Revisiting Mobile Advertising Threats with MAdLife

Gong Chen
Georgia Institute of Technology
gchen@ece.gatech.edu

Wei Meng
Chinese University of Hong Kong
wei@cse.cuhk.edu.hk

John A. Copeland
Georgia Institute of Technology
jcopeland@ece.gatech.edu

ABSTRACT

Online advertising is one of the primary sources of funding for content, services, and applications on both web and mobile platforms. Mobile in-app advertisements are implemented on top of existing web technologies with the same ad-serving model (i.e., users - publishers - ad networks - advertisers). Even so, in-app advertising is different from traditional web advertising. For example, malicious mobile app developers can generate fraudulent ad clicks in an automated fashion, but malicious web publishers have to leverage bots to launch click fraud. In spite of using the same underlying web infrastructure, these ad threats behave differently on different platforms.

Existing works have separately studied click fraud and malvertising in mobile settings. However, it is not known if there is a strong relationship between these two dominant threats. In this paper, we develop an ad collection framework – MAdLife– on Android to capture all in-app ad traffic generated during each ad's entire lifespan. We revisit both threats in a fine-grained manner with MAdLife to determine the relationship between them. Furthermore, MAdLife also allows us to explore other threats related to landing pages.

We analyzed 5.7K Android apps crawled from Google Play store, and collected 83K ads and their landing pages with MAdLife. 58K ads land on a web page, which is similar to traditional web ads. We discovered 37 click-fraud apps, and revealed that 1.49% of the 58K ads are related to malvertising. We also found a strong correlation between fraudulent apps and malicious ads. Specifically, over 15.44% of malicious ads originate from the fraudulent apps. Conversely, 18.36% of the ads displayed in the fraudulent apps are malicious, compared to only 1.28% found in the rest apps. Due to fraudulent apps, users are much more (14x) likely to encounter malvertising ads. Additionally, we discovered 243 popular JavaScript snippets embedded by over 10% of the landing pages are malicious. Finally, we also present the first analysis on inappropriate mobile in-app ads.

CCS CONCEPTS

• **Information systems → Online advertising**; • **Security and privacy → Software and application security**.

KEYWORDS

Online Advertising; Mobile Apps; Ad Fraud; Malvertising; Measurement

ACM ISBN 978-1-4503-6674-8/19/05.
https://doi.org/10.1145/3308558.4089729

ACM Reference Format:
Gong Chen, Wei Meng, and John A. Copeland. 2019. Revisiting Mobile Advertising Threats with MAdLife. In *Proceedings of the 2019 World Wide Web Conference (WWW'19), May 13–17, 2019, San Francisco, CA, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3308558.4089729

1 INTRODUCTION

Online digital advertising is the primary way to monetize content, services, and applications (apps) on the Internet. Companies like Google and Facebook generated 85-95% of their revenues through running the largest digital advertising platforms in Q2 2018 [1, 21]. Traditionally, these ad platforms, or ad networks, display advertisements (ads) from advertisers on a publisher's website that is part of the network. This practice is known as web advertising, which has generated over \$35B in revenue per year in the last five years according to recent IAB reports [28]. Meanwhile, web advertising has rapidly expanded to mobile platforms. Mobile advertising revenue experienced a substantial (7x) growth from \$7.1B in 2013 to \$49.9B in 2017.

Mobile in-app ads are usually implemented on top of existing web technologies by rendering an HTML ad within a mobile app's WebView. Although sharing many underlying technologies, mobile in-app advertising is different from traditional web advertising. A website includes a JavaScript snippet to display ads, whereas a mobile app embeds a custom SDK to load the JavaScript code and ads in a special WebView – AdView. To date, ad blocking solutions, such as DNS66[1], DISCONNECT[2], NoMoAds [43], [6], are not widely used by real users for in-app ads, but ad blockers are very popular for traditional web advertising. As a result, the abusive practices in web advertising, such as *click fraud* and *malvertising*, exhibit different characteristics on mobile platforms. In particular, a web browser allows an ad script to detect automatically generated fraudulent ad clicks. Therefore, malicious publishers have to leverage bots instead of real users to automate ad clicks, which can be easily detected by ad networks. On the contrary, mobile users need to interact with HTML ads through their app's user interface (UI). This enables a malicious app developer to automatically generate fraudulent ad clicks, which are difficult for the restricted JavaScript running in the AdView to detect. Therefore, it is easier to launch click fraud from a genuine user device with the help of UI automation on mobile platforms. Malvertising on both platforms behaves very similarly. By clicking on an ad, a user may be brought to a malicious page that serves drive-by-downloads or scams. In particular, drive-by downloads have been the primary form of malvertising for web advertising, according to the study in [31]. Scams are, however, the dominating malvertising form for mobile in-app ads [39].

The abusive practices in mobile advertising have been explored by prior works. Crussell et al. examined click fraud in each app's

[1]https://f-droid.org/en/packages/org.jak_linux.dns66/
[2]https://disconnect.me

first/main activity by running in an emulator without UI intervention [15]. However, their work did not reveal how the fraudulent apps initiate automatic ad clicks. Rastogi et al. [39] detected malvertising cases by collecting and scanning redirect chains using VirusTotal[3]. However, their work did not look into JavaScript code loaded from external sources, which can also be malicious. Overall, both frameworks are time consuming for app collection. Also, click fraud and malvertising may happen together, and these tools cannot be used to study both threats altogether.

To overcome the above limitations, we present MAdLife, a framework for studying mobile in-app advertising. It can record all necessary data generated during each ad's entire lifespan (i.e., request → load/render → click → redirect → land) on Android. Specifically, we modify the Android WebView to collect pre-click data. After each ad is completed, we automatically trigger a click through lightweight UI automation, and then redirect the post-click traffic to our data collection app. The two datasets allow us to conduct a more fine-grained study on click fraud and malvertising. We can also investigate other related threats in mobile in-app advertising. Usually, advertisers' landing pages embed external JavaScript code. Unlike the prior work on mobile malvertising, we reveal that the external JavaScript snippets can also be malicious even though the embedding landing pages are considered benign. Besides, following up with two trending news event related to political advertising on Facebook [13, 41], we recognize that the contents appear in advertising have been receiving more and more public concerns. Thus, we also take a first attempt to analyze inappropriate content for mobile in-app ads. Specifically, by employing content analysis techniques, we are able to detect and classify inappropriate ads into two categories: 1) policy non-compliant [20, 24, 25, 46], which are ads that do not comply with the policies of ad networks (e.g., including age-restricted content); and 2) controversial, which are ads that are generally believed inappropriate to some population groups (e.g., promoting online gambling).

We used MAdLife to collect 84K ads from 5.7K Android apps from January to February in 2018. First, we identified that 37 apps had committed click fraud, and 1.49% of ads were malicious according to the VirusTotal query results on their landing pages. Second, we discovered a strong statistical relation between click-fraud apps and malicious ads. On one hand, over 15.44% of infected ads originated from those fraudulent apps. On the other hand, 18.36% of the ads displayed by click-fraud apps are malicious, whereas in non click-fraud apps only 1.25% of ads are malicious. Therefore, users of the click-fraud apps are much more (14x) likely to be exposed to malicious ads than other users. Also, we found over 32 cryptojacking ads in our dataset. Moreover, we discovered 243 (0.21%) of the 115K external JavaScript snippets are unsafe. They are very popular and were used in over 10% of the landing pages in our dataset. Finally, we identified that around 8% of ad landing pages were inappropriate – 7.9% are non-compliant with ad policies, and 0.266% are controversial.

Our contributions are threefold:

1) We developed MAdLife– the first framework to monitor an ad' entire life span on the mobile platform. As a result, we are able to build a panoramic view for digital ads.

[3]https://www.virustotal.com

2) We explored the abusive practices involved in mobile in-app advertising, and demonstrated a strong statistical relation between click fraud and malvertising. We discovered that due to click fraud, mobile app users are much more likely to experience malvertising. Moreover, we also found several emerging threats (i.e, cryptojacking ads) in mobile in-app advertising.

3) We are the first to extend the research scope of analyzing landing pages from two aspects. First, we discovered that a few unsafe external JavaScript code may occur in a great number of landing pages. Second, we classified inappropriate ads into two categories (i.e., policy non-compliant, and controversial).

In summary, previous works treated click fraud and malvertising as two independent mobile ad threats, and thus studied them separately. In contrast, MAdLife allows us to study mobile ad threats comprehensively. Therefore, researchers can have a broader view of mobile ad threats and thus build a safer mobile ad ecosystem.

