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 funding sources for various of content, services, and applications on both web and mobile platforms. Mobile in-app advertising reuses many existing web technologies under the same ad-serving model (i.e., users - publishers - ad networks - advertisers). Nevertheless, mobile in-app advertising is different from the traditional web advertising in many aspects. For example, malicious app developers can generate fraudulent ad clicks in an automated fashion, but malicious web publishers have to launch click fraud with bots. In spite of using the same underlying web infrastructure, advertising threats behave differently on the two platforms.

Existing works have studied separately click fraud and malvertising in the mobile setting. However, it is unknown if there exists a relationship between these two dominant threats. In this paper, we present an ad collection framework – MAdLife– on Android to capture all the in-app ad traffic generated during an ad's entire lifespan. MAdLife allows us to revisit both threats in a fine-grained manner and study the relationship between them. It further enables the exploration of other threats related to ad landing pages.

We analyzed 5.7K Android apps crawled from the Google Play Store, and collected 83K ads and their landing pages using MAdLife. Similar to traditional web ads, 58K ads landed on web pages. We discovered 37 click-fraud apps, and found that 1.49% of the 58K ads were malicious. We also revealed a strong correlation between fraudulent apps and malicious ads. Specifically, 15.44% of malicious ads originated from the fraudulent apps. Conversely, 18.36% of the ads served in the fraudulent apps were malicious, while only 1.28% were malicious in the rest apps. This suggests that users of fraudulent apps are much more (14x) likely to encounter malicious ads. Additionally, we discovered that 243 popular JavaScript snippets embedded by over 10% of the landing pages were malicious. Finally, we conducted 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; Click Fraud; Malvertising; Measurement

*Both authors contributed equally to this work.

WWW '19, May 13–17, 2019, San Francisco, CA, USA

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 one of the primary ways to monetize content, services, and applications (apps) on the Internet. Companies like Google and Facebook generated 85-95% of their revenues in Q2 2018 [1, 21] through running the largest digital advertising platforms. Traditionally, these ad platforms, or ad networks, display advertisements (ads) from advertisers on a publisher's website. This practice, known as web advertising, has generated over $35B annual revenue 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 HTML ads within a mobile app's WebView. Although sharing many underlying technologies, mobile in-app advertising is different from the 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, in-app ad blocking solutions, such as DNS66[1], DISCONNECT[2], NoMoAds [43], and [6], have not been widely used by mobile users. On the contrary, blockers for traditional web ads have become very popular among web users.

As a result, the abusive practices in web advertising, such as *click fraud* and *malvertising*, exhibit different characteristics on mobile platforms. First, web browsers allow ad scripts to detect automatically generated fraudulent ad clicks. Malicious publishers have to leverage bots to automate ad clicks, which can be easily detected by ad networks. On the contrary, mobile users interact with HTML ads through each app's user interface (UI). This enables malicious app developers to automatically generate fraudulent ad clicks, which are difficult to detect by the restricted JavaScript code running in the AdView. Therefore, it is easier to launch click fraud from genuine user devices with the help of UI automation on mobile platforms. Second, by clicking on ads, users may be brought to malicious pages that serve drive-by download malware or scams. In particular, drive-by downloads have been the primary form of malvertising for web advertising [31]. On the contrary, scams are the dominating malvertising form for mobile in-app advertising [39].

The abusive practices in mobile advertising have been explored by prior works. MAdFraud [15] 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. However, it did not reveal how fraudulent apps initiated automatic ad clicks. Another work [39] detected malvertising cases by collecting and scanning redirect chains using VirusTotal[3]. However, it did not look into JavaScript code loaded from external sources, which could be malicious. Overall, the frameworks in these works require a time consuming app collection process. In addition, click fraud and malvertising may happen together. 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 an ad's entire lifespan (i.e., request → load/render → click → redirect → land) on Android. Specifically, we customize the Android WebView to collect pre-click data. After automatically triggering ad clicks with lightweight UI automation, we redirect and log 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, and investigate other relevant threats in mobile in-app advertising.

We reveal that a benign mobile ad landing page may include malicious external JavaScript snippets. Besides, we recognize that the public has raised concerns about the content in digital ads. For example, political ads on Facebook are the topics of two trending news [13, 41]. Thus, we also take a first attempt to analyze inappropriate mobile in-app ads. By employing several content analysis techniques, we classify inappropriate ads into two categories: 1) policy non-compliant [20, 24, 25, 46] ads, which do not comply with the policies of ad networks (e.g., containing age-restricted information); and 2) controversial ads, which are generally believed inappropriate to some population groups (e.g., promoting online gambling).

