

Characterizing the Global Mobile App Developers: A Large-scale Empirical Study

Haoyu Wang¹, Xupu Wang², Yao Guo²

¹ Beijing University of Posts and Telecommunications ² MOE Key Lab of HCST, Peking University

Abstract—The rapid growth of the mobile app ecosystem has attracted a great number of mobile app developers. However, few previous studies have analyzed app developers comprehensively at a large scale, and little is known of the characteristics of mobile app developers. In this work, we take the first step to understand global app developers from the view of mobile app life-cycle. We have created a repository of over 1.03 million Android app developers distributed at Google Play and 16 popular alternative markets, along with over 6.2 million Android apps (including apks and metadata) they created. We then explored various characteristics of these app developers from different angles including *developing behaviors*, *releasing behaviors*, *app maintenance behaviors* and *malicious behaviors*. The experimental results have revealed various interesting findings, as well as insights for future research directions. Our efforts can contribute to different stakeholders of the mobile app ecosystem.

I. INTRODUCTION

Mobile apps have seen widespread adoption in recent years. There were more than 2.5 million Android apps on Google Play by January 2019 [1], and the app economy was estimated to achieve more than 6.3 trillion US dollars by 2021 according to a recent report from App Annie [2]. The great success of the mobile app ecosystem has attracted a large number of app developers. It is reported that there are over 12 million mobile app developers worldwide and almost half of them focus on the Android app ecosystem [3].

The great potential of the mobile app ecosystem has attracted a large number of researchers in recent years. One line of research studies focuses on *app store mining* [4], [5]. Typically, after crawling a large amount of data (including metadata and apks) from app markets first, they then measure the ecosystem from different angles including app popularity analysis [6], [7], user comment analysis [8], [9], privacy policy violation [10], [11], market evolution and comparison [12]–[15], and developer misbehaviors (e.g., app promotion, spam apps and malware) [16], [17], etc. The other line of work is focused on mobile app analysis, mainly relying on program analysis techniques, including security and privacy analysis [18]–[20], mobile app testing [21], [22], app repackaging detection [23], [24], third-party library analysis [25], [26], etc.

App developers are the keystones in the mobile ecosystem, ranging from individual developers (both professional and amateur), mobile app development companies (e.g., Hedgehog Lab and Algoworks) [27], to large-scale IT companies (e.g., Google and Facebook). App developers take part in the whole *life cycle* of mobile apps. They create apps with numerous features (**app development phase**), upload them to Google Play and third-party markets (**app release phase**), maintain their apps to

fix bugs or introduce new functionalities (**app maintenance phase**), and promote their apps (via advertisement, app store optimization and pay-per-install services) in order to attract mobile users and boost their profit. App developers arguably play the most important roles in the mobile app ecosystem, and almost all the issues (e.g., security, privacy and vulnerabilities) can be traced back to developers (either benign or malicious).

However, app developers have not received extensive attention from our research community. Only limited research studies [28]–[31] were focused on app developers. For example, Wang *et al.* [28] have studied the distribution of app developers on Google Play and proposed to analyze spam developers. Balenako *et al.* [29] have studied how app developers make security and privacy decisions. Wang *et al.* [32] have studied the evolution of developers in the mobile app ecosystem.

To the best of our knowledge, *no previous work has characterized the behaviors of app developers comprehensively at a large scale*. Our research community knows little on the diverse behaviors of mobile app developers, for example, their *development behaviors* (e.g., behavior on code reuse), *release behaviors* (e.g., releasing apps to different markets), *maintenance behaviors* (e.g., updating apps actively or never updating them), and *malicious behaviors* (e.g., creating malicious apps). Specially, when considering millions of developers distributed in the official and many third-party markets (globally or region-based), it is unknown to our community the characteristics of these (millions of) global mobile app developers.

This paper. We take the first step in this work to understand the diverse behaviors of global app developers across multiple markets at a large scale. We have crawled more than 6.2 million apps from 17 Android app markets (including Google Play and 16 popular Chinese app markets), and created an index of over 1.03 million Android app developers based on developers’ unique signatures (cf. **Section II**). To explore different characteristics of app developers, we seek to answer the following main research questions:

- **RQ1: What is the distribution of developers across app markets? (Section III).** Although previous work [28] has investigated the distribution of app developers in Google Play, considering that there are potentially hundreds of third-party markets beyond Google Play (especially in regions where Google Play is blocked), how the app developers are distributed across the *global app ecosystem* is still unknown. It is interesting to analyze the differences between the official and alternative app markets.

- **RQ2: What are the development behaviors of massive app developers? (Section IV).** The development behaviors of app developers have not been well explored. As developers may release more than one app, it is interesting to see whether they reuse code when developing apps and analyze the differences between popular developers and unpopular ones.
- **RQ3: What are the release behaviors of millions of app developers? (Section V).** As there are hundreds of app release channels around the world, it will be interesting to investigate whether app developers have different release strategies and whether/how they release apps to multiple markets.
- **RQ4: What are the maintenance behaviors of app developers? (Section VI)** For the millions of app developers, how many of them are active developers that may update their apps timely? Are the maintenance behaviors related to app/developer popularity?
- **RQ5: Developer misbehaviors (Section VII).** Since previous studies mostly focus on analyzing malware, how many developers have displayed misbehaviors such as distributing malicious apps? What is the distribution of malicious developers across multiple markets?

Among a number of interesting findings, the following are the most prominent:

- The distribution of app developers is very diverse across different markets, as over 50% of developers in our dataset do not release apps to Google Play. The download patterns of mobile app developers follow the typical *Pareto Effect*, as the top 1% of developers account for over 83% (Google Play) to 92% (alternative markets) of total app downloads.
- Although over 80% of developers have only created one app, we found that many developers have released thousands of apps (e.g., the most aggressive one released over 41K apps). Our investigation has revealed that this seemingly unexpected phenomenon is actually mainly introduced by app generators and publicly known signing keys (i.e., multiple developers using the same signature).
- Code reuse is a common practice among many developers. In general, the more apps one developer created, the higher percentage of code has been reused. We also observed that most developers do not target the most recent API levels (target SDK version), while they may try to accommodate more low-end users (minimum SDK version).
- Roughly 50% of the developers only release apps to one single market. These single-market developers are mainly vendor-specific developers, incentive-driven developers, developers with language barriers and amateur developers with little experience. Roughly 10% of the developers release channel-specific apps to different markets.
- We observed poor app maintenance practices for most of the developers. In general, more than 80% of developers were not active since January 2017, which is 7 months before our crawling process. This finding indicates that most apps in the markets are not maintained timely.

- We analyzed the presence of malware and malicious developers across markets. Roughly 8% of the developers in our dataset have released at least one malicious app. Some developers are found with the tendency to release malicious apps, i.e., 0.08% of the developers in our dataset are responsible for more than 50% of all the (potential) malware, while 0.8% of the developers are responsible for over 70% of all the malware.

To the best of our knowledge, our work is the first large-scale study focusing on app developers across different markets. We believe that our research efforts can positively contribute to bring user and developer awareness, boost the focus of the research community and regulators, and promote best operational practices across app store operators.

II. DATA COLLECTION

To the best of knowledge, no existing studies have released datasets regarding app developers¹. Thus, We first make efforts to implement different crawlers to harvest mobile apps from Google Play and 16 alternative Android app stores in August 2017, as listed in Table I. We seek to understand the global developer distribution and compare it with Google Play, as Google Play is restricted in several countries (e.g., China) and Android users have to resort to alternative app markets for searching and installing apps.

A. Crawler Design

1) *App Markets*: Besides the official Android app market, Google Play, We cover the app stores for the top five smartphone vendors in China, three well-known Chinese web companies, and eight specialized Android app markets. For each app, besides the Android apk files, we also collected publicly available information as provided by the app markets, including the (1) app name, (2) app category, (3) app description, (4) app version name, (5) app release (update) note, (6) developer name, (7) the number of aggregated app downloads, and (8) user ratings.

2) *Crawling Strategy*: We used different strategies to crawl each market. In the case of Google Play, we use a list of 1.5 million package names provided by the PrivacyGrade project [34] as the *searching seeds*, and then use a breadth-first-search (BFS) approach to crawl (1) additional related apps recommended for each app of our seeds by Google Play and (2) other apps released by the same developer. Note that Google Play could be accessed in different world regions, we have instrumented our crawler to support both English and Chinese languages. For crawling the alternative app markets, we use two different strategies to harvest as many apps as possible: an *enumerating-based approach* and a *BFS-based approach*. These two strategies are used simultaneously to complement each other in each app store. For instance, some markets index apps by using incremental integers. This is the case of Baidu's app market, which indexes apps with the following

¹Although several work [16], [33] shared large-scale mobile app dataset, most of them have incomplete information on app developers and app metadata (e.g., app description and app installs).

