RacketStore: Measurements of ASO Deception in Google Play via Mobile and App Usage

Nestor Hernandez
FIU, Miami, USA
nestorghh@gmail.com

Ruben Recabarren
FIU, Miami, USA
recabarren@gmail.com

Bogdan Carbunar
FIU, Miami, USA
carbunar@cs.fiu.edu

Syed Ishtiaque Ahmed
University of Toronto, Toronto, CA
ishtiaque@cs.toronto.edu

ABSTRACT

Online app search optimization (ASO) platforms that provide bulk installs and fake reviews for paying app developers in order to fraudulently boost their search rank in app stores, were shown to employ diverse and complex strategies that successfully evade state-of-the-art detection methods. In this paper we introduce RacketStore, a platform to collect data from Android devices of participating ASO providers and regular users, on their interactions with apps which they install from the Google Play Store. We present measurements from a study of 943 installs of RacketStore on 803 unique devices controlled by ASO providers and regular users, that consists of 58,362,249 data snapshots collected from these devices, the 12,341 apps installed on them and their 110,511,637 Google Play reviews. We reveal significant differences between ASO providers and regular users in terms of the number and types of user accounts registered on their devices, the number of apps they review, and the intervals between the installation times of apps and their review times. We leverage these insights to introduce features that model the usage of apps and devices, and show that they can train supervised learning algorithms to detect paid app installs and fake reviews with an F1-measure of 99.72% (AUC above 0.99), and detect devices controlled by ASO providers with an F1-measure of 95.29% (AUC = 0.95). We discuss the costs associated with evading detection by our classifiers and also the potential for app stores to use our approach to detect ASO work with privacy.

CCS CONCEPTS

• **Security and privacy → Social network security and privacy**; **Social aspects of security and privacy**;

KEYWORDS

App Store Optimization; Crowdturfing; Fake Review; Opinion Spam

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
IMC '21, November 2–4, 2021, Virtual Event, USA
© 2021 Association for Computing Machinery.
ACM ISBN 978-1-4503-9129-0/21/11…$15.00
https://doi.org/10.1145/3487552.3487837

1 INTRODUCTION

The global mobile application market is worth hundreds of billions of USD and is expected to grow by more than 10% per year until 2027 [22]. To stand out among millions of apps hosted in app stores [8, 24] and get a share of this market, many app developers resort to app search optimization (ASO) to increase the rank of their apps during search. ASO platforms use a variety of techniques to achieve this [83], including providing retention installs [12] and posting fake reviews [3, 7]. Such activities can be illegal in countries like the US [1], Canada [14], Australia [77], are banned in the EU [16], violate the terms of service of app stores [2, 11], and influence users to install and purchase low quality apps and even malware [46, 68, 80].

Identifying ASO-promoted apps and the accounts from which they are promoted allows app stores to filter fake reviews and ratings, generate more accurate install count and aggregate rating values thus compute more accurate search ranks for apps, and enable users to make better informed app-installation decisions.

A key to achieve this is to build an accurate understanding of the behaviors and strategies employed by fraudulent app search optimization (ASO) workers. In previous work, Farooqi et al. [38] have shown that incentivized app install platforms (IIP) are able to provide thousands of installs that successfully evade Google defenses. Rahman et al. [67] have reported a variety of detection-avoidance techniques employed by organizations that specialize in retention installs and fake reviews. Such techniques include crowdsourcing ASO work to *organic workers*, who use their personal devices to conceal ASO work among everyday activities.

Identifying organic ASO activities is an open problem due to the ability of such workers to evade existing detection solutions, e.g., that leverage lockstep behaviors [32, 43, 51, 72, 73, 78, 85, 86, 89, 92, 93] or review bursts [27, 28, 35, 39, 40, 42, 44, 45, 50–53, 53, 57–59, 65, 66, 84, 87, 88, 92].

In an effort to determine whether solutions can be developed to detect these ASO strategies, in this paper we seek to measure and compare the device and app usage of ASO workers and regular users. Our work is partially motivated by Farooqi et al. [38]'s finding that ASO workers lack interest in the apps that they promote. We conjecture this also results in, e.g., workers posting reviews for promoted apps soon after installing them, see Figure 1.

To enable such measurements, we develop RacketStore, a platform to collect and analyze app and device-use data from consenting ASO workers and regular users. The RacketStore mobile app periodically collects data from the devices where it is installed, e.g., the

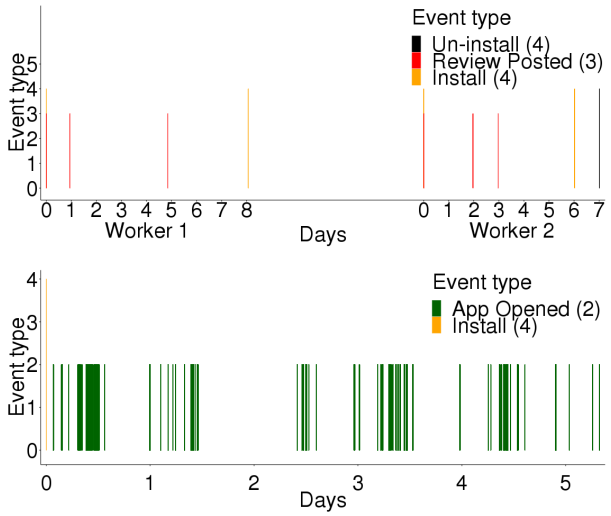


Figure 1: On-device app interaction timelines for two ASO workers (top) and one regular user (bottom). Worker timelines start with the app installation event (type 4 on y axis), followed by several review posting events across several days (type 3), with no interaction with the app. In contrast, the regular user timeline shows frequent interaction with the app, e.g., placing the app in the foreground (type 2 event), but no review even after 5 days of monitoring.

foreground app and the list of installed apps with 5s granularity, and the types and number of registered accounts, with 2 min granularity. The RacketStore server aggregates this information with data collected from the Play Store and VirusTotal [15]. RacketStore first discloses the types of data it collects. RacketStore collects the data only after receiving participant consent (§ 4.1 and Appendix D).

We present measurements from a study of ASO workers and regular users recruited to keep the RacketStore app installed on their devices for at least two days. In total, RacketStore was installed 943 times on 803 unique devices: 580 devices controlled by ASO workers recruited from Facebook groups that specialize in ASO work, and 223 devices of regular Android users recruited through ads purchased in Instagram. We have collected 58,362,249 snapshots from the participating devices, including their 12,341 apps installed and in-use, and their 110,511,637 reviews from the Play Store.

We found that ASO work continues to be successful and evade app store detection: The worker-controlled devices of participants in our studies had 10,310 Gmail accounts registered on them and at the time of writing, Google Play was still displaying 217,041 reviews posted from them.

Measurements reveal that many participant ASO worker devices have organic-indicative behaviors, i.e., similar to those of regular devices, in terms of their app churn (daily installed and uninstalled apps), permissions granted, the total number of installed apps, stopped apps, or daily used apps. However, we found significant differences between regular user and worker-controlled devices in terms of their number and types of registered accounts, the number of apps reviewed, and the intervals between the installation times of apps and their review times. This suggests that the constraints associated with ASO work provide opportunities to detect even organic workers.

To validate this, we leverage our findings and the RacketStore-collected data to develop features that model the interaction of a user with a device and the engagement of the user with the apps installed on the device. We found that supervised learning algorithms trained with these features distinguish between apps installed for promotion purposes and those installed for personal use, thus detect incentivized installs and fake reviews, with an F1-measure of 99.72% and AUC over 0.99. Further, our classifiers detect worker-controlled devices with an F1-measure of 95.29% and AUC of 0.95.

We found that 69.1% of the worker devices that we analyzed have organic-indicative behaviors, while the remaining devices were seemingly used exclusively for app promotion activities. This suggests that our device and app-engagement features can train classifiers to accurately detect not only promotion-dedicated devices but even elusive organic ASO efforts that hide low levels of app promotion activities among regular, personal use of devices and of the apps installed therein.

We note that to protect user privacy, the proposed classifiers can execute directly on the user device (e.g., implemented into the Play Store app). Locally-running classifiers can access sensitive app and device usage data and do not need to report it remotely (§ 9).

In summary, we introduce the following contributions:

- **RacketStore**. We develop a platform to collect information about the interaction of users with their Android devices and the apps installed therein, with user consent. RacketStore was compatible with 298 device models from 28 Android manufacturers [§ 3]. The RacketStore code is available at [10].
- **App and Device Use Measurements**. We present measurements from a study of the device and app use of regular users and ground truth ASO workers, through a deployment of RacketStore on 803 unique devices [§ 4]. We build datasets of app and device usage, integrated with Google Play reviews and VirusTotal analysis. We present findings from this data in the context of feedback obtained from participants during a follow-up discussion [§ 6].
- **Fraud Detector and Classifier**. We introduce novel features that model the user interaction with devices and installed apps and use them to train classifiers to detect ASO activities [§ 7] and worker-controlled devices [§ 8]. We report differences in app and device-engagement for workers and regular users that explain the accuracy of the classifiers.

2 SYSTEM MODEL

We consider the ecosystem depicted in Figure 2. In the following we describe its main components.

The App Store and Consumers. We focus our work on the Google Play app store [6]. Consumers use the pre-installed Play Store app to search and install other apps on their Android devices. A consumer can register multiple accounts on an Android device, including Gmail and other services. The consumer is then able to post reviews for an app, from all the Gmail accounts registered on the device where the app was installed.

App Developers. Developers upload their apps on the Play Store [6]. To monetize these apps while facing intense competition, they need

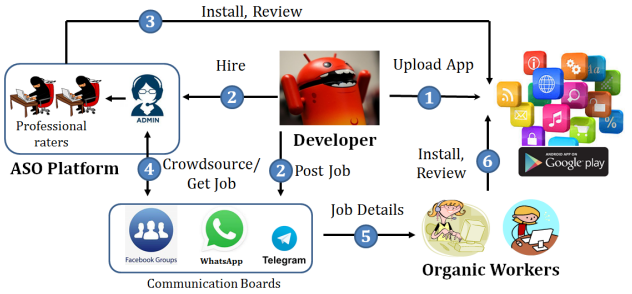


Figure 2: System model. Developers recruit ASO platforms to promote their apps in app stores. ASO platforms leverage in-house, dedicated workers, and organic workers accessed through communication boards to install and review apps.

to achieve top-5 rank in keyword searches [25]. Some of the factors with most impact on search rank are the number of installs and reviews, and the aggregate rating of the app: 80% of consumers check reviews and ratings before installing an app [34], and a 1-star increase in aggregate rating was shown to increase app store conversion by up to 280% [83].

ASO Organizations. Many developers hire specialized app search optimization (ASO) organizations to improve the search rank of their apps. While some ASO organizations are white hat [25], providing advice on e.g., app naming and keyword optimization, others provide illegal, banned or discouraged services that include installing an app on many devices and keeping it installed for prolonged intervals, a.k.a., *retention installs*, and writing fake reviews with high ratings.

We consider the ASO organizations studied in [67], see Figure 2, that employ combinations of (1) ASO admins who organize and coordinate communities of workers, and act as intermediaries between developers and workers, (2) professional workers who use multiple devices and accounts dedicated to ASO work, and (3) organic workers who blend product promotion with personal activities from their devices and accounts. In the following we informally use the terms organic device and account, to denote devices and accounts used by organic workers.

Developers can either directly hire ASO organizations or post jobs in online boards dedicated to ASO work, see next.

ASO Communication Boards. Communications between developers, admins and crowdsourced workers often occur through dedicated online boards in e.g., Facebook, WhatsApp, Telegram [67], see Figure 2. In this paper we recruited participants through Facebook groups that we identified using Facebook's search functionality (using keywords that include reviews, google reviews, app reviews, app installs, android promotion) to identify relevant groups.

We found 11 public and 5 closed groups that matched our criteria. We became members of the public groups and sent requests to closed groups which were all accepted. These groups had 86,718 members (Min = 354, Max = 26896, M = 2840.5, SD = 6787.96) in total. We detail our recruitment process in § 4.

3 MEASUREMENTS INFRASTRUCTURE

We have built the RacketStore platform to measure and compare the app and device usage of ASO workers and regular users. RacketStore consists of a mobile app to be installed by study participants on their Android devices, a web app to collect and validate data from the installed app, and database servers to store the data, see Figure 3. In the following we describe the main components of RacketStore.

RacketStore Mobile App. We have developed the RacketStore app in Android to help us investigate fraudulent and honest behaviors of Google services users. The app needs to be installed by study participants on their devices. Upon first start-up, the app displays the consent form (see Appendix C for excerpts) which the participant needs to approve. Then, to comply with the Google anti-abuse policy [23], the app asks for explicit consent of our privacy policy (Figure 18(a) in Appendix C) then shows an in-app disclosure of the data being collected (Figure 18(b)). In the following we detail the main components of the rest of the app.

The **sign-in interface** asks the participant to enter a unique *participant ID*, a 6-digit code, that we send upon recruitment (§ 4) through a different channel, i.e., e-mail or Facebook messenger. This code serves the dual goal of preventing RacketStore use by non-recruited users, and of allowing us to match data and send payments to the correct participants. The passcode is given only after the user has accepted to participate in our study and has agreed to the data collection process. RacketStore does not collect any information if the user has not entered the 6-digit passcode. Upon sign-in, the app generates the *install ID*, a 10-digit random identifier.