2 BACKGROUND

2.1 Android WebView

Android apps are usually required to include custom AdViews, and the underlying implementation of most AdViews is based on WebView. While transparent to app developers and mobile users, the nature of the WebView component has been changed a lot with the evolution of Android OS. Since Android 4.4, it has moved from using the WebKit rendering engine to sharing the same Chromium-based codebase with Chrome for Android. In Android 5.0, the component came out as a standalone APK, where mobile users can find its updates on Play Store. Starting from Android 6.0, only prebuilt WebView versions are shared within the Android Open Source Project (AOSP). Afterward in Android 7.0, the Chrome APK is used for provide and render WebView. Recently, the Safe Browsing feature has been available in WebView with Android 8.0.

Based on Chromium source tree, WebView depends on a C++-compiled shared library and a source set. Within the source set, Java code encapsulates C++ and provides an AwContents object. By adding logcat messages for function calls within AwContents, we can thus use adb to communicate with an Android emulator and view the device log in real-time.

As a View object that displays webpages, WebView cannot be used standalone without an Activity component. With UI Automation tools, developers can access, identify, and manipulate the UI elements. However, such tools cannot access the Document Object Model (DOM) within a WebView. Unlike ads in a browser where researchers can directly resolve ad URLs from their DOM trees, when an ad WebView receives a touch event, it triggers an AwContents object's `onTouchEvent()` first, and then consecutively tops down to the Blink rendering engine for resolving the embedded URL from its DOM tree.

2.2 Mobile Advertising

Serving in-app ads is a pervasive approach to app monetization for *publishers/developers*. In order to get continuous revenue, app developers register their apps, and app store links to one or several *ad networks*. In turn, the developers receive a unique identifier per ad space, all related *ad libraries*, which fetch ads with the resolved URLs at runtime, and common guidelines. For example, an ad's standard Cascading Style Sheets (CSS) size, which is different from a smartphone's resolution size, could be recommended. While developers have rights to choose ad types, including video ads and display ads (i.e., small-size banner ads, and full-screen interstitial ads) shown on their apps and earn from displaying/clicking ads, *advertisers* sit on the other end to provide funds for spreading out the content they wish to publicize. Together with innocent users downloading and launching apps, all other players, including publishers/developers, ad networks/ad libraries, and advertisers, form a perfect ecosystem in the mobile settings [9].

However, such an ecosystem may temporarily be interrupted, when a user launches an app, the app requests an ad for a vacant space from the linked ad network, but the ad network has nothing in its inventory when the app is launched. As a result, the user may see a void ad slot. In the case that ads are not loaded within the WebView, we call it *incapable ad loading*. Otherwise, different *ad syndications* are created by ad networks in order to fill the gap in time. Therefore, when the subscribed ad network does not find an ad for the ad slot, the ad request will be forwarded to syndicated ad networks. Imaginably, communications between the app and multiple ad networks under the ad syndication lead to a longer *redirect chain*, before arriving at the final *landing page*. Also, an ad may not be properly displayed at a WebView, due to network overload or ad networks' backend algorithms. We call that *improper ad rendering*. As for *app install ads*, their final destination may end up with Play Store.

2.3 Ad Threats

Click fraud. In order to earn more, unscrupulous app developers use click fraud to programmatically trigger fake ad clicks.

Malvertising. Due to untrusted ad networks, or unwanted redirect chains, malvertising may result in either drive-by downloads, or scams and phishing (e.g., pornography).

Vulnerable External JavaScript Code. Usually, advertisers use JavaScript snippets to track user behaviors, or more. However, landing pages containing vulnerable external JavaScript code may be exposed to both security and privacy risks.

Ad Inappropriateness. Ad networks regulate advertised content, but untrusted advertisers can still smuggle inappropriate ads. It is infelicitous, especially for children. That's why Unity Ads [47] talk about COPPA [12] within their common guidelines.

2.4 Public APIs

VirusTotal aggregates over 60 scanners, and offers APIs to analyze files/URLs. A rationale behind using a cluster of antivirus scanners is to fight against mistaken detection by individual scanning tool, or so called *false positive*.

Google Cloud APIs[4] uses REST calls to automate different workflows. We used the Natural Language API to reveal the structure and meaning of text from different aspects (e.g., "entity analysis" and "content classification"). Also, we called the Vision API to find similar images and web entities as well as detect explicit content (i.e., adult, medical, spoof, and violence), respectively.

[4]https://cloud.google.com/apis/

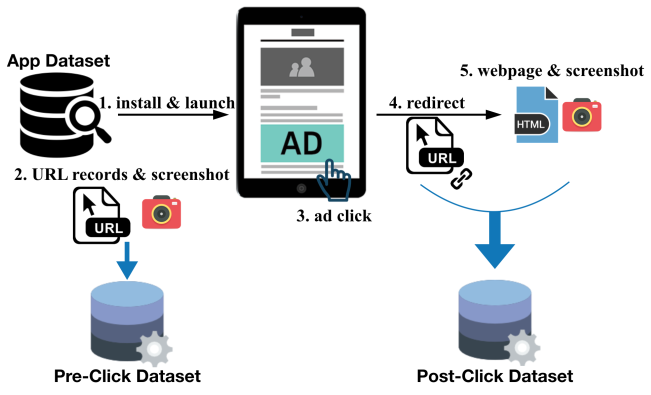


Figure 1: Workflow for data collection

3 METHODOLOGY

In this section, we present the design and implementation of MAdLife (Section 3.1), the steps we took to select our app dataset (Section 3.2), and the statistics about that dataset (Section 3.3).

3.1 MAdLife

MAdLife is a data collection framework on a mobile platform for in-app ads. It can record all necessary data generated within each ad's entire lifespan (i.e., request → load/render → click → redirect → land). Figure 1 demonstrates the workflow of MAdLife. First, MAdLife uses the AndroidViewClient[5] tool (version 13.6.0) to install and launch an app automatically. Afterwards, an ad is returned from a remote host. Second, MAdLife listens to the ad traffic at the same time, and then stores the *pre-click* data into a table in our database. Third, MAdLife clicks the ad. Fourth, the embedded ad URL is sent to our data collection app – Depot, and then redirected to the landing page. Fifth, MAdLife stores the *post-click* data into another table in our database. Finally, MAdLife stops all launched activities after the post-click data is collected. A timeout is set to stop the data collection when no ad is loaded or rendered.

We designed and implemented MAdLife on top of an Android emulator – Genymotion[6] 2.11. We did not select the AOSP emulators and those provided by Android Studio, because apps running in these emulator can receive only test ads for the AdMob SDK, which is the most widely seen ad SDK on Android. Instead, we equipped our testbed with Android 7.1 (API level 25) on Genymotion. After installing Play Store, we set the "Parental Controls" setting to the most restrictive level[7]. We took this step to ensure that we do not reach an age-restricted Play Store page via clicking the ads. We will show later in Section 4.3.3 that age-restricted ads cannot be avoided. Also, we enabled the ARM translator to run ARM apps on an x86 machine.

In order to collect the pre-click data, we modified the Android WebView (20 LOC in Java) to intercept a WebView's network traffic. This allowed us to determine if WebView is used to communicate

[5]https://github.com/dtmilano/AndroidViewClient
[6]https://www.genymotion.com
[7]"Apps & games": Everyone, "Movies": G, "TV": C, and restrictions on: "Books" and "Music".

Table 1: Our collected datasets

Pre-click Data		package name, ad size, ad screenshot, pre-click URLs, pre-click time
Post-click Data	Browser	screenshot of the landing page, HTML source of the landing page, URL redirect chain, post-click time
	Play Store	screenshot of the landing page, maturity rating

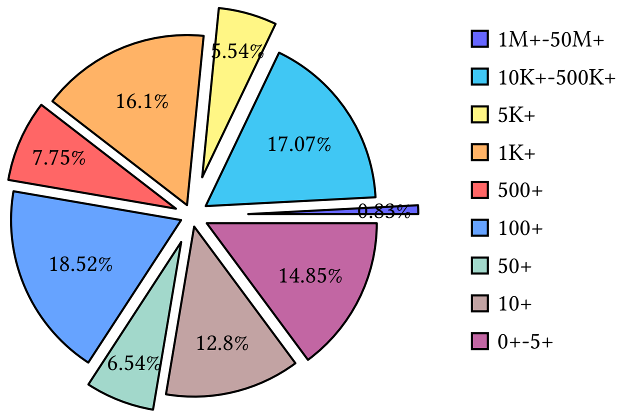


Figure 2: Distribution by app downloads

with a known ad host. After an ad is returned from a remote host, MAdLife saves the ad's screenshot and logs all pre-click URLs when several methods[8] related to ad rendering are called.