TABLE I
DISTRIBUTION OF APPS AND DEVELOPERS IN OUR DATASET.

App Market	Type	#Apps	#Developers	Unique Developer
Google Play	Official	2,031,946	538,283	307,047
Tencent Myapp	Web Company	636,265	294,950	31,283
Baidu Market	Web Company	227,454	107,698	16,267
360 Market	Web Company	163,121	90,226	6,134
Huawei Market	Vendor	51,303	32,927	1,865
Xiaomi Market	Vendor	91,190	55,669	3,216
OPPO Market	Vendor	426,419	209,197	30,054
MeiZu Market	Vendor	80,573	50,451	295
Lenovo MM	Vendor	37,716	24,565	195
Wandoujia	Specialized	554,138	291,114	2,819
HiApk	Specialized	246,023	115,191	4,204
AnZhi Market	Specialized	223,043	74,145	16,262
25PP	Specialized	1,013,208	470,073	89,594
Sougou	Specialized	128,403	66,759	2,700
LIQU	Specialized	179,147	101,336	6,180
App China	Specialized	42,435	23,699	764
PC Online	Specialized	134,863	65,225	1,680
Total	-	6,267,247	1,035,992	

syntax: <http://shouji.baidu.com/software/INTEGER.html>. This feature eases notably our APK collection process by using an enumerating approach.

Parallel Search. Note that we use a “parallel search” optimization in our crawler. As long as we have identified a new app (package name) in either of the 17 markets, we will search this app in all the remaining markets immediately to make sure that we could crawl this app/developer simultaneously.

Crawling Process. The crawlers have been launch in parallel via 50 Aliyun Cloud Servers [35] (with 8 crawler processes running in parallel for each server) during August 2017. This allows us to obtain a temporal screen-shot of the mobile app ecosystem at a given time.

B. Dataset

Our crawled dataset is shown in Table I. We have crawled 6.2 million app items in total, which were created by more than 1 million app developers (signatures). Note that we use developer signatures as the identifier to count developers, rather than using the *App Creator* names listed on app markets as previous work did [28], mainly due to two reasons. On one hand, the creator names shown on each market varies (e.g., different languages, abbreviate Vs. full name, etc), it is hard to figure out whether two developer names are identical, while one developer signature may always correspond to several creator names. On the other hand, previous work [12] and reports [36] suggested that fake apps and fake developers usually use the same or similar names to cheat app users, as some markets do not have strict regulations on app metadata, while the leakage of the developers’ private keys (signatures) are not a common case in practices.

We believe our dataset has covered a large portion of Android developers in the world. As one developer signature may corresponds to more than one “real developer” (especially for mobile app development companies), it is reasonable to estimate that our dataset contains several millions of Android developers. As a recent report [3] suggested that there are roughly 6 million Android developers around the world, we believe our dataset is representative and large enough to investigate developers’ diverse behaviors.

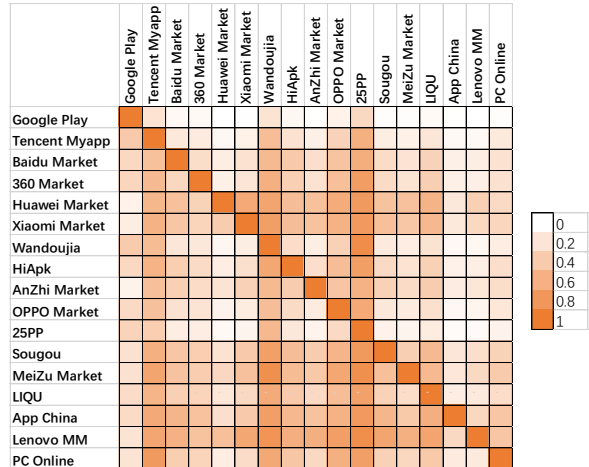


Fig. 1. The overlapped app developers across the 17 app markets. For each cell (row X, column Y), the color represents the percentage of developers in market X that also published apps in market Y.

General overview of our dataset. It is interesting to see that, although Google Play holds the largest number of app developers (over 538K), roughly 50% of app developers in our dataset do not release apps on Google Play. This figure suggests that a large portion of the mobile app ecosystem (over 50% of app developers, and the apps they created) has been overlooked by our community, as most previous studies only focus on Google Play. On average, a developer releases 1.54 (Lenovo MM market) to 3.77 apps (Google Play) to different markets, which suggests that our dataset is diverse enough to study the behaviors of app developers.

III. APP DEVELOPER STATISTICS

We now study high-level characteristics of app developers. In this Section, we seek to answer the first research question. We first analyze the distribution of app developers on Google Play and the 16 alternative app stores, then we analyze the developer popularity based on the app downloads, and we further define popular developers, which will be used for further comparison in the following Sections.

A. Developer Distribution

As shown in Table I, more than 50% of app developers release their apps to Google Play, and more than 70% of developers release apps to Chinese alternative app markets.

Overlapped Developers. We analyzed the overlapped developers across the 17 markets, as shown in Figure 1. It is obvious that Chinese markets share more similarity. For example, more than 80% of developers on Wandoujia, Meizu and Lenovo MM markets also release apps to 25PP market. The 25PP market hosts the most number of developers among Chinese third-party markets, however, it only covered roughly 30% of the developers in Google Play. Several markets have shown great difference with Google Play when considering app developers. For example, less than 1% of the developers in Google Play are active on Huawei and Xiaomi markets, and roughly 10% of app developers in Huawei and Xiaomi markets release apps

	0-10	10-100	100-1K	1K-10K	10K-100K	100K-1M	>1M
Google Play	0.35%	4.78%	21.01%	29.56%	24.88%	13.85%	5.55%
Tencent Myapp	56.08%	11.89%	14.15%	9.91%	5.32%	1.88%	0.78%
Baidu Market	0.00%	31.93%	23.72%	20.00%	13.04%	7.27%	4.05%
360 Market	19.95%	17.88%	16.87%	18.07%	15.21%	8.58%	3.44%
Huawei Market	0.08%	0.00%	34.70%	29.06%	17.99%	12.09%	6.07%
Wandoujia	1.77%	4.47%	39.89%	33.58%	16.86%	2.71%	0.72%
HiApk	0.00%	0.00%	0.00%	88.61%	7.56%	2.77%	1.06%
AnZhi Market	0.07%	0.00%	21.03%	60.94%	13.73%	3.16%	1.08%
OPPO Market	0.00%	0.00%	66.78%	22.48%	7.29%	2.45%	0.99%
25PP	0.23%	4.22%	61.10%	25.28%	6.10%	2.32%	0.75%
Sougou	0.85%	16.29%	53.09%	20.95%	6.47%	1.77%	0.58%
Meizu Market	6.08%	12.62%	43.69%	21.12%	9.17%	4.98%	2.35%
LIQU	0.01%	0.02%	0.01%	42.13%	37.88%	15.83%	4.12%
Lenovo MM	0.04%	0.00%	14.78%	40.27%	29.24%	11.10%	4.57%
PC Online	8.97%	73.11%	11.78%	4.23%	1.47%	0.38%	0.05%
TOTAL	3.72%	6.24%	27.56%	32.51%	19.29%	7.91%	2.78%

Fig. 2. The distribution of app downloads per developer. to Google Play conversely. This result further indicates the diverse app ecosystem across the global app markets.

Unique Developers (Single-market Developers). We further investigate the distribution of unique developers. We define “unique developers” as the ones who only release apps to one market². Google Play hosts the most number of unique developers, roughly 60% of developers on Google Play are unique. The percentage of unique developers in Chinese markets vary from 0.5% (Meizu market) to 21% (Anzhi market). In general, to gain more app downloads and attack more users, a developer may want to release their apps to as many markets as possible. We will further analyze the unique developers and investigate the underlying reasons in Section V.

B. Developer Popularity

We calculate the aggregated app downloads at the per-developer level. Then we analyze whether the popularity of app developers follows the Pareto Effect.

1) *Distribution of App Downloads Per Developer:* For each market, we first aggregate the app downloads for all the apps released by each developer (cf. Figure 2). The average number of downloads per developer varies greatly across markets, from over 3K (PC Online) to over 2.5 million (Huawei Market).