The **initial data collector** module operates once the app has been installed and the user has used the sign-in interface to enter the participant ID. It retrieves the list of other apps installed on the device, and device information including Android API version, device model, manufacturer, and Android ID [21].

Following the installation of the RacketStore app, the **snapshot collector** module periodically collects information with two levels of granularity, slow and fast. The slow snapshot collector is triggered by an alarm every 2 minutes, and collects (1) *identifiers*: Install ID, participant ID, and Android ID, (2) *registered accounts*, the accounts registered on the device across different services, (3) *device status*, i.e., save mode status (on/off), and (4) *stopped apps*, the list of stopped apps. Starting with Android 3.1 all applications upon installation are placed in a stopped state: the application will only run after a manual launch of an activity, or an explicit intent that addresses an activity, service or broadcast. The user can also manually force stop an app.

The fast snapshot collector module further activates every 5s and collects (1) *identifiers*, i.e., install ID and participant ID, (2) the *foreground app* currently running on the device foreground, (3) the *device status*, i.e., the screen status (on/off) and battery level, and (4) *app install/uninstall events*, i.e., deltas between the current and previously reported sets of installed apps. For each installed app we collect the install time, the last update time, the required permissions and the MD5 hash of the app apk file.

RacketStore requires participants to explicitly grant two permissions, `PACKAGE_USAGE_STATS` and `GET_ACCOUNTS` [9]. Participants can accept any subset of the requested permissions. If they do not grant a permission, we do not collect the corresponding data. RacketStore also uses install-time permissions (`GET_TASKS`, `RECEIVE_BOOT_COMPLETED`, `INTERNET`, `ACCESS_NETWORK_STATE`, and

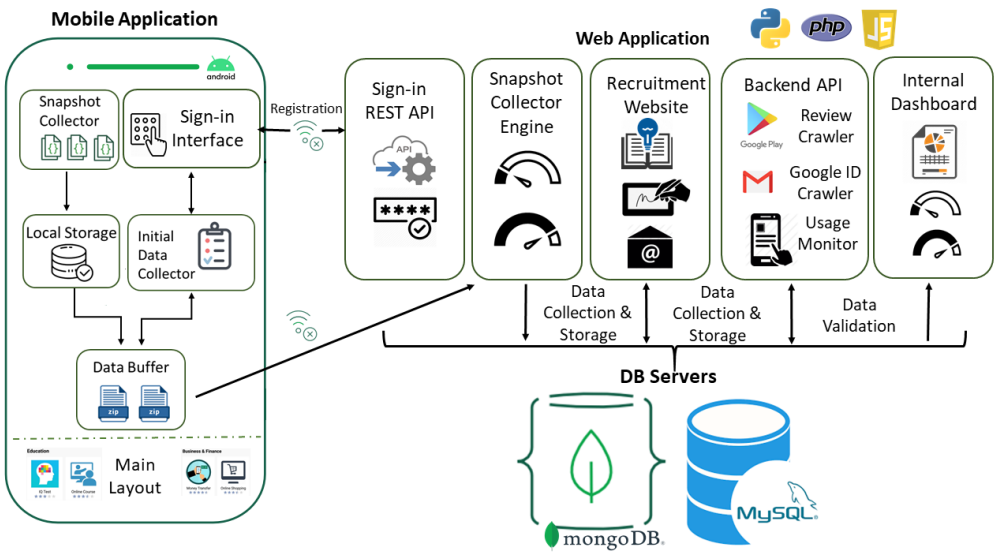


Figure 3: RacketStore architecture consists of a mobile app installed by participants and a back-end server that collects and aggregates snapshots reported by deployed apps.

WAKE_LOCK) that are automatically granted when the app is installed. The RacketStore app was approved as compliant by the Play Store.

Data Buffer Module: Snapshot Processor. The *data buffer* module (see Figure 3) leverages the device storage to process both types of snapshots. The snapshot data is written to different accumulating files depending on the snapshot type. When the slow snapshot accumulation file reaches 8KB and the fast snapshot file reaches 100KB, this module compresses the file and creates a new accumulation file to store the following snapshots. We selected these threshold values based on the observed battery and bandwidth consumption, which we sought to minimize.

The slow snapshot alarm that fires every 2 minutes looks for any existing compressed files in the mobile app's directory and sends them to our server. To enable resilient communications, upon file reception, the server returns the crypto hash of the received data in order for the mobile app to validate the transfer with its own hash calculation. If the hashes are equal, the data buffer module deletes the file.

Device Compatibility. The RacketStore app was targeted for devices with Android version 9 (compile SDK and target SDK version are Android Pie, API level 28) and is compatible with devices with Android version at or above Lollipop (min SDK version = 5, API level 21). In our deployment study (§ 4), RacketStore was compatible with 298 unique device models from 28 Android device manufacturers. The top 5 most popular Android manufacturers were Samsung, Huawei, Oppo, Xiaomi, Vivo).

RacketStore Web App. We have built a web app that supports RacketStore on the server side, see Figure 3. The Sign-in component processes registration requests from the client app, interacts with the Mongo database where the credentials are stored and sends the response back to the client. The snapshot collector engine receives the compressed snapshot files from the app, decompresses, and inserts them into the database. The backend component consists of two subsystems: (1) a review crawler that scraps reviews from Google Play given an app name and a device model and (2) a Google ID crawler that maps Gmail accounts to a unique Google ID (see § 5). The internal dashboard allows researchers to monitor the data collection process, and test and validate the data sent from the app to the server. Our stack is Linux-based and built on Python, PHP, JavaScript, MongoDB, and MySQL. The recruitment website is an informative website where we offer information to participants about our study, ask for consent, and collect their emails to send instructions on how to proceed (§ 4).

Security and Privacy Risks of RacketStore. RacketStore uses TLS to encrypt all collected data while in-transit. We also securely store all participant data on a server that is accessible from only 4 IP addresses in our campus. Further, RacketStore does not collect any information if the user has not entered a 6-digit passcode. Each study participant received a unique, random code. Participants can deny access to any of the two permissions requested by RacketStore.

RacketStore minimizes collected Personal Identifiable Information (PII) by only collecting PII that is needed for the study. Table 3 summarizes the participant PII that RacketStore collected, the reasons for the collection and how long it was stored.

4 DATA COLLECTION

In order to collect app and device usage, we deployed RacketStore with a group composed of both ASO workers and regular Google Play users. In this section we detail our recruitment process and discuss ethical considerations.

Recruitment of ASO Workers. We recruited ASO workers from Facebook groups that we found to be dedicated to product promotion (see § 2). Specifically, we posted calls to recruit participants who would be able to install and provide reviews. Many group members commented on our posts, expressed their interest, and asked us to communicate with them over the Facebook inbox for further details. We contacted such members over the Facebook

inbox. We shared with them the details of the recruitment instructions which we include in Appendix B. We asked them to reply if they were interested and also to answer screening questions, i.e., confirm that they have posted paid reviews, specify how many devices and accounts they have and on how many devices they can install RacketStore.

We sent to each prospective participant, their participant ID (§ 3) and a YouTube video that explains how to sign up for RacketStore. We have received 672 installs from 549 unique worker-controlled devices.

Recruitment of Regular Users. We have recruited regular Android device users through commercial advertisements on Instagram (see Figure 16 in Appendix B) that point to a landing page that explains the study (see Figure 17(a) Appendix B). We chose Instagram in order to minimize the exposure of Facebook groups of ASO workers to our ads.

We posted the ads intermittently between December 17, 2019 and April 15, 2020, spending a total of $79.23. Since cultural differences could affect patterns in mobile device use [17, 82], we targeted regular users of similar demographic with the recruited ASO workers. Concretely, we used Facebook's audience creation functionality to ensure that our ads were shown only to mobile devices of Instagram users who are from the countries of the above workers, are between 18 and 40 years old, speak English, and show interests related to Google Play and Android applications as specified on their Facebook profiles.

According to the Facebook Ads Manager, our ad was shown a total of 136,022 times and reached 61,748 users. 2,471 of these users clicked on the Instagram ad and made it to our landing page. The landing page introduces our study, explains the payment method (i.e., Paypal, Bitcoin or Litecoin), presents the consent form, and allows visitors to either withdraw or sign up. To sign up, a visitor needs to acknowledge and agree with the terms and conditions that are included in the consent form explaining in detail the information that we would collect from their phones (see § 4.1 for more details). If the visitor consents, the landing page asks them to register in the study by submitting their email address.

Of the consenting visitors, we have filtered out those who claimed to have written paid reviews in Google Play (question 1 in the recruitment message, see Appendix B) and who claimed to be administrators (question 6).

To each of the remaining 614 visitors, we sent an automatic confirmation email along with the Google Play link to download the RacketStore app (§ 3) and a six-digit unique participant ID that the participant would need to type-in to the app. RacketStore received 233 installs from these participants.

Participant Payments. We paid each participant who installed RacketStore on a per device basis: $1 to install the app and $0.2 for each day on which the participant kept the app installed. The process of registering in the study, providing consent and installing RacketStore takes an average of 3 minutes. Participation does not require any subsequent user interaction. We paid participants in our follow-up study $5 for each 15 minutes of their time.

Overall, we recruited a ground truth set of 587 ASO workers and 233 regular users. The participants used IP addresses from Pakistan (420), India (210), Bangladesh (148), USA (10) and other countries from Africa, Asia, South America and Europe (15). The distribution of Worker (W) and Regular (R) participants for the most represented countries was as follows: Pakistan (W: 364, R: 56), India (W: 57, R: 153), Bangladesh (W: 143, R: 5) and USA (W: 8, R: 2). We note that IP addresses can only provide an approximate measure of geolocation [56]. The RacketStore app did not collect location information from participant devices thus we cannot corroborate this information.

4.1 Ethical Considerations

Some ASO work is considered unethical according to several ethical frameworks, and many ASO workers belong to low-paid vulnerable groups. This is why our study took utmost care to follow the best ethical practices for conducting sensitive research with vulnerable populations [30]. We did not use any deception in our study. Participation in this study was completely voluntary. We did not ask any participant to write reviews for any app, including for the RacketStore app. We included the consent form both in the landing page for our study and in the RacketStore app (see excerpts in Appendix C). The consent form explicitly mentions the identity of the researchers, the research objectives, the data that we wanted to collect, and the potential impacts on participants, including risks. Our team members were also available to explain this to the participants if needed. Each participant needed to explicitly provide consent. The full study procedure was examined and approved by the Institutional Review Board of our university (IRB-19-0392@FIU).

In Appendix D we further discuss the privacy policy and permissions requested by the RacketStore app, our data protection procedure, and participant compensation.

5 DATA

We now detail the data collected by RacketStore from 943 devices between October 2019 and April 2020.

Snapshot Fingerprinting and Coalescing. We used a combination of the install ID, participant ID and Android ID collected by RacketStore, along with its install interval to address repeat installs and suspected incompatibilities of the RacketStore app, see Appendix A. After this process, we identified 803 unique devices: 580 devices controlled by ASO workers and 223 devices controlled by regular participants recruited through Instagram ads (§ 4).

We have then collected a total of 592,045 slow snapshots and 57,770,204 fast snapshots (§ 3).

Google Play Review Dataset. The review crawler of the Backend component of RacketStore's web app (see Figure 3) collects reviews posted for apps installed on participant devices every 12 hours. For each of these apps, we collected the most recent reviews by querying Google Play for reviews sorted by timestamp. The first time an app was processed, we collected reviews until hitting a threshold of 100,000 reviews. In subsequent collection efforts, we collected the most recent reviews until finding a previously collected review. This procedure allowed us to collect reviews "live" as soon as our RacketStore mobile application discovered a new app on a participant's device.

Further, we collected reviews posted by accounts registered on participant devices. This process uses the Google ID crawler component of the Backend API in RacketStore's web app (Figure 3),

maps e-mail accounts to Google IDs. To achieve this, for each Gmail account registered on a device, the ID crawler issues requests to Gmail's email search functionality. This is because we found that responses of Gmail's email search functionality embed the Google ID. We then list all the apps installed on each device, that are hosted on the Play Store. Then, for each such app, we search the Google IDs corresponding to accounts registered on the device, among the Google IDs collected by the above review crawler from the Play Store for the app.

We collected 110,511,637 reviews from 12,341 apps installed on participant devices. Each review includes metadata, e.g., the user's Google ID, posting timestamp (1s granularity) and rating.

We also collected 217,041 reviews posted by the 10,310 Gmail addresses registered on the worker-controlled devices of the participants in our studies, with participant consent (§ 4.1). We have reported to the Google Vulnerability Reward Program (VRP, issue ID: 156369357) the functionality that we used to collect this data. More specifically, our finding that responses of Gmail's email search functionality embed the user's Google ID. This allows a third party with a Gmail address, to obtain the account's Google ID, then determine if the account has posted a review in the Play Store for any app of interest. Google responded that this is "intended behavior" and made the decision "not to track it as a security bug". We notified study participants about this data collection (§ 4.1).

6 DEVICE USAGE MEASUREMENTS

We use the data collected from the participant devices (§ 5) to investigate differences between workers and regular users in terms of the accounts registered and the apps installed on their devices.

We used the Kolmogorov–Smirnov (KS) test to compare the distributions of the features, and non-parametric and parametric ANOVA (Analysis of Variance) to check for significant differences in the means across groups. The reason for this is that after performing the Shapiro test, we can not claim normality for any of the features (p-value$<$ 0.05). Similarly, after performing the Fligner-Killen test, we found significant differences in the variances for all the features (p-value$<$ 0.05). Thus, we have also performed non-parametric ANOVA since normality is not assumed. We report the result of the three approaches.

