Dissecting Mobile Offerwall Advertisements: An Explorative Study

Guosheng Xu[1], Yangyu Hu[1], Qian Guo[1], Ren He[1], L i L i [2], Guoai Xu[1]*, Zhihui Han[3], and Haoyu Wang[1]
[1]Beijing University of Posts and Telecommunications, Beijing, China
[2]Monash University, Australia
[3]CNCERT, Beijing, China

***Abstract*—Mobile advertising has become the most popular monetizing way in the Android app ecosystem. Offerwall, as a new form of mobile ads, has been widely adopted by apps, and a number of ad networks have provided such services. Although new to the ecosystem, offerwall ads have been criticized for being aggressive, and the contents disseminated are prone to security issues. However, to date, our community has not proposed any studies to dissect such issues related to offerwall ads. To this end, we present the first work to fill this gap. Specifically, we first develop a robust approach to identify apps that have embedded with offerwall ads. Then, we apply the tool to 10K apps and experimentally discover 312 offerwall apps. We go one step further to characterize them from several aspects, including security issues. Our observation reveals that offerwall ads could indeed be manipulated by hackers to fulfill malicious purposes.**

***Index Terms*—Offerwall, Mobile Advertising, Android, Pay-per Install, Malware**

I. Introduction

Mobile apps have seen widespread adoption in recent years. The number of apps in Google Play has exceeded 2.7 million mark as of August 2019 [1]. A recent study [2] suggested that the app monetization scheme evolves during the evolution of the mobile app ecosystem – the number of paid apps has decreased significantly in Google Play, while mobile advertising is getting more and more popular. Indeed, mobile advertisements have been widely adopted by app developers to make profits from their freely released apps. It is estimated that the overall market volume of mobile advertising would achieve 243 billion US dollars by 2022 [3].

Mobile ads can be displayed in different forms. In general, there are three common ways to display mobile ads: 1) **Banner ad** is located in the top or bottom of the screen; 2) **Interstitial ad** is square and located in the center of the screen; 3) **Full-screen ad** fills the whole screen. In these forms, the ad views are either embedded in the UI of apps or popped up during execution. A number of previous studies have analyzed the security and privacy behaviors of traditional mobile advertising, including ad fraud detection [4]–[6], malicious ad content [7], [8], and privacy escalation of ad libraries [9], [10], etc.

With the evolution of mobile advertising, new forms of ads are emerging, with the goal of attracting more user clicks. **Offerwall ad** is such a popular type innovated for mobile advertising in recent years. An offerwall ad is a UI page that appears within the hosting app that offers users rewards or different incentives in exchange for spending money. For example, Figure 1 shows a motivating example of offerwall ads from the third-party library "Adwo", which can award mobile users in-game currency in exchange for downloading other apps (i.e., incentive installs). Unlike other kinds of mobile ads, offerwall ads are particularly effective and convenient because the advertising comes right to the user, usually in the form of a landing page, and they do not have to go out of their way to click a banner or separate ad view. Thus, a number of popular ad networks (e.g., Tapjoy, Airpush, and Chartboost) have provided such kind of ad forms in their released ad SDKs and allow developers to create an offerwall and insert it into the apps they developed.

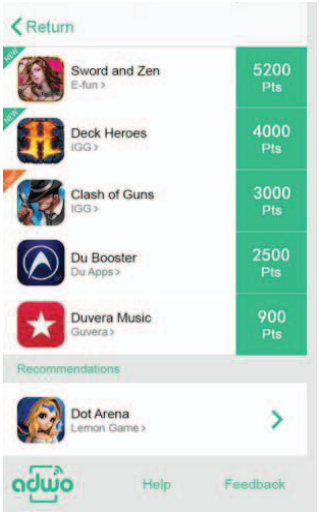

Fig. 1. A Motivating Example of Mobile Offerwall Advertisement.

Despite being effective in attracting ad-based revenues, offerwall ad is criticized for being aggressive, as it completely interrupts the behavior of mobile app and cannot be ignored by users. Also, as most of offerwall ads charge on pay-per-download basis (e.g., offer virtual rewards in exchange for downloading free apps), which may provide a way for developers to manipulate the app ranking in app markets. As a result, it is reported [11] in May 2019 that Apple started banning offerwall ads that incentivized rewards for app downloads. For example, apps that have embedded Tapjoy ad SDK would be rejected by Apple.

*Corresponding author

978-1-7281-8913-0/20/$31.00 ©2020 IEEE
DOI 10.1109/QRS51102.2020.00072

Besides the aggressive ad behaviors, the contents disseminated (i.e., downloaded apps) in the offerwall ads are possible to introduce security issues. Many advertisers (developers) rely on such kinds of incentive installs to promote their apps and help boost their app rankings, *so do the malicious developers and attackers*. *It is quite possible that attackers can leverage such kinds of app promotion channel to distribute malicious apps and infect unsuspicious users*.

This paper. To the best of our knowledge, the mobile offerwall ads have not been systematically studied, and it is unclear to us the issues introduced by offerwall ads in the wild. To this end, we propose to perform *the first characterization study of mobile offerwall ads*. We seek to uncover the characteristics of mobile offerwall ads from two aspects. We first propose to identify mobile offerwall ads, and collect apps that take advantage of them. We then propose to study aggressive and malicious behaviors in mobile offerwall ads.