Considering all the markets, we further aggregate the downloads for each developer signature (cf. the last row of Figure 2). In general, 90% of the developers have aggregate app installs less than 100K. Subtle differences arise when looking at the details on a per-market basis. For example, the app downloads of developers in Google Play mainly fall into the range of 100 to 1 Million. Huawei Market follows the similar pattern with Google Play. However, over 56% of app developers in the Tencent market have only fewer than 10 app downloads in total. Over 80% of developers in the PC Online market have achieved fewer than 100 app installs. It suggests that **the popularity of app developers (and apps) are significant different across markets**, as we will investigate later.

2) *Pareto Effect:* Previous studies have shown that web content downloads and mobile app downloads [28] usually follow the Pareto principle [37]: 20% of the objects are

²Note that it is possible that the “unique developers” would appear in other markets that are not covered in this paper. This, however, does not affect our comparative study of the 17 markets and the investigation of app developers.

responsible for 80% of the downloads. Thus we further analyze whether the Pareto effect exists when analyzing app developers, especially when comparing Google Play with Chinese third-party markets. As shown in Figure 3(1), the result clearly suggests that **the download pattern of mobile app developers follows a typical Pareto Effect, as a few developers account for most of the downloads**. Top 1% of developers in Google Play account for 83.87% of total app downloads, while the number in Chinese markets is 92.43%.

C. Popular Developers

As the popularity of app developers follows Pareto effect, we regard the top 1% developers for each market as *popular developers*, who have occupied over 80% of app downloads in total. We have identified 8,963 popular developers, more than 60% of them (5,382) have released apps to Google Play, and over 65% of them (5,902) have released apps to third-party app markets. In the following studies, we will characterize the differences between popular developers and non-popular developers on various behaviors.

IV. DEVELOPMENT BEHAVIORS

In this Section, we seek to answer the second research question, i.e., understanding the development behaviors of diverse developers. We first investigate the number of apps each developer created, and then classify developers into different categories. We further analyze the development behaviors from two aspects: (1) to what extent they reuse code when developing apps, and (2) the SDK versions (API levels) they focused on.

A. The Number of Created Apps

Figure 3(2) shows the distribution of the number of created apps per developer. Note that we count the unique package names (app IDs) released by each developer. As one developer may release multiple app versions to different markets, we will investigate it further in Section V.

It is obvious that, over 80% of developers only created one app. More than 94% of the developers created fewer than 5 apps. However, **many of them (1,561) have released more than a hundred apps, while 4 developers have released even over 20 thousand apps**. Table II shows the top 10 developers who created the most number of apps. It is surprising to see that the top developer has created 41,287 apps. In general, a normal developer would not release so many apps. Therefore, we further analyzed these developers in detail.

1) *Analyzing the frequent app developers:* We first analyze the creator names shown on app stores in order to know who they are. To our surprise, we found that all of the top 10 developers have multiple creator names, and **most of them have thousands of different names**, which means that **these developers use the same signature to create apps and then release apps to markets under different creator names**. Thus we further analyzed the CN (Common Name) and OU (Organization Unit) fields in the developers’ signatures and searched them with Google. We found that 6 of them are *app generation tools*, including Qbiki Networks [38], Andromo

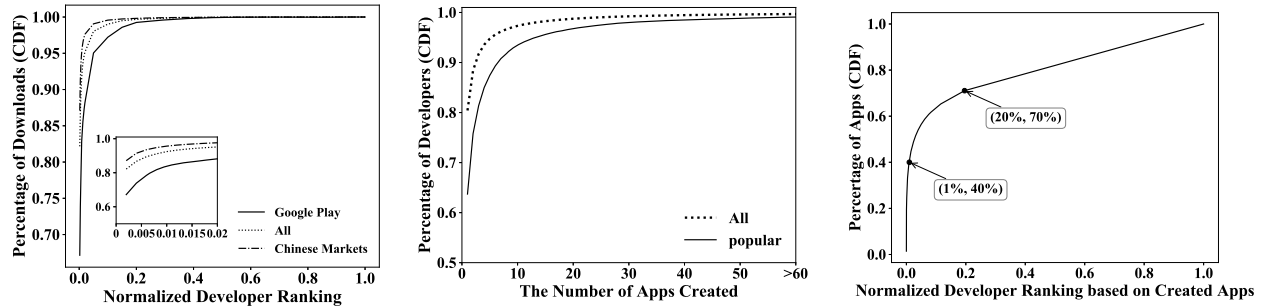


Fig. 3. The distribution of app downloads per developer and the number of created apps per developer .

TABLE II

TOP 10 DEVELOPERS RANKED BY THE NUMBER OF RELEASED APPS.

Developer (signature)	# Created Apps	# Creator Names	# Market	# Avg Installs (Median)	Who is the developer and Why
5de9d45fed20f9db2033e5803704d156	41,287	3,263	16	3,825 (495)	Qbiki Network (App generator)
e89b158e4bcf988ebd09eb83f5378e87	34,964	9,422	17	107,004 (1,000)	Android Release Key
f478b46338b428770da4137545f40d74	30,998	3,060	17	9,720 (933)	Andromo App (App generator)
e03db148616053a4c7bd317b09a28700	27,196	3,046	16	541 (100)	Bizness Apps (App generator)
5dc8201f7db1ba4b9c8fc44146c5bcc2	19,789	182	15	2,394 (1,000)	Theme Apps (moxiu.com)
5b120e96b20f5b4ec695d79b20d18753	15,027	277	17	41,993 (500)	Theme Apps (Panda Theme)
c228fd74f0175aa72c871167407cac92	14,275	7,897	15	1,249 (100)	Conduit (App generator)
8a1a4b3824b35dc964cfa7565904b49c	10,464	3,783	16	2,980 (233)	BestToolbars (App generator)
c385b82bdf173cd94e2893283ac6099c	8,887	720	16	2,782 (1,000)	Dzs.mobi (App generator)
c584e3992c76706541a59975caabd127	8,858	2,895	16	2,525 (100)	DigiCert

App [39], Bizness Apps [40], Conduit [41], BestToolbars [42] and Dzs.mobi, which provide app frameworks for users without developing knowledge to create apps. The second largest signature is *Android Release Key*, which is a publicly known key that embedded in the Android framework. Apps signed with publicly known keys will lead to great security issues, i.e., allowing attackers to modify the apps without developers' knowledge. Two signatures are related to *theme apps* (Panda theme and Moxiu theme), and the developers created thousands of theme apps with identical code while only replacing the resource files. Besides, we also found a signature that is signed by DigiCert, a leading SSL certificate authority, which we were unable to know who created the apps.

To further understand the apps they created, we use Wukong [24] and Boreas [43] to compare the code-level similarity of the apps per-developer released. It is interesting to see that, for the top 10 developers, all of the apps they created have almost the same code (similarity higher than 95%). Furthermore, these developers release apps to almost all the markets we studied in this paper, but they generally do not have much app downloads on average (cf. Column 5). As Google Play does not allow developers release apps with repetitive contents [44] (copying content from other apps or creating multiple apps with highly similar content), most of the apps should actually be removed, especially for those apps with little or no downloads.

2) *Developer Classification*: We further classify developers into different groups based on the number of apps they created. In this paper, we followed the classification criteria provided in Wang *et al.* [28]. They have categorized the developers into *aggressive developers* (created more than 50 apps), *active*

	Conservative (1 app)	Moderate (2-9 apps)	Active (10-49 apps)	Aggressive (>= 50 apps)
Google Play	0.73	0.22	0.04	0.01
Tencent	0.70	0.24	0.05	0.01
Baidu	0.62	0.27	0.09	0.02
360	0.58	0.30	0.10	0.02
Huawei	0.69	0.25	0.04	0.01
Xiaomi	0.71	0.23	0.05	0.01
Wandoujia	0.69	0.25	0.05	0.01
HiAPK	0.64	0.27	0.08	0.02
AnZhi	0.73	0.21	0.05	0.01
OPPO	0.69	0.24	0.06	0.01
25PP	0.75	0.20	0.04	0.01
SOUYOU	0.63	0.27	0.08	0.02
Meizu	0.62	0.27	0.08	0.03
LIQU	0.63	0.27	0.07	0.02
App China	0.52	0.32	0.12	0.04
Lenovo MM	0.61	0.29	0.07	0.03
PC Online	0.60	0.28	0.10	0.03

Fig. 4. The distribution of different developer groups across markets.

developers (created 10 to 49 apps), *moderate developers* (created 2 to 9 apps) and *conservative developers* (created only one app). Based on this taxonomy, we further investigate the distribution of different developer groups on various markets.