6.1 Participant Engagement

Figure 4 shows the scatterplot of the average number of snapshots per day vs. the number of active days over the participant devices. Larger dots denote multiple overlapping devices. The average number of daily snapshots collected from regular devices is 9,430.71 (M = 3,097.67, SD = 12,789.14, max = 63,452) and from worker devices is 8,208.10 (M = 3,669, SD = 10,303.42, max = 55,281.38). The maximum number of snapshots per day is 55,281.38. We observe that 529 devices have reported at least 100 snapshots per day.

6.2 Registered Accounts

To post a review, a user needs to have a Gmail account. For one app, a single review can be posted from any Gmail account. We now investigate differences in the number and types of accounts controlled by workers and regular users. We expect that workers will have more Gmail accounts registered on their devices than regular users since this impacts the number of reviews that they can post.

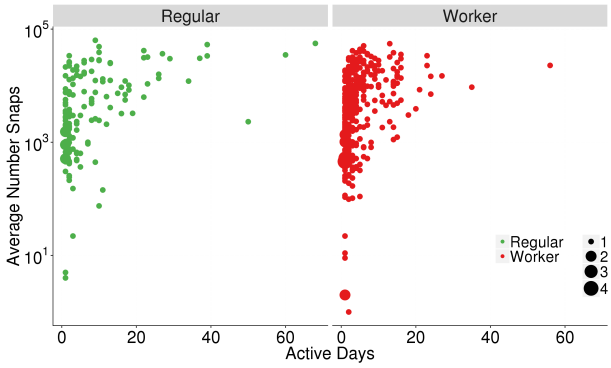


Figure 4: Scatterplot of average number of snapshots collected per day vs active days over regular (green), and worker (red) devices. Dot size indicates the number of overlapping devices. Most devices report at least 100 snapshots per day.

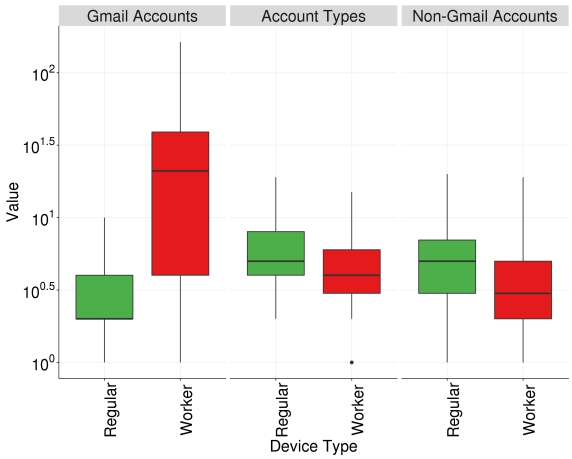


Figure 5: Comparison of the number and types of accounts registered on devices controlled by ASO workers and regular participants. Worker devices tend to have more Gmail accounts, but fewer account types and non-Gmail accounts than regular devices.

Figure 5 (left) shows the number of Gmail accounts registered on the 145 regular and 390 worker devices that have reported such information. The other participants either did not grant this permission or the server did not receive enough snapshots from their devices. We found significant differences between regular user and worker-controlled devices. Worker devices have an average of 28.87 accounts registered per device (M = 21, SD = 29.37); 13 worker devices have more than 100 Gmail accounts registered, with a maximum of 163 accounts per device. In contrast, regular devices have a maximum of 10 accounts registered (M = 2 , SD = 1.66). The KS test, and parametric and non-parametric ANOVA found statistically significant differences between workers and regular users (p-value$<$ 0.05).

Figure 5 (center) shows the number of different account types registered on participant devices. On average, regular devices have

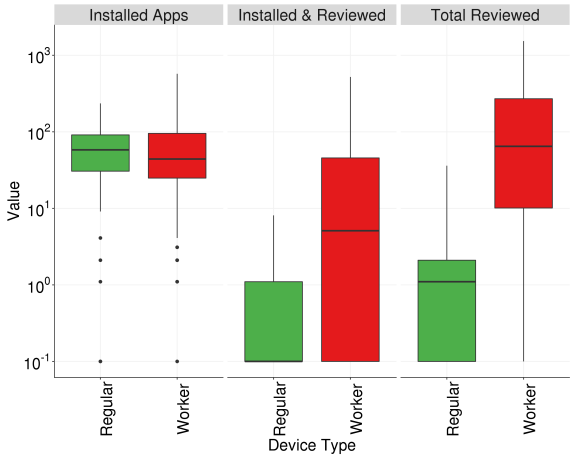


Figure 6: Number of installed apps (left), installed and reviewed (center), and total number of reviews posted from all accounts registered (right). We see dramatic differences between worker and regular devices in the apps reviewed from all their registered accounts.

registered accounts for 6 services (max = 19), mostly for different social networks (Facebook, WhatsApp, Telegram, etc). In contrast, worker devices have accounts mainly for Google services and other services useful for ASO work, e.g., dualspace.daemon (to enable installation of the same app multiple times) and freelancer (to find work). Both KS and ANOVA analyses reveal significant differences between workers and regular users ($p-$value<0.05) in terms of their numbers of non-Gmail accounts.

We have followed up with several participant workers. Six of the ASO workers who replied claimed that they personally own only 1-4 Google accounts. For instance, one worker said "*I have two accounts. One account is mine, another is my mom's.*" Four other workers however claimed (and we verified) to control between 10 to 50 Gmail accounts, and one claimed to have "*many accounts*".

Summary of Findings. We confirmed that participant ASO workers have registered significantly more accounts on their devices than regular users. Workers have however less diversity in the online sites for which they registered accounts. Their accounts are specialized for ASO work, focusing on Gmail and Dualspace that enable them to install and review a single app multiple times.

6.3 Installed Apps

We now investigate the hypothesis (inspired from [38]) that workers and regular users differ in the manner in which they interact with installed apps.

Apps Installed and Reviewed. Figure 6 compares the distribution of the number of installed apps (left), the number of apps installed and reviewed (center), and the total number of apps reviewed from any account registered (right) for the 143 regular and 400 worker devices that reported this data. We observe dramatic differences between worker and regular devices in terms of the total number of reviews posted from registered accounts: On average, a worker device is responsible for a total of 208.91 reviews, while a regular user device has only posted an average of 1.91 reviews. We found 11 worker-controlled devices each responsible for more than 1,000 total reviews. In contrast, the maximum number of total reviews from

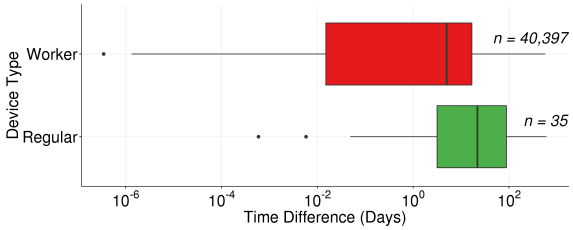


Figure 7: Distribution of time between app install and app review, for regular and worker devices. Each point is one review. Unlike regular users, worker-controlled accounts post many more reviews and tend to do it soon after installation. 13,376 of the reviews from worker accounts were posted after at most one day after installation.

a regular device is only 36. We found statistically significant differences between workers and regular users via KS and parametric and non-parametric ANOVA (p-value< 0.05).

We observe an average of 65.45 and 77.56 apps installed on regular and worker devices respectively. KS reported significant differences in the distributions (p-value$= 0.008$), but ANOVA did not find a statistically significant difference (p-value$= 0.301$). This is expected since the number of installations is limited by the device resources. However, on average, worker devices have posted reviews for 40.51 of the currently installed apps, while regular devices did it for an average of 0.7 apps.

Install-to-Review Time. The Android API reports the time of each app's last install. There are thus two cases. First, the time when an app was installed on a participant device is before the time when the participant reviewed the app. This is likely the case where the collected review is from the currently installed version of the app on that device. Second, the install time is after the review time. This implies that the review we collected is from a previous install of the app. For our analysis, we only consider the first case.

Figure 7 shows the distribution of the time between app install and app review, for regular and worker devices. Each point is one review. An app can be reviewed from multiple accounts registered on the same device; each such review is a different point. To compute the install-to-review time, we used the Android API to get the installation time, and our review crawler to get the review timestamp. However the Android API only retains the last installation time of an app in a device. We have not considered reviews whose install-to-review times were negative, since they are the result of a past install.

We observe substantial difference in the number of reviews posted from accounts registered on worker vs. regular devices for apps that provided an installation time: accounts on regular devices only wrote 35 reviews, while those on worker devices posted 40,397 reviews. Both ANOVA and KS tests found statistically significant differences between the two groups ($p-$value<0.05).

Further, workers tend to review apps much sooner after installation. 13,376 of the 40,397 reviews posted from the accounts registered on worker devices were posted after at most one day after the app was installed. Workers register an average of 10.4 days of waiting time between installation and review (M = 5.00 days, SD = 13.72 days, max = 574 days). We have observed 25 cases with waiting times longer than 100 days and 4 cases of reviews posted

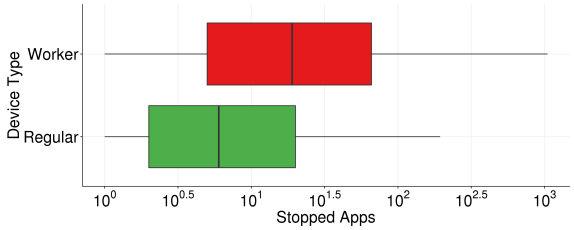


Figure 8: Boxplot of stopped apps for regular and worker devices. Worker devices tend to have more stopped apps, but we also observe substantial overlap with regular devices.

after more than 1,000 days from 2 workers and for 2 apps (Facebook and Easypaisa). These rare cases of prolonged waiting times are expected of apps used for personal purposes.

In contrast, only 4 out of the 35 reviews posted from the accounts of regular users were posted after at most one day after install. Regular users also wait for 85.09 days to post a review on average (M=21.92 days, SD=140.56 days, max=606.11 days) with only 12 users waiting less than 12 days to post a review. This longer waiting time is consistent with a review activity that proceeds from a previous interaction to form a judgment, and is inconsistent with paid promotion services. KS and ANOVA found statistically significant differences ($p-$value<0.05).

We have observed 25 cases for worker devices having waiting times longer than 100 days, and 4 cases of reviews posted after more than 1,000 days (from 2 workers for Facebook and 2 for Easypaisa). These rare cases of prolonged waiting times may be indicative of apps used for personal purposes.

Stopped Apps. We have further studied the number of apps that are stopped on the devices of regular and worker participants. A freshly installed app is in a stopped state until the user opens it for the first time. Android devices also allow users to stop apps, instead of uninstalling them. Figure 8 shows that some worker devices have significantly more stopped apps than regular devices. KS test and ANOVA found statistically significant differences between workers and regular users (p-value< 0.05). We conjecture that this occurs because ASO workers (1) often do not open the apps that they install in order to promote, and (2) even if they open them and need to keep them installed, e.g., to provide *retention installs* (§ 2), they prefer to stop apps that misbehave.

We followed up with several participants to clarify this point, i.e., whether they stop apps and why. Eight workers claimed to never stop apps, which suggests reason #1 applies. One worker however admitted reason #2 applies, i.e., "*The quality of some apps was bad, I stopped those apps*". Another claimed that limited storage is to blame, "*Sometimes the apps get hanged due to a lack of storage*".

Third-Party App Stores. We observed that some participant devices had apps installed that were not available in Google Play. We followed up with participants to ask if they install apps from other app stores. The workers conjectured that Google does not host such apps because they violate Google's policy, e.g., "*Google's policy prohibits the use of such apps*, or because they are not secure.

One worker admitted to have installed apps from third-party stores, i.e., "*The client gives us a link, we go and install that app*". Three other workers claimed to install apps from other app stores

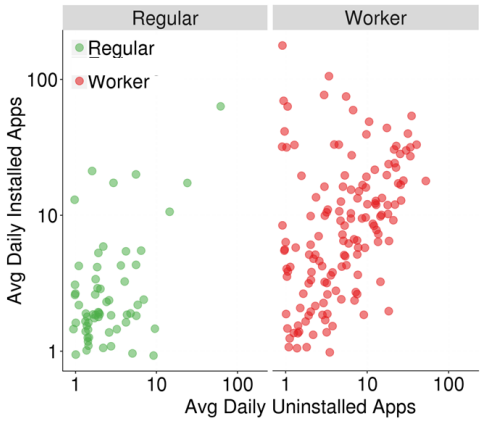


Figure 9: App churn: Scatterplot of average number of daily installs vs. average number of daily uninstalls (log scale) for regular and worker-controlled devices. Each dot is one device. The app churn of most regular devices is less than 10 apps per day, while for many worker devices it is above 10 apps per day.

for personal reasons, e.g., to play games (Dream11) or avoid subscription fees (Netflix, Hotstar) by installing *modded* apps [1]:

"*I use a modded version of the apps that are not in the Google Play Store. You do not have to open an account to use these. For instance, Netflix or Hotstar apps charge a subscription fee every month. But I don't have that much money so I install the modded version. By doing this I get premium access for free.*"

App Churn: Install and Uninstall Events. Figure 9 shows the average number of daily install events and daily uninstall events for participant worker and regular devices, computed over all the days when RacketStore was installed. Workers tend to install apps more often compared to regular users. Concretely, worker devices had an average of 15.94 daily installs (M = 6.41, SD = 27.37) while regular devices had an average of 3.88 daily installs (M = 2.0, SD = 7.29). KS test and ANOVA reported statistically significant differences between the two groups (p-value< 0.05).