Unfortunately, it is not straightforward to achieve the aforementioned objectives. Indeed, as the ad libraries (SDKs) provided by ad networks usually provide different forms of mobile ads (e.g., banner ads and offerwall ads), it is non-trivial for us to identify which kinds of mobile ads were actually used in the corresponding apps. To resolve this challenge, we first provide a robust semi-automated method to identify *offerwall apps, deemed as such apps that embedded offerwall ads*. We have manually analyzed over 60 mobile advertising networks to select the ones that provide offerwall advertising services and then summarized the key characteristics by analyzing the provided developer documents. In this way, we have created a detector to accurately identify apps that have offered offerwall ads. We then applied the detector to over 10K Android apps that were flagged containing mobile ads, and from which we have identified over 300 apps that use 20 kinds of offerwall ad services. Based on this dataset, we further characterize the offerwall ads from various aspects. We have observed the following main findings in this paper:

- **Offerwall ad services have exposed security and privacy issues.** Most of the studied offerwall ad services have accessed into undocumented permissions, suggesting that these offerwall services could take advantage of these permissions to covertly access sensitive data. Over 90% of the offerwall apps were flagged as malware by at least one anti-virus engine on VirusTotal.
- **Offerwall ad services could be maliciously used to deliver malware**. By looking into the apps distributed over offerwall ad services, we have identified porn apps and gambling apps, indicating no censorship applied by such ad services to control the apps to be distributed (and hence may distribute malware). Furthermore, a large portion of the distributed apps are not with their latest versions, and they could contain bugs and vulnerabilities.

To the best of our knowledge, this is the first work proposed to systematically study the behaviors of mobile offerwall advertisements. We have proposed an automated tool to identify offerwall-related apps and revealed various interesting findings based on a thorough investigation of such apps reported by our tool. Our experimental results suggest that offerwall ads could be manipulated by hackers to fulfill malicious purposes. We believe that our efforts can positively contribute to bringing the awareness of user and developer, attracting the focus of the research community and, and promoting best operational practices across app store operators and regulators.

II. Detecting Offerwall Apps

As mobile offerwall ads have not yet been systematically explored in existing studies, there is no existing dataset of offerwall apps that we can study. Furthermore, the characteristics of offerwall ad services have been generally unknown as well, which made it hard for us to identify offerwall apps in the wild. One major ***challenge*** we face here is that, as ad networks usually provide various kinds of mobile ads for app developers in the form of ad libraries, it is hard for us to know whether the developers have invoked the offerwall ads in their apps. Thus, we cannot directly adopt existing ad library detection tools to identify offerwall apps. To fill this gap, we first propose a robust semi-automated approach to identify offerwall apps.

Fig 2 presents the working process of our approach for identifying offerwall apps. First, we collect 62 most popular mobile ad networks in both China and worldwide from AppBrain [1] and LibRadar [12], [13], and manually browse their websites and library documents to confirm whether they provide offerwall ad services. For the mobile advertisers who offer offerwall ad services, we manually download their developer documentations and corresponding SDKs to summarize exclusive characteristics from multiple perspectives. *Note that, we especially focus on the features that could be used to differentiate the offerwall ads and other types of mobile ads.* After that, we leverage a two-phase detection technique to check the summarized characteristics against large-scale market apps, in order to identify offerwall apps. We now give more detail about these processes.

A. Analyzing the Static Features of Offerwall Ad Services

1) Collecting Offerwall Ad Services: Here, we resort to the ad libraries labeled by AppBrain [1] and LibRadar [12]. We harvest 62 popular mobile ad networks in total. By manually analyzing the content of their websites, we have identified 29 mobile ad networks that provide offerwall ad services, as shown in the Table I. Then, in the following, to perform automated detection of offerwall apps, we further analyze the features of such offerwall ad services.

2) Summarizing the Static Features: We download and analyze their developer documentation and corresponding SDKs. For each offerwall ad service, we extract its characteristics from four perspectives, including its package name, the configuration features defined in the Android Manifest file, and the API features in the code, as shown in Table I.

(1) Package Name. We first extract the package name of each ad network that provides offerwall ad services. Note that the package name could not be simply used as the features to

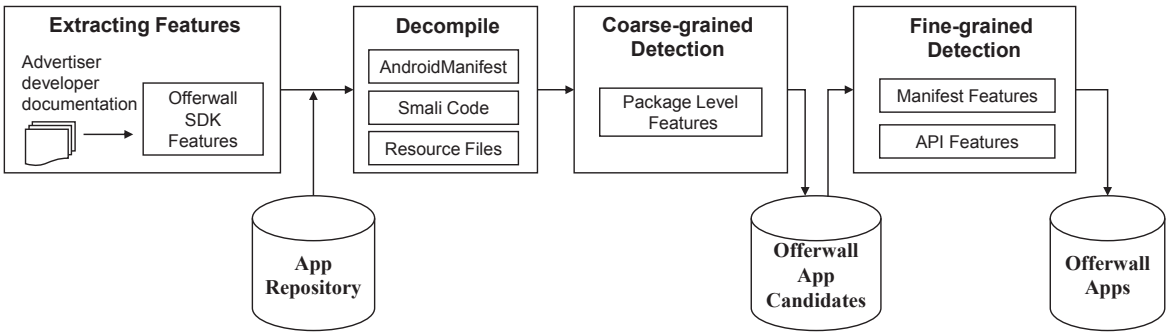

Fig. 2. Our approach to identify offerwall apps.

identify offerwall ads since Android apps can be obfuscated. Indeed, the package names would be obfuscated to meaningless strings (e.g., a.b.c). However, we could combine it with the existing third-party library detection tools to filter offerwall app candidates. State-of-the-art third-party library detection tools (e.g., LibRadar [12]) could perform obfuscation-resilient detection of third-party libraries. Thus, in our detection, we have combined LibRadar with the package names we labeled to first identify apps that have embedded with such libraries.

(2) Multiple Ad Types. Some ad networks simply provide an SDK for different types of ads (e.g., banner ads, video ads, and offerwall ads), and apps need to invoke different methods in the SDK to show appropriate ads. As shown in Table I, 12 ad networks belong to such a category. Therefore, for those advertisements, we need to extract fine-grained features to identify offerwall apps.

(3) Configuration Features in the Manifest To obtain and show the ads, offerwall ad services generally require apps to declare typical activities, services, and permissions in the file of AndroidManifest.xml. We extract features of these typical declarations from the app manifest file, as shown in column 3 of Table I.

(4) API Features Apps show offerwall ads by invoking corresponding APIs. Thus, we further extract the strings of the offerwall-related APIs from the developer documentation as features, as shown in column 4 of Table I.