As shown in Figure 4, the distribution of different developer groups follows the similar patterns across markets. Roughly 60% to 70% of developers are conservative developers that only released only 1 app, while moderate developers who created 2 to 9 apps have occupied 20% to 30% of them. Active developers account for roughly 5% to 10% of them, while only 1% to 4% of developers across markets belong to aggressive developers that created over 50 apps.

B. Code Reuse

Although a large number of previous studies have investigated the code reuse behaviors across a large number of apps

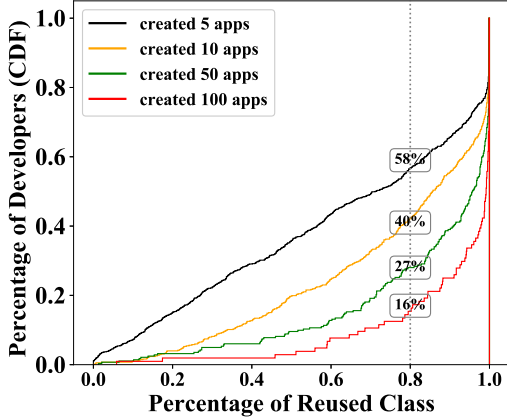


Fig. 5. CDF of developers vs. the percentage of reused code (class-level).

(e.g., repackaged/similar apps [23], [24], [45], and third-party libraries [25], [26], [46], [47]), we still do not know the code reuse behaviors within the same developer in large scale.

Developer Sampling. We sample developers who created 5 apps (moderate developers, with 12,267 developers in total), 10 apps (active developers, with 3,185 developers in total), 50 apps (aggressive developers, with 75 developers in total) and 100 apps (aggressive developers, with 1,561 developers in total), and analyze the proportion of reused code within all the apps each developer created. We seek to explore and compare the code reuse behaviors across different developer groups.

Method. We use a simple but effective approach to conduct this comparison. We first decompiled each app and parsed each decompiled Smali file into an AST (Abstract Syntax Tree). Then, we extracted features from ASTs including the number of different kinds of nodes in the AST, and further represented each Smali (class) as a feature vector. Here, we perform class-level code reuse analysis. For each developer, we extracted the features for each class they created and then enforce strict comparison to analyze the percentage of reused code classes.

The result is shown in Figure 5. In general, code reuse is a common behavior across different developer levels. It is a trend that the more apps one developer created, the higher percentage of code he will reuse. For the developers who created 5 apps, over 40% of them have more than 80% of the code in their apps reused more than once. For the developers who created 100 apps, more than 84% of them have reused at least 80% of the code in their apps. Even over 40% of the developers who created 100 apps have all of the classes in their apps being reused at least once, which further indicates our observation that some developers use app generation tools to create a large number of similar apps.

C. Target/Minimum API Level

Android app developers can declare in the app manifest the **Target SDK Version** and the **Minimum SDK Version** supported by their apps. Configuring the apps to target a recent API level could ensure that app users can benefit the improvements introduced in the new system versions (e.g., security and performance improvements), while still allowing

it to run on older Android versions for compatibility reasons. It is reported [48] that Google Play requires that new apps target at least Android 8.0 (API level 26) from August 2018, and that app updates target Android 8.0 from November 2018. The declared Minimum SDK Versions could offer insights about whether app developers are trying to maximize app customers, or whether they try to target top-end users.

For each developer, we first extract the declared target/minimum API levels from Android Manifest files for all the released apps, and then we could get the *primary target/minimum API levels* for every developer, i.e., the API levels that occupy the largest proportion. Figure 6 shows the distribution of primary target/minimum API levels for all the 1 million developers.

Although the newest Android API level is 26 by the time of our dataset crawling, almost 67% of developers mainly target API levels prior to 19. Over 16% of developers have primary target API level of 19, and over 14% of developers have primary target API level of 17. For the minimum API levels, over 31% of them primarily focus on API level 8. This result suggested that **most developers do not target a recent API level**, which means that the apps they created cannot benefit the improvements introduced in the new API levels. At the same time, **most developers primarily declared minimum SDK versions prior to API level 10**, which means that they try to accommodate more low-end users, as many Android devices have embedded out-dated system versions.

V. RELEASE STRATEGY

In this Section, we seek to answer the third research question, i.e., exploring the release behaviors of app developers. We first analyze the number of target markets for each developer. Then we classify them into “single-market release developer” and “multi-market release developer”, and further compare their characteristics from various aspects. At last, we analyze the primary market for each developer to understand whether each market has its own target developers and target user groups.

A. Target Markets

Figure 7 (1) shows the distribution of the number of target markets for each developer. In general, roughly 50% of developers (520,559) only release apps to one market, while the percentage decreases to 5% for popular developers. Over 19% of developers and 81% of popular developers release apps to more than 5 markets. 696 developers in total release their apps to all the market, while 62% of them are popular developers.

B. Single-market Developers vs. Multi-market Developers

We then compare the single-market release developers with multi-market release developers on app downloads and the number of created apps.

App Downloads. As shown in Table III, over half of the multi-market developers could get more than 10K app installs, while the percentage of single-market developers is roughly 10%. 8,542 out of 8,943 popular developers are multi-market developers, while only 421 of them are single-market developers. We further compare the popularity of single-market

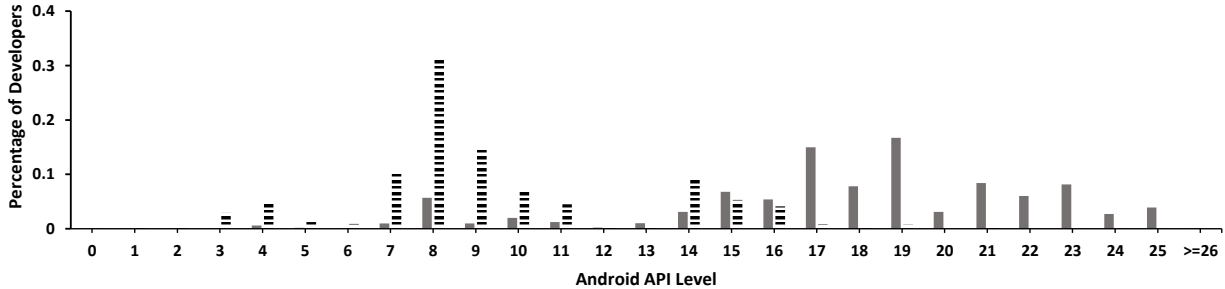


Fig. 6. The distribution of main target/minimum API level per developer.

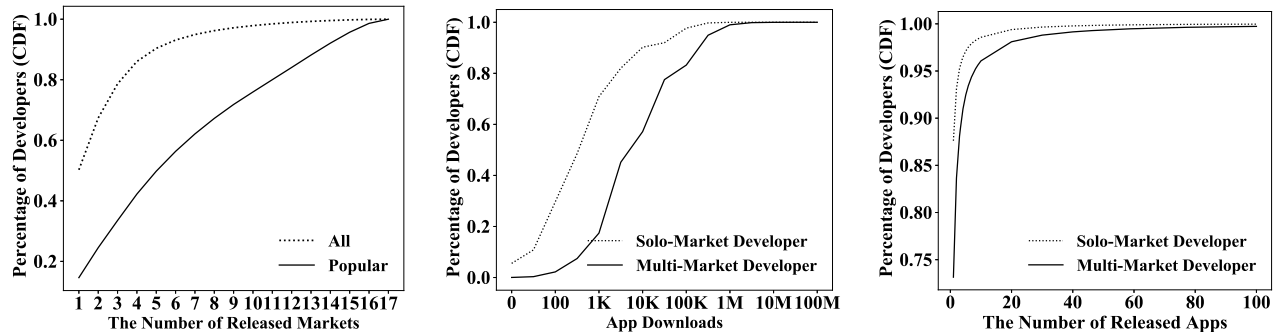


Fig. 7. The Release behavior of app developers.

TABLE III
CATEGORIZATION OF APP DEVELOPERS BASED ON THE NUMBER OF RELEASED MARKETS.

type	developers #number	total #Downloads	avg #Downloads
single market developer	520,559	13,856,386,024	16,381
multi market developer	514,737	406,367,252,890	96,372
all market developer	696	288,637,454,403	855,942

developers and multi-market developers (we further split “all-market” developers from it) in Table III. It is interesting to see that, the app downloads of multi-market developers (besides all market developers) could achieve six times as much as those of single-market developers, while the app downloads of all market developers could be roughly 9 folds of that number of multi-market developers, and 50 folds of the downloads of single-market developers. It further indicates that, in general, the more markets a developer releases to, the more app installs one developer could get.