We recorded fewer daily uninstalls, suggesting that participant devices tend to retain apps: worker devices recorded an average of 7.02 daily uninstalls (M = 2.73, SD = 15.69) and regular devices had an average of 3.29 daily uninstalls (M = 1.8, SD = 6.87). The KS and ANOVA tests reported significant differences at p-value<0.05. We observe however that several worker devices have a low daily app churn, while some regular devices have a higher daily app churn, making them harder to distinguish based on this feature alone. We note that the differences in app churn between workers and regular participants could also be due to background differences: ASO workers may be more technically skilled due to the nature of their work. However, we were also surprised by the relatively high app churn of regular users. One reason for this may be that active Instagram users may be more technically skilled than regular people. We did not investigate the technical expertise of participants in our study.

[1] A *mod app* is a modified version of an original apk, not signed by the original developers. A modded app may have additional features, unlocked features, and unlimited in-app currency.

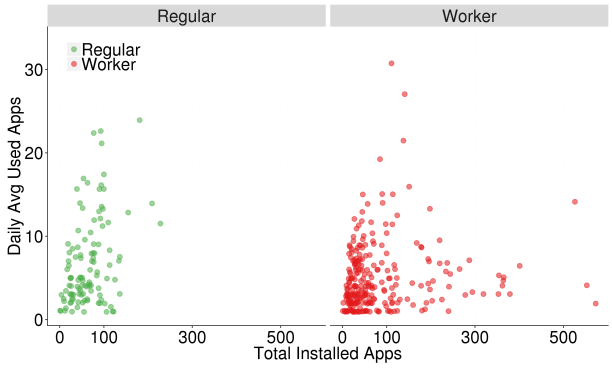


Figure 10: Scatterplot of the average number of apps used per day per device and the number of apps installed in a device, for regular and worker devices. We observe substantial overlap between regular and worker devices.

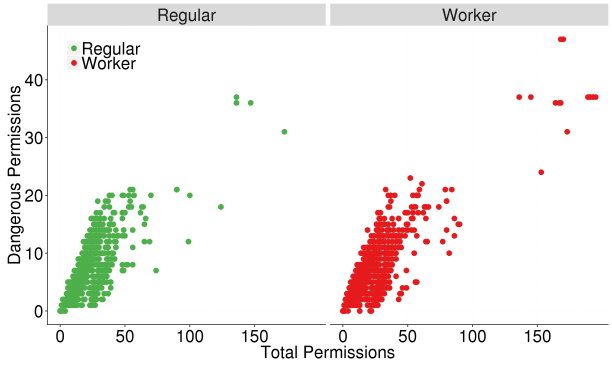


Figure 11: Comparison of exclusive app permissions for regular and worker devices. Worker devices host apps with the largest ratio of dangerous to total number of permissions.

Number of Apps Used Per Day. Figure 10 shows for each of the 141 regular and 399 worker devices in our studies (total of 540), the average number of apps opened per day on the device vs. the total number of apps installed on that device. We observe that several worker devices have many more apps installed than regular devices, and also have more apps used per day. Nevertheless, we also observe substantial overlap in these features between regular and worker devices, perhaps due to the fact that several of the worker devices are organic. This suggests that the daily number of used apps cannot accurately distinguish between worker and regular devices.

App Permissions. We studied the distribution of permission requirements for unique apps found on participant devices. Figure 11 shows the number of dangerous permissions vs. the total number of permissions for each app found exclusively on regular and worker devices. We found that while some worker devices host apps with the largest number of dangerous permissions, most installed apps share a similar permission profile across all device types. This suggests that the number of permissions requested by an app, including the dangerous permissions, will be ineffective in detecting promoted apps.

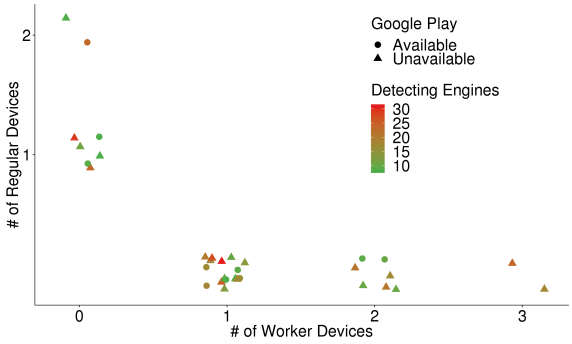


Figure 12: Comparison of malware occurrence in regular versus worker devices. Each point corresponds to a unique app apk hash, with at least 7 VirusTotal flags. The color in the legend refers to the number of VT engines that flagged the app. Worker devices host more unique malware which tends to be present on more devices than for regular users.

We contacted workers to ask about their policy for granting permissions to apps they promote. Five claimed to grant all permissions requested by the apps that they install. However, one participant claimed selective granting, i.e., "*Permissions are given based on the client request. If the client does not ask, we do not give all permissions*". Four other workers said that there are permissions that they grant grudgingly, e.g., one claimed "*I don't like the location permission because it violates my privacy*", while two others said claimed to dislike permissions associated with personal data. Two regular participants claimed not to grant all requested permissions. One claimed to avoid granting location permissions, the other was concerned about contacts, images and phone storage permissions.

Summary of Findings. In our study, workers posted reviews for a significantly higher number of installed apps. Following app installation, participant workers waited significantly shorter times to post reviews than regular users. While workers tend to install more apps per day than regular users, their devices also had significantly more stopped apps. We conjecture that this happens due to retention install requirements: workers need to keep promoted apps installed, but want to avoid the clutter.

6.4 Investigation of Malware and Attitudes

We now investigate the potential and perceived impact of malware installations on the workers who participated in our study. We used the VirusTotal research license reports [79] to analyze the presence of malware on the participant devices. VirusTotal uses 62 detection engines to process apk files. We used the *snapshot collector module* (§3) to collect 18,079 distinct hashes corresponding to 9,911 unique mobile app identifiers installed in 713 participant devices (549 devices of workers and 164 devices of regular users). The remaining 90 devices did not provide hash information either because of permissions or API incompatibility problems. We collected reports for these hashes in VirusTotal; 12,431 hashes were available in VirusTotal. We did not collect details about the specific malware families and types detected by the VirusTotal engines, including information about potentially unwanted programs (PUP).

177 of these apps were flagged malicious by more than one VirusTotal AV tool. We found at least one of these flagged apps in 183 unique devices: 122 devices controlled by workers and 61 devices of regular users. We found 70 unique mobile app identifiers with at least one VT engine flag, that received at least one review from our participants: 64 of these apps were reviewed by workers, and 9 apps were reviewed by regular users.

Since a single VT flag may be a false positive, we further compared the occurrence of the most malicious malware samples (flagged by more than 7 VirusTotal engines) in regular user devices versus worker devices. The 7 flag threshold exceeds the value 4 identified in [64], and most of these samples were later removed from Google Play. Figure 12 shows that malicious samples are more likely to appear in several worker devices when compared to regular users.

Anti Virus (AV) Apps. To determine if participants had sufficient security concerns to install anti-virus (AV) apps, we first identified 250 anti-virus (AV) apps from Google Play, by doing a search on the app category in the website. We have joined these apps against the apps installed in each of the participant devices that sent at least one snapshot. We found only 19 devices that installed 15 AV apps: 8 worker devices, 7 regular user devices, and 4 unknown (i.e., either Google testing our infrastructure or participants who managed to bypass our invitation code).

Participant Feedback. We asked participants if they (1) are concerned about installing malware apps on their devices, (2) have anti virus software installed, and (3) are concerned about the privacy of their device data, including contacts, login info, pictures, videos, text messages, location. Two workers were not concerned about malware or privacy leaks and did not have AV apps installed. One worker said "*I am confident on the ability of my phone to prevent any mishap*".

Five workers reported being concerned about malware; four claimed to use AV apps. One worker claimed that "*I find a lot of apps like this, which contain a lot of viruses.*" However, he also claimed to not be concerned about privacy leaks because "*I have 5 devices, 3 mobile devices and 2 computers. 2 out of the 3 mobiles, I use for apps testing and review, 1 mobile I use for my personal work.*"

Summary of Findings. Worker-controlled devices have lower instantaneous malware infection rate than regular devices. However, malware installed by worker devices tends to be flagged by more anti-virus engines. Workers had mixed attitudes toward privacy and installing malware and AV apps. This suggests potential vulnerabilities and concerns among workers toward keeping apps installed for longer intervals.

7 FAKE REVIEW DETECTION

In this section we investigate whether the app usage data collected by RacketStore (§ 5), can identify apps installed to be promoted, thus detect fake app installs and reviews. For this, we first introduce app usage features (§ 7.1) then investigate their ability to train supervised models to distinguish promotion-related vs. personal app use (§ 7.2).

7.1 App Usage Features

We extracted the following features for each app installed on participant devices: (1) the number of accounts registered on the device that reviewed the app, before RacketStore was installed, while it was installed, and after it was uninstalled, (2) the install-to-review time (§ 6.3), (3) inter-review times, i.e., statistics over the time difference between all consecutive reviews posted for the app from accounts registered on the device, (4) whether app was opened on multiple days, (5) the number of snapshots per day when the app was the on-screen app, (6) the number of snapshots collected per day from device, (7) *inner retention*, i.e., the duration over which the app was installed on the device (while RacketStore was installed), whether the app was installed before RacketStore and was still installed when RacketStore was uninstalled, (8) the number of normal and dangerous permissions requested, (9) the number of permissions requested by the app that have been granted and denied by the user, (10) the number of flags raised by VirusTotal AV tools, and (11) the number of times the app was installed and uninstalled while RacketStore was installed.

ML Algorithm	Precision	Recall	F1
XGB	**99.78%**	**99.67%**	**99.72%**
RF	99.33%	99.23%	99.27%
LR	99.22%	99.00%	99.11%
KNN	96.88%	96.88%	96.88%
LVQ	90.99%	94.54%	92.73%

Table 1: Precision, recall, and F-1 measure of app usage classifier (CV $k = 10$) using Extreme Gradient Boosting (XGB), Random Forrest (RF), Logistic Regression (LR), K-Nearest Neighbors (KNN), and Learning Vector Quantization (LVQ). XGB performed the best.

7.2 App Classification

We use these features and the datasets of § 5 to train an app classifier that determines if an app has been installed for promotion or personal use.

Training and Validation Datasets. We use the 178 worker and 88 regular devices from which we have received at least two days of fast and slow snapshots. For the other devices we lack enough data to extract good features.

We have set aside randomly selected 20% (i.e., 38) of the worker-controlled devices and 42% (i.e., 37) of the regular devices. We use these devices to select a set of train-and-validate apps with suspicious and regular usage. Specifically, we label an app to be suspicious if (1) it was advertised by workers for promotion on the Facebook groups we infiltrated (§ 2), (2) it was installed on at least five worker devices, and (3) was not installed on any regular devices. The rationale for this selection is that co-installing apps that are not popular and we know have been promoted, is likely the result of ASO work. Further, we label an app to be non-suspicious or regular, if (1) it was not installed on any worker-controlled device, (2) was installed in at least one regular device, and (3) has received at least 15,000 reviews. We have identified 1,041 suspicious apps among the ones installed on the above 38 worker-controlled devices, and 474 non-suspicious apps among those installed on the 37 training regular devices.

We use these apps to build a *train-and-validate app usage dataset* consisting of 2,994 suspicious instances and 345 non-suspicious instances. An instance consists of an app A and a device D on

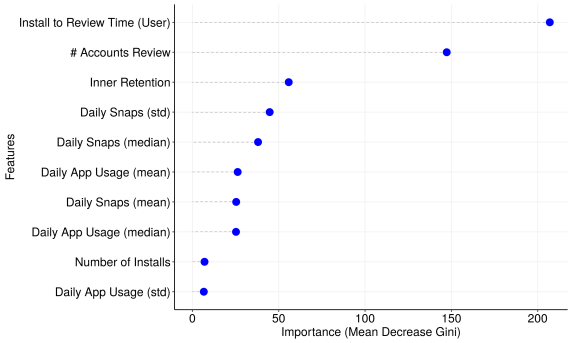


Figure 13: Top 10 most important features for the app classifier, measured by mean decrease in Gini. The number of accounts that have reviewed the app from the device and the average time between install and review are most important.

which A has been installed, features extracted from the use of A on the device D (§ 7.1), and a class label (1 for promotion and 0 for personal usage instance).

Classifier Performance. We evaluate the performance of supervised learning algorithms trained with the features introduced in § 7.1 on the train-and-validate app usage dataset. For this, we use repeated 10-fold cross-validation (n=5) over the 2,994 suspicious app usage instances and 345 regular usage instances. Table 1 shows the precision, recall and F1-measure of tested algorithms. Extreme Gradient Boosting (XGB) outperforms the other algorithms, achieving an F1-measure of 99.72%. KNN achieved best performance for $K = 5$.

Figure 13 shows the top 10 most important features to classify app usage, measured by the mean decrease in Gini [31]. A higher decrease in Gini indicates higher variable importance. We observe the importance of the number of accounts registered on the device that have reviewed the app (§6.2) and the average time between install and review (§6.3).