B. TWO-PHASE Offerwall App Identification

Based on the summarized features we propose a two-phase method to identify offerwall apps from a large number of apps hosted in app markets. As shown in Figure 2, in order to achieve fast identification of offerwall apps, we first performed coarse-grained detection by detecting whether the offerwall ad SDKs are used. We take advantage of LibRadar [12] to identify third-party libraries used in Android apps, to decompile the app and confirm whether it has used the offerwall ad SDKs. However, as shown in column 5 of Table I, 12 advertisers provide the same SDK for different types of ads. For these advertisers, we conduct a fine-grained offerwall app identification based on the features we summarized in Table I.

III. Characterizing Offerwall Advertising

After introducing the prototype tool for automatically detecting offerwall apps, we further propose to investigate the mobile offerwall ads to better understand their characteristics. In this work, we seek to achieve this purpose by answering the following three research questions (RQs):

- ***RQ1: Will the offerwall ad services introduce security and privacy issues?*** As previous work suggested that third-party services are largely invisible to users, it is important to characterize the security and privacy aspects of offerwall ad services. Besides, there is also a need to understand how many of them would be flagged as adware or malware by existing anti-virus engines.
- ***RQ2: What are the contents disseminated over offerwall ad services? Are there any malicious contents?*** We are interested in the contents (apps) disseminated over the offerwall ad services. More specifically, since we do not know if the apps promoted through offerwall ads are trustworthy, it is also worthwhile to characterize the disseminated apps.

A. Dataset

To answer the aforementioned three research questions, we need to first collect a set of apps embedded with offerwall ads. However, since offerwall apps have never been explored before, our community does not have such a dataset readily for analyses. To this end, we resort to our own efforts to collect such a dataset by applying our tool to a dataset of 10K Android apps collected from Google Play and Chinese third-party markets [13], [14].

Eventually, we are able to obtain 312 offerwall apps. Table II shows the distribution of the apps we identified. Out of the 29 advertisers mentioned above, we eventually obtained 312 apps covering 20 advertisers. Some apps integrate more than one offerwall ad services. Specifically, five offerwall ad services have been invoked by more than 50 apps, and an offerwall ad service provided by advertiser "DOMOB" has been invoked by 99 apps.

B. RQ1: Security Analysis of Offerwall Apps

We then perform a general security analysis on the collected offerwall ad services and offerwall apps. We first investigate

TABLE I
THE FEATURES SUMMARIZED IN THIS PAPER TO IDENTIFY ANDROID OFFERWALL SDKS.

Ad Network (URL)	Package Level Feature	Manifest Feature	API Feature	Multiple Ad Types
Paymentwall (https://www.paymentwall.com)	com/paymentwall/pwunifiedsdk	activity.OfferwallActivity	NA	✓
Airpush (https://airpush.com)	com/sgppnqgub/bmvjlevxs323254	SmartWallActivity	airPlay.startSmartWallAd airPlay.startAppWall	×
Adscend Media (http://adscendmedia.com)	com/adscendmedia	ui.OffersActivity	OffersActivity.*	×
ironSource (https://www.ironsrc.com)	com/ironsource/mediationsdk /sdk/OfferwallListener	controller.ControllerActivity controller.InterstitialActivity	IronSource.showOfferwall IronSource.setOfferwallListener	×
Tyroo (http://www.tyroo.com)	com/tyroo/tva/sdk	activities.DiscoverActivity activities.AdViewActivity	NA	✓
Chartboost (https://www.chartboost.com)	com/chartboost/sdk	sdk.CBImpressionActivity	NA	✓
DOMOB (https://www.domob.cn)	cn/dow/android	cn.dow.android.Dservice	NA	✓
Youmi (https://www.youmi.net)	com/youmi/android net/youmi/android	NA	showOffersWall showOffersWallDialog	×
IMmob (http://www.immob.cn)	cn/immob/sdk	net.DownloadService service.GetLocationService AppChangeBrocastreceiver	AdType.WALL	×
Guomob (http://www.guomob.com)	cn/gm/tasklist	gm.tasklist.AlertActivity gm.tasklist.Service01 gm.tasklist.MyBrocastReciver	OpenIntegralWall	×
Yijifen (http://www.yijifen.com)	com/eadver/offer/sdk	sdk.util.AdScoreService sdk.view.EadverReceiver sdk.view.WallActivity	RecommendWallSDK	×
WAPS (http://www.waps.cn)	cn/waps	NA	showAppOffers showGameOffers	×
Miidi (https://www.miidi.net)	net/midi/wall/sdk	sdk.MyWallActivity	AdWall showAppOffers showAppOfferNoScore	×
BillionMobi (http://www.chinazmob.com)	com/zy/phone	sdk.SDKActivity service.BootReceiver service.ZYService	NA	✓
WQMobile (http://weiqiandongmei.lofter.com)	com/wqmobile/sdk	csdk.WQActionHandler sdk.WQBrowse	openAdWall	×
Kuguo (http://www.kuguopush.com)	com/pkfg	pkfg.k.MyKAActivity pkfg.k.MyKBActivity pkfg.k.MyKReceiver	showKuguoSprite	×
Dianjoy (http://www.dianjoy.com)	com/dlnetwork	dlnetwork.DevNativeActivity dlnetwork.DevNativeService	NA	✓
iMopan (http://www.imopan.com)	com/imopan/ad	sdk.ProxyService	NA	✓
AppDriver (http://www.appdriver.com.cn)	net/adways/appdriver2	offerWall.OfferWallActivity	AppdriverManager()	×
Mobsmar (http://www.mobsmar.com)	com/ZMAD	score.ScoreActivity score.PopDetailActivity score.PackageInstallService	NA	×
JUZI (http://www.juzichuanmei.com)	com/JUZI	JUZI_APPID	NA	✓
Tapjoy (http://www.tapjoy.cn)	com/tapjoy	TJAdUnitActivity mraid.view.ActionHa	NA	✓
Adhub (http://adhub.com.cn/index.html)	com/hubcloud/adhubsdk	adhubsdk.AdActivity	NA	✓
Dianmoney (http://www.dianmoney.com)	com/dc/wall	dc.wall.DcActivity dc.wall.AC dc.wall.DC	NA	×
Adwo (http://www.adwo.com)	com/adwo/adsdk	NA	NA	✓
Chance (http://www.chance-ad.com)	com/chance	chance.ads.AdActivity engine.ChanceAdService chance_publisherid	NA	×
Baidu (http://e.baidu.com/product)	com/baidu/appx	uikit.BDActivity app_download.CompleteReceiver	NA	×
Tencent (https://e.qq.com/ads/?from=02_PINPAI_145)	com/qq/e	comm.DownloadService ads.ADActivity	NA	×
Datouniao (http://www.datouniao.com)	com/datouniao/AdPublisher	AdsOffersWebView service.AdsService	NA	✓