The number of created apps. We then compared the number of created apps for single-market and multi-market developers, as shown in Figure 7 (3). In general, the multi-market developers create more apps than single-market developers. More than 87% of single-market developers only created one app, while the percentage for multi-market developers is 73%. Single-market developers create 1.6 apps on average, while the number for multi-market developers is 3.9.

As our empirical study suggested that multi-market developers are more popular than single-market developers, it is strange that **over half of the developers choose to be single-market developers**. Thus, we further investigate the reasons.

Characterizing the single-market developers. By analyzing developers’ meta information we crawled from the markets

and the apps they released, we further categorize the single-market developers into three categories. First, some single-market developers are **vendor-specific developers**, who are very popular in our dataset. They only release apps to the vendor specific markets. For example, the developer signature “7bf32cb332a1789f4744b18d38703c79” belongs to Huawei company, and it only releases apps to the Huawei market (e.g., com.huawei.smarthome), as the apps they designed are explicitly developed for Huawei handsets. Second, most of the app markets we studied in the work have **incentive mechanisms** for encouraging app developers to publish their apps. For example, most of the markets encourage developers to publish exclusive apps in their own markets in order to target at potential market users, in exchange, they promise to actively taking measures to promote the apps (e.g., recommending the apps on the start page to make them more visible to users). As a result, some developers are signed by app markets to publish exclusive apps. Third, we found that a number of the single-market developers in Google Play may have **language-barriers** (based on the regions specified in these developers’ signatures), which stops them from publishing apps in markets beyond Google Play. Furthermore, we believe the remaining single-market developers are mostly **amateur developers that have no experiences in releasing apps**, as most of these developers have almost no downloads. We can assume that they do not have much time and efforts to maintain apps across multiple markets.

C. Primary Market for Each Developer

We further seek to investigate the primary market for each multi-market developer. We define one’s **primary market** as

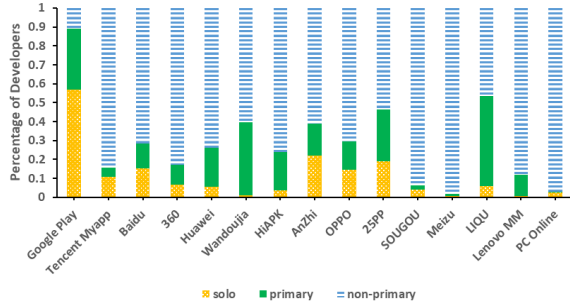


Fig. 8. The distribution of primary-market developers.

TABLE IV

AN EXAMPLE SHOWING THE APP “COM.KUGOU.ANDROID” (WITH IDENTICAL SIGNATURE “FE4A24D80FCF253A00676A808F62C2C6”).

Market	Version	APK MD5	kgchannel
360 Market	8798 (8.7.0)	0ff50f0874567e0715c9b98e85b5b715	kgchannel_15
Wandoujia	8798 (8.7.0)	e39c9f8b6d76de9152a78bcc1a6fe692	kgchannel_5
Z5PP	8798 (8.7.0)	6eaa922cb97d559812b2b45d15247870	kgchannel_62
Lenovo	8798 (8.7.0)	d39af2f9733efdbb18e9128c38a1331	kgchannel_39
Tencent	8798 (8.7.0)	b2c3a00f2e5763cf9a0bdf220e5760a9	kgchannel_14

the market that the developer gained the most number of app downloads from. As a result, for developers in each market, we could further classify them into *solo-market developers*, *primary market developers* and *non-primary market developers*.

Figure 8 shows the comparison result across markets. It is interesting to see that, over half of the developers in Google Play are single-market developers, and Google Play is also the primary market for over 30% of app developers hosted on it. For alternative markets, the percentage of primary-market developers are relative high in markets including LIQU (47.4%), Wandoujia (38.7%), and Z5PP (27.5%). As a contrast, PC Online and Meizu markets hosted the least percentages of primary-market developers, with 0.4% and 1.4% respectively.

This result shows the diversity of developer popularity across markets, which also indicates that the mobile app users of different markets may have diverse characteristics. That is to say, **each market has its own target developers and target user groups**. This result could offer insights to developers when choosing the desired markets to release their apps.

D. Releasing Channel Specific Apps

Our investigation further discloses that many developers release plausibly “different” apps (with same app version) to multiple markets. For example, there are 14 different “com.kugou.android” apks in our dataset, although they have the identical package names, version code and developer signature. Table 5 lists five of them. By manually comparing the decompiled code, we find that the DEX files (i.e., main function code) are identical for all of them, while only the META-INF/kgchannel file is different. The suffix of the file name is labeled with a different number, which indicates the market that it released though. **This finding somehow demonstrates that “different” apps do appear in the Android ecosystem as developers may leverage this means to distinguish the source of their app users**, i.e., from which market the app is installed. Moreover, our manual investigation also reveals another possibility where “different” apps exist due to some

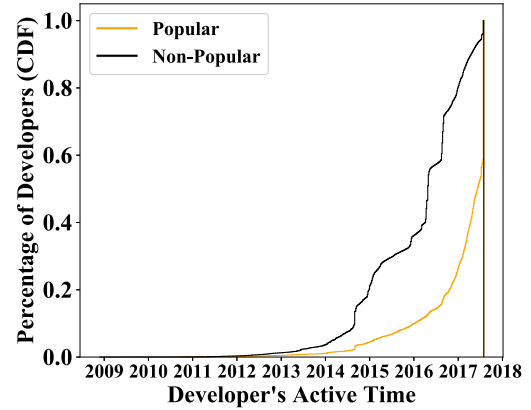


Fig. 9. The distribution of developer active time.

certain market regulations. For instance, 360 market requires Android developers to pack and obfuscate their apps with 360 Jiagubao (a packing tool) before uploading. It is hence reasonable that apps with the same package name, version code and signature could be “different” in terms of hash value.

We group the apks in our dataset based on their package names, version codes and developer signatures. At last, we have identified 113,003 developers (10.9%) who have released channel specific apps to different markets.

VI. MAINTENANCE BEHAVIORS

In this Section, we seek to explore the app maintenance behaviors and answer the fourth research question. Previous work suggested that some popular developers usually update their apps every one week or two [49], [50], it is interesting to study how often do app developers update their apps in large-scale, especially when considering that roughly 50% of them are multiple-market developers. However, as our dataset only covers one snapshot of the app ecosystem across 17 markets, it is not feasible for us to study the app updating behaviors across different developers. As a result, we resort to an alternative way to investigate the maintenance behaviors.

For each developer, we first calculate the “active time”, the latest time that he releases/updates the apps in our crawled dataset across markets. Then we analyze the distribution of active time, as shown in Figure 10. It is surprising to see that, in general, more than 30% of developers were only active before 2016, and more than 80% of developers were not active since January 2017 (7 months before our crawling process). **This result suggested the poor app maintenance practices for most of the developers in our dataset.**

The popular developers and the non-popular ones have great diversity. For the popular developers, over 90% of them are still active after 2016, while roughly 40% of them are active by the month of our crawling (August 2017).

We further analyze the correlations of developers’ active time with the number of downloads, the number of released markets, and the number of created apps separately, as shown in Figure 10. Most of the popular developers (with high downloads) are active recently, although the distribution of

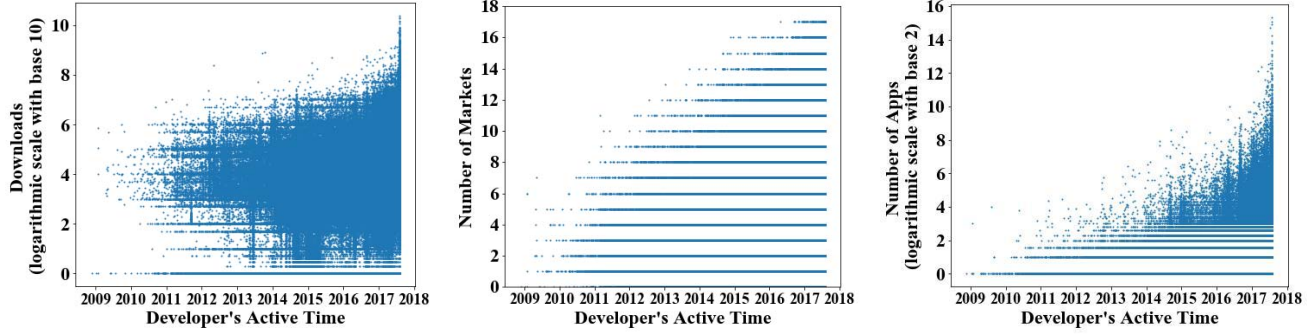


Fig. 10. Developer's active time vs. (1) the number of downloads, (2) the number of released markets, and (3) the number of created apps.

different developer downloads are scattered across the timeline (cf. Figure 10 (1)). Developers that release apps to more markets are more active than those who release apps to fewer markets (cf. Figure 10 (2)). Developers who created large number of apps are likely to be active than those who only release fewer apps (cf. Figure 10 (3)).