After selecting 5.7K from 143K apps crawled from the Play Store, we collected 84K ads (58K landed on web pages) using MAdLife from January to February 2018. Accordingly, ad threats were exposed from several aspects. First, we detected 37 click-fraud apps using a heuristic method. Second, 1.49% of these ads were malicious (32 were cryptojacking ads), according to VirusTotal. Third, we discovered a strong statistical relation between click-fraud apps and malicious ads. On one hand, over 15.44% of malicious ads originated from those fraudulent apps. On the other hand, 18.36% of the ads displayed by click-fraud apps were malicious, whereas in non click-fraud apps only 1.25% of ads were malicious. Therefore, users of click-fraud apps are much more (14x) likely to encounter malicious ads than other users. Fourth, we discovered 243 (0.21% of 115K) popular external JavaScript snippets were unsafe. They were included in over 10% of the landing pages in our dataset. Finally, we identified that around 8% of landing pages were inappropriate – 7.9% did not comply with ad policies and 0.266% were 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. Our contributions can be summarized in threefold:

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

1) We design and implement MAdLife– the first framework to monitor an ad's entire lifespan on the mobile platform. MAdLife enables us to build a panoramic view of mobile ads.

2) We explore the abusive practices involved in mobile in-app advertising, and demonstrate a strong statistical relation between click fraud and malvertising. We discover that users of click fraud apps are much more likely to experience malvertising.

3) We are the first to extend the research scope of analyzing landing pages from two aspects. First, we discover that a few unsafe external JavaScript code may occur in a great number of landing pages. Second, we classify inappropriate ads into two categories (i.e., policy noncompliance, and public concerns).

2 BACKGROUND

2.1 Android WebView

To display an in-app ad, Android apps are usually required to include custom ad SDKs to load the ad in an AdView. The underlying implementations of most AdViews are based on Android's own WebView. While being transparent to mobile app developers and users, the nature of the WebView component has evolved a lot with the development of the Android OS. Since Android 4.4, the WebView component had shifted from using the WebKit rendering engine to sharing the same codebase with Chrome for Android. In Android 5.0, it came out as a standalone APK, which can be independently updated through the Play Store. Starting from Android 6.0, only pre-built WebView versions were shared within the Android Open Source Project (AOSP). Afterward in Android 7.0, the Chrome APK was used to provide and render WebView. Recently, the Safe Browsing feature has been added into WebView with Android 8.0.

In the current Chromium source tree, WebView depends on a C++ shared library and a source set. Within the source set, an AwContents object is implemented in C++ and further encapsulated in Java. By adding logcat messages in function calls within AwContents, we can thus use adb to communicate with an Android emulator and collect the custom log in real-time.

However, obtaining the ad navigation URL (the URL to visit when clicking an ad) in Android is not straightforward, unlike web ads loaded in a browser where developers can directly access ad navigation URLs from the Document Object Model (DOM). As a View object that displays web pages, 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 directly access the DOM within a WebView to obtain the ad navigation URL. A workaround is to hook the AwContents class's `onTouchEvent()` handler, that can catch the touch event when a user touches an ad element rendered in an WebView.

2.2 Mobile Advertising

Serving in-app ads is a very common approach to app monetization for mobile *publishers*/*developers*. To earn income from mobile in-app advertising, app developers register their apps with one or multiple *ad networks*. The developers then receive a unique identifier per ad slot from an ad network, and are instructed to include its *ad library*, which then loads ads to the corresponding ad slots in

the app's UI. While developers can choose different types of ads – including video ads and display ads (i.e., small-size banner ads, and full-screen interstitial ads) – to be shown within their apps, *advertisers* on the other end purchase ad impressions and/or clicks from the developers through the ad networks to promote their content. Together with users viewing/clicking ads in mobile apps, all the other players, including publishers/developers, ad networks/platforms, and advertisers, form a well-functioning ecosystem in the mobile settings [9].

However, such an ecosystem may temporarily be interrupted. For example, when a user launches an app, the app requests an ad for this impression from the subscribed ad network, which cannot return an appropriate ad from its inventory for this ad request. As a result, the user may see a void ad. The case that ads are not loaded within the WebView is called *incapable ad loading*. On the other hand, ads may not be properly displayed due to network overload or ad networks' back-end algorithms, which we call *improper ad rendering*. To fulfill an ad request, *ad syndication* is created to resell the ad impression to other partner ad networks when incapable ad loading happens. Specifically, if the subscribed ad network cannot find an ad from its inventory, the ad request is forwarded to syndicated ad networks. Therefore, both when an ad is being loaded and when it is clicked, communications between the app and multiple ad networks within the ad syndication can lead to multiple *redirects*. The final destination of an ad click is the *landing page*, which is usually controlled by an advertiser. As for *app install ads*, their final destination may end up with an app page in the Play Store.