how offerwall ad services request permissions to access sensitive information. As previous work suggested that permission escalation was found in Ad libraries, we seek to investigate whether the offerwall ad services have permission escalation behaviors. Then, we rely on VirusTotal [15], a widely used

TABLE II
OVERALL RESULT OF APP IDENTIFICATION

Ad Network	# Apps	# (%) AV-rank$\geq$ 1	# (%) Adware
DOMOB	99	87(87.9%)	84(84.8%)
BillionMobi	96	86(89.6%)	82(85.4%)
Dianmoney	89	85(95.5%)	80(89.9%)
Datouniao	87	77(88.5%)	73(83.9%)
WAPS	78	74(94.9%)	72(92.3%)
Youmi	48	40(83.3%)	40(83.3%)
IMmob	39	36(92.3%)	32(82.1%)
Guomob	38	30(78.9%)	27(71.1%)
Mobsmar	27	25(92.6%)	24(88.9%)
Miidi	20	18(90.0%)	16(80.0%)
Yijifen	13	11(84.6%)	10(76.9%)
Adscend Media	5	5(100.0%)	3(60.0%)
Tapjoy	5	2(40.0%)	2(40.0%)
AppDriver	3	3(100.0%)	3(100.0%)
Dianjoy	2	2(100.0%)	2(100.0%)
chartboost	2	1(50.0%)	1(50.0%)
JUZI	1	1(100.0%)	1(100.0%)
Chance	1	1(100.0%)	1(100.0%)
Tencent	1	0(0.0%)	0(0.0%)
WQMobile	1	1(100.0%)	1(100.0%)
Total	312	297(95.2%)	239(76.6%)

anti-virus service to understand whether these offerwall ad services would be flagged as malicious by anti-virus engines.

1) RQ1.1: Permission Analysis: Previous work [16] has suggested that permission misuse is prevalent in the third-party libraries of mobile apps. As third-party libraries share permissions with the host apps, ad libraries can take advantage of the sensitive permissions requested by the host apps to covertly access sensitive data. To ensure these undocumented uses do not cause the application to crash, ad libraries can dynamically check if they have some specific sensitive permissions or catch a thrown SecurityException.

Permission Classification. Thus, we have classified the permissions related to each offerwall ad SDK into three categories as previous work [16]: 1) **required permissions**, each ad library specifies the permissions it requires to operate; 2) **optional permissions**, the ad library will further specify a number of optional permissions that the ad library can take advantage of, in order to deliver more targeted advertising; and 3) **undocumented permissions**, which are the permissions that libraries would trigger at runtime but without specifically declaring them in the document. As the permissions in Android apps have shared between the custom code and third-party libraries, ad libraries may check the permissions of host apps, and then take advantage of these permissions.

Approach to identify permission misuse. We first investigate how offerwall ad services request sensitive permissions. For each offerwall ad service, by reviewing the SDK documentation, we have obtained a list of required and optional permissions. Then, we perform static analysis to extract the permissions each library used in practice in the apps we harvested by checking the permission-related sensitive APIs. PScout has provided an internal mapping between API methods and required permissions [17]. By comparing the used permissions and required permissions, we could then get the list of undocumented permissions.

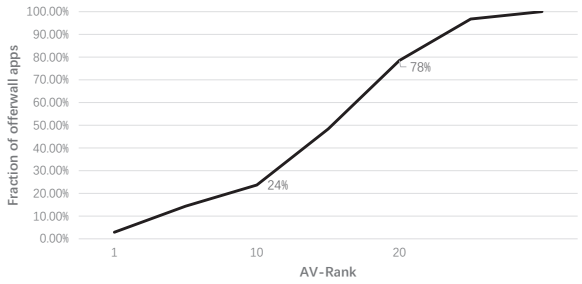

Fig. 3. Distribution of Apps according to AV-Rank.

Result. Table III presents the experimental results. Most offerwall ad services require access permissions like "INTERNET", "READ_PHONE_STATE", "ACCESS_NETWORK_STATE" and "ACCESS_WIFI_STATE". However, 12 of the 20 (60%) studied offerwall ad services have accessed to undocumented permissions. The "Datouniao" and "WQMobile" offerwall libraries are particularly noteworthy among the set we analyzed because they utilized four undocumented permissions, including "ACCESS_FINE_LOCATION" and "ACCESS_COARSE_LOCATION". This result suggests that these offerwall services could take advantage of these permissions to covertly access sensitive data. As long as the host apps have declared the relevant sensitive permissions, such mobile offerwall services could leverage them to access sensitive data without the knowledge of mobile users.

2) RQ1.2: Malware Presence: We further seek to explore whether the offerwall apps would be reported as malware or adware by existing anti-virus engines. To this end, we upload all the 312 apps to VirusTotal [15], an online analysis service that aggregates more than 60 anti-virus engines. As previous studies [18], [19] have shown that some anti-virus engines may not always report reliable results, thus we analyzed the result grouped by how many engines (AV-rank) flag an app as malware, in order to cope with such potential false positives.

Overall Results. Fig 3 shows the overall detection results. Remarkably, as shown in Table II, around 95% (297 apps) of the collected apps are flagged by at least one anti-virus engine. When using 10 as the threshold of $AV-Rank$, around 76% of the apps (237 apps) are labeled much more than that. Especially, around 21% apps (66 apps) are flagged as malware by more than 20 anti-virus engines. This result shows that most of the offerwall apps are identified as malicious apps.