VII. MALICIOUS DEVELOPERS

In this Section, we seek to answer the fifth research question and study the prevalence of malicious developers. Specifically, we resort to VirusTotal³, a widely used online service that aggregates over 60 anti-virus engines, to measure the potential malicious behaviors of mobile apps.

A. The Distribution of Malicious Apps and Developers

We first upload all the apps in our dataset to VirusTotal⁴. Previous studies have suggested that some anti-virus engines may not always report reliable results. In order to deal with such potential false positives, we analyzed the results grouped by how many engines (**AV-score**) flag an app as malware.

We analyzed the distribution of malicious apps and malicious developers (the developers who created at least one malicious app) across markets under different AV-scores, as shown in Figure 11. Previous work suggested that the threshold of 10 engines is a robust choice [51], [52]. Based on the threshold of $AV\text{-score} \geq 10$, around 2% of the apps in Google Play are labeled as malware, while the percentage in Chinese markets is much higher, e.g., for 11 out of the 16 Chinese markets the percentage of malware exceeds 10%. As to malicious developers, roughly 2% of developers in Google Play has released at least one app exceeds the AV-Score threshold, for 14 out of the 16 Chinese markets the percentage of malicious developers exceeds 10%. It is surprising to see that over 20% of developers in App China and PC Online markets have released at least one possible malware. According to the threshold of 10 engines, there are 87,691 malicious developers, accounting for 8.46% of developers in our dataset.

³<https://www.virustotal.com>

⁴Note that the maximum apk size accepted by VirusTotal using API interface is 32MB. As a result, we did not upload apps with larger size than 32MB.

	AV-Score ≥ 10		AV-Score ≥ 20	
	App	Developer	App	Developer
Google Play	2.09%	2.37%	0.32%	0.61%
Tencent	11.16%	11.82%	3.45%	4.32%
Baidu	12.24%	14.09%	3.30%	5.83%
360	12.35%	16.24%	3.10%	6.36%
Huawei	16.43%	9.41%	6.00%	2.28%
Xiaomi	9.12%	13.71%	1.82%	4.38%
Wandoujia	10.70%	10.14%	3.14%	3.67%
HiApk	4.71%	14.43%	0.57%	5.65%
AnZhi	7.53%	18.09%	1.52%	6.48%
OPPO	8.26%	14.80%	2.06%	5.91%
25PP	7.98%	8.75%	2.19%	2.98%
Sougou	11.12%	18.72%	2.72%	8.15%
MeiZu	11.37%	16.75%	2.41%	6.42%
LIQU	13.00%	16.42%	4.27%	6.96%
App China	24.01%	24.43%	8.37%	11.47%
Lenovo MM	16.53%	15.32%	4.59%	5.11%
PC Online	14.13%	27.93%	4.27%	11.43%

Fig. 11. Malicious distributions.

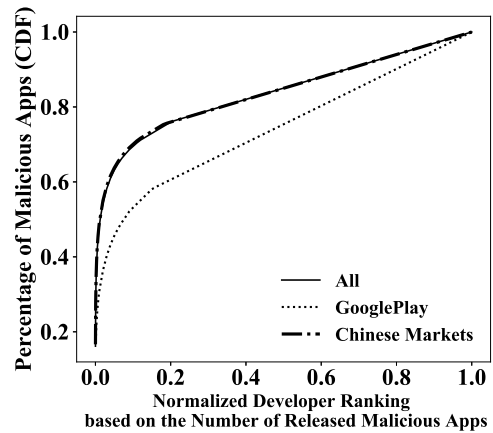


Fig. 12. Developer ranking based on the number of created malware.

B. Top Malicious Developers

We then analyzed the malicious developers who released the most number of malware, as shown in Table V. It is surprising to see that, based on the threshold of “AVscore ≥ 10 ”, some developers released over 10K malicious apps. We further analyzed the proportion of released malicious apps out of all the created apps for each developer, and the top 5 developers are shown in Table V too. Almost all of the apps they released are supposed to be malware. It suggests that **some**

TABLE V

TOP 5 DEVELOPERS RANKED BY THE NUMBER OF CREATED MALICIOUS APPS AND THE PORTION OF CREATED MALICIOUS APPS SEPARATELY.

Top 5 developers ranked by the number of created malicious apps			
Signature	# Apps	# Malware	% Malware
5B120E96B20F5B4EC695D79B20D18753	24,390	11,734	48.1%
E89B158E4BCF988EBD09EB83F5378E87	72,882	11,493	15.8%
5DE9D45FED20F9DB2033E5803704D156	51,408	10,527	20.5%
43D6BFCDD0EEB50F103E20DF6EDBDCC72	5,013	4,172	83.2%
00469D701B96B9FCD9F6AA4CIA8044FB	7,150	2,760	38.6%
Top 5 developers ranked by the percentage of created malicious apps			
1E987918B5365AF83A8A3E654AE0406C	102	102	100%
5042474ED55FEBF96155906FC75A8584	117	116	99.1%
3FED5340FBA5461E0CB414B0A943ADD8	115	114	99.1%
AB53865CDFD609E6F9BFB9C781FB964D	240	237	98.8%
B9CAD7C695ED11B8A93992353C69EEBE	206	203	98.5%

developers have the tendency to release malicious apps.

We further rank developers based on the number of released malicious apps. As shown in Figure 12, **roughly 1% of malicious developers (0.08% of all the developers) created more than 50% of potential malicious apps, roughly 10% of malicious developers (0.8% of all the developers) created over 70% of potential malicious apps.** For the Chinese app markets, the distribution of malicious apps is more concentrated than that of Google Play, i.e., a few developers have created a large amount of malware.

VIII. DISCUSSIONS

A. Limitations

Signature vs. developer. We have tried to distinguish developers using the app signatures, as we assume that each developer will use the same signature to sign their apps and developers' privacy keys will not be leaked, which is a commonsense adopted by the research community. However, this assumption sometimes is not true as we have identified during the study. Some developers may use different signatures to sign the different versions of their apps, while some developers may use public app keys or app generator keys to sign their apps. This obviously *bad practice* may affect our study, however most of the developers are still using a single key for all of their apps and different releasing versions. Furthermore, it is also an interesting topic to investigate further on how developers sign their apps based on our results.

Completeness of the dataset. Although we have tried our best to crawl a complete snapshot of all the available apps in the studied markets, it is obviously impossible to achieve perfect coverage. However, according to the released data from third-party sources (AppBrain [1] and several ASO markets [53], [54]), we believe our dataset contains most of the popular apps and it is by far the largest dataset with the most number of apps and developer information. Nevertheless, our work still faces several limitations when studying the maintenance behaviors, as we only crawled one snapshot.

B. Implication

We believe that our efforts and the revealed insights can contribute to different stakeholders of the mobile app ecosystem. For app developers, the practices we revealed in this paper could help them make better decisions when developing and releasing apps. For app markets, we observed that some spam developers

that release thousands of low-quality and repetitive apps, which may cause disruption to the mobile app ecosystem and should be removed. Besides, we also revealed that malicious developers are widespread across markets and some developers have the tendency to release malicious apps. As a result, it is urgent for market maintainers to identify and further monitor these developers and the apps they released.

IX. RELATED WORK

A. Analyzing App Ecosystem and Mobile Apps

A large number of research efforts have been focused on analyzing the mobile app ecosystem from various dimensions. AndroZoo [33] project focused on crawling a large-scale dataset of APKs, which has enabled a number of studies focusing on malware analysis and app repository mining [55]. PlayDrone [16] is the first work that performed large-scale characterization of 1.1 million apps published in Google Play, with exploring issues such as app evolution and authentication schemes. A few studies focus on analyzing market evolution and comparing app markets [12], [13], [17]. Besides, many studies proposed to analyze/optimize app ecosystem from user comments [8], [9], [56]–[60], privacy policy [10], [11], app ratings [61]–[65], app updates [14], [50], [66], [67], and app descriptions [68]–[70], etc.