2.3 Ad Threats

We consider the following ad-related threats in this work.

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

Malvertising. Due to untrusted ad networks or unwanted redirect chains, malvertising may lead an user to a page hosting either drive-by downloads, scams or phishing content (e.g., pornography).

Malicious External JavaScript Code. Usually, advertisers use JavaScript snippets to track user behaviors. However, landing pages containing malicious external JavaScript code may expose both security and privacy risks to end users.

Ad Inappropriateness. Ad networks normally regulate advertised content. But untrusted advertisers can still smuggle inappropriate ads, which can be infelicitous especially for children. That is why Unity Ads [47] talks about COPPA [12] in its common guidelines.

2.4 Content Analysis Services

We used the following services in our ad studies.

VirusTotal aggregates over 60 malware scanners, and offers APIs to analyze files/URLs. The rationale behind using a cluster of antivirus scanners is to fight against mistaken detection by individual scanning tools. i.e., *false positives*.

Google Cloud APIs[4] use REST calls to automate different workflows. We used the Natural Language API to understand the structure and meaning of texts in different aspects (e.g., *entity analysis* and *content classification*). 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).

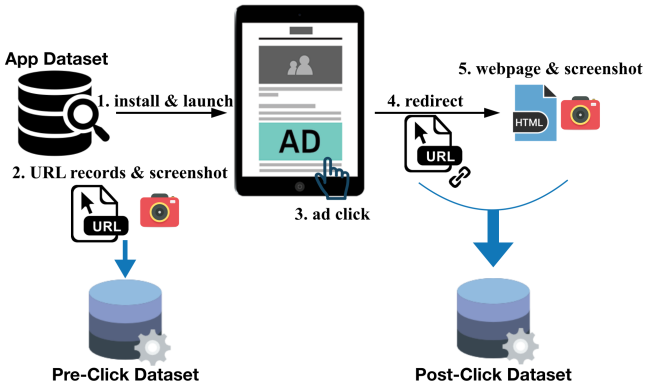


Figure 1: Workflow for data collection

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 our ad 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 an 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 monitors the ad traffic at the same time, and stores the *pre-click* data into a table in our database. Third, MAdLife identifies and clicks the ad. Fourth, the ad navigation URL is sent to our data collection app – Depot, which visits the ad navigation URL and then be 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 in case no ad is loaded/rendered.

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

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

[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

ads cannot be avoided. We enable the ARM translator on an x86 machine to run ARM apps.

In order to collect the pre-click data, we customize the Android WebView (20 LoC in Java) to intercept a WebView's network traffic. This allows us to determine if WebView would be used to communicate with a known ad domain. 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, each ad navigation URL is replaced with a custom URL scheme (i.e., MAdLife://) to direct the navigation to Depot, instead of the default mobile web browser. An ad click is generated to let the app switch to Depot or Play Store. In the case of switching to Depot, it takes a screenshot of the landing page, saves its HTML source, and records the redirects if there are any. However, it could not 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 takes a screenshot of the Play Store page, and logs the advertised app's maturity rating (e.g., Everyone, Teen, Mature 17+).

MAdLife can also handle 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 two clicks, first to display a control button and then to trigger the redirection. In such cases, MAdLife calls AndroidViewClient's `dump()` to discover the button with keywords, and then programmatically clicks such ads. Therefore, the record of each ad's lifespan consists of two parts, as shown in Table 1.

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 host 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. We decided to narrow down our analysis space by applying an app selection procedure. Further, 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.