Malware Category and Malware Family. We find the malware categories mainly correspond to 4 types: Adware (77%), Trojan (65%), Android-PUP (43%), and Android-PUA (27%). As shown in Table II, it is interesting to observe that 76.6% of the apps (239 apps) are identified as Adware [20] by at least one anti-virus engines. However, 44 offerwall apps in our dataset failed to be identified, even if some of them use the same offerwall ad services with identified offerwall apps. We then use AVClass [21] to extract the family name (label)

TABLE III

OFFERWALL AD SDK PERMISSION USAGE AS SPECIFIED IN THEIR ONLINE DOCUMENTATION, AD LIBRARIES MAY REQUIRE A PERMISSION (R) OR DECLARE IT OPTIONAL, BUT USE IT IF AVAILABLE (O). WORRYINGLY, SOME AD LIBRARIES CHECK FOR AND USE UNDOCUMENTED PERMISSIONS (X).

Ad Library	INTERNET	READ_PHONE_STATE	VIBRATE	ACCESS_WIFI_STATE	GET_TASKS	ACCESS_NETWORK_STATE	WRITE_EXTERNAL_STORAGE	ACCESS_FINE_LOCATION	ACCESS_COARSE_LOCATION	WAKE_LOCK	BLUETOOTH	READ_EXTERNAL_STORAGE	CHANGE_CONFIGURATION	MOUNT_UNMOUNT_FILESYSTEMS	SYSTEM_ALERT_WINDOW	PACKAGE_USAGE_STATS	RESTART_PACKAGES	RECEIVE_BOOT_COMPLETED	DOWNLOAD_WITHOUT_NOTIFICATION
DOMOB	R	R		O		O	O					O	O						
BillionMobi	R	R	R	R	X	R	R			O				O					
Dianmoney	R	O		O	O	R	O								O	O			
Datouniao	R	R	X		X	R	O	X	X										
WAPS	R	R	X	R	R	R	O												
Youmi	R	R		X		R	R	R	R			O							
iMmob	R	R	X	R	X	R	O	R	R	X									
Guomob	R	R	X	R	X	R	O					O		O					
Mobsmar	R	R	R	R	O	R	O										O	O	
Miidi	R	O			O	R	O	R	R										
Yijifen	R	R	X	R	O	R						O							
Adscend Media	R	R								X									
Tapjoy	R	O		R		R		X	X	X									
AppDriver	R	R	X	R	R		R							O					O
Dianjoy	R	R	X	O	R	R									O				
Chartboost	R	R				R	R					O							
JUZI	R	R		R	O	R	O												
Chance	R	R		R		R	R	R	R			O							
Tencent	R	R		O		R	O	R	R										
WQMobile	R	R	X	R		R	R	X	X		X								

of each identified malware. The "youmi" and "gappusin" families are among the most popular ones, i.e., more than 40% (126) apps (or 25% (78) of malicious apps) belong to them. Nevertheless, we find that family names are not always accurate. In our dataset, we only have 48 apps leveraged the Youmi ad library, while there are 126 apps are labeled as such. This result suggests that existing anti-virus engines cannot well detect and categorize offerwall ad services.

Table IV lists the top 10 malware according to their AV-Rank. For example, the app "cn.fiker.moreMoney" was flagged by 30 anti-virus engines ("gappusion" family [22]), which is actually a trojan that aiming at stealing users' sensitive information. As another example, app "com.rrxszkj", which has embedded the WAPS library, was reported as belonging to the "Youmi" family.

C. RQ2: Characterizing the Disseminated Contents

To answer the second research question, we hence study the contents (i.e., apps) disseminated by these offerwall apps, i.e., what kinds of apps were disseminated over the offerwall ads, and whether this specific app distribution channel could be maliciously exploited. We manually installed all the 312 offerwall apps on real Android devices to check whether the services embedded in these apps are still available. We found that 33 out of the 312 apps have stopped services, and we cannot obtain any content from them. At last, 279 apps can still work properly on real smartphones.

1) Approach.: For the remaining 279 offerwall apps, we design an approach to automatically interact with offerwall apps so as to harvest the apps promoted via these offerwall ad services as much as possible. Considering that offerwall ad services will help users to automatically download the promoted apps when starting a task, we simulate the process of completing tasks to obtain the apps. However, existing automated testing approaches have the limitation of low UI coverage so that it can hardly extract all the promoted apps [23] [24]. Furthermore, if we select script-based automated testing methods, it will take a considerable amount of time to write test scripts for each app manually. Therefore, we

TABLE IV
TOP 10 MALICIOUS APPS THAT HAVE EMBEDED MOBILE OFFERWALL ADVERTISEMENT SERVICES ACCORDING TO AV-RANK.

Package_name	MD5	AV-Rank	Reported family	Advertiser
cn.fiker.moreMoney	2a0c02cbaa7b8c2c65870c550b5fa9e8	30	gappusin	Datouniao
com.kuaizhuan.omgorgtwob	23cc77cba667a1ed5ca1b1c4c3888c96	29	youmi	IMmob;WAPS;Youmi;BillionMobi;Dianmoney
com.dou.zhuandou2	02ba039548a017690ec173bcb68754d4	27	youmi	Youmi;Dianmoney;Datouniao
com.terry.makemoney	7d6be65530a60108dae968b8a1bd834a	27	youmi	Tapjoy;Youmi
com.dconn.tuiqianer	28661a95e0ea31cef727705f0cf391f1	26	youmi	Yijifen;WAPS;Miidi;Youm;BillionMobi; AppDriver;Mobsmar;Datouniaoi
com.rrxszkj	c1d12497cc4efd44e7ebe1ff71263d6d	26	youmi	WAPS
com.sdy.douzhuan	c213f6f027 af0a40a87f70b85f2dfedf	26	gappusin	WAPS
com.meituo.zhuanjifenbao	a9cbff1d64520020ab1819bfa717d1cd	25	gappusin	Yijifen;WAPS;Miidi;BillionMobi;Dianmoney
com.bruceliu.androidmoney	9a536e4ddbfe4e4fe81f53818089d926	25	youmi	WAPS
com.jing.zhuanfengle	d02ea0243bca24ffa4e5b60f3f43ec4d	25	gappusin	IMmob;WAPS