B. Understanding Mobile App Developers

A limited number of research studies [28]–[31], [71]–[74] focus on analyzing mobile app developers. For example, Nayebi et al. [71] conducted surveys with users and developers to understand the release strategies adopted by app developers and found that an app's release strategy affects its success. Wang et al. [28] investigated over 320K app developers in Google Play, and analyzed developers' distribution and proposed to identify spam developers and developers with privacy-risks. Dong et al. [75], [76] studied developers' fraudulent behaviors on using mobile advertisement. To the best of our knowledge, no previous work has characterized app developers' behaviors from different phases in app life-cycle. Besides, our work is the first paper that cover over 1 million Android developers from 17 app markets.

X. CONCLUDING REMARKS

In this paper, we take the first step to understand the diverse behaviors of app developers. We first created a dataset of over 1 million app developers and over 6 million app items crawled from 17 app markets including Google Play. Then we characterize app developers from development behaviors, release behaviors, app maintenance behaviors and malicious behaviors. We believe that our research efforts can positively contribute to bring user and developer awareness, attract the focus of the research community and regulators, and promote best operational practices across app store operators.

ACKNOWLEDGEMENT

This work is supported by the National Natural Science Foundation of China (grants No.61702045 and No.61772042). Yao Guo is the corresponding author.

REFERENCES

- [1] “Android and Google Play statistics,” 2019, <https://www.appbrain.com/stats>.
- [2] “App economy to grow to \$6.3 trillion in 2021, user base to nearly double to 6.3 billion,” 2017, <https://techcrunch.com/2017/06/27/app-economy-to-grow-to-6-3-trillion-in-2021-user-base-to-nearly-double/\-to-6-3-billion/>.
- [3] “There are 12 million mobile developers worldwide, and nearly half develop for android first,” 2016, <http://www.businessofapps.com/12-million-mobile-developers-worldwide-nearly-half-develop-android-first/>.
- [4] M. Harman, Y. Jia, and Y. Zhang, “App store mining and analysis: Msr for app stores,” in *Proceedings of the 9th IEEE Working Conference on Mining Software Repositories (MSR '12)*, 2012, pp. 108–111.
- [5] W. Martin, M. Harman, Y. Jia, F. Sarro, and Y. Zhang, “The app sampling problem for app store mining,” in *Proceedings of the 12th Working Conference on Mining Software Repositories (MSR '15)*, 2015, pp. 123–133.
- [6] T. Petsas, A. Papadogiannakis, M. Polychronakis, E. P. Markatos, and T. Karagiannis, “Measurement, modeling, and analysis of the mobile app ecosystem,” *ACM Transactions on Modeling and Performance Evaluation of Computing Systems (TOMPECS)*, vol. 2, no. 2, p. 7, 2017.
- [7] —, “Rise of the planet of the apps: A systematic study of the mobile app ecosystem,” in *Proceedings of the 2013 Internet Measurement Conference (IMC '13)*, 2013, pp. 277–290.
- [8] N. Chen, J. Lin, S. C. Hoi, X. Xiao, and B. Zhang, “Ar-miner: mining informative reviews for developers from mobile app marketplace,” in *Proceedings of the 36th International Conference on Software Engineering (ICSE '14)*, 2014, pp. 767–778.
- [9] B. Fu, J. Lin, L. Li, C. Faloutsos, J. Hong, and N. Sadeh, “Why people hate your app: Making sense of user feedback in a mobile app store,” in *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2013, pp. 1276–1284.
- [10] R. Slavina, X. Wang, M. B. Hosseini, J. Hester, R. Krishnan, J. Bhatia, T. D. Breaux, and J. Niu, “Toward a framework for detecting privacy policy violations in android application code,” in *Proceedings of the 38th International Conference on Software Engineering (ICSE '16)*, 2016, pp. 25–36.
- [11] L. Yu, X. Luo, X. Liu, and T. Zhang, “Can we trust the privacy policies of android apps?” in *Proceedings of the 46th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN '16)*, 2016, pp. 538–549.
- [12] 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 (IMC '18)*, 2018, pp. 293–307.
- [13] M. Ali, M. E. Joorabchi, and A. Mesbah, “Same app, different app stores: A comparative study,” in *Proceedings of the 4th International Conference on Mobile Software Engineering and Systems (MobileSoft '17)*, 2017, pp. 79–90.
- [14] V. F. Taylor and I. Martinovic, “To update or not to update: Insights from a two-year study of android app evolution,” in *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security (AsiaCCS '17)*, 2017, pp. 45–57.
- [15] B. Carbanar and R. Potharaju, “A longitudinal study of the google app market,” in *Proceedings of the 2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2015*, 2015, pp. 242–249.
- [16] N. Viennot, E. Garcia, and J. Nieh, “A measurement study of google play,” in *ACM SIGMETRICS Performance Evaluation Review*, vol. 42, no. 1, 2014, pp. 221–233.
- [17] 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 *Proceedings of the 15th International Conference on Mining Software Repositories (MSR '18)*, 2018, pp. 231–242.
- [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 *Proceedings of the Network and Distributed System Security Symposium (NDSS '14)*, vol. 14, 2014, pp. 23–26.
- [19] M. Grace, Y. Zhou, Q. Zhang, S. Zou, and X. Jiang, “Riskranker: scalable and accurate zero-day android malware detection,” in *Proceedings of the 10th International Conference on Mobile systems, Applications, and Services*, 2012, pp. 281–294.
- [20] H. Wang, J. Hong, and Y. Guo, “Using text mining to infer the purpose of permission use in mobile apps,” in *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, 2015, pp. 1107–1118.
- [21] K. Mao, M. Harman, and Y. Jia, “Sapienz: multi-objective automated testing for android applications,” in *Proceedings of the 25th International Symposium on Software Testing and Analysis (ISSTA '16)*, 2016, pp. 94–105.
- [22] D. Amalfitano, A. R. Fasolino, P. Tramontana, B. D. Ta, and A. M. Memon, “Mobiguitar: Automated model-based testing of mobile apps,” *IEEE Software*, vol. 32, no. 5, pp. 53–59, 2015.
- [23] K. Chen, P. Liu, and Y. Zhang, “Achieving accuracy and scalability simultaneously in detecting application clones on android markets,” in *Proceedings of the 36th International Conference on Software Engineering (ICSE '14)*, 2014, pp. 175–186.
- [24] H. Wang, Y. Guo, Z. Ma, and X. Chen, “Wukong: a scalable and accurate two-phase approach to android app clone detection,” in *Proceedings of the 2015 International Symposium on Software Testing and Analysis (ISSTA '15)*, 2015, pp. 71–82.
- [25] 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 (CCS '16)*, 2016, pp. 356–367.
- [26] 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 (ICSE-C)*, 2016, pp. 653–656.
- [27] “Top mobile app development companies,” 2019, <https://clutch.co/directory/mobile-application-developers>.
- [28] H. Wang, Z. Liu, Y. Guo, X. Chen, M. Zhang, G. Xu, and J. Hong, “An explorative study of the mobile app ecosystem from app developers’ perspective,” in *Proceedings of the 26th International Conference on World Wide Web (WWW '17)*, 2017, pp. 163–172.
- [29] R. Balebako, A. Marsh, J. Lin, J. I. Hong, and L. F. Cranor, “The privacy and security behaviors of smartphone app developers,” in *Proceedings of the Workshop on Usable Security*, 2014.
- [30] A. I. Wasserman, “Software engineering issues for mobile application development,” in *Proceedings of the FSE/SDP workshop on Future of software engineering research*, 2010, pp. 397–400.
- [31] A. Holzer and J. Ondrus, “Mobile application market: A developer’s perspective,” *Telematics and informatics*, vol. 28, no. 1, pp. 22–31, 2011.
- [32] H. Wang, H. Li, and Y. Guo, “Understanding the evolution of mobile app ecosystems: A longitudinal measurement study of google play,” in *Proceedings of the Web Conference 2019 (WWW '19)*, 2019.
- [33] L. Li, J. Gao, M. Hurier, P. Kong, T. F. Bissyandé, A. Bartel, J. Klein, and Y. Le Traon, “Androzo+: Collecting millions of android apps and their metadata for the research community,” *arXiv preprint arXiv:1709.05281*, 2017.
- [34] “Privacygrade: Grading the privacy of smartphone apps,” 2018, <http://privacygrade.org/>.
- [35] “Aliyun,” 2019, www.aliyun.com.
- [36] “Fake apps in google play get over half a million installs,” 2018, <https://www.bleepingcomputer.com/news/security/fake-apps-in-google-play-get-over-half-a-million-installs/>.
- [37] “Pareto principle,” 2019, https://en.wikipedia.org/wiki/Pareto_principle.
- [38] “Qbiki Networks,” 2019, www.apptism.com/entertainment/qbiki-networks/.
- [39] “Andromo,” 2019, <https://www.andromo.com/>.
- [40] “Bizness Apps,” 2019, <https://www.biznessapps.com/>.
- [41] “Conduit,” 2019, mobile.conduit.com.
- [42] “BestToolbars,” 2019, <https://besttoolbars.net/>.
- [43] Y. Yuan and Y. Guo, “Boreas: an accurate and scalable token-based approach to code clone detection,” in *Proceedings of the 27th IEEE/ACM International Conference on Automated Software Engineering (ASE '12)*, 2012, pp. 286–289.
- [44] “Spam and minimum functionality,” 2019, <https://play.google.com/intl/en-US/about/spam-min-functionality/>.
- [45] I. J. Mojica, B. Adams, M. Nagappan, S. Dienst, T. Berger, and A. E. Hassan, “A large-scale empirical study on software reuse in mobile apps,” *IEEE software*, vol. 31, no. 2, pp. 78–86, 2014.
- [46] H. Wang and Y. Guo, “Understanding third-party libraries in mobile app analysis,” in *Proceedings of the 2017 IEEE/ACM 39th International Conference on Software Engineering Companion (ICSE-C)*, 2017, pp. 515–516.