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

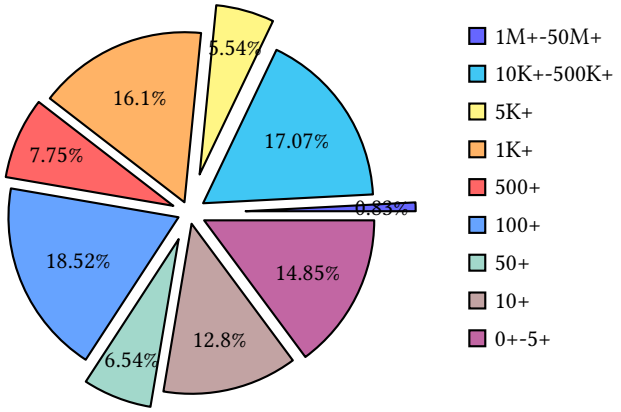


Figure 2: App distribution by downloads

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

Second, we filtered out 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 activities. 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 change of the current foreground activity after their only WebView is clicked, because users are normally redirected to either a landing page or the Play Store app after an ad click. We finally retained 5.7K apps. Moreover, we tried also the method in [19] to identify ads by using WebView sizes. However, more than 300 different WebView sizes in the 11.8K apps' first activities were identified. For example, 320x50 is a standard CSS size for mobile ads, but we also observed ad sizes such as 320x49 and 320x51.

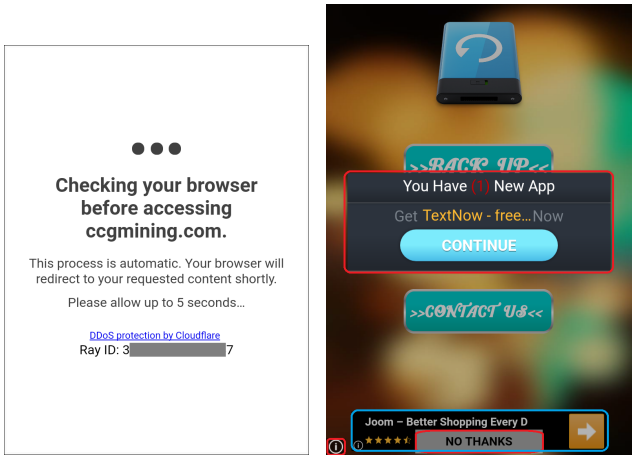
The selected 5.7K apps, developed by over 2.5K developers, exhibit high diversity, in terms of both app categories[10] and number of app downloads (see Figure 2). Compared with multiple statistical results [3, 4, 44], we believe that our app dataset is unbiased.

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 MAdScope [36], even though the subsequent ad requests are similar to their first ones, ads returned by similar requests can be different. Therefore, in order to make sure our ad

[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".



(a) Landing page redirect (b) Overlapped red and blue ads

Figure 3: Special cases encountered during our ad collection

collection was unbiased, we crawled ads for each app in 10 runs at a time, which took about 32-45 seconds to complete per ad. It is worth pointing out that, MAdLife is more efficient, as it took at least 2 minutes for other frameworks [15, 37, 39] to complete a similar task. At the end, we collected nearly 48GB of data, representing over 84K ads.

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 new 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 both ad traffic from the top red WebView and from the overlapped 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 discuss about how we process our collected initial data (Section 4.1), analyze the existing security issues for in-app mobile ads (Section 4.2), and finally explore new threats related to landing pages (Section 4.3).

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 were 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 had 56,914 URLs left for our malvertising analysis (Section 4.2.2). The landing pages collected in Depot belonged to nearly 3.4K distinct advertisers.

In order to conduct further analysis, we matched both pre- and post-click data. In MAdFraud [15], the researchers used a fixed time window to group all data generated by each Android app. However, we were unable to apply a fixed time window, since the time of each ad collection varied. Therefore, a heuristic strategy was devised 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 was found. Finally, we got over 83K matched ads after linking the pre- and post-click data. In particular, 25,764 ads landed directly in the Play Store app, and 57,880 ads were directed to Depot.

4.2 Click Fraud & Malvertising

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

4.2.1 *Click Fraud.*

In our work, we detect if an app automatically takes a user to an ad landing page without any user click. If an app automatically clicks an ad after the app starts, the foreground running Activity may have already been switched to Depot, before MAdLife calls AndroidViewClient's `dump()` to collect the pre-click logs and takes a screenshot of that ad. As a result, in the case of click fraud, MAdLife may lose the pre-click data so that we cannot match the post-click data with any pre-click data, or the captured screenshot in the pre-click data may belong to the landing page instead of the ad. Therefore, we took both situations into account. We found 1) 575 cases belonging to 356 apps without a matched pre-click data; and 2) 132 cases from 38 apps with identical screenshots of a landing page in both the pre-click data and the post-click data. Both situations totaled 372 unique apps. To verify the results, we repeated the experiment over these 372 apps again. This time, only 81 apps were flagged. We found that not all those apps performed click fraud each time we launched them. Through our manual analysis, we confirmed that 37 apps would always try to fabricate an ad click automatically. In total, they were downloaded for more than 660K times. Two of them even obtained over 100K downloads.