In order to collect the post-click data, we replaced each ad navigation URL with a custom URL scheme (i.e., MAdLife://) to direct the navigation to Depot, instead of the default mobile web browser. MAdLife generates an ad click to let the app switch to Depot or Play Store for post-click data collection. In case of switching to Depot, the app will take a screenshot of the landing page, save its HTML source, and records the redirects if there are any. However, it cannot directly identify which app initiates the ad click. To solve this problem, we link the pre-click data and post-click data based on their timestamps. In the case of switching to Play Store, MAdLife will take a screenshot of the Play Store page, and log the advertised app's maturity rating (e.g., Everyone, Teen, Mature 17+).

MAdLife also handles special cases where some ads require users to click on a control button or require more than one click to trigger a redirection. In particular, almost all full-screen interstitial ads require a click on a control button (e.g., "INSTALL", "Visit Site", "Learn More", and "Click Now"). Further, a few banner ads require to click twice, first to display a control button, and second to trigger the redirection. Therefore, MAdLife calls AndroidViewClient's `dump()` to discover the button with keywords, and then programmatically clicks such ads. The record for an ad click consists of two parts, as shown in Table 1.

[8]Methods include `loadUrl()`, `didFinishNavigation()`, `onTouchEvent()`, `didFinishLoad()`, `didStopLoading()`, `shouldIgnoreNavigation()`, `shouldInterceptRequest()`, and `loadDataWithBaseUrl()`.

3.2 App Selection

Initially, we crawled 143K free Play Store apps in December 2016 and March 2017. Due to the fact that not all of these apps have ads, it is not necessary to analyze the entire app dataset. Additionally, we were not able to analyze all of them due to our limited computing power. Thus, we decided to narrow down our analysis space by applying an app selection procedure. Also, it is not practical to analyze all ads displayed within one app, which requires complex UI automation. Given that the subsequent requests are similar to their first ad-related HTTP request for 94.7% of the top 3K ad-supported free apps [36], we focused on the apps that display ads in their first activities.

First, we excluded any apps whose first activities do not contain a WebView, with AndroidViewClient. We also removed apps with multiple WebViews in the first activities, because it cannot distinguish which WebView is clicked for data generation. As a result, 29.3K apps were kept.

Second, we further excluded apps that do not communicate with a known ad domain. In particular, we constructed a list of 1,183 known ad domains[9]. Only 11.8K apps were observed to communicate with one of the domains within 15 seconds. We set this 15-second threshold, since "most apps make the first ad request during launch time", as observed in [36]. However, we discovered that around 50% of the 11.8K apps did not display an ad in their first activity. It might be possible that those requests in the first activities were sent for tracking purposes, and the ads might be shown in other activities.

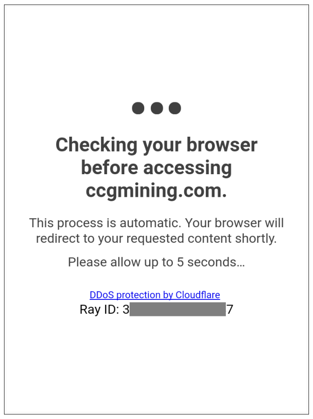
Last, we ruled out apps that do not exhibit any changes in their first activity after the ads are clicked, because users will normally be redirected to either a landing page or the Play Store app after an ad click. We finally retained 5.7K apps. Moreover, we tried the method in [19] to identify ads by using WebView sizes. However, we found that this method may not be robust because we identified more than 300 different WebView sizes in the 11.8K apps' first activities. For example, 320x50 is a standard CSS size for mobile ads, but we also observed ad sizes such as 320x49 and 320x51.

The selected 5.7K apps, developed by over 2.5K developers, exhibit high diversity, in terms of both app categories[10] and number of app downloads (Figure 2). According to [3, 4, 44], we believe that our app dataset is unbiased.

3.3 Ad Collection

We deployed MAdLife in two locations in the United States and one location in Canada between mid January and late February in 2018. According to [36], even though the subsequent requests are similar to their first ad request, ads returned by similar requests can be different. Therefore, in order to make sure our ad collection was unbiased, we crawled ads for each app in 10 runs at a time, of which each takes about 32-45 seconds to complete. It is worth pointing out that, MAdLife is more efficient, as it takes at least 2 minutes for other frameworks [15, 37, 39] to complete. At the end, we collected nearly 48GB of data, representing over 84K ads.

[9]https://adaway.org/hosts.txt, https://filters.adtidy.org/windows/filters/11.txt?id=11
[10]The apps are in 47 categories, including one in "Dating", four in "Casino", and over 300 in "Books & References", "Education", "Entertainment", "Lifestyle", "Music & Audio", "Personalization", and "Tools".


Checking your browser before accessing ccgmining.com.

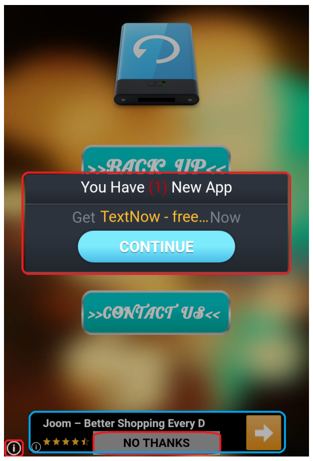
This process is automatic. Your browser will redirect to your requested content shortly.

Please allow up to 5 seconds...

DDoS protection by Cloudflare

Ray ID: 3[redacted]7

(a) Landing page redirect



(b) Overlapped red and blue ads

Figure 3: Special cases encountered during our ad collection

During the ad collection phase, we encountered a great number of special cases (e.g., "incapable ad loading", and "improper ad rendering"). Additionally, we encountered two special situations. First, an ad may have more than one landing page due to *landing page redirect*, as shown in Figure 3(a). Such a redirection would occur after the initial landing page is loaded in Depot. As a result, MAdLife could collect two post-click records for that ad, but in such cases, we retain only the final landing page as the post-click record.

Second, the pre-click data we collected may sometimes come from different sources. Overlapped ads, a kind of ad fraud behavior in [32], may be the cause. For example, as shown in Figure 3(b), a sample app may satisfy all the criteria we define in Section 3.2. It contains a full-screen ad WebView (red) and another WebView (blue) which is covered by the top ad WebView. The pre-click data we collected from this app could contain ad traffic from the top ad WebView as well as background traffic from the blue WebView. We are not able to distinguish the exact WebView URLs from all the labeled network traffic in the pre-click data. It is a limitation of our framework. Therefore, we only used pre-click URLs for identifying the communications with one of the ad domains.

4 EVALUATION

In this section, we talk about how to preprocess our collected initial data, then analyze the security issues in mobile advertising, and finally study new threats related to advertisers.

4.1 Ad Dataset

Our pre-click data matched 133 rules of 97 ad networks in the list of ad-related domains. For example, Leadbolt uses four different domains (i.e., *leadbolt.net*, *leadboltads.net*, *leadboltmobile.net*, and *leadboltapps.net*). The matched ads in our pre-click data are in 103 distinct sizes. Meanwhile, we observed Google's ad domains (i.e., *admob.com*, and *doubleclick.net*) in over 90% of all ads. As for our post-click data, we got 58,876 unique redirect chains, which totaled 203,783 unique URLs. Excluding URLs from well-known domains (e.g., *doubleclick.net*, *google.com*, *facebook.com*, *amazon.com*, and *yahoo.com*), we still have 56,914 URLs left for our malvertising analysis (Section 4.2.2). The landing pages collected in Depot belong to nearly 3.4K distinct advertisers.

In order to conduct further analysis, we matched both pre- and post-click data. In [15], the authors used a fixed time window to group all data generated by each Android app. However, we could not apply a fixed time window, since the time of each ad collection varied. Therefore, we devised a heuristic strategy to match pre-click ads and their post-click landing pages. For the pre-click data of a particular ad, its post-click data should be generated within 45 seconds in our data collection process. Therefore, we gradually increased the window size from 32 to 45 seconds until a match is found. Finally, we got over 83K matched ads after linking the pre- and post-click data. In particular, 25,764 ads landed at Play Store, and 57,880 ads ended up with Depot.

4.2 Click Fraud & Malvertising

Here we revisited the two prominent ad threats. After revealing their own trends, we further explore their interconnection.