TABLE V
DISSEMINATION APPS IN OFFERWALL.

offerwall	# Apps	# Pkgs (# unique pkgs)	AV-rank (#pkgs) ≥ 1	≥ 10	#pkgs not released in markets	#pkgs are not the latest version
Youmi	126	21 (18)	14	1	6	8
WAPS	305	15 (13)	8	1	12	0
Dianjoy	158	6 (3)	3	1	0	4
Dianmoney	17	17 (12)	10	2	1	4
Datouniao	238	6 (5)	2	1	0	5
BillionMobi	418	21 (16)	9	6	4	8

TABLE VI
MALICIOUS APPS DISTRIBUTED IN OFFERWALL APPS.

package	AV-rank	advertiser
com.expflow.reading	17	Datouniao
me.mizhuan	17	Youmi
com.goldmf.GM	17	BillionMobi
com.cool.ddz	17	Dianmoney
com.anroid.mylockscreen	14	BillionMobi
com.flowerlive.qp	14	Dianmoney
com.xyue.xy	13	BillionMobi
com.andbase.y	11	BillionMobi
com.sgjr.sg	11	BillionMobi
com.inke.gaia	14	Dianjoy
com.sup.android.superb	12	BillionMobi
com.ledexiang.game.ninenine	11	WAPS

propose an improved semi-automated method to generate the test scripts so as to improve the efficiency of app extraction.

Specifically, our proposed method is based on the following findings: the operating process of completing tasks in each offerwall apps is similar (click the widget of a task in the task list, click the install button, go back and continue to choose the next task). To convert the process of downloading all the promoted apps into the corresponding test script, for each offerwall app, we need to get into the UI of displaying offerwall ads, and then extract two values: *a coordinate in the widget of the first task*, and *the height of the task widget*. Based on the two extracted values, we can calculate a coordinate in other task widgets directly by linear superposition, as the height of each task widget is the same. Moreover, we predefined the starting and ending coordinates to help in scrolling down the task list, so as to ensure that all of the promoted apps will be downloaded.

We generate the testing script for each of the offerwall apps (279 apps in total), and convert the script into the "input" operation command [25], which is provided by "Android command-line" for simulating interaction with UI. Considering that offerwall ad services may show different apps on different offerwall apps, to optimize ad recommendation. We generate the testing script for all the apps, even though they may use the same offerwall ad service. For each app, we run the testing script once a day in a week duration.

2) Result.: Eventually, We collect 1,262 APKs from 6 offerwall ad services, as shown in Tab V. Among the collected apps, we found that many of them are redundant ones, which have been promoted on multiple platforms for many times. By comparing the package name of the 1,262 APKs, we obtain 67 unique apps (as shown in column 3 in Tab V). Note that some of them are published on more than one offerwall service, for example, three distinct apps which are published on "Youmi" can also be found on other offerwall services.

Malware Presence. We upload all the 1,262 APKs that are collected from offerwall apps to VirusTotal to examine how many of them are flagged by existing anti-virus engines. The experimental result suggests that 69% of the downloaded apps are labeled as malware by at least one anti-virus engine (as shown in column 4 in Tab V). When using the threshold of "AV-Rank $>=$ 10", around 18% of the downloaded apps are labeled as such (as shown in column 5 in Tab V). Remarkably, as shown in Table VI, four apps (e.g., com.goldmf.GM, com.cool.ddz, me.mizhuan, com.expflow.reading, which are promoted by Youmi, BillionMobi, Dianmoney, and Datouniao, respectively) are labeled by 17 anti-virus engines.

Release Channels. We further check whether these apps are published in popular app markets (e.g., Google Play, popular Chinese app markets). As shown in column 6 in Tab V, we found that 23 apps (34%) cannot be found by directly package name matching, which indicates that these apps have

not ever been published to these markets, or they were already removed [26], [27]. We further manually analyze these apps, and found that 6 of them are labeled as malware, 2 of them are scam financial apps (e.g., com.xyue.xy, com.sgjr.sg), and 12 of them are gambling apps (e.g., com.flowerlive.qp, com.ruiqugames.buyuqianpao, com.school.gdcp365). This result suggests that a considerable number of apps distributed over the offerwall ad services may contain inappropriate contents, indicating that advertisers may have missed the chance to gatekeep the apps disseminated over their ad networks.

Up-to-date versions. For the other 44 apps which can be found in markets, we verify whether it is the latest version by comparing the hashing value of the latest apps downloaded directly from the markets (as shown in column 7 in Tab V). We found that 29 apps (66%) are not the latest version. For example, by the time of our study, the app "com.ninexiu.sixninexiu" disseminated over offerwall "Youmi" (d0c8feb711f8cd790c78a1d16bf3cf81) has the version number "3.8.0.1", while the official app in the market is with version "3.9.4.6" (330eb27a6008725e80865a52eace4f42).

IV. Discussions

Implications. Our findings suggested that offerwall ad services could be used to deliver malicious and illegal contents to mobile users, leading to a poor user experience that harms the reputation of hosting apps. We argue that app markets and regulators should introduce new automated tools to identify such apps and regulate the apps disseminated in them. The features we summarized and the tool we created in this paper are our initial step towards filling this gap. We hence appeal to the community for putting more efforts into exploring this new research direction. From the perspective of app developers, to improve trustworthiness in the ecosystem, as well as to mitigate the possibility of cheating users, developers should resort to certified third-party platforms to implement their offerwall ads or implementing other types of mobile ads. From the perspective of mobile users, they should pay more attention to the apps that have embedded offerwall advertisement libraries and avoid downloading apps from them, as demonstrated by the fact that many of the disseminated apps are malware.