Performance Under Balanced Datasets. We also evaluate these algorithms trained with balanced datasets of promotion and personal app use instances [90]. Experiments with undersampling the majority class and oversampling the minority class obtain similar performance (F-1 value of 98.76% and 99.22% respectively for XGB). The AUC value is over 0.99 across all the algorithms except for KNN where the AUC decreased to 0.90 and 0.92 in undersampling and oversampling respectively. For XGB, the false positive rate is 1.94% when using oversampling.

8 ASO DEVICE DETECTION

We now investigate whether the device usage data collected by RacketStore (§ 5) can be used to identify devices controlled by app search optimization workers.

8.1 Device Usage Features

We introduce the following features that model the use of a device: (1) the number of pre-installed and user-installed apps, (2) *app suspiciousness*, i.e., the number of apps that were flagged by the app classifier of § 7, over the total number of apps installed on the device, (3) the number of apps that were stopped (§ 6.3), (4) the average number of apps installed and uninstalled per day, (5) the number of device-registered Gmail and non-Gmail accounts, and the number of distinct account types, (6) the number of apps installed on the device that have been reviewed from accounts registered on the device, and (7) the total number of apps reviewed by accounts registered on the device. For most features we use both the user-installed apps and the pre-installed apps, since even the use of pre-installed apps like the app store, e-mail, maps, and browser apps can distinguish regular devices from those controlled by workers.

ML Algorithm	Precision	Recall	F1
XGB	**96.81%**	**93.81%**	**95.29%**
RF	93.95%	96.06%	94.99%
SVM	96.64%	89.03%	92.68%
KNN	94.29%	90.58%	92.40%
LVQ	96.40%	82.84%	89.11%

Table 2: Precision, recall, and F-1 measure of device classifier (CV $k = 10$) using Extreme Gradient Boosting (XGB), Random Forrest (RF), K-Nearest Neighbors (KNN), Learning Vector Quantization (LVQ),and Support Vector Machines (SVM). XGB performed the best.

8.2 Device Classification

We evaluate the ability of these features to train classifiers that differentiate between devices controlled by workers and regular users. For this, we use the 178 worker devices and the 88 regular devices that have reported snapshots over at least 2 days. We prioritize precision, since a low precision would lead the app market to take wrong actions against many regular devices [90].

Table 2 compares the performance of five supervised learning algorithms trained with the device usage features introduced in § 8.1. KNN achieved best performance for $K = 5$. To balance the worker and regular user device classes, we oversampled the minority class using the SMOTE algorithm [33]. We use 10-fold cross-validation over the data from the 178 worker and 88 regular devices. Extreme Gradient Boosting (XGB) outperforms the other algorithms, achieving an F1-measure of 95.29% and AUC of 0.9455. The precision is 96.81% and the false positive rate is 1.41%.

When we undersample the majority class, XGB's recall decreases to 92.97% with an F-1 value of 95.18% and AUC of 0.9074. When using no sampling strategy the F-1 increases to 96.86%, at the expense of the AUC (0.9083).

Figure 14 shows the top 10 most important features in classifying devices as worker-controlled or regular, as measured by the mean decrease in Gini. Four features stand out, confirming their ability to detect worker-controlled devices, i.e., (1) the total number of apps reviewed from accounts registered on the device, (2) the percent of installed apps that were detected to have been used suspiciously by the classifier of § 7, (3) the number of stopped apps on the device and (4) the average number of reviews posted from an account registered on the device.

Figure 15 shows the scatterplot of app suspiciousness vs. the total number of reviewed apps for each of the 178 worker-controlled devices. Out of these 178 devices, 123 devices have organic-indicative behaviors, with at least one of the installed apps being predicted to

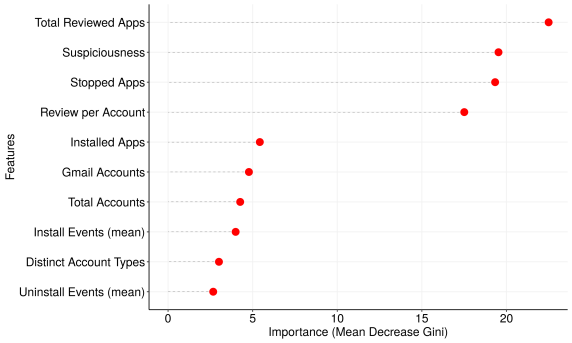


Figure 14: Top 10 most important features for the device classifier, measured by mean decrease in Gini. This suggests that devices controlled by workers are distinguishable from regular devices on their total number of apps reviewed, percentage of apps used suspiciously, and number of stopped apps.

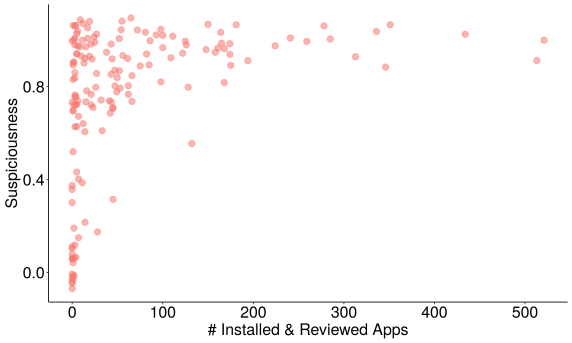


Figure 15: Scatterplot of 178 worker-controlled devices (one dot per device): app suspiciousness vs. the number of apps installed and reviewed from accounts registered on the device. This reveals that classifiers can detect a range of worker-controlled devices that includes promotion-dedicated devices and devices with organic-indicative usage.

be used for personal purposes. The remaining 55 devices seem to have been used exclusively for app promotion purposes: all their apps have promotion-indicative behaviors, their median number of Gmail accounts is 31 (M = 37.18, max = 114), and have a median of 23 stopped apps (M = 66.23).

We have manually investigated the devices with high but under 100% app suspiciousness, and confirmed that such devices are likely to have installed and used apps for personal purposes. Examples include train ticketing apps used at similar times over multiple days, photo gallery apps used in alternation with video players, Samsung pre-installed messaging (`com.samsung.android.messaging`) and call (`com.samsung.android.incallui`) apps, and music apps such as Google Play Music being used every day.

However, the classifiers were able to accurately detect even worker-controlled devices with low app suspiciousness, that may belong to novice workers.

9 DISCUSSION AND LIMITATIONS

Who Should Deploy RacketStore? The classifiers proposed in § 7 and 8 need more information than what is made publicly available by app stores (e.g., via the Google APIs). Thus, the classifiers can only be effective for RacketStore users. We also note that ASO workers would likely be reluctant to install RacketStore without proper incentives. Therefore, to scale RacketStore's processing of all the apps in an app store, the proposed classifiers should be embedded by app store developers into their pre-installed apps, e.g., the Google Play Store app or Digital Well Being app. Unlike third-party apps, such clients have by default the permissions to access the required data, and are known to access at least app usage details [5, 49].

Privacy-Preserving Classifiers. We note that our classifiers need access to sensitive data from user devices, which general users may be reluctant to share. To address this problem, we propose a privacy-preserving approach where our pre-trained models (§ 7 and § 8) execute on the user device on locally computed features to detect ASO activities [4, 81]. This approach will only report ASO suspicious activities but no private app and device-usage information. A red flag can be raised if the user uninstalls or blocks this pre-installed client (e.g., the Play Store app), and posts suspicious reviews from accounts registered on such a non-consenting device.

Worker Strategy Evolution. ASO workers may attempt to develop strategies to avoid detection by our classifiers. However, our engagement-based features exploit the lack of genuine interest of workers on promoted apps and introduce a tradeoff between detectability and operational costs and exposure to malware. These features include the number of accounts registered on the device, the interval between app install and first review, and user interaction with the app (opened, daily app usage) see § 7.1.

Workers may attempt to manipulate these features using perhaps existing software. However, workers will still need to keep promoted apps installed for longer intervals, wait more before reviewing them, and interact more with them. When promoted apps are malicious, and workers use personal devices, this may increase the exposure of their personal information to threats. In § 6.4 we show that workers expressed awareness and concern about malware apps. Further, to avoid detection, workers will be forced to register fewer accounts and post fewer reviews for promoted apps from these accounts. This can significantly reduce the amount of fraud posted, thus reduce worker profits from ASO activities.

Recruitment Bias. We contacted ASO workers using Facebook groups dedicated to product promotion, and recruited only those who responded, were English speakers, and were willing to participate after approving the consent form. Our Instagram recruitment process reached 61,748 Instagram users who speak English, are of restricted age, show interests related to Android applications, and were willing to participate after approving the consent form.

To reduce the impact of cultural factors in our analysis, we have attempted to recruit both workers and regular users from roughly the same regions. While the distribution of workers and regular users is not uniform for most countries of our participants, 96% of the worker devices and 92% of the regular devices seem to be (according to the unreliable IP-based geolocation) from the geographically close Pakistan, India and Bangladesh.

We do not claim that our results generalize to all workers and regular users, including from the same and other countries. Further, workers accessible through other recruitment channels, e.g., [13] may have different behaviors and strategies. A larger scale recruitment process may identify further types of ASO workers and more

diverse regular users. However, the data that we collected from 803 participant devices provides evidence on the ability of device and app usage data to detect the devices controlled, and the reviews posted by different types of workers.

Classifier Performance. Several machine learning algorithms achieve an F1-measure that exceeds 99% for the app classification problem (§ 7.2), while one algorithm achieved an F1-measure over 95% for the device classification problem (§ 8.2). The investigation in § 6 provides an intuition for the ability of several features to help classifiers distinguish between apps and devices used for personal purposes vs. ASO work. This suggests that these algorithms did not overfit the data. Further, the success of these classifiers suggests that standard ML algorithms are suitable and preferable for these classification problems, where they can provide valuable interpretation.

We acknowledge however that the relatively small and biased data that we used to train the app and device classifiers (see recruitment bias above) may lead to reduced applicability to data from other ASO workers and regular users.

Influence of RacketStore on Participant Behaviors. Knowledge of being monitored might have influenced participant behaviors. We note however that all participants, including ASO workers and regular users, installed the same version of RacketStore and were provided with the same information before and during the study. Further, our classifiers were able to distinguish between apps and devices used by ASO workers and regular users, even if ASO workers attempted to modify their behaviors during the study.

10 RELATED WORK

Farooqi et al. [38] studied the market of incentivized app install platforms (IIP) through a honey app that collects the device id, the list of installed apps and events such as opening the app and in-app interaction. We leverage Farooqi et al. [38]'s finding of a lack of interest in the app among the workers that installed it for money. RacketStore extends Farooqi et al. [38]'s work by collecting and analyzing additional key data that notably includes the list of user accounts registered on the participant device, the reviews posted from those accounts, and the foreground app at 5s intervals. This data enables us to claim a first success in identifying organic ASO activities. Further, our study involved diverse types of ASO workers that we recruited from Facebook groups, and regular users that we recruited using Instagram ads.

Our work is particularly relevant in light of findings that some ASO workers have evolved strategies [67, 94] to evade detection by both app stores and academic solutions, e.g., [27, 28, 32, 32, 35, 39, 40, 42, 44, 45, 48, 50, 51, 51–53, 53, 57–59, 72, 73, 78, 84–89, 92, 92, 93]. For instance, Zheng et al. [94] report the emergence of organic workers who attempt to mimic the behavior of real users. Rahman et al. [67] provide insights from studied ASO workers, that confirm the existence of organic workers in the wild. In this paper we provide measurements from devices of ASO workers and regular Android users. Our data suggests that the use of apps installed for promotion differs from that of apps used for personal purposes. Further, even organic workers tend to use their devices in a manner that distinguishes them from regular users.

Related efforts also include extensive work to detect malware Android apps, e.g., [29, 41, 54, 60, 61, 69–71, 95]. Notably, Yang et al. [91] differentiate malware from benign apps based on the contexts that trigger security-sensitive behaviors. RacketStore detects ASO-promoted apps and devices of workers based on the context of the user interaction with them. While we seek to detect worker interactions with apps, we note that ASO work has been shown to be used to promote malware apps and improve their search rank, thus increase their consumer appeal [68].

Our study of the fraud market for Google services is related to other exploration of fraud markets [55, 74–76]. For instance, Dou et al. [37] developed a honeypot app and collect data to detect fraudulent bot-generated downloads. Mirian et al. [55] explore the market for Gmail account hijacking by creating synthetic but realistic victim personas and hiring services to hack into such accounts, while DeBlasio et al. [36] characterize the search engine fraud ecosystem using ground truth data internal to the Bing search engine. Stringhini et al. [74] studied Twitter follower markets by purchasing followers from different merchants and used such ground truth to discover patterns and detect market-controlled accounts in the wild. In this paper we leverage our finding of an abundant fraud market for Google services (i.e., review groups with tens of thousands of members) to recruit hundreds of worker-controlled devices, study their usage, and propose solutions to detect and distinguish them from devices used for personal purposes.

11 CONCLUSIONS

In this paper we have developed RacketStore, the first platform to collect detailed app and device usage information from the devices of app search optimization workers and regular users of Google Play services. We have presented empirical data from RacketStore installs on 803 devices and from interviews with some of their owners. We have developed a classifier to identify apps installed solely to be promoted and we have shown that on our data, it achieves an F1-measure that exceeds 99%. We have shown that features that model the user interaction with a device can be used to detect even organic devices with low levels of ASO work hidden among personal activities. Our techniques are resilient to worker strategy modifications, that would impose high overhead on the operation of their devices and the usage of the apps that they promote.