We compare our result with that of MAdFraud [15] in Table 2. Both studies demonstrate that click fraud is still not a well-solved problem. Surprisingly, one of the seven fraudulent app developers owned 30 out of the 37 apps. VirusTotal reported that 3 apps were benign. We further looked into their network traffic by running our framework again. We discovered that all samples started loading

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)

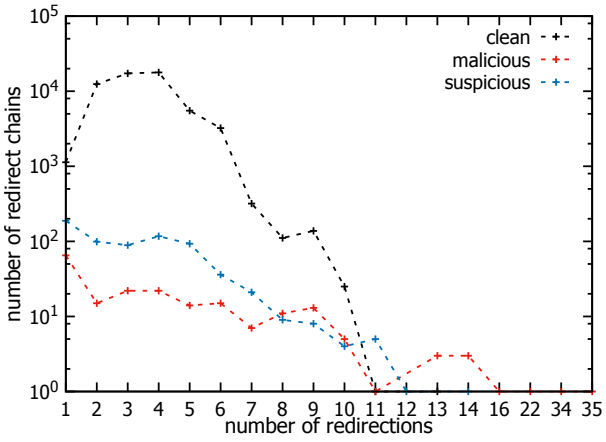


Figure 4: Statistics about different types of redirect chains

ads from three sources, of which two are the apps' local files. The 30 apps always loaded ads starting from the same local file, which we will explain in more detail in Section 4.2.5.

4.2.2 Malvertising.

We used VirusTotal to analyze the 56,914 URLs to determine if the associated ads were malicious or not. 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 were found in 199 redirect chains. The rest of those 1,127 URLs were flagged by one or two scanners. We call them *suspicious* URLs. They belonged to 147 unique domains, and were found in the other 669 redirect chains[11]. In summary, 1.49% of the 57,880 ads were associated with a malicious or suspicious redirect chain. Comparing with a study [39] in 2016 as shown in Table 3, we detected relatively more malvertising domains (65 in 203.7K URLs vs. 64 in 1M URLs). Since we analyzed much fewer apps and collected fewer links from in-app mobile ads, we think that more attackers have participated in malvertising, making the problem worse.

[11] Among these 879 URLs 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

Total Number of Ads	Malvertising Ads by the Click Fraud Apps	Percentage	
Malicious (199)	23	11.56%	15.44%
Suspicious (669)	111	16.59%	
Click Fraud (730)	134	18.36%	

We looked into the results from four aspects. First, similar to the observations in [31, 39], longer redirect chains were more likely to be malicious. This is even more evident with our dataset. Figure 4 depicts the relations between number and length in different kinds of redirect chains. Only two clean redirect chains with all benign URLs had more than 10 hops. On the contrary, the length of a malicious redirect chain can be as high as 35.

Second, only 32.16% of the malicious, and 64.87% of the suspicious redirect chains had non-blank landing pages. Moreover, two of the networks where we set our test environments installed the Palo Alto Firewall[12] software, which blocked 26 ad clicks. We found that 14 of these 26 problematic cases were malicious, and 11 had no redirections. During our ad collection process, no drive-by-download case was encountered.

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

Last, malicious and suspicious redirect chains were detected from 134 and 263 apps, respectively. On the other hand, VirusTotal detected 74 malicious apps and 133 suspicious apps in our 5.7K-app dataset. In particular, 168 apps flagged by at least one VirusTotal scanner were related to malvertising.

4.2.3 Correlation.

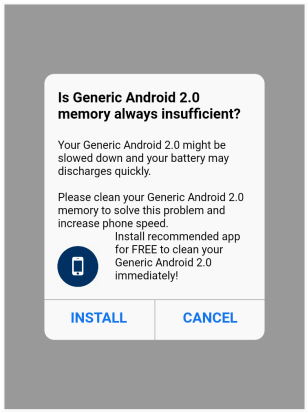
We found statistical correlations between click fraud and malvertising. Table 4 shows the numbers and percentages of malvertising ads loaded by the click-fraud apps. 15.44% of the malvertising ads were loaded by the 37 click-fraud apps. In particular, 11.56% of and 16.59% of the malicious and suspicious redirect chains were collected from those apps, respectively. In total, we collected 730 ads from the 37 click-fraud apps. 134 of them were either malicious or suspicious. Therefore, the click-fraud apps have a 18.36% chance to load malvertising ads. On the other hand, the other 734 malvertising cases were found in the rest 57,150 ads. The non click-fraud apps merely had a 1.28% chance to load malvertising ads. Therefore, users of click-fraud apps are 14.34x more likely to encounter malicious ads compared with users of legitimate apps.

