile). Intuitively, if the ad network has learnt about the users' interests and locations from the training step and if it does behavioral targeting, it should show different ads to different profiles even though the attributes in ad requests do not indicate any specific location or interest. We use the same χ^2 test to decide if the ads received by different profiles are distributed uniformly across profiles.

We found that doubleclick.net, used by 36% ad requests in our datasets, differentiates between profiles in a statistically significant way while serving ads. We did not observe any statistically significant impact of profiles for other networks, but that could well be due to the limitations mentioned in Section 4.2. For doubleclick.net, the impact of profiles is significant—we found that 80% of the ads were targeted towards specific profiles (e.g., distributed non-uniformly).

Note that in building profiles, MAdScope used four different types of activities mentioned in Section 4.1.2 (e.g., using a specific location L, using apps of a specific category I, etc.) We now investigate which of these tasks affect the profiles actually used by ad networks. To do that, we repeat the previous experiment four times, with MAdScope configured to build profiles based on only one task in each experiment. For example, in the first experiment, MAdScope builds profiles with location L only, but uses apps, ads, and web pages of many different categories. In the second experiment, it uses apps on category I only, but uses many different locations and categories of ads and web pages, and so on.

We observed that for doubleclick.net, only two types of actions impact targeting: types of used apps and types of clicked ads. Historical locations and web browser usage information do not seem to play any role on targeting.

5. COMPARISON OF WEB AND APP

We here contrast our findings for in-app ads with that from a recent study on in-browser ads [1].

▶ **Advertiser category.** Top categories of advertiser seem to be different for in-app ads than for in-browser ads. In [1], authors report that top three advertiser categories for in-browser ads are Financial Services, Shopping, and Computers. In contrast, our study shows that top thee categories for in-app ads are Games (mostly games in app stores), Arts (e.g., wallpapers), Recreation (e.g., ring tones). This probably aligns with general usage of browser and apps: financial services and shopping are mostly browser-based activities, while apps are used mostly for entertainment purposes.

▶ **Behavioral targeting.** As authors in [1] reports, behavioral targeting is common in in-browser ads. We used a conceptually similar methodology, but observed such targeting for only one of the top ten in-app ad networks.

▶ **Demographics.** Demographic information is a common targeting attribute in in-browser ads [1]. In contrast, it is not commonly used in in-app ads. Only one of top ten ad networks supports targeting based on demographic information; but this it an optional attribute and many app developers omit the attribute. Even if app developers provide demographic information of the user, the top ad network use it only marginally to target users.

▶ **Location.** Authors in [1] did not report location as a major targeting attribute in in-browser ads. In contrast, most (seven out of top ten) in-app ad networks collect location information, and many use the information to target users.

6. RELATED WORK

▶ **In-app ads.** A few studies characterize various aspects of in-app ad traffic. In [28], authors analyzed a data set corresponding to one day of traffic for a major European mobile carrier with more than 3 million subscribers and characterize mobile ad traffic along a number of dimensions, such as overall traffic, frequency, as well as possible implications in terms of benefits of well-known techniques, such as pre-fetching and caching, to limit the energy and network signalling overhead caused by current systems. CAMEO [15] analyzed AdMob ads collected from 100+ PlanetLab nodes and 20 phones and showed feasibility of caching and prefetching of ads. Mohan et al. [19] also show a similar result. (A recent study [4], however, showed that the power savings due to ad prefetching is small.) Unlike our work, these works do not characterize targeted ads. The only work we are aware of that aims to characterize user targeting in in-app ads is a study by Ullah et al. [27], which shows various categories of targeted ads shown to 100 apps by one ad network. Our study is more comprehensive and it differs from this previous study in its (1) scale (we analyze ad requests from 150K apps and 101 ad networks, while Ullah et al. analyze requests from 100 apps and mostly from one ad network, AdMob), (2) analysis of targeting attributes in ad requests, and (3) active probing (we probe top ten ad networks to understand what targeting attribute they actually use for targeting).