Limitations. Our work still faces several limitations that could be further improved. First, the features we used to identify the offerwall services and offerwall ads were manually crafted, which is straightforward and conservative. It is quite possible that we have missed some of the offerwall services. On the other hand, the number of offerwall apps we studied in this paper is not large enough, i.e., we have considered only 312 apps with 20 offerwall ad services. However, we believe that our approach can be easily extended to take into account more apps. We take that as our future work.

V. Related Work

A. Mobile Ad Library Detection and Analysis

The majority of research studies targeting the mobile ad ecosystem are actually focused on ad libraries. One line of work focuses on identifying ad libraries [12], [28]–[33]. For example, LibRadar [12] and LibD [29] are both clustering-based approach to identify third-party libraries. AdDetect [32] is a machine-learning based approach to identify ad libraries. The other line of work focuses on the security and privacy issues of ad libraries [9], [10], [16], [34], [35]. These studies suggested that ad libraries may perform privacy escalation behaviors due to the shared permission mechanism between custom code and third-party libraries, and a number of studies focus on separating libraries and app code.

B. The Security Analysis of Mobile Advertisements

A few research studies were focused on analyzing the malicious behaviors related to mobile advertisement, including fraudulent behaviors that entice users to click ads or push notifications [4]–[6], [36], [37], and the devious contents that displayed in the ad loading contents and landing pages [38]. For example, Liu et al. [4] have investigated static placement frauds on Windows Phone via analyzing the layouts of apps. Crussell et al. [5] have developed an approach for automatically identifying click frauds (fake impressions and clicks). Dong et al. [6] have studied the new kinds of dynamic ad fraud. However, to the best of our knowledge, no previous studies have detailed characterized the security issues in the offerwall ads. Compared with other types of mobile ads, offerwall ads are more aggressive and posing great security threats, i.e., users are required to download malicious apps.

C. Malicious Web Advertising Analysis

Malicious ads have been extensively studied in the context of web advertising (or often referred to as web malvertising). Cova et al. [39] and Lu et al. [40] proposed to detect drive-by-download attack and malicious Javascripts that embedded in the advertising. Stringhini et al. [41] and Mekky et al. [42] used the properties of HTTP redirections to identify malicious advertisement behavior. Although these approaches have studied malicious ads from one way to another, the security issues in the offerwall ads have not been well studied.

VI. Conclusion

In this paper, we performed the first explorative study of mobile offerwall advertisements. We first proposed a semi-automated approach to identify offerwall services and offerwall apps. Then, we applied the tool to analyze over 10K Android apps and identified over 300 offerwall apps. Leveraging these apps, we go one step deeper to dissect the security and privacy behaviors of their offerwall ads. The experimental results suggest that (1) offerwall ad services have accessed to undocumented permissions, and (2) offerwall ad services could be maliciously used to deliver malware.

Acknowledgment

This work is supported by the National Key Research and Development Program of China (grant No.2017YFB0801903), and the National NSF of China (grants No.61702045).

REFERENCES

[1] "Number of Android Applications," 2019, https://www.appbrain.com/stats.

[2] H. Wang, H. Li, and Y. Guo, "Understanding the evolution of mobile app ecosystems: A longitudinal measurement study of google play," in *The World Wide Web Conference*. ACM, 2019, pp. 1988–1999.

[3] "Mobile Advertising Market," 2019, https://www.alliedmarketresearch.com/mobile-advertising-market.

[4] B. Liu, S. Nath, R. Govindan, and J. Liu, "{DECAF}: Detecting and characterizing ad fraud in mobile apps," in *11th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 14)*, 2014, pp. 57–70.

[5] J. Crussell, R. Stevens, and H. Chen, "Madfraud: Investigating ad fraud in android applications," in *Proceedings of the 12th annual international conference on Mobile systems, applications, and services*. ACM, 2014, pp. 123–134.

[6] F. Dong, H. Wang, L. Li, Y. Guo, T. F. Bissyandé, T. Liu, G. Xu, and J. Klein, "Frauddroid: Automated ad fraud detection for android apps," in *Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. ACM, 2018, pp. 257–268.

[7] V. Rastogi, R. Shao, Y. Chen, X. Pan, S. Zou, and R. Riley, "Are these ads safe: Detecting hidden attacks through the mobile app-web interfaces." in *NDSS*, 2016.

[8] R. Shao, V. Rastogi, Y. Chen, X. Pan, G. Guo, S. Zou, and R. Riley, "Understanding in-app ads and detecting hidden attacks through the mobile app-web interface," *IEEE Transactions on Mobile Computing*, vol. 17, no. 11, pp. 2675–2688, 2018.

[9] M. C. Grace, W. Zhou, X. Jiang, and A.-R. Sadeghi, "Unsafe exposure analysis of mobile in-app advertisements," in *Proceedings of the fifth ACM conference on Security and Privacy in Wireless and Mobile Networks*. ACM, 2012, pp. 101–112.

[10] P. Pearce, A. P. Felt, G. Nunez, and D. Wagner, "Addroid: Privilege separation for applications and advertisers in android," in *Proceedings of the 7th ACM Symposium on Information, Computer and Communications Security*. Acm, 2012, pp. 71–72.

[11] "Devs lose out on thousands of dollars as Apple cracks down on offerwall ads," 2019, https://www.pocketgamer.biz/news/70610/devs-lose-thousands-of-dollars-as-apple-cracks-down-on-offerwall-ads/.

[12] Z. Ma, H. Wang, Y. Guo, and X. Chen, "Libradar: fast and accurate detection of third-party libraries in android apps," in *Proceedings of the 38th International Conference on Software Engineering Companion*, 2016, pp. 653–656.

[13] H. Wang, Z. Liu, J. Liang, N. Vallina-Rodriguez, Y. Guo, L. Li, J. Tapiador, J. Cao, and G. Xu, "Beyond google play: A large-scale comparative study of chinese android app markets," in *Proceedings of the Internet Measurement Conference 2018*. ACM, 2018, pp. 293–307.

[14] L. Li, J. Gao, M. Hurier, P. Kong, T. F. Bissyandé, A. Bartel, J. Klein, and Y. Le Traon, "Androzoo++: Collecting millions of android apps and their metadata for the research community," *arXiv preprint arXiv:1709.05281*, 2017.