Why are the two ad-related threats correlated? Click-fraud developers may usually select ad networks with insufficient scrutiny of illicit ad behaviors. Attackers would also spread malicious content easily through these careless ad networks. Therefore, ad networks should take more responsibilities (e.g., strengthening their SDKs and scrutinizing malicious ads) to mitigate such risks.

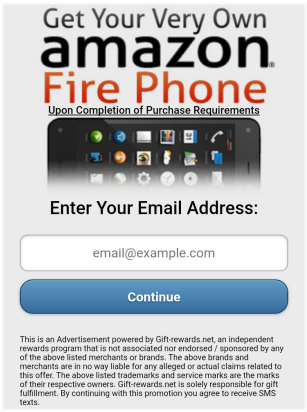
4.2.4 Scam Cases.

Although scams have been well studied in [39], we observed new and interesting cases. Figure 5(a) illustrates an ad for a fake antivirus app. No matter whether users clicked "Install" or "Cancel", the

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

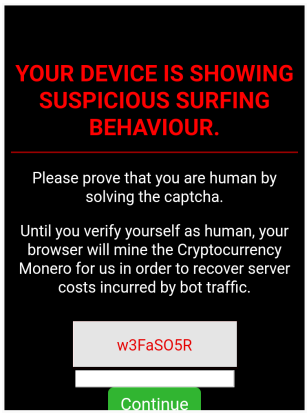


(a) Ad with deceptive download

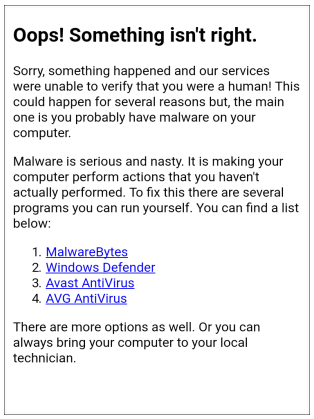


(b) Ad with fraudulent reward

Figure 5: Scam examples



(a) Drive-by cryptomining



(b) Deceptive cryptojacking

Figure 6: Cryptojacking examples

same malicious link would be visited. It revealed that ad viewers were more likely to visit a malicious website with such a trick.

Figure 5(b) shows an unexpected prize and lottery scam, which alleged to offer a free phone. We found two ads with the same image in our dataset. However, these screenshots were loaded from two different malicious domains (i.e., *primerewardz.com* and *premium-rewardsusa.com*), with no redirections. Users should refrain from surrendering their private information, as some scammers asked for email addresses to send a fraudulent reward.

4.2.5 *Cryptojacking Cases.*

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

```
<script src="https://coinhive.com/lib/coinhive.min.js"></script>
<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 6(a)

```
<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 6(b)

The former cases, with up to 9 redirections, involving 13 different ad domains[13]. However, all these redirections closed at one of the two websites – *rcyclmnr.com* and *rcyclmnrepv.com*. Listing 1 shows the COINHIVE[14] script used in 6(a). We submitted the embedded JavaScript file[15] to VirusTotal, where 33 out of 58 scanners reported this script. We later found out that our discovery was confirmed by both Symantec [30] and Malwarebytes [40].

The latter cases included at most 2 redirections, originated from the same domain (*ezanga.com*), and ended up with *dropped-click.com*. A closer inspection revealed two interesting tactics used by these attackers: 1) the website provided benign links for users to click; and 2) the website used three different cryptomining scripts. One example was with *coinhive*; whereas, two instances were with *jsecoin.com*, and another two cases were with *claimers.io*, as shown in Listing 2. To the best of our knowledge, we were the first to find the *claimers.io* cases. According to the landing page HTML, the last two cases were undergoing landing page redirects; therefore, we had no further information about those web pages.

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

Second, fraudulent app developers were involved as well. Specifically, 6 apps among our revisited samples belonged to the aforementioned developer, who owned 30 click-fraud apps. Our first impression of the developer's apps was 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, or pornographic chat rooms). We found that a local file, named EXIT.HTML, was called in each of the 6 apps. Although the file was totally clean under VirusTotal, it called the `doStartAppClick()` method at every app start, to automatically trigger a call to visit an ad URL[16]. Ironically, this URL was also

[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 5d514880ad502302dd4bf0ef8da5d38356385d1c43689f6739f6771ed7a4ef73

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

considered benign by VirusTotal. But it served as an entry point for redirecting users to different landing pages. After loading the URL in a web browser, we were redirected to a couple of automatic malware downloading pages[17]. Although on January 15th, 2018 the developer switched to other ad networks and then disabled the auto click feature, these apps were still listed on the Play Store at the time of this writing.