4.2.1 *Click Fraud.*

Identifying click fraud is more or less related to detecting fraudulent app developers and their apps. Ad networks provide their own WebView-based AdView. In our settings, if an app automatically clicks an ad right after the app starts, the foreground running view may have already been switched to Depot before MAdLife calls AndroidViewClient's `dump()` to log the pre-click data. MAdLife would take a screenshot of the landing page as usual. However, MAdLife may also accidentally record the landing page in both the pre- and post-click data of an ad. Therefore, we took both situations into account: 1) post-click data from the unmatched collection, where we found 356 apps from 575 cases; and 2) pre-click data from the matched dataset, whose screenshots have the same resolution size as Depot, and where we got 38 apps from 132 cases. Both situations totaled 372 unique apps. We run our test with these samples at a slower speed in order to properly trigger click fraud. Still, 81 apps fell into one of the above situations. We could run such tests recursively, until their click-fraud behaviors were confirmed. But after our manual inspection we found 37 click-fraud apps. These click-fraud apps experienced more than 660K total downloads, and two of them obtained over 100K downloads each.

According to Table 2, we saw that click fraud is still not a well-solved problem. Surprisingly, one of the seven fraudulent app developers owns 30 out of the 37 apps. After checking with VirusTotal, we obtained three benign apps within them. We further looked into their network traffic by running our framework again. Our test showed that all samples have only three origins, and two of the three sources are fed by the apps' local files, including those 30 apps.

4.2.2 *Malvertising.*

We used VirusTotal to analyze the 56,914 URLs to determine if the associated ads are malicious or not. Specifically, we got 1,127 positive URL cases, in which 248 were flagged by at least three scanners. We call these cases *malicious* URLs. These 248 URLs belonged to 65 unique domains, and also resulted in 199 redirect

Table 2: MAdFraud [15] vs. MAdLife (click fraud)

	MAdFraud	MAdLife
# Samples	130K from third-party markets + 35K malware	143K from Play Store
# Positive Cases	13 from third-party markets + 3 from Play Store + 6 malware	37 from Play Store

Table 3: Rastogi et al. [39] vs. MAdLife (malvertising)

	[39] (US & CN)	MAdLife (US & CA)
# Samples	600K from 5 markets	5.7K from Play Store
# Collected Links	1M (US)+415K (CN)	203.7K (US & CA)
# Malicious URLs	948 (US)+1,475 (CN)	248 (US & CA)
# Unique Domains	64 (US)+139 (CN)	65 (US & CA)

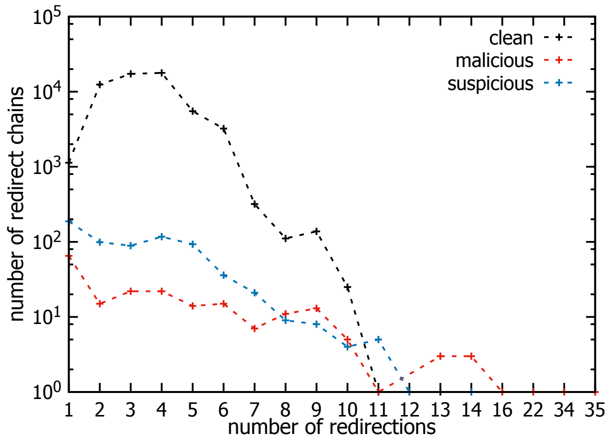


Figure 4: Statistics about different types of redirect chains

chains. The rest of those 1,127 URLs are flagged by one or two scanners. We call them *suspicious* URLs. They came from 147 unique domains, and resulted in the other 669 redirect chains[11]. Therefore, 1.49% of the 57,880 ads were from malicious or suspicious redirect chains. In Table 3, comparing with the findings from [39] in 2016, we found relatively more malvertising domains. However, we analyzed less apps in our experiment and got less links crawled from mobile ads. Therefore, we believe that more attackers are getting into this battlefield, and the situation related to malvertising is becoming worse.

Afterwards, we looked into the results from four aspects. First, we made an observation similar to the ones in [31, 39] that, longer redirect chains are more likely to be malicious. It is even more evident with our dataset. For the three kinds of redirect chains, we plot their relations between number and length in Figure 4. Accordingly, while only two cases with all benign URLs have more than 10 hops, the length of a malicious chain can even reach 35 redirections.

Second, only 32.16% of the malicious, and 64.87% of the suspicious redirect chains had non-blank landing pages. Moreover, two of the locations where we collected ads used the Palo Alto Firewall[12], which blocked 26 ad clicks. 14 of these 26 problematic cases are malicious, and 11 have no redirections. During our ad collection process, no drive-by-download case was encountered.

[11] Among these 879 links flagged by less than three scanners, we excluded those URLs appeared in the previously mentioned 199 redirect chains.

Table 4: Malvertising traffic from the click-fraud apps

	Malicious	Suspicious
# Post-click	199	669
# Pre-click	187	580
# Unmatched Post-click	12	89
# Matched Pre-click	11	22
Percentage	11.56%	16.59%
Overall Percentage	15.44%	

Third, we discovered 92 from the 199 malicious, and 166 advertisers from the 669 suspicious redirect chains, respectively. There were 236 unique advertisers in total.

Last, malicious and suspicious redirect chains came from 134 and 263 apps, respectively. Also, 74 of the malicious , and 133 of the suspicious apps were detected by VirusTotal. We found that, 168 unique apps getting at least one positive from VirusTotal contributed to malvertising.

4.2.3 *Correlation.*

We found statistical correlations between click fraud and malvertising. According to Table 4, after giving equal opportunities to run target apps, we found that 134 malvertising cases (15.44%) came from the click-fraud apps, including 11.56% of the malicious and 16.59% of the suspicious redirect chains, respectively. Furthermore, as over 730 ads are generated by the 37 click-fraud apps in the dataset, these apps have a 18.36% chance to initiate malvertising. Also, since the other 734 malvertising cases came from the rest 57,150 ads, the non click-fraud apps merely had a 1.28% chance to initiate malvertising. As a result, a user of a click-fraud app, is 14.34x more likely to be shown a malicious ad compared to a user of a legitimate app.

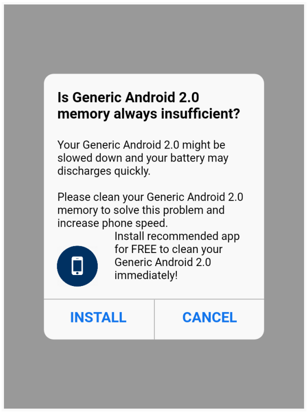
Why are the two ad-related security issues more likely correlated? A mobile user needs to interact with an HTML ad through the app's UI. It enables attackers to generate fraudulent ad clicks automatically, which is difficult to detect for the restricted JavaScript running in the app's WebView. Such attackers may usually select ad networks with poor SDK implementations and vulnerable scrutiny processes for malvertising. Therefore, ad networks should take on more responsibility to mitigate such risks, as they are responsible for strengthening their SDKs and scrutinizing malvertising instances.

4.2.4 *Scam Cases.*

Although scam cases have been well studied [39], we found new and interesting observations by looking into the HTML files of our scam cases. Figure 5(a) illustrates the first case, an ad for a fake anti-virus app. No matter whether users click "Install" or "Cancel", the same malicious link will be triggered.It reveals that, ad viewers are more likely to enter a malicious website with such a trick.

Figure 5(b) shows an unexpected prize and lottery scam, which is alleged to offer a free phone. In our dataset, we found two ads

[12] http://www.paloalto-firewalls.com

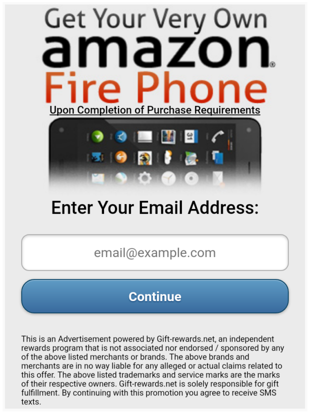

Is Generic Android 2.0 memory always insufficient?

Your Generic Android 2.0 might be slowed down and your battery may discharges quickly.

Please clean your Generic Android 2.0 memory to solve this problem and increase phone speed.

Install recommended app for FREE to clean your Generic Android 2.0 immediately!