Most prior work on in-app ads focus primarily on security and privacy. Grace et al. [12] use their automated tool called AdRisk to analyze 100 ad libraries and investigate potential privacy and security risks of these libraries involving their resource permissions, dynamic code loading, permission probing and JavaScript linkages. Concurrent to this work, Stevens et al. [26] also investigate various privacy vulnerabilities in the most popular Android advertising libraries. They point out whether permissions required by various ad libraries are required, optional, or undocumented and investigate privacy concerns such as how UDIDs are handled and what user data is sent in ad requests. Several other works acknowledged the dangers of the lack privilege separation between Android application and and ad code and propose methods of separating them [20,24]. Leontiadis et al. [17] analyze an impressive number of 250K Android apps and show that current privacy protection mechanisms are not effective as developers and advert companies are not deterred and propose a market-aware privacy protection framework that aims to achieve an equilibrium between the developer's revenue and the user's privacy. The primary focus of all these papers is privacy and security issues of in-app ads, while ours is on characterization of user targeting in in-app ad landscape.

▶ **In-browser ads.** Previous work on online advertising has focused primarily on in-browser advertising and most work has considered the problems associated with privacy [7,11]. The study by Guha et al. [13] describes challenges in measuring online advertising systems. Roesner et al. [23] present a taxonomy of different trackers i.e., in-site, cross site, cookie sharing, and social media trackers, demonstrate how prevalent tracking is, and propose an extension that helps to protect user privacy. Castelluccia et al. [3] demonstrate that it is possible to reverse engineer a user's profiles and interests by analyzing ads targeted to her. Another thread of previous work investigated ad fraud, which hurts the advertisers who are paying for the ads [10]. XRay [16] is a differential correlation technique to identify how various tracking information are being used by targeted ads. MAdScope's overall goal is similar, but for in-app ads. More relevant to our study is a recent study of online in-browser display advertising by Barford et al. [1], where authors provided a general understanding of the characteristics and dynamics of online display advertising and insights on the targeting mechanisms that are used by ad serving entities. Our goal is similar, but for in-app advertising. As mentioned before, in-app advertising works differently from in-browser advertising and our results indicate that they significantly differ in important characteristics.

► **Dynamic analysis of apps.** MAdScope uses a Monkey to passively harvest ads. Similar Monkeys have been used recently for other applications such as app testing [22], ad frauds detection [5, 18], but none of these work studies user targeting by in-app ads.

7. CONCLUSION

We reported an in-depth study that seeks to broadly understand what targeting information mobile apps send to ad networks and how effectively, if at all, ad networks utilize the information for targeting purpose. Using a novel tool, called MAdScope, we analyzed 500K ad requests from 150K Android apps and showed that targeting is limited in in-app ads: even though ad controls provide APIs to send a lot of information to ad networks, much key targeting information is optional and is often not provided by apps. We also used MAdScope to systematically probe top 10 in-app ad networks to harvest over 1 million ads and found that while targeting is used by many of the top networks, there remain many instances where targeting information or behavioral profile does not have a statistically significant impact on how ads are chosen. We also contrasted our findings with a recent study of targeted in-browser ads.

In-app ad ecosystem is evolving and this study represents an important snapshot. It will be interesting to see how in-app targeting evolves in a few years. MAdScope can be a valuable tool for such future studies.

Acknowledgements

The author would like to thank the shepherd, Ben Greenstein, and anonymous reviewers of ACM MobiSys 2015 for their feedback, Shuai Hao for developing the PUMA tool; Hao Chen and Jon Crussell for sharing the MadFraud dataset.

8. REFERENCES

[1] P. Barford, I. Canadi, D. Krushevskaja, Q. Ma, and S. Muthukrishnan. AdScape: harvesting and analyzing online display ads. In *Int. conf. on WWW*, 2014.

[2] R. Bhoraskar, S. Han, J. Jeon, T. Azim, S. Chen, J. Jung, S. Nath, R. Wang, and D. Wetherall. Brahmastra: Driving apps to test the security of third-party components. In *USENIX Security*, 2014.

[3] C. Castelluccia, M. A. Kaafar, and T. Minh-Dung. Betrayed by Your Ads! In *PETS- Privacy Enhancing Tools Symposium*, 2012.

[4] X. Chen, A. Jindal, and Y. C. Hu. How much energy can we save from prefetching ads?: energy drain analysis of top 100 apps. In *Proceedings of the Workshop on Power-Aware Computing and Systems (HotPower)*. ACM, 2013.

[5] J. Crussell, R. Stevens, and H. Chen. MAdFraud: Investigating ad fraud in android applications. In *ACM MobiSys*, 2014.

[6] T. Danny. Flurry releases newest app use stats. http://tech.co/flurry-app-stats-2014-09.

[7] D. S. Evans. The online advertising industry: Economics, evolution, and privacy. In *Journal of Economic Perspectives*, 2009.