To sum up, the tricks of fraudulent app developers and of untrusted ad networks were continuously evolving. It was 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 security issues, we look into 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 visually-identical pages may be developed using different HTML code, we did not use the traditional cryptographic hashing algorithm to calculate file fingerprints to classify websites. Instead, we used the CETD algorithm [45] to extract main text content and hyperlinks from webpages. We retained 12,036 distinct documents.

Second, some landing page may include empty content. We filtered out pictures with pure white or black background. The file size of such images was usually less than 15KB. Further, the same website with animated content might result in different screenshots in two ad instances. Therefore, we consecutively employed three image fingerprinting methods[18] (i.e., wavelet hashing, perception hashing, and difference hashing) to group similar images. We finally got 6,337 unique screenshots.

4.3.2 Malicious External JavaScript Code.
Advertisers may embed JavaScript code in their landing pages. Some external JavaScript code could be malicious. However, VirusTotal scanners would analyze only the HTML content of a page, excluding its embedded external scripts. Especially, it is very hard for VirusTotal scanners to recognize a known malicious JavaScript snippet after code obfuscation. As a result, the default scan setting could potentially miss malicious content presented to users on a landing page. Therefore, we further leveraged VirusTotal to scan all external scripts on 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), or 1,384 unique landing pages. Among these 5,880 landing pages, we discovered 53 that were related to malvertising.

We discovered malicious external JavaScript snippets loaded from both the first-party hosts and the third-party hosts. For example, a "benign" landing page included more than 41 positively flagged scripts from its own host. In other words, illicit advertisers may avoid traditional scrutiny techniques using such tricks. On the other hand, third-party scripts are dynamically loaded and can be updated without the control of the landing pages. Interestingly, 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 the positively flagged JavaScript URLs.

Table 5: Top 10 domains with positively flagged URLs

# Occurrences	# Unique URL(s)	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 written various acts (e.g., COPPA [12], CIPA [11]) to safeguard children online, including from advertising. In addition, well-known ad companies [20, 23–25, 46] also established content policies for advertisers. But, self regulation is not enough. 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 contained 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 did 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 Noncompliance: 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)

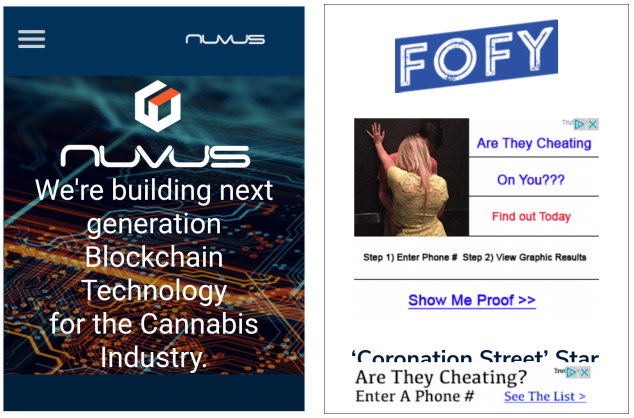
[17] Their VirusTotal SHA-256's are 0c6e40eb1c3b00de1c72f22dec5cffddc3df66672360f79d54d9922c018f4aa6 and 1654cf25365332198c84f7e1e17b237abf0447a97e18800cc349e91cc2eb9a71

[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 Noncompliance	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%



(a) Controversial topics in an ad (b) Ads within an adware

Figure 7: Inappropriate ad examples

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)

Table 6 depicts the statistics based on unique ads and total number of ads in each category. 345 out of 7,067 unique ad landing pages are in the "Policy Noncompliance" category, and 24 are in the "Public Concerns" category; 4,577 (7.9%) out of 57,880 ad landing impressions are in the "Policy Noncompliance" category, and 154 (0.26%) are in the "Public Concerns" category.

We also leveraged the Google Vision API with our screenshots to detect offensive content, extract web entities and find 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 present only the analysis result using the Natural Language API.

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 demonstrate the aggravating circumstances, as depicted in Figure 7. First, along with their price volatility, cryptocurrencies attracted attention from people all over the world. As a result, the entire industry suddenly boomed. During our research, we collected ads of 39 different cryptocurrencies and ads 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. At the time of our experiment, marijuana was not legalized at the federal level in the United States and in Canada [48]. Such ads should thus have been 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 ads and their landing pages, their arms may not be able to reach at nested ads. To the best of our knowledge, we do not find any policies to directly regulate such a situation. As a result, inappropriate ad content may be broadcasted within the landing pages that subscribe to well-known ad networks. Moreover, we consider it as an adware, because for more than 30 FOFY landing pages we collected, only ads were shown within the screenshots but the real content was displayed underneath the ads. Accordingly, the situation of inappropriate ads was getting worse.