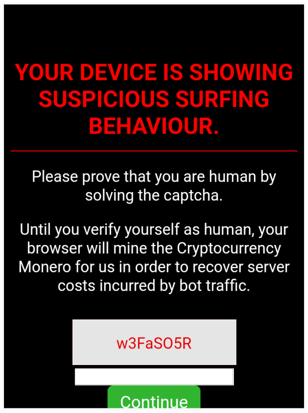
(a) Ad with deceptive download


Get Your Very Own
amazon
Fire Phone
Upon Completion of Purchase Requirements
Enter Your Email Address:
email@example.com
Continue

This is an Advertisement powered by Gift-rewards.net, an independent rewards program that is not associated nor endorsed / sponsored by any of the above listed merchants or brands. The above brands and merchants are in no way liable for any alleged or actual claims related to this offer. The above listed trademarks and service marks are the marks of their respective owners. Gift-rewards.net is solely responsible for gift fulfillment. By continuing with this promotion you agree to receive SMS texts.

(b) Ad with fraudulent reward

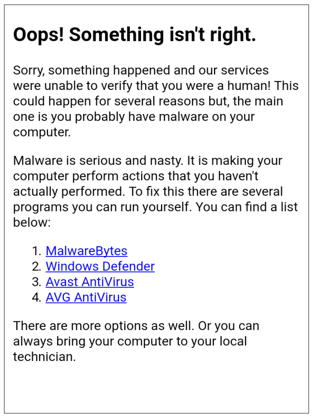
Figure 5: Scam examples


YOUR DEVICE IS SHOWING SUSPICIOUS SURFING BEHAVIOUR.

Please prove that you are human by solving the captcha.

Until you verify yourself as human, your browser will mine the Cryptocurrency Monero for us in order to recover server costs incurred by bot traffic.

(a) Drive-by cryptomining


Oops! Something isn't right.

Sorry, something happened and our services were unable to verify that you were a human! This could happen for several reasons but, the main one is you probably have malware on your computer.

Malware is serious and nasty. It is making your computer perform actions that you haven't actually performed. To fix this there are several programs you can run yourself. You can find a list below:

1. MalwareBytes
2. Windows Defender
3. Avast AntiVirus
4. AVG AntiVirus

There are more options as well. Or you can always bring your computer to your local technician.

(b) Deceptive cryptojacking

Figure 6: Cryptojacking examples

with the same image. However, these screenshots were loaded from two different malicious domains (i.e., *primerewardz.com* and *premiumrewardsusa.com*), with no redirections. Users should refrain from surrendering their private information, as some scams ask for emails to get a fraudulent reward.

```
<script>
  ...
  var miner = new CoinHive.User("...", "tt", {throttle:0});
  miner.start(CoinHive.FORCE_EXCLUSIVE_TAB)
</script>
```

Listing 1: Drive-by cryptomining script found in Figure 6a

```
<script type="text/javascript" defer="" async="" src="https://load
      .jsecoin.com/load/.../0/0/"></script>
<iframe src="https://claimers.io/a-mining?address=..." style="
      visibility: hidden;"></iframe>
```

Listing 2: Two cryptomining scripts found in Figure 6b

4.2.5 *Cryptojacking Cases.*

Here we show two cryptojacking instances, in which attackers secretly mine cryptocurrency with victims' computing power. While we discovered over 25 obvious drive-by cryptomining cases, as seen in Figure 6(a), we also found 7 subtle examples like that in Figure 6(b).

The former case, with up to 9 redirections, was not from a single domain, but 13 different ad domains[13]. However, all these redirections finished at one of the two websites (i.e., *rcyclmnr.com*, and *rcyclmnrepv.com*). Code 1 shows that the COINHIVE[14] script was used in 6(a). We submitted the embedded JavaScript file[15] to VirusTotal, where 33 out of 58 scanners detected this script. We later found out that our discovery was confirmed by both Symantec [30] and Malwarebytes [40].

The latter case includes at most 2 redirections, originates from the same domain (i.e., *ezanga.com*), and ends up with *droppedclick.com*. A closer inspection revealed two interesting tactics used by these developers: 1) the website provided benign links for users to click; and 2) the website used three different cryptomining scripts. According to the HTML files, the other two cases were undergoing landing page redirects; therefore, we had no further information for those webpages. One example was with *coinhive*, two instances were with *jsecoin.com*, and another two cases were with *claimers.io*. See Code 2. To the best of our knowledge, we were the first to find these samples, while no scanner of VirusTotal flagged the *claimers.io* domain as a potential risk yet.

Furthermore, we looked into the issues from two aspects. First, we examined the 49 apps where these cryptojacking sample were found. Surprisingly, 21 of the sample apps were benign after scanning with VirusTotal. Thus, the malvertising cases found in these apps were solely due to untrusted ad networks.

Second, fraudulent app developers contributed to the issues as well. Specifically, the aforementioned developer, who owned 30 click-fraud apps, has raised our attention again due to the 6 apps among our revisited samples. Our first impression of the developer's apps is about click fraud: after launching any of these apps, users would sometimes be automatically redirected to a landing page showing arbitrary content (e.g., automotive sales, lottery rewards, and pornographic chat rooms). We found that a local file, named EXIT.HTML, was called in each of the 6 apps. Although the file is totally clean under VirusTotal, it calls the `doStartAppClick()` method at every app start, to automatically trigger a call to visit an ad URL[16]. Ironically, this URL also seemed benign under VirusTotal, but it could serve as an entry point for redirecting users to different sites, either dirty or clean. We loaded the URL in a web browser, and were redirected to a couple of automatic malware downloading pages[17]. At the time of this writing, these apps were still listed on

[13]Domains of the first link in a redirect chain include: *leadbolt.net*, *tc-clicks.com*, *apperol.com*, *shootmedia-hk.com*, *despiteracy.com*, *leadzuaf.com*, *spxtraff.com*, *smartoffer.site*, *bestperforming.site*, *mobcampaign.site*, *tracknet.site*, *topcampaign.site*, and *ads.gold*

[14]https://coinhive.com

[15]Its VirusTotal SHA-256 is 5d514880ad502302dd4bf0ef8da5d38356385d1c43689f6739f67 71ed7a4ef73

[16]The URL, *http://pub.reacheffect.com/ra/878/1042/p/m/%7Bcampaign_id%7 D/CA*, was taken down in April, 2018

[17]Their VirusTotal SHA-256's are 0c6e40eb1c3b00de1c72f22dec5cffddc3df66672360f79d5 4d9922c018f4aa6 and 1654cf25365332198c84f7e1e17b237abf0447a97e18800cc349e91cc2eb9a71

Play Store, although on January 15th, 2018 the developer switched to other ad networks, and then finally disabled the auto click feature.

To sum up, the tricks of fraudulent app developers and of untrusted ad networks were continuously evolving. It is not limited to only deceiving users but also developing new techniques to hijack their computing power.

4.3 Other Threats

Besides the previous two well-known threats, we have also studied two new types of ad threats on a large scale: 1) malicious external JavaScript on landing pages; and 2) inappropriate ads.

4.3.1 Data Preparation.

In order to facilitate our analysis, we removed duplicates from both landing pages and screenshots. First, as the HTML files of the same website we crawled might contain different metadata, we were not able to use the traditional cryptographic hashing algorithm to classify websites. Therefore, we used the CETD algorithm [45] to extract text content and anchored links from webpages. We finally retained 12,036 distinct documents.

Second, the same website with a countdown timer may be phototaken differently in two ad instances. Therefore, in order to retain useful content, we filtered out pictures with pure white or black background. The file size of such images is usually less than 15KB in size. Afterwards, we consecutively employed three image fingerprinting methods[18] (i.e., wavelet hashing, perception hashing, and difference hashing), which generate similar output hashes between two cognate input files. The process produced 6,337 unique images.

4.3.2 Malicious External JavaScript Code.

Advertisers may embed JavaScript code from their own host or a third-party host in their landing pages. Some external scripts could be malicious. However, VirusTotal scanners would report the results for the only submitted URL, not including its embedded external scripts. Especially, it is very hard for VirusTotal scanners to detect a known malicious JavaScript snippet after code obfuscation. As a result, this approach would miss malicious content presented to users on a landing page. Therefore, we further leveraged VirusTotal to scan all external scripts of each landing page.

After extracting 146K unique external script URLs and distilling 115K URLs starting with HTTP(S) from almost 58K HTML files, we found only 243 positively flagged URLs (0.21%) with VirusTotal. Surprisingly, these scripts were used in more than 5,880 landing pages (i.e., 10.16% of the 57,880 HTML file). Among these 5,880 landing pages, we discovered 53 malvertising cases and 5,827 benign landing pages. Lastly, after running the CETD algorithm, we got 1,384 unique documents (i.e., 11.5% of the 12,036 distinct documents).