- [47] H. Wang, Y. Guo, Z. Ma, and X. Chen, "Automated detection and classification of third-party libraries in large scale android apps," *Journal of Software*, vol. 28, no. 6, pp. 1373–1388, 2017.
- [48] "Meet google play's target api level requirement," 2019, <https://developer.android.com/distribute/best-practices/develop/target-sdk>.
- [49] S. Hassan, W. Shang, and A. E. Hassan, "An empirical study of emergency updates for top android mobile apps," *Empirical Software Engineering*, vol. 22, no. 1, pp. 505–546, 2017.
- [50] S. McIlroy, N. Ali, and A. E. Hassan, "Fresh apps: an empirical study of frequently-updated mobile apps in the google play store," *Empirical Software Engineering*, vol. 21, no. 3, pp. 1346–1370, 2016.
- [51] M. Ikram, N. Vallina-Rodriguez, S. Seneviratne, M. A. Kaafar, and V. Paxson, "An analysis of the privacy and security risks of android vpn permission-enabled apps," in *Proceedings of the 2016 Internet Measurement Conference*, 2016, pp. 349–364.
- [52] M. Zheng, P. P. Lee, and J. C. Lui, "Adam: an automatic and extensible platform to stress test android anti-virus systems," in *Proceedings of the 2012 International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*, 2012, pp. 82–101.
- [53] "App Store Optimization - Chandashi," 2019, <https://www.chandashi.com/>.
- [54] "App Store Optimization - qimai," 2019, <https://www.qimai.cn/>.
- [55] Y. Ishii, T. Watanabe, F. Kanei, Y. Takata, E. Shioji, M. Akiyama, T. Yagi, B. Sun, and T. Mori, "Understanding the security management of global third-party android marketplaces," in *Proceedings of the 2nd ACM SIGSOFT International Workshop on App Market Analytics*. ACM, 2017, pp. 12–18.
- [56] L. Villarroel, G. Bavota, B. Russo, R. Oliveto, and M. Di Penta, "Release planning of mobile apps based on user reviews," in *Proceedings of the 38th International Conference on Software Engineering (ICSE '16)*, 2016, pp. 14–24.
- [57] L. Guerrouj, S. Azad, and P. C. Rigby, "The influence of app churn on app success and stackoverflow discussions," in *Proceedings of the 22nd International Conference on Software Analysis, Evolution and Reengineering (SANER '15)*, 2015, pp. 321–330.
- [58] W. Maalej and H. Nabil, "Bug report, feature request, or simply praise? on automatically classifying app reviews," in *Proceedings of the 23rd International Requirements Engineering Conference (RE '15)*, 2015, pp. 116–125.
- [59] C. Iacob and R. Harrison, "Retrieving and analyzing mobile apps feature requests from online reviews," in *Proceedings of the 10th Working Conference on Mining Software Repositories (MSR '13)*, 2013, pp. 41–44.
- [60] S. Panichella, A. Di Sorbo, E. Guzman, C. A. Visaggio, G. Canfora, and H. C. Gall, "How can i improve my app? classifying user reviews for software maintenance and evolution," in *Proceedings of the 2015 IEEE International Conference on Software Maintenance and Evolution (ICSME '15)*, 2015, pp. 281–290.
- [61] I. J. M. Ruiz, M. Nagappan, B. Adams, T. Berger, S. Dienst, and A. E. Hassan, "Examining the rating system used in mobile-app stores," *IEEE Software*, vol. 33, no. 6, pp. 86–92, 2016.
- [62] Y. Tian, M. Nagappan, D. Lo, and A. E. Hassan, "What are the characteristics of high-rated apps? a case study on free android applications," in *Software Maintenance and Evolution (ICSME), 2015 IEEE International Conference on*, 2015, pp. 301–310.
- [63] R. Chandy and H. Gu, "Identifying spam in the ios app store," in *Proceedings of the 2nd Joint WICOW/AIRWeb Workshop on Web Quality*, 2012, pp. 56–59.
- [64] I. M. Ruiz, M. Nagappan, B. Adams, T. Berger, S. Dienst, and A. Hassan, "On the relationship between the number of ad libraries in an android app and its rating," *IEEE Software*, no. 1, pp. 1–1, 2014.
- [65] H. Hu, C.-P. Bezemer, and A. E. Hassan, "Studying the consistency of star ratings and the complaints in 1 & 2-star user reviews for top free cross-platform android and ios apps," *Empirical Software Engineering*, pp. 1–34, 2016.
- [66] M. Linares-Vásquez, "Supporting evolution and maintenance of android apps," in *Proceedings of the 36th International Conference on Software Engineering Companion*, 2014, pp. 714–717.
- [67] Y. Tian, B. Liu, W. Dai, B. Ur, P. Tague, and L. F. Cranor, "Supporting privacy-conscious app update decisions with user reviews," in *Proceedings of the 5th Annual ACM CCS Workshop on Security and Privacy in Smartphones and Mobile Devices*, 2015, pp. 51–61.
- [68] A. Gorla, I. Tavecchia, F. Gross, and A. Zeller, "Checking app behavior against app descriptions," in *Proceedings of the 36th International Conference on Software Engineering (ICSE '14)*, 2014, pp. 1025–1035.
- [69] K. Kuznetsov, V. Avdiienko, A. Gorla, and A. Zeller, "Checking app user interfaces against app descriptions," in *Proceedings of the International Workshop on App Market Analytics*, 2016, pp. 1–7.
- [70] A. A. Al-Subaihini, F. Sarro, S. Black, L. Capra, M. Harman, Y. Jia, and Y. Zhang, "Clustering mobile apps based on mined textual features," in *Proceedings of the 10th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*, 2016, p. 38.
- [71] M. Nayebi, B. Adams, and G. Ruhe, "Release practices for mobile apps—what do users and developers think?" in *Proceedings of the 23rd International Conference on Software Analysis, Evolution and Reengineering (SANER '16)*, 2016, pp. 552–562.
- [72] H. J. Kim, I. Kim, and H. Lee, "Third-party mobile app developers' continued participation in platform-centric ecosystems: An empirical investigation of two different mechanisms," *International Journal of Information Management*, vol. 36, no. 1, pp. 44–59, 2016.
- [73] C. Rosen and E. Shihab, "What are mobile developers asking about? a large scale study using stack overflow," *Empirical Software Engineering*, vol. 21, no. 3, pp. 1192–1223, 2016.
- [74] S. Barnett, R. Vasa, and J. Grundy, "Bootstrapping mobile app development," in *Proceedings of the 37th International Conference on Software Engineering (ICSE '15)*, 2015, pp. 657–660.
- [75] 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.
- [76] 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.