4.3.5 *Takeaways.*

Our studies revealed that embedded external scripts in landing pages should also be examined for two reasons: 1) illicit advertisers might use this approach to escape from ad networks' security checks; and 2) third-party scripts might change their code arbitrarily. Therefore, rather than checking only inappropriate ad content, ad networks should also scrutinize the JavaScript files embedded by advertisers. Furthermore, nested ads could be used to bypass new content-based regulations.

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. On the contrary, after banning cryptocurrency ads for a few months, Facebook granted Coinbase a privilege to allow its crypto-related ads again in July 2018 [29]. Ad policies thus might be more inconsistent than what people had expected.

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 ads is necessary for studying 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-effective for advertisers. Moreover, we

designed our experiment in a way that MAdLife's artificial behaviors can be easily detected by ad networks (i.e., 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 split their logcat traffic. Third, to the best of our knowledge, the Google Cloud APIs are among the best APIs available on the market. We have also 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 web pages. Last, Google asks that, ads displayed in family apps should be consistent with their maturity ratings [26]. The researchers in [10] talked about this issue. However, regardless if the parental control settings are on or off, children are able to access all installed apps on each device. Therefore, our studies only focused 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 in the web settings. In online advertising, researchers can either run scripts with their add-ons [7, 31] or build a Selenium-based crawler [50]. They can also intercept HTTP traffic [31, 50] or harvest only ad-related visual elements [7]. Nevertheless, ads' lifespans 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 is more difficult to track an ad's lifespans. For example, since no user interaction (except for click fraud) was involved in [15], their studies stopped at ad loading/rendering. Besides, both MAdScope [36] and the previous work related to in-app malvertising [39] considered only events after a touch.

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 a previous work [35] depicted two clickbot families, a work related to ghost clicks [2] traced from an IP address 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 had focused 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 study of redirect chains would be more beneficial. With the help of studies like [33], authorities would be able to take down malvertising networks without case-by-case investigations.

Malicious JavaScript Detection. The studies of detecting malicious JavaScript code are numerous. JSAND [14] conducted classification based on static and dynamic features, and instrumented JavaScript runtime environments to detect the execution of malicious scripts. Prophiler [8] statically analyzed features of the HTML page, of the embedded JavaScript code, and of the associated URL, and examined malicious content on massive webpages with obfuscated JavaScript. Similarly, ZOZZLE [16] leveraged features associated with JavaScript context to detect unobfuscated exploits. As for code obfuscation, conditional code obfuscation could be used to hinder malware analysis [42].

Content Analysis. Nowadays, contextual advertising is used to target users. A work [49] described a feature-based keyword extraction system for online contextual advertising. In order to analyze web content, TrackMeOrNot [34] used the Alchemy API to label all crawled webpages, and categorized them into different topic categories. Likewise, we used the Google Natural Language API to analyze our dataset before applying the CETD algorithm [45] to only encompass web content and hyperlinks. Both DECAF [32] and SmartAds [37] captured third-party apps' page content on Windows Phone. While the former extracted page information with the UI extraction channel, the latter instrumented app binaries to insert custom logging code. Unlike our work that evaluates ad inappropriateness in general, another work [10] studied whether ads were consistent with host apps' maturity ratings with below 4K ads. Also, their "topic classification" part was not well elaborated such that we cannot make any technical comparison.

7 CONCLUSION

Although implemented on top of existing web technologies, mobile in-app ads face similar threats in different forms. We developed MAdLife– a data collection framework on Android to record all necessary data generated within an ad's entire lifespan. Using MAdLife, we explored the abusive practices, including click fraud, malvertising, and other threats related to landing pages, with 58K ads. We discovered 37 click-fraud apps and over 860 malicious ads. More importantly, we revealed that users of click-fraud apps were 14.34x more likely to encounter malicious ads compared with users of legitimate apps. Further, we observed that 250 widely used JavaScript snippets were harmful to end users. These snippets were used on over 10% of the landing pages in our dataset. Lastly, we identified that 8% ads were inappropriate, either not complying with ad policies or including controversial content. We suggest that ad networks should, and only they can, take more effective measures to protect mobile users and themselves from 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 coinbase 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.