External JavaScript code can be loaded either from the same domain or from third-party domains. For example, we witnessed that a "benign" landing page consisted of more than 41 positively flagged URLs for the advertiser's own JavaScript code. That is to say, illicit advertisers may use such tricks to avoid traditional scrutiny techniques. Third-party JavaScript code is usually dynamically loaded; therefore, it may be subject to change on the third-party side. In our dataset, we discovered that a few snippets from Google-registered domains (e.g., gstatic.com, and ytimg.com) were also positively flagged. Table 5 shows the statistics of positively flagged URLs.

Table 5: Top 10 domains with positively flagged URLs

# Occurrences	# Unique URLs	Domain
409	6	gstatic.com
287	4	ytimg.com
84	1	adroll.com
64	1	engagio.com
50	1	parastorage.com
44	1	bootstrapcdn.com
18	1	bkrtx.com
14	1	googleapis.com
13	1	roberthalfgcs.com
10	1	ucarecdn.com

4.3.3 Ad Inappropriateness.

The issue of inappropriate ads on the Internet has existed for a long time but has not yet been fully explored. Legislatures in the United States have created various acts (e.g., COPPA [12], CIPA [11]) to safeguard children online, including from ad views/clicks. In addition, well-known ad networks [20, 23–25, 46] also establish content policies for advertisers. But, self regulation is not enough. Here we took the first step to analyze and classify inappropriate ad content on a large scale.

We used the Google Cloud Natural Language API to get known entities and content categories. After identifying documents with the same known entities, we reduced duplicate files with a list of the same entities from 12,036 to 7,067. These documents contain all 27 level-1 categories[19]. We selected a few sensitive categories (i.e., Adult, and Sensitive Subjects) for our analysis. Afterwards, we used known entities to recursively select other sensitive categories.

Determining the inappropriateness of an ad could be very subjective. Therefore, we adhered to the following guidelines based on ad network policies and public opinion to identify inappropriate ads. First, AdWords does not allow "collecting personal information from children under 13 or targeting interest content to children under the age of 13" [25]. Thus, we listed all ads related to age-restricted content. Second, well-known ad networks [17, 18, 22] announced their plan of blocking cryptocurrency-related ads, one after another, as the content is often associated with deception and fraud [38]. Thereafter, we labeled all related ads. Third, due to Russian agents-used political ads and the Cambridge Analytica scandal, Facebook received criticism and modified its policy for political ads [5]. Therefore, we put all political ads into the "public concerns" category. Accordingly, we classified inappropriate ads into two severity levels:

Policy Compliance: adult (e.g., pornography), criminal record, cryptocurrency, drug (e.g., marijuana), prize (e.g., fake awards), security (e.g., fake antivirus), store (e.g., age-restricted product)

Public Concerns: age (e.g., tattoo), extramarital affair (e.g., Ashley Madison), gambling (e.g., casino), health (e.g., plastic surgery), political campaign (e.g., voting), privacy (e.g., phone number)

The result is depicted in Table 6. In summary, 345 out of 7,067 unique ad landing pages are in the "Policy Compliance" category,

[18]https://github.com/JohannesBuchner/imagehash

[19]https://cloud.google.com/natural-language/docs/categories

Table 6: Inappropriate ads and their categories

		Ad Category	Unique Ads (#)	Full Dataset (%)
Policy Compliance	345 & 7.9%	adult	14	0.33%
		criminal record	7	1.44%
		cryptocurrency	146	63.49%
		drug	3	0.17%
		prize	46	2.43%
		security	6	0.33%
		store	123	31.81%
Public Concerns	24 & 0.266%	age	2	1.30%
		extramarital affair	3	0.26%
		gambling	5	20.78%
		health	3	5.19%
		political campaign	8	33.12%
		privacy	3	31.82%

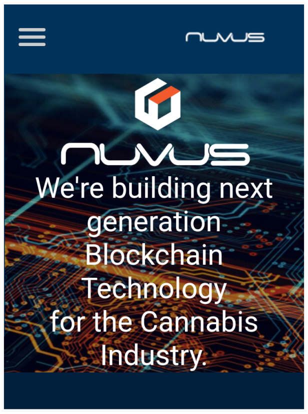
and 24 are in the "Public Concerns" category; 4,577 (7.9%) out of 57,880 ad landing impressions are in the "Policy Compliance" category, and 154 (0.26%) are in the "Public Concerns" category.

We also leveraged the Google Vision API on our screenshots to detect offensive contents and find web entities with similar images. We found the API performed quite well in detecting pornography. The API was also good at finding similar images that contain very few characters. For example, a prize wheel image with barely any characters was flagged as potentially dangerous by the API, because similar images were found at some malicious websites. However, it was difficult to accurately detect any other kinds of offensive ad content using the Vision API in general. Thus, we used only the Natural Language API in our analysis.

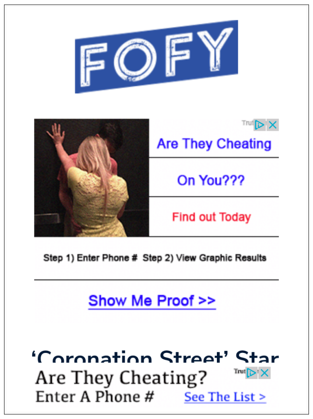
4.3.4 Case Studies.

Unlike click fraud and malvertising which affect all users, inappropriate ads may merely influence a specific subset of population. Here we exemplify two cases to show the aggravating circumstances, as depicted in Figure 7. First, along with their price volatility, cryptocurrencies attract attention from people all over the world. As a result, the entire industry suddenly booms. During our ad collection, we collected mobile ads of 39 different cryptocurrencies and of 27 cryptocurrency-related reports from 15 online media. One of such ads, shown in Figure 7(a), publicized its cryptocurrency for the cannabis industry. As we learned from the Internet, marijuana is not legalized at the federal level in the United States and in Canada [48] at the time of our experiment. Such ads should thus be prohibited.

Second, it is even worse that *nested ads* can bypass all restrictions. Figure 7(b) illustrates the landing page of a blog website, where another ad is nested. Although ad networks can scrutinize matches between ads and their landing pages, their arms may not be able to reach at nested ads. To the best of our knowledge, we did not find any policies to directly regulate such a situation. As a result, inappropriate ad content may be broadcast within landing pages subscribed to well-known ad networks. Moreover, we considered it as an adware, because for more than 30 FOFY ads we collected, only ads were shown within the screenshots but their real content displayed underneath. Accordingly, the situation of inappropriate ads is getting worse.


NUVUS
NUVUS
We're building next generation Blockchain Technology for the Cannabis Industry.

(a) Controversial topics in an ad



(b) Ads within an adware

Figure 7: Examples for ad inappropriateness

4.3.5 Takeaways.

Our studies revealed that embedded external scripts in landing pages should also be examined for two reasons: 1) illicit advertisers may use this approach to escape from ad networks' security lookups; and 2) third-party JavaScript links may change their backend code for their own interests. Therefore, rather than only checking inappropriate ad content, ad networks should also scrutinize the JavaScript files embedded by advertisers.

Last but not the least, after Google blocked cryptocurrency ads in June 2018 [18], we utilized MAdLife again to crawl ads in the next month. We did not find any cryptocurrency ad in this crawl. In reverse, after banning cryptocurrency ads for a few months, Facebook gave Coinbase a privileged position to allow its crypto-related ads again in July 2018 [29]. Ad policies thus may be more inconsistent than what we expected. More importantly, nested ads could be used to bypass new content-based regulations.

5 DISCUSSION

Ethics. Our ad collection experiment may inflate advertisers' budgets, like other studies [7, 10, 31, 36, 39, 50]. However, we tried to reduce the impact on the real ad ecosystem in our research. On one hand, MAdLife clicked less than 15 ads within each app and visited the landing page of each advertiser for less than 25 times on average. Therefore, each individual app developer's income and each individual advertiser's cost would not be significantly affected. On the other hand, clicking the ads is necessary for studying these ad threats. Although our research might have inflated the budgets of some advertisers, our findings can benefit the whole ad ecosystem in the long run from a broader perspective. We believe that a better ad ecosystem will be more cost-efficient for advertisers. Moreover, we designed our experiment in a way that MAdLife's artificial behaviors can be easily detected by ad networks (e.g., 10 successive clicks within one app and from the same IP addresses). A responsible ad network shall be able to detect them as invalid clicks, and thus invalidate any charges incurred to advertisers.