12 ACKNOWLEDGMENTS

This research was supported by NSF grants CNS-2013671 and CNS-2114911, and CRDF grant G-202105-67826. This publication is based on work supported by a grant from the U.S. Civilian Research & Development Foundation (CRDF Global). Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of CRDF Global.

REFERENCES

[1] [n. d.]. 15 U.S. Code § 45 - Unfair methods of competition unlawful; prevention by Commission. Legal Information Institute, https://www.law.cornell.edu/uscode/text/15/45.

[2] [n. d.]. Apple Media Services Terms and Conditions. https://www.apple.com/legal/internet-services/itunes/us/terms.html.

[3] [n. d.]. Boostyourapps: Buy App Reviews and Get Installs and Rating for Free. https://boostyourapps.org/.

[4] [n. d.]. Deploy machine learning models on mobile and IoT devices. https://www.tensorflow.org/lite.

[5] [n. d.]. Google Digital Wellbeing. https://wellbeing.google/tools/.

[6] [n. d.]. Google Play. https://play.google.com/store?hl=en.

[7] [n. d.]. MobiASO: Mobile App Marketing Agency. https://mobiaso.com/.

[8] [n. d.]. Number of Android apps on Google Play: 2,969,894. AppBrain, https://www.appbrain.com/stats/number-of-android-apps, month=.

[9] [n. d.]. Permissions Overview. https://bit.ly/2x4HKiW.

[10] [n. d.]. RacketStore Code. https://github.com/nestorghh/racketstore.

[11] [n. d.]. Ratings and Reviews on the Play Store. https://play.google.com/about/comment-posting-policy/.

[12] [n. d.]. Retention Rate Meaning. https://www.adjust.com/glossary/retention-rate/.

[13] [n. d.]. TapJoy: Mobile Advertising and App Monetization Platform. https://www.tapjoy.com/.

[14] [n. d.]. Untrue, misleading or unauthorized use of tests and testimonials. https://www.competitionbureau.gc.ca/eic/site/cb-bc.nsf/eng/00527.html.

[15] [n. d.]. VirusTotal. https://www.virustotal.com/gui/home.

[16] 2005. Directive 2005/29/EC of the European Parliament and of the Council. Official Journal of the European Union, https://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=OJ:L:2005:149:0022:0039:en:PDF.

[17] 2012. Cell phone culture: How cultural differences affect mobile use. CNN Business, https://www.cnn.com/2012/09/27/tech/mobile-culture-usage/.

[18] 2018. Freedom on the Net, Bangladesh. Freedom House, https://freedomhouse.org/report/freedom-net/2018/bangladesh.

[19] 2018. Journalists, activists in Bangladesh arrested under ICT Act for posting on social media. AccessNow, https://www.accessnow.org/bangladesh-ict-act/.

[20] 2018. No Place for Criticism. Bangladesh Crackdown on Social Media Commentary. https://www.hrw.org/report/2018/05/09/no-place-criticism/bangladesh-crackdown-social-media-commentary.

[21] 2020. AndroidID. , https://developer.android.com/reference/android/provider/Settings.Secure.html#ANDROID_ID.

[22] 2020. Mobile Application Market Size, and Trends Analysis Report By Store Type (Google Store, Apple Store), By Application (Gaming, Music and Entertainment, Health and Fitness), By Region, And Segment Forecasts, 2020 - 2027. https://www.grandviewresearch.com/industry-analysis/mobile-application-market.

[23] 2020. Privacy, Security, and Deception. Developer Policy Center, https://play.google.com/about/privacy-security-deception/user-data/.

[24] 2021. Mobile App Download and Usage Statistics (2021). BuildFire, https://buildfire.com/app-statistics/.

[25] 2021. What is App Store Optimization? Ultimate Guide to ASO in 2021. App Radar, https://appradar.com/academy/aso-basics/what-is-app-store-optimization-aso.

[26] Syed Ishtiaque Ahmed, Md. Romael Haque, Shion Guha, Md. Rashidujjaman Rifat, and Nicola Dell. 2017. Privacy, Security, and Surveillance in the Global South: A Study of Biometric Mobile SIM Registration in Bangladesh. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems (CHI '17)*. 906–918.

[27] Prudhvi Ratna Badri Satya, Kyumin Lee, Dongwon Lee, Thanh Tran, and Jason (Jiasheng) Zhang. 2016. Uncovering Fake Likers in Online Social Networks. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management (CIKM '16)*. Association for Computing Machinery, New York, NY, USA, 2365–2370. https://doi.org/10.1145/2983323.2983695

[28] Alex Beutel, Wanhong Xu, Venkatesan Guruswami, Christopher Palow, and Christos Faloutsos. 2013. CopyCatch: Stopping Group Attacks by Spotting Lockstep Behavior in Social Networks. In *Proceedings of the 22nd International Conference on World Wide Web (WWW '13)*. Association for Computing Machinery, New York, NY, USA, 119–130. https://doi.org/10.1145/2488388.2488400

[29] A. Bianchi, J. Corbetta, L. Invernizzi, Y. Fratantonio, C. Kruegel, and G. Vigna. 2015. What the App is That? Deception and Countermeasures in the Android User Interface. In *2015 IEEE Symposium on Security and Privacy*. 931–948. https://doi.org/10.1109/SP.2015.62

[30] Dearbhail Bracken-Roche, Emily Bell, Mary Ellen Macdonald, and Eric Racine. 2017. The concept of 'vulnerability'in research ethics: an in-depth analysis of policies and guidelines. *Health research policy and systems* 15, 1 (2017), 8.

[31] Leo Breiman. 2001. Random Forests. *Mach. Learn.* 45, 1 (2001), 5–32. https://doi.org/10.1023/A:1010933404324

[32] Qiang Cao, Xiaowei Yang, Jieqi Yu, and Christopher Palow. 2014. Uncovering Large Groups of Active Malicious Accounts in Online Social Networks. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security (CCS '14)*. Association for Computing Machinery, New York, NY, USA, 477–488. https://doi.org/10.1145/2660267.2660269

[33] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer. 2002. SMOTE: Synthetic Minority Over-sampling Technique. *Journal of Artificial Intelligence Research* 16 (2002).

[34] Erika Chin, Adrienne Porter Felt, Vyas Sekar, and David Wagner. 2012. Measuring User Confidence in Smartphone Security and Privacy. In *Proceedings of the Eighth Symposium on Usable Privacy and Security (SOUPS '12)*. Association for Computing Machinery, New York, NY, USA, Article 1, 16 pages. https://doi.org/10.1145/2335356.2335358

[35] Emiliano De Cristofaro, Arik Friedman, Guillaume Jourjon, Mohamed Ali Kaafar, and M. Zubair Shafiq. 2014. Paying for Likes? Understanding Facebook Like Fraud Using Honeypots. In *Proceedings of the 2014 Conference on Internet Measurement Conference (IMC '14)*. Association for Computing Machinery, New York, NY, USA, 129–136. https://doi.org/10.1145/2663716.2663729

[36] Joe DeBlasio, Saikat Guha, Geoffrey M. Voelker, and Alex C. Snoeren. 2017. Exploring the Dynamics of Search Advertiser Fraud. In *Proceedings of the 2017 Internet Measurement Conference (IMC '17)*. Association for Computing Machinery, New York, NY, USA, 157–170. https://doi.org/10.1145/3131365.3131393

[37] Yingtong Dou, Weijian Li, Zhirong Liu, Zhenhua Dong, Jiebo Luo, and Philip S. Yu. 2019. Uncovering Download Fraud Activities in Mobile App Markets. In *Proceedings of the 2019 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM '19)*. Association for Computing Machinery, New York, NY, USA, 671–678. https://doi.org/10.1145/3341161.3345306

[38] Shehroze Farooqi, Álvaro Feal, Tobias Lauinger, Damon McCoy, Zubair Shafiq, and Narseo Vallina-Rodriguez. 2020. Understanding Incentivized Mobile App Installs on Google Play Store. In *Proceedings of the ACM Internet Measurement Conference (IMC '20)*. Association for Computing Machinery, New York, NY, USA, 696–709. https://doi.org/10.1145/3419394.3423662

[39] Amir Fayazi, Kyumin Lee, James Caverlee, and Anna Squicciarini. 2015. Uncovering Crowdsourced Manipulation of Online Reviews. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '15)*. Association for Computing Machinery, New York, NY, USA, 233–242. https://doi.org/10.1145/2766462.2767742

[40] G Fei, A Mukherjee, B Liu, M Hsu, M Castellanos, and R Ghosh. 2013. Exploiting burstiness in reviews for review spammer detection. In *Proceedings of the 7th International Conference on Weblogs and Social Media, ICWSM 2013*. 175–184.

[41] Michael Grace, Yajin Zhou, Qiang Zhang, Shihong Zou, and Xuxian Jiang. 2012. Riskranker: scalable and accurate zero-day android malware detection. In *Proceedings of the 10th international conference on Mobile systems, applications, and services*. 281–294.

[42] Stephan Günnemann, Nikou Günnemann, and Christos Faloutsos. 2014. Detecting Anomalies in Dynamic Rating Data: A Robust Probabilistic Model for Rating Evolution. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '14)*. Association for Computing Machinery, New York, NY, USA, 841–850. https://doi.org/10.1145/2623330.2623721

[43] Nestor Hernandez, Mizanur Rahman, Ruben Recabarren, and Bogdan Carbunar. 2018. Fraud De-Anonymization for Fun and Profit. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. 115–130.

[44] Atefeh Heydari, Mohammadali Tavakoli, and Naomie Salim. 2016. Detection of Fake Opinions Using Time Series. *Expert Syst. Appl.* 58, C (Oct. 2016), 83–92. https://doi.org/10.1016/j.eswa.2016.03.020

[45] Bryan Hooi, Neil Shah, Alex Beutel, Stephan Günnemann, Leman Akoglu, Mohit Kumar, Disha Makhija, and Christos Faloutsos. 2015. BIRDNEST: Bayesian Inference for Ratings-Fraud Detection. *CoRR* abs/1511.06030. arXiv:1511.06030 http://arxiv.org/abs/1511.06030

[46] Yangyu Hu, Haoyu Wang, Ren He, Li Li, Gareth Tyson, Ignacio Castro, Yao Guo, Lei Wu, and Guoai Xu. 2020. Mobile App Squatting. In *Proceedings of The Web Conference 2020 (WWW '20)*. Association for Computing Machinery, New York, NY, USA, 1727–1738. https://doi.org/10.1145/3366423.3380243

[47] Lilly Irani, Janet Vertesi, Paul Dourish, Kavita Philip, and Rebecca E. Grinter. 2010. Postcolonial Computing: A Lens on Design and Development. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '10)*. 1311–1320.

[48] Meng Jiang, Peng Cui, Alex Beutel, Christos Faloutsos, and Shiqiang Yang. 2014. Inferring Strange Behavior from Connectivity Pattern in Social Networks. In *Advances in Knowledge Discovery and Data Mining*, Vincent S. Tseng, Tu Bao Ho, Zhi-Hua Zhou, Arbee L. P. Chen, and Hung-Yu Kao (Eds.).

[49] Dave Johnson. 2020. How to check your app usage stats an Android device to figure out which apps you spend the most battery, data, and time on. Business Insider, https://www.businessinsider.com/how-to-check-app-usage-on-android.

[50] Parisa Kaghazgaran, James Caverlee, and Anna Squicciarini. 2018. Combating Crowdsourced Review Manipulators: A Neighborhood-Based Approach. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining (WSDM '18)*. Association for Computing Machinery, New York, NY, USA, 306–314. https://doi.org/10.1145/3159652.3159726

[51] Huayi Li, Geli Fei, Shuai Wang, Bing Liu, Weixiang Shao, Arjun Mukherjee, and Jidong Shao. 2017. Bimodal Distribution and Co-Bursting in Review Spam Detection. In *Proceedings of the 26th International Conference on World Wide Web (WWW '17)*. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE, 1063–1072. https://doi.org/10.1145/3038912.3052582

[52] Shanshan Li, James Caverlee, Wei Niu, and Parisa Kaghazgaran. 2017. Crowdsourced App Review Manipulation. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '17)*. Association for Computing Machinery, New York, NY, USA, 1137–1140. https://doi.org/10.1145/3077136.3080741

[53] Ee-Peng Lim, Viet-An Nguyen, Nitin Jindal, Bing Liu, and Hady Wirawan Lauw. 2010. Detecting Product Review Spammers Using Rating Behaviors. In *Proceedings of the 19th ACM International Conference on Information and Knowledge Management (CIKM '10)*. Association for Computing Machinery, New York, NY, USA, 939–948. https://doi.org/10.1145/1871437.1871557

[54] Aravind Machiry, Nilo Redini, Eric Gustafson, Yanick Fratantonio, Yung Ryn Choe, Christopher Kruegel, and Giovanni Vigna. 2018. Using Loops For Malware Classification Resilient to Feature-Unaware Perturbations. In *Proceedings of the 34th Annual Computer Security Applications Conference (ACSAC '18)*. Association for Computing Machinery, New York, NY, USA, 112–123. https://doi.org/10.1145/3274694.3274731

[55] Ariana Mirian, Joe DeBlasio, Stefan Savage, Geoffrey M. Voelker, and Kurt Thomas. 2019. Hack for Hire: Exploring the Emerging Market for Account Hijacking. In *The World Wide Web Conference (WWW '19)*. Association for Computing Machinery, New York, NY, USA, 1279–1289. https://doi.org/10.1145/3308558.3313489

[56] James A Muir and PC Van Oorschot. 2006. Internet geolocation and evasion. *Ottawa: School of Computer Science, Carleton University* (2006).