[8] A. Farahat and M. C. Bailey. How effective is targeted advertising? In *Int. conf. on WWW*, 2012.

[9] GameHouse. Mobile advertising statistics—5 big trends you need to know! http://partners.gamehouse.com/mobile-advertising-statistics-5-big-trends-need-know/.

[10] M. Gandhi, M. Jakobsson, and J. Ratkiewicz. Badvertisements: Stealthy click-fraud with unwitting accessories. *Online Fraud, Part I Journal of Digital Forensic Practice*, 1(2), 2006.

[11] A. Goldfarb and C. Tucker. Online display advertising: Targeting and obtrusiveness. In *Marketing Science*, 2010.

[12] M. Grace, W. Zhou, X. Jiang, and A. Sadeghi. Unsafe exposure analysis of mobile in-app advertisements. In *Conference on Security and Privacy in Wireless and Mobile Networks (WiSEC)*, 2012.

[13] S. Guha, B. Cheng, and P. Francis. Challenges in measuring online advertising systems. In *ACM SIGCOMM Internet Measurement Conference*, 2010.

[14] S. Hao, B. Liu, S. Nath, W. G. Halfond, and R. Govindan. PUMA: Programmable ui-automation for large scale dynamic analysis of mobile apps. In *ACM MobiSys*, 2014.

[15] A. J. Khan, K. Jayarajah, D. Han, A. Misra, R. Balan, and S. Seshan. CAMEO: A middleware for mobile advertisement delivery. In *ACM MobiSys*, 2013.

[16] M. Lécuyer, G. Ducoffe, F. Lan, A. Papancea, T. Petsios, R. Spahn, A. Chaintreau, and R. Geambasu. Xray: Enhancing the web's transparency with differential correlation. In *USENIX Security*, 2014.

[17] I. Leontiadis, C. Efstratiou, M. Picone, and C. Mascolo. Don't kill my ads!: balancing privacy in an ad-supported mobile application market. In *Workshop on Mobile Computing Systems & Applications*, page 2. ACM, 2012.

[18] B. Liu, S. Nath, R. Govindan, and J. Liu. DECAF: Detecting and characterizing ad fraud in mobile apps. In *USENIX NSDI*, 2014.

[19] P. Mohan, S. Nath, and O. Riva. Prefetching mobile ads: Can advertising systems afford it? In *ACM EuroSys*, 2013.

[20] P. Pearce, A. Felt, G. Nunez, and D. Wagner. AdDroid: Privilege separation for applications and advertisers in android. In *Symposium on Information, Computer and Communications Security*, 2012.

[21] T. Peterson. Google tests way to track consumers from mobile browsers to the apps they use. http://adage.com/article/digital/google-tie-mobile-web-app-trackers-ad-targeting/294502/.

[22] L. Ravindranath, S. Nath, J. Padhye, and H. Balakrishnan. Automatic and scalable fault detection for mobile applications. In *ACM MobiSys*, 2014.

[23] F. Roesner, T. Kohno, and D. Wetherall. Betrayed by Your Ads! In *USENIX NSDI*, 2012.

[24] S. Shekhar, M. Dietz, and D. S. Wallach. Adsplit: Separating smartphone advertising from applications. In *USENIX Security Symposium*, 2012.

[25] A. N. Srivastava and M. Sahami. *Text mining: classification, clustering, and applications.* CRC Press, 2010.

[26] R. Stevens, C. Gibler, J. Crussell, , J. Erickson, and H. Chen. Investigating user privacy in android ad libraries. In *IEEE Mobile Security Technologies (MoST)*, 2012.

[27] I. Ullah, R. Boreli, D. Kaafar, and S. Kanhere. Characterising user targeting for in-app mobile ads. In *INFOCOM International Workshop on Security and Privacy in Big Data (BigSecurity 2014)*, 2014.

[28] N. Vallina-Rodriguez, J. Shah, A. Finamore, Y. Grunenberger, H. Haddadi, K. Papagiannaki, and J. Crowcroft. Breaking for commercials: Characterizing mobile advertising. In *ACM SIGCOMM Internet Measurement Conference*, 2012.

[29] W3C. Same origin policy. http://www.w3.org/Security/wiki/Same_Origin_Policy.

[30] J. Yan, N. Liu, G. Wang, W. Zhang, Y. Jiang, and Z. Chen. How much can behavioral targeting help online advertising? In *Int. conf. on WWW*, 2009.