Limitations. First, we run the experiment on API 25 with Genymotion 2.11.0. Nowadays, Android 8.0 (API level 26) has enabled safe browsing on WebView. If ad networks adapt the feature, the malvertising issues could be somewhat mitigated. Second, our pre-click data may contain URLs other than those we expect, since we cannot detect WebViews underneath a full-screen WebView and separate each WebView's logcat traffic. Third, to the best of our knowledge, the Google Cloud APIs are among the best APIs available on the market. We also have tested APIs provided from two other services, which did not outperform Google's APIs. However, we cannot measure the accuracy of Google's APIs as we are unable to manually build ground truth for thousands of images and webpages. Last, Google asks that, ads displayed in family apps are expected to be consistent with their maturity ratings [26]. In [10], authors talk about this issue. However, whether the parental control settings are on or off, children are able to access all installed apps on a device. Therefore, our studies only focus on inappropriate ads in general instead of with age-restricted apps.

6 RELATED WORK

Ad Collection. Due to the fact that ad content is dynamically generated, various methodologies are used to crawl ads. Collecting online ads is relatively easy when not in mobile settings. In online advertising, researchers can either run scripts with their add-ons [7, 31] or build a Selenium-based crawler [50], with Alexa's top-ranked website lists. Also, the researchers can either intercept HTTP traffic [31, 50] or only harvest ad-related visual elements [7]. Nevertheless, ads' life spans can be easily deduced without triggering ad clicks. Whereas, in mobile advertising, researchers are still able to analyze HTTP traffic [15, 36, 39], but it's more difficult to track ads' lifespans. For example, since no user interaction (except for click fraud) is involved in [15], their studies stop at ad loading/rendering. Besides, both [36] and [39] consider every touch event as a starting point.

Click Fraud. Ad fraud can be conducted with [2, 32] or without [15, 35] human intervention. Likewise, researchers reveal that such attacks in web browsers [2, 35] are technically more advanced than with handheld devices [15, 32]. While [35] depicts two clickbot families, [2] traces from an IP back to the conspiracy with DNS hijacking. In mobile advertising, the term can be subdivided into display fraud [32] and click fraud [15]. Other than these, researchers in [51] detected duplicate clicks in pay-per-click streams with two bloom filter related algorithms.

Malvertising. Malvertising lurks at the pre-click phase, but is exposed at the post-click phase. Researchers, who collect ads in online advertising, focus on either ad networks [50] or redirect chains [31]. Likewise, such research works in mobile advertising result in the exploration of either ad networks [27, 52] or redirect chains [39]. Because of ad syndication, we believe that the use of redirect chains would be more beneficial. With the help of studies like [33], authorities will be able to take down malvertising networks without a case-by-case investigation.

Dangerous JavaScript Detection. The studies of detecting dangerous JavaScript code are numerous. JSAND [14] conducts classification based on static and dynamic features, and instruments JavaScript runtime environments to detect the execution of malicious scripts. Prophiler [8] statically analyze features of the HTML page, of the embedded JavaScript code, and of the associated URL, and examines malicious content on massive webpages with obfuscated JavaScript. Similarly, ZOZZLE [16] leverages features associated with JavaScript context to detect unobfuscated exploits. As for code obfuscation, Sharif et al. [42] revealed that conditional code obfuscation could be used to hinder malware analysis.

Content Analysis. Nowadays, contextual advertising is used to target users. [49] describes a feature-based keyword extraction system for online contextual advertising. In order to analyze web content, [34] uses the Alchemy API to label all crawled webpages, and categorizes them into different topic categories. Likewise, we use Google Cloud's Natural Language API to analyze our dataset, besides using the CETD algorithm [45] to only encompass web content and anchored links. Similar to finding resolved ad URLs, content scraping in the mobile settings is not so easy as that in a browser. Therefore, both [32] and [37] capture third-party apps' page contents on Windows-based mobile platforms. While the former extracts page information with the UI extraction channel, the latter instruments app binaries to insert custom logging code. Unlike our work that evaluates ad inappropriateness in general, [10] studies whether ads are consistent with host apps' maturity ratings on a small scale that is below 4K ads. Also, their "topic classification" part is not well elaborated; therefore, we are thus unable to make any technical comparison.

7 CONCLUSION

While implemented on top of existing web technologies, mobile in-app ads face similar threats in different forms. In this paper, we implemented MAdLife– a data collection framework on Android to record all necessary data generated within each ad's entire lifespan. By using MAdLife, we explored the abusive practices, including click fraud, malvertising, and other threats on landing pages, with 84K ads. We discovered 37 click-fraud apps and over 860 malvertising redirect chains. More importantly, we revealed that a user of a click-fraud app is 14.34x more likely to be shown with a malicious ad comparing with a user of a legitimate app. Further, we revealed that 250 widely used JavaScript snippets are actually harmful to users. These snippets were used on over 10% of our collected landing pages. Lastly, we identified that 8% ads are inappropriate, either not complying with ad policies or showing controversial content. We suggest that ad networks should, and only they can, take more effective measures to protect mobile users and themselves from the above ad threats.

REFERENCES

[1] Alphabet. 2018. Google Second Quarter 2018 Results. https://abc.xyz/investor/pdf/2018Q2_alphabet_earnings_release.pdf. [Online; accessed 17-February-2019].

[2] Sumayah A Alrwais, Alexandre Gerber, Christopher W Dunn, Oliver Spatscheck, Minaxi Gupta, and Eric Osterweil. 2012. Dissecting ghost clicks: Ad fraud via misdirected human clicks. In *Proceedings of the Annual Computer Security Applications Conference (ACSAC).*

[3] Appbrain. 2018. Android app download statistics on google play. https://www.appbrain.com/stats/android-app-downloads. [Online; accessed 17-February-2019].

[4] Appbrain. 2018. Most popular google play categories. https://www.appbrain.com/stats/android-market-app-categories. [Online; accessed 17-February-2019].

[5] Daniel Arkin. 2018. Facebook announces major changes to political ad policies. https://www.nbcnews.com/tech/social-media/facebook-announces-majorchanges-political-ad-policies-n863416. [Online; accessed 17-February-2019].

[6] Michael Backes, Sven Bugiel, Philipp von Styp-Rekowsky, and Marvin Wißfeld. 2017. Seamless in-app ad blocking on stock android. In *2017 IEEE Security and Privacy Workshops (SPW)*. IEEE, 163–168.

[7] Paul Barford, Igor Canadi, Darja Krushevskaja, Qiang Ma, and S Muthukrishnan. 2011. Adscape: Harvesting and analyzing online display ads. In *Proceedings of the 21st International World Wide Web Conference (WWW)*. Seoul, Korea.

[8] Davide Canali, Marco Cova, Giovanni Vigna, and Christopher Kruegel. 2011. Prophiler: a fast filter for the large-scale detection of malicious web pages. In *Proceedings of the 20th International World Wide Web Conference (WWW)*. Hyderabad, India.

[9] Gong Chen, Jacob H Cox, A Selcuk Uluagac, and John A Copeland. 2016. In-depth survey of digital advertising technologies. *IEEE Communications Surveys & Tutorials* 18, 3 (2016), 2124–2148.

[10] Ying Chen, Sencun Zhu, Heng Xu, and Yilu Zhou. 2013. Children's exposure to mobile in-app advertising: An analysis of content appropriateness. In *2013 International Conference on Social Computing (SocialCom)*. IEEE, 196–203.

[11] Federal Communications Commission. 2018. Children's internet protection act. https://www.fcc.gov/consumers/guides/childrens-internet-protection-act. [Online; accessed 17-February-2019].

[12] Federal Trade Commission. 2018. Children's Online Privacy Protection Rule. https://www.ftc.gov/enforcement/rules/rulemaking-regulatory-reform-proceedings/childrens-online-privacy-protection-rule. [Online; accessed 17-February-2019].

[13] Josh Constine and Taylor Hatmaker. 2018. Facebook admits Cambridge Analytica hijacked data on up to 87M users. https://techcrunch.com/2018/04/04/cambridge-analytica-87-million. [Online; accessed 17-February-2019].

[14] Marco Cova, Christopher Kruegel, and Giovanni Vigna. 2010. Detection and analysis of drive-by-download attacks and malicious JavaScript code. In *Proceedings of the 19th International World Wide Web Conference (WWW)*. Raleigh, NC.

[15] Jonathan Crussell, Ryan Stevens, and Hao Chen. 2014. Madfraud: Investigating ad fraud in android applications. In *Proceedings of the 12th annual international conference on Mobile systems, applications, and services (MOBISYS)*. ACM, 123–134.

[16] Charlie Curtsinger, Benjamin Livshits, Benjamin G Zorn, and Christian Seifert. 2011. ZOZZLE: Fast and Precise In-Browser JavaScript Malware Detection.. In *Proceedings of the 20th USENIX Security Symposium (Security)*. San Francisco, CA.