[15] "VirusTotal," 2019, https://www.virustotal.com.

[16] R. Stevens, C. Gibler, J. Crussell, J. Erickson, and H. Chen, "Investigating user privacy in android ad libraries," in *Workshop on Mobile Security Technologies (MoST)*, vol. 10. Citeseer, 2012.

[17] W. Y. A. Kathy, F. Yi, H. Zheng, and L. David, "Pscout: analyzing the android permission specification," in *Proceedings of the 2012 ACM conference on Computer and communications security*. ACM, 2012, pp. 217–228.

[18] D. Arp, M. Spreitzenbarth, M. Hubner, H. Gascon, K. Rieck, and C. Siemens, "Drebin: Effective and explainable detection of android malware in your pocket." in *NDSS 2014*.

[19] F. Wei, Y. Li, S. Roy, X. Ou, and W. Zhou, "Deep ground truth analysis of current android malware," in *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*. Springer, 2017, pp. 252–276.

[20] "Adware," 2019, https://www.malwarebytes.com/adware/.

[21] e. a. Sebastián, Marcos, "Avclass: A tool for massive malware labeling." in *International Symposium on Research in Attacks, Intrusions, and Defenses*. Springer, 2016.

[22] "Android/Gappusin.A — ESET Virusradar," 2019, https://www.virusradar.com/en/Android_Gappusin.A/description.

[23] H. Zhu, X. Ye, X. Zhang, and K. Shen, "A context-aware approach for dynamic gui testing of android applications," in *Computer Software and Applications Conference (COMPSAC), 2015 IEEE 39th Annual*, vol. 2. IEEE, 2015, pp. 248–253.

[24] N. I. Azim T, "Targeted and depth-first exploration for systematic testing of android apps," in *ACM SIGPLAN Notices*. ACM, 2013, pp. 641–660.

[25] "Command-line Tool," 2019, https://developer.android.com/studio/test/command-line.

[26] H. Wang, H. Li, L. Li, Y. Guo, and G. Xu, "Why are android apps removed from google play? a large-scale empirical study," in *The 15th International Conference on Mining Software Repositories (MSR 2018)*, 2018.

[27] H. Wang, J. Si, H. Li, and Y. Guo, "Rmvdroid: Towards a reliable android malware dataset with app metadata," in *Proceedings of the 16th International Conference on Mining Software Repositories*, 2019, p. 404–408.

[28] L. Li, T. F. Bissyandé, J. Klein, and Y. Le Traon, "An investigation into the use of common libraries in android apps," in *2016 IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering (SANER)*, vol. 1. IEEE, 2016, pp. 403–414.

[29] M. Li, W. Wang, P. Wang, S. Wang, D. Wu, J. Liu, R. Xue, and W. Huo, "Libd: scalable and precise third-party library detection in android markets," in *2017 IEEE/ACM 39th International Conference on Software Engineering (ICSE)*. IEEE, 2017, pp. 335–346.

[30] M. Backes, S. Bugiel, and E. Derr, "Reliable third-party library detection in android and its security applications," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2016, pp. 356–367.

[31] H. Wang and Y. Guo, "Understanding third-party libraries in mobile app analysis," in *2017 IEEE/ACM 39th International Conference on Software Engineering Companion (ICSE-C)*. IEEE, 2017, pp. 515–516.

[32] A. Narayanan, L. Chen, and C. K. Chan, "Addetect: Automated detection of android ad libraries using semantic analysis," in *2014 IEEE Ninth International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP)*. IEEE, 2014, pp. 1–6.

[33] L. Li, T. Riom, T. F. Bissyandé, H. Wang, J. Klein, and Y. Le Traon, "Revisiting the impact of common libraries for android-related investigations," *Journal of Systems and Software (JSS)*, 2019.

[34] E. Derr, S. Bugiel, S. Fahl, Y. Acar, and M. Backes, "Keep me updated: An empirical study of third-party library updatability on android," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2017, pp. 2187–2200.

[35] A. G. Aritz, G. C. Jose, C. Angel, C. Maria, and C. R. Ruben, "Large-scale analysis of user exposure to online advertising on facebook," *IEEE Access*, vol. 7, pp. 11 959–11 971, 2019.

[36] F. Dong, H. Wang, L. Li, Y. Guo, G. Xu, and S. Zhang, "How do mobile apps violate the behavioral policy of advertisement libraries?" in *Proceedings of the 19th International Workshop on Mobile Computing Systems & Applications*, 2018, pp. 75–80.

[37] T. Liu, H. Wang, L. Li, G. Bai, Y. Guo, and G. Xu, "Dapanda: Detecting aggressive push notifications in android apps," in *2019 34th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 2019, pp. 66–78.

[38] T. Liu, H. Wang, L. Li, X. Luo, F. Dong, Y. Guo, L. Wang, T. F. Bissyande, and J. Klein, "Maddroid: Characterising and detecting devious ad content for android apps," in *Proceedings of the Web Conference 2020 (WWW'20)*, 2020.

[39] M. Cova, C. Kruegel, and G. Vigna, "Detection and analysis of drive-by-download attacks and malicious javascript code," in *Proceedings of the 19th international conference on World wide web*. ACM, 2010, pp. 281–290.

[40] L. Lu, V. Yegneswaran, P. Porras, and W. Lee, "Blade: an attack-agnostic approach for preventing drive-by malware infections," in *Proceedings of the 17th ACM conference on Computer and communications security*. ACM, 2010, pp. 440–450.

[41] G. Stringhini, C. Kruegel, and G. Vigna, "Shady paths: Leveraging surfing crowds to detect malicious web pages," in *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*. ACM, 2013, pp. 133–144.

[42] H. Mekky, R. Torres, Z.-L. Zhang, S. Saha, and A. Nucci, "Detecting malicious http redirections using trees of user browsing activity," in *IEEE INFOCOM 2014-IEEE Conference on Computer Communications*. IEEE, 2014, pp. 1159–1167.