[57] Arjun Mukherjee, Abhinav Kumar, Bing Liu, Junhui Wang, Meichun Hsu, Malu Castellanos, and Riddhiman Ghosh. 2013. Spotting Opinion Spammers Using Behavioral Footprints. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '13)*. Association for Computing Machinery, New York, NY, USA, 632–640. https://doi.org/10.1145/2487575.2487580

[58] Arjun Mukherjee, Vivek Venkataraman, Bing Liu, and Natalie Glance. 2013. What Yelp Fake Review Filter Might Be Doing? https://www.aaai.org/ocs/index.php/ICWSM/ICWSM13/paper/view/6006

[59] Shirin Nilizadeh, Hojjat Aghakhani, Eric Gustafson, Christopher Kruegel, and Giovanni Vigna. 2019. Think Outside the Dataset: Finding Fraudulent Reviews Using Cross-Dataset Analysis. In *The World Wide Web Conference (WWW '19)*. Association for Computing Machinery, New York, NY, USA, 3108–3115. https://doi.org/10.1145/3308558.3313647

[60] Dario Nisi, Antonio Bianchi, and Yanick Fratantonio. 2019. Exploring Syscall-Based Semantics Reconstruction of Android Applications. In *22nd International Symposium on Research in Attacks, Intrusions and Defenses (RAID 2019)*. USENIX Association, Chaoyang District, Beijing, 517–531. https://www.usenix.org/conference/raid2019/presentation/nisi

[61] Lucky Onwuzurike, Enrico Mariconti, Panagiotis Andriotis, Emiliano De Cristofaro, Gordon Ross, and Gianluca Stringhini. 2019. MaMaDroid: Detecting Android Malware by Building Markov Chains of Behavioral Models (Extended Version). *ACM Trans. Priv. Secur.* 22, 2, Article 14 (April 2019), 34 pages. https://doi.org/10.1145/3313391

[62] Joyojeet Pal, Meera Lakshmanan, and Kentaro Toyama. 2009. "My child will be respected": Parental perspectives on computers and education in Rural India. *Information Systems Frontiers* 11, 2 (2009), 129–144.

[63] TE Parliament. 2016. Regulation (eu) 2016/679 of the european parliament and of the council. *Official Journal of the European Union* (2016).

[64] Feargus Pendlebury, Fabio Pierazzi, Roberto Jordaney, Johannes Kinder, and Lorenzo Cavallaro. 2019. TESSERACT: Eliminating Experimental Bias in Malware Classification across Space and Time. In *28th USENIX Security Symposium (USENIX Security 19)*. USENIX Association, Santa Clara, CA, 729–746. https://www.usenix.org/conference/usenixsecurity19/presentation/pendlebury

[65] Mizanur Rahman, Nestor Hernandez, Bogdan Carbunar, and Duen Horng Chau. 2018. Search Rank Fraud De-Anonymization in Online Systems. In *Proceedings of ACM Conference on Hypertext and Social Media*.

[66] Mizanur Rahman, Nestor Hernandez, Bogdan Carbunar, and Duen Horng Chau. 2020. Towards De-Anonymization of Google Play Search Rank Fraud. *IEEE Transactions on Knowledge and Data Engineering* (2020), 1–1. https://doi.org/10.1109/TKDE.2020.2975170

[67] Mizanur Rahman, Nestor Hernandez, Ruben Recabarren, Syed Ishtiaque Ahmed, and Bogdan Carbunar. 2019. The Art and Craft of Fraudulent App Promotion in Google Play. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*. 2437–2454.

[68] Mahmudur Rahman, Mizanur Rahman, Bogdan Carbunar, and Polo Chau. 2017. Search Rank Fraud and Malware Detection in Google Play. *IEEE Transactions on Knowledge and Data Engineering* 29, 6 (2017).

[69] Siegfried Rasthofer, Steven Arzt, Marc Miltenberger, and Eric Bodden. 2016. Harvesting Runtime Values in Android Applications That Feature Anti-Analysis Techniques. In *23rd Annual Network and Distributed System Security Symposium, NDSS 2016, San Diego, California, USA, February 21-24, 2016*. The Internet Society. https://bit.ly/38ZZ4qe

[70] Joel Reardon, Álvaro Feal, Primal Wijesekera, Amit Elazari Bar On, Narseo Vallina-Rodriguez, and Serge Egelman. 2019. 50 Ways to Leak Your Data: An Exploration of Apps' Circumvention of the Android Permissions System. In *28th USENIX Security Symposium (USENIX Security 19)*. USENIX Association, Santa Clara, CA, 603–620. https://www.usenix.org/conference/usenixsecurity19/presentation/reardon

[71] Laura Shipp and Jorge Blasco Alis. 2020. How private is your period?: A systematic analysis of menstrual app privacy policies. In *Proceedings on Privacy Enhancing Technologies 2020 (PoPETs 2020)*, Vol. 2020(4). 491–510. https://doi.org/10.2478/popets-2020-0083

[72] Jonghyuk Song, Sangho Lee, and Jong Kim. 2015. CrowdTarget: Target-Based Detection of Crowdturfing in Online Social Networks. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security (CCS '15)*. Association for Computing Machinery, New York, NY, USA, 793–804. https://doi.org/10.1145/2810103.2813661

[73] Gianluca Stringhini, Pierre Mourlanne, Gregoire Jacob, Manuel Egele, Christopher Kruegel, and Giovanni Vigna. 2015. EVILCOHORT: Detecting Communities of Malicious Accounts on Online Services. In *24th USENIX Security Symposium (USENIX Security 15)*. USENIX Association, Washington, D.C., 563–578. https://www.usenix.org/conference/usenixsecurity15/technical-sessions/presentation/stringhini

[74] Gianluca Stringhini, Gang Wang, Manuel Egele, Christopher Kruegel, Giovanni Vigna, Haitao Zheng, and Ben Y. Zhao. 2013. Follow the Green: Growth and Dynamics in Twitter Follower Markets. In *Proceedings of the 2013 Conference on Internet Measurement Conference (IMC '13)*. Association for Computing Machinery, New York, NY, USA, 163–176. https://doi.org/10.1145/2504730.2504731

[75] Kurt Thomas, Chris Grier, Dawn Song, and Vern Paxson. 2011. Suspended Accounts in Retrospect: An Analysis of Twitter Spam. In *Proceedings of the 2011 ACM SIGCOMM Conference on Internet Measurement Conference (IMC '11)*. Association for Computing Machinery, New York, NY, USA, 243–258. https://doi.org/10.1145/2068816.2068840

[76] Kurt Thomas, Damon McCoy, Chris Grier, Alek Kolcz, and Vern Paxson. 2013. Trafficking Fraudulent Accounts: The Role of the Underground Market in Twitter Spam and Abuse. In *Proceedings of the 22nd USENIX Conference on Security (SEC'13)*. USENIX Association, USA, 195–210.

[77] Ben Thorn. 2019. Fake Reviews and the Australian Consumer Law. https://xuveo.legal/blog/fake-reviews-australian-consumer-law.

[78] Tian Tian, Jun Zhu, Fen Xia, Xin Zhuang, and Tong Zhang. 2015. *Crowd Fraud Detection in Internet Advertising*. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE, 1100–1110. https://doi.org/10.1145/2736277.2741136

[79] Virus Total. 2012. Virustotal-free online virus, malware and url scanner. *Online: https://www. virustotal. com/en* (2012).

[80] Haoyu Wang, Zhe Liu, Jingyue Liang, Narseo Vallina-Rodriguez, Yao Guo, Li Li, Juan Tapiador, Jingcun Cao, and Guoai Xu. 2018. Beyond Google Play: A Large-Scale Comparative Study of Chinese Android App Markets. In *Proceedings of the Internet Measurement Conference 2018 (IMC '18)*. Association for Computing Machinery, New York, NY, USA, 293–307. https://doi.org/10.1145/3278532.3278558

[81] Ji Wang, Jianguo Zhang, Weidong Bao, Xiaomin Zhu, Bokai Cao, and Philip S. Yu. 2018. Not Just Privacy: Improving Performance of Private Deep Learning in Mobile Cloud. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '18)*. Association for Computing Machinery, New York, NY, USA, 2407–2416. https://doi.org/10.1145/3219819.3220106

[82] C. Wei and B. E. Kolko. 2005. Studying mobile phone use in context: cultural, political, and economic dimensions of mobile phone use. In *IPCC 2005*.

[83] Madeleine Wilson. 2019. How to Improve App Ratings and Reviews. Apptentive, https://www.apptentive.com/blog/2019/08/22/improve-mobile-app-ratings-and-reviews/.

[84] Sihong Xie, Guan Wang, Shuyang Lin, and Philip S. Yu. 2012. Review Spam Detection via Temporal Pattern Discovery. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '12)*. Association for Computing Machinery, New York, NY, USA, 823–831. https://doi.org/10.1145/2339530.2339662

[85] Zhen Xie and Sencun Zhu. 2014. GroupTie: Toward Hidden Collusion Group Discovery in App Stores. In *Proceedings of the 2014 ACM Conference on Security and Privacy in Wireless and Mobile Networks (WiSec '14)*. Association for Computing Machinery, New York, NY, USA, 153–164. https://doi.org/10.1145/2627393.2627409

[86] Zhen Xie and Sencun Zhu. 2015. AppWatcher: Unveiling the Underground Market of Trading Mobile App Reviews. In *Proceedings of the 8th ACM Conference on Security and Privacy in Wireless and Mobile Networks (WiSec '15)*. Association for Computing Machinery, New York, NY, USA, Article 10, 11 pages. https://doi.org/10.1145/2766498.2766510

[87] Zhen Xie, Sencun Zhu, Qing Li, and Wenjing Wang. 2016. You Can Promote, but You Can't Hide: Large-Scale Abused App Detection in Mobile App Stores. In *Proceedings of the 32nd Annual Conference on Computer Security Applications (ACSAC '16)*. Association for Computing Machinery, New York, NY, USA, 374–385. https://doi.org/10.1145/2991079.2991099

[88] Chang Xu. 2013. Detecting Collusive Spammers in Online Review Communities *(PIKM '13)*. Association for Computing Machinery, New York, NY, USA, 33–40. https://doi.org/10.1145/2513166.2513176

[89] Chang Xu and Jie Zhang. [n. d.]. Combating Product Review Spam Campaigns via Multiple Heterogeneous Pairwise Features. In *Proceedings of the 2015 SIAM International Conference on Data Mining*. 172–180. https://doi.org/10.1137/1.9781611974010.20

[90] Xu, Teng and Goossen, Gerard and Cevahir, Huseyin Kerem and Khodeir, Sara and Jin, Yingyezhe and Li, Frank and Shan, Shawn and Patel, Sagar and Freeman, David and Pearce, Paul. 2021. Deep Entity Classification: Abusive Account Detection for Online Social Networks. In *Usenix Security Symposium*.

[91] Wei Yang, Xusheng Xiao, Benjamin Andow, Sihan Li, Tao Xie, and William Enck. 2015. Appcontext: Differentiating malicious and benign mobile app behaviors using context. In *37th IEEE International Conference on Software Engineering*, Vol. 1. 303–313.

[92] Junting Ye, Santhosh Kumar, and Leman Akoglu. 2016. Temporal Opinion Spam Detection by Multivariate Indicative Signals. In *Proceedings of the Tenth International Conference on Web and Social Media, Cologne, Germany, May 17-20, 2016*. 743–746.

[93] Dong Yuan, Yuanli Miao, Neil Zhenqiang Gong, Zheng Yang, Qi Li, Dawn Song, Qian Wang, and Xiao Liang. 2019. Detecting Fake Accounts in Online Social Networks at the Time of Registrations. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security (CCS '19)*. Association for Computing Machinery, New York, NY, USA, 1423–1438. https://doi.org/10.1145/3319535.3363198

[94] Haizhong Zheng, Minhui Xue, Hao Lu, Shuang Hao, Haojin Zhu, Xiaohui Liang, and Keith W. Ross. 2018. Smoke Screener or Straight Shooter: Detecting Elite Sybil Attacks in User-Review Social Networks. In *25th Annual Network and Distributed System Security Symposium, NDSS 2018, San Diego, California, USA, February 18-21, 2018*.

[95] Yajin Zhou, Zhi Wang, Wu Zhou, and Xuxian Jiang. 2012. Hey, you, get off of my market: detecting malicious apps in official and alternative android markets. In *Network and Distributed System Security Symposium, NDSS*, Vol. 25. 50–52.

A SNAPSHOT FINGERPRINTING

To properly analyze the data collected from participating devices, we needed to map each collected device snapshot to a single device. As mentioned in § 3, the first snapshot from a device includes (1) the 10-digit *install ID* computed by RacketStore upon installation, (2) the 6-digit *participant ID*, uniquely generated by us and assigned to each participant, and (3) the Android ID. We expected that the combination of the participant ID, install ID and Android ID will be enough to provide this mapping.