[17] Megan Rose Dickey. 2018. Facebook is banning cryptocurrency and ico ads. https://techcrunch.com/2018/01/30/facebook-is-banning-cryptocurrency-and-ico-ads/. [Online; accessed 17-February-2019].

[18] Jillian D'Onfro. 2018. Google' will ban all cryptocurrency-related advertising. https://www.cnbc.com/2018/03/13/google-bans-crypto-ads.html. [Online; accessed 17-February-2019].

[19] Feng Dong, Haoyu Wang, Yuanchun Li, Yao Guo, Li Li, Shaodong Zhang, and Guoai Xu. 2017. Fraudroid: An accurate and scalable approach to automated mobile ad fraud detection. *arXiv preprint arXiv:1709.01213* (2017).

[20] Facebook. 2018. Advertising Policies. https://www.facebook.com/policies/ads/. [Online; accessed 17-February-2019].

[21] Facebook. 2018. Facebook Second Quarter 2018 Results. https://investor.fb.com/investor-news/press-release-details/2018/Facebook-Reports-Second-Quarter-2018-Results/default.aspx. [Online; accessed 17-February-2019].

[22] Jon Fingas. 2018. Twitter will ban most cryptocurrency ads. https://www.engadget.com/2018/03/26/twitter-bans-most-cryptocurrency-ads. [Online; accessed 17-February-2019].

[23] Google. 2018. Ad policies. https://support.google.com/youtube/topic/30084?hl=en&ref_topic=2972865. [Online; accessed 17-February-2019].

[24] Google. 2018. AdMob & AdSense Policies. https://support.google.com/admob/answer/6128543?hl=en. [Online; accessed 17-February-2019].

[25] Google. 2018. AdWords policies. https://support.google.com/adwordspolicy. [Online; accessed 17-February-2019].

[26] Google. 2018. Parent guide to google play. https://support.google.com/googleplay/answer/6209547?hl=en. [Online; accessed 17-February-2019].

[27] Michael C Grace, Wu Zhou, Xuxian Jiang, and Ahmad-Reza Sadeghi. 2012. Unsafe exposure analysis of mobile in-app advertisements. In *Proceedings of the fifth ACM conference on Security and Privacy in Wireless and Mobile Networks (WiSec)*. ACM, 101–112.

[28] IAB. 2019. IAB Internet Advertising Revenue Report. https://www.iab.com/insights/iab-internet-advertising-revenue-report-conducted-by-pricewaterhousecoopers-pwc-2. [Online; accessed 17-February-2019].

[29] Jose Antonio Lanz. 2018. Facebook restores coinbased cryptocurrency ads. https://ethereumworldnews.com/facebook-restores-coinbase-cryptocurrency-ads/. [Online; accessed 17-February-2019].

[30] Hon Lau. 2017. Browser-based cryptocurrency mining makes unexpected return from the dead. https://www.symantec.com/blogs/threat-intelligence/browser-mining-cryptocurrency. [Online; accessed 17-February-2019].

[31] Zhou Li, Kehuan Zhang, Yinglian Xie, Fang Yu, and XiaoFeng Wang. 2012. Knowing your enemy: understanding and detecting malicious web advertising. In *Proceedings of the 19th ACM Conference on Computer and Communications Security (CCS)*. Raleigh, NC.

[32] Bin Liu, Suman Nath, Ramesh Govindan, and Jie Liu. 2014. {DECAF}: Detecting and Characterizing Ad Fraud in Mobile Apps. In *Proceedings of the 11th USENIX Symposium on Networked Systems Design and Implementation (NSDI)*. Seattle, WA.

[33] Hesham Mekky, Ruben Torres, Zhi-Li Zhang, Sabyasachi Saha, and Antonio Nucci. 2014. Detecting malicious http redirections using trees of user browsing activity. In *IEEE INFOCOM 2014-IEEE Conference on Computer Communications*. IEEE, 1159–1167.

[34] Wei Meng, Byoungyoung Lee, Xinyu Xing, and Wenke Lee. 2016. Trackmeornot: Enabling flexible control on web tracking. In *Proceedings of the 25th International World Wide Web Conference (WWW)*. Montreal, Canada.

[35] Brad Miller, Paul Pearce, Chris Grier, Christian Kreibich, and Vern Paxson. 2011. What's clicking what? techniques and innovations of today's clickbots. In *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment (DIMVA)*. Springer, 164–183.

[36] Suman Nath. 2015. Madscope: Characterizing mobile in-app targeted ads. In *Proceedings of the 13th Annual International Conference on Mobile Systems, Applications, and Services (MOBISYS)*. ACM, 59–73.

[37] Suman Nath, Felix Xiaozhu Lin, Lenin Ravindranath, and Jitendra Padhye. 2013. SmartAds: bringing contextual ads to mobile apps. In *Proceeding of the 11th annual international conference on Mobile systems, applications, and services (MOBISYS)*. ACM, 111–124.

[38] Stephen O'Neal. 2018. Big tech are banning crypto and ico ads - is there a reason to panic? https://cointelegraph.com/news/big-tech-are-banning-crypto-and-ico-ads-is-there-a-reason-to-panic. [Online; accessed 17-February-2019].

[39] Vaibhav Rastogi, Rui Shao, Yan Chen, Xiang Pan, Shihong Zou, and Ryan Riley. 2016. Are these Ads Safe: Detecting Hidden Attacks through the Mobile App-Web Interfaces.. In *Proceedings of the 2016 Annual Network and Distributed System Security Symposium (NDSS)*. San Diego, CA.

[40] Jérôme Segura. 2018. Drive-by cryptomining campaign targets millions of android users. https://blog.malwarebytes.com/threat-analysis/2018/02/drive-by-cryptomining-campaign-attracts-millions-of-android-users/. [Online; accessed 17-February-2019].

[41] Scott Shane. 2017. Ads Bought by Russia on Facebook. https://www.nytimes.com/2017/11/01/us/politics/russia-2016-election-facebook.html. [Online; accessed 17-February-2019].

[42] Monirul I Sharif, Andrea Lanzi, Jonathon T Giffin, and Wenke Lee. 2008. Impeding Malware Analysis Using Conditional Code Obfuscation.. In *Proceedings of the 15th Annual Network and Distributed System Security Symposium (NDSS)*. San Diego, CA.

[43] Anastasia Shuba, Athina Markopoulou, and Zubair Shafiq. 2018. NoMoAds: Effective and Efficient Cross-App Mobile Ad-Blocking. *Proceedings on Privacy Enhancing Technologies* 2018, 4 (2018), 125–140.

[44] Statista. 2018. Most popular google play app categories as of 1st quarter 2018, by share of available apps. https://www.statista.com/statistics/279286/google-play-android-app-categories/. [Online; accessed 17-February-2019].

[45] Fei Sun, Dandan Song, and Lejian Liao. 2011. Dom based content extraction via text density. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*. ACM, 245–254.

[46] Twitter. 2018. Prohibited Content Policies. https://business.twitter.com/en/help/ads-policies/prohibited-content-policies.html. [Online; accessed 17-February-2019].

[47] Unity. 2018. Unity monetization service. https://unity3d.com/legal/monetization-services-terms-of-service. [Online; accessed 17-February-2019].

[48] Wikipedia. 2018. Legality of cannabis by country. https://en.wikipedia.org/wiki/Legality_of_cannabis_by_country. [Online; accessed 17-February-2019].

[49] Wen-tau Yih, Joshua Goodman, and Vitor R Carvalho. 2006. Finding advertising keywords on web pages. In *Proceedings of the 15th International World Wide Web Conference (WWW)*. Edinburgh, Scotland.

[50] Apostolis Zarras, Alexandros Kapravelos, Gianluca Stringhini, Thorsten Holz, Christopher Kruegel, and Giovanni Vigna. 2014. The dark alleys of madison avenue: Understanding malicious advertisements. In *Proceedings of the ACM Internet Measurement Conference (IMC)*. Vancouver, Canada.

[51] Linfeng Zhang and Yong Guan. 2008. Detecting click fraud in pay-per-click streams of online advertising networks. In *2008 The 28th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 77–84.

[52] Yajin Zhou, Zhi Wang, Wu Zhou, and Xuxian Jiang. 2012. Hey, you, get off of my market: detecting malicious apps in official and alternative android markets.. In *Proceedings of the 19th Annual Network and Distributed System Security Symposium (NDSS)*. San Diego, CA.