However, we found that the same device can be responsible for multiple install events of RacketStore, i.e., where a different combination of install ID, participant ID and Android ID is reported in different snapshots from the same device. For instance, we encountered cases of different ASO workers, with different assigned participant ids, who shared some devices. This can occur for instance if the workers are employed by the same organization, thus have access to a common set of devices or (2) the same workers pretending to be a different worker. Such workers can install RacketStore at different times believing this to be a repeat campaign. We also observed workers who repeatedly install and uninstall RacketStore, in order to get paid multiple times for the installation. Further, for some installs, due to suspected incompatibilities (there are over 24,000 types of device models), the collected snapshots did not include the Android ID and device information. We note that we did not collect device IMEI since it requires an additional dangerous permission which we wanted to avoid.

Figure 16: Ad shown to audience on Instagram Feed, Explore, and Stories. Upon clicking, users are sent to the website where the study is explained, see Figure 17.

To address this problem, we used the following process to fingerprint snapshots. Specifically, we first grouped all the collected device snapshots into *n candidate devices*, based on their install ID. We then compared the $\binom{n}{2}$ pairs of candidates to identify and coalesce candidate devices with different install IDs that are actually the same device. First, for each install ID x, we compute the RacketStore *install interval* $[T_f, t_l]$ where t_f and t_l are the first and last timestamp recorded in our database from snapshots that belong to x. We then declare as different devices, install pairs (x, y) that have overlapping installation intervals. We then coalesced candidate device pairs that do not overlap on installation intervals based on their Android ID (if present): if the pairs have the same Android ID the two installs belong to the same device, otherwise they are different devices. To validate this approach, we have computed the Jaccard similarity between candidate device pairs, i.e., (1) their sets of tuples (a, t) where a is an app and t is the install time registered by the Android API for app a, and (2) their sets of registered accounts. Candidate device pairs with different Android IDs had a Jaccard similarity for installed apps of at most 0.5625. Candidate device pairs with Jaccard similarity above 0.53 for registered accounts, had low similarity for installed apps.

B RECRUITMENT MATERIAL

Figure 16 shows a snapshot of the ad we have shown to audience on Instagram Feed, Explore, and Stories to recruit regular users. Upon clicking, users are sent to a website, shown in Figure 17(a) where we explain the study. Figure 17(b) shows the registration page, displayed to the user only after the user has clicked Sign Me Up and read the content on a web-page that shows the following recruiting message. This message is also shown to recruited ASO workers on a one-on-one basis over messenger apps.

"We are researchers from a US university, looking for people who write paid reviews in Google Play, and are willing to participate in a user study. We are conducting this study as part of an effort to increase our understanding of how app search optimization workers

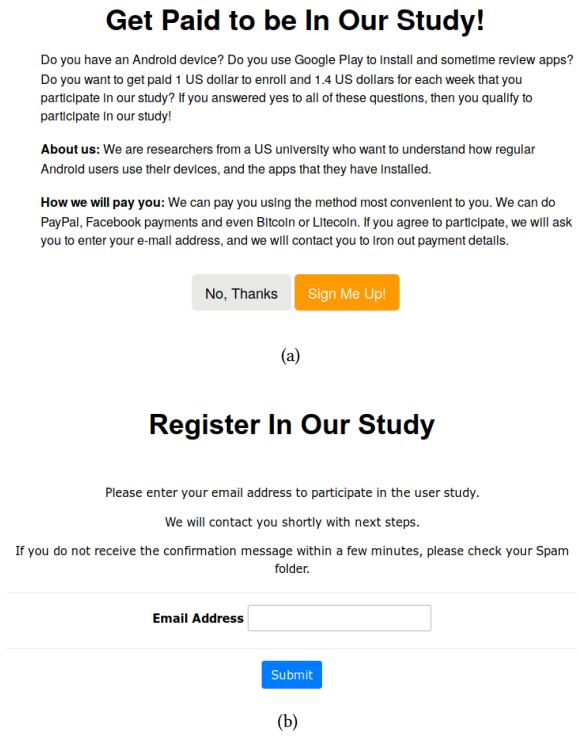


Figure 17: Screenshots of the user study web page. (a) User study landing page. (b) Registration page shown only after user has consented to participate. We ask users to enter their email addresses to contact them with next steps.

interact with Google Play apps.

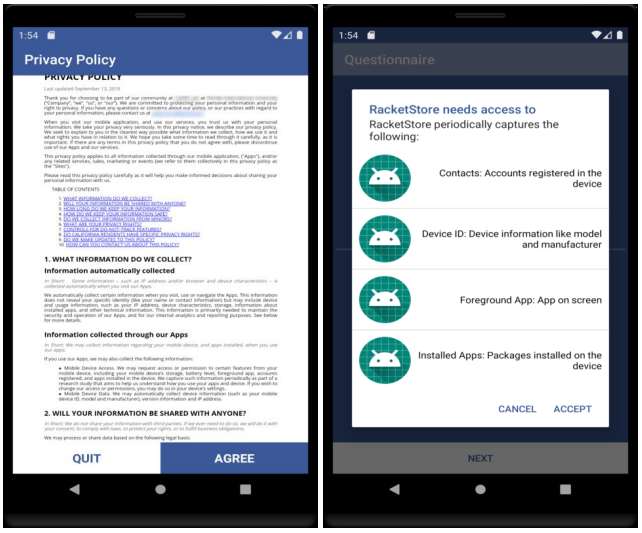
If you agree to participate in the study, we will ask you to install an app from Google Play and keep it installed for at least two days. We will pay you $1 when you install the app. We will then pay you 20 cents for each day when you keep the app installed, on a weekly basis. That is, we will pay you $1.40 per week, for just keeping the app installed. We may also ask you to use the app to write reviews. If this happens, we will pay you additional money, at a rate that we will negotiate.

Please note that we will guard the information that you provide and that we collect, with the utmost secrecy. We will never reveal to anyone any information that may be linked to you, including the fact that you participated in our study

Your participation is completely voluntary and you may choose to withdraw at any time. If you agree to participate, please reply to this message. Also, please send us answers to the following questions:

1. Have you ever written paid reviews in Google Play?
2. How many user accounts do you control in Google Play?
3. How many mobile devices do you own or can access?
4. On how many devices can you install our app?
5. For how many days can you keep our app installed?
6. Are you an administrator or do you post reviews yourself?
7. How many ASO jobs are you currently working on?"

Figure 18 shows screenshots of information shown to participants by the RacketStore app upon installation. They include the privacy policy that discloses how RacketStore accesses, collects, and uses user data (Figure 18(a)) and the in-app disclosure that summarizes the data collected (Figure 18(b)).



(a) (b)

Figure 18: RacketStore mobile app: screenshots of the first two screens shown upon installation. (a) A privacy policy that comprehensively discloses how RacketStore accesses, collects, and uses user data. (b) In-app disclosure to summarize data being collected. RacketStore does not access or collect any personal or sensitive data until the user consents.

C CONSENT FORM EXCERPTS

The consent form that we presented to each participant includes the following information:

Procedures. If you agree to be in the study, we will ask you to do the following things:

1. You will need to click on "I Approve" checkbox at the end of this form. 2. We will send you a link for the RacketStore app in Google Play and you will install it. 3. You will need to keep the app installed as long as you desire to participate in the study, but preferably for at least two days.

Please be assured that your participation in this study is confidential. We will not make your answers public, and we will store them securely (under password protection) in the computers of the researchers.

Risks And/Or Discomforts. Risks of participating in this study do not exceed those that you would encounter in a regular interaction with other prospective employers. Findings from this study may be used by providers like Google to attempt to detect and eliminate fraud. However, we will never share your data with anyone, for any reason. In fact, we will only store and publish aggregate information from multiple participants that we recruit. Aggregates will not be useful to someone to, e.g., de-anonymize you or any of the participants.

Use of Your Information. We automatically collect certain information that includes device and app usage information, your IP address, device characteristics, storage, information about installed

and used apps, and other technical information. We also collect public information hosted on the Google Play Store for the reviews that you posted. We collect such information periodically as part of a research study that aims to help us understand how you use your apps and device in order to detect review manipulation or fraud. If you wish to change our access or permissions, you may do so in your device's settings.

Mobile Device Access. We request access or permission to certain features from your mobile device, including your mobile device's storage, battery level, foreground app, accounts registered, and apps installed in the device.

Mobile Device Data. We automatically collect device information (such as your mobile device ID, model and manufacturer) and version information.

Confidentiality. The records of this study will be kept private and will be protected to the fullest extent provided by law. In any sort of report we might publish, we will not include any information that will make it possible to identify a subject. Research records will be stored securely and only the researcher team will have access to the records. However, your records may be reviewed for audit purposes by authorized University or other agents who will be bound by the same provisions of confidentiality. Those agents will be unable to recover your identity as we do not collect any personally identifiable information.

Compensation. You will receive a payment of $1 for installing the app. You will then receive 20 cents for each day when you keep the app installed, on a weekly basis. That is, you will be paid $1.40 per week, for just keeping the app installed.

Right to Decline or Withdraw. Your participation in this study is voluntary. You are free to participate in the study or withdraw your consent at any time during the study. Your withdrawal or lack of participation will not affect any benefits to which you are otherwise entitled.

D ADDITIONAL ETHICAL CONSIDERATIONS

Privacy Policy. We included RacketStore's *privacy policy* both in its Google Play profile page, and inside the app, right after the main layout that summarizes the study. The privacy policy explains the data collection process in terms of the information being collected, including that we collect the data periodically and that we collect the reviews posted from the accounts registered on the device (see Figure 18(a) in Appendix C). The in-app disclosure (Figure 18(b)) further summarizes the privacy policy. In addition to the consent form, participants also had to give explicit consent to the in-app disclosure. The RacketStore app only collected data if the user provided explicit consent. Therefore, all consenting participants were aware of all the data that we collected.

Requested Permissions. RacketStore explicitly requests two Android permissions, `PACKAGE_USAGE_STATS` and `GET_ACCOUNTS`, which participant need to explicitly grant. If any permission is not granted, RacketStore does not collect the corresponding information.

Data Protection. We used GDPR [63] recommended pseudonymisation for data processing and statistics, and other generally accepted good practices for privacy preservation. At the completion of the study, we deleted all PII. We also only generated aggregated statistics. No PII of our participants was disclosed outside of the research team.

PII	Collector	Reasons	Deletion
Accounts	RacketStore	Classification	After use
Accounts	RacketStore	Review collection	After use
Email	Website	Recruitment	After use
IP address	Backend	Statistics	Not stored
Device ID	RacketStore	Snap. fingerprint	After use
Payment Info	Author	Payment	Not stored

Table 3: Personally Identifiable information (PII) that we collected, how we collected it, the reasons, and time of deletion.

Collected PII. Table 3 summarizes the PII we collected, how we collected it, the reasons, and how long it was stored. It shows that the PII we collected consists of user accounts registered on participant devices, e-mail address, the device IP address and IDs. The RacketStore app collected the accounts and device IDs; the backend server collected the device IP address and the recruitment website collected the participant e-mail address.

We needed registered accounts for the device classification task. We needed the participant e-mail account in order to contact them with details of the main study and for the follow-up study. We used device IP addresses to help us recruit regular users from the same regions with the ASO worker participants. Further, we needed the device IDs to help us fingerprint snapshots, i.e., group collected snapshots by the device that reported them.

We have deleted all accounts, device IDs and IP addresses after use. We have deleted participant e-mail addresses after the study.

Risk To Benefits Assessment. The consent form informed participants on the risks associated with the research. We presented risks in relations to the participant regular interactions. We did not store PII after the completion of the study, and only published aggregate data. We believe that participation risks are reasonable in relation to benefits. We informed participants that benefits for participation include helping us understand and model app search optimization, and also raising their awareness to security and privacy risks that stem from installing malware, granting requested permissions, and account/password management strategies.

ASO Legality and Stigma. There are no direct local legal policies to criminalize black hat app search optimization in many countries of the Global South. For example, in Bangladesh, there is not direct law to prevent such activities. The law closest to this issue is a recently passed ICT Act that prohibits the dissemination of incorrect information over the Internet [19]. However, this law has mostly been applied to control the dissemination of politically motivated, unfounded information over social media (see [18, 20, 26], for example). However, ASO work has never been addressed by law enforcing agencies. A similar situation is also present in many other countries in the Global South including India and Pakistan. Hence, the job of our participants was not illegal or unethical according to their own law of the land.

We also asked several participants questions on the legal aspects of their work, *Do you need to be careful about anything? What are your common fear or risks?*. Participants claimed that they are not afraid, for instance, one worker said that "*What we're doing is a legal and right job. So no need to be afraid*".

Moreover, ASO work is not stigmatized in most countries in the Global South. HCI scholars in post-colonial computing argue that many ideas that western scholars hold around how computers are used in non-western contexts are often biased by their own experiences in the West [47]. We argue that the stigma around ASO is a similar case: While in many parts of the West, ASO work might occur as a crime, a job that needs to be hidden, that is not true in countries like Bangladesh, India, Pakistan, or Vietnam. Most local citizens do not understand the technical details of ASO, making it hard for them to judge the work, while in fact, any work with computers and the Internet is considered prestigious in many communities [62]. These arguments further suggest that the use of deception is not required in studies such as the ones we conducted in this work, when recruiting participants from countries in the Global South.

Compensation and Professional Security. The compensation of the participants was determined according to the fair market rate. We made sure that the rate is not so low that the participants were exploited, and also not so high that they were coerced. There might also be a larger concern about the overall impact of our research on ASO work, in general, that might impact their profession. Previous work explains why studying the ASO workers does not impact their livelihood [67]. Nonetheless, we disclosed and explained this possibility to our participants, and we did not hear any concern from any of them.